Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Modelagem Matemática e Computacional

UMA COMPARAÇÃO DAS DISTRIBUIÇÕES INFLACIONADAS PARA MODELAR DADOS DUPLAMENTE LIMITADOS

ISAAC FERREIRA DE LIMA

Dissertação de Mestrado

João Pessoa

Maio de 2024

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Modelagem Matemática e Computacional

ISAAC FERREIRA DE LIMA

UMA COMPARAÇÃO DAS DISTRIBUIÇÕES INFLACIONADAS PARA MODELAR DADOS DUPLAMENTE LIMITADOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional, UFPB, da Universidade Federal da Paraíba, como parte dos requisitos necessários à obtenção do título de Mestre em Modelagem Matemática e Computacional.

Orientadoras: Tatiene Correia de Souza

Tarciana Liberal Pereira de Araújo

João Pessoa

Maio de 2024

Catalogação na publicação
Seção de Catalogação e Classificação

L732c Lima, Isaac Ferreira de.
Uma comparação das distribuições inflacionadas para modelar dados duplamente limitados / Isaac Ferreira de Lima. - João Pessoa, 2024.
336 f. : il.

Orientação: Tatine Correia de Souza.
Coorientação: Tarciana Liberal Pereira de Araújo.
Dissertação (Mestrado) - UFPB/CI.

1. Matemática computacional. 2. Distribuições inflacionadas. 3. Distribuição beta. 4. Distribuição kumaraswamy. 5. Distribuição gama unitária. I. Souza, Tatine Correia de. II. Araújo, Tarciana Liberal Pereira de. III. Título.

UFPB/BC CDU 519.6(043)

Elaborado por Larissa Silva Oliveira de Mesquita - CRB-15/746

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **ISAAC FERREIRA DE LIMA**, candidato ao título de Mestre em Modelagem Matemática e Computacional, realizada no dia 31 de maio de 2024.

Aos trinta e um dias do mês de maio do ano de dois mil e vinte e quatro, às 09h30, via videoconferência, reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final do discente ISAAC FERREIRA DE LIMA, vinculado a Universidade Federal da Paraíba sob a matrícula nº 20221002621, candidato ao grau de Mestre em "*Modelagem Matemática e Computacional*", na linha de pesquisa *"Modelagem Probabilística"*, do Programa de Pós-Graduação em Modelagem Matemática e Computacional. A comissão examinadora foi composta pelos professores Tatiene Correia de Souza, Orientadora e Presidente da Banca; Tarciana Liberal Pereira de Araújo, Coorientadora; Sérgio de Carvalho Bezerra, Examinador Interno ao Programa; e Ana Hermínia Andrade e Silva, Examinadora Externa ao Programa. Dando início aos trabalhos, a Presidente da Banca cumprimentou os presentes, comunicou a finalidade da reunião e passou a palavra ao candidato para que fizesse, oralmente, a exposição do trabalho de dissertação intitulado *"UMA COMPARAÇÃO DAS DISTRIBUIÇÕES INFLACIONADAS PARA MODELAR DADOS DUPLAMENTE LIMITADOS"*. Concluída a exposição, o candidato foi arguido pela Banca Examinadora, que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Gean Paulo P. M. de Barros, secretário do Programa de Pós-Graduação em Modelagem Matemática e Computacional (PPGMMC), lavrei a presente ata, que vai assinada por mim e pelos membros da Banca Examinadora.

João Pessoa, 31 de maio de 2024.

Gean Paulo Pereira Maurício de Barros
Secretário do PPGMMC
SIAPE 2326476

Profª. Drª. Tatiene Correia de Souza
Orientadora (PPGMMC)

Documento assinado digitalmente
gov.br
TATIENE CORREIA DE SOUZA
Data: 31/05/2024 09:45:14-0300
Verifique em https://validar.iti.gov.br

Profª. Drª. Tarciana Liberal Pereira de Araujo
Coorientadora (PPGMMC)

Documento assinado digitalmente
gov.br
TARCIANA LIBERAL PEREIRA DE ARAUJO
Data: 31/05/2024 10:19:25-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Sérgio de Carvalho Bezerra
Examinador Interno ao Programa (PPGMMC)

Documento assinado digitalmente
gov.br
SERGIO DE CARVALHO BEZERRA
Data: 31/05/2024 15:58:50-0300
Verifique em https://validar.iti.gov.br

Profª. Drª. Ana Hermínia Andrade e Silva
Examinadora Externa ao Programa (UFPB)

Documento assinado digitalmente
gov.br
ANA HERMINIA ANDRADE E SILVA
Data: 31/05/2024 14:30:12-0300
Verifique em https://validar.iti.gov.br

Dedico este trabalho a todos aqueles que torcem pelo meu crescimento profissional, que me apoiam e ajudam.

Agradecimentos

Agradeço a Deus por todas as bênçãos concedidas na minha vida.
À minha família, em especial, aos meus pais e meu irmão, por serem inspiração de força e dedicação. Por não medirem esforços para me apoiar e incentivar. Obrigado por tudo.
Às minhas orientadoras, professora Tatiene Correia e Tarciana Liberal, por toda orientação, competência e amizade ao longo desse tempo. Obrigado por todo conhecimento transmitido e direcionamento para a conclusão desta dissertação.
Aos amigos que fiz ao longo do curso, obrigado pelos momentos de descontração e partilha de conhecimento. Em especial, Valter, Franklin, Kleby, Valdemi, Anderson, Adriana e Eduarda, pela amizade e troca de experiências. Foi muito gratificante conviver com vocês.
Agradeço à banca examinadora, antecipadamente, pelas contribuições dadas para a melhoria deste trabalho desde a qualificação. A todos os professores do Programa de Pós-graduação em Modelagem Matemática e Computacional, por contribuírem para minha formação acadêmica.
À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, pelo apoio financeiro.
Por fim, em geral agradeço a todos aqueles que por algum motivo ou situação, compartilharam conhecimento e ajudaram para que eu pudesse desfrutar deste momento.

Resumo da Dissertação apresentada ao PPGMMC/CI/UFPB como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA COMPARAÇÃO DAS DISTRIBUIÇÕES INFLACIONADAS PARA MODELAR DADOS DUPLAMENTE LIMITADOS

ISAAC FERREIRA DE LIMA

Maio/2024

Orientadores: Tatiene Correia de Souza
Tarciana Liberal Pereira de Araújo

Programa: Modelagem Matemática e Computacional

No contexto de dados duplamente limitados, as distribuições inflacionadas são muito utilizadas para casos em que os dados contenham excesso de zeros e/ou uns. Existem na literatura algumas distribuições para lidar com dados nesse domínio como, por exemplo: beta inflacionada (Ospina e Ferrari, 2010), kumaraswamy inflacionada (Cribari-Neto e Santos, 2019) e gama unitária inflacionada (Silva, 2023). Neste trabalho, iremos apresentar as distribuições de probabilidade mais utilizadas no contexto de modelagem de dados distribuídos no intervalo unitário padrão, mas que podem conter zeros e/ou uns e, a partir de um estudo detalhado, via simulação de Monte Carlo, identificar os cenários, de acordo com as características que são mais adequados para cada distribuição.

Abstract of Dissertation presented to PPGMMC/CI/UFPB as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A COMPARISON OF INFLATED DISTRIBUTIONS FOR MODELING DOUBLY BOUNDED DATA

ISAAC FERREIRA DE LIMA

May/2024

Advisors: Tatiene Correia de Souza
Tarciana Liberal Pereira de Araújo

Program: Computational Mathematical Modelling

In the context of doubly bounded data, inflated distributions are widely used for cases where the data contain an excess of zeros and/or ones. There are several distributions in the literature to handle data in this domain, such as inflated beta (Ospina and Ferrari, 2010), inflated Kumaraswamy (Cribari-Neto and Santos, 2019), and inflated unitary gamma (Silva, 2023). In this work, we will present the most commonly used probability distributions in the context of modeling data distributed in the standard unit interval, but which may contain zeros and/or ones. Through a detailed study, via Monte Carlo simulation, we will identify scenarios according to the characteristics that are most suitable for each distribution.

Sumário

Lista de Figuras **x**

Lista de Tabelas **xi**

1 Introdução **1**

1.1 Organização da dissertação 2

2 Distribuições Inflacionadas no Intervalo Unitário Padrão **3**

2.1 Distribuição beta inflacionada 3

2.1.1 Distribuição beta 3

2.1.2 Distribuição beta inflacionada em zero ou um 4

2.1.3 Distribuição beta inflacionada em zero e um 5

2.2 Distribuição gama unitária Inflacionada 6

2.2.1 Distribuição gama unitária 6

2.2.2 Distribuição gama unitária inflacionada 6

2.3 Distribuição kumaraswamy inflacionada 8

2.3.1 Distribuição kumaraswamy 8

2.3.2 Distribuição kumaraswamy inflacionada em zero ou um 9

2.3.3 Distribuição kumaraswamy inflacionada em zero e/ou um . . . 10

2.4 Critérios de Seleção 10

2.4.1 Critério de Informação de Akaike 11

2.4.2 Critério de Informação Bayesiano 11

2.4.3 HQ 11

2.5 Teste de aderência 12

2.5.1 Kolmogorov-Smirnov 12

2.5.2 Anderson-Darling 12

2.5.3 Cramér–von Mises 13

3 Estudo de simulação **14**

3.1 Resultados numéricos 14

3.1.1 Inflacionamento em zero 15

3.1.2 Inflacionamento em zero e um 19

3.1.3 Inflacionamento em um 24

4 Aplicações das Distribuições Inflacionadas 28

4.1 Introdução 28
4.2 Proporção de pessoas que usa internet 28
4.3 Proporção de nascidos vivos com 4000 gramas ou mais 30
4.4 Proporção de pacientes com tuberculose e HIV 33
4.5 Proporção da população urbana que residente em domicílios com rede de esgoto sanitário nos municípios do estado do Espírito Santo 35
4.6 Proporção de população urbana residente em domicílios ligados a rede de esgoto sanitário nos municípios do estado do Rio Grande do Norte 37
4.7 Proporção de votos para o candidato Jair Bolsonaro por seções eleitorais da cidade de Chaves–PA 39
4.8 Proporção de pessoas que residem em domicílios com acesso a energia elétrica 41

5 Conclusão 44

Referências Bibliográficas 48

A 52

B 74

C 271

Lista de Figuras

4.1 Histograma da proporção de pessoas dos países que utilizaram internet em 209 países. 30
4.2 QQ-Plot da proporção de pessoas que utilizaram internet em 209 países. 30
4.3 Histograma da proporção de nascidos com 4000 gramas ou mais em 223 municípios. 32
4.4 QQ-Plot da proporção de nascidos vivos com 4000 gramas ou mais em 223 municípios. 32
4.5 Histograma da proporção de pacientes com tuberculose e HIV em 178 países. 34
4.6 QQ-Plot da proporção de pacientes com tuberculose e HIV em 178 países. 34
4.7 Histograma da proporção de pessoas que residem em domicílio com rede esgoto sanitário em 52 municípios. 36
4.8 QQ-Plot da proporção de pessoas que residem em domicílio com rede de esgoto sanitário em 52 municípios. 36
4.9 Histograma da proporção da população que residem em domicílio com rede esgoto sanitário em 55 municípios. 38
4.10 QQ-Plot da porcentagem da população que residem em domicílios com rede de esgoto sanitário em 55 municípios. 39
4.11 Histograma da proporção de votos para Bolsonaro em 53 seções eleitorais. 41
4.12 QQ-Plot da proporção de votos para Bolsonaro em 53 seções eleitorais. 41
4.13 Histograma da proporção de pessoas que residem em domicílio com acesso a energia elétrica em 293 municípios. 43
4.14 QQ-Plot da proporção de pessoas que residem em domicílio com acesso a energia elétrica em 293 municípios. 43

Lista de Tabelas

3.1 Cenários considerados. 15
3.2 Resultados da proporção de escolha dos critérios para dados gerados com média $0,3$. 17
3.3 Resultados da proporção de escolha dos critérios para dados gerados com média $0,5$. 18
3.4 Resultados da proporção de escolha dos critérios para dados gerados com média $0,3$. 21
3.5 Resultados da proporção de escolha dos critérios para dados gerados com média $0,5$. 22
3.6 Resultados da proporção de escolha dos critérios para dados gerados com média $0,8$. 23
3.7 Resultados da proporção de escolha dos critérios para dados gerados com média $0,5$. 26
3.8 Resultados da proporção de escolha dos critérios para dados gerados com média $0,8$. 27

4.1 Medidas descritivas da proporção de pessoas que utilizaram internet. 29
4.2 p-valores dos testes de hipóteses. 29
4.3 Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - dados da proporção de pessoas utilizaram a internet em 209 países. 29
4.4 Medidas descritivas da proporção de nascidos vivos com 4000 gramas ou mais. 31
4.5 P-valores dos testes de hipóteses. 31
4.6 Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - dados da proporção de nascidos com 4000 gramas ou mais nos 223 municípios. 32
4.7 Medidas descritivas da proporção de pacientes com tuberculose e HIV. 33
4.8 P-valores dos testes de hipóteses. 33

4.9 Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pacientes com tuberculose e HIV em 178 países. 34
4.10 Medidas descritivas da proporção de pessoas que residem em domicílios com rede de esgoto sanitário. 35
4.11 P-valores dos testes de hipóteses. 35
4.12 Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pessoas que reside em domicílio com rede de esgoto sanitário em 52 municípios. 37
4.13 Medidas descritivas da proporção de pessoas que residem em domicílios com rede de esgoto sanitário. 37
4.14 P-valores dos testes de hipóteses 38
4.15 Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pessoas que residem em domicílio com rede de esgoto sanitário em 55 municípios. 39
4.16 Medidas descritivas da proporção de votos para o candidato Jair Bolsonaro. 39
4.17 P-valores dos testes de hipóteses. 40
4.18 Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de votos para o candidato Jair Bolsonaro em 53 seções eleitorais. 40
4.19 Medidas descritivas da proporção de pessoas que residem em domicílios com acesso a energia elétrica. 42
4.20 P-valores dos testes de hipóteses. 42
4.21 Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pessoas que reside em domicílios com acesso a energia elétrica por municípios do estado de Santa Catarina. 43

5.1 Melhores cenários e distribuições mais escolhidas de acordo com os critérios. 45
5.2 Melhores cenários e distribuições mais escolhidas de acordo com os testes de hipótese. 46
5.3 Resumo das aplicações. 47

A.1 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$. 53
A.2 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 54

A.3 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$. 55
A.4 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 56
A.5 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$. 57
A.6 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 58
A.7 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$. 59
A.8 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 60
A.9 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$. 61
A.10 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$. 62
A.11 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 63
A.12 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$. 64
A.13 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$. 65
A.14 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 66
A.15 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$. 67
A.16 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 68
A.17 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$. 69
A.18 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 70
A.19 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$. 71
A.20 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$. 72
A.21 Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$. 73

Capítulo 1

Introdução

áreas.

Em estudos recentes, novas classes de distribuições de probabilidade têm sido propostas buscando fornecer modelos mais adequados para dados no intervalo (0,1). Grassia (1977) propôs a distribuição Gama unitária, Kumaraswamy (1980) propôs a distribuição kumaraswamy, Barndorff-Nielsen e Jϕrgensen (1991) introduziram a distribuição Simplex, Ferrari e Cribari-neto (2004) propuseram a distribuição beta, Mazucheli et al. (2018) propuseram a distribuição weibull unitária, etc.

Contudo, na presença de zeros e/ou uns, essas distribuições, que são absolutamente contínuas, não são mais adequadas. Para estes casos, faz-se necessário utilizar outra classe de distribuições chamada de distribuições inflacionadas, que são uma classe de modelos estatísticos usados para lidar com dados que possuem uma quantidade significativa de zeros e/ou uns. Essas distribuições são amplamente utilizadas em diversos campos, como epidemiologia, economia, ciências sociais, ecologia, entre outros.

Tong et al. (2013) utilizaram um modelo de distribuição gama com zeros ajustado para avaliar as perdas de empréstimos hipotecários em situações de inadimplência. Broek (1995) utilizou a distribuição Poisson inflacionada em zero para investigar o número de casos de infecção pelo HIV em homens. Nobre et al. (2017) utilizaram a distribuição gama com zeros ajustados para investigar o comportamento de indivíduos envolvidos em atividades físicas e determinar a quantidade de horas que dedicam ao treinamento diariamente.

Esses trabalhos apresentam vantagens em se usar cada distribuição abordada. Contudo, é de extrema importância identificar qual distribuição se adequa melhor a determinado conjunto de dados, seja pela sua natureza variabilidade ou proporção de zeros e/ou uns. Dessa forma, o objetivo dessa dissertação é, através de simulação de Monte Carlo, identificar padrões e características dos dados, distribuídos no intervalo unitário padrão mas que podem conter zeros e/ou uns para fornecer aos usuários cenários mais adequados para cada distribuição.

Dentre as distribuições inflacionadas que existem para modelar dados com excesso de zero e/ou uns serão abordadas, neste trabalho, as distribuições: beta inflacionada (Ospina e Ferrari, 2010), kumaraswamy inflacionada (Cribari-Neto e Santos, 2019) e gama unitária inflacionada (Silva, 2023). Essa comparação será realizada utilizando os critérios de informação (AIC, AICc, BIC, BICc, HQ e HQc) e testes de hipóteses: Kolmogorov–Smirnoff, Anderson–Darling e Cramér–Von Mises.

1.1 Organização da dissertação

Em termos de estrutura, esta dissertação está dividida em quatro Capítulos, além da Introdução. No Capítulo 2, apresentamos as distribuições de probabilidade inflacionadas, abordadas nessa dissertação bem como os critérios de seleção e testes de hipóteses para comparar as distribuições. No Capítulo 3, apresentamos a avaliação numérica incluindo a descrição do método de Monte Carlo, o processo gerador dos dados e os resultados da avaliação computacional. No Capítulo 4, apresentamos aplicações das distribuições inflacionadas a dados reais. O Capítulo 5 apresenta as Conclusões deste trabalho.

Capítulo 2

Distribuições Inflacionadas no Intervalo Unitário Padrão

Neste Capítulo, apresentamos as distribuições de probabilidade contínuas no intervalo unitário padrão, abordadas nessa dissertação, bem como suas versões inflacionadas. Nas duas últimas seções deste Capítulo, são apresentados os critérios de informações e os testes de hipótese que serão úteis para comparação das distribuições inflacionadas, ou seja, verificar a adequabilidade das três distribuições aos dados.

2.1 Distribuição beta inflacionada

2.1.1 Distribuição beta

Seja Y uma variável aleatória que segue distribuição beta se sua densidade é dada por:

$$f(y;\mu,\phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \;\; y \in (0,1), \qquad (2.1)$$

em que $0 < \mu < 1$ e $\phi > 0$, em que a esperança e variância são dadas, respectivamente, por, $\mathrm{E}(Y) = \mu$ e $\mathrm{Var}(Y) = \mu(1-\mu)/(1+\phi)$. Essa parametrização da distribuição beta foi proposta por Ferrari e Cribari-Neto (2004). Dizemos que Y tem distribuição beta com média $\mu \in (0,1)$, precisão ϕ e expressamos $Y \sim \mathcal{B}(\mu,\phi)$. Naturalmente, a partir da escolha de (μ,ϕ), pode-se obter diferentes formas de densidades. Sendo assim, (2.1) pode formar uma classe de densidades beta.

A função de distribuição acumulada de Y, por sua vez, é:

$$F(Y) = \frac{\mathcal{B}_y(\mu,\phi)}{\mathcal{B}(\mu,\phi)},$$

para $0 < y < 1$, em que $\mathcal{B}_y(\mu, \phi)$ é a função beta incompleta e $\mathcal{B}(\mu, \phi)$ é a função beta, sendo expressas respectivamente por:

$$\mathcal{B}_y(\mu, \phi) = \int_0^y t^{\mu-1}(1-t)^{\phi-1}dt \quad \text{e} \quad \mathcal{B}(\mu, \phi) = \frac{\Gamma(\mu)\Gamma(\phi)}{\Gamma(\mu+\phi)}.$$

2.1.2 Distribuição beta inflacionada em zero ou um

Segundo Ospina e Ferreira (2010), em contextos reais, é possível encontrar dados representados por taxas, frações ou proporções que incluem zeros e/ou uns. Esses valores extremos podem estar relacionados a intervenções, truncamentos ou censura nos dados. Nesta situação, a distribuição beta não é mais adequada para representar tais dados.

Quando o conjunto de dados contêm zeros e/ou uns (mas não ambos), uma abordagem natural seria incorporar à distribuição beta um ponto de massa em zero ou um. Sendo assim, pode-se obter modelos para frações observadas nos intervalos [0, 1) ou (0, 1].

A partir deste contexto, podemos supor que o componente contínuo dos dados pode ser modelado pela distribuição beta (2.1), pelo fato de que esta distribuição é bastante flexível e pode se ajustar aos dados no intervalo (0, 1). Já o componente discreto, ou seja, o ponto de massa, será modelado por uma distribuição degenerada no valor conhecido c, em que c é igual a zero ou um, dependendo das características dos dados.

A função densidade de probabilidade de y com respeito à medida gerada pela mistura é expressa da seguinte forma:

$$bi_c(y; \alpha, \mu, \phi) = \begin{cases} \alpha, & \text{se } y = c \\ (1-\alpha)f(y; \mu, \phi), & \text{se } y \in (0,1), \end{cases} \tag{2.2}$$

em que $0 < \alpha,\ \mu < 1$ e $\phi > 0$, sendo $f(y; \mu, \phi)$ a função densidade (2.1). Observe que $\alpha = P(y = c)$ representa a probabilidade de se ter zero ($c = 0$) ou um ($c = 1$), de acordo com o cenário. A densidade (2.2) pode ser escrita da seguinte forma:

$$bi_c(y; \alpha, \mu, \phi) = \{\alpha^{1_c(y)}(1-\alpha)^{1-1_c(y)}\}\{f(y; \mu, \phi)^{1-1_c(y)}\}.$$

Podemos ver que esta densidade, fatora em dois termos, em que o primeiro depende somente de α e o segundo depende somente de (μ, ϕ). Se $c = 0$, a distribuição (2.2) é chamada de distribuição beta inflacionada no ponto zero e escrevermos $Y \sim BIZ(\alpha, \mu, \phi)$ e $\alpha = P(y = 0)$. Se $c = 1$, a distribuição (2.2) é chamada de distribuição beta inflacionada no ponto um e escrevermos $Y \sim BIU(\alpha, \mu, \phi)$ e $\alpha = P(y = 1)$. A média e a variância de uma variável aleatória com distribuição

beta inflacionada em zero ou um, são dadas por:

$$E(Y^r) = \alpha c + (1-\alpha)\mu_r,$$
$$Var(Y) = (1-\alpha)\frac{V(\mu)}{\phi+1} + \alpha(1-\alpha)(c-\mu)^2,$$

em que $c = 0$ ou $c = 1$, indicando o inflacionamento em zero ou um, respectivamente, $V(\mu) = \mu(1-\mu)$, μ_r é o r-ésimo momento ao redor de zero da distribuição $\mathcal{B}(\mu, \phi)$, dada em (2.1) e ϕ é interpretado como um parâmetro de precisão uma vez que a variância diminui, a medida que o parâmetro ϕ aumenta.

A função de distribuição acumulada é dada por

$$BI_c(y; \alpha, \mu, \phi) = \alpha \mathbf{1}_c(y) + (1-\alpha)F(y; \mu, \phi),$$

em que $\mathbf{1}_A(y)$ é uma função indicadora, que assume valor 1 se $y \in A$ e 0 se $y \notin A$. A função $F(\cdot; \mu, \phi)$ é a função de distribuição acumulada beta $\mathcal{B}(\mu, \phi)$ e $0 < \alpha < 1$ trata-se do parâmetro de mistura.

2.1.3 Distribuição beta inflacionada em zero e um

De acordo com Ospina e Ferreira (2010), nas situações em que a variável está distribuída no intervalo $[0, 1]$, admite-se que as probabilidades de observar os valores zero e um são positivas. Para esse tipo de dado, faz se necessário utilizar uma mistura entre uma distribuição beta e uma distribuição Bernoulli, a qual atribui probabilidades não-negativas aos inteiros 0 e 1. A distribuição beta cumpre a função de modelar o componente contínuo e a distribuição de Bernoulli adequa o componete discreto, ou seja, os pontos de massa em zero e um.

Seja Y uma variável aleatória que tem distribuição beta inflacionada em zero e um (BIZU), a função densidade com respeito à medida gerada pela mistura é dada por:

$$bizu(y; \alpha, \gamma, \mu, \phi) = \begin{cases} \alpha\gamma & \text{se} \quad y = 1 \\ \alpha(1-\gamma) & \text{se} \quad y = 0 \\ (1-\alpha)f(y; \mu, \phi) & \text{se} \quad y \in (0, 1), \end{cases}$$

com $0 < \alpha, \gamma, \mu < 1$ sendo $f(y; \mu, \phi)$ a função densidade beta (2.1) e escrevemos $Y \sim BIZU(\alpha, \gamma, \mu, \phi)$.

Note que $P(Y = 1) = \alpha\gamma$, $P(Y = 0) = \alpha(1-\gamma)$. Podemos calcular $\mathrm{E}(Y^r)$ e

Var(Y) da seguinte maneira:

$$\mathrm{E}(Y^r) = \alpha\gamma + (1-\alpha)\mu_r,$$
$$\mathrm{Var}(Y) = \alpha\gamma(1-\gamma) + (1-\alpha)\frac{\mu(1-\mu)}{(\phi+1)} + \alpha(1-\alpha)(\gamma-\mu)^2.$$

A função de distribuição acumulada é dada por

$$BIZU(y;\alpha,\gamma,\mu,\phi) = \alpha Ber(y,\gamma) + (1-\alpha)F(y;\mu,\phi),$$

em que $Ber(\cdot;\gamma)$ é a função de distribuição acumulada de uma variável aleatória de Bernoulli de parâmetro γ e $F(\cdot;\mu,\phi)$ é a função de distribuição acumulada de uma variável aleatória com distribuição beta $\mathcal{B}(\mu,\phi)$. Os parâmetros são $0 < \mu$, γ, $\alpha < 1$ e $\phi > 0$, em que o parâmetro de mistura α permite combinar de forma convexa as duas distribuições.

2.2 Distribuição gama unitária Inflacionada

2.2.1 Distribuição gama unitária

A distribuição gama unitária, proposta por Grassia (1977), é uma distribuição probabilística bastante presente na literatura e útil para modelar variáveis no intervalo $(0,1)$. Uma variável aleatória Y segue a distribuição gama unitária quando sua função de densidade é representada por:

$$f(y;\mu,\phi) = \frac{\left(\frac{\mu^{1/phi}}{1-\mu^{1/\phi}}\right)^{\phi}}{\Gamma(\phi)} y^{\left(\frac{\mu^{1/\phi}}{1-\mu^{1/\phi}}-1\right)} \left[\log\left(\frac{1}{y}\right)\right]^{\phi-1}, \ 0 < y, \mu < 1, \ \text{e} \ \phi > 0. \quad (2.3)$$

A distribuição gama unitária é bastante flexível e, dependendo dos valores de seus parâmetros, pode assumir diversas formas. Tal distribuição é adequada para modelar variáveis que estejam no intervalo unitário padrão. Existem situações nas quais temos interesse em investigar variáveis que são limitadas aos intervalos $(0,1]$, $[0,1)$ ou até mesmo $[0,1]$. Para estes casos, faz-se necessário o uso de distribuições inflacionadas.

2.2.2 Distribuição gama unitária inflacionada

A distribuição gama unitária inflacionada foi proposta por Silva (2023). Uma variável aleatória Y que segue a distribuição gama unitária inflacionada em zero e

um, $Y \sim GUI(\delta_0, \delta_1, \mu, \phi)$, pode ser expressa por:

$$f(y; \delta_0, \delta_1, \mu, \phi) = \begin{cases} \delta_0 & \text{se} \quad y = 0 \\ (1 - \delta_0 - \delta_1) f(y; \mu, \phi) & \text{se} \quad y \in (0, 1), \\ \delta_1 & \text{se} \quad y = 1 \end{cases} \tag{2.4}$$

em que δ_0 denota probabilidade de y assumir o valor 0, δ_1 a probabilidade de y ser igual a 1, $f(y; \mu, \phi)$ é a função densidade da distribuição gama unitária apresentada em (2.3), $0 < \delta_0, \delta_1, \mu < 1$, $0 < \delta_0 + \delta_1 < 1$ e $\phi > 0$. A média e variância de uma variável aleatória $Y \sim GUI(\delta_0, \delta_1, \mu, \phi)$ são expressas por:

$$\begin{aligned} \mathrm{E}(Y) &= \delta_1 + (1 - \delta_0 - \delta_1)\mu, \\ \mathrm{Var}(Y) &= \delta_1(1 - \delta_1) + (1 - \delta_0 - \delta_1)\left[\mathrm{Var}_{GU} - 2\mu\delta_1 + \mu^2(\delta_0 + \delta_1)\right], \end{aligned}$$

em que a $\mathrm{Var}_{\mathrm{GU}}$ é a variância de uma variável aleatória, que segue a distribuição gama unitária,

$$\mathrm{Var}_{GU} = \mu \left[\frac{1}{(2 - \mu^{1/\phi})^{\phi}} - \mu \right].$$

De forma semelhante ao trabalho de Bayes e Valdivieso (2016), Silva (2023) propôs reparametrizar a distribuição gama unitária inflacionada (2.4) com intuito de modelar a média da distribuição gama unitária inflacionada e não a média da distribuição gama unitária. Dessa forma o valor esperado de $Y \sim GUI(\delta_0, \delta_1, \mu, \phi)$, $\gamma = E(Y)$, depende de μ e é restrito para os valores δ_0 e δ_1 de acordo com

$$\delta_1 < \gamma < 1 - \delta_0.$$

Deste modo, para permitir que os parâmetros do modelo sejam estimados sem restrições, Silva (2023) sugeriu utilizar a seguinte parametrização:

$$\gamma = \delta_1 + (1 - \delta_0 - \delta_1)\mu, \quad \alpha_0 = \frac{\delta_0}{1 - \gamma} \quad \text{e} \quad \alpha_1 = \frac{\delta_1}{\gamma},$$

em que $0 < \alpha_0$, α_1, $\gamma < 1$, $\phi > 0$ e $\delta_1 < \gamma < 1 - \delta_0$. Deste modo, considerando a nova parametrização tem-se que a densidade é dada por:

$$f(y; \alpha_0, \alpha_1, \gamma, \phi) = \begin{cases} \alpha_0(1 - \gamma) & \text{se} \quad y = 0 \\ (1 - \alpha_0(1 - \gamma) - \alpha_1\gamma) f\left(y; \frac{y(1-\alpha_1)}{1-\alpha_0(1-\gamma)-\alpha_1\gamma}, \phi\right) & \text{se} \quad y \in (0, 1) \\ \alpha_1\gamma & \text{se} \quad y = 1. \end{cases} \tag{2.5}$$

Desta forma, a esperança e a variância da variável aleatória Y com distribuição

gama unitária inflacionada reparametrizada é dada por

$$\mathrm{E}(Y) = \gamma,$$

$$\mathrm{Var}(Y) = \alpha_1\gamma(1-\alpha_1\gamma) + \gamma(1-\alpha_1)\left[\left(2 - \left(\frac{\gamma(1-\alpha_1)}{1-\alpha_0(1-\gamma)-\alpha_1\gamma}\right)^{1/\phi}\right)^{-\phi} - \gamma(1-\alpha_1)\right].$$

Perceba que a expressão (2.5) é facilmente compreendida para casos em que a variável de interesse assume apenas um dos extremos, com $\delta_0 = 0$ e $\delta_1 > 0$ (distribuição inflacionada em 1), $\delta_0 > 0$ e $\delta_1 = 0$ (distribuição inflacionada em 0) ou $\delta_0 = \delta_1 = 0$ (distribuição Gama Unitária). Quando a distribuição é inflacionada em um, temos que $\gamma = \delta_1 + (1-\delta_1)\mu$ e $\alpha_1 = \frac{\delta_1}{\gamma}$, já quando a distribuição é inflacionada em zero, temos que $\gamma = (1-\delta_0)\mu$ e $\alpha_0 = \frac{\delta_0}{1-\gamma}$. A função distribuição acumulada da variável aleatória $Y \sim GUI(\alpha_0, \alpha_1, \gamma, \phi)$ é dada por:

$$F(y;\alpha_0,\alpha_1,\gamma,\phi) = \begin{cases} 0, & \text{se } y < 0 \\ \alpha_0(1-\gamma), & \text{se } y = 0 \\ \alpha_0(1-\gamma) + \psi F\left(y|\frac{y(1-\alpha_1)}{1-\alpha_0(1-\gamma)-\alpha_1\gamma}, \phi\right), & \text{se } 0 < y < 1 \\ 1, & \text{se } y > 1, \end{cases}$$

em que $\psi = 1 - \alpha_0(1-\gamma) - \alpha_1\gamma$.

2.3 Distribuição kumaraswamy inflacionada

2.3.1 Distribuição kumaraswamy

A distribuição kumaraswamy foi proposta por Kumaraswamy (1976), e foi desenvolvida para ajustar dados delimitados por um limite superior e inferior. Mitnik e Baek (2013), propuseram uma parametrização baseada na mediana da distribuição kumaraswamy com suporte do intervalo unitário aberto (0,1). Seja Y uma variável aleatória segue a distribuição kumaraswamy, cuja notação é $Y \sim K(\omega, \phi)$, com função densidade de probabilidade (f.d.p) dada por:

$$f(y;\omega,\phi) = \frac{\phi\log(0.5)}{\log(1-\omega^\phi)} y^{\phi-1}(1-y^\phi)^{\frac{\phi\log(0.5)}{\log(1-\omega^\phi)}-1}, \tag{2.6}$$

em que $\omega \in (0,1)$ é a mediana e $\phi > 0$ é o parâmetro de precisão. O valor esperado e a variância de $Y \sim K(\omega, \phi)$ são expressos por

$$\mathrm{E}(Y) = \delta\mathrm{B}\left(1 + \frac{1}{\phi}, \delta\right),$$

$$\text{Var}(Y) = \sigma^2 = \delta \text{B}\left(1 + \frac{2}{\phi}, \delta\right) - \left[\delta \text{B}\left(1 + \frac{1}{\phi}, \delta\right)\right]^2, \tag{2.7}$$

em que $\delta = \frac{\phi \log(0.5)}{\log(1-\omega^\phi)}$, $\text{B}(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ é a função beta e $\Gamma(x) = \int_0^\infty t^{v-1}e^{-t}dt$ é a função Gama. A função de distribuição acumulada (f.d.a) dada por:

$$F(y;\omega,\phi) = 1 - (1 - y^\phi)^{\frac{\phi \log(0.5)}{\log(1-\omega^\phi)}}.$$

2.3.2 Distribuição kumaraswamy inflacionada em zero ou um

A distribuição kumaraswamy inflacionada foi proposta por Cribari-Neto e Santos (2019). Quando os dados contêm observações em um dos extremos do intervalo unitário padrão (mas não em ambos simultaneamente) com probabilidade não nula, a distribuição kumaraswamy inflacionada em zero ou um é uma mistura de duas distribuições: uma distribuição $K(\omega,\phi)$ e uma distribuição degenerada no ponto c(c = 0 ou c = 1).

A função densidade de probabilidade de uma variável aleatória Y que segue distribuição kumaraswamy inflacionada em zero ou um, $Y \sim KI(y;\omega,\phi)$, pode ser expressa por:

$$ki_c(y;\gamma,\omega,\phi) = \left[\gamma^{\mathbb{I}_c(y)}(1-\gamma)^{1-\mathbb{I}_c(y)}\right]\left[f(y;\omega,\phi)^{1-\mathbb{I}_c(y)}\right],$$

em que $0 < \gamma, \mu < 1$, $\gamma = P(Y = c)$, ω é a mediana e $\phi > 0$ é o parâmetros de precisão, $f(y;\omega,\phi)$, é a densidade expressa em (2.6).

A função de densidade acumulada inflacionada em c é dada por

$$KI_c(y;\gamma,\omega,\phi) = \gamma + (1-\gamma)F(y;\omega,\phi),$$

em $0 < \gamma < 1$, $0 < \omega < 1$ é a mediana, $\phi > 0$ é o parâmmetro de precisão e $F(y;\omega,\phi)$ é a distribuição acumulada da densindade $K(y;\omega,\phi)$.

A média e a variância da distribuição $Y \sim KI(\gamma,\omega,\phi)$ são dadas, respectivamente, por:

$$\text{E}(Y) = \gamma c + (1-\gamma)\omega_K,$$

$$\text{Var}(Y) = (1-\gamma)\sigma^2 + \gamma(1-\gamma)(c-\omega_K)^2,$$

em que ω_K é o primeiro momento da distribuição kumaraswamy e σ^2 é variância da distribuição $K(\omega,\phi)$ dada pela expressão (2.7).

2.3.3 Distribuição kumaraswamy inflacionada em zero e/ou um

A distribuição da seção anterior não é adequada para modelar dados fracionários quando ocorre inflacionamento dos dados em ambos os extremos do intervalo unitário padrão. Bayer et al. (2021) fez uma reparametrização da distribuição proposta por Cribari-Neto e Santos (2019), baseado na mediana de Mitnik e Baek (2013). Dizemos que a variável aleatória Y segue a distribuição Kumaraswamy inflacionada em zero e/ou um, se sua função densidade de probabilidade de Yé dada por:

$$zoik(y;\lambda,p,\omega,\phi) = \begin{cases} \lambda(1-p) & \text{se} \quad y=0 \\ \lambda p & \text{se} \quad y=1 \\ (1-\lambda)f(y;\omega,\phi) & \text{se} \quad y\in(0,1), \end{cases}$$

em que $0<\lambda<1$ é o parâmetro de mistura, p é a probabilidade de que uma variável aleatória de distribuída Bernoulli seja igual a um e $f(y;\omega,\phi)$ é a função densidade kumaraswamy dada em (2.6), que é indexada por $0<\omega<1$ (mediana) e $\phi>0$ (parâmetro de precisão). Vale ressaltar que, quando $p=0$, tem-se a distribuição kumaraswamy inflacionada em zero, e quando $p=1$, tem-se a distribuição kumaraswamy inflacionada em um, como casos particulares.

A média e a variância da distribuição $Y\sim zoik(y;\lambda,p,\omega,\phi)$ são dadas, respectivamente, por:

$$\mathrm{E}(Y) = \lambda p + \phi(1-\lambda)\mathbb{B}\left(1+\frac{1}{\omega},\phi\right),$$

$$\mathrm{Var}(Y) = \lambda p(1-\lambda p) + (1-\lambda)\phi\left\{\mathbb{B}\left(1+\frac{2}{\omega},\phi\right) - \mathbb{B}\left(1+\frac{1}{\omega},\phi\right)[\Phi_1]\right\},$$

em que $\Phi_1 = 2\lambda p + \phi(1-\lambda)\mathbb{B}\left(1+\frac{1}{\omega},\phi\right)$ e $\mathbb{B}(.,.)$ é a função beta.

2.4 Critérios de Seleção

Não há dúvidas de que, sob a perspectiva estatística, a seleção da distribuição mais apropriada para representar um fenômeno que está sendo estudado é uma etapa de extrema importância. Na literatura muitos autores utilizam os critérios de informação para comparar a adequabilidades das distribuições de probabilidades a certos conjuntos de dados, (MAZUCHELI; MENEZES; GHITANY, 2018; MAZUCHELI; MENEZES; DEY, 2019; BAPAT; BHARDWAJ, 2021).

Nos critérios que serão apresentados a seguir, serão utilizados os seguintes elementos: $l(\hat{\theta})$, k e n. Neste contexto, $l(\hat{\theta})$ representa a função de log-verossimilhança

maximizada, $\hat{\theta}$ denota o estimador de máxima verossimilhança do vetor θ, k indica a quantidade de parâmetros e n representa o número de observações.

2.4.1 Critério de Informação de Akaike

O critério de informação de Akaike (AIC) é o critério de seleção mais utilizado para seleção de modelos estatísticos. Foi proposto por Akaike (1973) a partir da minimização da informação de Kullback-Leibler. O AIC é expresso por:

$$\text{AIC} = -2l(\hat{\theta}) + 2k.$$

Com o propósito de melhorar a performance do AIC em casos de tamanho finito, Sugiura (1978) apresentou um estimador não-viesado para a distância de Kullback-Leibler em regressão linear, o AICc. Já Hurvich e Tsai (1989) expandiu o uso do AICc para modelos de regressão não-lineares e modelos autorregressivos, justificando que o mesmo apresenta desempenho superior ao AIC em pequenas amostras. O AICc é definido como:

$$\text{AICc} = -2l(\hat{\theta}) + \frac{2nk}{n-k-1}.$$

2.4.2 Critério de Informação Bayesiano

Considerando uma perspectiva bayesiana, há o critério de informação BIC proposto por Schwarz (1978). O BIC é definido como:

$$\text{BIC} = -2l(\hat{\theta}) + klog(n).$$

De acordo com McQuarrie (1999), a relação entre os critérios de informação de Akaike (AIC) e sua versão corrigida (AICc), serviu como fundamento para desenvolver uma correção para BIC em pequenas amostras. O BIC é assintoticamente consistente, e essa mesma propriedade se estende ao BICc que é dado por:

$$\text{BICc} = -2l(\hat{\theta}) + \frac{nk\log(n)}{n-k-1}.$$

2.4.3 HQ

O critério HQ foi proposto por Hannan e Quinn (1979). E é dado por:

$$\text{HQ} = -2l(\hat{\theta}) + 2k\log(\log(n)).$$

Assim como o critério AIC e BIC, o critério HQ também tem sua versão corrigida para amostras finitas dada por:

$$\text{HQc} = -2l(\hat{\theta}) + \frac{2nk\log(\log(n))}{n-k-1}.$$

2.5 Teste de aderência

Há diversas avaliações e métricas disponíveis para determinar se o modelo estimado é apropriado para fins de inferência estatística. Na literatura alguns autores utilizaram os testes de hipótese para comprar distribuições inflacionadas a certos conjuntos de dados (CRIBARI-NETO; SANTOS, 2019; CHAKRABORTY; BHATTACHARJEE, 2021). Testes de hipótese são utilizados para validar uma afirmação, frequentemente é referida como hipótese nula, (H_0). Quando se trata de avaliar a adequação, as hipóteses são:

$$\begin{cases} H_0 : \text{A amostra segue o modelo estipulado} \\ H_1 : \text{A amostra não segue o modelo estipulado} \end{cases}$$

2.5.1 Kolmogorov-Smirnov

Kolmogoriff (1941) e Smirnoff (1939) são notáveis por sua contribuição fundamental na concepção do teste de Komogorov-Smirnov (KS). Este método foi desenvolvido com o objetivo de avaliar a disparidade entre distribuições empíricas e uma distribuição teórica postulada. A estatística do teste KS, aplicada a uma distribuição cumulativa teórica $F(x)$ específica, quantifica essa disparidade através da seguinte expressão:

$$KS_n = \sqrt{n}\sup_x |F_n(x) - F(x)|$$

em que $F(x)$ representa o valor teórico da distribuição acumulada em x, enquanto $F_n(x)$ é o valor empírico da distribuição acumulada com base em uma amostra de tamanho n. Se a estatística KS_n ultrapassar o valor crítico KS_α em um determinado nível de significância α, a hipótese nula, que pressupõe que $F_n(x)$ se origina da distribuição teórica $F(x)$, é rejeitada. Isso implica que uma faixa com uma altura KS_α é delimitada em ambos os lados da distribuição teórica, e se a distribuição empírica estiver fora dessa faixa em qualquer ponto, a hipótese nula é rejeitada.

2.5.2 Anderson-Darling

Anderson e Darling (1952) foram os pioneiros no desenvolvimento do teste de Anderson-Darling, apresentando-o como uma alternativa eficaz às outras análises

estatísticas convencionais para detecção de desvios nas distribuições de amostras em relação à normalidade. Similar ao teste de Komogorov-Smirnov, inicialmente sua aplicação estava centrada em contextos de engenharia. A estatística do teste de Anderson-Darling para uma única amostra é calculada conforme a seguinte fórmula:

$$\mathrm{AD} = A(1 + \frac{0,75}{n} + \frac{2,25}{n^2}),$$

em que A é calculado a seguir:

$$\mathrm{A} = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)(\ln(p_{(i)}) + \ln(1 - (p_{(n+1-i)}))),$$

as quantidades $p(i)$ são percentis ordenados da distribuição e n é o tamanho amostral.

2.5.3 Cramér–von Mises

Este teste é baseado na distribuição acumulada e foi proposto por Darling (1957). A estatística do teste de Cramér-von Mises é calculada a partir da seguinte expressão:

$$\mathrm{CVM} = W(1,0 + \frac{0,5}{n}),$$

em que W é expressa por:

$$W = \frac{1}{12n} + \sum_{i=1}^{n}\left(p(i) - \frac{2i-1}{2n}\right),$$

as quantidades $p(i)$ são os percentis ordenados da distribuição e n é o número de observações.

Capítulo 3

Estudo de simulação

Nesta seção são apresentados os resultados de uma simulação de Monte Carlo para comparar as distribuições de probabilidade inflacionadas. Para essas comparações, foram utilizados os critérios de seleção (AIC, AICc, BIC, BICc, HQ e HQc), mencionados na seção 2.4, bem como os testes de hipóteses: Kolmogorov–Smirnov, Anderson–Darling e Cramér–Von Mises.

As simulações foram realizadas utilizando o *software* `R` (R Core Team, 2024), versão R 4.3.3, disponível em `https` : //`posit.co/download/rstudio` − `desktop`/. Os resultados foram baseados em 10.000 réplicas de Monte Carlo. Foram utilizado diversos pacotes do software `R`: `goeftest` (CSÖRGŐ; FARAWAY, 1996), `goeftest` (MARSAGLIA; MARSAGLIA, 2004), `gamlss` (RIGBY; STASINOPOULOS, 2005), `VGAM` (YEE, 2008), `MASS` (RIPLEY; ET. AL, 2013), `gamlss.dist` (RIGBY; et. al, 2019), `extraDistr` e (WOLODZKO; WOLODZKO, 2020). Os cenários foram estabelecidos levando em consideração as seguintes distribuições: beta inflacionada, kumaraswamy inflacionada e gama unitária inflacionada, com inflacionamento em um, zero ou zero e um.

Foram considerados os seguintes tamanhos amostrais: $n = 30$, 60, 100 e 200. As amostras utilizadas nas simulações foram geradas a partir das distribuições beta inflacionada (BI$(0, 1]$, BI$[0, 1)$ e BI$[0, 1]$), kumaraswamy inflacionada (KI$(0, 1]$, KI$[0, 1)$ e KI$[0, 1]$) e gama unitária inflacionada (GUI$(0, 1]$, GUI$[0, 1)$ e GUI$[0, 1]$). Foram calculadas a média, proporção de inflacionamento, variância, proporção de vezes que uma das distribuições foi escolhida em relação aos critérios (AIC, AICc, BIC, BICc, HQ e HQc) e a proporção de não rejeição da hipótese nula H_0: a amostra segue a distribuição testada, ao nível de significância de 5%.

3.1 Resultados numéricos

As simulação foram realizadas considerando nove cenários para cada distribuição, representados na Tabela 3.1.

Tabela 3.1: Cenários considerados.

	Cenários	μ	inflacionamento
	1		5%
$\phi = 10; 30$	2	0,3	7%
	3		10%
	4		5%
$\phi = 10; 50$	5	0,5	7%
	6		10%
	7		5%
$\phi = 10; 50$	8	0,8	7%
	9		10%

Os resultados das simulações em relação aos critérios de escolha encontram-se nas Tabelas 3.2 a 3.8, em que 3.2 e 3.3 são para inflacionamento em zero, 3.4, 3.5 e 3.6 são para inflacionamento em zero e um, e 3.7 e 3.8 são para inflacionamento em um. Vale ressaltar que o modelo selecionado foi o mesmo considerando todos os critérios. Os resultados da proporção de vezes que os testes não rejeitaram a hipótese nula encontra-se no Apêndice A.

3.1.1 Inflacionamento em zero

Para os Cenários 1, 2, 3, 4, 5 e 6, quando os dados são gerados considerando a distribuição beta inflacionada(BI), de acordo com os critérios, foi escolhida a distribuição gama unitária inflacionada(GUI) com maior frequência. Percebeu-se que no cenário 6 (Tabela 3.3), com uma precisão maior ($\phi = 50$), os critérios indicaram a distribuição GUI 70% das vezes quando o tamanho da amostra foi de 200.

No que se refere aos testes de hipóteses, considerando a hipótese nula H_0: os dados seguem distribuição BI, 5% de significância, é possível observar (Tabelas A.1 e A.2) que a proporção de não rejeição de H_0 esteve, em geral, entre 91% e 95%, com exceção de alguns cenários, como por exemplo quando $\mu = 0,5$, $\phi = 50$, $n = 200$, porcentagem de inflacionamento de 10% e considerando o teste de AD (Tabela A.2), observamos um percentual de $83,55\%$.

É importante citar que mesmo os dados sendo gerados de uma distribuição beta inflacionada a hipótese nula de que os dados seguem distribuição gama unitária inflacionada não foi rejeitada na maioria dos cenários, com um percentual de não rejeição de até 90%.

Considerando que os dados foram gerados de uma distribuição KI, de acordo com os critérios, foi escolhida a distribuição KI com maior frequência. Percebeu-se que no cenário 1 (Tabela 3.2), com uma precisão maior ($\phi = 30$), os critérios indicaram a distribuição KI $99,43\%$ das vezes quando o tamanho da amostra foi de 200.

Já sobre os testes de hipóteses (Tabela A.3 e A.4), é possível verificar proporções

de não rejeição altas quando a hipótese nula é de que os dados segue distribuição KI, com valores um pouco menor nos cenários com maior proporção de inflacionamento. Por exemplo, ao observar a proporção de não rejeição da distribuição KI nos cenários 1, 2 e 3 (Tabela A.3), a proporção de não rejeição não são tão altos em 3, como é no cenário 1, os resultados são mais altos ainda quando a precisão é mais baixa ($\phi = 10$) neste cenário, a mesma observação pode ser percebida na Tabela A.4.

É importante citar que a porcentagem de vezes que a hipótese nula foi não rejeitada considerando a distribuição BI e GUI foi zero na maioria dos cenários, com poucas exceções quando $n = 30$ e H_0 de a distribuição GUI e BI é adequada.

Considerando que os dados foram gerados de uma distribuição GUI, de acordo com os critérios, foi escolhida com maior frequência a própria distribuição inflacionada. Percebeu-se que no cenário 6 (Tabela 3.3) com uma precisão maior ($\phi = 50$), os critérios indicam a distribuição GUI $73,35\%$ das vezes quando o tamanho da amostra foi de 200.

No que se refere aos testes de hipóteses, considerando a hipótese nula H_0: os dados seguem distribuição GUI, com 5% de significância, é possível observar (Tabelas A.5 e A.6) que a proporção de vezes que H_0 não foi rejeitado esteve, em geral, entre 91% e 95%, com exceção de alguns casos, como por exemplo os cenários 3 e 6(Tabelas A.5 e A.6), quando $n = 200$, porcentagem de inflacionamento de 10% e considerando os testes de AD e KS.

É importante citar que a proporção de vezes que H_0 foi não rejeitada, considerando a distribuição KI, foi zero na maioria dos casos, com poucas exceções quando $n = 30$. Já a proporção de vezes que H_0 não foi rejeitada, considerando distribuição BI, foram resultados altos, mas não superior à da distribuição GUI.

Tabela 3.2: Resultados da proporção de escolha dos critérios para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim$ BI$[0,1)$			$y \sim$ KI$[0,1)$			$y \sim$ GUI$[0,1)$		
			% de escolha			% de escolha			% de escolha		
			BI$[0,1)$	KI$[0,1)$	GUI$[0,1)$	BI$[0,1)$	KI$[0,1)$	GUI$[0,1)$	BI$[0,1)$	KI$[0,1)$	GUI$[0,1)$
1	10	30	6,75	40,03	53,22	12,30	77,63	10,07	9,21	30,98	59,81
		60	11,51	35,79	52,70	9,01	87,60	3,39	16,13	23,66	60,21
		100	17,87	32,36	49,77	6,24	92,79	0,97	22,99	17,12	59,89
		200	28,60	25,94	45,46	1,52	98,43	0,05	29,49	8,71	61,79
	30	30	13,05	31,21	55,74	23,99	76,01	0,00	14,73	25,65	59,62
		60	22,88	23,32	53,76	10,94	89,06	0,00	22,44	16,59	60,97
		100	29,14	17,10	53,76	5,52	94,48	0,00	29,43	9,99	60,58
		200	41,06	8,37	50,57	0,64	99,36	0,00	36,99	2,99	60,02
2	10	30	5,92	39,51	54,57	11,30	77,56	11,14	7,49	31,51	61,00
		60	10,03	36,73	54,57	8,65	87,17	4,18	15,04	24,00	60,96
		100	16,13	32,62	51,25	5,81	92,79	1,40	21,27	17,55	61,18
		200	24,89	26,43	48,68	1,76	98,13	0,11	28,36	8,80	62,84
	30	30	10,38	31,48	58,14	24,01	75,99	0,00	11,80	26,27	61,93
		60	21,37	23,74	54,89	11,63	88,37	0,00	21,35	16,64	62,01
		100	27,58	17,16	55,26	5,45	94,55	0,00	27,06	10,38	62,56
		200	37,66	8,45	53,89	1,10	98,90	0,00	34,75	3,28	61,97
3	10	30	5,48	39,94	54,58	9,90	76,95	13,15	6,95	32,21	60,84
		60	9,25	36,66	54,09	7,70	86,52	5,78	13,53	24,62	61,85
		100	12,88	33,42	53,70	5,09	92,77	2,14	19,47	18,32	62,21
		200	21,67	26,68	51,65	1,67	98,16	0,17	27,30	9,61	63,09
	30	30	8,99	31,93	59,08	24,27	75,73	0,00	9,80	24,47	63,73
		60	19,51	23,90	56,59	11,79	88,21	0,00	11,81	25,34	62,85
		100	26,17	17,62	56,21	5,63	94,37	0,00	18,54	19,32	62,14
		200	33,48	9,11	57,41	1,04	98,96	0,00	27,30	3,63	63,57

Tabela 3.3: Resultados da proporção de escolha dos critérios para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim$ BI$[0,1)$			$y \sim$ KI$[0,1)$			$y \sim$ GUI$[0,1)$		
			% de escolha			% de escolha			% de escolha		
			BI$[0,1)$	KI$[0,1)$	GUI$[0,1)$	BI$[0,1)$	KI$[0,1)$	GUI$[0,1)$	BI$[0,1)$	KI$[0,1)$	GUI$[0,1)$
4	10	30	3,70	38,89	57,41	6,50	73,30	20,20	6,44	32,14	61,42
		60	6,25	35,21	58,54	8,23	83,30	8,47	11,81	25,34	62,85
		100	9,59	32,28	58,13	6,14	89,52	4,34	18,54	19,32	62,14
		200	14,73	26,24	59,03	2,83	96,94	0,23	27,99	10,60	61,41
	50	30	8,28	28,41	63,31	23,61	76,39	0,00	8,42	24,39	67,19
		60	16,32	20,00	63,68	10,71	89,29	0,00	14,33	14,99	70,68
		100	23,60	13,51	62,89	4,94	95,06	0,00	21,86	8,41	69,73
		200	32,86	5,52	61,62	2,11	97,89	0,00	33,28	2,16	64,56
5	10	30	2,46	38,73	58,81	4,99	73,37	21,64	4,56	32,31	63,13
		60	4,96	35,89	59,15	7,25	83,09	9,66	9,56	25,63	64,81
		100	7,97	32,35	59,68	5,65	89,15	5,20	16,18	19,46	64,36
		200	10,59	26,76	62,65	3,36	96,05	0,59	25,17	10,92	63,91
	50	30	5,29	29,11	65,60	23,51	76,49	0,00	5,32	24,76	69,92
		60	13,88	20,25	65,87	11,57	88,43	0,00	10,91	14,68	74,41
		100	20,36	13,90	65,74	5,35	94,65	0,00	17,33	8,70	73,97
		200	27,81	5,79	66,40	2,33	97,67	0,00	28,57	2,48	68,95
6	10	30	1,97	39,08	58,95	3,46	72,92	23,62	3,43	32,97	63,60
		60	4,03	36,37	59,60	6,34	82,24	11,42	7,06	26,01	66,93
		100	5,12	33,33	61,55	5,19	89,09	5,72	13,22	20,19	66,59
		200	7,41	27,54	65,05	3,17	96,02	0,81	20,25	11,56	68,19
	50	30	3,70	29,54	66,76	24,14	75,86	0,00	3,18	24,70	72,12
		60	10,75	20,97	68,28	11,74	88,26	0,00	7,05	15,65	77,30
		100	15,67	14,39	69,94	5,72	94,28	0,00	14,31	8,86	76,83
		200	23,03	6,44	70,53	1,28	98,72	0,00	24,00	2,65	73,35

3.1.2 Inflacionamento em zero e um

Para os cenários de 1 a 9, quando os dados são geradas pela distribuição BI, de acordo com os critérios, foi escolhida a distribuição GUI com maior frequência. Percebeu-se que nos cenários 7, 8 e 9 (Tabela 3.6), com uma precisão maior ($\phi = 50$), os critérios indicaram a distribuição GUI cerca 80% das vezes quando o tamanho das amostras foram de 200.

No que se refere aos testes de hipóteses, considerando a hipótese nula H_0: os dados seguem distribuição BI, 5% de significância, é possível observar (Tabelas A.7, A.7 e A.9) que a proporção de não rejeição de H_0 esteve, em geral, entre 92% e 95% para os testes KS e CVM, já considerando o teste de AD , observamos um percentuais mais inferiores chegando até a zero, como por exemplo o cenário 6 (Tabela A.8)quando $n = 200$.

É importante citar que mesmo os dados sendo gerados de uma distribuição beta inflacionada, a hipótese nula de que os dados seguem distribuição gama unitária inflacionada, a proporção de não rejeição nos cenários de 1 a 6(Tabelas A.7 e A.8) foram altos($n = 30$, $n = 60$ e $n = 100$), com um percentual de não rejeição de até 90%.

Considerando que os dados foram gerados de uma distribuição KI, de acordo com os critérios, foi escolhida a distribuição KI com maior frequência. Percebeu-se que nos cenários 1, 2 e 3 (Tabela 3.4), com uma precisão maior ($\phi = 30$), os critérios indicaram a distribuição KI 99% das vezes quando o tamanho da amostra foi de 200.

Já sobre os testes de hipóteses (Tabelas A.10, A.11 e A.12), é possível verificar proporções de não rejeição altas quando a hipótese nula é de que os dados segue distribuição KI, com valores um pouco menor nos cenários com precisão maior. Por exemplo, ao observar a proporção de não rejeição da distribuição KI nos cenários 1, 2 e 3 (Tabela A.10), a proporção de não rejeição nos cenários com precisão menor ($\phi = 10$) é mais alta do que os nos cenários com precisão maior ($\phi = 10$). A mesma observação pode ser percebida na Tabela A.11.

É importante citar que a porcentagem de vezes hipótese nula foi não rejeitada considerando a distribuição BI e GUI foi zero na maioria dos cenários, com poucas exceções quando $n = 30$ e H_0 de a distribuição GUI e BI é adequada.

Considerando que os dados foram gerados de uma distribuição GUI, de acordo com os critérios, foi escolhida com maior frequência a própria distribuição inflacionada. Percebeu-se que no cenário 9 (Tabela 3.6) com uma precisão maior ($\phi = 50$), os critérios indicam a distribuição GUI $95,42\%$ das vezes quando o tamanho da amostra foi de 200.

No que se refere aos testes de hipóteses, considerando a hipótese nula H_0: os dados seguem distribuição GUI, com 5% de significância, é possível observar (Tabelas

A.13, A.14 e A.15) que a proporção de vezes que H_0 não foi rejeitado esteve, em geral, entre 94% e 95%, com exceção de alguns casos, como por exemplo os cenários 9(Tabela A.15), quando $n = 200$, porcentagem de inflacionamento de 10% e considerando o teste de KS. Considerando o teste de AD a porcentagem de não rejeição não foram altas como os de KS e CVM.

