Faculdade de Ciências e Tecnologias

Universidade NOVA de Lisboa
Estatística Numérica Computacional

PROJETO 2

Ana Breia - 61877
Gonçalo Santos - 55585
João Funenga - 61635
Mário Miranda - 62286

2021

Conteúdo

Introdução

Neste segundo projeto de Estatística Numérica Computacional, tivemos como objetivo trabalhar sobre três grandes temas abordados nas aulas bem como algumas variantes específicas de algoritmos estudados à parte.

Primeiramente, começamos por abordar o método de reamostragem ***Jackknife*** e um exercício de aplicação sobre o mesmo.

De seguida, iremos recair sobre o ***Bootstrap*** de modo a calcularmos intervalos de confiança para parâmetros desconhecidos bem como a estimação das incertezas associadas a esse, usando o ***Jackknife***. Para além do cálculo dos intervalos de confiança para um conjunto de dados, iremos também verificar se uma hipótese postulada é passível de rejeição ou não, usando uma das versões do *bootstrap*.

Posteriormente, abordaremos o tema da regressão linear associada a um conjunto de dados que temos para análise e a estimação associada aos parâmetros que definem essa mesma reta também recorrendo ao *Bootstrap*, mais precisamente usando duas metodologias distintas, o *Bootstrap* dos pares e o *Wild Bootstrap*. Para este tipo de problemas, iremos também realizar uma análise cuidada da verificação dos pressupostos de normalidade e linearidade para confirmar que existe de facto uma correlação entre os dois grupos de dados.

No final iremos entrar no capítulo da **otimização**, no qual faremos a dedução analítica de vários estimadores para o parâmetro desconhecido na função de densidade de probabilidade que nos foi apresentada e analisaremos a relação entre as funções de verosimilhança, log-verosimilhança e a *score*. Ainda nesta última secção, iremos abordar três outros métodos para estimação falados nas aulas, *Bissection Method, Newton-Raphson* e *secant*, e compará-los às estimativas feitas inicialmente.

Jackknife

O Método de Jackknife é um método de reamostragem que tem como objetivo estimar o viés e o erro padrão dos estimadores. De uma forma resumida, vamos estimar θ (a estatística que se pretende analisar), sistemáticamente, usando todas as amostras obtidas da original, porém cada amostra com um determinado valor retirado.

Este método tem a vantagem de, em geral, ser computacionalmente mais