É importante citar que a proporção de vezes que H_0 foi não rejeitada, considerando a distribuição KI, foi zero na maioria dos cenários de 1 a 6, com poucas exceções quando $n = 30$ ou $n = 60$. Já a proporção de vezes que H_0 não foi rejeitada, considerando distribuição BI nos cenários de 1 a 6, foi alta,mas não superior à da distribuição GUI. Para os cenários de 7 a 9, quando considerado a KI na hipótese nula os resultados foram melhores que quando considerado a BI na hipótese nula.

Tabela 3.4: Resultados da proporção de escolha dos critérios para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim$ BI$[0,1]$			$y \sim$ KI$[0,1]$			$y \sim$ GUI$[0,1]$		
			% de escolha			% de escolha			% de escolha		
			BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$	BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$	BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$
1	10	30	7,18	39,68	53,14	13,06	77,69	9,25	9,77	30,84	59,39
		60	12,73	35,51	51,76	9,53	87,41	3,06	16,50	22,93	60,57
		100	19,42	32,19	48,39	6,32	93,11	0,57	23,67	16,62	59,71
		200	30,13	25,25	44,62	1,51	98,47	0,02	29,72	8,14	62,14
	30	30	13,45	31,03	55,52	23,91	76,09	0,00	15,03	25,42	59,55
		60	23,59	23,20	53,21	11,00	89,00	0,00	23,12	16,28	60,60
		100	31,04	16,74	52,22	5,33	94,64	0,00	31,59	9,60	58,81
		200	43,98	8,18	47,84	0,74	99,26	0,00	38,52	3,07	58,41
2	10	30	6,69	39,75	53,56	12,20	77,39	10,41	8,74	30,75	60,51
		60	12,09	36,21	51,70	9,21	87,25	3,54	16,30	23,28	60,42
		100	18,61	32,37	49,02	6,12	92,98	0,90	22,32	17,20	60,48
		200	29,73	25,72	44,68	1,48	98,45	0,07	28,72	8,45	62,83
	30	30	12,49	31,05	56,46	24,19	75,81	0,00	13,51	25,40	61,09
		60	22,79	23,57	53,64	11,29	88,71	0,00	22,24	16,54	61,22
		100	29,71	17,26	53,03	5,56	94,44	0,00	30,47	10,00	59,53
		200	43,63	8,44	47,93	0,73	99,27	0,00	38,57	2,90	58,53
3	10	30	5,73	40,03	54,24	11,37	77,43	11,20	7,87	30,71	61,42
		60	11,07	36,80	52,13	8,88	87,00	4,12	15,50	23,61	60,80
		100	17,33	33,22	49,45	6,25	92,50	1,25	22,15	17,34	60,51
		200	27,94	26,09	45,97	1,64	98,14	0,22	27,43	8,68	63,89
	30	30	11,13	30,83	58,04	24,43	75,57	0,00	11,33	25,89	62,78
		60	21,80	23,74	54,46	11,63	88,37	0,00	20,79	16,94	62,27
		100	28,13	17,68	54,19	5,32	94,68	0,00	28,82	10,79	60,39
		200	41,54	8,60	49,86	0,96	99,04	0,00	36,65	3,07	60,28

Tabela 3.5: Resultados da proporção de escolha dos critérios para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim$ BI$[0,1]$			$y \sim$ KI$[0,1]$			$y \sim$ GUI$[0,1]$		
			% de escolha			% de escolha			% de escolha		
			BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$	BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$	BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$
4	10	30	4,60	37,84	57,56	6,76	73,50	19,74	6,80	31,51	61,69
		60	7,67	34,91	57,42	9,52	82,64	7,84	14,31	24,89	60,80
		100	12,91	31,11	55,98	8,33	89,28	2,39	21,84	18,68	59,48
		200	43,63	8,44	47,93	3,36	96,42	0,22	28,80	10,62	60,58
	50	30	8,98	27,69	63,33	24,39	75,61	0,00	8,53	23,70	67,77
		60	18,96	19,72	61,32	11,19	88,81	0,00	18,54	14,87	66,59
		100	28,64	12,87	58,49	5,07	94,93	0,00	30,10	7,91	61,99
		200	39,75	5,26	54,99	2,12	97,88	0,00	36,54	2,12	61,34
5	10	30	3,62	38,62	57,76	5,49	73,11	21,40	5,60	32,01	62,39
		60	6,49	35,42	58,09	8,65	82,63	8,72	13,02	25,43	61,55
		100	11,50	31,66	56,84	8,13	89,39	2,48	19,92	19,57	60,51
		200	20,19	24,99	54,82	3,30	96,39	0,31	27,49	10,89	61,62
	50	30	6,74	28,62	64,64	24,23	75,77	0,00	6,39	24,71	68,90
		60	17,79	20,20	62,01	11,29	88,71	0,00	17,45	15,07	67,48
		100	25,99	13,71	60,30	5,30	94,70	0,00	27,14	8,58	64,28
		200	37,57	5,85	56,58	2,15	97,85	0,00	34,24	2,68	63,08
6	10	30	2,85	38,81	58,34	3,56	73,35	23,09	4,03	32,93	63,04
		60	5,12	35,74	59,14	7,85	82,42	9,73	11,41	25,59	63,00
		100	10,52	32,37	57,11	7,29	89,22	3,49	18,46	19,87	61,67
		200	17,73	26,05	56,22	3,63	96,03	0,34	25,50	11,77	62,73
	50	30	4,85	29,19	65,96	24,80	75,20	0,00	4,23	25,35	70,42
		60	15,52	20,48	64,00	11,81	88,19	0,00	14,06	15,74	70,20
		100	23,50	14,09	62,41	5,59	94,41	0,00	24,09	9,03	66,88
		200	35,52	5,78	58,70	1,02	98,98	0,00	32,61	2,53	64,86

Tabela 3.6: Resultados da proporção de escolha dos critérios para dados gerados com média $0,8$.

Cenários	ϕ	n	$y \sim$ BI$[0,1]$			$y \sim$ KI$[0,1]$			$y \sim$ GUI$[0,1]$		
			% de escolha			% de escolha			% de escolha		
			BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$	BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$	BI$[0,1]$	KI$[0,1]$	GUI$[0,1]$
7	10	30	0,63	40,33	59,04	2,14	56,20	41,66	4,30	32,04	63,66
		60	0,73	40,69	58,58	2,29	62,21	35,50	3,88	25,64	70,48
		100	1,04	39,44	59,52	3,07	66,47	30,46	4,83	19,93	75,24
		200	2,22	37,34	60,44	5,01	73,73	21,26	7,64	11,74	80,62
	50	30	4,31	32,24	63,45	10,16	77,40	12,44	5,51	24,37	70,12
		60	3,68	25,84	70,48	3,90	87,32	8,87	2,53	14,95	82,52
		100	4,74	20,29	74,48	1,52	93,53	4,95	1,56	9,02	89,42
		200	7,75	12,11	80,14	0,19	98,53	1,28	4,59	2,50	92,91
8	10	30	0,50	40,97	58,53	1,30	56,68	42,02	2,92	32,78	64,30
		60	0,68	40,38	58,94	1,84	61,58	36,58	2,19	25,93	64,30
		100	0,83	40,41	58,76	2,56	66,70	30,74	3,71	20,47	75,82
		200	1,91	37,80	60,29	4,57	74,37	21,06	6,74	11,93	81,33
	50	30	2,89	32,96	64,15	9,41	77,87	12,72	2,92	25,20	71,86
		60	2,29	26,20	71,51	4,08	86,83	9,09	1,34	15,09	83,57
		100	3,78	20,83	75,39	1,42	93,32	5,26	0,93	8,94	90,13
		200	6,92	12,16	80,92	0,14	98,32	1,54	3,46	2,50	94,04
9	10	30	0,38	40,55	59,07	0,88	55,31	43,81	1,45	33,13	65,42
		60	0,38	40,42	59,40	1,18	61,09	37,73	1,39	26,41	72,20
		100	0,69	39,39	59,92	2,03	66,03	31,94	2,54	20,62	76,84
		200	1,39	37,12	61,49	4,13	73,43	22,44	5,84	12,24	81,92
	50	30	1,42	33,28	65,30	9,17	76,63	14,20	1,63	25,33	73,04
		60	1,37	26,70	71,93	3,75	86,54	9,71	0,60	15,68	83,72
		100	2,53	20,95	76,52	1,60	92,33	6,07	0,47	9,14	90,39
		200	5,88	12,53	81,59	0,19	98,19	1,62	1,92	2,66	95,42

3.1.3 Inflacionamento em um

Para os cenários de 4 a 9, quando os dados são geradas pela distribuição BI, de acordo com os critérios, foi escolhida a distribuição GUI com maior frequência. Percebeu-se que no cenário 9 (Tabela 3.8), com uma precisão maior ($\phi = 50$), os critérios indicaram a distribuição GUI $80,84\%$ das vezes quando o tamanho das amostras foram de 200.

No que se refere aos testes de hipóteses, considerando a hipótese nula H_0: os dados seguem distribuição BI, 5% de significância, é possível observar (Tabelas A.16 e A.17) que a proporção de não rejeição de H_0 esteve, em geral, entre 89% e 95% para os testes KS e CVM, com exceção do cenário 6 com $n = 100$ e $n = 200$ para o teste de KS. Considerando o teste de AD , observamos percentuais mais inferiores chegando até a zero, como por exemplo o cenário 6 (Tabela A.16) quando $n = 200$.

Considerando que os dados foram gerados de uma distribuição KI,de acordo com os critérios, foi escolhida a distribuição KI com maior frequência. Percebeu-se que nos cenários 6, 7, 8 e 9 (Tabela 3.7 e 3.8), com uma precisão maior ($\phi = 50$), os critérios indicaram a distribuição KI cerca de 99% das vezes quando o tamanho da amostra foi de 200.

Já sobre os testes de hipóteses (Tabelas A.18 e A.19), é possível verificar proporções de não rejeição altas quando a hipótese nula é de que os dados segue distribuição KI, com valores um pouco menor nos cenários com precisão maior. Por exemplo, ao observar a proporção de não rejeição da distribuição KI nos cenários 4, 5 e 6 (Tabela A.18), a proporção de não rejeição nos cenários com precisão menor ($\phi = 10$) é mais alta do que os nos cenários com precisão maior ($\phi = 10$). A mesma observação pode ser percebida na Tabela A.19.

É importante citar que a porcentagem de vezes hipótese nula foi não rejeitada considerando a distribuição BI e GUI foi zero na maioria dos cenários, com algumas exceções quando $n = 30$ ou $n = 60$.

Considerando que os dados foram gerados de uma distribuição GUI, de acordo com os critérios, foi escolhida a própria distribuição inflacionada com maior frequência. Percebeu-se que no cenário 9 (Tabela 3.8) com uma precisão maior ($\phi = 50$), os critérios indicam a distribuição GUI $94,34\%$ das vezes quando o tamanho da amostra foi de 200.

No que se refere aos testes de hipóteses, considerando a hipótese nula H_0: os dados seguem distribuição GUI, com 5% de significância, é possível observar (Tabelas A.20 e A.21) que a proporção de vezes que H_0 não foi rejeitado esteve, em geral, entre 89% e 95%, com exceção de alguns casos, como por exemplo os cenários 6 e 9(Tabelas A.20 e A.21), quando $n = 200$ ou $n = 100$, porcentagem de inflacionamento de 10% e considerando o teste de KS. Considerando o teste de AD a porcentagem de não

rejeição não foram altas como os de KS e CVM.

É importante citar que a proporção de vezes que H_0 foi não rejeitada, considerando a distribuição KI, foi zero na maioria dos cenários de 4 a 9, com algumas exceções quando $n = 30$ ou $n = 60$. Já a proporção de vezes que H_0 não foi rejeitada, considerando distribuição BI nos cenários de 4 a 6, foram resultados altos quando $n = 30$, mas não superior à da distribuição GUI.

Tabela 3.7: Resultados da proporção de escolha dos critérios para dados gerados com média $0,5$.

Cenários	ϕ	n	gerado pela BI$(0,1]$			$y \sim$ KI$(0,1]$			$y \sim$ GUI$(0,1]$		
			% de escolha			% de escolha			% de escolha		
			BI$(0,1]$	KI$(0,1]$	GUI$(0,1]$	BI$(0,1]$	KI$(0,1]$	GUI$(0,1]$	BI$(0,1]$	KI$(0,1]$	GUI$(0,1]$
4	10	30	4,59	38,44	56,97	5,63	73,52	20,85	7,15	31,90	60,95
		60	7,60	34,11	58,29	8,58	82,79	8,63	12,64	24,71	62,65
		100	11,70	31,38	56,92	6,66	89,27	4,07	19,18	19,31	61,51
		200	16,99	25,47	57,54	3,37	96,30	0,33	27,84	10,56	61,60
	50	30	8,80	28,24	62,96	24,01	75,99	0,00	8,11	24,05	67,84
		60	18,12	19,45	62,43	11,15	88,85	0,00	16,41	14,76	68,83
		100	25,44	13,21	61,35	5,14	94,93	0,00	23,76	8,75	67,49
		200	34,13	5,33	60,54	2,09	97,91	0,00	35,20	2,17	62,63
5	10	30	3,97	38,38	57,65	4,56	73,07	22,37	6,24	31,76	62,00
		60	7,45	34,18	58,37	7,64	82,37	9,99	11,88	24,43	63,69
		100	10,20	31,92	57,88	5,88	89,09	5,03	17,62	18,99	63,39
		200	15,58	25,63	58,79	3,13	96,41	0,46	27,45	10,44	62,11
	50	30	6,91	28,41	64,68	24,20	75,80	0,00	6,74	24,30	68,98
		60	17,74	19,42	62,84	11,64	88,36	0,00	14,83	14,77	70,40
		100	24,32	13,12	62,56	5,44	94,56	0,00	21,03	8,43	70,54
		200	32,27	5,39	62,34	2,22	97,78	0,00	34,71	2,36	62,93
6	10	30	3,36	38,87	57,77	3,28	72,50	24,22	5,21	32,31	62,45
		60	6,66	35,44	57,90	6,33	81,96	11,71	9,62	25,72	64,66
		100	8,65	32,65	58,70	5,45	88,75	5,80	14,27	19,89	65,84
		200	12,81	26,33	60,86	2,91	96,33	0,76	24,90	11,26	63,84
	50	30	5,86	28,63	65,51	24,44	75,56	0,00	5,67	24,59	69,74
		60	15,65	20,10	64,25	11,76	88,24	0,00	13,49	15,38	71,13
		100	20,89	13,66	65,45	5,38	94,62	0,00	18,39	8,93	72,68
		200	29,34	5,65	65,01	1,27	98,73	0,00	32,25	2,53	65,22

Tabela 3.8: Resultados da proporção de escolha dos critérios para dados gerados com média $0,8$.

Cenários	ϕ	n	$y \sim$ BI$(0,1]$			$y \sim$ KI$(0,1]$			$y \sim$ GUI$(0,1]$		
			% de escolha			% de escolha			% de escolha		
			BI$(0,1]$	KI$(0,1]$	GUI$(0,1]$	BI$(0,1]$	KI$(0,1]$	GUI$(0,1]$	BI$(0,1]$	KI$(0,1]$	GUI$(0,1]$
7	10	30	0,78	40,67	58,55	2,25	56,34	41,41	4,55	32,31	63,14
		60	0,97	38,83	60,20	2,37	60,64	36,99	4,01	25,43	70,56
		100	1,20	38,83	59,96	2,80	65,98	31,22	5,78	20,26	73,96
		200	2,24	36,68	61,08	4,93	73,79	21,28	9,54	11,73	78,73
	50	30	4,65	32,19	63,16	10,06	77,47	12,47	5,76	24,23	70,01
		60	4,06	25,26	70,68	3,82	87,25	8,93	3,21	15,00	81,79
		100	5,85	20,04	74,11	1,74	93,00	5,26	1,97	8,97	89,06
		200	9,90	11,52	78,58	0,13	98,42	1,45	7,52	2,31	90,17
8	10	30	0,54	40,28	58,18	1,61	56,00	42,39	3,22	32,18	64,60
		60	0,72	39,14	60,14	1,90	61,09	37,01	3,09	25,15	71,76
		100	0,98	39,11	59,91	2,45	66,03	3,52	4,58	20,02	75,40
		200	1,77	36,82	61,41	3,92	73,77	22,31	8,74	11,59	79,67
	50	30	3,35	31,97	64,68	9,40	77,28	13,32	3,14	24,48	72,38
		60	3,07	24,87	72,06	3,92	87,01	9,07	1,92	15,02	83,06
		100	4,74	19,60	75,66	1,56	92,91	5,53	1,35	8,63	90,02
		200	8,99	11,22	79,79	0,15	98,23	1,62	4,85	2,62	92,53
9	10	30	0,42	40,28	59,30	0,84	55,82	43,34	1,66	32,69	65,65
		60	0,48	39,54	54,98	1,39	60,75	37,86	2,06	26,45	71,49
		100	0,72	38,18	61,10	2,01	66,14	31,85	3,28	20,89	75,83
		200	1,73	36,27	62,00	2,68	73,99	23,33	7,24	12,51	80,25
	50	30	1,72	32,30	65,98	9,38	76,38	14,24	1,69	24,68	73,63
		60	1,91	25,74	72,35	3,82	86,37	9,81	1,30	15,58	83,12
		100	3,27	20,16	76,57	1,66	92,77	5,57	0,97	9,16	89,87
		200	7,34	11,82	80,84	0,12	98,32	1,56	2,97	2,69	94,34

Capítulo 4

Aplicações das Distribuições Inflacionadas

4.1 Introdução

Neste Capítulo serão apresentadas sete aplicações a dados reais: nas seções 4.2, 4.3 e 4.4 são apresentadas aplicações para dados Inflacionados em zero, nas seções 4.5 e 4.6 são apresentadas aplicações para dados Inflacionados em zero e um, nas seções 4.7 e 4.8 são apresentadas aplicações para dados Inflacionados em um.

As tabelas de medidas descritivas da variável de interesse apresentam as seguintes informações: o valor mínimo, $Q_{1/4}$, mediana, média, $Q_{3/4}$, valor máximo, DP e CV, em que $Q_{1/4}$ é o primeiro quartil, $Q_{3/4}$ é o terceiro quartil, DP é o desvio padrão e CV que é o coeficiente de variação.

Com o intuito de avaliar a adequabilidade das distribuições aos dados, foi realizada uma comparação a partir das medidas de bondade de ajuste: critério de informação de Akaike (AIC), critério de informação de Akaike corrigido (AICc), critério de informação Bayesiano (BIC), critério de informação Bayesiano corrigido (BICc), critério de Hannan-Quinn (HQ) e critério de Hannan-Quinn corrigido (HQc). Estes resultados foram sumarizados nas tabelas de critérios de adequabilidade de cada aplicação.

4.2 Proporção de pessoas que usa internet

A variável de interesse (*Internet*) é a proporção de pessoas por país que utilizaram internet no ano de 2000 em 209 países (`https://ourworldindata.org/`). Vale ressaltar que, de acordo com os dados, há 17 nações que apresentaram valor igual a zero, ou seja, apresentando 8,13% de inflacionamento em zero. De acordo com a Tabela 4.1, a proporção média de pessoas que utilizaram internet é de $0,0647$.

Tabela 4.1: Medidas descritivas da proporção de pessoas que utilizaram internet.

Mínimo	$Q_{1/4}$	Mediana	Média	$Q_{3/4}$	Máximo	DP	CV
0,0000	0,0020	0,0140	0,0647	0,0650	0,5130	0,1096	169,25%

A Tabela 4.2 apresenta os p-valores dos testes de hipóteses realizados para verificar a adequação dos dados às distribuições abordadas, a partir dos seguintes testes de hipótese: Kolmogorov–Smirnov (KS), Anderson–Darling (AD) e T Cramer–Von Mises (CVM). Ao nível de significância de 5%, os testes de Kolmogorov-Smirnov (KS) e Cramér-Von Mises(CVM) não rejeitam a hipótese que os dados seguem distribuição GUI e, ao mesmo nível de significância, apenas o teste de CVM não rejeitou a hipótese que os dados seguem distribuição KI. A Figura 4.1 apresenta o histograma, em que a barra vertical representa a frequência de zeros. Já a Figura 4.2 apresenta os gráficos qq-plot das três distribuições ajustadas, em que o eixo x apresenta as probabilidades teóricas e o eixo y apresenta as probabilidades empíricas. Observando os gráficos, vemos que as distribuições Beta Inflacionada(BI) e Kumaraswamy Inflacionada (KI) tiverem desempenho semelhantes, na Gama unitária Inflacionada (GUI), os pontos ficaram próximos da diagonal, mas para o intervalo de 0,4 a 0,8 os pontos ficaram um pouco afastados da diagonal.

Tabela 4.2: p-valores dos testes de hipóteses.

	Testes		
Distribuição	KS	CVM	AD
BI	0,0033	0,0469	0,0198
KI	0,0292	0,0699	0,0332
GUI	0,0877	0,1098	0,0473

A tabela 4.3 apresenta os resultados sumarizados dos critérios de adequabilidade. Para todas as medidas analisadas, há uma melhor adequabilidade da distribuição Gama unitária inflacionada quando comparadas com as outras medidas.

Tabela 4.3: Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - dados da proporção de pessoas utilizaram a internet em 209 países.

Distribuição	AIC	AICc	BIC	BICc	HQ	HQc
BI	$-609,094$	$-608,976$	$-599,067$	$-598,754$	$-605,040$	$-604,843$
KI	$-621,409$	$-621,292$	$-611,382$	$-611,069$	$-617,355$	$-617,159$
GUI	$-630,198$	$-630,081$	$-620,171$	$-619,859$	$-626,144$	$-625,948$

Desta forma, considerando os resultados dos critérios e dos testes de hipóteses, podemos concluir que a distribuição que melhor representa os dados da proporção de pessoas que utilizaram internet é a distribuição gama unitária inflacionada.

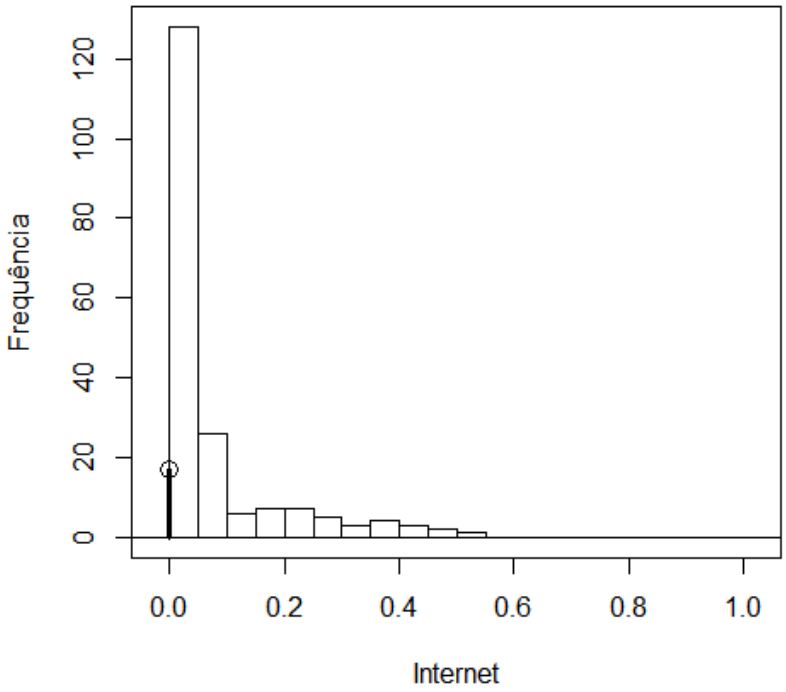

Figura 4.1: Histograma da proporção de pessoas dos países que utilizaram internet em 209 países.

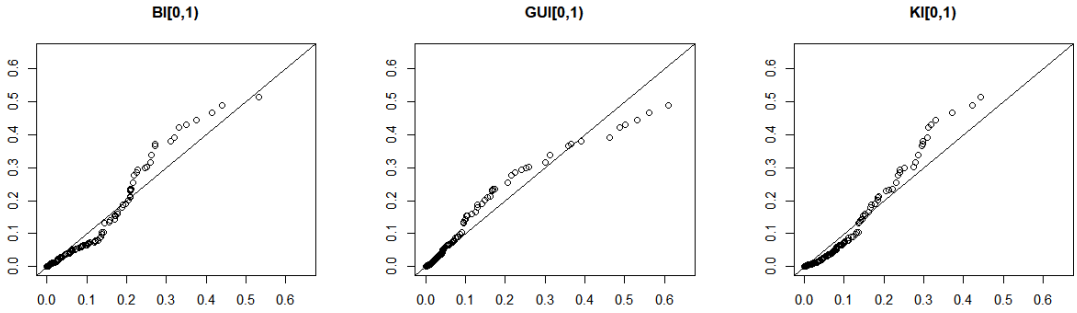

Figura 4.2: QQ-Plot da proporção de pessoas que utilizaram internet em 209 países.

4.3 Proporção de nascidos vivos com 4000 gramas ou mais

A variável de interesse (*Nascidos vivos*) é a proporção de crianças do estado da Paraíba que nasceram vivas com 4000 gramas ou mais, no ano de 2021 em 223 municípios(`https : //datasus.saude.gov.br/`). Vale ressaltar que, de acordo com os dados, há 13 municípios que apresenta valor igual a zero, ou seja, apresentando

5,83% de inflacionamento em zero. De acordo com a Tabela 4.4, a proporção média de crianças que nasceram vivas com 4000 gramas ou mais é de 0,0625.

Tabela 4.4: Medidas descritivas da proporção de nascidos vivos com 4000 gramas ou mais.

Mínimo	$Q_{1/4}$	Mediana	Média	$Q_{3/4}$	Máximo	DP	CV
0,0000	0,0430	0,0625	0,0625	0,0827	0,1429	0,0309	49,40%

A tabela 4.5 apresenta os p-valores dos teste de hipóteses realizados para verificar a adequação dos dados às distribuições abordadas, a partir dos seguintes testes de hipótese: Kolmogorov–Smirnov (KS), Anderson–Darling (AD) e Cramér–Von Mises (CVM). Ao nível de significância de 5%, os três testes não rejeitaram a hipótese de que os dados seguem distribuição GUI e KI, o teste de KS foi o único que não rejeitou a hipótese de que os dados seguem distribuição BI. A Figura 4.3 apresenta o histograma, em que a barra vertical representa a frequência de zeros. A Figura 4.4 apresenta os gráficos qq-plot das três distribuições ajustadas, em que o eixo x apresenta as probabilidades teóricas e o eixo y apresenta as probabilidades empíricas. Podemos observar que para essa aplicação a distribuição BI teve pontos mais próximos da linha vertical que as demais distribuições.

Tabela 4.5: P-valores dos testes de hipóteses.

	Testes		
Distribuição	KS	CVM	AD
BI	0,0570	0,0431	0,0399
KI	0,4347	0,7899	0,5458
GUI	0,4347	0,5934	0,4567

A Tabela 4.6 apresenta os resultados sumarizados dos critérios de adequabilidade. Para todas medidas analisadas, há uma melhor adequabilidade da distribuição Beta inflacionada quando comparadas com as outras medidas.

Desta forma, considerando os resultados dos critérios e testes de hipóteses, podemos concluir que a distribuição que melhor representa os dados da proporção de crianças que nasceram vivas com 4000 gramas ou mais é a distribuição beta inflacionada.

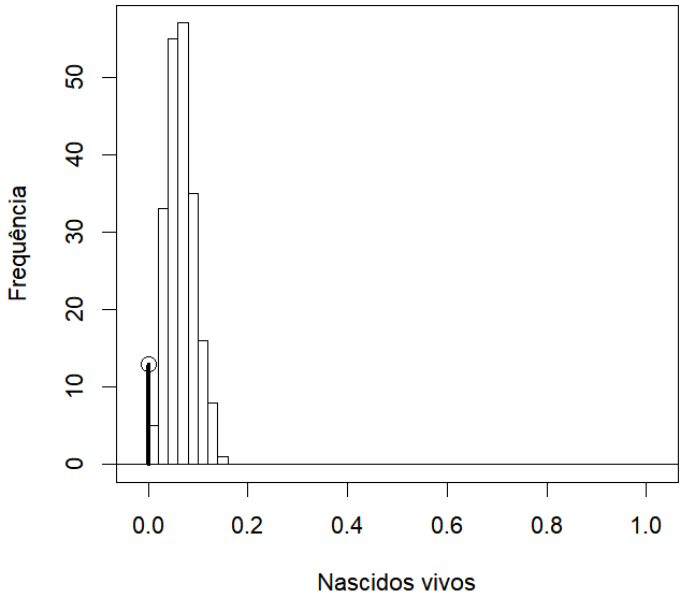

Figura 4.3: Histograma da proporção de nascidos com 4000 gramas ou mais em 223 municípios.

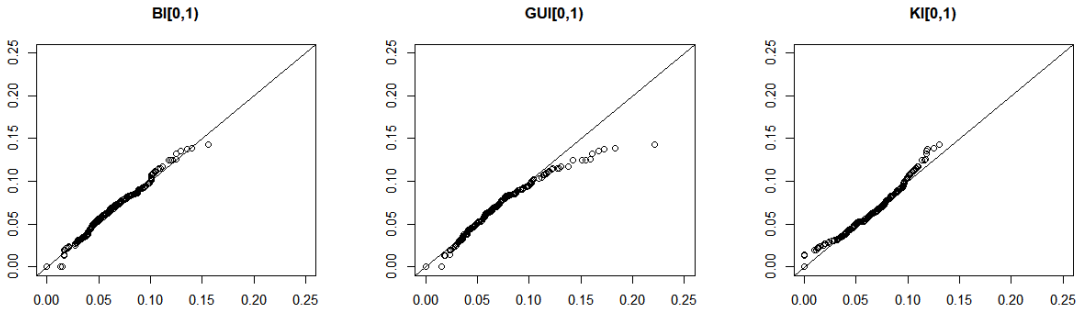

Figura 4.4: QQ-Plot da proporção de nascidos vivos com 4000 gramas ou mais em 223 municípios.

Tabela 4.6: Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - dados da proporção de nascidos com 4000 gramas ou mais nos 223 municípios.

Distribuição	AIC	AICc	BIC	BICc	HQ	HQc
BI	$-820,922$	$-820,813$	$-810,701$	$-810,405$	$-816,796$	$-816,611$
KI	$-820,831$	$-820,722$	-810.610	$-810,313$	$-816,705$	$-816,520$
GUI	$-817,609$	$-817,499$	-807.387	-807.091	$-813,483$	$-813,298$

4.4 Proporção de pacientes com tuberculose e HIV

A variável de interesse (*Pacientes*) é a proporção pacientes por país que além de ter a Tuberculose, também têm HIV no ano de 2005 em 178 países(`https://ourworldindata.org/`). Vale ressaltar que, de acordo com os dados, há 13 nações que apresenta valor igual a zero, ou seja apresentando 5,06% de inflacionamento em zero. De acordo com a Tabela 4.7, a proporção média de pacientes que têm tuberculose e também têm HIV é de 0,1365.

Tabela 4.7: Medidas descritivas da proporção de pacientes com tuberculose e HIV.

Mínimo	$Q_{1/4}$	Mediana	Média	$Q_{3/4}$	Máximo	DP	CV
0,0000	0,0095	0,0535	0,1365	0,1975	0,7900	0,1876	137,49%

A Tabela 4.8 apresenta os resultados dos p-valores dos testes de hipóteses realizados para verificar a adequação dos dados às distribuições abordadas, a partir dos seguintes testes de hipótese: Kolmogorov–Smirnov (KS), Anderson–Darling (AD) e Cramér–Von Mises (CVM). Ao nível de significância de 5%, as três distribuições não são rejeitadas a hipótese de os dados seguem as distribuições BI, KI e GUI. A Figura 4.5 apresenta o histograma, em que a barra vertical representa a frequência de zeros. A Figura 4.6 apresenta os gráficos qq-plot das três distribuições ajustadas, em que o eixo x apresenta as probabilidades teóricas e o eixo y apresenta as probabilidades empíricas. Podemos ver que para essa aplicação nenhuma das três distribuições foi boa de modo geral para os gráficos qq-plot.

Tabela 4.8: P-valores dos testes de hipóteses.

	Testes		
Distribuição	KS	CVM	AD
BI	0,370	0,332	0,228
KI	0,326	0,305	0,265
GUI	0,502	0,427	0,349

A Tabela 4.9 apresenta os resultados sumarizados dos critérios de adequabilidade. Para todas as medidas analisadas, há uma melhor adequabilidade da distribuição Gama Unitária inflacionada quando comparadas com as outras medidas.

Diante do que foi apresentado e a partir dos resultados dos critérios e testes de hipóteses, podemos concluir que a distribuição que melhor representa os dados da proporção de pacientes com tuberculose e HIV é a gama unitária inflacionada.

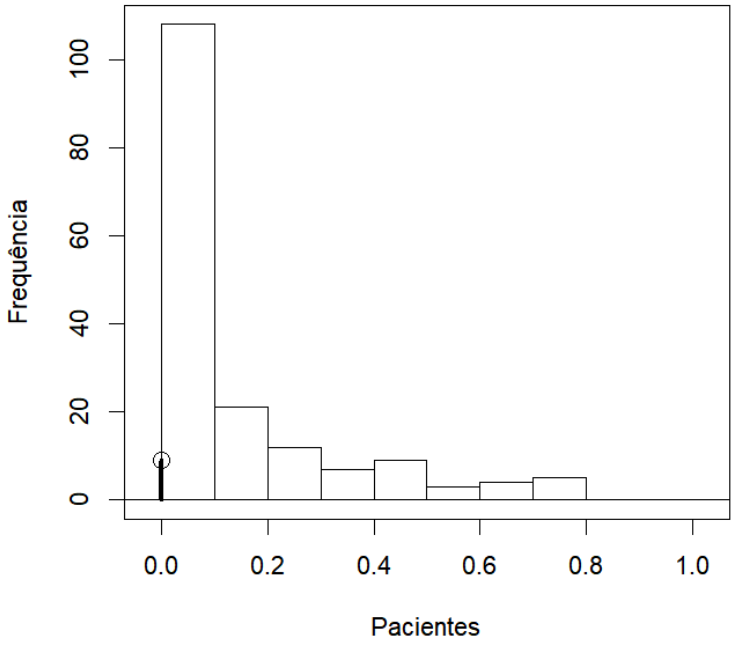

Figura 4.5: Histograma da proporção de pacientes com tuberculose e HIV em 178 países.

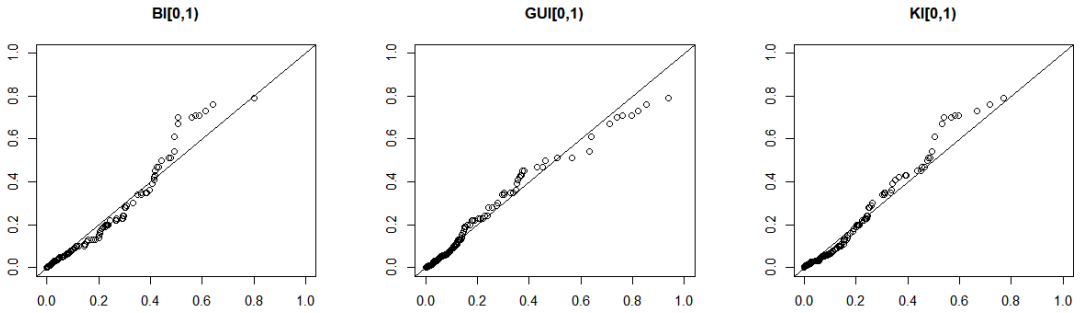

Figura 4.6: QQ-Plot da proporção de pacientes com tuberculose e HIV em 178 países.

Tabela 4.9: Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pacientes com tuberculose e HIV em 178 países.

Distribuição	AIC	AICc	BIC	BICc	HQ	HQc
BI	$-285,380$	$-285,242$	$-275,835$	$-275,477$	$-281,509$	$-281,282$
KI	$-290,143$	$-290,005$	$-280,598$	$-280,241$	$-286,272$	$-286,045$
GUI	$-293,642$	$-293,504$	$-284,096$	$-283,739$	$-289,771$	$-289,544$

4.5 Proporção da população urbana que residente em domicílios com rede de esgoto sanitário nos municípios do estado do Espírito Santo

A variável de interesse (*Esgoto sanitrio ES*) é a proporção de pessoas que reside em zona urbana com domicílio ligado à rede de esgoto sanitário no ano de 2013 em 52 municípios do Espírito Santo (`http://www.atlasbrasil.org.br/`). É importante mencionar que, de acordo com os dados, há 1 município que apresentou valor igual a zero e 4 municípios apresentou valor igual a um, ou seja, apresentando 1,92% de inflacionamento em zero e 7,69% de inflacionamento em um. De acordo com Tabela 4.10 a proporção média de pessoas que residem em domicílios com rede de esgoto sanitário é de 0,5987.

Tabela 4.10: Medidas descritivas da proporção de pessoas que residem em domicílios com rede de esgoto sanitário.

Mínimo	$Q_{1/4}$	Mediana	Média	$Q_{3/4}$	Máximo	DP	CV
0,0000	0,4078	0,5985	0,5987	0,8790	1,0000	0,3034	50,67%

A Tabela 4.11 apresenta os p-valores dos testes de hipóteses realizados para verificar a adequação dos dados às distribuições abordadas, a partir dos seguintes testes de hipóteses: Kolmogorov–Smirnov (KS), Anderson–Darling (AD) e Cramér–Von Mises (CVM). Ao nível de significância de 5%, o teste de AD rejeitou a hipótese que os dados seguem distribuição BI, KI ou GUI, já para os testes KS e CVM não foi rejeitada a hipótese de que os dados seguem a distribuição BI, KI e GUI. A Figura 4.7 apresenta o histograma, em que as barras verticais representam a frequência de zero e a frequência de uns. A Figura 4.8 apresenta os gráficos qq-plot das três distribuições ajustadas, em que o eixo x apresenta as probabilidades teóricas e o eixo y apresentam as probabilidades empíricas. Podemos observar que de os resultados destes gráficos não foram tão bom, pois para essa aplicação nas três distribuições os valores ficaram um pouco afastado da reta diagonal dos gráficos qq-plot.

Tabela 4.11: P-valores dos testes de hipóteses.

	Testes		
Distribuição	KS	CVM	AD
BI	0,9180	0,9406	$< 1\text{x}10^{-05}$
KI	0,9180	0,9141	$< 1\text{x}10^{-05}$
GUI	0,9180	0,9150	$< 1\text{x}10^{-05}$

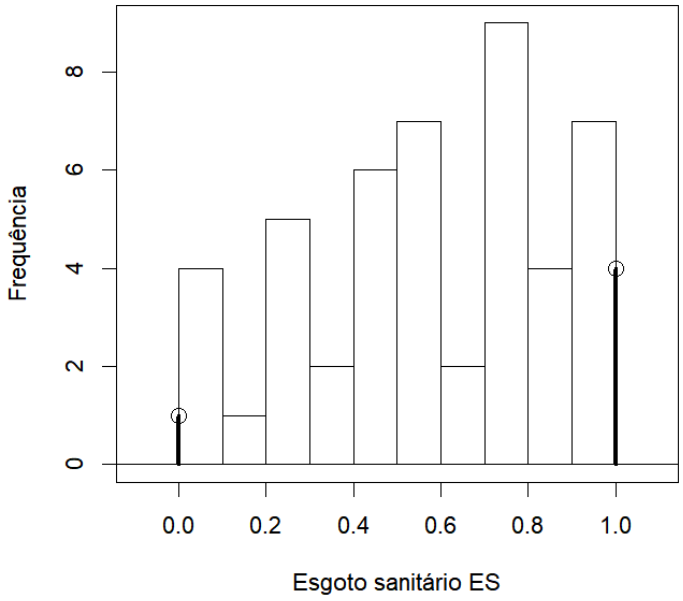

Figura 4.7: Histograma da proporção de pessoas que residem em domicílio com rede esgoto sanitário em 52 municípios.

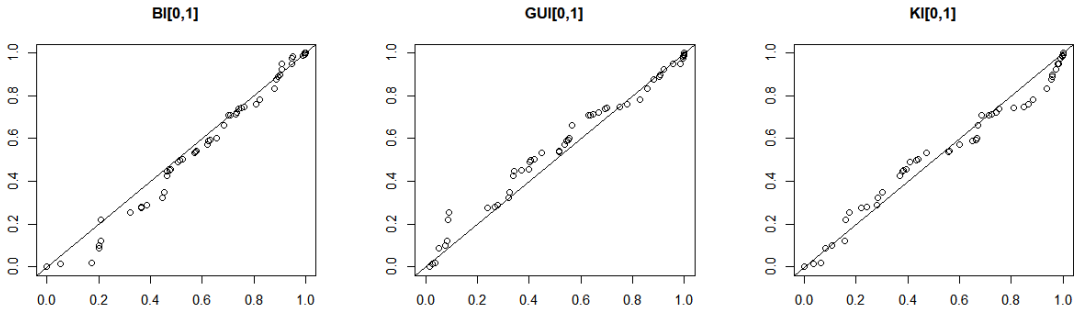

Figura 4.8: QQ-Plot da proporção de pessoas que residem em domicílio com rede de esgoto sanitário em 52 municípios.

A Tabela 4.12 apresenta os resultados sumarizados dos critérios de adequabilidade. Para todas as medidas analisadas, há uma melhor adequabilidade da distribuição gama unitária inflacionada quando comparada com as outras medidas.

Desta forma, considerando os resultados dos critérios e testes de hipóteses, podemos concluir que a distribuição que melhor representa os dados da proporção de pessoas que reside em domicílios urbanos com acesso a rede de esgoto sanitário é a distribuição gama unitária inflacionada.

Tabela 4.12: Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pessoas que reside em domicílio com rede de esgoto sanitário em 52 municípios.

Distribuição	AIC	AICc	BIC	BICc	HQ	HQc
BI	42,790	43,641	50,595	52,276	45,782	46,951
KI	42,358	43,209	50,163	51,845	45,350	46,520
GUI	42,315	43,166	50,119	51,801	45,307	46,476

4.6 Proporção de população urbana residente em domicílios ligados a rede de esgoto sanitário nos municípios do estado do Rio Grande do Norte

A variável de interesse(*Esgoto sanitrio RN*) é a proporção de pessoas que residem em zona urbana com domicílio conectado à rede de esgoto sanitário no ano de 2016 em 55 municípios do Rio Grande do Norte(`http://www.atlasbrasil.org.br/`). Vale ressaltar que, de acordo com os dados, um município que apresenta valor igual a zero e sete municípios com valor igual a um, ou seja, apresentando 1,82% de inflacionamento em zero e 12,73% de inflacionamento em um. De acordo com a Tabela 4.13, a proporção média de pessoas que residem em domicílios ligados a rede de esgoto sanitário é de 0,5471.

Tabela 4.13: Medidas descritivas da proporção de pessoas que residem em domicílios com rede de esgoto sanitário.

Mínimo	$Q_{1/4}$	Mediana	Média	$Q_{3/4}$	Máximo	DP	CV
0,0000	0,2639	0,4994	0,5471	0,8917	1,0000	0,3411	62,34%

A Tabela 4.14 apresenta os p-valores referentes aos testes de hipóteses realizados para verificar a adequação dos dados às distribuições abordadas, a partir dos seguintes testes de hipótese: Kolmogorov–Smirnov (KS), Anderson–Darling (AD) e Cramér–Von Mises (CVM). Ao nível de significância de 5%, o teste de AD rejeitou a hipótese que os dados seguem a distribuição distribuição BI, KI, ou GUI, já para os testes de KS e CVM não foi rejeitada a hipótese de que os dados seguem a distribuição BI, KI e GUI. A Figura 4.9 apresenta o histograma, em que as barras verticais representam a frequência de zero e a frequência de uns. A Figura 4.10 apresenta os gráficos qq-plot das três distribuições ajustadas, em que o eixo x apresenta as probabilidades teóricas e o eixo y apresenta as probabilidades empíricas. Podemos ver que os resultados destes gráficos não foram tão bom, assim como também podemos observar que o gráfico qq-plot da distribuição beta inflacionada foi um pouco pior que os demais pelo fato dos pontos estarem mais afastado da diagonal em comparação com os outros dois gráficos.

Tabela 4.14: P-valores dos testes de hipóteses

	Testes		
Distribuição	KS	CVM	AD
BI	0,3351	0,3936	$< 1,1\text{x}10^{-05}$
KI	0,3351	0,7669	$< 1,1\text{x}10^{-05}$
GUI	0,3351	0,7671	$< 1,1\text{x}10^{-05}$

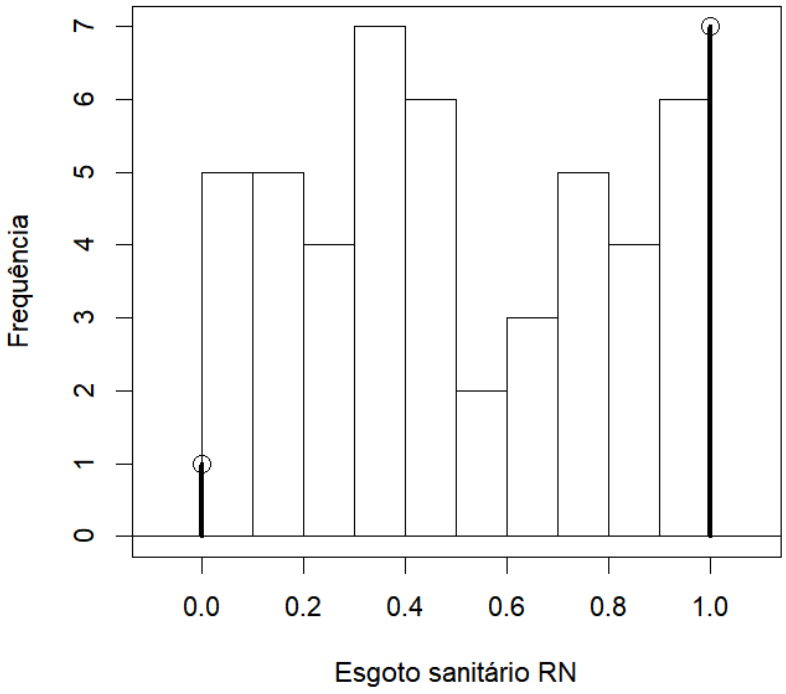

Figura 4.9: Histograma da proporção da população que residem em domicílio com rede esgoto sanitário em 55 municípios.

A Tabela 4.15 os resultados sumarizados dos critérios de adequabilidade. Para todas a medidas analisadas, há uma melhor adequabilidade da distribuição kumaraswamy inflacionada comparadas com as outras medidas.

Diante do que que foi apresentado e a partir dos resultados dos critérios e testes de hipóteses podemos concluir que a distribuição que melhor representa os dados da proporção de pessoas que residem em domicílios ligados a rede de esgoto é a distribuição kumaraswamy inflacionada.

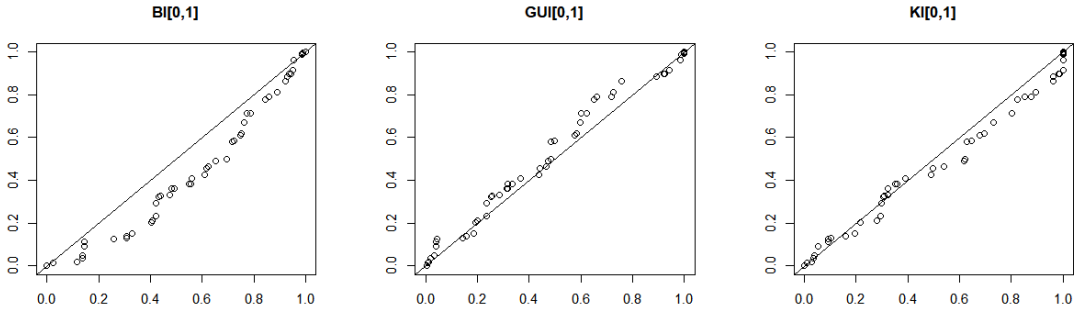

Figura 4.10: QQ-Plot da porcentagem da população que residem em domicílios com rede de esgoto sanitário em 55 municípios.

Tabela 4.15: Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pessoas que residem em domicílio com rede de esgoto sanitário em 55 municípios.

Distribuição	AIC	AICc	BIC	BICc	HQ	HQc
BI	58,430	59,230	66,459	68,062	61,535	62,645
KI	58,179	58,979	66,209	67,812	61,285	62,395
GUI	58,281	59,081	66,311	67,914	61,386	62,497

4.7 Proporção de votos para o candidato Jair Bolsonaro por seções eleitorais da cidade de Chaves–PA

A variável de interesse(*Eleio*) é a proporção de votos que o candidato Jair Bolsonaro obteve nas seções eleitorais do município de Chaves-PA no segundo turno das eleições no ano de 2022 em 53 seções (`https://www.tse.jus.br/`). Vale ressaltar que, de acordo com os dados, há uma seção que apresentou valor igual a um, ou seja, apresentando 1,89% inflacionamento em um. De acordo com a Tabela 4.16 a proporção média de votos para candidato em questão nas seções do 2º turno é de 0,3274. Mesmo com o inflacionamento em um, a média ficou bem abaixo de 0,5, o que se deve ao fato do inflacionamento ser abaixo de 2% e o valor mínimo dos dados é 0,0833.

Tabela 4.16: Medidas descritivas da proporção de votos para o candidato Jair Bolsonaro.

Mínimo	$Q_{1/4}$	Mediana	Média	$Q_{3/4}$	Máximo	DP	CV
0,0833	0,1829	0,2749	0,3274	0,4148	1,0000	0,1912	58,40%

A Tabela 4.17 apresenta os p-valores referentes aos testes de hipóteses realizados para verificar a adequação dos dados às distribuições abordadas, a partir dos

seguintes testes de hipótese: Kolmogorov–Smirnov (KS), Anderson–Darling (AD) e Cramér–Von Mises (CVM). Ao nível de significância de 5%, o teste de Anderson-Darling (AD) rejeitou a hipótese de que os dados seguem distribuição BI, KI ou GUI, já para os outros dois testes (KS e CVM), não rejeitaram a hipótese de que os dados seguem as distribuições BI, KI e GUI. A Figura 4.11 apresenta o histograma, em que a barra vertical representa a frequência de um. Observando o histograma, é possível entender o motivo em que mesmo com inflacionamento em 1, esses dados teve uma média abaixo de 0,5, os dados estão mais concentrados no intervalo de 0,1 a 0,5. A Figura 4.12 apresenta os gráficos qq-plot das três distribuições ajustadas, em que o eixo x apresenta as probabilidades teóricas e o eixo y apresenta as probabilidades empíricas. Podemos observar que a partir dos gráficos, o da GUI teve um resultado um pouco melhor com exceção dos últimos três pontos.

Tabela 4.17: P-valores dos testes de hipóteses.

	Testes		
Distribuição	KS	CVM	AD
BI	0,4067	0,2686	$< 1,2\text{x}10^{-05}$
KI	0,8210	0,5345	$< 1,2\text{x}10^{-05}$
GUI	0,7654	0,7044	$< 1,2\text{x}10^{-05}$

A Tabela 4.18 apresenta os resultados sumarizados dos critérios de adequabilidade. Para todas medidas analisadas, há uma melhor adequabilidade da distribuição Gama Unitária Inflacionada quando comparadas com as outras medidas.

Desta forma, considerando os resultados dos critérios e testes de hipóteses, podemos concluir que a distribuição que melhor representa os dados da proporção de votos para o candidato Jair Bolsonaro por seções eleitorais do segundo turno na cidade de Chaves–PA é a distribuição gama unitária inflacionada.

Tabela 4.18: Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de votos para o candidato Jair Bolsonaro em 53 seções eleitorais.

Distribuição	AIC	AICc	BIC	BICc	HQ	HQc
BI	$-31,1810$	$-30,691$	$-25,270$	$-24,298$	$-28,908$	$-28,233$
KI	$-28,745$	$-28,255$	$-22,834$	$-21,862$	$-26,472$	$-25,797$
GUI	$-32,396$	$-31,906$	$-26,485$	$-25,512$	$-30,123$	$-29,447$

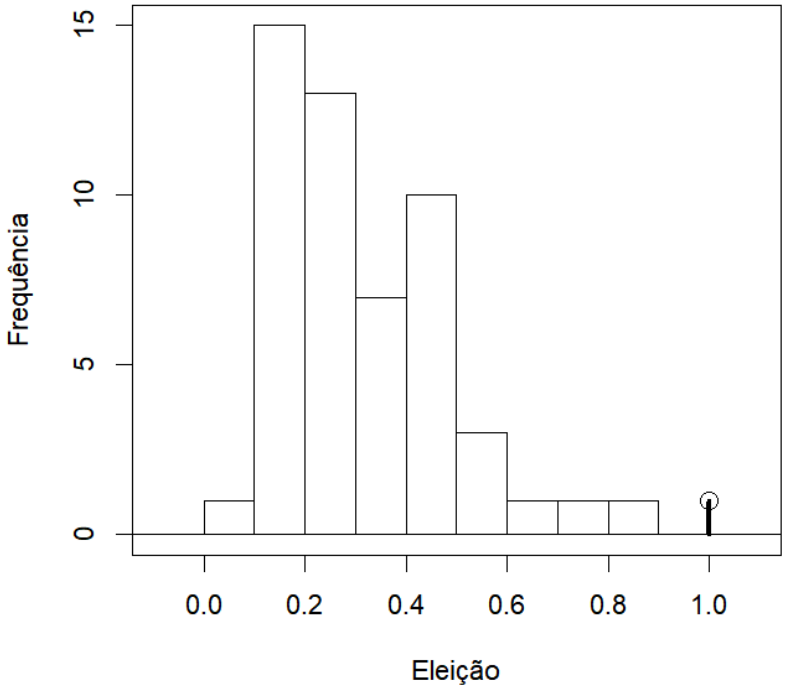

Figura 4.11: Histograma da proporção de votos para Bolsonaro em 53 seções eleitorais.

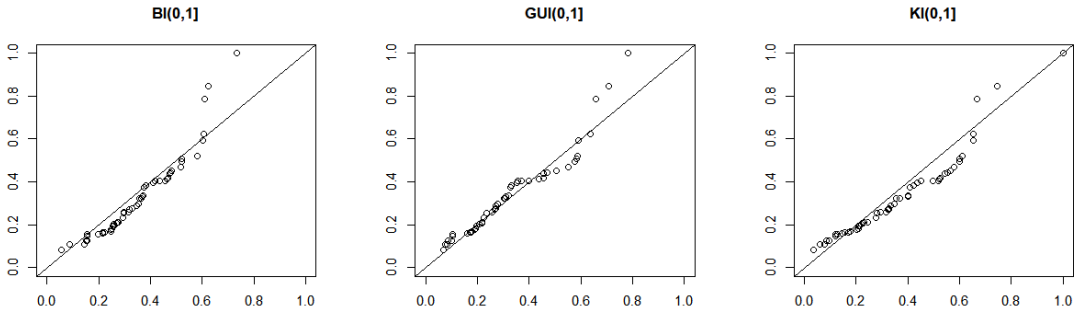

Figura 4.12: QQ-Plot da proporção de votos para Bolsonaro em 53 seções eleitorais.

4.8 Proporção de pessoas que residem em domicílios com acesso a energia elétrica

A variável de interesse(*Energia eltrica*) é a proporção de pessoas que residem em domicílio com acesso a energia elétrica no ano de 1991 em 293 municípios do estado de Santa Catarina (`http://www.atlasbrasil.org.br/`). Vale ressaltar que, de acordo com os dados, há 9 municípios que apresentaram valor igual a um, ou

seja, apresentando 3,07% de inflacionamento em um. De acordo com a Tabela 4.19 a proporção média de pessoas que reside em domicílio com acesso a energia elétrica é de 0,8906 e o mínimo observado foi de 0,3570. A Tabela 4.20 apresenta

Tabela 4.19: Medidas descritivas da proporção de pessoas que residem em domicílios com acesso a energia elétrica.

Mínimo	$Q_{1/4}$	Mediana	Média	$Q_{3/4}$	Máximo	DP	CV
0,3570	0,8406	0,9361	0,8906	0,9870	1,0000	0,1283	144,03%

os p-valores dos testes de hipóteses realizados para verificar a adequação dos dados às distribuições abordadas, a partir dos seguintes testes de hipótese: Kolmogorov–Smirnov (KS), Anderson–Darling (AD) e Cramér–Von Mises (CVM). Ao nível de significância de 5%, para o teste de Anderson-Darling(AD) foi rejeitado a hipótese de que os dados seguem as distribuições BI, KI ou GUI, entretanto, para os outros dois testes (KS e CVM), não foi rejeitada a hipótese de que os dados seguem as distribuições BI, KI e GUI. A Figura 4.13 apresenta o histograma, em que a barra vertical representa a frequência de uns. Em comparação com a aplicação anterior, mesmo sendo as duas inflacionadas em 1, esta aplicação tem uma média de 0,89. Quando observamos o histograma, percebemos o motivo da média ser bem próxima de 0,9: os dados estão concentrados no intervalo de 0,9 a 1. A Figura 4.14 apresenta os gráficos qq-plot das três distribuições ajustadas, em que o eixo x apresenta as probabilidades teóricas e o eixo y apresenta as probabilidades empíricas. Observando esses gráficos, podemos notar que no gráfico da GUI houve um melhor resultado, por os pontos do gráfico dessa distribuição ficaram mais próximos da diagonal do que nas outras duas distribuições.

Tabela 4.20: P-valores dos testes de hipóteses.

	Testes		
Distribuição	KS	CVM	AD
BI	0,8338	0,7988	$< 2,1\text{x}10^{-06}$
KI	0,5378	0,448	$< 2,1\text{x}10^{-06}$
GUI	0,5471	0,5133	$< 2,1\text{x}10^{-06}$

A Tabela 4.21 apresenta os resultados sumarizados dos critérios de adequabilidade. Ao analisar os critérios utilizados, percebe-se que a distribuição GUI obteve uma melhor adequabilidade aos dados quando comparadas com as outras medidas.

Desta forma, considerando os resultados dos critérios e testes de hipóteses, podemos concluir que a distribuição que melhor representa os dados da proporção de pessoas que reside em domicílio com acesso à energia elétrica é a distribuição gama unitária inflacionada.

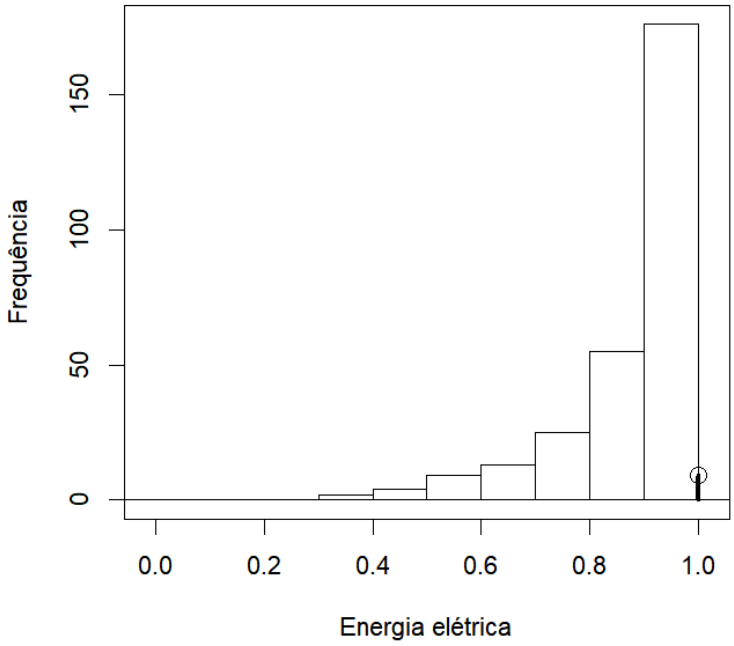

Figura 4.13: Histograma da proporção de pessoas que residem em domicílio com acesso a energia elétrica em 293 municípios.

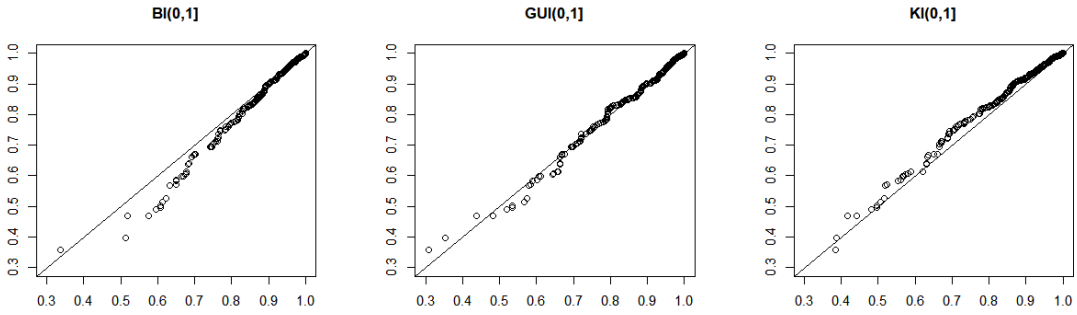

Figura 4.14: QQ-Plot da proporção de pessoas que residem em domicílio com acesso a energia elétrica em 293 municípios.

Tabela 4.21: Critérios de adequabilidade das distribuições: AIC, AICc, BIC, BICc, HQ e HQc - Dados sobre a proporção de pessoas que reside em domicílios com acesso a energia elétrica por municípios do estado de Santa Catarina.

Distribuição	AIC	AICc	BIC	BICc	HQ	HQc
BI	$-611,987$	$-611,903$	$-600,946$	$-600,711$	$-607,565$	$-607,421$
KI	$-611,424$	$-611,340$	$-600,383$	$-600,147$	$-607,002$	$-606,857$
GUI	$-612,043$	$-611,960$	$-601,002$	$-600,766$	$-607,621$	$-607,477$

Capítulo 5

Conclusão

Quando os dados são gerados pela distribuição beta inflacionada (BI), para os três tipos de inflacionamentos, de acordo com os critérios, foi escolhida com maior frequência a distribuição gama unitária inflacionada (GUI). Para inflacionamento em zero, os melhores resultados foram quando os dados tinham média $0,5$ e proporção de inflacionamento de 10%. Já para o inflacionamento em zero e um ou em um, os melhores resultados foram quando os dados tinha média de $0,8$ e uma precisão mais alta ($\phi = 50$).

Para os dados gerados pela distribuição kumaraswamy inflacionada (KI), para todos 3 tipos de inflacionamento, de acordo com os critérios, foi escolhida a distribuição KI com maior frequência. Para o inflacionamento em zero, os melhores resultados foram quando os dados tinha uma precisão maior. Para o inflacionamento em zero e um, os melhores resultados foram quando os dados tinha média $0,3$ e precisão maior ($\phi = 50$). E para o inflacionamento em um, os melhores resultados foram quando os dados pertenciam aos cenários 6, 7, 8 e 9 com precisão maior.

Para os dados gerados pela distribuição GUI, para os três tipos de inflacionamento, de acordo com os critérios, foi escolhida a distribuição GUI com maior frequência. Para o inflacionamento em zero os melhores resultados foram quando os dados tinham média $0,5$, proporção de inflacionamento de 7% ou 10% e uma precisão maior ($\phi = 50$). Pra o inflacionamento em zero e um ou em um, os melhores resultados foram quando os dados tinha média $0,5$ e uma precisão maior ($\phi = 50$).

Para uma melhor compreensão, a Tabela 5.1 apresenta um pequeno resumo dos melhores cenários e distribuições mais escolhidas a partir dos critérios, para cada tipo de amostra gerada, assim como também algumas características evidentes.

Com relação aos testes de hipóteses, quando os dados são gerados pela distribuição BI, a própria distribuição BI foi mais escolhida. É possível perceber que, quando os dados são inflacionados em zero, os resultados não são tão bons quando a proporção de inflacionamento é 10%. Para os cenários de inflacionamento em zero e um, o teste Cramér–von Mies (CVM) indica que todos cenários têm bons resultados, mas

Tabela 5.1: Melhores cenários e distribuições mais escolhidas de acordo com os critérios.

Amostra	mais escolhida	melhores cenários	características
$y \sim$ BI[0,1)	GUI[0,1)	6	$\phi = 50$ e $\mu = 0,5$
$y \sim$ GUI[0,1)			
$y \sim$KI[0,1)	KI[0,1)	1	$\phi = 30$ e $\mu = 0,3$
$y \sim$ BI[0,1]	GUI[0,1]	7, 8 e 9	$\phi = 50$ e $\mu = 0,8$
$y \sim$ GUI[0,1]			
$y \sim$KI[0,1]	KI[0,1]	1, 2 e 3	$\phi = 30$ e $\mu = 0,3$
$y \sim$ BI(0,1]	GUI(0,1]	7,8 e 9	$\phi = 50$ e $\mu = 0,8$
$y \sim$ GUI(0,1]			
$y \sim$KI(0,1]	KI[0,1)	6, 7, 8 e 9	$\phi = 50$ e $\mu = 0,5; 0,8$

para o teste Kolmogorov-Smirnov (KS) indica que, com uma média $0,8$ e proporção de inflacionamento de 10%, os resultados não são tão bons quando o tamanho da amostra é 200. Para os cenários de inflacionamento em um, o teste CVM indicou os melhores os resultados para quando o inflacionamento é 5% ou 7%, enquanto o teste KS indicou apenas quando o inflacionamento é de 5%.

Com relação aos testes de hipóteses, quando os dados são gerados pela distribuição KI, a própria KI foi mais escolhida. É possível perceber que, quando os dados são inflacionados em zero, os melhores resultados para os três testes são quando os dados tem proporção de inflacionamento de 5% ou 7% e uma precisão mais baixa ($\phi = 10$), tanto para média de $0,3$ quanto para $0,5$. Para os cenários com inflacionamento em zero e um, o teste de CVM indica que todos cenários têm bons resultados, entretanto, para o teste KS indica que todos resultados são bons quando a precisão é mais baixa ($\phi = 10$), com exceção quando a média é $0,8$ e a proporção de inflacionamento é de 10% (cenário 9). Para os cenários com inflacionamento em um, o teste de CVM indicou que todos cenários têm bons resultados, enquanto o teste de KS indica que os melhores resultados serão quando a média é $0,5$, proporção de inflacionamento de 5% ou 7%, uma precisão mais baixa, e para média de $0,8$ o teste KS indicou os cenários em que havia uma proporção de inflacionamento de 5% ou 7%.

Com relação aos testes de hipóteses, quando os dados são gerados pela distribuição GUI, a própria GUI foi a melhor. É possível perceber que, quando os dados são inflacionados em zero, para os três testes todos resultados foram bom com exceção quando a proporção era de 10% de inflacionamento. Para dados com inflacionamento em zero e um, para o teste de KS, os resultados não são tão bons quando a média é $0,8$ e a proporção de inflacionamento é de 10%. Ainda para dados com inflacionamento em zero e um, para teste CVM, todos cenários teve bons resultados. Para dados com inflacionamento em um, o teste CVM e KS, os melhores resultados

foram quando os dados tinha uma proporção de inflacionamento de 5% ou 7%.

Para uma melhor compreensão, a Tabela 5.2 apresenta um pequeno resumo dos melhores cenários e distribuições mais escolhidas a partir dos testes de hipótese para cada tipo de amostra gerada, assim como também algumas características evidentes.

Tabela 5.2: Melhores cenários e distribuições mais escolhidas de acordo com os testes de hipótese.

Amostra	mais escolhida	melhores cenários	características
$y \sim$ BI[0,1)	BI[0,1)	1, 2, 4 e 5	5% ou 7% inflacionamento
$y \sim$ GUI[0,1)	GUI[0,1)		
$y \sim$KI[0,1)	KI[0,1)		
$y \sim$ BI[0,1]	BI[0,1]	1-8	
$y \sim$ GUI[0,1]	GUI[0,1]		
$y \sim$KI[0,1]	KI[0,1]		
$y \sim$ BI(0,1]	BI(0,1]	4, 5, 7 e 8	5% ou 7% inflacionamento
$y \sim$ GUI(0,1]	GUI(0,1]		
$y \sim$KI(0,1]	KI[0,1)		

Em relação às aplicações, quando o inflacionamento é em zero e a média é próxima a zero, a distribuição escolhida foi a GUI quando o CV foi maior (130%; 169%. Ainda com inflacionamento em zero, a distribuição BI foi escolhida quando o CV foi próximo 50%. Para a aplicação com inflacionamento em zero e um, em relação aos CV, média e tamanho da amostra, os dois conjuntos de dados das duas aplicações têm características semelhantes, a maior diferença esta na proporção de inflacionamento, a distribuição KI foi escolhida quando a proporção foi próximo a 15% e a GUI quando a proporção foi próximo a 10%. Para aplicação com inflacionamento um, em relação ao CV, média e tamanho da amostra, são um pouco diferentes, mas a proporção de inflacionamento ($1,89\%$ e $3,07\%$) são valores semelhantes e, para os dois conjuntos de dados, a GUI foi a escolhida.

Para uma melhor compreensão a Tabela 5.3 é apresentado o resumo das aplicações, na primeira coluna é a primeira aplicação é a 1, a segunda aplicação é a 2 e assim por diante,a segunda coluna apresenta a proporção de inflacionamento, a terceira coluna a distribuição que melhor representou o conjunto de dados, a quarta coluna o coeficiente de variação, a quinta coluna e média do conjunto e na ultima coluna o tamanho do conjunto de dados.

Tabela 5.3: Resumo das aplicações.

Aplicação	% inflacionamento	distribuição	CV	μ	n
1	$8,13\%$	GUI	$169,25\%$	$0,0647$	209
2	$5,83\%$	BI	$49,40\%$	$0,0625$	223
3	$5,06\%$	GUI	$137,49\%$	$0,1365$	178
4	$9,61\%$	GUI	$50,67\%$	$0,5987$	52
5	$14,55\%$	KI	$62,43\%$	$0,5471$	55
6	$1,89\%$	GUI	$58,40\%$	$0,3274$	53
7	$3,07\%$	GUI	$144,03\%$	$0,8906$	293

Referências Bibliográficas

[1] AKAIKE, H. Information theory and an extension of the maximum likelihood principle. In: **Selected papers of hirotugu akaike**. New York, NY: Springer New York, 1998. p. 199-213.

[2] ANDERSON, Theodore W.; DARLING, Donald A. Asymptotic theory of certain"goodness of fit"criteria based on stochastic processes. **The annals of mathematical statistics**, p. 193-212, 1952.

[3] BAYER, Fábio M.; CRIBARI-NETO, Francisco; SANTOS, Jéssica. Inflated Kumaraswamy regressions with application to water supply and sanitation in Brazil. **Statistica Neerlandica**, v. 75, n. 4, p. 453-481, 2021.

[4] BARNDORFF-NIELSEN, O. E. JϕRGENSEN, B. some parametric models on the simplex. **Journal of Multivariate analysis**, v. 39, n. 1, p. 106-116, 1991.

[5] BAPAT, Sudeep R.; BHARDWAJ, Rohit. On an inflated Unit-Lindley distribution. **arXiv preprint arXiv:2102.04687**, 2021.

[6] BERNARDI, P. B. Análise de risco de Investimento Imobiliário por Simulação . 117p. Dissertação de mestrado - UFSC, Florianópolis-SC, 2002.

[7] BOLFARINE, H.; SANDOVAL, M. C. **Introdução à inferência estatística**, 2ª Ed. Rio de Janeiro: SBM, 2010.

[8] BROEK, J. Van den. A score test for zero inflation in a Poisson distribution. **Biometrics**, p. 738-743, 1995.

[9] CHAKRABORTY, Subrata; BHATTACHARJEE, Sahana. Modeling proportion of success in high school leaving examination-A comparative study of Inflated Unit Lindley and Inflated Beta distribution. **arXiv preprint arXiv:2103.08916**, 2021.

[10] CRIBARI-NETO, F.; SANTOS, J. Inflated Kumaraswamy distributions. **Anais da Academia Brasileira de Ciências**, v. 91, 2019.

[11] CSÖRGŐ, Sándor; FARAWAY, Julian J. The exact and asymptotic distributions of Cramér-von Mises statistics. **Journal of the Royal Statistical Society Series B: Statistical Methodology**, v. 58, n. 1, p. 221-234, 1996.

[12] DARLING, Donald A. The kolmogorov-smirnov, cramer-von mises tests. **The Annals of Mathematical Statistics**, v. 28, n. 4, p. 823-838, 1957.

[13] FERRARI, Silvia; CRIBARI-NETO, Francisco. Beta regression for modelling rates and proportions. **Journal of applied statistics**, v. 31, n. 7, p. 799-815, 2004.

[14] GOKHALE, D. V.; KULLBACK, S. The minimum discrimination information approach in analyzing categorical data: The minimum discrimination information. **Communications in Statistics-Theory and Methods**, v. 7, n. 10, p. 987-1005, 1978.

[15] GRASSIA, A. On a family of distributions with argument between 0 and 1 obtained by transformation of the gamma and derived compound distributions. **Australian Journal of Statis**, Wiley Online Library, V.19, n.2, p. 108-114, 1977.

[16] HALTON, John H. A retrospective and prospective survey of the Monte Carlo method. **Siam review**, v. 12, n. 1, p. 1-63, 1970.

[17] HANNAN, Edward J.; QUINN, Barry G. The determination of the order of an autoregression. **Journal of the Royal Statistical Society: Series B (Methodological)**, v. 41, n. 2, p. 190-195, 1979.

[18] HELU, Siosaia Langitoto. Statistical model selection criteria and their application to the Stock Synthesis assessment program.1998.

[19] HURVICH, C M.; TSAI, Chih-Ling. Regression and time series model selection in small samples. **Biometrika**, v. 76, n. 2, p. 297-307, 1989.

[20] KOLMOGOROFF, Andrey. Confidence limits for an unknown distribution function. **The annals of mathematical statistics**, v. 12, n. 4, p. 461-463, 1941.

[21] KUMARASWAMY, Ponnambalam. A generalized probability density function double-bounded random processes. **Journal of hydrology**, v. 46, n. 1-2, p. 79-88, 1980.

[22] MARSAGLIA, George; MARSAGLIA, John. Evaluating the anderson-darling distribution. **Journal of statistical software**, v. 9, p. 1-5, 2004.

[23] MAZUCHELI, J.; MENEZES, A. F. B.; GHITANY, M. E. The unit-Weibull distribution and associated inference. **J. Appl. Probab. Stat**, v. 13, n. 2, p. 1-22, 2018.

[24] MAZUCHELI, Josmar; MENEZES, André Felipe; DEY, Sanku. Unit-Gompertz distribution with applications. **Statistica**, v. 79, n. 1, p. 25-43, 2019.

[25] MITNIK, P. A.; BAEK, S. The Kumaraswamy distribution: median-dispersion re-parameterizations for regression modeling and simulation-based estimation. **Statistical Papers**, v. 54, p. 177-192, 2013.

[26] MCQUARRIE, Allan D. A small-sample correction for the Schwarz SIC model selection criterion. **Statistics & probability letters**, v. 44, n. 1, p. 79-86, 1999.

[27] OSPINA, Raydonal. **Modelos de regressão beta inflacionados** 174P. Tese de doutorado - USP, São Paulo: SP, 2008.

[28] NOBRE, Aline Araújo et al. Multinomial model and zero-inflated gamma model to study time spent on leisure time physical activity: an example of ELSA-Brasil. **Revista de saude publica**, v. 51, p. 76, 2017.

[29] OSPINA, R.; FERRARI, S. L.P. Inflated beta distributions Statistical papers, v. 51, p. 111-126, 2010.

[30] RIGBY, Robert A. et al. **Distributions for modeling location, scale, and shape: Using GAMLSS in R.** Chapman and Hall/CRC, 2019.

[31] RIGBY, Robert A.; STASINOPOULOS, D. Mikis. Generalized additive models for location, scale and shape. **Journal of the Royal Statistical Society Series C: Applied Statistics**, v. 54, n. 3, p. 507-554, 2005.

[32] RIPLEY, Brian et al. Package 'mass'. **Cran r**, v. 538, p. 113-120, 2013.

[33] SANTOS, J. P. R. Modelos kumaraswamy inflacionados. 115p. Tese de doutorado - UFPE, Recife-PE, 2018.

[34] SCHWARZ, G., Estimating the dimension of a model. **The annals of statistics**, p. 461-464, 1978.

[35] SILVA, C.R. (2023), **Gráfico de controle do tipo shewhart considerando a distribuição gama unitária inflaccionada**. Tese de doutorado - Programa de Pós Graduação em Biometria e Estatística Aplicada, Universidade Federal Rural de Pernambuco.

[36] SMIRNOFF, N. Sur les écarts de la courbe de distribution empirique. **Matematicheskii Sbornik**, v. 48, n. 1, p. 3-26, 1939.

[37] SONG, P. X. -K., **Correlated Data Analysis: Modeling, Analytics, and Applications**. Springer Series in Statistics New York: 2007.

[38] SUGIURA, N. Further analysis of the data by Akaike's information criterion and the finite corrctions: further analysis of the data by Akaike's. **Communications in Statstics-theory and Methods**, v. 7, n. 1, p. 13-26, 1978.

[39] TONG, Edward NC; MUES, Christophe; THOMAS, Lyn. A zero-adjusted gamma model for mortgage loan loss given default. **International Journal of Forecasting**, v. 29, n. 4, p. 548-562, 2013.

[40] WOLODZKO, Tymoteusz; WOLODZKO, Maintainer Tymoteusz. Package 'extraDistr'. 2020.

[41] YEE, Thomas W. The VGAM package. **R News**, v. 8, n. 2, p. 28-39, 2008.

Apêndice A

Tabela A.1: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim \text{BI}[0,1)$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
1	10	30	95,42	0,95	89,25	95,07	0,00	82,43	94,84	0,79	88,90
		60	95,27	0,00	83,53	94,55	0,00	70,77	94,76	0,00	84,23
		100	95,35	0,00	75,95	94,59	0,00	52,27	94,82	0,00	75,35
		200	95,53	0,00	51,32	93,86	0,00	16,78	95,01	0,00	46,68
	30	30	95,42	0,00	89,71	95,07	0,00	82,69	94,84	0,00	89,29
		60	95,27	0,00	84,31	94,55	0,00	71,45	94,76	0,00	85,20
		100	95,35	0,00	77,67	94,59	0,00	53,92	94,82	0,00	77,14
		200	95,53	0,00	54,90	93,86	0,00	17,89	95,01	0,00	49,79
2	10	30	95,60	1,32	89,61	95,06	0,00	83,27	94,80	1,24	89,11
		60	95,33	0,00	84,06	94,31	0,00	71,89	94,72	0,00	84,71
		100	95,35	0,00	76,38	93,76	0,00	52,53	94,64	0,00	75,88
		200	95,53	0,00	52,66	91,73	0,00	14,65	94,63	0,00	47,05
	30	30	95,60	0,00	90,01	95,06	0,00	83,46	94,80	0,00	89,48
		60	95,33	0,00	84,75	94,31	0,00	72,25	94,72	0,00	85,51
		100	95,35	0,00	77,81	93,76	0,00	53,62	94,64	0,00	77,50
		200	95,53	0,00	56,11	91,73	0,00	15,48	94,63	0,00	49,96
3	10	30	95,78	1,81	90,07	94,88	0,00	83,93	94,78	1,67	89,42
		60	95,34	0,00	84,68	93,61	0,00	71,07	94,40	0,00	84,69
		100	95,35	0,00	77,23	91,79	0,00	48,49	94,16	0,00	75,35
		200	0,00	0,00	0,00	83,53	0,00	7,29	93,00	0,00	43,46
	30	30	95,78	0,00	90,46	94,88	0,00	84,12	94,78	0,01	89,85
		60	95,34	0,00	85,28	93,61	0,00	71,34	94,40	0,00	85,63
		100	95,35	0,00	78,40	91,79	0,00	49,37	94,16	0,00	76,87
		200	0,00	0,00	0,00	83,53	0,00	7,55	93,00	0,00	45,77

Tabela A.2: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim \text{BI}[0,1)$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	95,42	8,22	80,08	95,07	0,05	56,56	94,84	9,97	80,82
		60	95,27	0,09	64,57	94,55	0,00	26,82	94,76	0,11	62,31
		100	95,35	0,00	40,71	94,59	0,00	6,95	94,82	0,00	33,99
		200	95,53	0,00	6,65	93,86	0,00	0,01	95,01	0,00	3,09
	50	30	95,42	0,02	80,92	95,07	0,00	58,46	94,84	0,02	81,42
		60	95,27	0,00	66,07	94,55	0,00	29,55	94,76	0,00	64,23
		100	95,35	0,00	43,73	94,59	0,00	8,26	94,82	0,00	36,93
		200	95,53	0,00	8,07	93,86	0,00	0,04	95,01	0,00	4,05
5	10	30	95,60	6,43	80,37	95,06	0,02	57,13	94,80	8,50	81,00
		60	95,33	0,03	64,45	94,31	0,00	26,52	94,72	0,06	62,43
		100	95,35	0,00	39,95	93,76	0,00	6,20	94,64	0,00	33,57
		200	95,53	0,00	6,34	91,73	0,00	0,00	94,63	0,00	2,67
	50	30	95,60	0,00	81,21	95,06	0,00	59,08	94,80	0,00	81,69
		60	95,33	0,00	66,07	94,31	0,00	28,95	94,72	0,00	64,38
		100	95,35	0,00	42,96	93,76	0,00	7,48	94,64	0,00	36,25
		200	95,53	0,00	7,89	91,73	0,00	0,01	94,63	0,00	3,58
6	10	30	95,78	3,31	80,61	94,88	0,00	56,35	94,78	3,74	81,06
		60	95,34	0,00	63,56	93,61	0,00	23,53	94,40	0,00	61,65
		100	95,35	0,00	39,26	91,80	0,00	3,96	94,16	0,00	31,06
		200	0,00	0,00	0,00	83,55	0,00	0,00	93,00	0,00	1,55
	50	30	95,78	0,00	81,40	94,88	0,00	58,59	94,78	0,00	81,79
		60	95,34	0,00	65,44	93,61	0,00	26,32	94,40	0,00	63,51
		100	95,35	0,00	42,20	91,80	0,00	5,12	94,16	0,00	33,88
		200	0,00	0,00	0,00	83,55	0,00	0,00	93,00	0,00	2,17

Tabela A.3: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim \text{KI}[0,1)$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
1	10	30	0,00	95,42	0,01	0,00	95,07	0,01	0,00	94,84	0,05
		60	0,00	95,27	0,00	0,00	94,55	0,00	0,00	94,76	0,00
		100	0,00	95,35	0,00	0,00	94,59	0,00	0,00	94,82	0,00
		200	0,00	95,53	0,00	0,00	93,86	0,00	0,00	95,01	0,00
	30	30	0,00	74,67	0,00	0,00	83,20	0,00	0,00	90,26	0,00
		60	0,00	39,76	0,00	0,00	62,60	0,00	0,00	83,90	0,00
		100	0,00	0,00	0,00	0,00	17,36	0,00	0,00	66,02	0,00
		200	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
2	10	30	0,00	95,60	0,00	0,00	95,06	0,02	0,00	94,80	0,00
		60	0,00	95,33	0,00	0,00	94,31	0,00	0,00	94,72	0,00
		100	0,00	95,35	0,00	0,00	93,76	0,00	0,00	94,64	0,00
		200	0,00	95,53	0,00	0,00	91,73	0,00	0,00	94,63	0,00
	30	30	0,00	75,35	0,00	0,00	86,20	0,00	0,00	90,35	0,00
		60	0,00	39,39	0,00	0,00	71,01	0,00	0,00	84,63	0,00
		100	0,00	0,00	0,00	0,00	30,67	0,00	0,00	68,77	0,00
		200	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3	10	30	0,00	95,78	0,00	0,00	94,88	0,02	0,00	94,78	0,03
		60	0,00	95,34	0,00	0,00	93,61	0,00	0,00	94,40	0,00
		100	0,00	95,35	0,00	0,00	91,80	0,00	0,00	94,16	0,00
		200	0,00	0,00	0,00	0,00	83,55	0,00	0,00	93,00	0,00
	30	30	0,00	77,37	0,00	0,00	87,95	0,00	0,00	90,85	0,00
		60	0,00	46,28	0,00	0,00	76,14	0,00	0,00	85,21	0,00
		100	0,00	5,84	0,00	0,00	37,95	0,00	0,00	69,95	0,00
		200	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Tabela A.4: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim \text{KI}[0,1)$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	0,26	95,42	29,53	0,18	95,07	31,62	0,53	94,84	37,36
		60	0,00	95,27	0,25	0,00	94,55	0,08	0,00	94,76	0,42
		100	0,00	95,35	0,00	0,00	94,59	0,00	0,00	94,82	0,00
		200	0,00	95,53	0,00	0,00	93,86	0,00	0,00	95,01	0,00
	50	30	0,00	94,87	0,00	0,00	94,59	0,00	0,00	94,57	0,00
		60	0,00	93,56	0,00	0,00	93,56	0,00	0,00	94,48	0,00
		100	0,00	91,43	0,00	0,00	93,21	0,00	0,00	94,21	0,00
		200	0,00	85,75	0,00	0,00	90,99	0,00	0,00	93,67	0,00
5	10	30	0,08	95,60	18,97	0,10	95,06	21,63	0,21	94,80	25,47
		60	0,00	95,33	0,07	0,00	94,31	0,02	0,00	94,72	0,11
		100	0,00	95,35	0,00	0,00	93,76	0,00	0,00	94,64	0,00
		200	0,00	95,53	0,00	0,00	91,73	0,00	0,00	94,63	0,00
	50	30	0,00	94,37	0,00	0,00	94,91	0,00	0,00	94,59	0,00
		60	0,00	93,37	0,00	0,00	93,80	0,00	0,00	94,41	0,00
		100	0,00	91,27	0,00	0,00	92,69	0,00	0,00	94,02	0,00
		200	0,00	84,87	0,00	0,00	88,65	0,00	0,00	93,33	0,00
6	10	30	0,02	95,78	7,47	0,05	94,88	7,68	0,10	94,78	9,79
		60	0,00	95,34	0,03	0,00	93,61	0,00	0,00	94,40	0,02
		100	0,00	95,35	0,00	0,00	91,80	0,00	0,00	94,16	0,00
		200	0,00	0,00	0,00	0,00	83,55	0,00	0,00	93,00	0,00
	30	30	0,00	94,46	0,00	0,00	94,56	0,00	0,00	94,46	0,00
		60	0,00	93,60	0,00	0,00	93,03	0,00	0,00	94,09	0,00
		100	0,00	90,96	0,00	0,00	90,68	0,00	0,00	93,54	0,00
		200	0,00	0,00	0,00	0,00	78,69	0,00	0,00	91,36	0,00

Tabela A.5: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim \text{GUI}[0,1)$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
1	10	30	93,43	3,27	95,42	95,20	0,00	95,07	94,43	2,80	94,84
		60	89,12	0,01	95,27	88,29	0,00	94,55	90,28	0,00	94,76
		100	81,30	0,00	95,35	72,56	0,00	94,59	81,12	0,00	94,82
		200	54,44	0,00	95,53	26,87	0,00	93,86	48,50	0,00	95,01
	30	30	93,85	0,00	95,42	95,55	0,00	95,07	94,82	0,00	94,84
		60	90,11	0,00	95,27	89,12	0,00	94,55	91,10	0,00	94,76
		100	83,52	0,00	95,35	74,35	0,00	94,59	83,26	0,00	94,82
		200	58,50	0,00	95,53	29,05	0,00	93,86	52,76	0,00	95,01
2	10	30	93,54	4,89	95,60	95,05	0,00	95,06	94,37	5,46	94,80
		60	89,30	0,02	95,33	88,07	0,00	94,31	90,35	0,03	94,72
		100	81,67	0,00	95,35	71,93	0,00	93,76	81,27	0,00	94,64
		200	55,72	0,00	95,53	23,00	0,00	91,73	49,13	0,00	94,63
	30	30	93,89	0,00	95,60	95,39	0,00	95,06	94,77	0,00	94,80
		60	90,28	0,00	95,33	88,91	0,00	94,31	91,08	0,00	94,72
		100	83,54	0,00	95,35	73,43	0,00	93,76	83,17	0,00	94,64
		200	59,55	0,00	95,53	24,55	0,00	91,73	52,81	0,00	94,63
3	10	30	93,51	3,31	95,78	94,72	0,00	94,88	94,24	3,57	94,78
		60	89,45	0,01	95,34	86,74	0,00	93,61	89,95	0,01	94,40
		100	82,17	0,00	95,35	66,83	0,00	91,79	80,55	0,00	94,16
		200	0,00	0,00	0,00	11,24	0,00	83,53	44,96	0,00	93,00
	30	30	93,99	0,00	95,78	94,96	0,00	94,88	94,51	0,01	94,78
		60	90,46	0,00	95,34	87,46	0,00	93,61	90,61	0,00	94,40
		100	83,81	0,00	95,35	67,96	0,00	91,79	82,27	0,00	94,16
		200	0,00	0,00	0,00	11,69	0,00	83,53	48,28	0,00	93,00

Tabela A.6: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim$ GUI$[0,1)$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	89,69	23,30	95,42	90,69	1,52	95,07	91,43	27,63	94,84
		60	75,36	2,49	95,27	61,25	0,01	94,55	71,98	3,02	94,76
		100	47,49	0,05	95,35	19,12	0,00	94,59	36,57	0,07	94,82
		200	5,06	0,00	95,53	0,13	0,00	93,86	1,57	0,00	95,01
	50	30	90,06	0,00	95,42	91,22	0,00	95,07	91,88	0,05	94,84
		60	76,82	0,00	95,27	63,60	0,00	94,55	74,11	0,00	94,76
		100	50,49	0,00	95,35	22,20	0,00	94,59	40,20	0,00	94,82
		200	6,63	0,00	95,53	0,17	0,00	93,86	2,34	0,00	95,01
5	10	30	89,72	14,98	95,60	90,42	0,59	95,06	91,29	16,92	94,80
		60	74,92	0,83	95,33	60,35	0,00	94,31	71,92	1,00	94,72
		100	46,86	0,00	95,35	17,28	0,00	93,76	35,95	0,00	94,64
		200	4,90	0,00	95,53	0,06	0,00	91,73	1,40	0,00	94,63
	50	30	90,07	0,00	95,60	90,94	0,00	95,06	91,69	0,00	94,80
		60	76,79	0,00	95,33	63,05	0,00	94,31	73,98	0,00	94,72
		100	49,92	0,00	95,35	10,19	0,00	93,76	40,01	0,00	94,64
		200	6,60	0,00	95,53	0,09	0,00	91,73	2,08	0,00	94,63
6	10	30	89,54	4,05	95,78	89,80	0,10	94,88	90,83	3,00	94,78
		60	74,32	0,00	95,34	55,87	0,00	93,61	70,56	0,00	94,40
		100	45,54	0,00	95,35	11,25	0,00	91,80	32,96	0,00	94,16
		200	0,00	0,00	0,00	0,00	0,00	83,55	0,00	0,00	93,00
	50	30	89,85	0,00	95,78	90,47	0,00	94,88	91,17	0,00	94,78
		60	76,22	0,00	95,34	59,14	0,00	93,61	72,85	0,00	94,40
		100	49,23	0,00	95,35	14,09	0,00	91,80	36,94	0,00	94,16
		200	0,00	0,00	0,00	0,00	0,00	83,55	1,25	0,00	93,00

Tabela A.7: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim$ BI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
1	10	30	95,36	0,49	89,33	61,20	0,00	54,57	94,84	0,36	89,05
		60	95,24	0,00	83,95	38,40	0,00	31,68	94,80	0,00	84,83
		100	95,35	0,00	76,69	21,31	0,00	14,26	94,90	0,00	76,53
		200	95,53	0,00	53,11	4,72	0,00	1,60	95,12	0,00	49,75
	30	30	95,36	0,00	89,57	61,20	0,00	55,07	94,84	0,05	89,50
		60	95,24	0,00	84,79	38,40	0,00	32,32	94,80	0,00	85,74
		100	95,35	0,00	78,67	21,31	0,00	15,16	94,90	0,00	78,40
		200	95,53	0,00	57,08	4,72	0,00	1,83	95,12	0,00	53,47
2	10	30	95,55	0,72	89,82	50,89	0,00	46,14	94,84	0,57	89,38
		60	95,34	0,00	84,53	26,12	0,00	22,16	94,76	0,00	85,61
		100	95,35	0,00	77,55	11,34	0,00	8,01	94,81	0,00	77,78
		200	95,53	0,00	55,24	1,38	0,00	0,46	94,98	0,00	52,33
	30	30	95,55	0,00	90,13	50,89	0,00	46,65	94,84	0,00	89,55
		60	95,34	0,00	85,47	26,12	0,00	22,68	94,76	0,00	86,43
		100	95,35	0,00	79,45	11,34	0,00	8,49	94,81	0,00	79,64
		200	95,53	0,00	59,13	1,38	0,00	0,55	94,98	0,00	56,23
3	10	30	95,72	1,29	90,30	38,38	0,00	35,40	94,79	0,90	89,78
		60	95,38	0,00	85,39	15,30	0,00	13,36	94,65	0,00	86,38
		100	95,35	0,00	78,64	4,26	0,00	3,17	94,56	0,00	79,29
		200	95,53	0,00	58,27	0,17	0,00	0,05	94,52	0,00	55,13
	30	30	95,72	0,00	90,85	38,38	0,00	35,74	94,79	0,00	90,34
		60	95,38	0,00	86,23	15,30	0,00	13,72	94,65	0,00	87,16
		100	95,35	0,00	86,23	4,26	0,00	13,72	94,56	0,00	87,16
		200	95,53	0,00	6,71	0,17	0,00	0,06	94,52	0,00	58,73

Tabela A.8: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim \text{BI}[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	95,38	8,87	81,64	44,98	0,13	34,34	94,84	8,32	82,43
		60	95,23	0,14	68,16	20,52	0,00	11,15	94,80	0,06	67,05
		100	95,35	0,00	47,84	7,36	0,00	1,94	94,87	0,00	42,31
		200	95,53	0,00	59,13	1,38	0,00	0,55	94,98	0,00	56,23
	50	30	95,38	0,01	82,31	44,98	0,00	34,82	94,84	0,01	83,08
		60	95,23	0,00	69,52	20,52	0,00	11,65	94,80	0,00	68,92
		100	95,35	0,00	50,20	7,36	0,00	2,18	94,87	0,00	44,89
		200	95,53	0,00	12,75	0,60	0,00	0,03	95,09	0,00	7,68
5	10	30	95,53	10,15	82,62	32,81	0,25	26,55	94,84	10,00	83,43
		60	95,33	0,18	69,90	11,04	0,00	6,90	94,70	0,12	69,49
		100	95,35	0,00	50,00	2,65	0,00	0,82	94,81	0,00	46,00
		200	95,53	0,00	12,89	0,03	0,00	0,00	95,02	0,00	7,91
	50	30	95,53	0,00	83,20	32,81	0,00	26,76	94,84	0,02	84,01
		60	95,33	0,00	71,18	11,04	0,00	7,13	94,70	0,00	71,03
		100	95,35	0,00	52,14	2,65	0,00	0,89	94,81	0,00	48,90
		200	95,53	0,00	14,67	0,03	0,00	0,00	95,02	0,00	9,69
6	10	30	95,74	11,60	83,88	20,52	0,39	35,40	94,77	12,72	84,59
		60	95,38	0,29	71,88	4,21	0,00	10,53	94,58	0,14	72,55
		100	95,31	0,00	53,12	0,55	0,00	1,64	94,67	0,00	51,01
		200	95,31	0,00	16,06	0,00	0,00	0,00	94,71	0,00	10,69
	50	30	95,74	0,02	84,48	20,52	0,00	36,00	94,77	0,05	85,13
		60	95,38	0,00	73,07	4,21	0,00	10,87	94,58	0,00	73,92
		100	95,31	0,00	55,68	0,55	0,00	1,82	94,67	0,00	53,89
		200	95,31	0,00	18,40	0,00	0,00	0,00	94,71	0,00	12,64

Tabela A.9: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$.

Cenários	ϕ	n	$y \sim$ BI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
7	10	30	95,40	49,98	25,25	28,19	11,20	2,11	94,85	49,82	22,74
		60	95,22	39,74	2,37	8,07	3,61	0,01	94,72	40,64	1,20
		100	95,33	16,17	0,03	1,52	0,30	0,00	94,83	15,94	0,01
		200	95,53	0,02	0,00	0,02	0,00	0,00	95,00	0,01	0,00
	50	30	95,40	3,27	31,29	28,19	0,02	2,48	94,85	3,81	27,53
		60	95,22	0,03	3,48	8,07	0,00	0,02	94,72	0,03	1,99
		100	95,33	0,00	0,05	1,52	0,00	0,00	94,83	0,00	0,02
		200	95,53	0,00	0,00	0,02	0,00	0,00	95,00	0,00	0,00
8	10	30	95,59	45,57	28,36	16,93	7,45	2,19	94,74	44,68	26,22
		60	95,30	42,47	3,19	2,94	1,74	0,01	94,60	43,76	1,83
		100	95,24	18,27	0,06	0,24	0,10	0,00	94,67	18,61	0,03
		200	94,65	0,00	0,00	0,00	0,00	0,00	94,81	0,00	0,00
	50	30	95,59	3,39	34,62	16,93	0,03	2,33	94,74	4,17	31,04
		60	95,30	0,01	4,79	2,94	0,00	0,01	94,60	0,01	2,89
		100	95,24	0,00	0,08	0,24	0,00	0,00	94,67	0,00	0,02
		200	94,65	0,00	0,00	0,00	0,00	0,00	94,81	0,00	0,00
9	10	30	95,63	39,56	32,98	7,72	3,75	2,51	94,53	37,75	31,44
		60	94,81	9,46	4,81	0,63	0,13	0,05	94,23	9,31	2,97
		100	92,81	1,05	0,11	0,02	0,00	0,00	94,14	0,88	0,05
		200	79,17	0,00	0,00	0,00	0,00	0,00	93,75	0,00	0,00
	50	30	95,63	3,20	39,17	7,72	0,05	2,90	94,53	4,81	36,34
		60	94,81	0,00	6,88	0,63	0,00	0,05	94,23	0,01	4,38
		100	92,81	0,00	0,19	0,02	0,00	0,00	94,14	0,00	0,07
		200	79,17	0,00	0,00	0,00	0,00	0,00	93,75	0,00	0,00

Tabela A.10: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim$ KI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
1	10	30	0,16	95,36	2,00	0,00	61,20	0,02	0,08	94,84	1,17
		60	0,00	95,24	0,00	0,00	38,40	0,00	0,00	94,80	0,00
		100	0,00	95,35	0,00	0,00	21,31	0,00	0,00	94,90	0,00
		200	0,00	95,53	0,00	0,00	4,72	0,00	0,00	95,12	0,00
	30	30	0,00	73,44	0,00	0,00	50,41	0,00	0,00	90,25	0,00
		60	0,00	38,86	0,00	0,00	21,98	0,00	0,00	84,24	0,00
		100	0,00	0,00	0,00	0,00	1,80	0,00	0,00	66,21	0,00
		200	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
2	10	30	0,48	95,55	3,42	0,00	50,89	0,05	0,23	94,84	2,53
		60	0,00	95,34	0,00	0,00	26,12	0,00	0,00	94,76	0,00
		100	0,00	95,35	0,00	0,00	11,34	0,00	0,00	94,81	0,00
		200	0,00	95,53	0,00	0,00	1,38	0,00	0,00	94,98	0,00
	30	30	0,00	73,47	0,02	0,00	44,20	0,00	0,00	90,59	0,02
		60	0,00	45,57	0,00	0,00	17,06	0,00	0,00	85,11	0,00
		100	0,00	0,00	0,00	0,00	2,34	0,00	0,00	69,65	0,00
		200	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3	10	30	1,10	95,72	6,12	0,00	38,38	0,04	1,05	94,79	5,47
		60	0,00	95,38	0,00	0,00	15,30	0,00	0,00	94,65	0,00
		100	0,00	95,35	0,00	0,00	4,26	0,00	0,00	94,56	0,00
		200	0,00	95,53	0,00	0,00	0,17	0,00	0,00	94,52	0,00
	30	30	0,02	78,75	0,06	0,00	34,37	0,00	0,03	90,97	0,11
		60	0,00	46,97	0,00	0,00	11,09	0,00	0,00	86,07	0,00
		100	0,00	5,36	0,00	0,00	1,32	0,00	0,00	73,36	0,00
		200	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Tabela A.11: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim$ KI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	8,49	95,38	60,16	0,11	44,98	17,93	5,73	94,84	61,75
		60	0,03	95,23	10,66	0,00	20,52	0,05	0,01	94,80	8,06
		100	0,00	95,35	0,16	0,00	7,36	0,00	0,00	94,87	0,08
		200	0,00	95,53	0,00	0,00	0,60	0,00	0,00	95,09	0,00
	50	30	0,01	94,53	0,16	0,00	44,58	0,00	0,00	94,55	0,09
		60	0,00	93,16	0,00	0,00	20,29	0,00	0,00	94,46	0,00
		100	0,00	91,26	0,00	0,00	7,24	0,00	0,00	94,23	0,00
		200	0,00	86,29	0,00	0,00	0,58	0,00	0,00	93,83	0,00
5	10	30	11,91	95,53	63,05	0,02	32,81	12,69	9,20	94,84	65,22
		60	0,02	95,33	13,64	0,00	11,04	0,01	0,01	94,70	11,64
		100	0,00	95,35	0,47	0,00	2,65	0,00	0,00	94,81	0,17
		200	0,00	95,53	0,00	0,00	0,03	0,00	0,00	95,02	0,00
	50	30	0,02	94,82	0,39	0,00	32,53	0,00	0,00	94,57	0,29
		60	0,00	93,65	0,00	0,00	10,94	0,00	0,00	94,49	0,00
		100	0,00	91,09	0,00	0,00	2,60	0,00	0,00	94,30	0,00
		200	0,00	85,99	0,00	0,00	0,03	0,00	0,00	93,80	0,00
6	10	30	17,74	95,74	66,71	0,02	20,52	10,60	15,48	94,77	69,62
		60	0,17	95,38	18,72	0,00	4,21	0,00	0,05	94,58	17,72
		100	0,00	95,31	1,27	0,00	0,55	0,00	0,00	94,67	0,70
		200	0,00	95,31	0,00	0,00	0,00	0,00	0,00	94,71	0,00
	50	30	0,07	95,24	1,01	0,00	20,36	0,00	0,04	94,55	0,94
		60	0,00	93,96	0,00	0,00	4,16	0,00	0,00	94,41	0,00
		100	0,00	91,76	0,00	0,00	0,53	0,00	0,00	93,98	0,00
		200	0,00	87,04	0,00	0,00	0,00	0,00	0,00	93,47	0,00

Tabela A.12: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$.

Cenários	ϕ	n	$y \sim$ KI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
7	10	30	71,05	95,40	76,43	15,11	28,19	16,76	73,36	94,85	77,60
		60	36,25	95,22	56,46	0,26	8,07	2,47	38,04	94,72	54,56
		100	10,17	95,33	30,52	0,00	1,52	0,08	9,64	94,83	25,51
		200	0,11	95,53	2,55	0,00	0,02	0,00	0,00	95,00	1,20
	50	30	2,85	95,40	88,86	0,00	28,19	23,86	1,98	94,85	91,22
		60	0,00	95,22	68,53	0,00	8,07	68,53	0,00	94,72	76,09
		100	0,00	95,33	36,24	0,00	1,52	0,03	0,00	94,83	47,21
		200	0,00	95,53	2,76	0,00	0,02	0,00	0,00	95,00	4,95
8	10	30	72,41	95,59	77,83	7,61	16,93	10,60	74,72	94,74	78,82
		60	39,93	95,30	58,74	0,04	2,94	1,10	41,81	94,60	57,18
		100	12,96	95,24	32,90	0,00	0,24	0,02	12,52	94,67	28,26
		200	0,15	94,65	3,43	0,00	0,00	0,00	0,15	94,81	1,72
	50	30	5,10	95,59	89,04	0,00	16,93	13,42	3,70	94,74	91,31
		60	0,00	95,30	70,25	0,00	2,94	0,47	0,00	94,60	77,28
		100	0,00	95,24	39,83	0,00	0,24	0,00	0,00	94,67	51,03
		200	0,00	94,65	3,99	0,00	0,00	0,00	0,00	94,81	6,73
9	10	30	74,82	95,63	79,78	2,44	7,72	13,83	76,55	94,53	80,18
		60	43,85	94,81	61,76	0,00	0,63	1,09	46,73	94,23	60,60
		100	15,07	92,81	36,07	0,00	0,02	0,04	16,08	94,14	32,11
		200	0,04	79,17	4,67	0,00	0,00	0,00	0,16	93,75	2,66
	50	30	8,75	95,63	88,92	0,00	7,72	14,11	6,94	94,53	91,44
		60	0,00	94,81	71,57	0,00	0,63	0,12	0,00	94,23	79,13
		100	0,00	92,81	42,40	0,00	0,02	0,00	0,00	94,14	55,13
		200	0,00	79,17	1,91	0,00	0,00	0,00	0,00	93,75	8,02

Tabela A.13: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,3$.

Cenários	ϕ	n	$y \sim$ GUI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
1	10	30	93,66	2,28	95,36	60,98	0,05	61,20	94,69	2,43	94,84
		60	89,55	0,00	95,24	35,26	0,00	38,40	90,80	0,00	94,80
		100	82,34	0,00	95,35	15,24	0,00	21,31	82,52	0,00	94,90
		200	55,96	0,00	95,53	0,82	0,00	4,72	51,83	0,00	95,12
	30	30	93,97	0,00	95,36	60,98	0,00	61,20	94,95	0,00	94,80
		60	90,68	0,00	95,24	35,57	0,00	38,40	91,68	0,00	94,80
		100	84,31	0,00	95,35	15,54	0,00	21,31	84,56	0,00	94,90
		200	60,84	0,00	95,53	0,93	0,00	4,72	56,25	0,00	95,12
2	10	30	93,82	4,53	95,55	50,73	0,10	50,89	94,69	3,32	94,84
		60	89,99	0,00	95,34	24,05	0,00	26,12	91,06	0,00	94,76
		100	82,82	0,00	95,35	7,88	0,00	11,34	83,44	0,00	94,81
		200	58,07	0,00	95,53	0,19	0,00	1,38	54,92	0,00	94,98
	30	30	94,10	0,00	95,55	50,67	0,00	50,89	94,98	0,02	94,84
		60	90,90	0,00	95,34	24,06	0,00	26,12	91,87	0,00	94,76
		100	84,71	0,00	95,35	8,00	0,00	11,34	85,32	0,00	94,81
		200	62,23	0,00	95,53	0,18	0,00	1,38	54,19	0,00	94,98
3	10	30	94,07	6,51	95,72	38,21	0,25	38,38	94,68	5,32	94,79
		60	90,42	0,06	95,38	13,99	0,00	15,30	91,28	0,00	94,65
		100	83,58	0,00	95,35	2,91	0,00	4,26	84,27	0,00	94,56
		200	61,11	0,00	95,53	0,02	0,00	0,17	57,48	0,00	94,52
	30	30	94,35	0,02	95,72	38,20	0,00	38,38	94,95	0,04	94,79
		60	91,21	0,00	95,38	13,85	0,00	15,30	92,04	0,00	94,65
		100	85,03	0,00	95,35	2,88	0,00	4,26	85,98	0,00	94,56
		200	65,01	0,00	95,53	0,03	0,00	0,17	61,47	0,00	94,52

Tabela A.14: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim$ GUI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	90,87	48,10	95,38	41,77	11,38	44,98	92,16	50,26	94,84
		60	78,55	13,98	95,23	10,97	0,57	20,52	77,00	12,11	94,80
		100	54,64	1,24	95,35	0,76	0,00	7,36	46,23	0,62	94,87
		200	9,86	0,00	95,53	0,00	0,00	0,60	4,32	0,00	95,09
	50	30	91,18	0,55	95,38	41,77	0,00	44,98	92,47	0,41	94,84
		60	79,69	0,00	95,23	11,57	0,00	20,52	78,58	0,00	94,80
		100	57,40	0,00	95,35	0,97	0,00	7,36	49,76	0,00	94,87
		200	11,96	0,00	95,53	0,00	0,00	0,60	5,44	0,00	95,09
5	10	30	91,19	50,95	95,53	30,16	11,50	32,81	92,45	53,60	94,84
		60	79,67	16,45	95,33	5,48	0,56	11,04	78,70	14,33	94,70
		100	56,65	1,72	95,35	0,27	0,00	2,65	50,60	1,03	94,81
		200	11,93	0,00	95,53	0,00	0,00	0,03	5,94	0,00	95,02
	50	30	91,43	0,61	95,53	30,38	0,01	32,81	92,81	0,62	94,84
		60	80,71	0,00	95,33	5,81	0,00	11,04	80,06	0,00	94,70
		100	59,18	0,00	95,35	0,35	0,00	2,65	53,38	1,00	94,81
		200	14,38	0,00	95,53	0,00	0,00	0,03	7,52	0,00	95,02
6	10	30	91,65	44,09	95,74	18,73	5,76	49,19	92,91	49,54	94,77
		60	81,04	10,37	95,38	1,79	0,09	20,65	80,99	12,13	94,58
		100	59,55	0,67	95,31	0,03	0,01	8,90	55,48	0,80	94,67
		200	15,77	0,00	95,31	0,00	0,00	0,26	8,91	0,00	94,71
	50	30	91,94	0,87	95,74	18,84	0,01	49,19	93,10	0,87	94,77
		60	81,85	0,00	95,38	1,88	0,00	20,65	82,08	0,00	94,58
		100	61,84	0,00	95,31	0,04	0,00	8,90	58,38	0,00	94,67
		200	18,29	0,00	95,31	0,00	0,00	0,26	10,79	0,00	94,71

Tabela A.15: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$.

Cenários	ϕ	n	$y \sim$ GUI$[0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
7	10	30	28,39	87,22	95,40	0,43	24,34	28,19	22,12	89,00	94,85
		60	1,09	66,34	95,22	0,00	2,13	8,07	0,26	62,95	94,72
		100	0,01	31,45	95,33	0,00	0,03	1,52	0,00	23,40	94,83
		200	0,00	0,96	95,53	0,00	0,00	0,02	0,00	0,34	95,00
	50	30	40,76	80,73	95,40	1,15	24,35	28,19	30,83	82,79	94,85
		60	2,49	64,37	95,22	0,00	6,02	8,07	0,55	69,72	94,72
		100	0,01	40,93	95,33	0,00	0,80	1,52	0,00	48,68	94,83
		200	0,00	7,14	95,53	0,00	0,00	0,02	0,00	10,81	95,00
8	10	30	33,29	88,17	95,59	0,10	14,40	16,93	27,33	89,62	94,74
		60	1,94	68,75	95,30	0,00	0,58	2,94	0,44	65,85	94,60
		100	0,02	34,62	95,24	0,00	0,00	0,24	0,00	26,87	94,67
		200	0,00	1,38	94,65	0,00	0,00	0,00	0,00	0,59	94,81
	50	30	44,56	81,77	95,59	0,49	15,75	16,93	36,90	83,55	94,74
		60	3,86	66,18	95,30	0,00	2,54	2,94	1,06	71,39	94,60
		100	0,1	43,25	95,24	0,00	0,19	0,24	0,00	51,57	94,67
		200	0,00	8,94	94,65	0,00	0,00	0,00	0,00	13,02	94,81
9	10	30	39,32	89,28	95,63	0,04	6,33	25,52	34,89	90,30	94,53
		60	3,43	71,02	94,81	0,00	0,13	4,05	1,07	69,17	94,23
		100	0,01	38,25	92,81	0,00	0,00	0,62	0,00	31,86	94,14
		200	0,00	1,91	79,17	0,00	0,00	0,00	0,00	0,82	93,75
	50	30	50,43	82,96	95,63	0,15	7,64	25,52	44,21	84,28	94,53
		60	6,16	68,26	94,81	0,00	0,58	4,05	2,04	73,06	94,23
		100	0,05	46,91	92,81	0,00	0,03	0,62	0,00	54,64	94,14
		200	0,00	12,02	79,17	0,00	0,00	0,00	0,00	16,12	93,75

Tabela A.16: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim \text{BI}(0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	95,48	15,28	83,60	20,41	0,21	14,88	94,78	13,02	83,78
		60	95,25	0,41	72,29	4,21	0,00	2,17	94,62	0,15	70,99
		100	95,31	0,00	54,47	0,56	0,00	0,11	94,73	0,00	49,10
		200	95,31	0,00	16,50	0,00	0,00	0,00	94,90	0,00	9,82
	50	30	95,48	0,02	84,20	20,41	0,00	15,31	94,74	0,02	84,36
		60	95,25	0,00	73,40	4,21	0,00	2,24	94,62	0,00	72,42
		100	95,31	0,00	56,78	0,56	0,00	0,16	94,73	0,00	51,72
		200	95,31	0,00	18,54	0,00	0,00	0,00	94,90	0,00	11,83
5	10	30	95,56	13,99	85,11	10,77	0,06	8,01	94,63	12,75	84,91
		60	95,12	0,38	74,92	1,24	0,00	0,65	94,38	0,18	73,67
		100	94,42	0,00	58,51	0,06	0,00	0,02	94,35	0,00	54,26
		200	89,14	0,00	20,65	0,00	0,00	0,00	94,36	0,00	13,48
	50	30	95,56	0,00	85,54	10,77	0,00	8,24	94,63	0,00	85,53
		60	95,12	0,00	75,92	1,24	0,00	0,70	94,38	0,00	75,20
		100	94,42	0,00	60,42	0,06	0,00	0,02	94,35	0,00	56,78
		200	89,14	0,00	23,18	0,00	0,00	0,00	94,36	0,00	15,85
6	10	30	95,25	8,69	86,59	3,93	0,0,00	2,97	94,23	8,62	85,88
		60	92,32	0,16	76,92	0,13	0,00	0,06	93,77	0,09	76,42
		100	84,05	0,00	59,40	0,00	0,00	0,00	93,34	0,00	58,59
		200	45,03	0,00	16,89	0,00	0,00	0,00	91,76	0,00	17,75
	50	30	95,25	0,00	87,09	3,93	0,00	3,04	94,23	0,00	86,19
		60	92,32	0,00	77,84	0,13	0,00	0,09	93,77	0,00	77,58
		100	84,05	0,00	61,26	0,00	0,00	0,00	93,34	0,00	60,82
		200	45,03	0,00	18,80	0,00	0,00	0,00	91,76	0,00	20,37

Tabela A.17: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$.

Cenários	ϕ	n	$y \sim \text{BI}(0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
7	10	30	95,48	79,84	29,38	20,41	16,62	1,04	94,78	80,62	26,13
		60	95,25	64,06	3,43	4,21	2,60	0,01	94,62	64,48	1,84
		100	95,31	42,58	0,06	0,56	0,26	0,00	94,73	40,24	0,03
		200	95,31	8,60	0,00	0,00	0,00	0,00	94,90	5,89	0,00
	50	30	95,48	8,19	35,06	20,41	0,10	1,17	94,78	5,77	30,59
		60	95,25	0,03	5,05	4,21	0,00	0,01	94,62	0,02	2,93
		100	95,31	0,00	0,13	0,56	0,00	0,00	94,73	0,00	0,02
		200	95,31	0,00	0,00	0,00	0,00	0,00	94,90	0,00	0,00
8	10	30	95,56	83,29	33,97	10,77	8,86	0,97	94,63	84,16	31,23
		60	95,12	71,35	5,35	1,24	0,78	0,00	94,38	71,83	3,04
		100	94,42	52,81	0,17	0,06	0,03	0,00	94,35	50,29	0,05
		200	89,14	16,28	0,00	0,00	0,00	0,00	94,36	10,91	0,00
	50	30	95,56	9,21	39,28	10,77	0,06	0,99	94,63	6,26	35,33
		60	95,12	0,08	7,37	1,24	0,00	0,00	94,38	0,01	4,30
		100	94,42	0,00	0,27	0,06	0,00	0,00	94,35	0,00	0,07
		200	89,14	0,00	0,00	0,00	0,00	0,00	94,36	0,00	0,00
9	10	30	95,25	85,49	40,97	3,93	2,83	0,61	94,23	85,03	37,91
		60	92,32	74,10	9,02	0,13	0,06	0,00	93,77	74,13	5,64
		100	84,05	54,02	0,36	0,00	0,00	0,00	93,34	54,18	0,13
		200	45,03	12,63	0,00	0,00	0,00	0,00	91,76	13,27	0,00
	50	30	95,63	7,73	45,64	3,93	0,00	0,52	94,23	5,51	42,38
		60	92,32	0,03	11,40	0,13	0,00	0,00	93,77	0,04	7,30
		100	84,05	0,00	0,75	0,00	0,00	0,00	93,34	0,00	0,17
		200	45,03	0,00	0,00	0,00	0,00	0,00	91,76	0,00	0,00

Tabela A.18: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim$ KI$(0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	14,12	95,48	61,44	0,01	20,41	7,22	5,98	94,78	56,44
		60	0,03	95,25	16,65	0,00	4,21	0,00	0,00	94,62	8,17
		100	0,00	95,31	0,88	0,00	0,56	0,00	0,00	94,73	0,11
		200	0,00	95,31	0,00	0,00	0,00	0,00	0,00	94,90	0,00
	50	30	0,00	94,17	0,01	0,00	5,54	0,00	0,00	94,41	0,00
		60	0,00	93,85	0,00	0,00	0,26	0,00	0,00	94,29	0,00
		100	0,00	92,02	0,00	0,00	0,00	0,00	0,00	94,26	0,00
		200	0,00	87,26	0,00	0,00	0,00	0,00	0,00	93,54	0,00
5	10	30	14,26	95,56	53,36	0,01	10,77	3,46	6,56	94,63	48,68
		60	0,01	95,12	11,87	0,00	1,24	0,01	0,00	94,38	5,85
		100	0,00	94,42	0,53	0,00	0,06	0,00	0,00	94,35	0,05
		200	0,00	89,14	0,00	0,00	0,00	0,00	0,00	94,36	0,00
	50	30	0,00	95,08	0,00	0,00	2,78	0,00	0,00	94,41	0,00
		60	0,00	93,55	0,00	0,00	0,08	0,00	0,00	94,09	0,00
		100	0,00	91,15	0,00	0,00	0,00	0,00	0,00	93,81	0,00
		200	0,00	81,72	0,00	0,00	0,00	0,00	0,00	92,93	0,00
6	10	30	11,42	95,25	37,37	0,00	3,93	1,03	5,16	94,23	32,30
		60	0,02	92,32	4,21	0,00	0,13	0,00	0,00	93,77	1,77
		100	0,00	84,05	0,07	0,00	0,00	0,00	0,00	93,34	0,00
		200	0,00	45,03	0,00	0,00	0,00	0,00	0,00	91,76	0,00
	50	30	0,00	94,38	0,00	0,00	0,96	0,00	0,00	94,01	0,00
		60	0,00	91,00	0,00	0,00	0,01	0,00	0,00	93,45	0,00
		100	0,00	81,44	0,00	0,00	0,00	0,00	0,00	92,82	0,00
		200	0,00	41,65	0,00	0,00	0,00	0,00	0,00	90,23	0,00

Tabela A.19: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$.

infl.	ϕ	n	$y \sim \text{KI}(0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
7	10	30	84,97	95,48	76,58	14,87	20,41	11,86	87,26	94,78	75,66
		60	62,84	95,25	56,90	0,49	4,21	1,22	63,95	94,62	52,49
		100	33,70	95,31	32,19	0,00	0,56	0,05	31,19	94,73	23,96
		200	2,94	95,31	2,93	0,00	0,00	0,00	1,31	94,90	1,13
	50	30	7,59	95,48	65,53	0,00	20,41	16,05	2,47	94,78	64,09
		60	0,01	95,25	35,70	0,00	4,21	1,41	0,00	94,62	60,96
		100	0,00	95,31	11,34	0,00	0,56	0,03	0,00	94,73	7,90
		200	0,00	95,31	0,33	0,00	0,00	0,00	0,00	94,90	0,10
8	10	30	88,26	95,56	74,77	7,82	10,77	7,06	90,53	94,63	73,76
		60	72,92	95,12	53,58	0,16	1,24	0,55	74,10	94,38	49,62
		100	47,39	94,42	28,93	0,00	0,06	0,01	44,23	94,35	22,51
		200	6,28	89,14	2,55	0,00	0,00	0,00	3,65	94,36	0,96
	50	30	8,02	95,56	40,63	0,00	10,77	5,87	2,53	94,63	35,73
		60	0,00	95,12	9,58	0,00	1,24	0,13	0,00	94,38	6,74
		100	0,00	94,42	0,96	0,00	0,06	0,00	0,00	94,35	0,50
		200	0,00	89,14	0,00	0,00	0,00	0,00	0,00	94,36	0,00
9	10	30	91,23	95,25	69,88	2,93	3,93	3,07	92,61	94,23	67,79
		60	79,08	92,32	44,78	0,01	0,13	0,09	81,11	93,77	41,27
		100	53,34	84,05	20,54	0,00	0,00	0,00	55,16	93,34	16,00
		200	4,12	45,03	1,28	0,00	0,00	0,00	5,38	91,76	0,50
	50	30	4,62	95,25	11,54	0,00	3,93	0,86	1,43	94,23	7,80
		60	0,00	92,32	0,44	0,00	0,13	0,00	0,00	93,77	0,14
		100	0,00	84,05	0,01	0,00	0,00	0,00	0,00	93,34	0,00
		200	0,00	45,03	0,00	0,00	0,00	0,00	0,00	91,76	8,00

Tabela A.20: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,5$.

Cenários	ϕ	n	$y \sim$ GUI$(0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
4	10	30	91,79	38,22	95,48	18,42	2,62	20,41	92,89	37,79	94,78
		60	81,89	8,10	95,25	1,61	0,03	4,21	80,76	6,30	94,62
		100	61,45	0,62	95,31	0,01	0,00	0,56	54,29	0,32	94,73
		200	15,78	0,00	95,31	0,00	0,00	0,00	7,00	0,00	94,90
	50	30	92,10	0,05	95,48	18,56	0,00	20,41	93,14	0,01	94,78
		60	82,90	0,00	95,25	1,71	0,00	4,21	82,02	0,00	94,62
		100	63,86	0,00	95,31	0,02	0,00	0,56	57,30	0,00	94,73
		200	18,73	0,00	95,31	0,00	0,00	0,00	8,94	0,00	94,90
5	10	30	92,53	25,39	95,56	9,54	0,45	10,77	93,18	24,40	94,63
		60	83,76	3,00	95,12	0,34	0,00	1,24	82,95	2,01	94,38
		100	65,12	0,11	94,42	0,00	0,00	0,06	59,97	0,04	94,35
		200	19,493	0,00	89,14	0,00	0,00	0,00	9,72	0,00	94,36
	50	30	92,72	0,00	95,56	9,56	0,00	10,77	93,43	0,00	94,63
		60	84,57	0,00	95,12	0,36	0,00	1,24	84,15	0,00	94,38
		100	67,36	0,00	94,42	0,00	0,00	0,06	62,76	0,00	94,35
		200	22,39	0,00	89,93	0,00	0,00	0,00	12,11	0,00	94,36
6	10	30	92,68	9,72	95,25	3,28	0,00	3,93	93,35	8,61	94,23
		60	83,53	0,20	92,32	0,02	0,00	0,13	85,41	0,15	93,77
		100	62,33	0,00	84,05	0,0	0,00	0,00	65,21	0,00	93,34
		200	9,82	0,00	45,03	0,00	0,00	0,00	11,68	0,00	91,76
	50	30	92,77	0,00	95,25	3,25	0,00	3,93	93,45	0,00	94,23
		60	83,99	0,00	92,32	0,02	0,00	0,13	86,02	0,00	93,77
		100	63,52	0,00	84,05	0,0	0,00	0,00	67,58	0,00	93,34
		200	10,84	0,00	45,03	0,00	0,00	0,00	14,39	0,00	91,76

Tabela A.21: Resultado da proporção de não rejeição dos testes de hipóteses para dados gerados com média $0,8$.

Cenários	ϕ	n	$y \sim$ GUI$(0,1]$								
			% de não rejeição de KS			% de não rejeição de AD			% de não rejeição de CVM		
			BI	KI	GUI	BI	KI	GUI	BI	KI	GUI
7	10	30	34,60	84,21	95,48	0,25	13,90	20,41	27,401	85,58	94,78
		60	2,13	62,15	95,25	0,00	0,29	4,21	0,43	57,99	94,62
		100	0,01	32,59	95,31	0,00	0,00	0,56	0,00	22,47	94,73
		200	0,00	1,72	95,31	0,00	0,00	0,00	0,00	0,41	94,90
	50	30	45,63	49,37	95,48	0,68	6,05	20,41	36,56	46,64	94,78
		60	3,95	19,62	95,25	0,00	0,25	4,21	0,88	16,46	94,62
		100	0,00	4,25	95,31	0,00	0,00	0,56	0,00	2,87	94,73
		200	0,00	0,03	95,31	0,00	0,00	0,00	0,00	0,00	94,90
8	10	30	41,02	77,94	95,56	0,09	4,73	10,77	35,00	78,44	94,63
		60	4,06	50,94	95,12	0,00	0,03	1,24	1,03	48,35	94,38
		100	0,01	22,25	94,42	0,00	0,00	0,06	0,00	16,73	94,35
		200	0,00	0,56	89,14	0,00	0,00	0,00	0,00	0,16	94,36
	50	30	51,97	24,97	95,56	0,23	0,66	10,77	44,07	20,66	94,63
		60	6,49	3,24	95,12	0,00	0,00	1,24	1,75	2,15	94,63
		100	0,04	0,11	94,42	0,00	0,00	0,06	0,00	0,07	94,35
		200	0,00	0,00	89,14	0,00	0,00	0,00	0,00	0,00	94,36
9	10	30	49,45	65,42	95,25	0,00	0,40	3,93	44,85	65,72	94,23
		60	6,92	30,41	92,32	0,00	0,00	0,13	1,94	30,21	93,77
		100	0,01	5,33	84,05	0,00	0,00	0,00	0,00	5,73	93,34
		200	0,00	0,00	45,03	0,00	0,00	0,00	0,00	0,00	91,76
	50	30	59,29	4,70	95,25	0,01	0,00	3,93	53,53	3,04	94,23
		60	11,18	0,04	92,32	0,00	0,00	0,13	3,20	0,02	93,77
		100	0,08	0,00	84,05	0,00	0,00	0,00	0,00	0,00	93,34
		200	0,00	0,00	45,03	0,00	0,00	0,00	0,00	0,00	91,76

Apêndice B

```
                                    ##ikumafit.R##
source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }

  i01 <- (y==0)+(y==1)

  loglik <- function(theta)
  {
    omega <- theta[1]
```

```
  phi <- theta[2]

  ########
  l2 = 0
  if(u!=0)
  {
    l2a = ifelse((y == 1), log(p), 0)
    l2b = ifelse((y == 0), log(1-p), 0)
    l2 = l2a + l2b
  }

  l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

  l3 = ifelse((i01 == 1), 0,
  log(phi)+
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}

escore <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  delta = log(0.5)/log(1-omega^phi)
  ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
      +1)

  c = ifelse((y == 0), 0, ci)
  c = ifelse((y == 1), 0, c)

  Uomega <- phi*sum(c)

  v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
  (delta-1)*((y^phi)*log(y)/(1-y^phi)))
  v = ifelse((y == 1), 0, v)

  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

```

```
### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

##############

opt <- optim(ini, loglik, escore, # hessian = T,
method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))

# opt <- optim(ini, loglik, escore,
# method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e-9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double.eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
# upper = rep(.Machine$double.xmax,(r+2))
# )

# opt <- optim(ini, loglik, escore)

# print(opt$par)

# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)


if (opt$conv != 0)
warning("FUNCTION DID NOT CONVERGE!")

k <- c()

# print("AQUI1")
# print(gradient(escore, opt$par))

```

```
coef <- c(lambda,p,opt$par)
# coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
# print(coef)
k$coef <- coef
k$conv <- opt$conv
k$loglik <- opt$value
k$counts <- as.numeric(opt$counts[1])

# library(rootSolve)
# library(numDeriv)
# escores<-rbind(
# escore(coef),
# gradient(loglik, coef),
# grad(loglik, coef))
# print(round(escores,5))

# k$Knum<-gradient(escore, coef)

# print(k$Knum)
# k$Knum<-hessian(escore, coef, method="complex")
# print(k$Knum)


# # Expected information matrix
#
# T1 = diag(as.vector( 1/diflink1(k$lambda) ))
# T3 = diag(as.vector( 1/diflink3(k$omega) ))
# T4 = diag(as.vector( 1/diflink4(k$phi) ))
#
# h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
# h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
# delta <- log(0.5)/log(1-k$omega^k$phi)
#
# kapa<- 0.5772156649
# k0<- (pi^2)/6+kapa^2-2*kapa
#
# ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
    log(1-y^k$phi)+1)
#
# b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
# b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
# b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
# b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
#
# b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
#
# si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
```

```
#
# # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
# # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
    phi)) -
# # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
    k$phi^3))
# #
# # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
#
# mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
    +1)-kapa)/((delta-1)*k$phi)) )
#
# L = diag(-1/(k$lambda*(1-k$lambda)))
# V = diag((k$lambda-1)*k$phi^2*h2)
# M = diag(mi)
# S = diag(si)
#
#
# Igg = -t(z1) %*% L %*% (T1^2) %*% z1
# Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
# Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
# Isb = t(Ibs)
# Iss = -t(q1) %*% S %*% (T4^2) %*% q1
#
# if(u!=0)
# {
  # k$pi <- as.vector(coef[(m+1):(m+u)])
  # eta_hat2 <- as.vector(w1%*%k$pi)
  # k$p <- as.vector(linkinv2(eta_hat2))
  # T2 = diag(as.vector( 1/diflink2(k$p) ))
  # P = diag(-k$lambda/(k$p*(1-k$p)))
  # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
  #
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
      ncol=s)),
  # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
      ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
  # )
  #
  # }else{
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
```

```
    # )
    # }
  #
  # k$I <- I
  #
  #
  # ############################
  #
  # # print(-k$Knum)
  # # print(k$I)
  #
  # # vcov <- try(solve(-k$Knum))
  # vcov <- try(solve(k$I))
  # # vcov <- chol2inv(chol(k$I))
  # #vcov <- K #somente para nao trancar simulacao
  #
  # k$cov_ok = 0
  # if (class(vcov) == "try-error")
  # {
    # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
        with inversion
    # k$cov_ok = 1
    #
    # if (class(vcov) == "try-error") #
    # {
      # vcov <- k$Knum^2
      # warning("Problem with the variance and covariance matrix. The p-values
          are not correct.")
      # k$cov_ok = 2
      # }
    # }


  # k$vcov <- vcov
  # stderror <- sqrt(diag(k$vcov))
  # k$stderror <- stderror

  return(k)
}
```

```
                                ##iuga_fitv4.R##
  source("iuga_v2.R")
  # library(optimx)
  library(rootSolve)
  library(VGAM)
  # library(fitdistrplus)

```

```
8
9  diuGA <- function(y, alpha0, alpha1, gama, phi)
10
11 {
12   n <- length(y) # sample size
13   n0 <- sum(y == 0) # number of zeros
14   n1 <- sum(y == 1) # number of ones
15   m <- n0 + n1
16
17   alpha0 = ifelse(n0 == 0, 0, alpha0)
18   alpha1 = ifelse(n1 == 0, 0, alpha1)
19
20   c <- 1- alpha0*(1-gama)-alpha1*gama
21   mu <- gama*(1-alpha1)/c
22
23
24   f0 <- ifelse((y == 0), alpha0*(1-gama), 0)
25   f1 <- ifelse((y == 1), alpha1*gama, 0)
26
27   f <- ifelse((y == 0), 0, (1- alpha0*(1-gama)-alpha1*gama)*dGU(y,(gama*(1-
         alpha1))/(1-alpha0*(1-gama)-alpha1*gama),phi))
28   f <- ifelse((y == 1), 0, f)
29
30   return (f0 + f1 + f)
31
32 }
33
34 iuGAfit <- function(y)
35
36 {
37
38   n <- length(y) # sample size
39   n0 <- sum(y == 0) # number of zeros
40   n1 <- sum(y == 1) # number of ones
41   m <- n0 + n1
42
43   ys01 <- y
44   if(any(y==0)) ys01<- ys01[-which(y==0)]
45   if(any(y==1)) ys01<- ys01[-which(ys01==1)]
46
47   i01 <- (y==0)+(y==1)
48
49   #Maxima verossimilhanca
50
51   loglik <- function (par){
52
53     alpha0 <- par[1]
```

```
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

    alpha0 = ifelse(n0 == 0, 0, alpha0)
    alpha1 = ifelse(n1 == 0, 0, alpha1)

    c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
    mu <- gama * (1 - alpha1) / c
    d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))

    l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
    l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
    l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
    l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))
        ))

    #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))

    sum(l0 + l1 + l2 + l3)

  }
  escore <- function (par) {

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

    alpha0 <- ifelse(n0 == 0, 0, alpha0)
    alpha1 <- ifelse(n1 == 0, 0, alpha1)

    c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
    mu <- gama * (1 - alpha1) / c
    d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
    ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
    mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)


    Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
    Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d
        ) / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) /
        (c ^ 2))
    Ualpha0 <- sum(Ualpha01 + Ualpha03)

    Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)

```

```
  Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
  Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d))
      / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c
       ^ 2))
  Ualpha1 <- sum(Ualpha11 + Ualpha13)

  Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)


  Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
  Ugama02 <- ifelse((y == 1), 1 / gama, 0)
  Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d *
      (1 + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1)
      / (c ^ 2))

  Ugama <- sum(Ugama01 + Ugama02 + Ugama04)

  Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu
      ))/phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) -
      digamma(phi) - log(mu^(1/phi)))

  Uphi = sum(Uphi0)


  c(Ualpha0, Ualpha1, Ugama, Uphi)
}


#Inicializacao

y_bar <- mean(y)
y_bar0 <- mean(ys01)

delta0 <- n0/n
delta1 <- n1/n

a0 <- delta0/(1-y_bar)
a1 <- delta1/(y_bar)


#fit <- vglm(ys01 ~ 1, gamma2)
#par<-Coef(fit)

#mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
#phi_til <- -mean(log(ys01))*(mu/(1 - mu))
#phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
#phi_til = 5
```

```
    phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))

    par_ini <- c(a0, a1, y_bar, phi_til)


    max<-optim(par_ini, loglik, escore,method = "BFGS",
    #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
    #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
    control=list(fnscale=-1))

    if(max$convergence != 0) print("Non-convergence")
    # return
    z <- c()
    z$ini = max$par_ini
    z$mv <- max$par
    z$conv <- max$convergence
    z$loglik <- max$value
    return(z)
  }
```

```
                                ##ibetafit.R##
ibetafit<-function(x)
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

  xm0<-x[which(x>0)]
  x01<-xm0[which(xm0<1)]

  #MV
  loglik<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
    phi<-par[4]

    alpha0 = ifelse(n0==0,0,alpha0)
    alpha1 = ifelse(n1==0,0,alpha1)

    c<- 1- alpha0*(1-gama)-alpha1*gama
    mu<- gama*(1-alpha1)/c
```

```
26
27
28    l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
29    l1 = ifelse( (x == 1), log(alpha1*gama), 0)
30    l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
31    l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
          - mu) * phi, log = TRUE)))
32
33    #print("soma")
34    #print(sum(l0 + l1 + l2+ l3))
35    sum(l0 + l1 + l2+ l3)
36  }
37
38  escore<-function(par)
39  {
40    alpha0<-par[1]
41    alpha1<-par[2]
42    gama<-par[3]
43    phi<-par[4]
44
45    c<- 1- alpha0*(1-gama)-alpha1*gama
46    mu<- (gama*(1-alpha1))/c
47
48    a= mu * phi
49    b = a * (1 - mu)/mu
50
51    ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
52    mustar <- digamma(a) - digamma(b)
53
54    Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
55    #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
56    Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
          mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
57    Ualpha0 = sum(Ualpha01 + Ualpha03)
58
59    Ualpha0 = ifelse(n0==0,0,Ualpha0)
60
61
62    Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
63    #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
64    Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
          gama*(1-gama)*(1-alpha0))/ (c^2) ) )
65    #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
          digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
66    Ualpha1 = sum(Ualpha11 - Ualpha13)
67
68    Ualpha1 = ifelse(n1==0,0,Ualpha1)
```

```
69
70    Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
71    Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
72    Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
          mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
73
74    Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
75
76    Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
77    #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
          digamma(b))
78    mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
79    )
80    Uphi = sum(Uphi0)
81
82
83    c(Ualpha0,Ualpha1,Ugama,Uphi)
84  }
85
86  # metodo dos momentos para iniciarlizacao
87  x_bar<-mean(x01)
88  sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
89
90  p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
91  q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
92
93  #print(c(x_bar,sigma2_hat,p,q))
94
95  delta0<- n0/n
96  delta1<- n1/n
97
98  a0<- delta0/(1-x_bar)
99  a1<- delta1/(x_bar)
100
101 #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
102 par_ini<-c(a0,a1,p/(p+q),p+q)
103
104 #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
105
106 #print("par_ini")
107 #print(par_ini)
108
109 max<-optim(par_ini,loglik,
110 escore,
111 method="BFGS",
112 #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
```

```
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z<-c()
  z$mv<-max$par
  #print(z$mv)
  z$conv<-max$convergence
  z$loglik <- max$value
  return(z)
}
```

```
                                    ##iuga_v2.R##
rm(list=ls(all=TRUE))
# density function
library(extraDistr)
library(expint)

# if p=0 then y is zero inflated
# if p=1 then y is one inflated

# funcao densidade de probabilidade da gama unit?ria

dGU <- function (y, mu, phi) {

  d <- (mu ^ (1 / phi)) / (1 - mu ^ (1 / phi))

  (d ^ phi) / gamma(phi) * y ^ (d - 1) * (log(1 / y) ^ (phi - 1))

}

#funcao distribuicao acumulada da gama unitaria
pGU <- function (y, mu, phi) {
  alpha <- phi
  beta <- (mu ^ (1 / phi)) / (1 - mu ^ (1 / phi))
  p <- 1 - pgamma(beta * (-log(y)), alpha, lower.tail = TRUE)
  return(p)
}

#funcao densidade de probabilidade da gama unitaria inflacionada
dgui <- function (y, alpha0 = 0.2, alpha1 = 0.2, gama = 0.5, phi = 5) {

  f0 <- ifelse((y == 0), alpha0*(1-gama), 0)
  f1 <- ifelse((y == 1), alpha1*gama, 0)

  # f <- ifelse((y == 0), 0, (1-alpha0*(1-gama) - alpha1*gama)*dGU(y,phi,(gama*
      (1-alpha1))/(1-alpha0*(1-gama)-alpha1*gama)))
```

```
  # f <- ifelse((y == 1), 0, f)
  c = 1-alpha0*(1-gama) - alpha1*gama
  f = ifelse(((y == 0) | (y == 1)), 0, c*suppressWarnings(dGU(y,(gama*(1-alpha1))
      /(1-alpha0*(1-gama)-alpha1*gama),phi)))

  return (f0 + f1 + f)

}

#funcao distribuicao acumulada da gama unitaria inflacionada
pgui <- function (y, alpha0 = 0.2, alpha1 = 0.2, gama = 0.5, phi = 5) {

  mu <- (gama * (1 - alpha1)) / (1 - alpha0 * (1 - gama) - alpha1 * gama)

  p0 <- alpha0 * (1 - gama)
  p1 <- ifelse(y == 1, alpha1 * gama, 0)
  p <- (1 - alpha0 * (1 - gama) - alpha1 * gama) * pGU(y, mu, phi)
  ret <- ifelse(y == 1, p0 + p1 + p, p0 + p)

  return (ret)

}

# Gu's quantile function
qGU <- function (u, mu, phi) {
  alpha <- phi
  beta <- mu ^ (1 / phi) / (1 - mu ^ (1 / phi))
  qGA <- qgamma(1-u, scale = 1 / beta, shape = alpha, lower.tail = TRUE)
  exp(-qGA)
}

# quantile function iGU

qiGU<-function(u, alpha0 = 0.2, alpha1 = 0.2, gama = 0.5, phi = 5)

{
  if(u < (alpha0*(1-gama)) )
  {
    ret=0

  }else{
    if(u > (1- alpha1*gama) )
    {
      ret=1

    }else{
      ret= qGU((u-alpha0*(1-gama))/(1-alpha0*(1-gama)-alpha1*gama),(gama * (1 -
```

```
          alpha1)) / (1 - alpha0 * (1 - gama) - alpha1 * gama), phi)
    }
  }
  ret
}

# inversion method for randon generation
riGU<-function(n, alpha0 = 0.2, alpha1 = 0.2, gama = 0.5, phi = 5)
{
  ret <- rep(0, n)

  for(i in 1:n)
  {
    u <- runif(1)

    ret[i] <- qiGU(u, alpha0 = alpha0, alpha1 = alpha1, gama = gama, phi = phi)
  }
  ret
}
```

```
                                ##ikum-omega-phi.R##
# density function
library(extraDistr)

# if p=0 then y is zero inflated
# if p=1 then y is one inflated

# density function
dikum<-function(y,lambda=0.2,p=0.2,omega=0.5,phi=5)
{
  f0 = ifelse((y == 0), lambda*(1-p), 0)
  f1 = ifelse((y == 1), lambda*p, 0)

  f = ifelse((y == 0), 0, (1-lambda)*dkumar(y,phi,log(0.5)/log(1-omega^phi)))
  f = ifelse((y == 1), 0, f)

  (f0 + f1 + f)
}

# cumulative distribution function
pikum<-function(y,lambda=0.2,p=0.2,omega=0.5,phi=5)
{
  p0 = lambda*(1-p)
  p1 = ifelse((y == 1), lambda*(p), 0)
  p<- (1-lambda)*(1-(1-(y^phi))^( (log(0.5))/(log(1-(omega^phi))) ))
  ret = ifelse(y==1, p0+p1+p, p0+p)
  ret
```

```
}

# quantile function
qikum<-function(u,lambda=0.2,p=0.2,omega=0.5,phi=5)
{
  if(u < (lambda*(1-p)) )
  {
    ret=0
  }else{
    if(u > (1-lambda*p) )
    {
      ret=1
    }else{
      ret=qkumar((u-lambda*(1-p))/(1-lambda), phi, log(0.5)/log(1-omega^phi))
    }
  }
  ret
}

# inversion method for randon generation
rikum<-function(n,lambda=0.2,p=0.2,omega=0.5,phi=5)
{
  ret<-rep(0,n)
  for(i in 1:n)
  {
    u <- runif(1)
    ret[i]<-qikum(u,lambda=lambda,p=p,omega=omega,phi=phi)
  }
  ret
}
```

```
  ##Simulacao gerando amostras a partir da beta inflacionada em zero##
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
library(goftest)

source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")


set.seed(33) #Fixando a semente

```

```r

source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }

  i01 <- (y==0)+(y==1)

  loglik <- function(theta)
  {
    omega <- theta[1]
    phi <- theta[2]

    ########
    l2 = 0
    if(u!=0)
    {
      l2a = ifelse((y == 1), log(p), 0)
      l2b = ifelse((y == 0), log(1-p), 0)
      l2 = l2a + l2b
    }
```

```
64
65    l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))
66
67    l3 = ifelse((i01 == 1), 0,
68    log(phi)+
69    log(log(0.5)/log(1-omega^phi))+
70    (phi-1)*log(y)+
71    ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
72    )
73
74    sum(l1 + l2 + l3)
75  }
76
77  escore <- function(theta)
78  {
79    omega <- theta[1]
80    phi <- theta[2]
81
82    delta = log(0.5)/log(1-omega^phi)
83    ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
          +1)
84
85    c = ifelse((y == 0), 0, ci)
86    c = ifelse((y == 1), 0, c)
87
88    Uomega <- phi*sum(c)
89
90    v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
91    (delta-1)*((y^phi)*log(y)/(1-y^phi)))
92    v = ifelse((y == 1), 0, v)
93
94    Uphi <- sum(v)
95
96    derivada=c(Uomega,Uphi)
97    derivada
98  }
99
100 ### initial values
101 ys01<- y
102 if(any(y==0)) ys01<- ys01[-which(y==0)]
103 if(any(y==1)) ys01<- ys01[-which(ys01==1)]
104
105 # fit <- vglm(ys01 ~ 1, kumar)
106 # par<-Coef(fit)
107
108 phi <- 5 #par[1]
109 omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)
```

```
110
111 #print(c(lambda,p,omega,phi))
112 ini <- c(omega,phi)
113
114 #############
115
116 opt <- optim(ini, loglik, escore, # hessian = T,
117 method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))
118
119 # opt <- optim(ini, loglik, escore,
120 # method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e
    -9),
121 # # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double
    .eps),
122 # # upper = rep(.Machine$double.xmax,(r+2))
123 # lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
124 # upper = rep(.Machine$double.xmax,(r+2))
125 # )
126
127 # opt <- optim(ini, loglik, escore)
128
129 # print(opt$par)
130
131 # print(names(opt))
132 # print(summary(opt) )
133 # print(summary(opt)[1:6] )
134 # print(opt$kkt1)
135
136
137 if (opt$conv != 0)
138 warning("FUNCTION DID NOT CONVERGE!")
139
140 k <- c()
141
142 # print("AQUI1")
143 # print(gradient(escore, opt$par))
144
145 coef <- c(lambda,p,opt$par)
146 # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
147 # print(coef)
148 k$coef <- coef
149 k$conv <- opt$conv
150 k$loglik <- opt$value
151 k$counts <- as.numeric(opt$counts[1])
152
153 # library(rootSolve)
154 # library(numDeriv)
```

```
155  # escores<-rbind(
156  # escore(coef),
157  # gradient(loglik, coef),
158  # grad(loglik, coef))
159  # print(round(escores,5))
160
161  # k$Knum<-gradient(escore, coef)
162
163  # print(k$Knum)
164  # k$Knum<-hessian(escore, coef, method="complex")
165  # print(k$Knum)
166
167
168  # # Expected information matrix
169  #
170  # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
171  # T3 = diag(as.vector( 1/diflink3(k$omega) ))
172  # T4 = diag(as.vector( 1/diflink4(k$phi) ))
173  #
174  # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
175  # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
176  # delta <- log(0.5)/log(1-k$omega^k$phi)
177  #
178  # kapa<- 0.5772156649
179  # k0<- (pi^2)/6+kapa^2-2*kapa
180  #
181  # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
         log(1-y^k$phi)+1)
182  #
183  # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
184  # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
185  # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
186  # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
187  #
188  # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
189  #
190  # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
191  #
192  # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
193  # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
         phi)) -
194  # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
         k$phi^3))
195  # #
196  # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
197  #
198  # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
```

```
    +1)-kapa)/((delta-1)*k$phi)) )
#
# L = diag(-1/(k$lambda*(1-k$lambda)))
# V = diag((k$lambda-1)*k$phi^2*h2)
# M = diag(mi)
# S = diag(si)
#
#
# Igg = -t(z1) %*% L %*% (T1^2) %*% z1
# Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
# Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
# Isb = t(Ibs)
# Iss = -t(q1) %*% S %*% (T4^2) %*% q1
#
# if(u!=0)
# {
  # k$pi <- as.vector(coef[(m+1):(m+u)])
  # eta_hat2 <- as.vector(w1%*%k$pi)
  # k$p <- as.vector(linkinv2(eta_hat2))
  # T2 = diag(as.vector( 1/diflink2(k$p) ))
  # P = diag(-k$lambda/(k$p*(1-k$p)))
  # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
  #
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
      ncol=s)),
  # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
      ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
  # )
  #
  # }else{
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
  # )
  # }
#
# k$I <- I
#
#
# ############################
#
# # print(-k$Knum)
# # print(k$I)
```

```
243   #
244   # # vcov <- try(solve(-k$Knum))
245   # vcov <- try(solve(k$I))
246   # # vcov <- chol2inv(chol(k$I))
247   # #vcov <- K #somente para nao trancar simulacao
248   #
249   # k$cov_ok = 0
250   # if (class(vcov) == "try-error")
251   # {
252     # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
          with inversion
253     # k$cov_ok = 1
254     #
255     # if (class(vcov) == "try-error") #
256     # {
257       # vcov <- k$Knum^2
258       # warning("Problem with the variance and covariance matrix. The p-values
            are not correct.")
259       # k$cov_ok = 2
260       # }
261     # }
262
263
264   # k$vcov <- vcov
265   # stderror <- sqrt(diag(k$vcov))
266   # k$stderror <- stderror
267
268   return(k)
269 }
270
271 ibetafit<-function(x)
272 {
273
274   n<- length(x) # sample size
275   n0<- sum(x==0) # number of zeros
276   n1<- sum(x==1) # number of ones
277   m<- n0 + n1
278
279   xm0<-x[which(x>0)]
280   x01<-xm0[which(xm0<1)]
281
282   #MV
283   loglik<-function(par)
284   {
285     alpha0<-par[1]
286     alpha1<-par[2]
287     gama<-par[3]
```

```
288     phi<-par[4]
289
290     alpha0 = ifelse(n0==0,0,alpha0)
291     alpha1 = ifelse(n1==0,0,alpha1)
292
293     #print(c(alpha0,alpha1,gama,phi))
294
295     c<- 1- alpha0*(1-gama)-alpha1*gama
296     mu<- gama*(1-alpha1)/c
297
298
299     l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
300     l1 = ifelse( (x == 1), log(alpha1*gama), 0)
301     l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
302     l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
          - mu) * phi, log = TRUE)))
303
304
305     #print("soma")
306     #print(sum(l0 + l1 + l2+ l3))
307     sum(l0 + l1 + l2+ l3)
308   }
309
310   escore<-function(par)
311   {
312     alpha0<-par[1]
313     alpha1<-par[2]
314     gama<-par[3]
315     phi<-par[4]
316
317     c<- 1- alpha0*(1-gama)-alpha1*gama
318     mu<- (gama*(1-alpha1))/c
319
320     a= mu * phi
321     b = a * (1 - mu)/mu
322
323     ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
324     mustar <- digamma(a) - digamma(b)
325
326
327
328     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
329     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
330     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
          mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
331     Ualpha0 = sum(Ualpha01 + Ualpha03)
332
```

```
333   Ualpha0 = ifelse(n0==0,0,Ualpha0)
334
335
336   Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
337   #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
338   Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
          gama*(1-gama)*(1-alpha0))/ (c^2) ) )
339   #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
          digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
340   Ualpha1 = sum(Ualpha11 - Ualpha13)
341
342   Ualpha1 = ifelse(n1==0,0,Ualpha1)
343
344   Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
345   Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
346   Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
          mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
347
348   Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
349
350   Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
351   #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
          digamma(b))
352   mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
353   )
354   Uphi = sum(Uphi0)
355
356
357   c(Ualpha0,Ualpha1,Ugama,Uphi)
358 }
359
360 # metodo dos momentos para iniciarlizacao
361 x_bar<-mean(x01)
362 sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
363
364 p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
365 q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
366
367 #print(c(x_bar,sigma2_hat,p,q))
368
369 delta0<- n0/n
370 delta1<- n1/n
371
372 a0<- delta0/(1-x_bar)
373 a1<- delta1/(x_bar)
374
375 #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
```

```
  par_ini<-c(a0,a1,p/(p+q),p+q)

  #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)

  #print("par_ini")
  #print(par_ini)

  max<-optim(par_ini,loglik,
  escore,
  method="BFGS",
  #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z<-c()
  z$mv<-max$par
  #print(z$mv)
  z$conv<-max$convergence
  z$loglik <- max$value
  return(z)
}


iuGAfit <- function(y)

{

  n <- length(y) # sample size
  n0 <- sum(y == 0) # number of zeros
  n1 <- sum(y == 1) # number of ones
  m <- n0 + n1

  ys01 <- y
  if(any(y==0)) ys01<- ys01[-which(y==0)]
  if(any(y==1)) ys01<- ys01[-which(ys01==1)]

  i01 <- (y==0)+(y==1)

  #Maxima verossimilhanca

  loglik <- function (par){

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
```

```
422     phi <- par[4]
423
424     alpha0 = ifelse(n0 == 0, 0, alpha0)
425     alpha1 = ifelse(n1 == 0, 0, alpha1)
426
427     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
428     mu <- gama * (1 - alpha1) / c
429     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
430
431     l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
432     l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
433     l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
434     l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))
435
436     #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))
437
438     sum(l0 + l1 + l2 + l3)
439
440   }
441   escore <- function (par) {
442
443     alpha0 <- par[1]
444     alpha1 <- par[2]
445     gama <- par[3]
446     phi <- par[4]
447
448     alpha0 <- ifelse(n0 == 0, 0, alpha0)
449     alpha1 <- ifelse(n1 == 0, 0, alpha1)
450
451     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
452     mu <- gama * (1 - alpha1) / c
453     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
454     ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
455     mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
456
457
458     Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
459     Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
          / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
          ^ 2))
460     Ualpha0 <- sum(Ualpha01 + Ualpha03)
461
462     Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
463
464     Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
465     Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
          (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
```

```
      2))
  Ualpha1 <- sum(Ualpha11 + Ualpha13)

  Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)


  Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
  Ugama02 <- ifelse((y == 1), 1 / gama, 0)
  Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
      + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
       ^ 2))

  Ugama <- sum(Ugama01 + Ugama02 + Ugama04)

  Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
      /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
      phi) - log(mu^(1/phi)))

  Uphi = sum(Uphi0)


  c(Ualpha0, Ualpha1, Ugama, Uphi)
}


#Inicializacao

y_bar <- mean(y)
y_bar0 <- mean(ys01)

delta0 <- n0/n
delta1 <- n1/n

a0 <- delta0/(1-y_bar)
a1 <- delta1/(y_bar)


#fit <- vglm(ys01 ~ 1, gamma2)
#par<-Coef(fit)

#mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
#phi_til <- -mean(log(ys01))*(mu/(1 - mu))
#phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
#phi_til = 5
phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))

par_ini <- c(a0, a1, y_bar, phi_til)
```

```
508
509
510   max<-optim(par_ini, loglik, escore,method = "BFGS",
511   #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
512   #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
513   control=list(fnscale=-1))
514
515   if(max$convergence != 0) print("Non-convergence")
516   # return
517   z <- c()
518   z$ini <- par_ini
519   z$mv <- max$par
520   z$conv <- max$convergence
521   z$loglik <- max$value
522   return(z)
523 }
524
525 n=200
526 #n=60
527 #n=100
528 #n=200
529 NR=10000
530 k=3 # numero de parametros
531
532 Cont_AIC_IB=rep(0,NR)
533 Cont_AIC_IK=rep(0,NR)
534 Cont_AIC_IUG=rep(0,NR)
535
536 Cont_AICC_IB=rep(0,NR)
537 Cont_AICC_IK=rep(0,NR)
538 Cont_AICC_IUG=rep(0,NR)
539
540 Cont_BIC_IB=rep(0,NR)
541 Cont_BIC_IK=rep(0,NR)
542 Cont_BIC_IUG=rep(0,NR)
543
544 Cont_BICC_IB=rep(0,NR)
545 Cont_BICC_IK=rep(0,NR)
546 Cont_BICC_IUG=rep(0,NR)
547
548 Cont_HQ_IB=rep(0,NR)
549 Cont_HQ_IK=rep(0,NR)
550 Cont_HQ_IUG=rep(0,NR)
551
552 Cont_HQC_IB=rep(0,NR)
```

```
Cont_HQC_IK=rep(0,NR)
Cont_HQC_IUG=rep(0,NR)

Cont_LL_IB =rep(0,NR)
Cont_LL_IK=rep(0,NR)
Cont_LL_IUG=rep(0,NR)

contC=0
contC1=0

prop.z = rep(0,NR)
prop.u = rep(0,NR)
media.1=rep(0,NR)#isaacc
sd.1=rep(0,NR) #isaacc
var.am=rep(0,NR) #isaac

cvm.iBE = rep(0,NR)
ks.iBE = rep(0,NR)
ad.iBE = rep(0,NR)

cvm.iuGA = rep(0,NR)
ks.iuGA = rep(0,NR)
ad.iuGA = rep(0,NR)

cvm.ikum = rep(0,NR)
ks.ikum = rep(0,NR)
ad.ikum = rep(0,NR)

alpha0 = 0.1
alpha1 = 0
gama= 0.3
phi = 10

valores.parametros = c(alpha0, alpha1, gama, phi)

prob.z = alpha0*(1-gama)
prob.o = alpha1*gama

prob.z # probabilidade de zero
prob.o #probabilidade de um

media.v = gama #infla_0

var.v=gama*((1+alpha1*phi)/(1+phi)) + (((1-alpha1)*(1-alpha1)*phi)/((1-alpha0*(1-
    gama)-alpha1*gama)*(1+phi)) -1)*gama^(2)
media.v
var.v
```

```
599
600 c= 1- alpha0*(1-gama)-alpha1*gama
601 mu = gama*(1-alpha1)/c
602 delta0 = alpha0 * (1-gama)
603 delta1 = alpha1 * (gama)
604 p2= 1- delta0-delta1
605
606 mu
607 delta0
608 delta1
609 p2
610 dp = sqrt(1/(phi+1))
611 dp
612
613 cont.ib = 0
614 cont.ik = 0
615 cont.igu = 0
616
617 for(i in 1:NR) {
618
619   #y = riGU(n,alpha0 = alpha0, alpha1=alpha1, gama = gama, phi = phi)
620   y = rBEINF0(n, mu = mu , sigma=dp, nu=delta0/p2) # Inf.0
621   len = length(y)
622   prop.z[i] = sum(ifelse(y==0,1,0))/len
623   prop.u[i] = sum(ifelse(y==1,1,0))/len
624   media.1[i]=mean(y)#isaacc
625   sd.1[i]=sd(y)#isaacc
626   var.am[i]=var(y)#isaac
627
628   fitib = ibetafit(y)
629   if (fitib$conv != 0)
630   {
631     cont.ib = cont.ib+1
632   }
633
634   fitik = ikumafit(y)
635   if (fitik$conv != 0)
636   {
637     cont.ik = cont.ik+1
638   }
639
640   fitigu = iuGAfit(y)
641
642
643   if ((fitigu$conv != 0) ){
644     cont.igu=cont.igu+1
645   }
```

```
646
647 ks.iBE[i] = ifelse(ks.test(y, "pBEINFO",mu=mu , sigma=dp, nu=delta0/p2)$p.value
        >0.05,1,0)
648
649 ad.iBE[i] = ifelse(ad.test(y, null="pBEINFO",mu=mu , sigma=dp, nu=delta0/p2)$p.
        value>0.05,1,0)
650
651 cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINFO",mu=mu , sigma=dp, nu=delta0/p2)$
        p.value >0.05,1,0)
652
653 ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
        phi=phi)$p.value>0.05,1,0)
654
655 ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
656
657 cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
658
659
660 ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=alpha0, p=alpha1,omega=gama, phi=
        phi)$p.value>0.05,1,0)
661
662 ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=alpha0, p=alpha1,omega=gama,
         phi=phi)$p.value>0.05,1,0)
663
664 cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=alpha0, p=alpha1,omega=
        gama, phi=phi)$p.value>0.05,1,0)
665
666 #Log verossimilhanca
667 log.vero_IB = fitib$loglik #BI(Beta inflacionada)
668 log.vero_IK= fitik$loglik #KI(Kumaraswamy Inflacionada)
669 log.vero_IUG = fitigu$loglik #GUI(Gama Unitaria Inflacionada)
670
671 m.log.vero_IB = -fitib$loglik #BI
672 m.log.vero_IK= -fitik$loglik #KI
673 m.log.vero_IUG = -fitigu$loglik #GUI
674
675 #Menor -log verossimilhanca
676
677 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
678
679 Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
680 Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
681 Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
682
683
```

```
684  # Calculando os criterios/quantidades para comparacao
685
686  #criteiro_AIC#
687  AIC_IB = -2*log.vero_IB+2*k #BI
688  AIC_IK= -2*log.vero_IK+2*k #KI
689  AIC_IUG = -2*log.vero_IUG+2*k #GUI
690
691  # Selecionando o menor AIC
692  min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
693
694  Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
695  Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
696  Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
697
698  AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1) #BI
699  AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1) #KI
700  AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1) #GUI
701
702  # Selecionando o menor AICC
703  min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
704
705  Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
706  Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
707  Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)
708
709  #criterio_BIC#
710  BIC_IB=-2*log.vero_IB+k*log(n) #BI
711  BIC_IK=-2*log.vero_IK+k*log(n) #KI
712  BIC_IUG=-2*fitigu$loglik+k*log(n) #GUI
713
714  # Selecionando o menor BIC
715  min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))
716
717  Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
718  Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
719  Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)
720
721  #criterio_BICC#
722  BICC_IB=-2*log.vero_IB+(n*k*log(n))/(n-k-1) #BI
723  BICC_IK=-2*log.vero_IK+(n*k*log(n))/(n-k-1) #KI
724  BICC_IUG=-2*fitigu$loglik+(n*k*log(n))/(n-k-1) #GUI
725
726  # Selecionando o menor BICC
727  min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))
728
729  Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
730  Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
```

```
  Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)

  #criterio_HQ#

  HQ_IB=-2*log.vero_IB+2*k*log(log(n)) #BI
  HQ_IK=-2*log.vero_IK+2*k*log(log(n)) #KI
  HQ_IUG=-2*fitigu$loglik+2*k*log(log(n)) #GUI

  # Selecionando o menor HQ
  min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))

  Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
  Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
  Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)

  #criterio_HQC#
  HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1) #BI
  HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1) #KI
  HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1) #GUI

  # Selecionando o menor HQC
  min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))


  Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
  Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
  Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
  # FIM DO LACO DE MONTE CARLO

}

# Proporcao media de zeros
#mean(prop.z)
# Proporcao media de uns
#mean(prop.u)
#media das medias
mu=mean(media.1)
#media do desvio padrao
sd.of=mean(sd.1)
var.amostral=mean(var.am)

#Calculando percentuais escolha distribuicao(AIC)
p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
p_AICIB
p_AICIK
```

```
p_AICIUG

#Calculando percentuais escolha distribuicao(AICC)
p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
p_AICCIB
p_AICCIK
p_AICCIUG

#Calculando percentuais escolha distribuicao(BIC)
p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
p_BICIB
p_BICIK
p_BICIUG

#Calculando percentuais escolha distribuicao(BICC)
p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
p_BICCIB
p_BICCIK
p_BICCIUG

#Calculando percentuais escolha distribuicao(HQ)
p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
p_HQIB
p_HQIK
p_HQIUG

#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG


#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
```

```
p_LLIB
p_LLIK
p_LLIUG

#resultados -log.vero
result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
#Recebendo Resultados(AIC)
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
#Recebendo Resultados(AICC)
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
#Recebendo Resultados(BIC)
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
#Recebendo Resultados(BICC)
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
#Recebendo Resultados(HQ)
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
#Recebendo Resultados(HQC)
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC

cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100

ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100

ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100

cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100

ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100

ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100

cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100

ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100

ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
```

```
    rej.iuGA)

result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)

M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
result.taxa.rej.ad,result.taxa.rej.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
    ks",
"Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")

glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
    .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
"\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
    phi: {phi}","\n", "media: {media.v}",
"\n", "var: {round(var.v,4)}")
print("Resultados")
print(M)

M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
    )
colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
    Rej.cvm")
xtable(M.res)

cv=sd.of/mu
cv
var.teste=sd.of*sd.of
var.teste
#Salvando em um arquivo
#write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
var.amostral
```

```
  ##Simulacao gerando amostras a partir da beta inflacionada em zero e um##
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
library(goftest)
source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")
```

```r
set.seed(33) #Fixando a semente
source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }

  i01 <- (y==0)+(y==1)

  loglik <- function(theta)
  {
    omega <- theta[1]
    phi <- theta[2]

    ########
    l2 = 0
    if(u!=0)
    {
      l2a = ifelse((y == 1), log(p), 0)
      l2b = ifelse((y == 0), log(1-p), 0)
      l2 = l2a + l2b
    }
```

```
59
60    l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))
61
62    l3 = ifelse((i01 == 1), 0,
63    log(phi)+
64    log(log(0.5)/log(1-omega^phi))+
65    (phi-1)*log(y)+
66    ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
67    )
68
69    sum(l1 + l2 + l3)
70  }
71
72  escore <- function(theta)
73  {
74    omega <- theta[1]
75    phi <- theta[2]
76
77    delta = log(0.5)/log(1-omega^phi)
78    ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
          +1)
79
80    c = ifelse((y == 0), 0, ci)
81    c = ifelse((y == 1), 0, c)
82
83    Uomega <- phi*sum(c)
84
85    v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
86    (delta-1)*((y^phi)*log(y)/(1-y^phi)))
87    v = ifelse((y == 1), 0, v)
88
89    Uphi <- sum(v)
90
91    derivada=c(Uomega,Uphi)
92    derivada
93  }
94
95  ### initial values
96  ys01<- y
97  if(any(y==0)) ys01<- ys01[-which(y==0)]
98  if(any(y==1)) ys01<- ys01[-which(ys01==1)]
99
100 # fit <- vglm(ys01 ~ 1, kumar)
101 # par<-Coef(fit)
102
103 phi <- 5 #par[1]
104 omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)
```

```
105
106 #print(c(lambda,p,omega,phi))
107 ini <- c(omega,phi)
108
109 #############
110
111 opt <- optim(ini, loglik, escore, # hessian = T,
112 method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))
113
114 # opt <- optim(ini, loglik, escore,
115 # method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e
    -9),
116 # # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double
    .eps),
117 # # upper = rep(.Machine$double.xmax,(r+2))
118 # lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
119 # upper = rep(.Machine$double.xmax,(r+2))
120 # )
121
122 # opt <- optim(ini, loglik, escore)
123
124 # print(opt$par)
125
126 # print(names(opt))
127 # print(summary(opt) )
128 # print(summary(opt)[1:6] )
129 # print(opt$kkt1)
130
131
132 if (opt$conv != 0)
133 warning("FUNCTION DID NOT CONVERGE!")
134
135 k <- c()
136
137 # print("AQUI1")
138 # print(gradient(escore, opt$par))
139
140 coef <- c(lambda,p,opt$par)
141 # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
142 # print(coef)
143 k$coef <- coef
144 k$conv <- opt$conv
145 k$loglik <- opt$value
146 k$counts <- as.numeric(opt$counts[1])
147
148 # library(rootSolve)
149 # library(numDeriv)
```

```
150 # escores<-rbind(
151 # escore(coef),
152 # gradient(loglik, coef),
153 # grad(loglik, coef))
154 # print(round(escores,5))
155
156 # k$Knum<-gradient(escore, coef)
157
158 # print(k$Knum)
159 # k$Knum<-hessian(escore, coef, method="complex")
160 # print(k$Knum)
161
162
163 # # Expected information matrix
164 #
165 # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
166 # T3 = diag(as.vector( 1/diflink3(k$omega) ))
167 # T4 = diag(as.vector( 1/diflink4(k$phi) ))
168 #
169 # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
170 # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
171 # delta <- log(0.5)/log(1-k$omega^k$phi)
172 #
173 # kapa<- 0.5772156649
174 # k0<- (pi^2)/6+kapa^2-2*kapa
175 #
176 # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
        log(1-y^k$phi)+1)
177 #
178 # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
179 # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
180 # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
181 # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
182 #
183 # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
184 #
185 # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
186 #
187 # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
188 # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
        phi)) -
189 # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
        k$phi^3))
190 # #
191 # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
192 #
193 # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
```

```
    +1)-kapa)/((delta-1)*k$phi)) )
#
# L = diag(-1/(k$lambda*(1-k$lambda)))
# V = diag((k$lambda-1)*k$phi^2*h2)
# M = diag(mi)
# S = diag(si)
#
#
# Igg = -t(z1) %*% L %*% (T1^2) %*% z1
# Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
# Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
# Isb = t(Ibs)
# Iss = -t(q1) %*% S %*% (T4^2) %*% q1
#
# if(u!=0)
# {
  # k$pi <- as.vector(coef[(m+1):(m+u)])
  # eta_hat2 <- as.vector(w1%*%k$pi)
  # k$p <- as.vector(linkinv2(eta_hat2))
  # T2 = diag(as.vector( 1/diflink2(k$p) ))
  # P = diag(-k$lambda/(k$p*(1-k$p)))
  # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
  #
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
      ncol=s)),
  # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
      ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
  # )
  #
  # }else{
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
  # )
  # }
#
# k$I <- I
#
#
# ############################
#
# # print(-k$Knum)
# # print(k$I)
```

```
  #
  # # vcov <- try(solve(-k$Knum))
  # vcov <- try(solve(k$I))
  # # vcov <- chol2inv(chol(k$I))
  # #vcov <- K #somente para nao trancar simulacao
  #
  # k$cov_ok = 0
  # if (class(vcov) == "try-error")
  # {
    # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems with inversion
    # k$cov_ok = 1
    #
    # if (class(vcov) == "try-error") #
    # {
      # vcov <- k$Knum^2
      # warning("Problem with the variance and covariance matrix. The p-values are not correct.")
      # k$cov_ok = 2
      # }
    # }


  # k$vcov <- vcov
  # stderror <- sqrt(diag(k$vcov))
  # k$stderror <- stderror

  return(k)
}

ibetafit<-function(x)
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

  xm0<-x[which(x>0)]
  x01<-xm0[which(xm0<1)]

  #MV
  loglik<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
```

```
283     phi<-par[4]
284
285     alpha0 = ifelse(n0==0,0,alpha0)
286     alpha1 = ifelse(n1==0,0,alpha1)
287
288     #print(c(alpha0,alpha1,gama,phi))
289
290     c<- 1- alpha0*(1-gama)-alpha1*gama
291     mu<- gama*(1-alpha1)/c
292
293
294     l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
295     l1 = ifelse( (x == 1), log(alpha1*gama), 0)
296     l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
297     l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
          - mu) * phi, log = TRUE)))
298
299
300     #print("soma")
301     #print(sum(l0 + l1 + l2+ l3))
302     sum(l0 + l1 + l2+ l3)
303   }
304
305   escore<-function(par)
306   {
307     alpha0<-par[1]
308     alpha1<-par[2]
309     gama<-par[3]
310     phi<-par[4]
311
312     c<- 1- alpha0*(1-gama)-alpha1*gama
313     mu<- (gama*(1-alpha1))/c
314
315     a= mu * phi
316     b = a * (1 - mu)/mu
317
318     ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
319     mustar <- digamma(a) - digamma(b)
320
321
322
323     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
324     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
325     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
          mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
326     Ualpha0 = sum(Ualpha01 + Ualpha03)
327
```

```
328   Ualpha0 = ifelse(n0==0,0,Ualpha0)
329
330
331   Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
332   #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
333   Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
          gama*(1-gama)*(1-alpha0))/ (c^2) ) )
334   #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
          digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
335   Ualpha1 = sum(Ualpha11 - Ualpha13)
336
337   Ualpha1 = ifelse(n1==0,0,Ualpha1)
338
339   Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
340   Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
341   Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
          mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
342
343   Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
344
345   Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
346   #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
          digamma(b))
347   mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
348   )
349   Uphi = sum(Uphi0)
350
351
352   c(Ualpha0,Ualpha1,Ugama,Uphi)
353 }
354
355 # metodo dos momentos para iniciarlizacao
356 x_bar<-mean(x01)
357 sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
358
359 p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
360 q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
361
362 #print(c(x_bar,sigma2_hat,p,q))
363
364 delta0<- n0/n
365 delta1<- n1/n
366
367 a0<- delta0/(1-x_bar)
368 a1<- delta1/(x_bar)
369
370 #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
```

```
371   par_ini<-c(a0,a1,p/(p+q),p+q)
372
373   #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
374
375   #print("par_ini")
376   #print(par_ini)
377
378   max<-optim(par_ini,loglik,
379   escore,
380   method="BFGS",
381   #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
382   control=list(fnscale=-1))
383
384   if(max$convergence != 0) print("Non-convergence")
385   # return
386   z<-c()
387   z$mv<-max$par
388   #print(z$mv)
389   z$conv<-max$convergence
390   z$loglik <- max$value
391   return(z)
392 }
393
394
395 iuGAfit <- function(y)
396
397 {
398
399   n <- length(y) # sample size
400   n0 <- sum(y == 0) # number of zeros
401   n1 <- sum(y == 1) # number of ones
402   m <- n0 + n1
403
404   ys01 <- y
405   if(any(y==0)) ys01<- ys01[-which(y==0)]
406   if(any(y==1)) ys01<- ys01[-which(ys01==1)]
407
408   i01 <- (y==0)+(y==1)
409     #Maxima verossimilhanca
410
411   loglik <- function (par){
412
413     alpha0 <- par[1]
414     alpha1 <- par[2]
415     gama <- par[3]
416     phi <- par[4]
```

```
417
418   alpha0 = ifelse(n0 == 0, 0, alpha0)
419   alpha1 = ifelse(n1 == 0, 0, alpha1)
420
421   c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
422   mu <- gama * (1 - alpha1) / c
423   d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
424
425   l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
426   l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
427   l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
428   l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))
429
430   #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))
431
432   sum(l0 + l1 + l2 + l3)
433
434 }
435 escore <- function (par) {
436
437   alpha0 <- par[1]
438   alpha1 <- par[2]
439   gama <- par[3]
440   phi <- par[4]
441
442   alpha0 <- ifelse(n0 == 0, 0, alpha0)
443   alpha1 <- ifelse(n1 == 0, 0, alpha1)
444
445   c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
446   mu <- gama * (1 - alpha1) / c
447   d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
448   ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
449   mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
450
451
452   Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
453   Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
        / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
        ^ 2))
454   Ualpha0 <- sum(Ualpha01 + Ualpha03)
455
456   Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
457
458   Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
459   Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
        (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
        2))
```

```
460   Ualpha1 <- sum(Ualpha11 + Ualpha13)
461
462   Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
463
464
465   Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
466   Ugama02 <- ifelse((y == 1), 1 / gama, 0)
467   Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
          + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
           ^ 2))
468
469   Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
470
471   Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
          /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
          phi) - log(mu^(1/phi)))
472
473   Uphi = sum(Uphi0)
474
475
476   c(Ualpha0, Ualpha1, Ugama, Uphi)
477 }
478
479 #Inicializacao
480 y_bar <- mean(y)
481 y_bar0 <- mean(ys01)
482
483 delta0 <- n0/n
484 delta1 <- n1/n
485
486 a0 <- delta0/(1-y_bar)
487 a1 <- delta1/(y_bar)
488
489
490 #fit <- vglm(ys01 ~ 1, gamma2)
491 #par<-Coef(fit)
492
493 #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
494 #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
495 #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
496 #phi_til = 5
497 phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
498
499 par_ini <- c(a0, a1, y_bar, phi_til)
500
501
502 max<-optim(par_ini, loglik, escore,method = "BFGS",
```

```
503   #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
504   #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
505   control=list(fnscale=-1))
506
507   if(max$convergence != 0) print("Non-convergence")
508   # return
509   z <- c()
510   z$ini <- par_ini
511   z$mv <- max$par
512   z$conv <- max$convergence
513   z$loglik <- max$value
514   return(z)
515 }
516
517 n=100
518 #n=60
519 #n=100
520 #n=200
521 NR=10000
522 k=4 # numero de parametros
523
524 Cont_AIC_IB=rep(0,NR)
525 Cont_AIC_IK=rep(0,NR)
526 Cont_AIC_IUG=rep(0,NR)
527
528 Cont_AICC_IB=rep(0,NR)
529 Cont_AICC_IK=rep(0,NR)
530 Cont_AICC_IUG=rep(0,NR)
531
532 Cont_BIC_IB=rep(0,NR)
533 Cont_BIC_IK=rep(0,NR)
534 Cont_BIC_IUG=rep(0,NR)
535
536 Cont_BICC_IB=rep(0,NR)
537 Cont_BICC_IK=rep(0,NR)
538 Cont_BICC_IUG=rep(0,NR)
539
540 Cont_HQ_IB=rep(0,NR)
541 Cont_HQ_IK=rep(0,NR)
542 Cont_HQ_IUG=rep(0,NR)
543
544 Cont_HQC_IB=rep(0,NR)
545 Cont_HQC_IK=rep(0,NR)
546 Cont_HQC_IUG=rep(0,NR)
547
```

```
Cont_LL_IB =rep(0,NR)
Cont_LL_IK=rep(0,NR)
Cont_LL_IUG=rep(0,NR)

contC=0
contC1=0

prop.z = rep(0,NR)
prop.u = rep(0,NR)
media.1=rep(0,NR)#isaacc
sd.1=rep(0,NR) #isaacc

cvm.iBE = rep(0,NR)
ks.iBE = rep(0,NR)
ad.iBE = rep(0,NR)

cvm.iuGA = rep(0,NR)
ks.iuGA = rep(0,NR)
ad.iuGA = rep(0,NR)

cvm.ikum = rep(0,NR)
ks.ikum = rep(0,NR)
ad.ikum = rep(0,NR)

alpha0 = 0.07
alpha1 = 0.07
gama= 0.8
phi = 10

prob.0=alpha0*(1-gama) #(prob de zero)
prob.1=alpha1*gama #(prob de um)
prob.0
prob.1


lambda = prob.0+prob.1 #(prop.infl ou seja pro.0+prop.1)
lambda
p=(prob.1)/(lambda)
p


valores.parametros = c(alpha0, alpha1, gama, phi)

prob.z = alpha0*(1-gama)
prob.o = alpha1*gama

prob.z # probabilidade de zero
```

```
595 prob.o #probabilidade de um
596
597
598 media.v = gama
599
600 var.v=gama*((1+alpha1*phi)/(1+phi)) + (((1-alpha1)*(1-alpha1)*phi)/((1-alpha0*(1-
        gama)-alpha1*gama)*(1+phi)) -1)*gama^(2)
601 media.v
602 var.v
603
604 c= 1- alpha0*(1-gama)-alpha1*gama
605 mu = gama*(1-alpha1)/c
606 delta0 = alpha0 * (1-gama)
607 delta1 = alpha1 * (gama)
608 p2= 1- delta0-delta1
609
610 mu
611 delta0
612 delta1
613 p2
614 dp = sqrt(1/(phi+1))
615 dp
616
617
618 teste.cv=dp/media.v
619 teste.cv
620 cont.ib = 0
621 cont.ik = 0
622 cont.igu = 0
623
624 for(i in 1:NR) {
625
626   #y = riGU(n,alpha0 = alpha0, alpha1=alpha1, gama = gama, phi = phi)
627   y = rBEINF(n, mu = mu , sigma=dp, nu=delta0/p2, tau=delta1/p2) # Inf.0
628   len = length(y)
629   prop.z[i] = sum(ifelse(y==0,1,0))/len
630   prop.u[i] = sum(ifelse(y==1,1,0))/len
631   media.1[i]=mean(y)#isaacc
632   sd.1[i]=sd(y)#isaacc
633
634   fitib = ibetafit(y)
635   if (fitib$conv != 0)
636   {
637     cont.ib = cont.ib+1
638   }
639
640   fitik = ikumafit(y)
```

```
641  if (fitik$conv != 0)
642  {
643    cont.ik = cont.ik+1
644  }
645
646  fitigu = iuGAfit(y)
647
648  if ((fitigu$conv != 0) ){
649    cont.igu=cont.igu+1
650  }
651
652
653  ks.iBE[i] = ifelse(ks.test(y, "pBEINF",mu=mu , sigma=dp, nu=delta0/p2, tau=
         delta1/p2)$p.value>0.05,1,0)
654
655  ad.iBE[i] = ifelse(ad.test(y, null="pBEINF",mu=mu , sigma=dp, nu=delta0/p2, tau
         =delta1/p2)$p.value>0.05,1,0)
656
657  cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF",mu=mu , sigma=dp, nu=delta0/p2,
         tau=delta1/p2)$p.value >0.05,1,0)
658
659  ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
         phi=phi)$p.value>0.05,1,0)
660
661  ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
         gama, phi=phi)$p.value>0.05,1,0)
662
663  cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
         gama, phi=phi)$p.value>0.05,1,0)
664
665
666  ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=lambda, p=p,omega=gama, phi=phi)$
         p.value>0.05,1,0)
667
668  ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=lambda, p=p,omega=gama, phi=
         phi)$p.value>0.05,1,0)
669
670  cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=lambda, p=p,omega=gama,
         phi=phi)$p.value>0.05,1,0)
671
672  #Log verossimilhanca
673  # Beta inflacionada
674  log.vero_IB = fitib$loglik
675  # Kuma inflacionada
676  log.vero_IK= fitik$loglik
677  # Gamma Unitaria inflacionada
678  log.vero_IUG = fitigu$loglik
```

```
679
680 m.log.vero_IB = -fitib$loglik
681 # Kuma inflacionada
682 m.log.vero_IK= -fitik$loglik
683 # Gamma Unitaria inflacionada
684 m.log.vero_IUG = -fitigu$loglik
685
686 #Menor -log verossimilhanca
687
688 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
689
690 Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
691 Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
692 Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
693
694
695
696
697 # Calculando os criterios/quantidades para comparacao
698
699 #criteiro_AIC#
700 # Beta inflacionada
701 AIC_IB = -2*log.vero_IB+2*k
702 # Kuma inflacionada
703 AIC_IK= -2*log.vero_IK+2*k
704 # Gamma Unitaria inflacionada
705 AIC_IUG = -2*log.vero_IUG+2*k
706
707 # Selecionando o menor AIC
708 min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
709
710 Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
711 Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
712 Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
713
714
715 # Beta inflacionada
716 AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1)
717 # Kuma inflacionada
718 AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1)
719 # Gamma Unitaria inflacionada
720 AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1)
721
722 # Selecionando o menor AICC
723 min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
724
725 Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
```

```
Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)

#criterio_BIC#
# Beta inflacionada
BIC_IB=-2*log.vero_IB+k*log(n)
#Kuma inflacionada
BIC_IK=-2*log.vero_IK+k*log(n)
# Gamma Unitaria inflacionada
BIC_IUG=-2*fitigu$loglik+k*log(n)

# Selecionando o menor BIC
min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))

Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)

#criterio_BICC#
# Beta inflacionada
BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1)
#Kuma inflacionada
BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1)
# Gamma Unitaria inflacionada
BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1)

# Selecionando o menor BICC
min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))

Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)

#criterio_HQ#
# Beta inflacionada
HQ_IB=-2*log.vero_IB+2*log(log(n))
#Kuma inflacionada
HQ_IK=-2*log.vero_IK+2*log(log(n))
# Gamma Unitaria inflacionada
HQ_IUG=-2*fitigu$loglik+2*log(log(n))

# Selecionando o menor HQ
min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))

Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)
```

```
773
774   #criterio_HQC#
775   # Beta inflacionada
776   HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1)
777   #Kuma inflacionada
778   HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1)
779   # Gamma Unitaria inflacionada
780   HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1)
781
782   # Selecionando o menor HQC
783   min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))
784
785
786   Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
787   Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
788   Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
789   # FIM DO LACO DE MONTE CARLO
790
791 }
792
793 # Proporcao media de zeros
794 #mean(prop.z)
795 # Proporcao media de uns
796 #mean(prop.u)
797 #media das medias
798 mu.of=mean(media.1)
799 #media do desvio padrao
800 sd.of=mean(sd.1)
801
802
803
804 #Calculando percentuais escolha distribuicao(AIC)
805 p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
806 p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
807 p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
808 p_AICIB
809 p_AICIK
810 p_AICIUG
811
812 #Calculando percentuais escolha distribuicao(AICC)
813 p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
814 p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
815 p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
816 p_AICCIB
817 p_AICCIK
818 p_AICCIUG
819
```

```
#Calculando percentuais escolha distribuicao(BIC)
p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
p_BICIB
p_BICIK
p_BICIUG

#Calculando percentuais escolha distribuicao(BICC)
p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
p_BICCIB
p_BICCIK
p_BICCIUG

#Calculando percentuais escolha distribuicao(HQ)
p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
p_HQIB
p_HQIK
p_HQIUG

#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG


#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
p_LLIB
p_LLIK
p_LLIUG

#resultados -log.vero
result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
#Recebendo Resultados(AIC)
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
#Recebendo Resultados(AICC)
```

```
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
#Recebendo Resultados(BIC)
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
#Recebendo Resultados(BICC)
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
#Recebendo Resultados(HQ)
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
#Recebendo Resultados(HQC)
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC

cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100

ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100

ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100

cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100

ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100

ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100

cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100

ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100

ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)



M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
result.taxa.rej.ad,result.taxa.rej.cvm)
```

```
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.ks",
"Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")

glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob.Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
"\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "phi: {phi}","\n", "media: {media.v}",
"\n", "var: {round(var.v,4)}")
print("Resultados")
print(M)

M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm)
colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")
xtable(M.res)

cv=sd.of/mu.of
cv
var.teste=sd.of*sd.of
#Salvando em um arquivo
#write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

```
  ##Simulacao gerando amostras a partir da beta inflacionada em um##
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
library(goftest)

source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")


set.seed(33) #Fixando a semente


source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
```

```r
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }

  i01 <- (y==0)+(y==1)

  loglik <- function(theta)
  {
    omega <- theta[1]
    phi <- theta[2]

    ########
    l2 = 0
    if(u!=0)
    {
      l2a = ifelse((y == 1), log(p), 0)
      l2b = ifelse((y == 0), log(1-p), 0)
      l2 = l2a + l2b
    }

    l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

    l3 = ifelse((i01 == 1), 0,
    log(phi)+
```

```
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}

escore <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  delta = log(0.5)/log(1-omega^phi)
  ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
      +1)

  c = ifelse((y == 0), 0, ci)
  c = ifelse((y == 1), 0, c)

  Uomega <- phi*sum(c)

  v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
  (delta-1)*((y^phi)*log(y)/(1-y^phi)))
  v = ifelse((y == 1), 0, v)

  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

#############
```

```
115
116 opt <- optim(ini, loglik, escore, # hessian = T,
117 method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))
118
119 # opt <- optim(ini, loglik, escore,
120 # method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e
    -9),
121 # # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double
    .eps),
122 # # upper = rep(.Machine$double.xmax,(r+2))
123 # lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
124 # upper = rep(.Machine$double.xmax,(r+2))
125 # )
126
127 # opt <- optim(ini, loglik, escore)
128
129 # print(opt$par)
130
131 # print(names(opt))
132 # print(summary(opt) )
133 # print(summary(opt)[1:6] )
134 # print(opt$kkt1)
135
136
137 if (opt$conv != 0)
138 warning("FUNCTION DID NOT CONVERGE!")
139
140 k <- c()
141
142 # print("AQUI1")
143 # print(gradient(escore, opt$par))
144
145 coef <- c(lambda,p,opt$par)
146 # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
147 # print(coef)
148 k$coef <- coef
149 k$conv <- opt$conv
150 k$loglik <- opt$value
151 k$counts <- as.numeric(opt$counts[1])
152
153 # library(rootSolve)
154 # library(numDeriv)
155 # escores<-rbind(
156 # escore(coef),
157 # gradient(loglik, coef),
158 # grad(loglik, coef))
159 # print(round(escores,5))
```

```
160
161 # k$Knum<-gradient(escore, coef)
162
163 # print(k$Knum)
164 # k$Knum<-hessian(escore, coef, method="complex")
165 # print(k$Knum)
166
167
168 # # Expected information matrix
169 #
170 # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
171 # T3 = diag(as.vector( 1/diflink3(k$omega) ))
172 # T4 = diag(as.vector( 1/diflink4(k$phi) ))
173 #
174 # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
175 # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
176 # delta <- log(0.5)/log(1-k$omega^k$phi)
177 #
178 # kapa<- 0.5772156649
179 # k0<- (pi^2)/6+kapa^2-2*kapa
180 #
181 # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
      log(1-y^k$phi)+1)
182 #
183 # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
184 # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
185 # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
186 # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
187 #
188 # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
189 #
190 # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
191 #
192 # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
193 # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
      phi)) -
194 # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
      k$phi^3))
195 # #
196 # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
197 #
198 # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
      +1)-kapa)/((delta-1)*k$phi)) )
199 #
200 # L = diag(-1/(k$lambda*(1-k$lambda)))
201 # V = diag((k$lambda-1)*k$phi^2*h2)
202 # M = diag(mi)
```

```
203   # S = diag(si)
204   #
205   #
206   # Igg = -t(z1) %*% L %*% (T1^2) %*% z1
207   # Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
208   # Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
209   # Isb = t(Ibs)
210   # Iss = -t(q1) %*% S %*% (T4^2) %*% q1
211   #
212   # if(u!=0)
213   # {
214     # k$pi <- as.vector(coef[(m+1):(m+u)])
215     # eta_hat2 <- as.vector(w1%*%k$pi)
216     # k$p <- as.vector(linkinv2(eta_hat2))
217     # T2 = diag(as.vector( 1/diflink2(k$p) ))
218     # P = diag(-k$lambda/(k$p*(1-k$p)))
219     # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
220     #
221     # I <- rbind(
222     # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
          ncol=s)),
223     # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
          ncol=s)),
224     # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
225     # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
226     # )
227     #
228     # }else{
229     # I <- rbind(
230     # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
231     # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
232     # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
233     # )
234     # }
235   #
236   # k$I <- I
237   #
238   #
239   # ############################
240   #
241   # # print(-k$Knum)
242   # # print(k$I)
243   #
244   # # vcov <- try(solve(-k$Knum))
245   # vcov <- try(solve(k$I))
246   # # vcov <- chol2inv(chol(k$I))
247   # #vcov <- K #somente para nao trancar simulacao
```

```
  #
  # k$cov_ok = 0
  # if (class(vcov) == "try-error")
  # {
    # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
        with inversion
    # k$cov_ok = 1
    #
    # if (class(vcov) == "try-error") #
    # {
      # vcov <- k$Knum^2
      # warning("Problem with the variance and covariance matrix. The p-values
          are not correct.")
      # k$cov_ok = 2
      # }
    # }


  # k$vcov <- vcov
  # stderror <- sqrt(diag(k$vcov))
  # k$stderror <- stderror

  return(k)
}

ibetafit<-function(x)
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

  xm0<-x[which(x>0)]
  x01<-xm0[which(xm0<1)]

  #MV
  loglik<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
    phi<-par[4]

    alpha0 = ifelse(n0==0,0,alpha0)
    alpha1 = ifelse(n1==0,0,alpha1)

```

```
  #print(c(alpha0,alpha1,gama,phi))

  c<- 1- alpha0*(1-gama)-alpha1*gama
  mu<- gama*(1-alpha1)/c


  l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
  l1 = ifelse( (x == 1), log(alpha1*gama), 0)
  l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
  l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
      - mu) * phi, log = TRUE)))


  #print("soma")
  #print(sum(l0 + l1 + l2+ l3))
  sum(l0 + l1 + l2+ l3)
}

escore<-function(par)
{
  alpha0<-par[1]
  alpha1<-par[2]
  gama<-par[3]
  phi<-par[4]

  c<- 1- alpha0*(1-gama)-alpha1*gama
  mu<- (gama*(1-alpha1))/c

  a= mu * phi
  b = a * (1 - mu)/mu

  ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
  mustar <- digamma(a) - digamma(b)



  Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
  #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
  Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
      mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
  Ualpha0 = sum(Ualpha01 + Ualpha03)

  Ualpha0 = ifelse(n0==0,0,Ualpha0)


  Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
  #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
```

```
338   Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
          gama*(1-gama)*(1-alpha0))/ (c^2) ) )
339   #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
          digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
340   Ualpha1 = sum(Ualpha11 - Ualpha13)
341
342   Ualpha1 = ifelse(n1==0,0,Ualpha1)
343
344   Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
345   Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
346   Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
          mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
347
348   Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
349
350   Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
351   #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
          digamma(b))
352   mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
353   )
354   Uphi = sum(Uphi0)
355
356
357   c(Ualpha0,Ualpha1,Ugama,Uphi)
358 }
359
360 # metodo dos momentos para iniciarlizacao
361 x_bar<-mean(x01)
362 sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
363
364 p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
365 q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
366
367 #print(c(x_bar,sigma2_hat,p,q))
368
369 delta0<- n0/n
370 delta1<- n1/n
371
372 a0<- delta0/(1-x_bar)
373 a1<- delta1/(x_bar)
374
375 #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
376 par_ini<-c(a0,a1,p/(p+q),p+q)
377
378 #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
379
380 #print("par_ini")
```

```
  #print(par_ini)

  max<-optim(par_ini,loglik,
  escore,
  method="BFGS",
  #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z<-c()
  z$mv<-max$par
  #print(z$mv)
  z$conv<-max$convergence
  z$loglik <- max$value
  return(z)
}


iuGAfit <- function(y)

{

  n <- length(y) # sample size
  n0 <- sum(y == 0) # number of zeros
  n1 <- sum(y == 1) # number of ones
  m <- n0 + n1

  ys01 <- y
  if(any(y==0)) ys01<- ys01[-which(y==0)]
  if(any(y==1)) ys01<- ys01[-which(ys01==1)]

  i01 <- (y==0)+(y==1)

  #Maxima verossimilhanca

  loglik <- function (par){

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

    alpha0 = ifelse(n0 == 0, 0, alpha0)
    alpha1 = ifelse(n1 == 0, 0, alpha1)

```

```
427     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
428     mu <- gama * (1 - alpha1) / c
429     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
430 
431     l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
432     l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
433     l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
434     l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))
435 
436     #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))
437 
438     sum(l0 + l1 + l2 + l3)
439 
440   }
441   escore <- function (par) {
442 
443     alpha0 <- par[1]
444     alpha1 <- par[2]
445     gama <- par[3]
446     phi <- par[4]
447 
448     alpha0 <- ifelse(n0 == 0, 0, alpha0)
449     alpha1 <- ifelse(n1 == 0, 0, alpha1)
450 
451     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
452     mu <- gama * (1 - alpha1) / c
453     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
454     ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
455     mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
456 
457 
458     Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
459     Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
        / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
        ^ 2))
460     Ualpha0 <- sum(Ualpha01 + Ualpha03)
461 
462     Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
463 
464     Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
465     Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
        (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
        2))
466     Ualpha1 <- sum(Ualpha11 + Ualpha13)
467 
468     Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
469 
```

```
470
471   Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
472   Ugama02 <- ifelse((y == 1), 1 / gama, 0)
473   Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
          + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
           ^ 2))
474
475   Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
476
477   Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
          /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
          phi) - log(mu^(1/phi)))
478
479   Uphi = sum(Uphi0)
480
481
482   c(Ualpha0, Ualpha1, Ugama, Uphi)
483 }
484
485 #Inicializacao
486
487 y_bar <- mean(y)
488 y_bar0 <- mean(ys01)
489
490 delta0 <- n0/n
491 delta1 <- n1/n
492
493 a0 <- delta0/(1-y_bar)
494 a1 <- delta1/(y_bar)
495
496
497 #fit <- vglm(ys01 ~ 1, gamma2)
498 #par<-Coef(fit)
499
500 #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
501 #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
502 #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
503 #phi_til = 5
504 phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
505
506 par_ini <- c(a0, a1, y_bar, phi_til)
507
508
509 max<-optim(par_ini, loglik, escore,method = "BFGS",
510 #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
511 #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
```

```
      (0.99999,0.99999,0.99999,9999999999),
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z <- c()
  z$ini <- par_ini
  z$mv <- max$par
  z$conv <- max$convergence
  z$loglik <- max$value
  return(z)
}

n=200
#n=60
#n=100
#n=200
NR=10000
k=3 # numero de parametros

Cont_AIC_IB=rep(0,NR)
Cont_AIC_IK=rep(0,NR)
Cont_AIC_IUG=rep(0,NR)

Cont_AICC_IB=rep(0,NR)
Cont_AICC_IK=rep(0,NR)
Cont_AICC_IUG=rep(0,NR)

Cont_BIC_IB=rep(0,NR)
Cont_BIC_IK=rep(0,NR)
Cont_BIC_IUG=rep(0,NR)

Cont_BICC_IB=rep(0,NR)
Cont_BICC_IK=rep(0,NR)
Cont_BICC_IUG=rep(0,NR)

Cont_HQ_IB=rep(0,NR)
Cont_HQ_IK=rep(0,NR)
Cont_HQ_IUG=rep(0,NR)

Cont_HQC_IB=rep(0,NR)
Cont_HQC_IK=rep(0,NR)
Cont_HQC_IUG=rep(0,NR)

Cont_LL_IB =rep(0,NR)
Cont_LL_IK=rep(0,NR)
Cont_LL_IUG=rep(0,NR)
```

```
558
559 contC=0
560 contC1=0
561
562 prop.z = rep(0,NR)
563 prop.u = rep(0,NR)
564
565 cvm.iBE = rep(0,NR)
566 ks.iBE = rep(0,NR)
567 ad.iBE = rep(0,NR)
568
569
570 cvm.iuGA = rep(0,NR)
571 ks.iuGA = rep(0,NR)
572 ad.iuGA = rep(0,NR)
573
574 cvm.ikum = rep(0,NR)
575 ks.ikum = rep(0,NR)
576 ad.ikum = rep(0,NR)
577
578 alpha0 = 0
579 alpha1 = 0.1
580 gama = 0.5
581 phi = 10
582 valores.parametros = c(alpha0, alpha1, gama, phi)
583
584 # Isaac,
585 prob.z = alpha0*(1-gama)
586 prob.o = alpha1*gama
587
588 prob.z # probabilidade de zero
589 prob.o #probabilidade de um
590
591 # Isaac, v
592 media.v =gama
593 var.v =gama*((1+alpha1*phi)/(1+phi)) + (((1-alpha1)*(1-alpha1)*phi)/((1-alpha0*
      (1-gama)-alpha1*gama)*(1+phi)) -1)*gama^(2)
594
595 var.v
596 #Bayes - BEINF
597
598 c= 1- alpha0*(1-gama)-alpha1*gama
599 mu = gama*(1-alpha1)/c
600 delta0 = alpha0 * (1-gama)
601 delta1 = alpha1 * (gama)
602 p2= 1- delta0-delta1
603
```

```
mu
delta0
delta1
p2
dp = sqrt(1/(phi+1))


cont.ib = 0
cont.ik = 0
cont.igu = 0

for(i in 1:NR) {


  y = rBEINF1(n, mu = mu , sigma=dp, nu=delta1/p2)
  len = length(y)
  prop.z[i] = sum(ifelse(y==0,1,0))/len
  prop.u[i] = sum(ifelse(y==1,1,0))/len


  fitib = ibetafit(y)
  if (fitib$conv != 0)
  {
    cont.ib = cont.ib+1
  }

  fitik = ikumafit(y)
  if (fitik$conv != 0)
  {
    cont.ik = cont.ik+1
  }


  fitigu = iuGAfit(y)


  if ((fitigu$conv != 0) ){
    cont.igu=cont.igu+1
  }


  ks.iBE[i] = ifelse(ks.test(y, "pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$p.value
      >0.05,1,0)

  ad.iBE[i] = ifelse(ad.test(y, null="pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$p.
      value>0.05,1,0)

```

```
649 cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$
        p.value >0.05,1,0)
650
651
652 ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
        phi=phi)$p.value>0.05,1,0)
653
654 ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
655
656 cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
657
658
659 ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=alpha1, p=1,omega=gama, phi=phi)$
        p.value>0.05,1,0)
660
661 ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=alpha1, p=1,omega=gama, phi=
        phi)$p.value>0.05,1,0)
662
663 cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=alpha1, p=1,omega=gama,
        phi=phi)$p.value>0.05,1,0)
664
665
666 #Log verossimilhanca
667 # Beta inflacionada
668 log.vero_IB = fitib$loglik
669 # Kuma inflacionada
670 log.vero_IK= fitik$loglik
671 # Gamma Unitaria inflacionada
672 log.vero_IUG = fitigu$loglik
673
674 m.log.vero_IB = -fitib$loglik
675 # Kuma inflacionada
676 m.log.vero_IK= -fitik$loglik
677 # Gamma Unitaria inflacionada
678 m.log.vero_IUG = -fitigu$loglik
679
680 #Menor -log verossimilhanca
681
682 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
683
684 Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
685 Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
686 Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
687
688
```

```
689  # Calculando os criterios/quantidades para comparacao
690
691  #criteiro_AIC#
692  # Beta inflacionada
693  AIC_IB = -2*log.vero_IB+2*k
694  # Kuma inflacionada
695  AIC_IK= -2*log.vero_IK+2*k
696  # Gamma Unitaria inflacionada
697  AIC_IUG = -2*log.vero_IUG+2*k
698
699  # Selecionando o menor AIC
700  min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
701
702  Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
703  Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
704  Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
705
706  #criterio_AICC#
707  # Beta inflacionada
708  AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1)
709  # Kuma inflacionada
710  AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1)
711  # Gamma Unitaria inflacionada
712  AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1)
713
714  # Selecionando o menor AICC
715  min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
716
717  Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
718  Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
719  Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)
720
721  #criterio_BIC#
722  # Beta inflacionada
723  BIC_IB=-2*log.vero_IB+k*log(n)
724  #Kuma inflacionada
725  BIC_IK=-2*log.vero_IK+k*log(n)
726  # Gamma Unitaria inflacionada
727  BIC_IUG=-2*fitigu$loglik+k*log(n)
728
729  # Selecionando o menor BIC
730  min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))
731
732  Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
733  Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
734  Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)
735
```

```
#criterio_BICC#
# Beta inflacionada
BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1)
#Kuma inflacionada
BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1)
# Gamma Unitaria inflacionada
BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1)

# Selecionando o menor BICC
min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))

Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)

#criterio_HQ#
# Beta inflacionada
HQ_IB=-2*log.vero_IB+2*log(log(n))
#Kuma inflacionada
HQ_IK=-2*log.vero_IK+2*log(log(n))
# Gamma Unitaria inflacionada
HQ_IUG=-2*fitigu$loglik+2*log(log(n))

# Selecionando o menor HQ
min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))

Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)

#criterio_HQC#
# Beta inflacionada
HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1)
#Kuma inflacionada
HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1)
# Gamma Unitaria inflacionada
HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1)

# Selecionando o menor HQC
min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))


Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
# FIM DO LACO DE MONTE CARLO

```

```
}

# Proporcao media de zeros
#mean(prop.z)
# Proporcao media de uns
#mean(prop.u)


#Calculando percentuais escolha distribuicao(AIC)
p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
p_AICIB
p_AICIK
p_AICIUG

#Calculando percentuais escolha distribuicao(AICC)
p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
p_AICCIB
p_AICCIK
p_AICCIUG

#Calculando percentuais escolha distribuicao(BIC)
p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
p_BICIB
p_BICIK
p_BICIUG

#Calculando percentuais escolha distribuicao(BICC)
p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
p_BICCIB
p_BICCIK
p_BICCIUG

#Calculando percentuais escolha distribuicao(HQ)
p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
p_HQIB
p_HQIK
p_HQIUG
```

```
830
831 #Calculando percentuais escolha distribuicao(HQC)
832 p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
833 p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
834 p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
835 p_HQCIB
836 p_HQCIK
837 p_HQCIUG
838
839
840 #Calculando percentuais escolha distribuicao(-LL)
841 p_LLIB<-((sum(Cont_LL_IB))/NR)*100
842 p_LLIK<-((sum(Cont_LL_IK))/NR)*100
843 p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
844 p_LLIB
845 p_LLIK
846 p_LLIUG
847
848 #resultados -log.vero
849 result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
850 #Recebendo Resultados(AIC)
851 result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
852 result_AIC
853 #Recebendo Resultados(AICC)
854 result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
855 result_AICC
856 #Recebendo Resultados(BIC)
857 result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
858 result_BIC
859 #Recebendo Resultados(BICC)
860 result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
861 result_BICC
862 #Recebendo Resultados(HQ)
863 result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
864 result_HQ
865 #Recebendo Resultados(HQC)
866 result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
867 result_HQC
868
869 cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100
870
871 ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100
872
873 ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100
874
875 cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100
876
```

```r
ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100

ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100

cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100

ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100

ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)


M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
result.taxa.rej.ad,result.taxa.rej.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
    ks",
"Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")

glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
    .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
"\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
    phi: {phi}","\n", "media: {media.v}",
"\n", "var: {round(var.v,4)}")
print("Resultados")
print(M)


M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
    )
colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
    Rej.cvm")
xtable(M.res)

#Salvando em um arquivo
#write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

```r
  ##Simulacao gerando amostras a partir da kumaraswamy inflacionada em zero##
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
library(goftest)

source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")


set.seed(33) #Fixando a semente


source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }
```

```
48
49  i01 <- (y==0)+(y==1)
50
51  loglik <- function(theta)
52  {
53    omega <- theta[1]
54    phi <- theta[2]
55
56    ########
57    l2 = 0
58    if(u!=0)
59    {
60      l2a = ifelse((y == 1), log(p), 0)
61      l2b = ifelse((y == 0), log(1-p), 0)
62      l2 = l2a + l2b
63    }
64
65    l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))
66
67    l3 = ifelse((i01 == 1), 0,
68    log(phi)+
69    log(log(0.5)/log(1-omega^phi))+
70    (phi-1)*log(y)+
71    ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
72    )
73
74    sum(l1 + l2 + l3)
75  }
76
77  escore <- function(theta)
78  {
79    omega <- theta[1]
80    phi <- theta[2]
81
82    delta = log(0.5)/log(1-omega^phi)
83    ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
          +1)
84
85    c = ifelse((y == 0), 0, ci)
86    c = ifelse((y == 1), 0, c)
87
88    Uomega <- phi*sum(c)
89
90    v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
91    (delta-1)*((y^phi)*log(y)/(1-y^phi)))
92    v = ifelse((y == 1), 0, v)
93
```

```
  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

##############

opt <- optim(ini, loglik, escore, # hessian = T,
method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))

# opt <- optim(ini, loglik, escore,
# method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e-9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double.eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
# upper = rep(.Machine$double.xmax,(r+2))
# )

# opt <- optim(ini, loglik, escore)

# print(opt$par)

# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)


if (opt$conv != 0)
warning("FUNCTION DID NOT CONVERGE!")
```

```
139
140 k <- c()
141
142 # print("AQUI1")
143 # print(gradient(escore, opt$par))
144
145 coef <- c(lambda,p,opt$par)
146 # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
147 # print(coef)
148 k$coef <- coef
149 k$conv <- opt$conv
150 k$loglik <- opt$value
151 k$counts <- as.numeric(opt$counts[1])
152
153 # library(rootSolve)
154 # library(numDeriv)
155 # escores<-rbind(
156 # escore(coef),
157 # gradient(loglik, coef),
158 # grad(loglik, coef))
159 # print(round(escores,5))
160
161 # k$Knum<-gradient(escore, coef)
162
163 # print(k$Knum)
164 # k$Knum<-hessian(escore, coef, method="complex")
165 # print(k$Knum)
166
167
168 # # Expected information matrix
169 #
170 # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
171 # T3 = diag(as.vector( 1/diflink3(k$omega) ))
172 # T4 = diag(as.vector( 1/diflink4(k$phi) ))
173 #
174 # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
175 # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
176 # delta <- log(0.5)/log(1-k$omega^k$phi)
177 #
178 # kapa<- 0.5772156649
179 # k0<- (pi^2)/6+kapa^2-2*kapa
180 #
181 # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
      log(1-y^k$phi)+1)
182 #
183 # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
184 # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
```

```
185   # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
186   # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
187   #
188   # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
189   #
190   # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
191   #
192   # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
193   # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
        phi)) -
194   # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
        k$phi^3))
195   # #
196   # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
197   #
198   # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
        +1)-kapa)/((delta-1)*k$phi)) )
199   #
200   # L = diag(-1/(k$lambda*(1-k$lambda)))
201   # V = diag((k$lambda-1)*k$phi^2*h2)
202   # M = diag(mi)
203   # S = diag(si)
204   #
205   #
206   # Igg = -t(z1) %*% L %*% (T1^2) %*% z1
207   # Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
208   # Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
209   # Isb = t(Ibs)
210   # Iss = -t(q1) %*% S %*% (T4^2) %*% q1
211   #
212   # if(u!=0)
213   # {
214     # k$pi <- as.vector(coef[(m+1):(m+u)])
215     # eta_hat2 <- as.vector(w1%*%k$pi)
216     # k$p <- as.vector(linkinv2(eta_hat2))
217     # T2 = diag(as.vector( 1/diflink2(k$p) ))
218     # P = diag(-k$lambda/(k$p*(1-k$p)))
219     # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
220     #
221     # I <- rbind(
222     # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
          ncol=s)),
223     # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
          ncol=s)),
224     # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
225     # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
226     # )
```

```
227     #
228     # }else{
229     # I <- rbind(
230     # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
231     # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
232     # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
233     # )
234     # }
235   #
236   # k$I <- I
237   #
238   #
239   # ############################
240   #
241   # # print(-k$Knum)
242   # # print(k$I)
243   #
244   # # vcov <- try(solve(-k$Knum))
245   # vcov <- try(solve(k$I))
246   # # vcov <- chol2inv(chol(k$I))
247   # #vcov <- K #somente para nao trancar simulacao
248   #
249   # k$cov_ok = 0
250   # if (class(vcov) == "try-error")
251   # {
252     # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
          with inversion
253     # k$cov_ok = 1
254     #
255     # if (class(vcov) == "try-error") #
256     # {
257       # vcov <- k$Knum^2
258       # warning("Problem with the variance and covariance matrix. The p-values
            are not correct.")
259       # k$cov_ok = 2
260       # }
261     # }
262
263
264   # k$vcov <- vcov
265   # stderror <- sqrt(diag(k$vcov))
266   # k$stderror <- stderror
267
268   return(k)
269 }
270
271 ibetafit<-function(x)
```

```
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

  xm0<-x[which(x>0)]
  x01<-xm0[which(xm0<1)]

  #MV
  loglik<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
    phi<-par[4]

    alpha0 = ifelse(n0==0,0,alpha0)
    alpha1 = ifelse(n1==0,0,alpha1)

    #print(c(alpha0,alpha1,gama,phi))

    c<- 1- alpha0*(1-gama)-alpha1*gama
    mu<- gama*(1-alpha1)/c


    l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
    l1 = ifelse( (x == 1), log(alpha1*gama), 0)
    l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
    l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
        - mu) * phi, log = TRUE)))


    #print("soma")
    #print(sum(l0 + l1 + l2+ l3))
    sum(l0 + l1 + l2+ l3)
  }

  escore<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
    phi<-par[4]

    c<- 1- alpha0*(1-gama)-alpha1*gama
```

```
318     mu<- (gama*(1-alpha1))/c
319
320     a= mu * phi
321     b = a * (1 - mu)/mu
322
323     ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
324     mustar <- digamma(a) - digamma(b)
325
326
327
328     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
329     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
330     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
            mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
331     Ualpha0 = sum(Ualpha01 + Ualpha03)
332
333     Ualpha0 = ifelse(n0==0,0,Ualpha0)
334
335
336     Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
337     #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
338     Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
            gama*(1-gama)*(1-alpha0))/ (c^2) ) )
339     #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
            digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
340     Ualpha1 = sum(Ualpha11 - Ualpha13)
341
342     Ualpha1 = ifelse(n1==0,0,Ualpha1)
343
344     Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
345     Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
346     Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
            mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
347
348     Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
349
350     Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
351     #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
            digamma(b))
352     mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
353     )
354     Uphi = sum(Uphi0)
355
356
357     c(Ualpha0,Ualpha1,Ugama,Uphi)
358   }
359
```

```
360   # metodo dos momentos para iniciarlizacao
361   x_bar<-mean(x01)
362   sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
363
364   p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
365   q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
366
367   #print(c(x_bar,sigma2_hat,p,q))
368
369   delta0<- n0/n
370   delta1<- n1/n
371
372   a0<- delta0/(1-x_bar)
373   a1<- delta1/(x_bar)
374
375   #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
376   par_ini<-c(a0,a1,p/(p+q),p+q)
377
378   #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
379
380   #print("par_ini")
381   #print(par_ini)
382
383   max<-optim(par_ini,loglik,
384   escore,
385   method="BFGS",
386   #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
387   control=list(fnscale=-1))
388
389   if(max$convergence != 0) print("Non-convergence")
390   # return
391   z<-c()
392   z$mv<-max$par
393   #print(z$mv)
394   z$conv<-max$convergence
395   z$loglik <- max$value
396   return(z)
397 }
398
399
400 iuGAfit <- function(y)
401
402 {
403
404   n <- length(y) # sample size
405   n0 <- sum(y == 0) # number of zeros
```

```
406 n1 <- sum(y == 1) # number of ones
407 m <- n0 + n1
408
409 ys01 <- y
410 if(any(y==0)) ys01<- ys01[-which(y==0)]
411 if(any(y==1)) ys01<- ys01[-which(ys01==1)]
412
413 i01 <- (y==0)+(y==1)
414
415 #Maxima verossimilhanca
416
417 loglik <- function (par){
418
419   alpha0 <- par[1]
420   alpha1 <- par[2]
421   gama <- par[3]
422   phi <- par[4]
423
424   alpha0 = ifelse(n0 == 0, 0, alpha0)
425   alpha1 = ifelse(n1 == 0, 0, alpha1)
426
427   c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
428   mu <- gama * (1 - alpha1) / c
429   d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
430
431   l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
432   l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
433   l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
434   l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))
435
436   #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))
437
438   sum(l0 + l1 + l2 + l3)
439
440 }
441 escore <- function (par) {
442
443   alpha0 <- par[1]
444   alpha1 <- par[2]
445   gama <- par[3]
446   phi <- par[4]
447
448   alpha0 <- ifelse(n0 == 0, 0, alpha0)
449   alpha1 <- ifelse(n1 == 0, 0, alpha1)
450
451   c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
452   mu <- gama * (1 - alpha1) / c
```

```
453     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
454     ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
455     mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
456
457
458     Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
459     Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
            / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
            ^ 2))
460     Ualpha0 <- sum(Ualpha01 + Ualpha03)
461
462     Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
463
464     Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
465     Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
            (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
            2))
466     Ualpha1 <- sum(Ualpha11 + Ualpha13)
467
468     Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
469
470
471     Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
472     Ugama02 <- ifelse((y == 1), 1 / gama, 0)
473     Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
            + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
             ^ 2))
474
475     Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
476
477     Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
            /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
            phi) - log(mu^(1/phi)))
478
479     Uphi = sum(Uphi0)
480
481
482     c(Ualpha0, Ualpha1, Ugama, Uphi)
483   }
484
485   #Inicializacao
486   y_bar <- mean(y)
487   y_bar0 <- mean(ys01)
488
489   delta0 <- n0/n
490   delta1 <- n1/n
491
```

```
  a0 <- delta0/(1-y_bar)
  a1 <- delta1/(y_bar)


  #fit <- vglm(ys01 ~ 1, gamma2)
  #par<-Coef(fit)

  #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
  #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
  #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
  #phi_til = 5
  phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))

  par_ini <- c(a0, a1, y_bar, phi_til)


  max<-optim(par_ini, loglik, escore,method = "BFGS",
  #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z <- c()
  z$ini <- par_ini
  z$mv <- max$par
  z$conv <- max$convergence
  z$loglik <- max$value
  return(z)
}

n=200
#n=60
#n=100
#n=200
NR=10000
k=3 # numero de parametros


Cont_AIC_IB=rep(0,NR)
Cont_AIC_IK=rep(0,NR)
Cont_AIC_IUG=rep(0,NR)


Cont_AICC_IB=rep(0,NR)
```

```
Cont_AICC_IK=rep(0,NR)
Cont_AICC_IUG=rep(0,NR)

Cont_BIC_IB=rep(0,NR)
Cont_BIC_IK=rep(0,NR)
Cont_BIC_IUG=rep(0,NR)

Cont_BICC_IB=rep(0,NR)
Cont_BICC_IK=rep(0,NR)
Cont_BICC_IUG=rep(0,NR)

Cont_HQ_IB=rep(0,NR)
Cont_HQ_IK=rep(0,NR)
Cont_HQ_IUG=rep(0,NR)

Cont_HQC_IB=rep(0,NR)
Cont_HQC_IK=rep(0,NR)
Cont_HQC_IUG=rep(0,NR)

Cont_LL_IB =rep(0,NR)
Cont_LL_IK=rep(0,NR)
Cont_LL_IUG=rep(0,NR)

contC=0
contC1=0

prop.z = rep(0,NR)
prop.u = rep(0,NR)
var.am=rep(0,NR) #isaac

cvm.iBE = rep(0,NR)
ks.iBE = rep(0,NR)
ad.iBE = rep(0,NR)

cvm.iuGA = rep(0,NR)
ks.iuGA = rep(0,NR)
ad.iuGA = rep(0,NR)

cvm.ikum = rep(0,NR)
ks.ikum = rep(0,NR)
ad.ikum = rep(0,NR)

alpha0 = 0.05
alpha1 = 0
gama = 0.3
phi = 30

```

```
prob.0=alpha0*(1-gama) #(prob de zero)
prob.1=alpha1*gama #(prob de um)
prob.0
prob.1


lambda = prob.0+prob.1 #(prop.infl ou seja pro.0+prop.1)
lambda
p=(prob.1)/(lambda)
p

#lambda = alpha0
#(p=1 infl1 p=0 infla 0)
omega = gama
phi = phi

omega

valores.parametros = c(alpha0, alpha1, gama, phi)

prob.z = lambda*(1-p)
prob.o = lambda*p

prob.z # probabilidade de zero
prob.o #probabilidade de um

media.v = gama

bet=(log(0.5))/(log(1-omega^phi))
bet #expressao do paramentro beta

#var.v =lambda*p*(1-lambda*p)*bet*((beta(1+2/phi,bet))-(beta(1+1/phi,bet))*(2*
    lambda*p+bet*(1-lambda)*(beta(1+1/phi,bet))))


var.v =lambda*p*(1-lambda*p)+(1-lambda*p)*bet*((beta(1+2/phi,bet))-(beta(1+1/phi,
    bet))*(2*lambda*p+bet*(1-lambda)*(beta(1+1/phi,bet))))
#media.v
var.v

#Bayes - BEINF

c= 1- alpha0*(1-gama)-alpha1*gama
mu = gama*(1-alpha1)/c
delta0 = alpha0 * (1-gama)
delta1 = alpha1 * (gama)
p2= 1- delta0-delta1
```

```
629
630 mu
631 delta0
632 delta1
633 p2
634 dp = sqrt(1/(phi+1))
635
636
637 cont.ib = 0
638 cont.ik = 0
639 cont.igu = 0
640
641 for(i in 1:NR) {
642
643
644   y = rikum(n,lambda = lambda, p=p, omega = omega, phi = phi)
645   len = length(y)
646   prop.z[i] = sum(ifelse(y==0,1,0))/len
647   prop.u[i] = sum(ifelse(y==1,1,0))/len
648   var.am[i]=var(y) #isaac
649
650   fitib = ibetafit(y)
651   if (fitib$conv != 0)
652   {
653     cont.ib = cont.ib+1
654   }
655
656
657   fitik = ikumafit(y)
658   if (fitik$conv != 0)
659   {
660     cont.ik = cont.ik+1
661   }
662
663
664   fitigu = iuGAfit(y)
665
666
667   if ((fitigu$conv != 0) ){
668     cont.igu=cont.igu+1
669   }
670
671
672   ks.iBE[i] = ifelse(ks.test(y, "pBEINFO",mu=mu , sigma=dp, nu=delta0/p2)$p.value
        >0.05,1,0)
673
674   ad.iBE[i] = ifelse(ad.test(y, null="pBEINFO",mu=mu , sigma=dp, nu=delta0/p2)$p.
```

```
    value>0.05,1,0)
675
676 cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF0",mu=mu , sigma=dp, nu=delta0/p2)$
    p.value >0.05,1,0)
677
678
679 ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
    phi=phi)$p.value>0.05,1,0)
680
681 ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
    gama, phi=phi)$p.value>0.05,1,0)
682
683 cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
    gama, phi=phi)$p.value>0.05,1,0)
684
685
686 ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=lambda, p=0,omega=gama, phi=phi)$
    p.value>0.05,1,0)
687
688 ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=lambda, p=0,omega=gama, phi=
    phi)$p.value>0.05,1,0)
689
690 cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=lambda, p=0,omega=gama,
    phi=phi)$p.value>0.05,1,0)
691
692
693 #Log verossimilhanca
694 # Beta inflacionada
695 log.vero_IB = fitib$loglik
696 # Kuma inflacionada
697 log.vero_IK= fitik$loglik
698 # Gamma Unitaria inflacionada
699 log.vero_IUG = fitigu$loglik
700
701 m.log.vero_IB = -fitib$loglik
702 # Kuma inflacionada
703 m.log.vero_IK= -fitik$loglik
704 # Gamma Unitaria inflacionada
705 m.log.vero_IUG = -fitigu$loglik
706
707 #Menor -log verossimilhanca
708
709 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
710
711 Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
712 Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
713 Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
```

```
714
715  # Calculando os criterios/quantidades para comparacao
716
717  #criteiro_AIC#
718  # Beta inflacionada
719  AIC_IB = -2*log.vero_IB+2*k
720  # Kuma inflacionada
721  AIC_IK= -2*log.vero_IK+2*k
722  # Gamma Unitaria inflacionada
723  AIC_IUG = -2*log.vero_IUG+2*k
724
725  # Selecionando o menor AIC
726  min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
727
728  Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
729  Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
730  Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
731
732
733  # Beta inflacionada
734  AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1)
735  # Kuma inflacionada
736  AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1)
737  # Gamma Unitaria inflacionada
738  AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1)
739
740  # Selecionando o menor AICC
741  min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
742
743  Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
744  Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
745  Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)
746
747  #criterio_BIC#
748  # Beta inflacionada
749  BIC_IB=-2*log.vero_IB+k*log(n)
750  #Kuma inflacionada
751  BIC_IK=-2*log.vero_IK+k*log(n)
752  # Gamma Unitaria inflacionada
753  BIC_IUG=-2*fitigu$loglik+k*log(n)
754
755  # Selecionando o menor BIC
756  min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))
757
758  Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
759  Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
760  Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)
```

```
761
762   #criterio_BICC#
763   # Beta inflacionada
764   BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1)
765   #Kuma inflacionada
766   BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1)
767   # Gamma Unitaria inflacionada
768   BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1)
769
770   # Selecionando o menor BICC
771   min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))
772
773   Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
774   Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
775   Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)
776
777   #criterio_HQ#
778   # Beta inflacionada
779   HQ_IB=-2*log.vero_IB+2*log(log(n))
780   #Kuma inflacionada
781   HQ_IK=-2*log.vero_IK+2*log(log(n))
782   # Gamma Unitaria inflacionada
783   HQ_IUG=-2*fitigu$loglik+2*log(log(n))
784
785   # Selecionando o menor HQ
786   min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))
787
788   Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
789   Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
790   Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)
791
792   #criterio_HQC#
793   # Beta inflacionada
794   HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1)
795   #Kuma inflacionada
796   HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1)
797   # Gamma Unitaria inflacionada
798   HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1)
799
800   # Selecionando o menor HQC
801   min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))
802
803   Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
804   Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
805   Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
806   # FIM DO LACO DE MONTE CARLO
807 }
```

```
808
809 # Proporcao media de zeros
810 #mean(prop.z)
811 # Proporcao media de uns
812 #mean(prop.u)
813 var.amostral=mean(var.am) #isaac
814
815 #Calculando percentuais escolha distribuicao(AIC)
816 p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
817 p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
818 p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
819 p_AICIB
820 p_AICIK
821 p_AICIUG
822
823 #Calculando percentuais escolha distribuicao(AICC)
824 p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
825 p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
826 p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
827 p_AICCIB
828 p_AICCIK
829 p_AICCIUG
830
831 #Calculando percentuais escolha distribuicao(BIC)
832 p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
833 p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
834 p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
835 p_BICIB
836 p_BICIK
837 p_BICIUG
838
839 #Calculando percentuais escolha distribuicao(BICC)
840 p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
841 p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
842 p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
843 p_BICCIB
844 p_BICCIK
845 p_BICCIUG
846
847 #Calculando percentuais escolha distribuicao(HQ)
848 p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
849 p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
850 p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
851 p_HQIB
852 p_HQIK
853 p_HQIUG
854
```

```
#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG

#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
p_LLIB
p_LLIK
p_LLIUG

#resultados -log.vero
result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
#Recebendo Resultados(AIC)
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
#Recebendo Resultados(AICC)
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
#Recebendo Resultados(BIC)
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
#Recebendo Resultados(BICC)
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
#Recebendo Resultados(HQ)
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
#Recebendo Resultados(HQC)
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC

cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100

ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100

ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100

cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100

ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100

```

```r
ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100

cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100

ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100

ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)


M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
result.taxa.rej.ad,result.taxa.rej.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
    ks",
"Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")

glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
    .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
"\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
    phi: {phi}","\n", "media: {media.v}",
"\n", "var: {round(var.v,4)}")
print("Resultados")
print(M)

var.amostral

M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
    )
colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
    Rej.cvm")
xtable(M.res)

#Salvando em um arquivo
#write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

```r
  ##Simulacao gerando amostras a partir da kumaraswamy inflacionada em zero e um
```

```
     ##
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
library(goftest)

source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")


set.seed(33) #Fixando a semente


source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }
```

```
48
49  i01 <- (y==0)+(y==1)
50
51  loglik <- function(theta)
52  {
53    omega <- theta[1]
54    phi <- theta[2]
55
56    ########
57    l2 = 0
58    if(u!=0)
59    {
60      l2a = ifelse((y == 1), log(p), 0)
61      l2b = ifelse((y == 0), log(1-p), 0)
62      l2 = l2a + l2b
63    }
64
65    l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))
66
67    l3 = ifelse((i01 == 1), 0,
68    log(phi)+
69    log(log(0.5)/log(1-omega^phi))+
70    (phi-1)*log(y)+
71    ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
72    )
73
74    sum(l1 + l2 + l3)
75  }
76
77  escore <- function(theta)
78  {
79    omega <- theta[1]
80    phi <- theta[2]
81
82    delta = log(0.5)/log(1-omega^phi)
83    ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
          +1)
84
85    c = ifelse((y == 0), 0, ci)
86    c = ifelse((y == 1), 0, c)
87
88    Uomega <- phi*sum(c)
89
90    v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
91    (delta-1)*((y^phi)*log(y)/(1-y^phi)))
92    v = ifelse((y == 1), 0, v)
93
```

```
  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

##############

opt <- optim(ini, loglik, escore, # hessian = T,
method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))

# opt <- optim(ini, loglik, escore,
# method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e-9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double.eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
# upper = rep(.Machine$double.xmax,(r+2))
# )

# opt <- optim(ini, loglik, escore)

# print(opt$par)

# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)


if (opt$conv != 0)
warning("FUNCTION DID NOT CONVERGE!")
```

```
139
140 k <- c()
141
142 # print("AQUI1")
143 # print(gradient(escore, opt$par))
144
145 coef <- c(lambda,p,opt$par)
146 # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
147 # print(coef)
148 k$coef <- coef
149 k$conv <- opt$conv
150 k$loglik <- opt$value
151 k$counts <- as.numeric(opt$counts[1])
152
153 # library(rootSolve)
154 # library(numDeriv)
155 # escores<-rbind(
156 # escore(coef),
157 # gradient(loglik, coef),
158 # grad(loglik, coef))
159 # print(round(escores,5))
160
161 # k$Knum<-gradient(escore, coef)
162
163 # print(k$Knum)
164 # k$Knum<-hessian(escore, coef, method="complex")
165 # print(k$Knum)
166
167
168 # # Expected information matrix
169 #
170 # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
171 # T3 = diag(as.vector( 1/diflink3(k$omega) ))
172 # T4 = diag(as.vector( 1/diflink4(k$phi) ))
173 #
174 # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
175 # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
176 # delta <- log(0.5)/log(1-k$omega^k$phi)
177 #
178 # kapa<- 0.5772156649
179 # k0<- (pi^2)/6+kapa^2-2*kapa
180 #
181 # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
      log(1-y^k$phi)+1)
182 #
183 # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
184 # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
```

```
185   # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
186   # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
187   #
188   # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
189   #
190   # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
191   #
192   # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
193   # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
        phi)) -
194   # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
        k$phi^3))
195   # #
196   # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
197   #
198   # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
        +1)-kapa)/((delta-1)*k$phi)) )
199   #
200   # L = diag(-1/(k$lambda*(1-k$lambda)))
201   # V = diag((k$lambda-1)*k$phi^2*h2)
202   # M = diag(mi)
203   # S = diag(si)
204   #
205   #
206   # Igg = -t(z1) %*% L %*% (T1^2) %*% z1
207   # Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
208   # Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
209   # Isb = t(Ibs)
210   # Iss = -t(q1) %*% S %*% (T4^2) %*% q1
211   #
212   # if(u!=0)
213   # {
214     # k$pi <- as.vector(coef[(m+1):(m+u)])
215     # eta_hat2 <- as.vector(w1%*%k$pi)
216     # k$p <- as.vector(linkinv2(eta_hat2))
217     # T2 = diag(as.vector( 1/diflink2(k$p) ))
218     # P = diag(-k$lambda/(k$p*(1-k$p)))
219     # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
220     #
221     # I <- rbind(
222     # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
          ncol=s)),
223     # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
          ncol=s)),
224     # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
225     # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
226     # )
```

```
    #
    # }else{
    # I <- rbind(
    # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
    # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
    # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
    # )
    # }
  #
  # k$I <- I
  #
  #
  # ############################
  #
  # # print(-k$Knum)
  # # print(k$I)
  #
  # # vcov <- try(solve(-k$Knum))
  # vcov <- try(solve(k$I))
  # # vcov <- chol2inv(chol(k$I))
  # #vcov <- K #somente para nao trancar simulacao
  #
  # k$cov_ok = 0
  # if (class(vcov) == "try-error")
  # {
    # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
        with inversion
    # k$cov_ok = 1
    #
    # if (class(vcov) == "try-error") #
    # {
      # vcov <- k$Knum^2
      # warning("Problem with the variance and covariance matrix. The p-values
          are not correct.")
      # k$cov_ok = 2
      # }
    # }


  # k$vcov <- vcov
  # stderror <- sqrt(diag(k$vcov))
  # k$stderror <- stderror

  return(k)
}

ibetafit<-function(x)
```

```
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

  xm0<-x[which(x>0)]
  x01<-xm0[which(xm0<1)]

  #MV
  loglik<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
    phi<-par[4]

    alpha0 = ifelse(n0==0,0,alpha0)
    alpha1 = ifelse(n1==0,0,alpha1)

    #print(c(alpha0,alpha1,gama,phi))

    c<- 1- alpha0*(1-gama)-alpha1*gama
    mu<- gama*(1-alpha1)/c


    l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
    l1 = ifelse( (x == 1), log(alpha1*gama), 0)
    l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
    l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
        - mu) * phi, log = TRUE)))


    #print("soma")
    #print(sum(l0 + l1 + l2+ l3))
    sum(l0 + l1 + l2+ l3)
  }

  escore<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
    phi<-par[4]

    c<- 1- alpha0*(1-gama)-alpha1*gama
```

```
318     mu<- (gama*(1-alpha1))/c
319
320     a= mu * phi
321     b = a * (1 - mu)/mu
322
323     ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
324     mustar <- digamma(a) - digamma(b)
325
326
327     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
328     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
329     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
            mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
330     Ualpha0 = sum(Ualpha01 + Ualpha03)
331
332     Ualpha0 = ifelse(n0==0,0,Ualpha0)
333
334
335     Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
336     #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
337     Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
            gama*(1-gama)*(1-alpha0))/ (c^2) ) )
338     #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
            digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
339     Ualpha1 = sum(Ualpha11 - Ualpha13)
340
341     Ualpha1 = ifelse(n1==0,0,Ualpha1)
342
343     Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
344     Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
345     Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
            mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
346
347     Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
348
349     Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
350     #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
            digamma(b))
351     mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
352     )
353     Uphi = sum(Uphi0)
354
355
356     c(Ualpha0,Ualpha1,Ugama,Uphi)
357   }
358
359   # metodo dos momentos para iniciarlizacao
```

```
  x_bar<-mean(x01)
  sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)

  p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
  q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)

  #print(c(x_bar,sigma2_hat,p,q))

  delta0<- n0/n
  delta1<- n1/n

  a0<- delta0/(1-x_bar)
  a1<- delta1/(x_bar)

  #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
  par_ini<-c(a0,a1,p/(p+q),p+q)

  #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)

  #print("par_ini")
  #print(par_ini)

  max<-optim(par_ini,loglik,
  escore,
  method="BFGS",
  #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z<-c()
  z$mv<-max$par
  #print(z$mv)
  z$conv<-max$convergence
  z$loglik <- max$value
  return(z)
}


iuGAfit <- function(y)

{

  n <- length(y) # sample size
  n0 <- sum(y == 0) # number of zeros
  n1 <- sum(y == 1) # number of ones
```

```r
m <- n0 + n1

ys01 <- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

i01 <- (y==0)+(y==1)

#Maxima verossimilhanca

loglik <- function (par){

  alpha0 <- par[1]
  alpha1 <- par[2]
  gama <- par[3]
  phi <- par[4]

  alpha0 = ifelse(n0 == 0, 0, alpha0)
  alpha1 = ifelse(n1 == 0, 0, alpha1)

  c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
  mu <- gama * (1 - alpha1) / c
  d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))

  l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
  l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
  l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
  l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))

  #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))

  sum(l0 + l1 + l2 + l3)

}
escore <- function (par) {

  alpha0 <- par[1]
  alpha1 <- par[2]
  gama <- par[3]
  phi <- par[4]

  alpha0 <- ifelse(n0 == 0, 0, alpha0)
  alpha1 <- ifelse(n1 == 0, 0, alpha1)

  c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
  mu <- gama * (1 - alpha1) / c
  d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
```

```
453   ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
454   mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
455
456
457   Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
458   Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
          / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
          ^ 2))
459   Ualpha0 <- sum(Ualpha01 + Ualpha03)
460
461   Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
462
463   Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
464   Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
          (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
          2))
465   Ualpha1 <- sum(Ualpha11 + Ualpha13)
466
467   Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
468
469
470   Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
471   Ugama02 <- ifelse((y == 1), 1 / gama, 0)
472   Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
          + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
           ^ 2))
473
474   Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
475
476   Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
          /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
          phi) - log(mu^(1/phi)))
477
478   Uphi = sum(Uphi0)
479
480
481   c(Ualpha0, Ualpha1, Ugama, Uphi)
482 }
483
484
485 #Inicializacao
486
487 y_bar <- mean(y)
488 y_bar0 <- mean(ys01)
489
490 delta0 <- n0/n
491 delta1 <- n1/n
```

```
492
493   a0 <- delta0/(1-y_bar)
494   a1 <- delta1/(y_bar)
495
496
497   #fit <- vglm(ys01 ~ 1, gamma2)
498   #par<-Coef(fit)
499
500   #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
501   #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
502   #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
503   #phi_til = 5
504   phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
505
506   par_ini <- c(a0, a1, y_bar, phi_til)
507
508
509   max<-optim(par_ini, loglik, escore,method = "BFGS",
510   #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
511   #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
512   control=list(fnscale=-1))
513
514   if(max$convergence != 0) print("Non-convergence")
515   # return
516   z <- c()
517   z$ini <- par_ini
518   z$mv <- max$par
519   z$conv <- max$convergence
520   z$loglik <- max$value
521   return(z)
522 }
523
524 n=200
525 #n=60
526 #n=100
527 #n=200
528 NR=10000
529 k=4 # numero de parametros
530
531 Cont_AIC_IB=rep(0,NR)
532 Cont_AIC_IK=rep(0,NR)
533 Cont_AIC_IUG=rep(0,NR)
534
535
536 Cont_AICC_IB=rep(0,NR)
```

```
Cont_AICC_IK=rep(0,NR)
Cont_AICC_IUG=rep(0,NR)

Cont_BIC_IB=rep(0,NR)
Cont_BIC_IK=rep(0,NR)
Cont_BIC_IUG=rep(0,NR)

Cont_BICC_IB=rep(0,NR)
Cont_BICC_IK=rep(0,NR)
Cont_BICC_IUG=rep(0,NR)

Cont_HQ_IB=rep(0,NR)
Cont_HQ_IK=rep(0,NR)
Cont_HQ_IUG=rep(0,NR)

Cont_HQC_IB=rep(0,NR)
Cont_HQC_IK=rep(0,NR)
Cont_HQC_IUG=rep(0,NR)

Cont_LL_IB =rep(0,NR)
Cont_LL_IK=rep(0,NR)
Cont_LL_IUG=rep(0,NR)

contC=0
contC1=0

prop.z = rep(0,NR)
prop.u = rep(0,NR)
var.am=rep(0,NR) #isaac

cvm.iBE = rep(0,NR)
ks.iBE = rep(0,NR)
ad.iBE = rep(0,NR)

cvm.iuGA = rep(0,NR)
ks.iuGA = rep(0,NR)
ad.iuGA = rep(0,NR)

cvm.ikum = rep(0,NR)
ks.ikum = rep(0,NR)
ad.ikum = rep(0,NR)

#alpha0=P0/(1-gama) -> P0=P(y=0)
#alpha1=P1/(gama) -> P1=P(y=1)
#teste


```

```
alpha0 = 0.05
alpha1 = 0.05
gama = 0.7
phi = 50
alpha0
alpha1

prob.0=alpha0*(1-gama) #(prob de zero)
prob.1=alpha1*gama #(prob de um)
prob.0
prob.1


lambda = prob.0+prob.1 #(prop.infl ou seja pro.0+prop.1)
lambda
p=(prob.1)/(lambda)
p
#p = (alpha0*(1-gama))/gama #(p=1 infl1 p=0 infla 0)
omega = gama
phi = phi

lambda
p
omega
phi

omega

valores.parametros = c(alpha0, alpha1, gama, phi)

prob.z = lambda*(1-p)
prob.o = lambda*p

prob.z # probabilidade de zero
prob.o #probabilidade de um

media.v = gama

bet=(log(0.5))/(log(1-omega^phi))
bet #expressao do parametro beta

var.v =lambda*p*(1-lambda*p)+(1-lambda*p)*bet*((beta(1+2/phi,bet))-(beta(1+1/phi,
    bet))*(2*lambda*p+bet*(1-lambda)*(beta(1+1/phi,bet))))
#media.v
var.v


```

```
c= 1- alpha0*(1-gama)-alpha1*gama
mu = gama*(1-alpha1)/c
delta0 = alpha0 * (1-gama)
delta1 = alpha1 * (gama)
p2= 1- delta0-delta1

mu
delta0
delta1
p2
dp = sqrt(1/(phi+1))

cont.ib = 0
cont.ik = 0
cont.igu = 0

for(i in 1:NR) {

  y = rikum(n,lambda = lambda, p=p, omega = omega, phi = phi)
  len = length(y)
  prop.z[i] = sum(ifelse(y==0,1,0))/len
  prop.u[i] = sum(ifelse(y==1,1,0))/len
  var.am[i]=var(y) #isaac

  fitib = ibetafit(y)
  if (fitib$conv != 0)
  {
    cont.ib = cont.ib+1
  }


  fitik = ikumafit(y)
  if (fitik$conv != 0)
  {
    cont.ik = cont.ik+1
  }


  fitigu = iuGAfit(y)


  if ((fitigu$conv != 0) ){
    cont.igu=cont.igu+1
  }


  ks.iBE[i] = ifelse(ks.test(y, "pBEINF",mu=mu , sigma=dp, nu=delta0/p2, tau=
```

```
    delta1/p2)$p.value>0.05,1,0)
677
678 ad.iBE[i] = ifelse(ad.test(y, null="pBEINF",mu=mu , sigma=dp, nu=delta0/p2, tau
    =delta1/p2)$p.value>0.05,1,0)
679
680 cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF",mu=mu , sigma=dp, nu=delta0/p2,
    tau=delta1/p2)$p.value >0.05,1,0)
681
682
683 ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
    phi=phi)$p.value>0.05,1,0)
684
685 ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
    gama, phi=phi)$p.value>0.05,1,0)
686
687 cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
    gama, phi=phi)$p.value>0.05,1,0)
688
689
690 ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=lambda, p=p,omega=gama, phi=phi)$
    p.value>0.05,1,0)
691
692 ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=lambda, p=p,omega=gama, phi=
    phi)$p.value>0.05,1,0)
693
694 cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=lambda, p=p,omega=gama,
    phi=phi)$p.value>0.05,1,0)
695
696 #cvm.iuGA[i] = ifelse(cvm.test(y, "pgui",alpha0=fitigu$mv[1],alpha1=fitigu$mv
    [2],gama=fitigu$mv[3], phi=fitigu$mv[4])$p.value<0.05,1,0)
697 #cvm.ikum[i] = ifelse(cvm.test(y, "pikum",lambda=fitik$coef[1], p=fitik$coef
    [2],omega=fitik$coef[3], phi=fitik$coef[4])$p.value<0.05,1,0)
698
699 #Log verossimilhanca
700 # Beta inflacionada
701 log.vero_IB = fitib$loglik
702 # Kuma inflacionada
703 log.vero_IK= fitik$loglik
704 # Gamma Unitaria inflacionada
705 log.vero_IUG = fitigu$loglik
706
707 m.log.vero_IB = -fitib$loglik
708 # Kuma inflacionada
709 m.log.vero_IK= -fitik$loglik
710 # Gamma Unitaria inflacionada
711 m.log.vero_IUG = -fitigu$loglik
712
```

```
#Menor -log verossimilhanca

min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))

Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)




# Calculando os criterios/quantidades para comparacao

#criteiro_AIC#
# Beta inflacionada
AIC_IB = -2*log.vero_IB+2*k
# Kuma inflacionada
AIC_IK= -2*log.vero_IK+2*k
# Gamma Unitaria inflacionada
AIC_IUG = -2*log.vero_IUG+2*k

# Selecionando o menor AIC
min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))

Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)


# Beta inflacionada
AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1)
# Kuma inflacionada
AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1)
# Gamma Unitaria inflacionada
AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1)

# Selecionando o menor AICC
min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))

Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)

#criterio_BIC#
# Beta inflacionada
BIC_IB=-2*log.vero_IB+k*log(n)
#Kuma inflacionada
```

```
BIC_IK=-2*log.vero_IK+k*log(n)
# Gamma Unitaria inflacionada
BIC_IUG=-2*fitigu$loglik+k*log(n)

# Selecionando o menor BIC
min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))

Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)

#criterio_BICC#
# Beta inflacionada
BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1)
#Kuma inflacionada
BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1)
# Gamma Unitaria inflacionada
BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1)

# Selecionando o menor BICC
min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))

Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)

#criterio_HQ#
# Beta inflacionada
HQ_IB=-2*log.vero_IB+2*log(log(n))
#Kuma inflacionada
HQ_IK=-2*log.vero_IK+2*log(log(n))
# Gamma Unitaria inflacionada
HQ_IUG=-2*fitigu$loglik+2*log(log(n))

# Selecionando o menor HQ
min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))

Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)

#criterio_HQC#
# Beta inflacionada
HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1)
#Kuma inflacionada
HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1)
# Gamma Unitaria inflacionada
```

```
  HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1)

  # Selecionando o menor HQC
  min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))


  Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
  Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
  Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
  # FIM DO LACO DE MONTE CARLO
}

# Proporcao media de zeros
#mean(prop.z)
# Proporcao media de uns
#mean(prop.u)
var.amostral=mean(var.am) #isaac

#Calculando percentuais escolha distribuicao(AIC)
p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
p_AICIB
p_AICIK
p_AICIUG

#Calculando percentuais escolha distribuicao(AICC)
p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
p_AICCIB
p_AICCIK
p_AICCIUG

#Calculando percentuais escolha distribuicao(BIC)
p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
p_BICIB
p_BICIK
p_BICIUG

#Calculando percentuais escolha distribuicao(BICC)
p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
p_BICCIB
```

```
p_BICCIK
p_BICCIUG

#Calculando percentuais escolha distribuicao(HQ)
p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
p_HQIB
p_HQIK
p_HQIUG

#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG

#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
p_LLIB
p_LLIK
p_LLIUG

#resultados -log.vero
result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
#Recebendo Resultados(AIC)
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
#Recebendo Resultados(AICC)
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
#Recebendo Resultados(BIC)
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
#Recebendo Resultados(BICC)
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
#Recebendo Resultados(HQ)
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
#Recebendo Resultados(HQC)
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC
```

```
901
902 cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100
903 ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100
904 ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100
905 cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100
906 ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100
907 ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100
908 cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100
909 ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100
910 ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100
911 result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
        rej.iuGA)
912 result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
        rej.iuGA)
913 result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
        Nao.rej.iuGA)
914
915 M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
        result_HQC,result.taxa.rej.ks,
916 result.taxa.rej.ad,result.taxa.rej.cvm)
917 colnames(M) <- c("BIm.I","Kuma.I","GU.I")
918 rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
        ks",
919 "Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")
920
921 glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
        .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
922 "\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
        phi: {phi}","\n", "media: {media.v}",
923 "\n", "var: {round(var.v,4)}")
924 print("Resultados")
925 print(M)
926
927 var.amostral
928
929 M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
        )
930 colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
931 rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
        Rej.cvm")
932 xtable(M.res)
933
934 #Salvando em um arquivo
935 #write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

```
1   ##Simulacao gerando amostras a partir da kumaraswamy inflacionada em um##
2 library(gamlss)
```

```r
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
library(goftest)
source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")
set.seed(33) #Fixando a semente
source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }

  i01 <- (y==0)+(y==1)

  loglik <- function(theta)
  {
    omega <- theta[1]
    phi <- theta[2]
```

```
50
51    ########
52    l2 = 0
53    if(u!=0)
54    {
55      l2a = ifelse((y == 1), log(p), 0)
56      l2b = ifelse((y == 0), log(1-p), 0)
57      l2 = l2a + l2b
58    }
59
60    l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))
61
62    l3 = ifelse((i01 == 1), 0,
63    log(phi)+
64    log(log(0.5)/log(1-omega^phi))+
65    (phi-1)*log(y)+
66    ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
67    )
68
69    sum(l1 + l2 + l3)
70  }
71
72  escore <- function(theta)
73  {
74    omega <- theta[1]
75    phi <- theta[2]
76
77    delta = log(0.5)/log(1-omega^phi)
78    ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
          +1)
79
80    c = ifelse((y == 0), 0, ci)
81    c = ifelse((y == 1), 0, c)
82
83    Uomega <- phi*sum(c)
84
85    v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
86    (delta-1)*((y^phi)*log(y)/(1-y^phi)))
87    v = ifelse((y == 1), 0, v)
88
89    Uphi <- sum(v)
90
91    derivada=c(Uomega,Uphi)
92    derivada
93  }
94
95  ### initial values
```

```r
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

#############

opt <- optim(ini, loglik, escore, # hessian = T,
method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))

# opt <- optim(ini, loglik, escore,
# method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e-9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double.eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
# upper = rep(.Machine$double.xmax,(r+2))
# )

# opt <- optim(ini, loglik, escore)

# print(opt$par)

# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)


if (opt$conv != 0)
warning("FUNCTION DID NOT CONVERGE!")

k <- c()

# print("AQUI1")
# print(gradient(escore, opt$par))

coef <- c(lambda,p,opt$par)
```

```
141  # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
142  # print(coef)
143  k$coef <- coef
144  k$conv <- opt$conv
145  k$loglik <- opt$value
146  k$counts <- as.numeric(opt$counts[1])
147
148  # library(rootSolve)
149  # library(numDeriv)
150  # escores<-rbind(
151  # escore(coef),
152  # gradient(loglik, coef),
153  # grad(loglik, coef))
154  # print(round(escores,5))
155
156  # k$Knum<-gradient(escore, coef)
157
158  # print(k$Knum)
159  # k$Knum<-hessian(escore, coef, method="complex")
160  # print(k$Knum)
161
162
163  # # Expected information matrix
164  #
165  # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
166  # T3 = diag(as.vector( 1/diflink3(k$omega) ))
167  # T4 = diag(as.vector( 1/diflink4(k$phi) ))
168  #
169  # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
170  # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
171  # delta <- log(0.5)/log(1-k$omega^k$phi)
172  #
173  # kapa<- 0.5772156649
174  # k0<- (pi^2)/6+kapa^2-2*kapa
175  #
176  # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
         log(1-y^k$phi)+1)
177  #
178  # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
179  # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
180  # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
181  # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
182  #
183  # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
184  #
185  # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
186  #
```

```
# # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
# # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
    phi)) -
# # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
    k$phi^3))
# #
# # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
#
# mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
    +1)-kapa)/((delta-1)*k$phi)) )
#
# L = diag(-1/(k$lambda*(1-k$lambda)))
# V = diag((k$lambda-1)*k$phi^2*h2)
# M = diag(mi)
# S = diag(si)
#
#
# Igg = -t(z1) %*% L %*% (T1^2) %*% z1
# Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
# Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
# Isb = t(Ibs)
# Iss = -t(q1) %*% S %*% (T4^2) %*% q1
#
# if(u!=0)
# {
  # k$pi <- as.vector(coef[(m+1):(m+u)])
  # eta_hat2 <- as.vector(w1%*%k$pi)
  # k$p <- as.vector(linkinv2(eta_hat2))
  # T2 = diag(as.vector( 1/diflink2(k$p) ))
  # P = diag(-k$lambda/(k$p*(1-k$p)))
  # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
  #
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
      ncol=s)),
  # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
      ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
  # )
  #
  # }else{
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
  # )
```

```
    # }
  #
  # k$I <- I
  #
  #
  # ##########################
  #
  # # print(-k$Knum)
  # # print(k$I)
  #
  # # vcov <- try(solve(-k$Knum))
  # vcov <- try(solve(k$I))
  # # vcov <- chol2inv(chol(k$I))
  # #vcov <- K #somente para nao trancar simulacao
  #
  # k$cov_ok = 0
  # if (class(vcov) == "try-error")
  # {
    # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
        with inversion
    # k$cov_ok = 1
    #
    # if (class(vcov) == "try-error") #
    # {
      # vcov <- k$Knum^2
      # warning("Problem with the variance and covariance matrix. The p-values
          are not correct.")
      # k$cov_ok = 2
      # }
    # }


  # k$vcov <- vcov
  # stderror <- sqrt(diag(k$vcov))
  # k$stderror <- stderror

  return(k)
}

ibetafit<-function(x)
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

```

```
xm0<-x[which(x>0)]
x01<-xm0[which(xm0<1)]

#MV
loglik<-function(par)
{
  alpha0<-par[1]
  alpha1<-par[2]
  gama<-par[3]
  phi<-par[4]

  alpha0 = ifelse(n0==0,0,alpha0)
  alpha1 = ifelse(n1==0,0,alpha1)

  #print(c(alpha0,alpha1,gama,phi))

  c<- 1- alpha0*(1-gama)-alpha1*gama
  mu<- gama*(1-alpha1)/c


  l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
  l1 = ifelse( (x == 1), log(alpha1*gama), 0)
  l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
  l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
      - mu) * phi, log = TRUE)))


  #print("soma")
  #print(sum(l0 + l1 + l2+ l3))
  sum(l0 + l1 + l2+ l3)
}

escore<-function(par)
{
  alpha0<-par[1]
  alpha1<-par[2]
  gama<-par[3]
  phi<-par[4]

  c<- 1- alpha0*(1-gama)-alpha1*gama
  mu<- (gama*(1-alpha1))/c

  a= mu * phi
  b = a * (1 - mu)/mu

  ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
  mustar <- digamma(a) - digamma(b)
```

```
320
321
322     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
323     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
324     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
            mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
325     Ualpha0 = sum(Ualpha01 + Ualpha03)
326
327     Ualpha0 = ifelse(n0==0,0,Ualpha0)
328
329
330     Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
331     #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
332     Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
            gama*(1-gama)*(1-alpha0))/ (c^2) ) )
333     #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
            digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
334     Ualpha1 = sum(Ualpha11 - Ualpha13)
335
336     Ualpha1 = ifelse(n1==0,0,Ualpha1)
337
338     Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
339     Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
340     Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
            mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
341
342     Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
343
344     Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
345     #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
            digamma(b))
346     mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
347     )
348     Uphi = sum(Uphi0)
349
350
351     c(Ualpha0,Ualpha1,Ugama,Uphi)
352   }
353
354   # metodo dos momentos para iniciarlizacao
355   x_bar<-mean(x01)
356   sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
357
358   p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
359   q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
360
361   #print(c(x_bar,sigma2_hat,p,q))
```

```
362
363   delta0<- n0/n
364   delta1<- n1/n
365
366   a0<- delta0/(1-x_bar)
367   a1<- delta1/(x_bar)
368
369   #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
370   par_ini<-c(a0,a1,p/(p+q),p+q)
371
372   #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
373
374   #print("par_ini")
375   #print(par_ini)
376
377   max<-optim(par_ini,loglik,
378   escore,
379   method="BFGS",
380   #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
381   control=list(fnscale=-1))
382
383   if(max$convergence != 0) print("Non-convergence")
384   # return
385   z<-c()
386   z$mv<-max$par
387   #print(z$mv)
388   z$conv<-max$convergence
389   z$loglik <- max$value
390   return(z)
391 }
392
393
394 iuGAfit <- function(y)
395
396 {
397
398   n <- length(y) # sample size
399   n0 <- sum(y == 0) # number of zeros
400   n1 <- sum(y == 1) # number of ones
401   m <- n0 + n1
402
403   ys01 <- y
404   if(any(y==0)) ys01<- ys01[-which(y==0)]
405   if(any(y==1)) ys01<- ys01[-which(ys01==1)]
406
407   i01 <- (y==0)+(y==1)
```

```
408
409 #Maxima verossimilhanca
410
411 loglik <- function (par){
412
413   alpha0 <- par[1]
414   alpha1 <- par[2]
415   gama <- par[3]
416   phi <- par[4]
417
418   alpha0 = ifelse(n0 == 0, 0, alpha0)
419   alpha1 = ifelse(n1 == 0, 0, alpha1)
420
421   c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
422   mu <- gama * (1 - alpha1) / c
423   d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
424
425   l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
426   l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
427   l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
428   l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))
429
430   #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))
431
432   sum(l0 + l1 + l2 + l3)
433
434 }
435 escore <- function (par) {
436
437   alpha0 <- par[1]
438   alpha1 <- par[2]
439   gama <- par[3]
440   phi <- par[4]
441
442   alpha0 <- ifelse(n0 == 0, 0, alpha0)
443   alpha1 <- ifelse(n1 == 0, 0, alpha1)
444
445   c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
446   mu <- gama * (1 - alpha1) / c
447   d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
448   ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
449   mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
450
451
452   Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
453   Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
       / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
```

```
    ^ 2))
  Ualpha0 <- sum(Ualpha01 + Ualpha03)

  Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)

  Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
  Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
      (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
      2))
  Ualpha1 <- sum(Ualpha11 + Ualpha13)

  Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)


  Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
  Ugama02 <- ifelse((y == 1), 1 / gama, 0)
  Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
      + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
       ^ 2))

  Ugama <- sum(Ugama01 + Ugama02 + Ugama04)

  Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
      /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
      phi) - log(mu^(1/phi)))

  Uphi = sum(Uphi0)


  c(Ualpha0, Ualpha1, Ugama, Uphi)
}

#Inicializacao

y_bar <- mean(y)
y_bar0 <- mean(ys01)

delta0 <- n0/n
delta1 <- n1/n

a0 <- delta0/(1-y_bar)
a1 <- delta1/(y_bar)

#fit <- vglm(ys01 ~ 1, gamma2)
#par<-Coef(fit)

#mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
```

```
  #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
  #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
  #phi_til = 5
  phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))

  par_ini <- c(a0, a1, y_bar, phi_til)

  max<-optim(par_ini, loglik, escore,method = "BFGS",
  #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z <- c()
  z$ini <- par_ini
  z$mv <- max$par
  z$conv <- max$convergence
  z$loglik <- max$value
  return(z)
}

n=30
#n=60
#n=100
#n=200
NR=10000
k=3 # numero de parametros

Cont_AIC_IB=rep(0,NR)
Cont_AIC_IK=rep(0,NR)
Cont_AIC_IUG=rep(0,NR)


Cont_AICC_IB=rep(0,NR)
Cont_AICC_IK=rep(0,NR)
Cont_AICC_IUG=rep(0,NR)

Cont_BIC_IB=rep(0,NR)
Cont_BIC_IK=rep(0,NR)
Cont_BIC_IUG=rep(0,NR)

Cont_BICC_IB=rep(0,NR)
Cont_BICC_IK=rep(0,NR)
Cont_BICC_IUG=rep(0,NR)
```

```
539
540 Cont_HQ_IB=rep(0,NR)
541 Cont_HQ_IK=rep(0,NR)
542 Cont_HQ_IUG=rep(0,NR)
543
544 Cont_HQC_IB=rep(0,NR)
545 Cont_HQC_IK=rep(0,NR)
546 Cont_HQC_IUG=rep(0,NR)
547
548 Cont_LL_IB =rep(0,NR)
549 Cont_LL_IK=rep(0,NR)
550 Cont_LL_IUG=rep(0,NR)
551
552 contC=0
553 contC1=0
554
555 prop.z = rep(0,NR)
556 prop.u = rep(0,NR)
557 var.am=rep(0,NR) #isaac
558
559 cvm.iBE = rep(0,NR)
560 ks.iBE = rep(0,NR)
561 ad.iBE = rep(0,NR)
562
563 cvm.iuGA = rep(0,NR)
564 ks.iuGA = rep(0,NR)
565 ad.iuGA = rep(0,NR)
566
567 cvm.ikum = rep(0,NR)
568 ks.ikum = rep(0,NR)
569 ad.ikum = rep(0,NR)
570
571 alpha0 = 0
572 alpha1 = 0.07
573 gama = 0.7
574 phi = 10
575
576 lambda = alpha1
577 p = 1 #(p=1 infl1 p=0 infla 0)
578 omega = gama
579 phi = phi
580
581 omega
582
583 valores.parametros = c(alpha0, alpha1, gama, phi)
584
585 prob.z = lambda*(1-p)
```

```
prob.o = lambda*p

prob.z # probabilidade de zero
prob.o #probabilidade de um

media.v = gama

bet=(log(0.5))/(log(1-omega^phi))
bet #expressao do paramentro beta

#var.v =lambda*p*(1-lambda*p)*bet*((beta(1+2/phi,bet))-(beta(1+1/phi,bet))*(2*
    lambda*p+bet*(1-lambda)*(beta(1+1/phi,bet))))

var.v =lambda*p*(1-lambda*p)+(1-lambda*p)*bet*((beta(1+2/phi,bet))-(beta(1+1/phi,
    bet))*(2*lambda*p+bet*(1-lambda)*(beta(1+1/phi,bet))))
#media.v
var.v

#Bayes - BEINF

c= 1- alpha0*(1-gama)-alpha1*gama
mu = gama*(1-alpha1)/c
delta0 = alpha0 * (1-gama)
delta1 = alpha1 * (gama)
p2= 1- delta0-delta1

mu
delta0
delta1
p2
dp = sqrt(1/(phi+1))


cont.ib = 0
cont.ik = 0
cont.igu = 0

for(i in 1:NR) {


  y = rikum(n,lambda = lambda, p=p, omega = omega, phi = phi)
  len = length(y)
  prop.z[i] = sum(ifelse(y==0,1,0))/len
  prop.u[i] = sum(ifelse(y==1,1,0))/len
  var.am[i]=var(y) #isaac

  fitib = ibetafit(y)
```

```
631  if (fitib$conv != 0)
632  {
633    cont.ib = cont.ib+1
634  }
635
636
637  fitik = ikumafit(y)
638  if (fitik$conv != 0)
639  {
640    cont.ik = cont.ik+1
641  }
642
643  fitigu = iuGAfit(y)
644
645
646  if ((fitigu$conv != 0) ){
647    cont.igu=cont.igu+1
648  }
649
650  ks.iBE[i] = ifelse(ks.test(y, "pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$p.value
         >0.05,1,0)
651
652  ad.iBE[i] = ifelse(ad.test(y, null="pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$p.
         value>0.05,1,0)
653
654  cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$
         p.value >0.05,1,0)
655
656
657  ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
         phi=phi)$p.value>0.05,1,0)
658
659  ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
         gama, phi=phi)$p.value>0.05,1,0)
660
661  cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
         gama, phi=phi)$p.value>0.05,1,0)
662
663
664  ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=alpha1, p=1,omega=gama, phi=phi)$
         p.value>0.05,1,0)
665
666  ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=alpha1, p=1,omega=gama, phi=
         phi)$p.value>0.05,1,0)
667
668  cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=alpha1, p=1,omega=gama,
         phi=phi)$p.value>0.05,1,0)
```

```
669
670 #Log verossimilhanca
671 # Beta inflacionada
672 log.vero_IB = fitib$loglik
673 # Kuma inflacionada
674 log.vero_IK= fitik$loglik
675 # Gamma Unitaria inflacionada
676 log.vero_IUG = fitigu$loglik
677
678 m.log.vero_IB = -fitib$loglik
679 # Kuma inflacionada
680 m.log.vero_IK= -fitik$loglik
681 # Gamma Unitaria inflacionada
682 m.log.vero_IUG = -fitigu$loglik
683
684 #Menor -log verossimilhanca
685
686 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
687
688 Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
689 Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
690 Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
691
692 # Calculando os criterios/quantidades para comparacao
693
694 #criteiro_AIC#
695 AIC_IB = -2*log.vero_IB+2*k  #BI(Beta inflacionada)
696 AIC_IK= -2*log.vero_IK+2*k #KI(Kuma inflacionada)
697 AIC_IUG = -2*log.vero_IUG+2*k  #GUI(Gamma Unitaria inflacionada)
698
699 # Selecionando o menor AIC
700 min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
701
702 Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
703 Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
704 Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
705
706 AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1) #BI
707 AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1) #KI
708 AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1) #GUI
709
710 # Selecionando o menor AICC
711 min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
712
713 Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
714 Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
715 Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)
```

```
716
717  #criterio_BIC#
718  BIC_IB=-2*log.vero_IB+k*log(n) #BI
719  BIC_IK=-2*log.vero_IK+k*log(n) #KI
720  BIC_IUG=-2*fitigu$loglik+k*log(n) #GUI
721
722  # Selecionando o menor BIC
723  min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))
724
725  Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
726  Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
727  Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)
728
729  #criterio_BICC#
730  BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1) #BI
731  BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1) #KI
732  BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1) #GUI
733
734  # Selecionando o menor BICC
735  min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))
736
737  Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
738  Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
739  Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)
740
741  #criterio_HQ#
742  HQ_IB=-2*log.vero_IB+2*log(log(n)) #BI
743  HQ_IK=-2*log.vero_IK+2*log(log(n)) #KI
744  HQ_IUG=-2*fitigu$loglik+2*log(log(n)) #GUI
745
746  # Selecionando o menor HQ
747  min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))
748
749  Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
750  Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
751  Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)
752
753  #criterio_HQC#
754  HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1) #BI
755  HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1) #KI
756  HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1) #GUI
757
758  # Selecionando o menor HQC
759  min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))
760
761  Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
762  Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
```

```
763   Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
764   # FIM DO LACO DE MONTE CARLO
765 }
766
767 # Proporcao media de zeros
768 #mean(prop.z)
769 # Proporcao media de uns
770 #mean(prop.u)
771 var.amostral=mean(var.am) #isaac
772
773 #Calculando percentuais escolha distribuicao(AIC)
774 p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
775 p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
776 p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
777 p_AICIB
778 p_AICIK
779 p_AICIUG
780
781 #Calculando percentuais escolha distribuicao(AICC)
782 p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
783 p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
784 p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
785 p_AICCIB
786 p_AICCIK
787 p_AICCIUG
788
789 #Calculando percentuais escolha distribuicao(BIC)
790 p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
791 p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
792 p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
793 p_BICIB
794 p_BICIK
795 p_BICIUG
796
797 #Calculando percentuais escolha distribuicao(BICC)
798 p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
799 p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
800 p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
801 p_BICCIB
802 p_BICCIK
803 p_BICCIUG
804
805 #Calculando percentuais escolha distribuicao(HQ)
806 p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
807 p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
808 p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
809 p_HQIB
```

```
p_HQIK
p_HQIUG

#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG

#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
p_LLIB
p_LLIK
p_LLIUG

result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC

cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100
ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100
ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100
cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100
ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100
ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100
cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100
ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100
ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
```

```
    rej.iuGA)
856
857 result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)
858
859 M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
860 result.taxa.rej.ad,result.taxa.rej.cvm)
861 colnames(M) <- c("BIm.I","Kuma.I","GU.I")
862 rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
    ks",
863 "Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")
864
865 glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
    .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
866 "\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
    phi: {phi}","\n", "media: {media.v}",
867 "\n", "var: {round(var.v,4)}")
868 print("Resultados")
869 print(M)
870
871 var.amostral
872
873 M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
    )
874 colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
875 rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
    Rej.cvm")
876 xtable(M.res)
877
878 #Salvando em um arquivo
879 #write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

```
1  ##Simulacao gerando amostras a partir da gama unitaria inflacionada em zero##
2  library(gamlss)
3  library(gamlss.dist)
4  library(gamlss.data)
5  library(betareg)
6  library(MASS)
7  library(goftest)
8  source("ikumafit.R")
9  source("iuga_fitv4.R")
10 source("ibetafit.R")
11 source("iuga_v2.R")
12 set.seed(33) #Fixando a semente
13 source("ikum-omega-phi.R")
14 # library(optimx)
```

```r
library(rootSolve)
library(VGAM)
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }

  i01 <- (y==0)+(y==1)

  loglik <- function(theta)
  {
    omega <- theta[1]
    phi <- theta[2]

    ########
    l2 = 0
    if(u!=0)
    {
      l2a = ifelse((y == 1), log(p), 0)
      l2b = ifelse((y == 0), log(1-p), 0)
      l2 = l2a + l2b
    }

    l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

```

```
  l3 = ifelse((i01 == 1), 0,
  log(phi)+
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}

escore <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  delta = log(0.5)/log(1-omega^phi)
  ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
      +1)

  c = ifelse((y == 0), 0, ci)
  c = ifelse((y == 1), 0, c)

  Uomega <- phi*sum(c)

  v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
  (delta-1)*((y^phi)*log(y)/(1-y^phi)))
  v = ifelse((y == 1), 0, v)

  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)
```

```
108
109 #############
110
111 opt <- optim(ini, loglik, escore, # hessian = T,
112 method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))
113
114 # opt <- optim(ini, loglik, escore,
115 # method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e
    -9),
116 # # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double
    .eps),
117 # # upper = rep(.Machine$double.xmax,(r+2))
118 # lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
119 # upper = rep(.Machine$double.xmax,(r+2))
120 # )
121
122 # opt <- optim(ini, loglik, escore)
123
124 # print(opt$par)
125
126 # print(names(opt))
127 # print(summary(opt) )
128 # print(summary(opt)[1:6] )
129 # print(opt$kkt1)
130
131
132 if (opt$conv != 0)
133 warning("FUNCTION DID NOT CONVERGE!")
134
135 k <- c()
136
137 # print("AQUI1")
138 # print(gradient(escore, opt$par))
139
140 coef <- c(lambda,p,opt$par)
141 # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
142 # print(coef)
143 k$coef <- coef
144 k$conv <- opt$conv
145 k$loglik <- opt$value
146 k$counts <- as.numeric(opt$counts[1])
147
148 # library(rootSolve)
149 # library(numDeriv)
150 # escores<-rbind(
151 # escore(coef),
152 # gradient(loglik, coef),
```

```
153  # grad(loglik, coef))
154  # print(round(escores,5))
155
156  # k$Knum<-gradient(escore, coef)
157
158  # print(k$Knum)
159  # k$Knum<-hessian(escore, coef, method="complex")
160  # print(k$Knum)
161
162
163  # # Expected information matrix
164  #
165  # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
166  # T3 = diag(as.vector( 1/diflink3(k$omega) ))
167  # T4 = diag(as.vector( 1/diflink4(k$phi) ))
168  #
169  # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
170  # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
171  # delta <- log(0.5)/log(1-k$omega^k$phi)
172  #
173  # kapa<- 0.5772156649
174  # k0<- (pi^2)/6+kapa^2-2*kapa
175  #
176  # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
         log(1-y^k$phi)+1)
177  #
178  # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
179  # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
180  # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
181  # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
182  #
183  # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
184  #
185  # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
186  #
187  # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
188  # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
         phi)) -
189  # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
         k$phi^3))
190  # #
191  # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
192  #
193  # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
         +1)-kapa)/((delta-1)*k$phi)) )
194  #
195  # L = diag(-1/(k$lambda*(1-k$lambda)))
```

```
196   # V = diag((k$lambda-1)*k$phi^2*h2)
197   # M = diag(mi)
198   # S = diag(si)
199   #
200   #
201   # Igg = -t(z1) %*% L %*% (T1^2) %*% z1
202   # Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
203   # Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
204   # Isb = t(Ibs)
205   # Iss = -t(q1) %*% S %*% (T4^2) %*% q1
206   #
207   # if(u!=0)
208   # {
209     # k$pi <- as.vector(coef[(m+1):(m+u)])
210     # eta_hat2 <- as.vector(w1%*%k$pi)
211     # k$p <- as.vector(linkinv2(eta_hat2))
212     # T2 = diag(as.vector( 1/diflink2(k$p) ))
213     # P = diag(-k$lambda/(k$p*(1-k$p)))
214     # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
215     #
216     # I <- rbind(
217     # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
            ncol=s)),
218     # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
            ncol=s)),
219     # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
220     # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
221     # )
222     #
223     # }else{
224     # I <- rbind(
225     # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
226     # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
227     # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
228     # )
229     # }
230   #
231   # k$I <- I
232   #
233   #
234   # ############################
235   #
236   # # print(-k$Knum)
237   # # print(k$I)
238   #
239   # # vcov <- try(solve(-k$Knum))
240   # vcov <- try(solve(k$I))
```

```
241   # # vcov <- chol2inv(chol(k$I))
242   # #vcov <- K #somente para nao trancar simulacao
243   #
244   # k$cov_ok = 0
245   # if (class(vcov) == "try-error")
246   # {
247     # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
          with inversion
248     # k$cov_ok = 1
249     #
250     # if (class(vcov) == "try-error") #
251     # {
252       # vcov <- k$Knum^2
253       # warning("Problem with the variance and covariance matrix. The p-values
            are not correct.")
254       # k$cov_ok = 2
255       # }
256     # }
257
258
259   # k$vcov <- vcov
260   # stderror <- sqrt(diag(k$vcov))
261   # k$stderror <- stderror
262
263   return(k)
264 }
265
266 ibetafit<-function(x)
267 {
268
269   n<- length(x) # sample size
270   n0<- sum(x==0) # number of zeros
271   n1<- sum(x==1) # number of ones
272   m<- n0 + n1
273
274   xm0<-x[which(x>0)]
275   x01<-xm0[which(xm0<1)]
276
277   #MV
278   loglik<-function(par)
279   {
280     alpha0<-par[1]
281     alpha1<-par[2]
282     gama<-par[3]
283     phi<-par[4]
284
285     alpha0 = ifelse(n0==0,0,alpha0)
```

```
286     alpha1 = ifelse(n1==0,0,alpha1)
287
288     #print(c(alpha0,alpha1,gama,phi))
289
290     c<- 1- alpha0*(1-gama)-alpha1*gama
291     mu<- gama*(1-alpha1)/c
292
293
294     l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
295     l1 = ifelse( (x == 1), log(alpha1*gama), 0)
296     l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
297     l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
            - mu) * phi, log = TRUE)))
298
299
300     #print("soma")
301     #print(sum(l0 + l1 + l2+ l3))
302     sum(l0 + l1 + l2+ l3)
303   }
304
305   escore<-function(par)
306   {
307     alpha0<-par[1]
308     alpha1<-par[2]
309     gama<-par[3]
310     phi<-par[4]
311
312     c<- 1- alpha0*(1-gama)-alpha1*gama
313     mu<- (gama*(1-alpha1))/c
314
315     a= mu * phi
316     b = a * (1 - mu)/mu
317
318     ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
319     mustar <- digamma(a) - digamma(b)
320
321     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
322     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
323     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
            mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
324     Ualpha0 = sum(Ualpha01 + Ualpha03)
325
326     Ualpha0 = ifelse(n0==0,0,Ualpha0)
327
328
329     Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
330     #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
```

```
331     Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
            gama*(1-gama)*(1-alpha0))/ (c^2) ) )
332     #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
            digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
333     Ualpha1 = sum(Ualpha11 - Ualpha13)
334
335     Ualpha1 = ifelse(n1==0,0,Ualpha1)
336
337     Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
338     Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
339     Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
            mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
340
341     Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
342
343     Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
344     #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
            digamma(b))
345     mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
346     )
347     Uphi = sum(Uphi0)
348
349
350     c(Ualpha0,Ualpha1,Ugama,Uphi)
351   }
352
353   # metodo dos momentos para iniciarlizacao
354   x_bar<-mean(x01)
355   sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
356
357   p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
358   q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
359
360   #print(c(x_bar,sigma2_hat,p,q))
361
362   delta0<- n0/n
363   delta1<- n1/n
364
365   a0<- delta0/(1-x_bar)
366   a1<- delta1/(x_bar)
367
368   #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
369   par_ini<-c(a0,a1,p/(p+q),p+q)
370
371   #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
372
373   #print("par_ini")
```

```
374   #print(par_ini)
375
376   max<-optim(par_ini,loglik,
377   escore,
378   method="BFGS",
379   #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
380   control=list(fnscale=-1))
381
382   if(max$convergence != 0) print("Non-convergence")
383   # return
384   z<-c()
385   z$mv<-max$par
386   #print(z$mv)
387   z$conv<-max$convergence
388   z$loglik <- max$value
389   return(z)
390 }
391
392 iuGAfit <- function(y)
393
394 {
395   n <- length(y) # sample size
396   n0 <- sum(y == 0) # number of zeros
397   n1 <- sum(y == 1) # number of ones
398   m <- n0 + n1
399
400   ys01 <- y
401   if(any(y==0)) ys01<- ys01[-which(y==0)]
402   if(any(y==1)) ys01<- ys01[-which(ys01==1)]
403
404   i01 <- (y==0)+(y==1)
405
406   #Maxima verossimilhanca
407
408   loglik <- function (par){
409
410     alpha0 <- par[1]
411     alpha1 <- par[2]
412     gama <- par[3]
413     phi <- par[4]
414
415     alpha0 = ifelse(n0 == 0, 0, alpha0)
416     alpha1 = ifelse(n1 == 0, 0, alpha1)
417
418     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
419     mu <- gama * (1 - alpha1) / c
```

```
  d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))

  l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
  l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
  l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
  l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))

  #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))

  sum(l0 + l1 + l2 + l3)

}
escore <- function (par) {

  alpha0 <- par[1]
  alpha1 <- par[2]
  gama <- par[3]
  phi <- par[4]

  alpha0 <- ifelse(n0 == 0, 0, alpha0)
  alpha1 <- ifelse(n1 == 0, 0, alpha1)

  c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
  mu <- gama * (1 - alpha1) / c
  d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
  ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
  mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)


  Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
  Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
      / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
      ^ 2))
  Ualpha0 <- sum(Ualpha01 + Ualpha03)

  Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)

  Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
  Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
      (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
      2))
  Ualpha1 <- sum(Ualpha11 + Ualpha13)

  Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)


  Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
```

```
463   Ugama02 <- ifelse((y == 1), 1 / gama, 0)
464   Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
          + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
           ^ 2))
465
466   Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
467
468   Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
          /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
          phi) - log(mu^(1/phi)))
469
470   Uphi = sum(Uphi0)
471
472
473   c(Ualpha0, Ualpha1, Ugama, Uphi)
474 }
475
476 #Inicializacao
477
478 y_bar <- mean(y)
479 y_bar0 <- mean(ys01)
480
481 delta0 <- n0/n
482 delta1 <- n1/n
483
484 a0 <- delta0/(1-y_bar)
485 a1 <- delta1/(y_bar)
486
487
488 #fit <- vglm(ys01 ~ 1, gamma2)
489 #par<-Coef(fit)
490
491 #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
492 #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
493 #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
494 #phi_til = 5
495 phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
496
497 par_ini <- c(a0, a1, y_bar, phi_til)
498
499
500 max<-optim(par_ini, loglik, escore,method = "BFGS",
501 #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
502 #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
503 control=list(fnscale=-1))
```

```
504
505   if(max$convergence != 0) print("Non-convergence")
506   # return
507   z <- c()
508   z$ini <- par_ini
509   z$mv <- max$par
510   z$conv <- max$convergence
511   z$loglik <- max$value
512   return(z)
513 }
514
515 n=30
516 #n=60
517 #n=100
518 #n=200
519 NR=10000
520 k=3 # numero de parametros
521
522 Cont_AIC_IB=rep(0,NR)
523 Cont_AIC_IK=rep(0,NR)
524 Cont_AIC_IUG=rep(0,NR)
525
526 Cont_AICC_IB=rep(0,NR)
527 Cont_AICC_IK=rep(0,NR)
528 Cont_AICC_IUG=rep(0,NR)
529
530 Cont_BIC_IB=rep(0,NR)
531 Cont_BIC_IK=rep(0,NR)
532 Cont_BIC_IUG=rep(0,NR)
533
534 Cont_BICC_IB=rep(0,NR)
535 Cont_BICC_IK=rep(0,NR)
536 Cont_BICC_IUG=rep(0,NR)
537
538 Cont_HQ_IB=rep(0,NR)
539 Cont_HQ_IK=rep(0,NR)
540 Cont_HQ_IUG=rep(0,NR)
541
542 Cont_HQC_IB=rep(0,NR)
543 Cont_HQC_IK=rep(0,NR)
544 Cont_HQC_IUG=rep(0,NR)
545
546 Cont_LL_IB =rep(0,NR)
547 Cont_LL_IK=rep(0,NR)
548 Cont_LL_IUG=rep(0,NR)
549
550 contC=0
```

```
contC1=0

prop.z = rep(0,NR)
prop.u = rep(0,NR)

cvm.iBE = rep(0,NR)
ks.iBE = rep(0,NR)
ad.iBE = rep(0,NR)

cvm.iuGA = rep(0,NR)
ks.iuGA = rep(0,NR)
ad.iuGA = rep(0,NR)

cvm.ikum = rep(0,NR)
ks.ikum = rep(0,NR)
ad.ikum = rep(0,NR)

alpha0 = 0.14
alpha1 = 0
gama = 0.5
phi = 50

prob.0=alpha0*(1-gama) #(prob de zero)
prob.1=alpha1*gama #(prob de um)
prob.0
prob.1

lambda = prob.0+prob.1 #(prop.infl ou seja pro.0+prop.1)
lambda
p=(prob.1)/(lambda)
p

valores.parametros = c(alpha0, alpha1, gama, phi)

prob.z = alpha0*(1-gama)
prob.o = alpha1*gama

prob.z # probabilidade de zero
prob.o #probabilidade de um

media.v = gama
var.v = alpha1*gama*(1-alpha1*gama)+gama*(1-alpha1)*((2-((gama*(1-alpha1))/(1-
    alpha0*(1-gama)-alpha1*gama))^(1/phi))^(-phi)-gama*(1+alpha1))

media.v
var.v

```

```
#Bayes - BEINF

c= 1- alpha0*(1-gama)-alpha1*gama
mu = gama*(1-alpha1)/c
delta0 = alpha0 * (1-gama)
delta1 = alpha1 * (gama)
p2= 1- delta0-delta1

mu
delta0
delta1
p2
dp = sqrt(1/(phi+1))


cont.ib = 0
cont.ik = 0
cont.igu = 0

for(i in 1:NR) {


  y = riGU(n,alpha0 = alpha0, alpha1=alpha1, gama = gama, phi = phi)
  len = length(y)
  prop.z[i] = sum(ifelse(y==0,1,0))/len
  prop.u[i] = sum(ifelse(y==1,1,0))/len


  fitib = ibetafit(y)
  if (fitib$conv != 0)
  {
    cont.ib = cont.ib+1
  }


  fitik = ikumafit(y)
  if (fitik$conv != 0)
  {
    cont.ik = cont.ik+1
  }

  fitigu = iuGAfit(y)


  if ((fitigu$conv != 0) ){
    cont.igu=cont.igu+1
  }
```

```
644
645
646 ks.iBE[i] = ifelse(ks.test(y, "pBEINF0",mu=mu , sigma=dp, nu=delta0/p2)$p.value
        >0.05,1,0)
647
648 ad.iBE[i] = ifelse(ad.test(y, null="pBEINF0",mu=mu , sigma=dp, nu=delta0/p2)$p.
        value>0.05,1,0)
649
650 cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF0",mu=mu , sigma=dp, nu=delta0/p2)$
        p.value >0.05,1,0)
651
652 ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
        phi=phi)$p.value>0.05,1,0)
653
654 ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
655
656 cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
657
658 ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=lambda, p=p,omega=gama, phi=phi)$
        p.value>0.05,1,0)
659
660 ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=lambda, p=p,omega=gama, phi=
        phi)$p.value>0.05,1,0)
661
662 cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=lambda, p=p,omega=gama,
        phi=phi)$p.value>0.05,1,0)
663
664 #Log verossimilhanca
665 # Beta inflacionada
666 log.vero_IB = fitib$loglik
667 # Kuma inflacionada
668 log.vero_IK= fitik$loglik
669 # Gamma Unitaria inflacionada
670 log.vero_IUG = fitigu$loglik
671
672 m.log.vero_IB = -fitib$loglik
673 # Kuma inflacionada
674 m.log.vero_IK= -fitik$loglik
675 # Gamma Unitaria inflacionada
676 m.log.vero_IUG = -fitigu$loglik
677
678 #Menor -log verossimilhanca
679
680 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
681
```

```
682  Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
683  Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
684  Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
685
686  # Calculando os criterios/quantidades para comparacao
687
688  #criteiro_AIC#
689  AIC_IB = -2*log.vero_IB+2*k #BI(Beta inflacionada)
690  AIC_IK= -2*log.vero_IK+2*k #KI(Kuma inflacionada)
691  AIC_IUG = -2*log.vero_IUG+2*k #GUI(Gama Unitaria Inflacionada)
692
693  # Selecionando o menor AIC
694  min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
695
696  Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
697  Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
698  Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
699
700  AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1) #BI
701  AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1) #KI
702  AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1) #GUI
703
704  # Selecionando o menor AICC
705  min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
706
707  Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
708  Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
709  Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)
710
711  #criterio_BIC#
712  BIC_IB=-2*log.vero_IB+k*log(n) #BI
713  BIC_IK=-2*log.vero_IK+k*log(n) #KI
714  BIC_IUG=-2*fitigu$loglik+k*log(n) #GUI
715
716  # Selecionando o menor BIC
717  min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))
718
719  Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
720  Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
721  Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)
722
723  #criterio_BICC#
724  BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1) #BI
725  BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1) #KI
726  BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1) #GUI
727
728  # Selecionando o menor BICC
```

```
  min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))

  Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
  Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
  Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)

  #criterio_HQ#
  HQ_IB=-2*log.vero_IB+2*log(log(n)) #BI
  HQ_IK=-2*log.vero_IK+2*log(log(n)) #KI
  HQ_IUG=-2*fitigu$loglik+2*log(log(n)) #GUI

  # Selecionando o menor HQ
  min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))

  Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
  Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
  Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)

  #criterio_HQC#
  HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1) #BI
  HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1) #KI
  HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1) #GUI

  # Selecionando o menor HQC
  min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))

  Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
  Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
  Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
  # FIM DO LACO DE MONTE CARLO
}

# Proporcao media de zeros
#mean(prop.z)
# Proporcao media de uns
#mean(prop.u)

#Calculando percentuais escolha distribuicao(AIC)
p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
p_AICIB
p_AICIK
p_AICIUG

#Calculando percentuais escolha distribuicao(AICC)
p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
```

```
p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
p_AICCIB
p_AICCIK
p_AICCIUG

#Calculando percentuais escolha distribuicao(BIC)
p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
p_BICIB
p_BICIK
p_BICIUG

#Calculando percentuais escolha distribuicao(BICC)
p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
p_BICCIB
p_BICCIK
p_BICCIUG

#Calculando percentuais escolha distribuicao(HQ)
p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
p_HQIB
p_HQIK
p_HQIUG

#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG

#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
p_LLIB
p_LLIK
p_LLIUG

result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
```

```
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC

cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100

ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100

ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100

cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100

ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100

ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100

cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100

ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100

ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)

M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
result.taxa.rej.ad,result.taxa.rej.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
    ks",
"Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")
```

```
865
866 glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
    .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
867 "\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
    phi: {phi}","\n", "media: {media.v}",
868 "\n", "var: {round(var.v,4)}")
869 print("Resultados")
870 print(M)
871
872 M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
    )
873 colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
874 rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
    Rej.cvm")
875 xtable(M.res)
876
877 #Salvando em um arquivo
878 #write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

```
1  ##Simulacao gerando amostras a partir da gama unitaria inflacionada em zero e
       um##
2 library(gamlss)
3 library(gamlss.dist)
4 library(gamlss.data)
5 library(betareg)
6 library(MASS)
7 library(goftest)
8 source("ikumafit.R")
9 source("iuga_fitv4.R")
10 source("ibetafit.R")
11 source("iuga_v2.R")
12 set.seed(33) #Fixando a semente
13 source("ikum-omega-phi.R")
14 # library(optimx)
15 library(rootSolve)
16 library(VGAM)
17 library(xtable)
18 # library(fitdistrplus)
19
20 "ikumafit" <- function(y){
21
22   y <- as.vector(y)
23
24   n <- length(y)
25   n0<- sum(y==0) # number of zeros
26   n1<- sum(y==1) # number of ones
27
```

```
lambda <- (n0+n1)/n # lambda estimator
p <- n1/(n0+n1)# p estimator

u<-1
if(n0 == 0)
{
  p=1
  u=0
}

if(n1 == 0)
{
  p=0
  u=0
}

i01 <- (y==0)+(y==1)

loglik <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  ########
  l2 = 0
  if(u!=0)
  {
    l2a = ifelse((y == 1), log(p), 0)
    l2b = ifelse((y == 0), log(1-p), 0)
    l2 = l2a + l2b
  }

  l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

  l3 = ifelse((i01 == 1), 0,
  log(phi)+
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}

escore <- function(theta)
{
  omega <- theta[1]
```

```
  phi <- theta[2]

  delta = log(0.5)/log(1-omega^phi)
  ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
      +1)

  c = ifelse((y == 0), 0, ci)
  c = ifelse((y == 1), 0, c)

  Uomega <- phi*sum(c)

  v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
  (delta-1)*((y^phi)*log(y)/(1-y^phi)))
  v = ifelse((y == 1), 0, v)

  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

#############

opt <- optim(ini, loglik, escore, # hessian = T,
method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))

# opt <- optim(ini, loglik, escore,
# method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e
    -9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double
    .eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
```

```
119  # upper = rep(.Machine$double.xmax,(r+2))
120  # )
121
122  # opt <- optim(ini, loglik, escore)
123
124  # print(opt$par)
125
126  # print(names(opt))
127  # print(summary(opt) )
128  # print(summary(opt)[1:6] )
129  # print(opt$kkt1)
130
131
132  if (opt$conv != 0)
133  warning("FUNCTION DID NOT CONVERGE!")
134
135  k <- c()
136
137  # print("AQUI1")
138  # print(gradient(escore, opt$par))
139
140  coef <- c(lambda,p,opt$par)
141  # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
142  # print(coef)
143  k$coef <- coef
144  k$conv <- opt$conv
145  k$loglik <- opt$value
146  k$counts <- as.numeric(opt$counts[1])
147
148  # library(rootSolve)
149  # library(numDeriv)
150  # escores<-rbind(
151  # escore(coef),
152  # gradient(loglik, coef),
153  # grad(loglik, coef))
154  # print(round(escores,5))
155
156  # k$Knum<-gradient(escore, coef)
157
158  # print(k$Knum)
159  # k$Knum<-hessian(escore, coef, method="complex")
160  # print(k$Knum)
161
162
163  # # Expected information matrix
164  #
165  # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
```

```
166 # T3 = diag(as.vector( 1/diflink3(k$omega) ))
167 # T4 = diag(as.vector( 1/diflink4(k$phi) ))
168 #
169 # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
170 # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
171 # delta <- log(0.5)/log(1-k$omega^k$phi)
172 #
173 # kapa<- 0.5772156649
174 # k0<- (pi^2)/6+kapa^2-2*kapa
175 #
176 # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
        log(1-y^k$phi)+1)
177 #
178 # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
179 # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
180 # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
181 # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
182 #
183 # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
184 #
185 # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
186 #
187 # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
188 # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
        phi)) -
189 # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
        k$phi^3))
190 # #
191 # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
192 #
193 # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
        +1)-kapa)/((delta-1)*k$phi)) )
194 #
195 # L = diag(-1/(k$lambda*(1-k$lambda)))
196 # V = diag((k$lambda-1)*k$phi^2*h2)
197 # M = diag(mi)
198 # S = diag(si)
199 #
200 #
201 # Igg = -t(z1) %*% L %*% (T1^2) %*% z1
202 # Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
203 # Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
204 # Isb = t(Ibs)
205 # Iss = -t(q1) %*% S %*% (T4^2) %*% q1
206 #
207 # if(u!=0)
208 # {
```

```
209     # k$pi <- as.vector(coef[(m+1):(m+u)])
210     # eta_hat2 <- as.vector(w1%*%k$pi)
211     # k$p <- as.vector(linkinv2(eta_hat2))
212     # T2 = diag(as.vector( 1/diflink2(k$p) ))
213     # P = diag(-k$lambda/(k$p*(1-k$p)))
214     # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
215     #
216     # I <- rbind(
217     # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
            ncol=s)),
218     # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
            ncol=s)),
219     # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
220     # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
221     # )
222     #
223     # }else{
224     # I <- rbind(
225     # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
226     # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
227     # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
228     # )
229     # }
230   #
231   # k$I <- I
232   #
233   #
234   # ############################
235   #
236   # # print(-k$Knum)
237   # # print(k$I)
238   #
239   # # vcov <- try(solve(-k$Knum))
240   # vcov <- try(solve(k$I))
241   # # vcov <- chol2inv(chol(k$I))
242   # #vcov <- K #somente para nao trancar simulacao
243   #
244   # k$cov_ok = 0
245   # if (class(vcov) == "try-error")
246   # {
247     # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
            with inversion
248     # k$cov_ok = 1
249     #
250     # if (class(vcov) == "try-error") #
251     # {
252       # vcov <- k$Knum^2
```

```
253       # warning("Problem with the variance and covariance matrix. The p-values
              are not correct.")
254       # k$cov_ok = 2
255       # }
256     # }
257
258
259   # k$vcov <- vcov
260   # stderror <- sqrt(diag(k$vcov))
261   # k$stderror <- stderror
262
263   return(k)
264 }
265
266 ibetafit<-function(x)
267 {
268
269   n<- length(x) # sample size
270   n0<- sum(x==0) # number of zeros
271   n1<- sum(x==1) # number of ones
272   m<- n0 + n1
273
274   xm0<-x[which(x>0)]
275   x01<-xm0[which(xm0<1)]
276
277   #MV
278   loglik<-function(par)
279   {
280     alpha0<-par[1]
281     alpha1<-par[2]
282     gama<-par[3]
283     phi<-par[4]
284
285     alpha0 = ifelse(n0==0,0,alpha0)
286     alpha1 = ifelse(n1==0,0,alpha1)
287
288     #print(c(alpha0,alpha1,gama,phi))
289
290     c<- 1- alpha0*(1-gama)-alpha1*gama
291     mu<- gama*(1-alpha1)/c
292
293
294     l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
295     l1 = ifelse( (x == 1), log(alpha1*gama), 0)
296     l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
297     l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
          - mu) * phi, log = TRUE)))
```

```
298
299
300     #print("soma")
301     #print(sum(l0 + l1 + l2+ l3))
302     sum(l0 + l1 + l2+ l3)
303   }
304
305   escore<-function(par)
306   {
307     alpha0<-par[1]
308     alpha1<-par[2]
309     gama<-par[3]
310     phi<-par[4]
311
312     c<- 1- alpha0*(1-gama)-alpha1*gama
313     mu<- (gama*(1-alpha1))/c
314
315     a= mu * phi
316     b = a * (1 - mu)/mu
317
318     ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
319     mustar <- digamma(a) - digamma(b)
320
321     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
322     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
323     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
          mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
324     Ualpha0 = sum(Ualpha01 + Ualpha03)
325
326     Ualpha0 = ifelse(n0==0,0,Ualpha0)
327
328     Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
329     #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
330     Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
          gama*(1-gama)*(1-alpha0))/ (c^2) ) )
331     #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
          digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
332     Ualpha1 = sum(Ualpha11 - Ualpha13)
333
334     Ualpha1 = ifelse(n1==0,0,Ualpha1)
335
336     Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
337     Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
338     Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
          mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
339
340     Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
```

```
341
342   Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
343   #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
          digamma(b))
344   mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
345   )
346   Uphi = sum(Uphi0)
347
348
349   c(Ualpha0,Ualpha1,Ugama,Uphi)
350 }
351
352 # metodo dos momentos para iniciarlizacao
353 x_bar<-mean(x01)
354 sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
355
356 p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
357 q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
358
359 #print(c(x_bar,sigma2_hat,p,q))
360
361 delta0<- n0/n
362 delta1<- n1/n
363
364 a0<- delta0/(1-x_bar)
365 a1<- delta1/(x_bar)
366
367 #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
368 par_ini<-c(a0,a1,p/(p+q),p+q)
369
370 #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
371
372 #print("par_ini")
373 #print(par_ini)
374
375 max<-optim(par_ini,loglik,
376 escore,
377 method="BFGS",
378 #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
379 control=list(fnscale=-1))
380
381 if(max$convergence != 0) print("Non-convergence")
382 # return
383 z<-c()
384 z$mv<-max$par
385 #print(z$mv)
```

```
  z$conv<-max$convergence
  z$loglik <- max$value
  return(z)
}

iuGAfit <- function(y)

{

  n <- length(y) # sample size
  n0 <- sum(y == 0) # number of zeros
  n1 <- sum(y == 1) # number of ones
  m <- n0 + n1

  ys01 <- y
  if(any(y==0)) ys01<- ys01[-which(y==0)]
  if(any(y==1)) ys01<- ys01[-which(ys01==1)]

  i01 <- (y==0)+(y==1)

  #Maxima verossimilhanca

  loglik <- function (par){

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

    alpha0 = ifelse(n0 == 0, 0, alpha0)
    alpha1 = ifelse(n1 == 0, 0, alpha1)

    c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
    mu <- gama * (1 - alpha1) / c
    d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))

    l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
    l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
    l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
    l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))

    #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))

    sum(l0 + l1 + l2 + l3)

  }
  escore <- function (par) {
```

```
433
434     alpha0 <- par[1]
435     alpha1 <- par[2]
436     gama <- par[3]
437     phi <- par[4]
438
439     alpha0 <- ifelse(n0 == 0, 0, alpha0)
440     alpha1 <- ifelse(n1 == 0, 0, alpha1)
441
442     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
443     mu <- gama * (1 - alpha1) / c
444     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
445     ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
446     mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
447
448     Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
449     Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
        / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
        ^ 2))
450     Ualpha0 <- sum(Ualpha01 + Ualpha03)
451
452     Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
453
454     Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
455     Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
        (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
        2))
456     Ualpha1 <- sum(Ualpha11 + Ualpha13)
457
458     Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
459
460     Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
461     Ugama02 <- ifelse((y == 1), 1 / gama, 0)
462     Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
        + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
         ^ 2))
463
464     Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
465
466     Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
        /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
        phi) - log(mu^(1/phi)))
467
468     Uphi = sum(Uphi0)
469
470
471     c(Ualpha0, Ualpha1, Ugama, Uphi)
```

```
472   }
473
474   #Inicializacao
475
476   y_bar <- mean(y)
477   y_bar0 <- mean(ys01)
478
479   delta0 <- n0/n
480   delta1 <- n1/n
481
482   a0 <- delta0/(1-y_bar)
483   a1 <- delta1/(y_bar)
484
485   #fit <- vglm(ys01 ~ 1, gamma2)
486   #par<-Coef(fit)
487
488   #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
489   #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
490   #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
491   #phi_til = 5
492   phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
493
494   par_ini <- c(a0, a1, y_bar, phi_til)
495
496   max<-optim(par_ini, loglik, escore,method = "BFGS",
497   #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
498   #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
499   control=list(fnscale=-1))
500
501   if(max$convergence != 0) print("Non-convergence")
502   # return
503   z <- c()
504   z$ini <- par_ini
505   z$mv <- max$par
506   z$conv <- max$convergence
507   z$loglik <- max$value
508   return(z)
509 }
510
511 n=100
512 #n=60
513 #n=100
514 #n=200
515 NR=10000
516 k=4 # numero de parametros
```

```
517
518 Cont_AIC_IB=rep(0,NR)
519 Cont_AIC_IK=rep(0,NR)
520 Cont_AIC_IUG=rep(0,NR)
521
522 Cont_AICC_IB=rep(0,NR)
523 Cont_AICC_IK=rep(0,NR)
524 Cont_AICC_IUG=rep(0,NR)
525
526 Cont_BIC_IB=rep(0,NR)
527 Cont_BIC_IK=rep(0,NR)
528 Cont_BIC_IUG=rep(0,NR)
529
530 Cont_BICC_IB=rep(0,NR)
531 Cont_BICC_IK=rep(0,NR)
532 Cont_BICC_IUG=rep(0,NR)
533
534 Cont_HQ_IB=rep(0,NR)
535 Cont_HQ_IK=rep(0,NR)
536 Cont_HQ_IUG=rep(0,NR)
537
538 Cont_HQC_IB=rep(0,NR)
539 Cont_HQC_IK=rep(0,NR)
540 Cont_HQC_IUG=rep(0,NR)
541
542 Cont_LL_IB =rep(0,NR)
543 Cont_LL_IK=rep(0,NR)
544 Cont_LL_IUG=rep(0,NR)
545
546 contC=0
547 contC1=0
548
549 prop.z = rep(0,NR)
550 prop.u = rep(0,NR)
551
552 cvm.iBE = rep(0,NR)
553 ks.iBE = rep(0,NR)
554 ad.iBE = rep(0,NR)
555
556 cvm.iuGA = rep(0,NR)
557 ks.iuGA = rep(0,NR)
558 ad.iuGA = rep(0,NR)
559
560 cvm.ikum = rep(0,NR)
561 ks.ikum = rep(0,NR)
562 ad.ikum = rep(0,NR)
563
```

```
alpha0 = 0.1
alpha1 = 0.1
gama = 0.8
phi = 50

prob.0=alpha0*(1-gama) #(prob de zero)
prob.1=alpha1*gama #(prob de um)
prob.0
prob.1

lambda = prob.0+prob.1 #(prop.infl ou seja pro.0+prop.1)
lambda
p=(prob.1)/(lambda)
p
#p = (alpha0*(1-gama))/gama #(p=1 infl1 p=0 infla 0)
omega = gama
phi = phi

valores.parametros = c(alpha0, alpha1, gama, phi)

prob.z = alpha0*(1-gama)
prob.o = alpha1*gama

prob.z # probabilidade de zero
prob.o #probabilidade de um

media.v = gama
var.v = alpha1*gama*(1-alpha1*gama)+gama*(1-alpha1)*((2-((gama*(1-alpha1))/(1-
    alpha0*(1-gama)-alpha1*gama))^(1/phi))^(-phi)-gama*(1+alpha1))

media.v
var.v

#Bayes - BEINF

c= 1- alpha0*(1-gama)-alpha1*gama
mu = gama*(1-alpha1)/c
delta0 = alpha0 * (1-gama)
delta1 = alpha1 * (gama)
p2= 1- delta0-delta1

mu
delta0
delta1
p2
dp = sqrt(1/(phi+1))

```

```
cont.ib = 0
cont.ik = 0
cont.igu = 0

for(i in 1:NR) {

  y = riGU(n,alpha0 = alpha0, alpha1=alpha1, gama = gama, phi = phi)
  len = length(y)
  prop.z[i] = sum(ifelse(y==0,1,0))/len
  prop.u[i] = sum(ifelse(y==1,1,0))/len

  fitib = ibetafit(y)
  if (fitib$conv != 0)
  {
    cont.ib = cont.ib+1
  }

  fitik = ikumafit(y)
  if (fitik$conv != 0)
  {
    cont.ik = cont.ik+1
  }

  fitigu = iuGAfit(y)

  if ((fitigu$conv != 0) ){
    cont.igu=cont.igu+1
  }

  ks.iBE[i] = ifelse(ks.test(y, "pBEINF",mu=mu , sigma=dp, nu=delta0/p2, tau=
      delta1/p2)$p.value>0.05,1,0)

  ad.iBE[i] = ifelse(ad.test(y, null="pBEINF",mu=mu , sigma=dp, nu=delta0/p2, tau
      =delta1/p2)$p.value>0.05,1,0)

  cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF",mu=mu , sigma=dp, nu=delta0/p2,
      tau=delta1/p2)$p.value >0.05,1,0)


  ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
      phi=phi)$p.value>0.05,1,0)

  ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
      gama, phi=phi)$p.value>0.05,1,0)

  cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
      gama, phi=phi)$p.value>0.05,1,0)
```

```
651
652
653 ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=lambda, p=p,omega=gama, phi=phi)$
        p.value>0.05,1,0)
654
655 ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=lambda, p=p,omega=gama, phi=
        phi)$p.value>0.05,1,0)
656
657 cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=lambda, p=p,omega=gama,
        phi=phi)$p.value>0.05,1,0)
658
659 #Log verossimilhanca
660 # Beta inflacionada
661 log.vero_IB = fitib$loglik
662 # Kuma inflacionada
663 log.vero_IK= fitik$loglik
664 # Gamma Unitaria inflacionada
665 log.vero_IUG = fitigu$loglik
666
667 m.log.vero_IB = -fitib$loglik
668 # Kuma inflacionada
669 m.log.vero_IK= -fitik$loglik
670 # Gamma Unitaria inflacionada
671 m.log.vero_IUG = -fitigu$loglik
672
673 #Menor -log verossimilhanca
674
675 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
676
677 Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
678 Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
679 Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
680
681 # Calculando os criterios/quantidades para comparacao
682
683 #criteiro_AIC#
684 AIC_IB = -2*log.vero_IB+2*k #BI(Beta inflacionada)
685 AIC_IK= -2*log.vero_IK+2*k #KI(Kuma inflacionada)
686 AIC_IUG = -2*log.vero_IUG+2*k #GUI(Gamma Unitaria inflacionada)
687
688 # Selecionando o menor AIC
689 min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
690
691 Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0)
692 Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
693 Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
694
```

```
695 AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1) #BI
696 AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1) #KI
697 AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1) #GUI
698
699 # Selecionando o menor AICC
700 min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
701
702 Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0)
703 Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
704 Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)
705
706 #criterio_BIC#
707 BIC_IB=-2*log.vero_IB+k*log(n) #BI
708 BIC_IK=-2*log.vero_IK+k*log(n) #KI
709 BIC_IUG=-2*fitigu$loglik+k*log(n) #GUI
710
711 # Selecionando o menor BIC
712 min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))
713
714 Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
715 Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
716 Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)
717
718 #criterio_BICC#
719 BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1) #BI
720 BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1) #KI
721 BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1) #GUI
722
723 # Selecionando o menor BICC
724 min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))
725
726 Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
727 Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
728 Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)
729
730 #criterio_HQ#
731 HQ_IB=-2*log.vero_IB+2*log(log(n)) #BI
732 HQ_IK=-2*log.vero_IK+2*log(log(n)) #KI
733 HQ_IUG=-2*fitigu$loglik+2*log(log(n)) #GUI
734
735 # Selecionando o menor HQ
736 min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))
737
738 Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
739 Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
740 Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)
741
```

```
  #criterio_HQC#
  HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1) #BI
  HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1) #KI
  HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1) #GUI

  # Selecionando o menor HQC
  min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))

  Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
  Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
  Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
  # FIM DO LACO DE MONTE CARLO
}

# Proporcao media de zeros
#mean(prop.z)
# Proporcao media de uns
#mean(prop.u)

#Calculando percentuais escolha distribuicao(AIC)
p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
p_AICIB
p_AICIK
p_AICIUG

#Calculando percentuais escolha distribuicao(AICC)
p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
p_AICCIB
p_AICCIK
p_AICCIUG

#Calculando percentuais escolha distribuicao(BIC)
p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
p_BICIB
p_BICIK
p_BICIUG

#Calculando percentuais escolha distribuicao(BICC)
p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
```

```
p_BICCIB
p_BICCIK
p_BICCIUG

#Calculando percentuais escolha distribuicao(HQ)
p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
p_HQIB
p_HQIK
p_HQIUG

#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG

#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
p_LLIB
p_LLIK
p_LLIUG

result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC

cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100
ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100
ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100
cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100
ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100
```

```
ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100
cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100
ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100
ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)


M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
result.taxa.rej.ad,result.taxa.rej.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
    ks",
"Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")

glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
    .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
"\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
    phi: {phi}","\n", "media: {media.v}",
"\n", "var: {round(var.v,4)}")
print("Resultados")
print(M)

M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
    )
colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
    Rej.cvm")
xtable(M.res)

#Salvando em um arquivo
#write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

```
  ##Simulacao gerando amostras a partir da gama unitaria inflacionada em um##
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
```

```r
library(goftest)
source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")
set.seed(33) #Fixando a semente
source("ikum-omega-phi.R")
# library(optimx)
library(rootSolve)
library(VGAM)
library(xtable)
# library(fitdistrplus)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
  {
    p=0
    u=0
  }

  i01 <- (y==0)+(y==1)

  loglik <- function(theta)
  {
    omega <- theta[1]
    phi <- theta[2]

    ########
    l2 = 0
    if(u!=0)
```

```
  {
    l2a = ifelse((y == 1), log(p), 0)
    l2b = ifelse((y == 0), log(1-p), 0)
    l2 = l2a + l2b
  }

  l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

  l3 = ifelse((i01 == 1), 0,
  log(phi)+
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}

escore <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  delta = log(0.5)/log(1-omega^phi)
  ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
      +1)

  c = ifelse((y == 0), 0, ci)
  c = ifelse((y == 1), 0, c)

  Uomega <- phi*sum(c)

  v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
  (delta-1)*((y^phi)*log(y)/(1-y^phi)))
  v = ifelse((y == 1), 0, v)

  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

```

```
# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

#############

opt <- optim(ini, loglik, escore, # hessian = T,
method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))

# opt <- optim(ini, loglik, escore,
# method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e-9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double.eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
# upper = rep(.Machine$double.xmax,(r+2))
# )

# opt <- optim(ini, loglik, escore)

# print(opt$par)

# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)


if (opt$conv != 0)
warning("FUNCTION DID NOT CONVERGE!")

k <- c()

# print("AQUI1")
# print(gradient(escore, opt$par))

coef <- c(lambda,p,opt$par)
# coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
# print(coef)
k$coef <- coef
k$conv <- opt$conv
```

```
145  k$loglik <- opt$value
146  k$counts <- as.numeric(opt$counts[1])
147
148  # library(rootSolve)
149  # library(numDeriv)
150  # escores<-rbind(
151  # escore(coef),
152  # gradient(loglik, coef),
153  # grad(loglik, coef))
154  # print(round(escores,5))
155
156  # k$Knum<-gradient(escore, coef)
157
158  # print(k$Knum)
159  # k$Knum<-hessian(escore, coef, method="complex")
160  # print(k$Knum)
161
162
163  # # Expected information matrix
164  #
165  # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
166  # T3 = diag(as.vector( 1/diflink3(k$omega) ))
167  # T4 = diag(as.vector( 1/diflink4(k$phi) ))
168  #
169  # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
170  # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
171  # delta <- log(0.5)/log(1-k$omega^k$phi)
172  #
173  # kapa<- 0.5772156649
174  # k0<- (pi^2)/6+kapa^2-2*kapa
175  #
176  # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
         log(1-y^k$phi)+1)
177  #
178  # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
179  # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
180  # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
181  # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
182  #
183  # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
184  #
185  # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
186  #
187  # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
188  # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
         phi)) -
189  # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
```

```
    k$phi^3))
# #
# # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
#
# mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
    +1)-kapa)/((delta-1)*k$phi)) )
#
# L = diag(-1/(k$lambda*(1-k$lambda)))
# V = diag((k$lambda-1)*k$phi^2*h2)
# M = diag(mi)
# S = diag(si)
#
#
# Igg = -t(z1) %*% L %*% (T1^2) %*% z1
# Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
# Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
# Isb = t(Ibs)
# Iss = -t(q1) %*% S %*% (T4^2) %*% q1
#
# if(u!=0)
# {
  # k$pi <- as.vector(coef[(m+1):(m+u)])
  # eta_hat2 <- as.vector(w1%*%k$pi)
  # k$p <- as.vector(linkinv2(eta_hat2))
  # T2 = diag(as.vector( 1/diflink2(k$p) ))
  # P = diag(-k$lambda/(k$p*(1-k$p)))
  # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
  #
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
      ncol=s)),
  # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
      ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
  # )
  #
  # }else{
  # I <- rbind(
  # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
  # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
  # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
  # )
  # }
#
# k$I <- I
#
```

```
  #
  # ############################
  #
  # # print(-k$Knum)
  # # print(k$I)
  #
  # # vcov <- try(solve(-k$Knum))
  # vcov <- try(solve(k$I))
  # # vcov <- chol2inv(chol(k$I))
  # #vcov <- K #somente para nao trancar simulacao
  #
  # k$cov_ok = 0
  # if (class(vcov) == "try-error")
  # {
    # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
        with inversion
    # k$cov_ok = 1
    #
    # if (class(vcov) == "try-error") #
    # {
      # vcov <- k$Knum^2
      # warning("Problem with the variance and covariance matrix. The p-values
          are not correct.")
      # k$cov_ok = 2
      # }
    # }


  # k$vcov <- vcov
  # stderror <- sqrt(diag(k$vcov))
  # k$stderror <- stderror

  return(k)
}

ibetafit<-function(x)
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

  xm0<-x[which(x>0)]
  x01<-xm0[which(xm0<1)]

  #MV
```

```
278  loglik<-function(par)
279  {
280    alpha0<-par[1]
281    alpha1<-par[2]
282    gama<-par[3]
283    phi<-par[4]
284
285    alpha0 = ifelse(n0==0,0,alpha0)
286    alpha1 = ifelse(n1==0,0,alpha1)
287
288    #print(c(alpha0,alpha1,gama,phi))
289
290    c<- 1- alpha0*(1-gama)-alpha1*gama
291    mu<- gama*(1-alpha1)/c
292
293
294    l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
295    l1 = ifelse( (x == 1), log(alpha1*gama), 0)
296    l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
297    l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
         - mu) * phi, log = TRUE)))
298
299
300    #print("soma")
301    #print(sum(l0 + l1 + l2+ l3))
302    sum(l0 + l1 + l2+ l3)
303  }
304
305  escore<-function(par)
306  {
307    alpha0<-par[1]
308    alpha1<-par[2]
309    gama<-par[3]
310    phi<-par[4]
311
312    c<- 1- alpha0*(1-gama)-alpha1*gama
313    mu<- (gama*(1-alpha1))/c
314
315    a= mu * phi
316    b = a * (1 - mu)/mu
317
318    ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
319    mustar <- digamma(a) - digamma(b)
320
321
322
323    Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
```

```
  #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
  Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
      mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
  Ualpha0 = sum(Ualpha01 + Ualpha03)

  Ualpha0 = ifelse(n0==0,0,Ualpha0)


  Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
  #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
  Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
      gama*(1-gama)*(1-alpha0))/ (c^2) ) )
  #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
      digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
  Ualpha1 = sum(Ualpha11 - Ualpha13)

  Ualpha1 = ifelse(n1==0,0,Ualpha1)

  Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
  Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
  Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
      mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))

  Ugama = sum(-Ugama01 + Ugama02 + Ugama04)

  Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
  #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
      digamma(b))
  mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
  )
  Uphi = sum(Uphi0)


  c(Ualpha0,Ualpha1,Ugama,Uphi)
}

# metodo dos momentos para iniciarlizacao
x_bar<-mean(x01)
sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)

p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)

#print(c(x_bar,sigma2_hat,p,q))

delta0<- n0/n
delta1<- n1/n
```

```
366
367   a0<- delta0/(1-x_bar)
368   a1<- delta1/(x_bar)
369
370   #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
371   par_ini<-c(a0,a1,p/(p+q),p+q)
372
373   #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
374
375   #print("par_ini")
376   #print(par_ini)
377
378   max<-optim(par_ini,loglik,
379   escore,
380   method="BFGS",
381   #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
382   control=list(fnscale=-1))
383
384   if(max$convergence != 0) print("Non-convergence")
385   # return
386   z<-c()
387   z$mv<-max$par
388   #print(z$mv)
389   z$conv<-max$convergence
390   z$loglik <- max$value
391   return(z)
392 }
393
394
395 iuGAfit <- function(y)
396
397 {
398
399   n <- length(y) # sample size
400   n0 <- sum(y == 0) # number of zeros
401   n1 <- sum(y == 1) # number of ones
402   m <- n0 + n1
403
404   ys01 <- y
405   if(any(y==0)) ys01<- ys01[-which(y==0)]
406   if(any(y==1)) ys01<- ys01[-which(ys01==1)]
407
408   i01 <- (y==0)+(y==1)
409
410   #Maxima verossimilhanca
411   loglik <- function (par){
```

```
412
413     alpha0 <- par[1]
414     alpha1 <- par[2]
415     gama <- par[3]
416     phi <- par[4]
417
418     alpha0 = ifelse(n0 == 0, 0, alpha0)
419     alpha1 = ifelse(n1 == 0, 0, alpha1)
420
421     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
422     mu <- gama * (1 - alpha1) / c
423     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
424
425     l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
426     l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
427     l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
428     l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))
429
430     #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))
431
432     sum(l0 + l1 + l2 + l3)
433
434   }
435   escore <- function (par) {
436
437     alpha0 <- par[1]
438     alpha1 <- par[2]
439     gama <- par[3]
440     phi <- par[4]
441
442     alpha0 <- ifelse(n0 == 0, 0, alpha0)
443     alpha1 <- ifelse(n1 == 0, 0, alpha1)
444
445     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
446     mu <- gama * (1 - alpha1) / c
447     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
448     ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
449     mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
450
451
452     Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
453     Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
        / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
        ^ 2))
454     Ualpha0 <- sum(Ualpha01 + Ualpha03)
455
456     Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
```

```
457
458   Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
459   Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
          (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
          2))
460   Ualpha1 <- sum(Ualpha11 + Ualpha13)
461
462   Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
463
464   Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
465   Ugama02 <- ifelse((y == 1), 1 / gama, 0)
466   Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
          + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
           ^ 2))
467
468   Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
469
470   Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
          /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
          phi) - log(mu^(1/phi)))
471
472   Uphi = sum(Uphi0)
473
474   c(Ualpha0, Ualpha1, Ugama, Uphi)
475 }
476
477 #Inicializacao
478
479 y_bar <- mean(y)
480 y_bar0 <- mean(ys01)
481
482 delta0 <- n0/n
483 delta1 <- n1/n
484
485 a0 <- delta0/(1-y_bar)
486 a1 <- delta1/(y_bar)
487
488 #fit <- vglm(ys01 ~ 1, gamma2)
489 #par<-Coef(fit)
490
491 #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
492 #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
493 #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
494 #phi_til = 5
495 phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
496
497 par_ini <- c(a0, a1, y_bar, phi_til)
```

```
498
499   max<-optim(par_ini, loglik, escore,method = "BFGS",
500   #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
501   #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
502   control=list(fnscale=-1))
503
504   if(max$convergence != 0) print("Non-convergence")
505   # return
506   z <- c()
507   z$ini <- par_ini
508   z$mv <- max$par
509   z$conv <- max$convergence
510   z$loglik <- max$value
511   return(z)
512 }
513
514 n=100
515 #n=60
516 #n=100
517 #n=200
518 NR=10000
519 k=3 # numero de parametros
520
521 Cont_AIC_IB=rep(0,NR)
522 Cont_AIC_IK=rep(0,NR)
523 Cont_AIC_IUG=rep(0,NR)
524
525 Cont_AICC_IB=rep(0,NR)
526 Cont_AICC_IK=rep(0,NR)
527 Cont_AICC_IUG=rep(0,NR)
528
529 Cont_BIC_IB=rep(0,NR)
530 Cont_BIC_IK=rep(0,NR)
531 Cont_BIC_IUG=rep(0,NR)
532
533 Cont_BICC_IB=rep(0,NR)
534 Cont_BICC_IK=rep(0,NR)
535 Cont_BICC_IUG=rep(0,NR)
536
537 Cont_HQ_IB=rep(0,NR)
538 Cont_HQ_IK=rep(0,NR)
539 Cont_HQ_IUG=rep(0,NR)
540
541 Cont_HQC_IB=rep(0,NR)
542 Cont_HQC_IK=rep(0,NR)
```

```
Cont_HQC_IUG=rep(0,NR)

Cont_LL_IB =rep(0,NR)
Cont_LL_IK=rep(0,NR)
Cont_LL_IUG=rep(0,NR)

contC=0
contC1=0

prop.z = rep(0,NR)
prop.u = rep(0,NR)

cvm.iBE = rep(0,NR)
ks.iBE = rep(0,NR)
ad.iBE = rep(0,NR)

cvm.iuGA = rep(0,NR)
ks.iuGA = rep(0,NR)
ad.iuGA = rep(0,NR)

cvm.ikum = rep(0,NR)
ks.ikum = rep(0,NR)
ad.ikum = rep(0,NR)

alpha0 = 0
alpha1 = 0.0875
gama = 0.8
phi = 50

valores.parametros = c(alpha0, alpha1, gama, phi)

prob.z = alpha0*(1-gama)
prob.o = alpha1*gama

prob.z # probabilidade de zero
prob.o #probabilidade de um

media.v = gama
var.v = alpha1*gama*(1-alpha1*gama)+gama*(1-alpha1)*((2-((gama*(1-alpha1))/(1-
    alpha0*(1-gama)-alpha1*gama))^(1/phi))^(-phi)-gama*(1+alpha1))

media.v
var.v

#Bayes - BEINF

c= 1- alpha0*(1-gama)-alpha1*gama
```

```
mu = gama*(1-alpha1)/c
delta0 = alpha0 * (1-gama)
delta1 = alpha1 * (gama)
p2= 1- delta0-delta1

mu
delta0
delta1
p2
dp = sqrt(1/(phi+1))

cont.ib = 0
cont.ik = 0
cont.igu = 0

for(i in 1:NR) {


  y = riGU(n,alpha0 = alpha0, alpha1=alpha1, gama = gama, phi = phi)
  len = length(y)
  prop.z[i] = sum(ifelse(y==0,1,0))/len
  prop.u[i] = sum(ifelse(y==1,1,0))/len


  fitib = ibetafit(y)
  if (fitib$conv != 0)
  {
    cont.ib = cont.ib+1
  }

  fitik = ikumafit(y)
  if (fitik$conv != 0)
  {
    cont.ik = cont.ik+1
  }

  fitigu = iuGAfit(y)


  if ((fitigu$conv != 0) ){
    cont.igu=cont.igu+1
  }

  ks.iBE[i] = ifelse(ks.test(y, "pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$p.value
      >0.05,1,0)
  ad.iBE[i] = ifelse(ad.test(y, null="pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$p.
      value>0.05,1,0)
```

```
634 cvm.iBE[i] = ifelse(cvm.test(y, null="pBEINF1",mu=mu , sigma=dp, nu=delta1/p2)$
        p.value >0.05,1,0)
635 ks.iuGA[i] = ifelse(ks.test(y, "pgui",alpha0=alpha0,alpha1=alpha1,gama=gama,
        phi=phi)$p.value>0.05,1,0)
636 ad.iuGA[i] = ifelse(ad.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
637 cvm.iuGA[i] = ifelse(cvm.test(y, null="pgui",alpha0=alpha0,alpha1=alpha1,gama=
        gama, phi=phi)$p.value>0.05,1,0)
638 ks.ikum[i] = ifelse(ks.test(y, "pikum",lambda=alpha1, p=1,omega=gama, phi=phi)$
        p.value>0.05,1,0)
639 ad.ikum[i] = ifelse(ad.test(y, null="pikum",lambda=alpha1, p=1,omega=gama, phi=
        phi)$p.value>0.05,1,0)
640 cvm.ikum[i] = ifelse(cvm.test(y, null="pikum",lambda=alpha1, p=1,omega=gama,
        phi=phi)$p.value>0.05,1,0)
641
642 #Log verossimilhanca
643 # Beta inflacionada
644 log.vero_IB = fitib$loglik
645 # Kuma inflacionada
646 log.vero_IK= fitik$loglik
647 # Gamma Unitaria inflacionada
648 log.vero_IUG = fitigu$loglik
649
650 m.log.vero_IB = -fitib$loglik
651 # Kuma inflacionada
652 m.log.vero_IK= -fitik$loglik
653 # Gamma Unitaria inflacionada
654 m.log.vero_IUG = -fitigu$loglik
655
656 #Menor -log verossimilhanca
657 min_LL = which.min(c(m.log.vero_IB,m.log.vero_IK,m.log.vero_IUG))
658
659 Cont_LL_IB[i] = ifelse(min_LL==1,1,0)
660 Cont_LL_IK[i] = ifelse(min_LL==2,1,0)
661 Cont_LL_IUG[i] = ifelse(min_LL==3,1,0)
662
663 # Calculando os criterios/quantidades para comparacao
664
665 #criteiro_AIC#
666 AIC_IB = -2*log.vero_IB+2*k #BI(Beta Inflacionada)
667 AIC_IK= -2*log.vero_IK+2*k #KI(Kuma inflacionada)
668 AIC_IUG = -2*log.vero_IUG+2*k #GUI(Gamma Unitaria inflacionada)
669
670 # Selecionando o menor AIC
671 min_AIC = which.min(c(AIC_IB,AIC_IK,AIC_IUG))
672
673 Cont_AIC_IB[i] = ifelse(min_AIC==1,1,0) #BI
```

```
674  Cont_AIC_IK[i] = ifelse(min_AIC==2,1,0)
675  Cont_AIC_IUG[i] = ifelse(min_AIC==3,1,0)
676
677  AICC_IB= -2*log.vero_IB+(2*n*k)/(n-k-1) #BI
678  AICC_IK= -2*log.vero_IK+(2*n*k)/(n-k-1) #KI
679  AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1) #GUI
680
681  # Selecionando o menor AICC
682  min_AICC= which.min(c(AICC_IB,AICC_IK,AICC_IUG))
683
684  Cont_AICC_IB[i]=ifelse(min_AICC==1,1,0) #BI
685  Cont_AICC_IK[i]=ifelse(min_AICC==2,1,0)
686  Cont_AICC_IUG[i]=ifelse(min_AICC==3,1,0)
687
688  #criterio_BIC#
689  BIC_IB=-2*log.vero_IB+k*log(n) #BI
690  BIC_IK=-2*log.vero_IK+k*log(n) #KI
691  BIC_IUG=-2*fitigu$loglik+k*log(n) #GUI
692
693  # Selecionando o menor BIC
694  min_BIC = which.min(c(BIC_IB,BIC_IK,BIC_IUG))
695
696  Cont_BIC_IB[i]=ifelse(min_BIC==1,1,0)
697  Cont_BIC_IK[i]=ifelse(min_BIC==2,1,0)
698  Cont_BIC_IUG[i]=ifelse(min_BIC==3,1,0)
699
700  #criterio_BICC#
701  # Beta inflacionada
702  BICC_IB=-2*log.vero_IB+(k*log(n))/(n-k-1) #BI
703  BICC_IK=-2*log.vero_IK+(k*log(n))/(n-k-1) #KI
704  BICC_IUG=-2*fitigu$loglik+(k*log(n))/(n-k-1) #GUI
705
706  # Selecionando o menor BICC
707  min_BICC = which.min(c(BICC_IB,BICC_IK,BICC_IUG))
708
709  Cont_BICC_IB[i] = ifelse(min_BICC==1,1,0)
710  Cont_BICC_IK[i] = ifelse(min_BICC==2,1,0)
711  Cont_BICC_IUG[i] = ifelse(min_BICC==3,1,0)
712
713  #criterio_HQ#
714  HQ_IB=-2*log.vero_IB+2*log(log(n)) #BI
715  HQ_IK=-2*log.vero_IK+2*log(log(n)) #KI
716  HQ_IUG=-2*fitigu$loglik+2*log(log(n)) #GUI
717
718  # Selecionando o menor HQ
719  min_HQ = which.min(c(HQ_IB,HQ_IK,HQ_IUG))
720
```

```
  Cont_HQ_IB[i] = ifelse(min_HQ==1,1,0)
  Cont_HQ_IK[i] = ifelse(min_HQ==2,1,0)
  Cont_HQ_IUG[i] = ifelse(min_HQ==3,1,0)

  #criterio_HQC#
  HQC_IB=-2*log.vero_IB+(2*n*k*log(log(n)))/(n-k-1) #BI
  HQC_IK=-2*log.vero_IK+(2*n*k*log(log(n)))/(n-k-1) #KI
  HQC_IUG=-2*fitigu$loglik+(2*n*k*log(log(n)))/(n-k-1) #GUI

  # Selecionando o menor HQC
  min_HQC = which.min(c(HQC_IB,HQC_IK,HQC_IUG))

  Cont_HQC_IB[i] = ifelse(min_HQC==1,1,0)
  Cont_HQC_IK[i] = ifelse(min_HQC==2,1,0)
  Cont_HQC_IUG[i] = ifelse(min_HQC==3,1,0)
  # FIM DO LACO DE MONTE CARLO
}

# Proporcao media de zeros
#mean(prop.z)
# Proporcao media de uns
#mean(prop.u)

#Calculando percentuais escolha distribuicao(AIC)
p_AICIB<-((sum(Cont_AIC_IB))/NR)*100
p_AICIK<-((sum(Cont_AIC_IK))/NR)*100
p_AICIUG<-((sum(Cont_AIC_IUG))/NR)*100
p_AICIB
p_AICIK
p_AICIUG

#Calculando percentuais escolha distribuicao(AICC)
p_AICCIB<-((sum(Cont_AICC_IB))/NR)*100
p_AICCIK<-((sum(Cont_AICC_IK))/NR)*100
p_AICCIUG<-((sum(Cont_AICC_IUG))/NR)*100
p_AICCIB
p_AICCIK
p_AICCIUG

#Calculando percentuais escolha distribuicao(BIC)
p_BICIB<-((sum(Cont_BIC_IB))/NR)*100
p_BICIK<-((sum(Cont_BIC_IK))/NR)*100
p_BICIUG<-((sum(Cont_BIC_IUG))/NR)*100
p_BICIB
p_BICIK
p_BICIUG

```

```
#Calculando percentuais escolha distribuicao(BICC)
p_BICCIB<-((sum(Cont_BICC_IB))/NR)*100
p_BICCIK<-((sum(Cont_BICC_IK))/NR)*100
p_BICCIUG<-((sum(Cont_BICC_IUG))/NR)*100
p_BICCIB
p_BICCIK
p_BICCIUG

#Calculando percentuais escolha distribuicao(HQ)
p_HQIB<-((sum(Cont_HQ_IB))/NR)*100
p_HQIK<-((sum(Cont_HQ_IK))/NR)*100
p_HQIUG<-((sum(Cont_HQ_IUG))/NR)*100
p_HQIB
p_HQIK
p_HQIUG

#Calculando percentuais escolha distribuicao(HQC)
p_HQCIB<-((sum(Cont_HQC_IB))/NR)*100
p_HQCIK<-((sum(Cont_HQC_IK))/NR)*100
p_HQCIUG<-((sum(Cont_HQC_IUG))/NR)*100
p_HQCIB
p_HQCIK
p_HQCIUG

#Calculando percentuais escolha distribuicao(-LL)
p_LLIB<-((sum(Cont_LL_IB))/NR)*100
p_LLIK<-((sum(Cont_LL_IK))/NR)*100
p_LLIUG<-((sum(Cont_LL_IUG))/NR)*100
p_LLIB
p_LLIK
p_LLIUG

#resultados -log.vero
result_LL = cbind(p_LLIB,p_LLIK,p_LLIUG)
result_AIC=cbind(p_AICIB,p_AICIK,p_AICIUG)
result_AIC
result_AICC=cbind(p_AICCIB,p_AICCIK,p_AICCIUG)
result_AICC
result_BIC=cbind(p_BICIB,p_BICIK,p_BICIUG)
result_BIC
result_BICC=cbind(p_BICCIB,p_BICCIK,p_BICCIUG)
result_BICC
result_HQ=cbind(p_HQIB,p_HQIK,p_HQIUG)
result_HQ
result_HQC=cbind(p_HQCIB,p_HQCIK,p_HQCIUG)
result_HQC
cvm.Taxa.Nao.rej.iBE = (sum(cvm.iBE)/NR)*100
```

```
ks.Taxa.Nao.rej.iBE = (sum(ks.iBE)/NR)*100
ad.Taxa.Nao.rej.iBE = (sum(ad.iBE)/NR)*100
cvm.Taxa.Nao.rej.iuGA = (sum(cvm.iuGA)/NR)*100
ks.Taxa.Nao.rej.iuGA = (sum(ks.iuGA)/NR)*100
ad.Taxa.Nao.rej.iuGA = (sum(ad.iuGA)/NR)*100
cvm.Taxa.Nao.rej.ikum = (sum(cvm.ikum)/NR)*100
ks.Taxa.Nao.rej.ikum = (sum(ks.ikum)/NR)*100
ad.Taxa.Nao.rej.ikum = (sum(ad.ikum)/NR)*100

result.taxa.rej.ks = cbind(ks.Taxa.Nao.rej.iBE,ks.Taxa.Nao.rej.ikum,ks.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.ad = cbind(ad.Taxa.Nao.rej.iBE,ad.Taxa.Nao.rej.ikum,ad.Taxa.Nao.
    rej.iuGA)

result.taxa.rej.cvm = cbind(cvm.Taxa.Nao.rej.iBE,cvm.Taxa.Nao.rej.ikum,cvm.Taxa.
    Nao.rej.iuGA)

M=rbind(result_LL, result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,
    result_HQC,result.taxa.rej.ks,
result.taxa.rej.ad,result.taxa.rej.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("-LogVero","AIC","AICC","BIC", "BICC","HQ","HQC", "Taxa.Nao.Rej.
    ks",
"Taxa.Nao.Rej.ad","Taxa.Nao.Rej.cvm")

glue::glue("Numero de replicas: {NR}", "\n", "Tamanho amostral: {n}", "\n", "Prob
    .Zero: {prob.z}", "\n", "Prob.Um: {prob.o}",
"\n", "alpha.0: {alpha0}","\n", "alpha.1: {alpha1}", "\n", "gama: {gama}","\n", "
    phi: {phi}","\n", "media: {media.v}",
"\n", "var: {round(var.v,4)}")
print("Resultados")
print(M)

M.res=rbind( result_AIC,result.taxa.rej.ks,result.taxa.rej.ad,result.taxa.rej.cvm
    )
colnames(M.res) <- c("BIm.I","Kuma.I","GU.I")
rownames(M.res) <- c("criterios", "Taxa.Nao.Rej.ks","Taxa.Nao.Rej.ad","Taxa.Nao.
    Rej.cvm")
xtable(M.res)

#Salvando em um arquivo
#write.table(result_AIC, file = "Resultados_Escolha_Dist.txt", sep = " ")
```

Apêndice C

```
                    ##APLICACAO COM INFLACIONAMENTO EM ZERO##

#Proporcao de pessoas que usa internet em 2000 nos paises do mundo
#Fonte dos dados: https://ourworldindata.org/
library(gamlss.dist)
library(goftest) #ad e cvm
source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")
source("ikum-omega-phi.R")
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
  if(n0 == 0)
  {
    p=1
    u=0
  }

  if(n1 == 0)
```

```
{
  p=0
  u=0
}

i01 <- (y==0)+(y==1)

loglik <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  ########
  l2 = 0
  if(u!=0)
  {
    l2a = ifelse((y == 1), log(p), 0)
    l2b = ifelse((y == 0), log(1-p), 0)
    l2 = l2a + l2b
  }

  l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

  l3 = ifelse((i01 == 1), 0,
  log(phi)+
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}

escore <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  delta = log(0.5)/log(1-omega^phi)
  ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
      +1)

  c = ifelse((y == 0), 0, ci)
  c = ifelse((y == 1), 0, c)

  Uomega <- phi*sum(c)

```

```
  v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
  (delta-1)*((y^phi)*log(y)/(1-y^phi)))
  v = ifelse((y == 1), 0, v)

  Uphi <- sum(v)

  derivada=c(Uomega,Uphi)
  derivada
}

### initial values
ys01<- y
if(any(y==0)) ys01<- ys01[-which(y==0)]
if(any(y==1)) ys01<- ys01[-which(ys01==1)]

# fit <- vglm(ys01 ~ 1, kumar)
# par<-Coef(fit)

phi <- 5 #par[1]
omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)

#print(c(lambda,p,omega,phi))
ini <- c(omega,phi)

##############

opt <- optim(ini, loglik, escore, # hessian = T,
method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))

# opt <- optim(ini, loglik, escore,
# method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e
    -9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double
    .eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
# upper = rep(.Machine$double.xmax,(r+2))
# )

# opt <- optim(ini, loglik, escore)

# print(opt$par)

# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)
```

```
128
129
130 if (opt$conv != 0)
131 warning("FUNCTION DID NOT CONVERGE!")
132
133 k <- c()
134
135 # print("AQUI1")
136 # print(gradient(escore, opt$par))
137
138 coef <- c(lambda,p,opt$par)
139 # coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
140 # print(coef)
141 k$coef <- coef
142 k$conv <- opt$conv
143 k$loglik <- opt$value
144 k$counts <- as.numeric(opt$counts[1])
145
146 # library(rootSolve)
147 # library(numDeriv)
148 # escores<-rbind(
149 # escore(coef),
150 # gradient(loglik, coef),
151 # grad(loglik, coef))
152 # print(round(escores,5))
153
154 # k$Knum<-gradient(escore, coef)
155
156 # print(k$Knum)
157 # k$Knum<-hessian(escore, coef, method="complex")
158 # print(k$Knum)
159
160
161 # # Expected information matrix
162 #
163 # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
164 # T3 = diag(as.vector( 1/diflink3(k$omega) ))
165 # T4 = diag(as.vector( 1/diflink4(k$phi) ))
166 #
167 # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
168 # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
169 # delta <- log(0.5)/log(1-k$omega^k$phi)
170 #
171 # kapa<- 0.5772156649
172 # k0<- (pi^2)/6+kapa^2-2*kapa
173 #
```

```
174  # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
         log(1-y^k$phi)+1)
175  #
176  # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
177  # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
178  # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
179  # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
180  #
181  # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
182  #
183  # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
184  #
185  # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
186  # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
         phi)) -
187  # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
         k$phi^3))
188  # #
189  # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
190  #
191  # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
         +1)-kapa)/((delta-1)*k$phi)) )
192  #
193  # L = diag(-1/(k$lambda*(1-k$lambda)))
194  # V = diag((k$lambda-1)*k$phi^2*h2)
195  # M = diag(mi)
196  # S = diag(si)
197  #
198  #
199  # Igg = -t(z1) %*% L %*% (T1^2) %*% z1
200  # Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
201  # Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
202  # Isb = t(Ibs)
203  # Iss = -t(q1) %*% S %*% (T4^2) %*% q1
204  #
205  # if(u!=0)
206  # {
207    # k$pi <- as.vector(coef[(m+1):(m+u)])
208    # eta_hat2 <- as.vector(w1%*%k$pi)
209    # k$p <- as.vector(linkinv2(eta_hat2))
210    # T2 = diag(as.vector( 1/diflink2(k$p) ))
211    # P = diag(-k$lambda/(k$p*(1-k$p)))
212    # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
213    #
214    # I <- rbind(
215    # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
           ncol=s)),
```

```
216     # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
            ncol=s)),
217     # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
218     # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
219     # )
220     #
221     # }else{
222     # I <- rbind(
223     # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
224     # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
225     # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
226     # )
227     # }
228   #
229   # k$I <- I
230   #
231   #
232   # ############################
233   #
234   # # print(-k$Knum)
235   # # print(k$I)
236   #
237   # # vcov <- try(solve(-k$Knum))
238   # vcov <- try(solve(k$I))
239   # # vcov <- chol2inv(chol(k$I))
240   # #vcov <- K #somente para nao trancar simulacao
241   #
242   # k$cov_ok = 0
243   # if (class(vcov) == "try-error")
244   # {
245     # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
            with inversion
246     # k$cov_ok = 1
247     #
248     # if (class(vcov) == "try-error") #
249     # {
250       # vcov <- k$Knum^2
251       # warning("Problem with the variance and covariance matrix. The p-values
              are not correct.")
252       # k$cov_ok = 2
253       # }
254     # }
255
256
257   # k$vcov <- vcov
258   # stderror <- sqrt(diag(k$vcov))
259   # k$stderror <- stderror
```

```
260
261   return(k)
262 }
263
264 ibetafit<-function(x)
265 {
266
267   n<- length(x) # sample size
268   n0<- sum(x==0) # number of zeros
269   n1<- sum(x==1) # number of ones
270   m<- n0 + n1
271
272   xm0<-x[which(x>0)]
273   x01<-xm0[which(xm0<1)]
274
275   #MV
276   loglik<-function(par)
277   {
278     alpha0<-par[1]
279     alpha1<-par[2]
280     gama<-par[3]
281     phi<-par[4]
282
283     alpha0 = ifelse(n0==0,0,alpha0)
284     alpha1 = ifelse(n1==0,0,alpha1)
285
286     #print(c(alpha0,alpha1,gama,phi))
287
288     c<- 1- alpha0*(1-gama)-alpha1*gama
289     mu<- gama*(1-alpha1)/c
290
291
292     l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
293     l1 = ifelse( (x == 1), log(alpha1*gama), 0)
294     l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
295     l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
          - mu) * phi, log = TRUE)))
296
297
298     #print("soma")
299     #print(sum(l0 + l1 + l2+ l3))
300     sum(l0 + l1 + l2+ l3)
301   }
302
303   escore<-function(par)
304   {
305     alpha0<-par[1]
```

```
306     alpha1<-par[2]
307     gama<-par[3]
308     phi<-par[4]
309
310     c<- 1- alpha0*(1-gama)-alpha1*gama
311     mu<- (gama*(1-alpha1))/c
312
313     a= mu * phi
314     b = a * (1 - mu)/mu
315
316     ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
317     mustar <- digamma(a) - digamma(b)
318
319
320
321     Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
322     #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
323     Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
            mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
324     Ualpha0 = sum(Ualpha01 + Ualpha03)
325
326     Ualpha0 = ifelse(n0==0,0,Ualpha0)
327
328
329     Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
330     #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
331     Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
            gama*(1-gama)*(1-alpha0))/ (c^2) ) )
332     #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
            digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
333     Ualpha1 = sum(Ualpha11 - Ualpha13)
334
335     Ualpha1 = ifelse(n1==0,0,Ualpha1)
336
337     Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
338     Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
339     Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
            mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
340
341     Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
342
343     Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
344     #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
            digamma(b))
345     mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
346     )
347     Uphi = sum(Uphi0)
```

```
348
349
350     c(Ualpha0,Ualpha1,Ugama,Uphi)
351   }
352
353   # metodo dos momentos para iniciarlizacao
354   x_bar<-mean(x01)
355   sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
356
357   p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
358   q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
359
360   #print(c(x_bar,sigma2_hat,p,q))
361
362   delta0<- n0/n
363   delta1<- n1/n
364
365   a0<- delta0/(1-x_bar)
366   a1<- delta1/(x_bar)
367
368   #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
369   par_ini<-c(a0,a1,p/(p+q),p+q)
370
371   #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
372
373   #print("par_ini")
374   #print(par_ini)
375
376   max<-optim(par_ini,loglik,
377   escore,
378   method="BFGS",
379   #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
380   control=list(fnscale=-1))
381
382   if(max$convergence != 0) print("Non-convergence")
383   # return
384   z<-c()
385   z$mv<-max$par
386   #print(z$mv)
387   z$conv<-max$convergence
388   z$loglik <- max$value
389   return(z)
390 }
391
392
393 iuGAfit <- function(y)
```

```r

{

  n <- length(y) # sample size
  n0 <- sum(y == 0) # number of zeros
  n1 <- sum(y == 1) # number of ones
  m <- n0 + n1

  ys01 <- y
  if(any(y==0)) ys01<- ys01[-which(y==0)]
  if(any(y==1)) ys01<- ys01[-which(ys01==1)]

  i01 <- (y==0)+(y==1)

  #Maxima verossimilhanca

  loglik <- function (par){

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

    alpha0 = ifelse(n0 == 0, 0, alpha0)
    alpha1 = ifelse(n1 == 0, 0, alpha1)

    c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
    mu <- gama * (1 - alpha1) / c
    d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))

    l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
    l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
    l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
    l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))

    #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))

    sum(l0 + l1 + l2 + l3)

  }
  escore <- function (par) {

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

```

```
441     alpha0 <- ifelse(n0 == 0, 0, alpha0)
442     alpha1 <- ifelse(n1 == 0, 0, alpha1)
443
444     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
445     mu <- gama * (1 - alpha1) / c
446     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
447     ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
448     mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
449
450
451     Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
452     Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
            / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
            ^ 2))
453     Ualpha0 <- sum(Ualpha01 + Ualpha03)
454
455     Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
456
457     Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
458     Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
            (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
            2))
459     Ualpha1 <- sum(Ualpha11 + Ualpha13)
460
461     Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
462
463
464     Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
465     Ugama02 <- ifelse((y == 1), 1 / gama, 0)
466     Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
            + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
             ^ 2))
467
468     Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
469
470     Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
            /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
            phi) - log(mu^(1/phi)))
471
472     Uphi = sum(Uphi0)
473
474
475     c(Ualpha0, Ualpha1, Ugama, Uphi)
476   }
477
478   #Inicializacao
479
```

```
480   y_bar <- mean(y)
481   y_bar0 <- mean(ys01)
482
483   delta0 <- n0/n
484   delta1 <- n1/n
485
486   a0 <- delta0/(1-y_bar)
487   a1 <- delta1/(y_bar)
488
489   #fit <- vglm(ys01 ~ 1, gamma2)
490   #par<-Coef(fit)
491
492   #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
493   #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
494   #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
495   #phi_til = 5
496   phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
497
498   par_ini <- c(a0, a1, y_bar, phi_til)
499
500   max<-optim(par_ini, loglik, escore,method = "BFGS",
501   #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
502   #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
          (0.99999,0.99999,0.99999,9999999999),
503   control=list(fnscale=-1))
504
505   if(max$convergence != 0) print("Non-convergence")
506   # return
507   z <- c()
508   z$ini <- par_ini
509   z$mv <- max$par
510   z$conv <- max$convergence
511   z$loglik <- max$value
512   return(z)
513 }
514
515 #valores da aplicacao
516 y<-c( 0, 0.0010, 0.0050, 0, 0.1050, 0.0010, 0.0650, 0.0700, 0.0130, 0.1540,
517 0.4680, 0.3370, 0.0010, 0.0800, 0.0620, 0.0010, 0.0400, 0.0190, 0.2940, 0.0600,
518 0.0020, 0.4290, 0.0040, 0.0140, 0.0110, 0.0290, 0.0290, 0.1890, 0.0900, 0.0540,
519 0.0010, 0.0010, 0, 0.0030, 0.5130, 0.0180, 0.3800, 0.0010, 0, 0.1660,
520 0.0180, 0.0220, 0.0030, 0, 0.0580, 0.0020, 0.0660, 0.0050, 0.1530, 0.0980,
521 0, 0.3920, 0.0020, 0.0880, 0.0370, 0, 0.0150, 0.0060, 0.0120, 0.0010,
522 0.0010, 0.2860, 0.0090, 0, 0.0150, 0.3720, 0.1430, 0.0640, 0.0120, 0.0090,
523 0.0050, 0.3020, 0.0020, 0.1910, 0.0910, 0.3170, 0.0410, 0.1610, 0.0070, 0.0010,
524 0.0020, 0.0060, 0.0020, 0.0120, 0.2780, 0.0700, 0.4450, 0.0050, 0.0090, 0.0090,
```

```
0.0010, 0.1790, 0.2090, 0.2310, 0.0310, 0.3000, 0.0260, 0.0070, 0.0030, 0.0180,
0.0670, 0.0100, 0.0010, 0.0630, 0.0800, 0.0020, 0, 0.0020, 0.3650, 0.0640,
0.2290, 0.1360, 0.0010, 0.0010, 0.2140, 0.0220, 0.0010, 0.1310, 0.0150, 0.0020,
0.0730, 0.0510, 0.0370, 0.0130, 0.4220, 0.0130, 0.2540, 0.0070, 0.0010, 0,
0.0160, 0.0300, 0.0020, 0.0330, 0.1390, 0.0320, 0.0100, 0, 0.0010, 0,
0.0250, 0, 0.0420, 0.0350, 0.0130, 0.2020, 0.0110, 0.0660, 0.0080, 0.0070,
0.0310, 0.0200, 0.0040, 0.0070, 0.1050, 0.0490, 0.0360, 0.0010, 0.0010, 0.0590,
0.0510, 0.0320, 0.0060, 0.4880, 0.0460, 0.0220, 0.0040, 0.2350, 0.0740, 0.0010,
0.0120, 0.0030, 0.0110, 0.0050, 0, 0.0020, 0.0030, 0.0030, 0.0060, 0,
0.0250, 0.0340, 0.0270, 0.0020, 0, 0.0010, 0.0370, 0.0080, 0.0240, 0.0770,
0.0280, 0.0380, 0.0010, 0, 0.0520, 0.0020, 0.0070, 0.2360, 0.0100, 0.0490,
0.0090, 0.0010, 0.0050, 0.0210, 0.0010, 0.0030, 0.0010, 0.0020, 0.0040)

#descricao dos dados
summary(y)
var(y)
sd(y)
cv<- (sd(y))/(mean(y))
cv
len = length(y)
n=len
n
prop.z = sum(ifelse(y==0,1,0))/len
prop.z
prop.u = sum(ifelse(y==1,1,0))/len
prop.u
var_y=var(y)
media_y=mean(y)
media_y
var_y

fitib = ibetafit(y)
if (fitib$conv != 0)
{
  cont.ib = cont.ib+1
}

fitik = ikumafit(y)
if (fitik$conv != 0)
{
  cont.ik = cont.ik+1
}

fitigu = iuGAfit(y)

if ((fitigu$conv != 0) ){
  cont.igu=cont.igu+1
```

```
}

fitib
fitik
fitigu
fitigu$mv

##logverossimilhanaa
log.vero_IB = fitib$loglik #BI
log.vero_IK= fitik$loglik #KI
log.vero_IUG = fitigu$loglik #GUI

k=3 #inflacionado em um

#criterios#

##AIC##
AIC_IB=-2*fitib$loglik+2*k #BI
AIC_IK= -2*fitik$loglik+2*k #KI
AIC_IUG = -2*fitigu$loglik+2*k #GUI

##AICC##
AICC_IB=-2*fitib$loglik + (2*n*k)/(n-k-1) #BI
AICC_IK= -2*fitik$loglik + (2*n*k)/(n-k-1) #KI
AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1) #GUI

##BIC##
BIC_IB=-2*fitib$loglik+k*log(n) #BI
BIC_IK=-2*fitik$loglik+k*log(n) #KI
BIC_IUG=-2*fitigu$loglik+k*log(n) #GUI

##BICC##
BICC_IB=-2*fitib$loglik+(n*k*log(n))/(n-k-1) #BI
BICC_IK=-2*fitik$loglik+(n*k*log(n))/(n-k-1) #KI
BICC_IUG=-2*fitigu$loglik+(n*k*log(n))/(n-k-1) #GUI

##HQ##
HQ_IB= -2*fitib$loglik + 2*k*log(log(n)) #BI
HQ_IK= -2*fitik$loglik + 2*k*log(log(n)) #KI
HQ_IUG= -2*fitigu$loglik + 2*k*log(log(n)) #GUI

##HQC##
HQC_IB= -2*fitib$loglik +(2*n*k*log(log(n)))/(n-k-1) #BI
HQC_IK= -2*fitik$loglik +(2*n*k*log(log(n)))/(n-k-1) #KI
HQC_IUG= -2*fitigu$loglik +(2*n*k*log(log(n)))/(n-k-1) #GUI

##testes_hipotese#
```

```
#Gama Unitaria Inflacionada
ks.iuGA = ks.test(y, "pgui",alpha0=fitigu$mv[1],alpha1=fitigu$mv[2],gama=fitigu$
    mv[3], phi=fitigu$mv[4])$p.value
ks.iuGA

library(DescTools)
ad.iuGA = AndersonDarlingTest(y, null="pgui",alpha0=fitigu$mv[1], alpha1=fitigu$
    mv[2],gama=fitigu$mv[3], phi=fitigu$mv[4])$p.value
ad.iuGA

cvm.iuGA = cvm.test(y, null="pgui",alpha0=fitigu$mv[1],alpha1=fitigu$mv[2], gama=
    fitigu$mv[3], phi=fitigu$mv[4])$p.value
cvm.iuGA

#nu=p0/p2 tau=p1/p2
p0=prop.z
p1=prop.u
p2=1-p0-p1

dp=sqrt(1/((fitib$mv[4])+1))

#Beta Inflacionada
ks.iBE = ks.test(y, "pBEINF0",mu=fitib$mv[3] , sigma=dp, nu=p0/p2)$p.value
ks.iBE

#library(DescTools)
ad.iBE = AndersonDarlingTest(y, null="pBEINF0",mu=fitib$mv[3] , sigma=dp, nu=p0/
    p2)$p.value
ad.iBE

cvm.iBE = cvm.test(y, null="pBEINF0",mu=fitib$mv[3] , sigma=dp, nu=p0/p2)$p.value
cvm.iBE

#Kumaraswamy Iflacionada
ks.ikum = ks.test(y, "pikum",lambda=fitik$coef[1], p=fitik$coef[2],omega=fitik$
    coef[3], phi=fitik$coef[4])$p.value
ks.ikum

#library(DescTools)
ad.ikum = AndersonDarlingTest(y, null="pikum",lambda=fitik$coef[1], p=fitik$coef
    [2],omega=fitik$coef[3], phi=fitik$coef[4])$p.value
ad.ikum

cvm.ikum = cvm.test(y, null="pikum",lambda=fitik$coef[1], p=fitik$coef[2],omega=
    fitik$coef[3], phi=fitik$coef[4])$p.value
cvm.ikum
### Criterios de selecao
```

```
AIC_IUG
AICC_IUG
BIC_IUG

ks.iuGA # p-valores
ad.iuGA
cvm.iuGA

##resutados
#Recebendo Resultados(AIC)
result_AIC=cbind(AIC_IB,AIC_IK,AIC_IUG)
result_AIC
#Recebendo Resultados(AICC)
result_AICC=cbind(AICC_IB,AICC_IK,AICC_IUG)
result_AICC
#Recebendo Resultados(BIC)
result_BIC=cbind(BIC_IB,BIC_IK,BIC_IUG)
result_BIC
#Recebendo Resultados(BICC)
result_BICC=cbind(BICC_IB,BICC_IK,BICC_IUG)
result_BICC
#Recebendo Resultados(HQ)
result_HQ=cbind(HQ_IB,HQ_IK,HQ_IUG)
result_HQ
#Recebendo Resultados(HQC)
result_HQC=cbind(HQC_IB,HQC_IK,HQC_IUG)
result_HQC

cvm.p.valor.iBE = (cvm.iBE)
ks.p.valor.iBE = (ks.iBE)
ad.p.valor.iBE = (ad.iBE)
cvm.p.valor.iuGA = (cvm.iuGA)
ks.p.valor.iuGA = (ks.iuGA)
ad.p.valor.iuGA = (ad.iuGA)
cvm.p.valor.ikum = (cvm.ikum)
ks.p.valor.ikum = (ks.ikum)
ad.p.valor.ikum = (ad.ikum)
result.p.valor.ks = cbind(ks.p.valor.iBE,ks.p.valor.ikum,ks.p.valor.iuGA)
result.p.valor.ad = cbind(ad.p.valor.iBE,ad.p.valor.ikum,ad.p.valor.iuGA)
result.p.valor.cvm = cbind(cvm.p.valor.iBE,cvm.p.valor.ikum,cvm.p.valor.iuGA)
M=rbind( result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,result_HQC,
    result.p.valor.ks,
result.p.valor.ad,result.p.valor.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("AIC","AICC","BIC", "BICC","HQ","HQC", "p.valor.ks",
"p.valor.ad","p.valor.cvm")
```

```
glue::glue( "Tamanho amostral: {n}", "\n", "Prob.Zero: {round(prop.z,4)}", "\n",
    "Prob.Um: {round(prop.u,4)}",
"\n", "media: {round(media_y,4)}",
"\n", "var: {round(var_y,4)}")
print("Resultados")
print(M)
library(xtable)
xtable(M)

#NOVA TABELA
result.bi=cbind(AIC_IB,AICC_IB,BIC_IB,BICC_IB,HQ_IB,HQC_IB)
result.gui=cbind(AIC_IUG,AICC_IUG,BIC_IUG,BICC_IUG,HQ_IUG,HQC_IUG)
result.ki=cbind(AIC_IK,AICC_IK,BIC_IK,BICC_IK,HQ_IK,HQC_IK)
result.bi
result.ki
result.gui

#nova tabela p-valor
result_pvalor_bi=cbind({round(ks.p.valor.iBE,3)},{round(cvm.p.valor.iBE,4)},{
    round(ad.p.valor.iBE,4)})
result_pvalor_ki=cbind({round(ks.p.valor.ikum,4)},{round(cvm.p.valor.ikum,4)},{
    round(ad.p.valor.ikum,4)})
result_pvalor_gui=cbind({round(ks.p.valor.iuGA,4)},{round(cvm.p.valor.iuGA,4)},{
    round(ad.p.valor.iuGA,4)})
result_pvalor_bi
result_pvalor_ki
result_pvalor_gui

##grafico pp plot##
set.seed(33)
library(stats)
par(mfrow=c(1,3))
##grafico da BI
dados_teoricos <- rBEINF0(n,mu=fitib$mv[3] , sigma=dp, nu=p0/p2)
dados_observados <- y
xlim=c(0,1)
ylim=c(0,1)
plot(xlim = c(0,0.65), ylim = c(0,0.65),sort(dados_teoricos), sort(dados_
    observados), main = "BI[0,1)", xlab = "", ylab = "")
abline(0, 1, col = "black", lty = 1)

##grafico da GUI
dados_teoricos <- riGU(n,alpha0=fitigu$mv[1],alpha1=fitigu$mv[2],gama=fitigu$mv
    [3], phi=fitigu$mv[4])
dados_observados <- y
xlim=c(0,1)
ylim=c(0,1)
```

```
plot(xlim = c(0,0.65), ylim = c(0,0.65),sort(dados_teoricos), sort(dados_
    observados), main = "GUI[0,1)", xlab = "", ylab = "")
abline(0, 1, col = "black", lty = 1)

##grafico da KI
dados_teoricos <- rikum(n,lambda=fitik$coef[1], p=fitik$coef[2],omega=fitik$coef
    [3], phi=fitik$coef[4])
dados_observados <- y
plot(xlim = c(0,0.65), ylim = c(0,0.65),sort(dados_teoricos), sort(dados_
    observados), main = "KI[0,1)", xlab = "", ylab = "")
abline(0, 1, col = "black", lty = 1)
dev.off()
```

```
                ##APLICACAO COM INFLACIONAMENTO EM ZERO##
#Proporcao de nascidos vivos nos municipios do estado da Paraiba, com 4000 gramas
     ou mais em 2021.
#Obs: usando como referencia a residencia da mae
#Fonte dos dados: https://datasus.saude.gov.br/
#conjunto de dados#
y<-c(0.0833, 0.0794, 0.0694, 0.0583, 0.0565, 0.1146, 0.0862, 0.0686,
0.0851, 0.0357, 0.0217, 0.0973, 0.0735, 0.0531, 0.0386, 0.0882,
0.0708, 0.0723, 0.0833, 0.0722, 0.0675, 0.0635, 0.0851, 0.0629,
0.0825, 0.0644, 0.0701, 0.0492, 0.0141, 0.1143, 0.0494, 0.0625,
0, 0.0338, 0.0868, 0.0781, 0.0645, 0.0505, 0.0137, 0.0588,
0.1321, 0.0641, 0.0432, 0.0682, 0.0658, 0.0485, 0.1154, 0.0428,
0.0222, 0.0533, 0.0842, 0.0519, 0.0455, 0.0933, 0, 0.0976,
0.0313, 0.0355, 0.0241, 0.0285, 0.0674, 0.0629, 0.0348, 0.0699,
0, 0.0744, 0.1391, 0.0460, 0.0381, 0.1043, 0.0976, 0.0606,
0.0857, 0.0414, 0.0200, 0.0843, 0.0647, 0.0566, 0.0250, 0.0447,
0.0446, 0, 0.1007, 0.0702, 0.0909, 0.0566, 0.1169, 0.0567,
0.0785, 0.0929, 0.0765, 0.0836, 0.0585, 0.1121, 0.0723, 0.0588,
0.0317, 0.0733, 0.0270, 0.0735, 0.0603, 0.0943, 0.0301, 0.0505,
0.0444, 0.0795, 0.1250, 0.0536, 0.0500, 0, 0.0629, 0.0361,
0.0625, 0.0569, 0.0130, 0.0850, 0.0726, 0.0610, 0.0357, 0.0379,
0.0750, 0.0467, 0, 0.0597, 0.0642, 0.0690, 0.0309, 0.1176,
0.1154, 0, 0.0349, 0.0227, 0.0526, 0.1429, 0, 0.0634,
0.0705, 0.0952, 0.0533, 0.0828, 0.1000, 0.0283, 0.0909, 0.0938,
0.0825, 0.1260, 0.1088, 0.0517, 0.0196, 0.0682, 0.0833, 0.0417,
0.0498, 0.0480, 0.0916, 0.0526, 0.0518, 0.0845, 0.1081, 0.0520,
0.0779, 0.0526, 0.0319, 0.0817, 0.0698, 0.0464, 0.1024, 0.1094,
0.0435, 0, 0.0505, 0.0600, 0.1250, 0, 0.0651, 0.0833,
0.0714, 0.0350, 0.0926, 0.0625, 0, 0, 0.0556, 0.0377,
0.0704, 0.0860, 0.1071, 0.1250, 0.0290, 0.0377, 0.0800, 0,
0.0652, 0.0816, 0.1034, 0.0758, 0.0576, 0.1351, 0.0689, 0.0625,
0.0476, 0.0588, 0.0556, 0.0594, 0.0274, 0.0897, 0.0413, 0.0488,
0.0450, 0.0526, 0.0322, 0.0762, 0.0648, 0.0354, 0.0287, 0.0783,
0.0698, 0.0381, 0.0536, 0.0417, 0.1379, 0.1111, 0.0323)
```

```
                ##APLICACAO COM INFLACIONAMENTO EM ZERO##
#Porcentagem de pacientes com tuberculose e VIH em 2005 nos paises do mundo.
#Fonte dos dados: https://ourworldindata.org/
#conjunto de dados
y<-c(0.0000, 0.0030, 0.0030, 0.2000, 0.5000, 0.0600, 0.0310, 0.0490, 0.0370,
    0.0250,
0.4200, 0.0150, 0.0010, 0.3600, 0.0190, 0.0550, 0.2400, 0.1900, 0.0410, 0.0310,
0.0010, 0.7900, 0.0720, 0.0120, 0.0020, 0.2200, 0.3000, 0.0880, 0.4300, 0.0770,
0.0470, 0.5100, 0.2200, 0.0930, 0.0160, 0.0690, 0.0180, 0.4100, 0.1300, 0.4700,
0.0110, 0.0000, 0.0600, 0.0070, 0.2200, 0.0490, 0.3500, 0.0860, 0.0090, 0.0510,
0.0020, 0.1200, 0.4300, 0.1600, 0.0670, 0.7600, 0.2300, 0.0080, 0.0180, 0.1000,
0.4500, 0.2400, 0.0080, 0.0460, 0.2800, 0.0360, 0.0000, 0.0620, 0.2000, 0.3900,
0.1800, 0.2900, 0.0980, 0.0080, 0.0080, 0.1000, 0.0760, 0.0160, 0.0230, 0.0000,
0.0460, 0.0540, 0.0920, 0.3500, 0.0080, 0.0010, 0.0060, 0.5400, 0.0060, 0.0080,
0.0260, 0.0430, 0.0060, 0.7300, 0.2800, 0.0180, 0.0300, 0.0630, 0.0030, 0.0080,
0.6700, 0.1300, 0.0000, 0.2000, 0.0260, 0.0000, 0.0800, 0.0170, 0.0490, 0.0010,
0.0030, 0.0050, 0.0140, 0.6100, 0.1100, 0.7000, 0.0320, 0.0540, 0.0250, 0.0310,
0.0810, 0.2000, 0.0010, 0.0000, 0.0360, 0.0390, 0.0010, 0.0000, 0.1300, 0.0990,
0.0660, 0.0530, 0.0010, 0.0170, 0.2300, 0.3400, 0.0000, 0.0160, 0.0410, 0.4500,
0.2000, 0.0330, 0.0080, 0.1100, 0.0020, 0.1100, 0.0590, 0.0010, 0.0120, 0.0390,
0.7100, 0.1400, 0.0050, 0.0280, 0.1900, 0.0270, 0.1000, 0.0010, 0.0040, 0.4700,
0.2300, 0.3500, 0.3400, 0.0060, 0.0010, 0.0920, 0.5100, 0.1200, 0.0200, 0.0670,
0.1500, 0.1300, 0.0040, 0.1700, 0.0580, 0.0050, 0.7000, 0.7100)
```

```
    ##APLICACAO COM INFLACIONAMENTO EM ZERO e UM ##
    #aplicacao_inflacionado_zero_um
    #proporcao da populacao urbana residente em domicilios ligados a rede de
        esgotamento sanitario 2013 nos municipios do estado do Espirito Santo
    #Fonte dos dados: http://www.atlasbrasil.org.br/
    library(goftest) #ad e cvm
    source("ikumafit.R")
    source("iuga_fitv4.R")
    source("ibetafit.R")
    source("iuga_v2.R")
    source("ikum-omega-phi.R")
    library(gamlss)
    library(gamlss.dist)
    library(gamlss.data)
    library(betareg)
    library(MASS)


    "ikumafit" <- function(y){

     y <- as.vector(y)

```

```
n <- length(y)
n0<- sum(y==0) # number of zeros
n1<- sum(y==1) # number of ones

lambda <- (n0+n1)/n # lambda estimator
p <- n1/(n0+n1)# p estimator

u<-1
if(n0 == 0)
{
  p=1
  u=0
}

if(n1 == 0)
{
  p=0
  u=0
}

i01 <- (y==0)+(y==1)

loglik <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  ########
  l2 = 0
  if(u!=0)
  {
    l2a = ifelse((y == 1), log(p), 0)
    l2b = ifelse((y == 0), log(1-p), 0)
    l2 = l2a + l2b
  }

  l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

  l3 = ifelse((i01 == 1), 0,
  log(phi)+
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}
```

```
69
70      escore <- function(theta)
71      {
72        omega <- theta[1]
73        phi <- theta[2]
74
75        delta = log(0.5)/log(1-omega^phi)
76        ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi
              )+1)
77
78        c = ifelse((y == 0), 0, ci)
79        c = ifelse((y == 1), 0, c)
80
81        Uomega <- phi*sum(c)
82
83        v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
84        (delta-1)*((y^phi)*log(y)/(1-y^phi)))
85        v = ifelse((y == 1), 0, v)
86
87        Uphi <- sum(v)
88
89        derivada=c(Uomega,Uphi)
90        derivada
91      }
92
93      ### initial values
94      ys01<- y
95      if(any(y==0)) ys01<- ys01[-which(y==0)]
96      if(any(y==1)) ys01<- ys01[-which(ys01==1)]
97
98      # fit <- vglm(ys01 ~ 1, kumar)
99      # par<-Coef(fit)
100
101     phi <- 5 #par[1]
102     omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)
103
104     #print(c(lambda,p,omega,phi))
105     ini <- c(omega,phi)
106
107     ##############
108
109     opt <- optim(ini, loglik, escore, # hessian = T,
110     method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e
            -9))
111
112     # opt <- optim(ini, loglik, escore,
113     # method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol
```

```
    = 1e-9),
# # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$
    double.eps),
# # upper = rep(.Machine$double.xmax,(r+2))
# lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
# upper = rep(.Machine$double.xmax,(r+2))
# )

# opt <- optim(ini, loglik, escore)

# print(opt$par)

# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)


if (opt$conv != 0)
warning("FUNCTION DID NOT CONVERGE!")

k <- c()

# print("AQUI1")
# print(gradient(escore, opt$par))

coef <- c(lambda,p,opt$par)
# coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
# print(coef)
k$coef <- coef
k$conv <- opt$conv
k$loglik <- opt$value
k$counts <- as.numeric(opt$counts[1])

# library(rootSolve)
# library(numDeriv)
# escores<-rbind(
# escore(coef),
# gradient(loglik, coef),
# grad(loglik, coef))
# print(round(escores,5))

# k$Knum<-gradient(escore, coef)

# print(k$Knum)
# k$Knum<-hessian(escore, coef, method="complex")
# print(k$Knum)
```

```
159
160       # # Expected information matrix
161       #
162       # T1 = diag(as.vector( 1/diflink1(k$lambda) ))
163       # T3 = diag(as.vector( 1/diflink3(k$omega) ))
164       # T4 = diag(as.vector( 1/diflink4(k$phi) ))
165       #
166       # h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
167       # h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)
              ^(2))
168       # delta <- log(0.5)/log(1-k$omega^k$phi)
169       #
170       # kapa<- 0.5772156649
171       # k0<- (pi^2)/6+kapa^2-2*kapa
172       #
173       # ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(
              delta*log(1-y^k$phi)+1)
174       #
175       # b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
176       # b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
177       # b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
178       # b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
179       #
180       # b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
181       #
182       # si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
183       #
184       # # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
185       # # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)
              *k$phi)) -
186       # # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta
              -2)*k$phi^3))
187       # #
188       # # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
189       #
190       # mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(
              delta+1)-kapa)/((delta-1)*k$phi)) )
191       #
192       # L = diag(-1/(k$lambda*(1-k$lambda)))
193       # V = diag((k$lambda-1)*k$phi^2*h2)
194       # M = diag(mi)
195       # S = diag(si)
196       #
197       #
198       # Igg = -t(z1) %*% L %*% (T1^2) %*% z1
199       # Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
200       # Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
```

```
201       # Isb = t(Ibs)
202       # Iss = -t(q1) %*% S %*% (T4^2) %*% q1
203       #
204       # if(u!=0)
205       # {
206         # k$pi <- as.vector(coef[(m+1):(m+u)])
207         # eta_hat2 <- as.vector(w1%*%k$pi)
208         # k$p <- as.vector(linkinv2(eta_hat2))
209         # T2 = diag(as.vector( 1/diflink2(k$p) ))
210         # P = diag(-k$lambda/(k$p*(1-k$p)))
211         # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
212         #
213         # I <- rbind(
214         # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=
              m,ncol=s)),
215         # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=
              u,ncol=s)),
216         # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
217         # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
218         # )
219         #
220         # }else{
221         # I <- rbind(
222         # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
223         # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
224         # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
225         # )
226         # }
227       #
228       # k$I <- I
229       #
230       #
231       # ############################
232       #
233       # # print(-k$Knum)
234       # # print(k$I)
235       #
236       # # vcov <- try(solve(-k$Knum))
237       # vcov <- try(solve(k$I))
238       # # vcov <- chol2inv(chol(k$I))
239       # #vcov <- K #somente para nao trancar simulacao
240       #
241       # k$cov_ok = 0
242       # if (class(vcov) == "try-error")
243       # {
244         # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
              with inversion
```

```
245         # k$cov_ok = 1
246         #
247         # if (class(vcov) == "try-error") #
248         # {
249           # vcov <- k$Knum^2
250           # warning("Problem with the variance and covariance matrix. The p-values
                are not correct.")
251           # k$cov_ok = 2
252           # }
253         # }
254
255
256       # k$vcov <- vcov
257       # stderror <- sqrt(diag(k$vcov))
258       # k$stderror <- stderror
259
260       return(k)
261     }
262
263     ibetafit<-function(x)
264     {
265
266       n<- length(x) # sample size
267       n0<- sum(x==0) # number of zeros
268       n1<- sum(x==1) # number of ones
269       m<- n0 + n1
270
271       xm0<-x[which(x>0)]
272       x01<-xm0[which(xm0<1)]
273
274       #MV
275       loglik<-function(par)
276       {
277         alpha0<-par[1]
278         alpha1<-par[2]
279         gama<-par[3]
280         phi<-par[4]
281
282         alpha0 = ifelse(n0==0,0,alpha0)
283         alpha1 = ifelse(n1==0,0,alpha1)
284
285         #print(c(alpha0,alpha1,gama,phi))
286
287         c<- 1- alpha0*(1-gama)-alpha1*gama
288         mu<- gama*(1-alpha1)/c
289
290
```

```
291         l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
292         l1 = ifelse( (x == 1), log(alpha1*gama), 0)
293         l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
294         l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi,
                (1 - mu) * phi, log = TRUE)))
295
296
297         #print("soma")
298         #print(sum(l0 + l1 + l2+ l3))
299         sum(l0 + l1 + l2+ l3)
300       }
301
302       escore<-function(par)
303       {
304         alpha0<-par[1]
305         alpha1<-par[2]
306         gama<-par[3]
307         phi<-par[4]
308
309         c<- 1- alpha0*(1-gama)-alpha1*gama
310         mu<- (gama*(1-alpha1))/c
311
312         a= mu * phi
313         b = a * (1 - mu)/mu
314
315         ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
316         mustar <- digamma(a) - digamma(b)
317
318         Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
319         #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
320         Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar
                -mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
321         Ualpha0 = sum(Ualpha01 + Ualpha03)
322
323         Ualpha0 = ifelse(n0==0,0,Ualpha0)
324
325
326         Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
327         #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
328         Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar
                )*gama*(1-gama)*(1-alpha0))/ (c^2) ) )
329         #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
                digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
330         Ualpha1 = sum(Ualpha11 - Ualpha13)
331
332         Ualpha1 = ifelse(n1==0,0,Ualpha1)
333
```

```
334         Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
335         Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
336         Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*(
                ystar-mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))
337
338         Ugama = sum(-Ugama01 + Ugama02 + Ugama04)
339
340         Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
341         #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+
                b)-digamma(b))
342         mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
343         )
344         Uphi = sum(Uphi0)
345
346
347         c(Ualpha0,Ualpha1,Ugama,Uphi)
348       }
349
350       # metodo dos momentos para iniciarlizacao
351       x_bar<-mean(x01)
352       sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
353
354       p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
355       q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
356
357       #print(c(x_bar,sigma2_hat,p,q))
358
359       delta0<- n0/n
360       delta1<- n1/n
361
362       a0<- delta0/(1-x_bar)
363       a1<- delta1/(x_bar)
364
365       #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
366       par_ini<-c(a0,a1,p/(p+q),p+q)
367
368       #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
369
370       #print("par_ini")
371       #print(par_ini)
372
373       max<-optim(par_ini,loglik,
374       escore,
375       method="BFGS",
376       #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
              (0.99999,0.99999,0.99999,9999999999),
377       control=list(fnscale=-1))
```

```
378
379       if(max$convergence != 0) print("Non-convergence")
380       # return
381       z<-c()
382       z$mv<-max$par
383       #print(z$mv)
384       z$conv<-max$convergence
385       z$loglik <- max$value
386       return(z)
387      }
388
389      iuGAfit <- function(y)
390
391      {
392
393       n <- length(y) # sample size
394       n0 <- sum(y == 0) # number of zeros
395       n1 <- sum(y == 1) # number of ones
396       m <- n0 + n1
397
398       ys01 <- y
399       if(any(y==0)) ys01<- ys01[-which(y==0)]
400       if(any(y==1)) ys01<- ys01[-which(ys01==1)]
401
402       i01 <- (y==0)+(y==1)
403
404       #Maxima verossimilhanca
405
406       loglik <- function (par){
407
408         alpha0 <- par[1]
409         alpha1 <- par[2]
410         gama <- par[3]
411         phi <- par[4]
412
413         alpha0 = ifelse(n0 == 0, 0, alpha0)
414         alpha1 = ifelse(n1 == 0, 0, alpha1)
415
416         c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
417         mu <- gama * (1 - alpha1) / c
418         d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
419
420         l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
421         l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
422         l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
423         l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi)
              )))
```

```
424
425         #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))
426
427         sum(l0 + l1 + l2 + l3)
428
429       }
430       escore <- function (par) {
431
432         alpha0 <- par[1]
433         alpha1 <- par[2]
434         gama <- par[3]
435         phi <- par[4]
436
437         alpha0 <- ifelse(n0 == 0, 0, alpha0)
438         alpha1 <- ifelse(n1 == 0, 0, alpha1)
439
440         c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
441         mu <- gama * (1 - alpha1) / c
442         d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
443         ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
444         mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
445
446
447         Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
448         Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 +
                d) / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1)
                 / (c ^ 2))
449         Ualpha0 <- sum(Ualpha01 + Ualpha03)
450
451         Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
452
453         Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
454         Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d))
                 / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) /
                 (c ^ 2))
455         Ualpha1 <- sum(Ualpha11 + Ualpha13)
456
457         Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
458
459
460         Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
461         Ugama02 <- ifelse((y == 1), 1 / gama, 0)
462         Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d *
                (1 + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1
                ) / (c ^ 2))
463
464         Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
```

```
465
466       Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(
              mu))/phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) -
              digamma(phi) - log(mu^(1/phi)))
467
468       Uphi = sum(Uphi0)
469
470       c(Ualpha0, Ualpha1, Ugama, Uphi)
471     }
472
473     #Inicializacao
474
475     y_bar <- mean(y)
476     y_bar0 <- mean(ys01)
477
478     delta0 <- n0/n
479     delta1 <- n1/n
480
481     a0 <- delta0/(1-y_bar)
482     a1 <- delta1/(y_bar)
483
484     #fit <- vglm(ys01 ~ 1, gamma2)
485     #par<-Coef(fit)
486
487     #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
488     #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
489     #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
490     #phi_til = 5
491     phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))
492
493     par_ini <- c(a0, a1, y_bar, phi_til)
494
495     max<-optim(par_ini, loglik, escore,method = "BFGS",
496     #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
            (0.99999,0.99999,0.99999,9999999999),
497     #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
            (0.99999,0.99999,0.99999,9999999999),
498     control=list(fnscale=-1))
499
500     if(max$convergence != 0) print("Non-convergence")
501     # return
502     z <- c()
503     z$ini <- par_ini
504     z$mv <- max$par
505     z$conv <- max$convergence
506     z$loglik <- max$value
507     return(z)
```

```
508     }
509
510     #dados da aplicacao
511
512     y<-c(0.0193, 1, 0.2758, 0.7458, 1, 0.2185, 0.9853, 0.7473, 0.9885, 0.2561,
513     0.8328, 0.9000, 0.0162, 0.5427, 0.0000, 0.1208, 0.7202, 0.9242, 0.4548,
            0.8917,
514     0.9478, 0.7091, 0.7137, 0.2797, 0.9475, 0.9940, 0.4273, 0.7106, 0.5896,
            0.8747,
515     0.9758, 0.7600, 0.0997, 0.7841, 0.0883, 0.2894, 0.4514, 0.5736, 0.7382, 1,
516     0.4992, 0.5043, 0.5363, 0.5318, 0.5932, 0.4457, 0.4916, 0.6633, 0.3491, 1,
517     0.3216, 0.6037)
518     #descricao dos dados
519     summary(y)
520     var(y)
521     sd(y)
522     cv<- (sd(y))/(mean(y))
523     cv
524
525     len = length(y)
526     n=len
527     n #52
528     prop.z = sum(ifelse(y==0,1,0))/len
529     prop.z
530     prop.u = sum(ifelse(y==1,1,0))/len
531     prop.u
532     var_y=var(y)
533     media_y=mean(y)
534     media_y
535     var_y
536
537     fitib = ibetafit(y)
538     if (fitib$conv != 0)
539     {
540      cont.ib = cont.ib+1
541     }
542
543     fitik = ikumafit(y)
544     if (fitik$conv != 0)
545     {
546      cont.ik = cont.ik+1
547     }
548
549     fitigu = iuGAfit(y)
550
551     if ((fitigu$conv != 0) ){
552      cont.igu=cont.igu+1
```

```
}

fitib
fitik
fitigu
fitigu$mv
#fitigu$mv
#[1] 0.06218530 0.00000000 0.06253998 38.97782620


##logverossimilhanca
# Beta inflacionada
log.vero_IB = fitib$loglik
# Kuma inflacionada
log.vero_IK= fitik$loglik
# Gamma Unitaria inflacionada
log.vero_IUG = fitigu$loglik

k=4 #inflacionado em zero e um

#criterios#

##AIC##
#Beta Inflacionada
AIC_IB=-2*fitib$loglik+2*k
#Kumaraswamy Inflacionada
AIC_IK= -2*fitik$loglik+2*k
# Gamma Unitaria inflacionada
AIC_IUG = -2*fitigu$loglik+2*k

##AICC##
#Beta Inflacionada
AICC_IB=-2*fitib$loglik + (2*n*k)/(n-k-1)
#Kumaraswamy Inflacionada
AICC_IK= -2*fitik$loglik + (2*n*k)/(n-k-1)
# Gamma Unitaria inflacionada
AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1)

##BIC##
#Beta Inflacionada
BIC_IB=-2*fitib$loglik+k*log(n)
#Kumaraswamy Inflacionada
BIC_IK=-2*fitik$loglik+k*log(n)
# Gamma Unitaria inflacionada
BIC_IUG=-2*fitigu$loglik+k*log(n)

##BICC##
```

```
600     #Beta Inflacionada
601     BICC_IB=-2*fitib$loglik+(n*k*log(n))/(n-k-1)
602     #Kumaraswamy Inflacionada
603     BICC_IK=-2*fitik$loglik+(n*k*log(n))/(n-k-1)
604     # Gamma Unitaria inflacionada
605     BICC_IUG=-2*fitigu$loglik+(n*k*log(n))/(n-k-1)
606
607     ##HQ##
608     #Beta Inflacionada
609     HQ_IB= -2*fitib$loglik + 2*k*log(log(n))
610     #Kumaraswamy Inflacionada
611     HQ_IK= -2*fitik$loglik + 2*k*log(log(n))
612     # Gamma Unitaria inflacionada
613     HQ_IUG= -2*fitigu$loglik + 2*k*log(log(n))
614
615     ##HQC##
616     #Beta Inflacionada
617     HQC_IB= -2*fitib$loglik +(2*n*k*log(log(n)))/(n-k-1)
618     #Kumaraswamy Inflacionada
619     HQC_IK= -2*fitik$loglik +(2*n*k*log(log(n)))/(n-k-1)
620     # Gamma Unitaria inflacionada
621     HQC_IUG= -2*fitigu$loglik +(2*n*k*log(log(n)))/(n-k-1)
622
623     ##testes_hipotese#
624     #Gama Unitaria Inflacionada
625     ks.iuGA = ks.test(y, "pgui",alpha0=fitigu$mv[1],alpha1=fitigu$mv[2],gama=
            fitigu$mv[3], phi=fitigu$mv[4])$p.value
626     ks.iuGA
627
628     library(DescTools)
629     ad.iuGA = AndersonDarlingTest(y, null="pgui",alpha0=fitigu$mv[1], alpha1=
            fitigu$mv[2],gama=fitigu$mv[3], phi=fitigu$mv[4])$p.value
630     ad.iuGA
631
632     cvm.iuGA = cvm.test(y, null="pgui",alpha0=fitigu$mv[1],alpha1=fitigu$mv[2],
            gama=fitigu$mv[3], phi=fitigu$mv[4])$p.value
633     cvm.iuGA
634
635     #nu=p0/p2 tau=p1/p2
636     p0=prop.z
637     p1=prop.u
638     p2=1-p0-p1
639
640     dp=sqrt(1/((fitib$mv[4])+1))
641
642     #Beta Inflacionada
643     ks.iBE = ks.test(y, "pBEINF",mu=fitib$mv[3] , sigma=dp, nu=p0/p2, tau=p1/p2)
```

```
    $p.value
ks.iBE

#library(DescTools)
ad.iBE = AndersonDarlingTest(y, null="pBEINF",mu=fitib$mv[3] , sigma=dp, nu=
    p0/p2, tau=p1/p2)$p.value
ad.iBE

cvm.iBE = cvm.test(y, null="pBEINF",mu=fitib$mv[3] , sigma=dp, nu=p0/p2, tau
    =p1/p2)$p.value
cvm.iBE

#Kumaraswamy Iflacionada
ks.ikum = ks.test(y, "pikum",lambda=fitik$coef[1], p=fitik$coef[2],omega=
    fitik$coef[3], phi=fitik$coef[4])$p.value
ks.ikum

#library(DescTools)
ad.ikum = AndersonDarlingTest(y, null="pikum",lambda=fitik$coef[1], p=fitik$
    coef[2],omega=fitik$coef[3], phi=fitik$coef[4])$p.value
ad.ikum

cvm.ikum = cvm.test(y, null="pikum",lambda=fitik$coef[1], p=fitik$coef[2],
    omega=fitik$coef[3], phi=fitik$coef[4])$p.value
cvm.ikum
### Criterios de selecao
AIC_IUG
AICC_IUG
BIC_IUG

### p-valores
ks.iuGA
ad.iuGA
cvm.iuGA

##resutados
#Recebendo Resultados(AIC)
result_AIC=cbind(AIC_IB,AIC_IK,AIC_IUG)
result_AIC
#Recebendo Resultados(AICC)
result_AICC=cbind(AICC_IB,AICC_IK,AICC_IUG)
result_AICC
#Recebendo Resultados(BIC)
result_BIC=cbind(BIC_IB,BIC_IK,BIC_IUG)
result_BIC
#Recebendo Resultados(BICC)
result_BICC=cbind(BICC_IB,BICC_IK,BICC_IUG)
```

```
685     result_BICC
686     #Recebendo Resultados(HQ)
687     result_HQ=cbind(HQ_IB,HQ_IK,HQ_IUG)
688     result_HQ
689     #Recebendo Resultados(HQC)
690     result_HQC=cbind(HQC_IB,HQC_IK,HQC_IUG)
691     result_HQC
692
693     cvm.p.valor.iBE = (cvm.iBE)
694
695     ks.p.valor.iBE = (ks.iBE)
696
697     ad.p.valor.iBE = (ad.iBE)
698
699     cvm.p.valor.iuGA = (cvm.iuGA)
700
701     ks.p.valor.iuGA = (ks.iuGA)
702
703     ad.p.valor.iuGA = (ad.iuGA)
704
705     cvm.p.valor.ikum = (cvm.ikum)
706
707     ks.p.valor.ikum = (ks.ikum)
708
709     ad.p.valor.ikum = (ad.ikum)
710
711     result.p.valor.ks = cbind(ks.p.valor.iBE,ks.p.valor.ikum,ks.p.valor.iuGA)
712
713     result.p.valor.ad = cbind(ad.p.valor.iBE,ad.p.valor.ikum,ad.p.valor.iuGA)
714
715     result.p.valor.cvm = cbind(cvm.p.valor.iBE,cvm.p.valor.ikum,cvm.p.valor.iuGA
            )
716
717     M=rbind( result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,result_HQC
            ,result.p.valor.ks,
718     result.p.valor.ad,result.p.valor.cvm)
719     colnames(M) <- c("BIm.I","Kuma.I","GU.I")
720     rownames(M) <- c("AIC","AICC","BIC", "BICC","HQ","HQC", "p.valor.ks",
721     "p.valor.ad","p.valor.cvm")
722     glue::glue( "Tamanho amostral: {n}", "\n", "Prob.Zero: {round(prop.z,4)}", "
            \n", "Prob.Um: {round(prop.u,4)}",
723     "\n", "media: {round(media_y,4)}",
724     "\n", "var: {round(var_y,4)}")
725     print("Resultados")
726     print(M)
727     library(xtable)
728     xtable(M)
```

```
729
730     #nova tabela
731     result.bi=cbind(AIC_IB,AICC_IB,BIC_IB,BICC_IB,HQ_IB,HQC_IB)
732     result.gui=cbind(AIC_IUG,AICC_IUG,BIC_IUG,BICC_IUG,HQ_IUG,HQC_IUG)
733     result.ki=cbind(AIC_IK,AICC_IK,BIC_IK,BICC_IK,HQ_IK,HQC_IK)
734     result.bi
735     result.ki
736     result.gui
737     #nova tabela p_valor
738     result_pvalor_bi=cbind({round(ks.p.valor.iBE,4)},{round(cvm.p.valor.iBE,4)
            },{round(ad.p.valor.iBE,4)})
739     result_pvalor_ki=cbind({round(ks.p.valor.ikum,4)},{round(cvm.p.valor.ikum,4)
            },{round(ad.p.valor.ikum,4)})
740     result_pvalor_gui=cbind({round(ks.p.valor.iuGA,4)},{round(cvm.p.valor.iuGA
            ,4)},{round(ad.p.valor.iuGA,4)})
741     result_pvalor_bi
742     result_pvalor_ki
743     result_pvalor_gui
744
745     #grafico pp plot##
746     set.seed(33)
747     library(stats)
748     par(mfrow=c(1,3))
749     ##grafico da BI
750     dados_teoricos <- rBEINF(n,mu=fitib$mv[3] , sigma=dp, nu=p0/p2, tau=p1/p2)
751     dados_observados <- y
752     xlim=c(0,1)
753     ylim=c(0,1)
754     plot(xlim = c(0,1),ylim = c(0,1),sort(dados_teoricos), sort(dados_observados
            ), main = "Beta Inflacionada", xlab = "", ylab = "")
755     abline(0, 1, col = "black", lty = 1)
756
757     ##grafico da GUI
758     dados_teoricos <- riGU(n,alpha0=fitigu$mv[1],alpha1=fitigu$mv[2],gama=fitigu
            $mv[3], phi=fitigu$mv[4])
759     dados_observados <- y
760     xlim=c(0,1)
761     ylim=c(0,1)
762     plot(xlim = c(0,1),ylim = c(0,1),sort(dados_teoricos), sort(dados_observados
            ), main = "Gama Unitaria Inflacionada", xlab = "", ylab = "")
763     abline(0, 1, col = "black", lty = 1)
764
765     ##grafico da KI
766     dados_teoricos <- rikum(n,lambda=fitik$coef[1], p=fitik$coef[2],omega=fitik$
            coef[3], phi=fitik$coef[4])
767     dados_observados <- y
768     xlim=c(0,1)
```

```
    ylim=c(0,1)
    plot(xlim = c(0,1),ylim = c(0,1),sort(dados_teoricos), sort(dados_observados
        ), main = "Kumaraswamy Inflacionada", xlab = "", ylab = "")
    abline(0, 1, col = "black", lty = 1)
    dev.off()
```

```
  ##APLICACAO COM INFLACIONAMENTO EM ZERO e UM ##
  #Porcentagem da populacao urbana residente em domicilios ligados a rede de
      esgoto sanitario em 2016 no estado do Rio Grande do Norte.
  #Fonte dos dados: http://www.atlasbrasil.org.br/
  #dados da aplicacao
  y<-c( 1, 0.7913, 0.5856, 0.0193, 0, 0.3302, 0.0934, 0.1248, 0.7136, 0.4666,
  0.5787, 0.8989, 0.7766, 0.4275, 0.9869, 0.3277, 1, 0.2329, 0.9913, 0.3821,
  0.1131, 0.3597, 0.6715, 0.6107, 0.7127, 0.9976, 0.0501, 0.8636, 0.1316, 0.4896,
  0.3817, 0.2949, 0.6211, 1, 0.0358, 0.9001, 0.1501, 0.8130, 0.4072, 0.3600,
  0.9163, 0.9631, 1, 0.1367, 1, 0.2050, 0.0135, 0.3246, 0.7891, 0.2133,
  0.8845, 0.4994, 1, 1, 0.4557)
```

```
  ##APLICACAO COM INFLACIONAMENTO EM UM ##
#Proporcao de votos para o candidato bolsonaro nas secoes eleitorais da cidade de
    Chaves-PA no segundo turno do ano de 2022.
#Fonte dos dados: https://www.tse.jus.br/
library(goftest) #ad e cvm
source("ikumafit.R")
source("iuga_fitv4.R")
source("ibetafit.R")
source("iuga_v2.R")
source("ikum-omega-phi.R")
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(betareg)
library(MASS)
library(extraDistr)

"ikumafit" <- function(y){

  y <- as.vector(y)

  n <- length(y)
  n0<- sum(y==0) # number of zeros
  n1<- sum(y==1) # number of ones

  lambda <- (n0+n1)/n # lambda estimator
  p <- n1/(n0+n1)# p estimator

  u<-1
```

```
if(n0 == 0)
{
  p=1
  u=0
}

if(n1 == 0)
{
  p=0
  u=0
}

i01 <- (y==0)+(y==1)

loglik <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  l2 = 0
  if(u!=0)
  {
    l2a = ifelse((y == 1), log(p), 0)
    l2b = ifelse((y == 0), log(1-p), 0)
    l2 = l2a + l2b
  }

  l1 = ifelse((i01 == 1), log(lambda), log(1-lambda))

  l3 = ifelse((i01 == 1), 0,
  log(phi)+
  log(log(0.5)/log(1-omega^phi))+
  (phi-1)*log(y)+
  ( (log(0.5)/log(1-omega^phi))-1)*log(1-y^phi)
  )

  sum(l1 + l2 + l3)
}

escore <- function(theta)
{
  omega <- theta[1]
  phi <- theta[2]

  delta = log(0.5)/log(1-omega^phi)
  ci = ((omega^(phi-1))/((1-omega^phi)*log(1-omega^phi)))*(delta*log(1-y^phi)
      +1)
```

```
75
76    c = ifelse((y == 0), 0, ci)
77    c = ifelse((y == 1), 0, c)
78
79    Uomega <- phi*sum(c)
80
81    v = ifelse((y == 0), 0, 1/phi + log(y) + ci*omega*log(omega) -
82    (delta-1)*((y^phi)*log(y)/(1-y^phi)))
83    v = ifelse((y == 1), 0, v)
84
85    Uphi <- sum(v)
86
87    derivada=c(Uomega,Uphi)
88    derivada
89  }
90
91  ### initial values
92  ys01<- y
93  if(any(y==0)) ys01<- ys01[-which(y==0)]
94  if(any(y==1)) ys01<- ys01[-which(ys01==1)]
95
96  # fit <- vglm(ys01 ~ 1, kumar)
97  # par<-Coef(fit)
98
99  phi <- 5 #par[1]
100 omega <- median(ys01)# (1-0.5^(1/par[2]))^(1/phi)
101
102 #print(c(lambda,p,omega,phi))
103 ini <- c(omega,phi)
104
105 opt <- optim(ini, loglik, escore, # hessian = T,
106 method = "BFGS", control = list(fnscale = -1)) # , maxit = 500, reltol = 1e-9))
107
108 # opt <- optim(ini, loglik, escore,
109 # method = "L-BFGS-B", control = list(fnscale = -1),#, maxit = 500, reltol = 1e
      -9),
110 # # lower = c(rep(-.Machine$double.xmax, r),.Machine$double.eps,.Machine$double
      .eps),
111 # # upper = rep(.Machine$double.xmax,(r+2))
112 # lower = c(rep(-.Machine$double.xmax, r),1.0,0.001),
113 # upper = rep(.Machine$double.xmax,(r+2))
114 # )
115
116 # opt <- optim(ini, loglik, escore)
117
118 # print(opt$par)
119
```

```
# print(names(opt))
# print(summary(opt) )
# print(summary(opt)[1:6] )
# print(opt$kkt1)


if (opt$conv != 0)
warning("FUNCTION DID NOT CONVERGE!")

k <- c()

# print("AQUI1")
# print(gradient(escore, opt$par))

coef <- c(lambda,p,opt$par)
# coef <- as.numeric(summary(opt)[1:(m+u+r+s)]) # opt$par
# print(coef)
k$coef <- coef
k$conv <- opt$conv
k$loglik <- opt$value
k$counts <- as.numeric(opt$counts[1])

# library(rootSolve)
# library(numDeriv)
# escores<-rbind(
# escore(coef),
# gradient(loglik, coef),
# grad(loglik, coef))
# print(round(escores,5))

# k$Knum<-gradient(escore, coef)

# print(k$Knum)
# k$Knum<-hessian(escore, coef, method="complex")
# print(k$Knum)


# # Expected information matrix
#
# T1 = diag(as.vector( 1/diflink1(k$lambda) ))
# T3 = diag(as.vector( 1/diflink3(k$omega) ))
# T4 = diag(as.vector( 1/diflink4(k$phi) ))
#
# h1 <- k$omega^(k$phi-2)/ ((1-k$omega^k$phi)*log(1-k$omega^k$phi))
# h2 <- k$omega^(2*k$phi-2)/ ((1-k$omega^k$phi)^(2)*log(1-k$omega^k$phi)^(2))
# delta <- log(0.5)/log(1-k$omega^k$phi)
#
```

```
# kapa<- 0.5772156649
# k0<- (pi^2)/6+kapa^2-2*kapa
#
# ci = ((k$omega^(k$phi-1))/((1-k$omega^k$phi)*log(1-k$omega^k$phi)))*(delta*
    log(1-y^k$phi)+1)
#
# b1 <- delta*k$omega^2*h2*log(k$omega)^2*log(1-y^k$phi)
# b2 <- 2*delta*k$omega^2*log(k$omega)*h1*((y^k$phi * log(y))/(1-y^k$phi))
# b3 <- (delta -1)*((y^k$phi * log(y)^2)/((1-y^k$phi)^2))
# b4 <- (ci*k$omega*log(k$omega)^2)*(h1*k$omega^2+ 1/(1-k$omega^k$phi))
#
# b1 <- as.vector(-1/(k$phi^2) + b1 - b2 - b3 + b4)
#
# si <- ifelse((i01 == 1),0,b1)# (1-k$lambda)*(b)
#
# # b2 = -(-1/(k$phi^2) - (k$omega^2)*h2*log(k$omega)^2 -
# # 2*delta*k$omega^2*log(k$omega)*h1*((1-digamma(delta+1)-kapa)/((delta-1)*k$
    phi)) -
# # (digamma(delta)*(digamma(delta)+2*(kapa-1))-trigamma(delta)+k0)/((delta-2)*
    k$phi^3))
# #
# # si <- (1-k$lambda)*(b2) # ifelse(y==c,0,b2) #
#
# mi = (k$lambda-1)*k$phi*k$omega*(log(k$omega)*h2+delta*h1*((1-digamma(delta
    +1)-kapa)/((delta-1)*k$phi)) )
#
# L = diag(-1/(k$lambda*(1-k$lambda)))
# V = diag((k$lambda-1)*k$phi^2*h2)
# M = diag(mi)
# S = diag(si)
#
#
# Igg = -t(z1) %*% L %*% (T1^2) %*% z1
# Ibb = -t(x1) %*% V %*% (T3^2) %*% x1
# Ibs = -t(x1) %*% T3 %*% M %*% T4 %*% q1
# Isb = t(Ibs)
# Iss = -t(q1) %*% S %*% (T4^2) %*% q1
#
# if(u!=0)
# {
  # k$pi <- as.vector(coef[(m+1):(m+u)])
  # eta_hat2 <- as.vector(w1%*%k$pi)
  # k$p <- as.vector(linkinv2(eta_hat2))
  # T2 = diag(as.vector( 1/diflink2(k$p) ))
  # P = diag(-k$lambda/(k$p*(1-k$p)))
  # Ipp = -t(w1) %*% P %*% (T2^2) %*% w1
  #
```

```
210     # I <- rbind(
211     # cbind(Igg,matrix(0,nrow=m,ncol=u),matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,
            ncol=s)),
212     # cbind(matrix(0,nrow=u,ncol=m),Ipp,matrix(0,nrow=u,ncol=r),matrix(0,nrow=u,
            ncol=s)),
213     # cbind(matrix(0,nrow=r,ncol=m),matrix(0,nrow=r,ncol=u),Ibb,Ibs),
214     # cbind(matrix(0,nrow=s,ncol=m),matrix(0,nrow=s,ncol=u),Isb,Iss)
215     # )
216     #
217     # }else{
218     # I <- rbind(
219     # cbind(Igg,matrix(0,nrow=m,ncol=r),matrix(0,nrow=m,ncol=s)),
220     # cbind(matrix(0,nrow=r,ncol=m),Ibb,Ibs),
221     # cbind(matrix(0,nrow=s,ncol=m),Isb,Iss)
222     # )
223     # }
224   #
225   # k$I <- I
226   #
227   #
228   # ############################
229   #
230   # # print(-k$Knum)
231   # # print(k$I)
232   #
233   # # vcov <- try(solve(-k$Knum))
234   # vcov <- try(solve(k$I))
235   # # vcov <- chol2inv(chol(k$I))
236   # #vcov <- K #somente para nao trancar simulacao
237   #
238   # k$cov_ok = 0
239   # if (class(vcov) == "try-error")
240   # {
241     # vcov <- try(solve(-k$Knum)) # it use numeric hessian if we have problems
            with inversion
242     # k$cov_ok = 1
243     #
244     # if (class(vcov) == "try-error") #
245     # {
246       # vcov <- k$Knum^2
247       # warning("Problem with the variance and covariance matrix. The p-values
              are not correct.")
248       # k$cov_ok = 2
249       # }
250     # }
251
252
```

```
  # k$vcov <- vcov
  # stderror <- sqrt(diag(k$vcov))
  # k$stderror <- stderror

  return(k)
}

ibetafit<-function(x)
{

  n<- length(x) # sample size
  n0<- sum(x==0) # number of zeros
  n1<- sum(x==1) # number of ones
  m<- n0 + n1

  xm0<-x[which(x>0)]
  x01<-xm0[which(xm0<1)]

  #MV
  loglik<-function(par)
  {
    alpha0<-par[1]
    alpha1<-par[2]
    gama<-par[3]
    phi<-par[4]

    alpha0 = ifelse(n0==0,0,alpha0)
    alpha1 = ifelse(n1==0,0,alpha1)

    #print(c(alpha0,alpha1,gama,phi))

    c<- 1- alpha0*(1-gama)-alpha1*gama
    mu<- gama*(1-alpha1)/c


    l0 = ifelse( (x == 0), log(alpha0*(1-gama)), 0)
    l1 = ifelse( (x == 1), log(alpha1*gama), 0)
    l2 = ifelse(((x == 0) | (x == 1)), 0, log(c))
    l3 = ifelse(((x == 0) | (x == 1)), 0, suppressWarnings(dbeta(x, mu * phi, (1
        - mu) * phi, log = TRUE)))


    #print("soma")
    #print(sum(l0 + l1 + l2+ l3))
    sum(l0 + l1 + l2+ l3)
  }

```

```
escore<-function(par)
{
  alpha0<-par[1]
  alpha1<-par[2]
  gama<-par[3]
  phi<-par[4]

  c<- 1- alpha0*(1-gama)-alpha1*gama
  mu<- (gama*(1-alpha1))/c

  a= mu * phi
  b = a * (1 - mu)/mu

  ystar <- ifelse( ((x == 0) | (x == 1)), 0, log(x01/(1-x01)) )
  mustar <- digamma(a) - digamma(b)

  Ualpha01 = ifelse((x == 0), (1/alpha0), 0)
  #Ualpha02 = ifelse( (x == 1) ,((1-gama)/c) ,0)
  Ualpha03 = ifelse( ((x == 0) | (x == 1)), 0, -((1-gama)/c) + ((phi*( ystar-
      mustar )*gama*(1-gama)*(1-alpha1))/ (c^2) ) )
  Ualpha0 = sum(Ualpha01 + Ualpha03)

  Ualpha0 = ifelse(n0==0,0,Ualpha0)


  Ualpha11 = ifelse((x == 1), (1/alpha1), 0)
  #Ualpha12 = ifelse( (x == 1) , (gama/c) ,0)
  Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (gama/c) + ((phi*(ystar-mustar)*
      gama*(1-gama)*(1-alpha0))/ (c^2) ) )
  #Ualpha13 = ifelse( ((x == 0) | (x == 1)),0, (phi*(log(x01)-log(1-x01)-
      digamma(a)+digamma(b))*gama*(1-gama)*(1-alpha0))/c^2)
  Ualpha1 = sum(Ualpha11 - Ualpha13)

  Ualpha1 = ifelse(n1==0,0,Ualpha1)

  Ugama01 = ifelse((x == 0), (1/(1-gama)), 0)
  Ugama02 = ifelse( (x == 1) ,(1/gama) ,0)
  Ugama04 = ifelse( ((x == 0) | (x == 1)), 0, (alpha0-alpha1)/c +(phi*( ystar-
      mustar )*(1-alpha0)*(1-alpha1)/ (c^2)))

  Ugama = sum(-Ugama01 + Ugama02 + Ugama04)

  Uphi0 = ifelse( ((x == 0) | (x == 1)),0,
  #(mu * (log(x01)-log(1-x01)-digamma(a)+digamma(b)) + log(1-x01)+digamma(a+b)-
      digamma(b))
  mu*(ystar-mustar) + digamma(phi) + log(1-x01) - digamma( (1-mu)*phi )
  )
```

```
341     Uphi = sum(Uphi0)
342
343
344     c(Ualpha0,Ualpha1,Ugama,Uphi)
345   }
346
347   # metodo dos momentos para iniciarlizacao
348   x_bar<-mean(x01)
349   sigma2_hat<-(1/(n-m-1))*sum((x01-x_bar)^2)
350
351   p<-x_bar*(((x_bar*(1-x_bar))/sigma2_hat)-1)
352   q<-(1-x_bar)*(((x_bar*(1-x_bar))/sigma2_hat)-1)
353
354   #print(c(x_bar,sigma2_hat,p,q))
355
356   delta0<- n0/n
357   delta1<- n1/n
358
359   a0<- delta0/(1-x_bar)
360   a1<- delta1/(x_bar)
361
362   #par_ini<-c(a0+0.05,a1+0.05,p/(p+q)+0.1,p+q)
363   par_ini<-c(a0,a1,p/(p+q),p+q)
364
365   #par_ini<-c(a0,a1,delta1+(1-delta0-delta1)*(p/(p+q)),p+q)
366
367   #print("par_ini")
368   #print(par_ini)
369
370   max<-optim(par_ini,loglik,
371   escore,
372   method="BFGS",
373   #method="L-BFGS-B", lower = rep(0.00001,4), upper = c
        (0.99999,0.99999,0.99999,9999999999),
374   control=list(fnscale=-1))
375
376   if(max$convergence != 0) print("Non-convergence")
377   # return
378   z<-c()
379   z$mv<-max$par
380   #print(z$mv)
381   z$conv<-max$convergence
382   z$loglik <- max$value
383   return(z)
384 }
385
386 iuGAfit <- function(y)
```

```r

{

  n <- length(y) # sample size
  n0 <- sum(y == 0) # number of zeros
  n1 <- sum(y == 1) # number of ones
  m <- n0 + n1

  ys01 <- y
  if(any(y==0)) ys01<- ys01[-which(y==0)]
  if(any(y==1)) ys01<- ys01[-which(ys01==1)]

  i01 <- (y==0)+(y==1)

  #Maxima verossimilhanca

  loglik <- function (par){

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

    alpha0 = ifelse(n0 == 0, 0, alpha0)
    alpha1 = ifelse(n1 == 0, 0, alpha1)

    c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
    mu <- gama * (1 - alpha1) / c
    d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))

    l0 <- ifelse((y == 0), log(alpha0 * (1-gama)), 0)
    l1 <- ifelse((y == 1), log(alpha1 * gama), 0)
    l2 <- ifelse(((y == 0) | (y == 1)), 0, log(c))
    l3 <- ifelse(((y == 0) | (y == 1)), 0,suppressWarnings(log(dGU(y, mu, phi))))

    #phi*log(d)-log(gamma(phi))+(d-1)*log(y)+(phi-1)*log(log(1/y))

    sum(l0 + l1 + l2 + l3)

  }
  escore <- function (par) {

    alpha0 <- par[1]
    alpha1 <- par[2]
    gama <- par[3]
    phi <- par[4]

```

```
434     alpha0 <- ifelse(n0 == 0, 0, alpha0)
435     alpha1 <- ifelse(n1 == 0, 0, alpha1)
436
437     c <- 1 - alpha0 * (1 - gama) - alpha1 * gama
438     mu <- gama * (1 - alpha1) / c
439     d <- (mu ^ (1 / phi)) / (1 - mu ^(1 / phi))
440     ystar <- ifelse(((y == 0) | (y == 1)), 0, log(y))
441     mustar <-ifelse(((y == 0) | (y == 1)), 0, -phi / d)
442
443
444     Ualpha01 <- ifelse((y == 0), 1 / alpha0, 0)
445     Ualpha03 <- ifelse(((y == 0) | (y == 1)), 0, - (1 - gama) / c + (d * (1 + d)
            / (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha1) / (c
            ^ 2))
446     Ualpha0 <- sum(Ualpha01 + Ualpha03)
447
448     Ualpha0 = ifelse(n0 == 0, 0, Ualpha0)
449
450     Ualpha11 <- ifelse((y == 1), 1 / alpha1, 0)
451     Ualpha13 <- ifelse(((y == 0) | (y == 1)), 0, - (gama / c) - ((d * (1 + d)) /
            (mu * phi)) * (ystar - mustar) * gama * (1 - gama) * (1 - alpha0) / (c ^
            2))
452     Ualpha1 <- sum(Ualpha11 + Ualpha13)
453
454     Ualpha1 <- ifelse(n1 == 0, 0, Ualpha1)
455
456
457     Ugama01 <- ifelse((y == 0), -1 / (1 - gama), 0)
458     Ugama02 <- ifelse((y == 1), 1 / gama, 0)
459     Ugama04 <- ifelse(((y == 0) | (y == 1)), 0, (alpha0 - alpha1) / c + ((d * (1
            + d)) / (mu * phi)) * (ystar - mustar) * (1 - alpha0) * (1 - alpha1) / (c
             ^ 2))
460
461     Ugama <- sum(Ugama01 + Ugama02 + Ugama04)
462
463     Uphi0 <- ifelse(((y == 0) | (y == 1)), 0, log(-log(y)) + log(d) - (d*log(mu))
            /phi - ((d^2) *log(mu)* log(y)) / ((phi^2) * (mu ^ (1 / phi))) - digamma(
            phi) - log(mu^(1/phi)))
464
465     Uphi = sum(Uphi0)
466
467
468     c(Ualpha0, Ualpha1, Ugama, Uphi)
469   }
470
471   #Inicializacao
472
```

```
  y_bar <- mean(y)
  y_bar0 <- mean(ys01)

  delta0 <- n0/n
  delta1 <- n1/n

  a0 <- delta0/(1-y_bar)
  a1 <- delta1/(y_bar)

  #fit <- vglm(ys01 ~ 1, gamma2)
  #par<-Coef(fit)

  #mu <-(y_bar*(1-a1))/(1-a0*(1-y_bar)-a1*y_bar)
  #phi_til <- -mean(log(ys01))*(mu/(1 - mu))
  #phi_til <- -(log(mean(ys01)))*(y_bar0^{1/phi0})/(1 - y_bar0^{1/phi0})
  #phi_til = 5
  phi_til <- (y_bar0/(1 - y_bar0))*sum(-log(ys01)/(n-m))

  par_ini <- c(a0, a1, y_bar, phi_til)

  max<-optim(par_ini, loglik, escore,method = "BFGS",
  #method="Nelder-Mead",#lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  #method="L-BFGS-B",lower = rep(0.00001,4), upper = c
      (0.99999,0.99999,0.99999,9999999999),
  control=list(fnscale=-1))

  if(max$convergence != 0) print("Non-convergence")
  # return
  z <- c()
  z$ini <- par_ini
  z$mv <- max$par
  z$conv <- max$convergence
  z$loglik <- max$value
  return(z)
}

#dados da aplicacao
y<-c(0.4148, 0.3750, 0.1935, 0.0833, 0.5059, 0.5208, 0.4180, 0.1676, 0.1829,
0.3355, 0.2118, 0.2571, 0.2571, 0.2338, 0.1096, 0.1271, 0.1964, 0.1569,
0.2055, 0.4052, 0.2697, 0.4379, 0.1100, 0.2749, 0.5947, 0.2013, 0.3317,
0.3239, 0.3830, 0.1579, 0.2889, 0.3252, 0.1770, 0.8485, 0.7877, 1 ,
0.3950, 0.2980, 0.2533, 0.6238, 0.1633, 0.4056, 0.1631, 0.1622, 0.4534,
0.2136, 0.1250, 0.4417, 0.4685, 0.1477, 0.2697, 0.4045, 0.4956 )
#descricao dos dados
summary(y)
var(y)
```

```
sd(y)
cv<- (sd(y))/(mean(y))
cv

len = length(y)
n=len
n #n=53
prop.z = sum(ifelse(y==0,1,0))/len
prop.z
prop.u = sum(ifelse(y==1,1,0))/len
prop.u
var_y=var(y)
media_y=mean(y)
media_y
var_y

fitib = ibetafit(y)
if (fitib$conv != 0)
{
  cont.ib = cont.ib+1
}

fitik = ikumafit(y)
if (fitik$conv != 0)
{
  cont.ik = cont.ik+1
}

fitigu = iuGAfit(y)

if ((fitigu$conv != 0) ){
  cont.igu=cont.igu+1
}

fitib
fitik
fitigu
fitigu$mv

##logverossimilhanca
log.vero_IB = fitib$loglik #BI
log.vero_IK= fitik$loglik #KI
log.vero_IUG = fitigu$loglik #GUI

k=3 #inflacionado em um

#criterios#
```

```
##AIC##
AIC_IB=-2*fitib$loglik+2*k #BI(Beta Inflacionada)
AIC_IK= -2*fitik$loglik+2*k #KI(Kumaraswamy Inflacionada)
AIC_IUG = -2*fitigu$loglik+2*k #GUI(Gama Unitaria Inflacionada)

##AICC##
AICC_IB=-2*fitib$loglik + (2*n*k)/(n-k-1) #BI
AICC_IK= -2*fitik$loglik + (2*n*k)/(n-k-1) #KI
AICC_IUG=-2*fitigu$loglik+(2*n*k)/(n-k-1) #GUI

##BIC##
BIC_IB=-2*fitib$loglik+k*log(n) #BI
BIC_IK=-2*fitik$loglik+k*log(n) #KI
BIC_IUG=-2*fitigu$loglik+k*log(n) #GUI

##BICC##
BICC_IB=-2*fitib$loglik+(n*k*log(n))/(n-k-1) #BI
BICC_IK=-2*fitik$loglik+(n*k*log(n))/(n-k-1) #KI
BICC_IUG=-2*fitigu$loglik+(n*k*log(n))/(n-k-1) #GUI

##HQ##
HQ_IB= -2*fitib$loglik + 2*k*log(log(n)) #BI
HQ_IK= -2*fitik$loglik + 2*k*log(log(n)) #KI
HQ_IUG= -2*fitigu$loglik + 2k*log(log(n)) #GUI

##HQC##
HQC_IB= -2*fitib$loglik +(2*n*k*log(log(n)))/(n-k-1) #BI
HQC_IK= -2*fitik$loglik +(2*n*k*log(log(n)))/(n-k-1) #KI
HQC_IUG= -2*fitigu$loglik +(2*n*k*log(log(n)))/(n-k-1) #GUI


##testes_hipotese#
#Gama Unitaria Inflacionada
ks.iuGA = ks.test(y, "pgui",alpha0=fitigu$mv[1],alpha1=fitigu$mv[2],gama=fitigu$
    mv[3], phi=fitigu$mv[4])$p.value
ks.iuGA

library(DescTools)
ad.iuGA = AndersonDarlingTest(y, null="pgui",alpha0=fitigu$mv[1], alpha1=fitigu$
    mv[2],gama=fitigu$mv[3], phi=fitigu$mv[4])$p.value
ad.iuGA

cvm.iuGA = cvm.test(y, null="pgui",alpha0=fitigu$mv[1],alpha1=fitigu$mv[2], gama=
    fitigu$mv[3], phi=fitigu$mv[4])$p.value
cvm.iuGA

#nu=p0/p2 tau=p1/p2
```

```
p0=prop.z
p1=prop.u
p2=1-p0-p1

dp=sqrt(1/((fitib$mv[4])+1))

#Beta Inflacionada
ks.iBE = ks.test(y, "pBEINF1",mu=fitib$mv[3] , sigma=dp, nu=p1/p2)$p.value
ks.iBE

#library(DescTools)
ad.iBE = AndersonDarlingTest(y, null="pBEINF1",mu=fitib$mv[3] , sigma=dp, nu=p1/
    p2)$p.value
ad.iBE

cvm.iBE = cvm.test(y, null="pBEINF1",mu=fitib$mv[3] , sigma=dp, nu=p1/p2)$p.value
cvm.iBE

#Kumaraswamy Inflacionada
ks.ikum = ks.test(y, "pikum",lambda=fitik$coef[1], p=fitik$coef[2],omega=fitik$
    coef[3], phi=fitik$coef[4])$p.value
ks.ikum

#library(DescTools)
ad.ikum = AndersonDarlingTest(y, null="pikum",lambda=fitik$coef[1], p=fitik$coef
    [2],omega=fitik$coef[3], phi=fitik$coef[4])$p.value
ad.ikum

cvm.ikum = cvm.test(y, null="pikum",lambda=fitik$coef[1], p=fitik$coef[2],omega=
    fitik$coef[3], phi=fitik$coef[4])$p.value
cvm.ikum
### Criterios de selecao
AIC_IUG
AICC_IUG
BIC_IUG

### p-valores
ks.iuGA
ad.iuGA
cvm.iuGA

##resutados
#Recebendo Resultados(AIC)
result_AIC=cbind(AIC_IB,AIC_IK,AIC_IUG)
result_AIC
#Recebendo Resultados(AICC)
result_AICC=cbind(AICC_IB,AICC_IK,AICC_IUG)
```

```
result_AICC
#Recebendo Resultados(BIC)
result_BIC=cbind(BIC_IB,BIC_IK,BIC_IUG)
result_BIC
#Recebendo Resultados(BICC)
result_BICC=cbind(BICC_IB,BICC_IK,BICC_IUG)
result_BICC
#Recebendo Resultados(HQ)
result_HQ=cbind(HQ_IB,HQ_IK,HQ_IUG)
result_HQ
#Recebendo Resultados(HQC)
result_HQC=cbind(HQC_IB,HQC_IK,HQC_IUG)
result_HQC

cvm.p.valor.iBE = (cvm.iBE)
ks.p.valor.iBE = (ks.iBE)
ad.p.valor.iBE = (ad.iBE)
cvm.p.valor.iuGA = (cvm.iuGA)
ks.p.valor.iuGA = (ks.iuGA)
ad.p.valor.iuGA = (ad.iuGA)
cvm.p.valor.ikum = (cvm.ikum)
ks.p.valor.ikum = (ks.ikum)
ad.p.valor.ikum = (ad.ikum)

result.p.valor.ks = cbind(ks.p.valor.iBE,ks.p.valor.ikum,ks.p.valor.iuGA)
result.p.valor.ad = cbind(ad.p.valor.iBE,ad.p.valor.ikum,ad.p.valor.iuGA)
result.p.valor.cvm = cbind(cvm.p.valor.iBE,cvm.p.valor.ikum,cvm.p.valor.iuGA)

M=rbind( result_AIC,result_AICC,result_BIC, result_BICC,result_HQ,result_HQC,
    result.p.valor.ks,
result.p.valor.ad,result.p.valor.cvm)
colnames(M) <- c("BIm.I","Kuma.I","GU.I")
rownames(M) <- c("AIC","AICC","BIC", "BICC","HQ","HQC", "p.valor.ks",
"p.valor.ad","p.valor.cvm")
glue::glue( "Tamanho amostral: {n}", "\n", "Prob.Zero: {round(prop.z,4)}", "\n",
    "Prob.Um: {round(prop.u,4)}",
"\n", "media: {round(media_y,4)}",
"\n", "var: {round(var_y,4)}")
print("Resultados")
print(M)
library(xtable)
xtable(M)

#NOVA TABELA
result.bi=cbind(AIC_IB,AICC_IB,BIC_IB,BICC_IB,HQ_IB,HQC_IB)
result.gui=cbind(AIC_IUG,AICC_IUG,BIC_IUG,BICC_IUG,HQ_IUG,HQC_IUG)
result.ki=cbind(AIC_IK,AICC_IK,BIC_IK,BICC_IK,HQ_IK,HQC_IK)
```

```
result.bi
result.ki
result.gui
#nova tabela p_valor
result_pvalor_bi=cbind({round(ks.p.valor.iBE,4)},{round(cvm.p.valor.iBE,4)},{
    round(ad.p.valor.iBE,4)})
result_pvalor_ki=cbind({round(ks.p.valor.ikum,4)},{round(cvm.p.valor.ikum,4)},{
    round(ad.p.valor.ikum,4)})
result_pvalor_gui=cbind({round(ks.p.valor.iuGA,4)},{round(cvm.p.valor.iuGA,4)},{
    round(ad.p.valor.iuGA,4)})
result_pvalor_bi
result_pvalor_ki
result_pvalor_gui

#grafico pp plot##
set.seed(33)
library(stats)
par(mfrow=c(1,3))
##grafico da BI
dados_teoricos <- rBEINF1(n,mu=fitib$mv[3] , sigma=dp, nu=p1/p2)
dados_observados <- y
plot(xlim = c(0,1), ylim = c(0,1),sort(dados_teoricos), sort(dados_observados),
    main = "BI(0,1]", xlab = "", ylab = "")
abline(0, 1, col = "black", lty = 1)

##grafico da GUI
dados_teoricos <- riGU(n,alpha0=fitigu$mv[1],alpha1=fitigu$mv[2],gama=fitigu$mv
    [3], phi=fitigu$mv[4])
dados_observados <- y
plot(xlim = c(0,1), ylim = c(0,1),sort(dados_teoricos), sort(dados_observados),
    main = "GUI(0,1]", xlab = "", ylab = "")
abline(0, 1, col = "black", lty = 1)

##grafico da KI
dados_teoricos <- rikum(n,lambda=fitik$coef[1], p=fitik$coef[2],omega=fitik$coef
    [3], phi=fitik$coef[4])
dados_observados <- y
plot(xlim = c(0,1), ylim = c(0,1),sort(dados_teoricos), sort(dados_observados),
    main = "KI(0,1]", xlab = "", ylab = "")

abline(0, 1, col = "black", lty = 1)
dev.off()
```

```
##APLICACAO COM INFLACIONAMENTO EM UM ##
#Porcentagem de pessoas que reside em domicilio com acesso a energia eletrica
    no estado de santa catarina em 1991.
#Fonte dos dados: http://www.atlasbrasil.org.br/
```

```
#dados da aplicacao
y<-c(0.7218, 0.7738, 0.9557, 1, 0.9050, 0.8172, 0.9098, 0.9591, 0.9126, 0.9205,
0.7855, 0.9629, 0.4889, 0.8660, 0.9647, 0.9146, 0.9615, 0.9359, 0.9966, 1,
0.9810, 0.8235, 0.9818, 0.9869, 0.9950, 0.9663, 0.8925, 0.9963, 0.9767, 0.9677,
0.7782, 0.7497, 0.9418, 0.5249, 0.7767, 0.9861, 0.9938, 0.9993, 0.6707, 0.9886,
0.8288, 0.8421, 0.8168, 0.8635, 0.9985, 1, 0.9629, 0.3960, 0.9971, 0.9594,
0.9380, 0.4700, 0.9870, 0.9416, 0.5865, 0.7133, 0.8573, 0.9910, 0.8929, 0.6362,
0.9508, 0.9995, 0.9384, 0.8497, 0.6036, 0.3570, 0.9485, 0.9119, 1, 0.9625,
0.9122, 0.9182, 0.5836, 0.8621, 0.9769, 0.9963, 0.9222, 0.9361, 0.9344, 0.9250,
0.6623, 0.9497, 0.9637, 0.5034, 0.9952, 0.9471, 0.8561, 0.7231, 0.9953, 0.8936,
0.9969, 0.9317, 0.8239, 0.8329, 0.9888, 0.8353, 0.9932, 0.9953, 0.9424, 0.9940,
0.9944, 0.9083, 0.9851, 0.8754, 0.8220, 0.9729, 0.9011, 0.9665, 0.9694, 0.9981,
0.9931, 0.9710, 0.9901, 0.9656, 0.9949, 1, 0.9229, 0.9031, 0.6691, 0.8672,
0.8864, 0.8749, 0.6938, 0.7463, 0.7860, 0.9959, 0.9894, 0.9331, 0.8216, 0.9142,
0.9885, 0.9302, 0.9716, 0.9991, 0.9913, 0.6947, 0.9880, 0.9887, 0.8810, 0.7111,
0.9835, 0.9714, 0.9902, 0.9663, 0.9997, 0.9933, 0.6392, 0.9194, 0.8543, 0.9905,
0.9926, 0.9757, 0.7486, 0.9089, 0.9809, 0.6080, 0.9801, 0.9104, 0.8473, 0.9821,
0.7044, 0.9983, 0.8491, 0.8978, 0.8434, 0.9027, 0.6664, 1, 0.9967, 0.9827,
0.8957, 0.8729, 0.9679, 0.9887, 0.8726, 0.9911, 0.9316, 0.9716, 0.8363, 0.7689,
0.5986, 0.9874, 0.7826, 0.5986, 0.9091, 0.7500, 0.8308, 0.9475, 0.7617, 0.9834,
0.9931, 0.9928, 0.9587, 0.9469, 0.8984, 0.9981, 0.8344, 0.8036, 0.9950, 0.7068,
0.8387, 0.8263, 0.9895, 0.9136, 0.9695, 0.9526, 0.9778, 0.9959, 0.9330, 0.7353,
0.8015, 0.9864, 0.9092, 0.8893, 0.9931, 0.9805, 0.9989, 1, 0.9581, 0.7934,
0.7615, 0.9936, 0.7685, 0.9850, 0.5721, 0.9354, 0.9980, 0.8374, 0.8770, 0.9383,
0.9785, 0.5683, 0.6935, 0.8309, 0.9979, 0.9841, 0.6144, 0.9887, 0.9302, 0.7362,
0.8312, 0.9771, 0.9881, 0.8537, 0.9570, 0.9547, 0.9100, 0.8509, 0.4964, 0.9948,
0.8546, 1, 0.9721, 0.7675, 0.9469, 0.9437, 0.9356, 0.9648, 0.8918, 0.8406,
0.9884, 0.9941, 0.7900, 0.9676, 0.9508, 0.5143, 0.9857, 0.9885, 0.4693, 0.9973,
0.8732, 0.9985, 1, 0.9143, 0.9966, 0.9973, 0.9550, 0.9975, 0.8118, 0.9128,
0.8993, 0.9979, 0.8651, 0.9305, 0.6136, 0.9535, 0.9925, 0.8554, 0.9755, 0.9213,
0.9249, 0.9512, 0.9579)
```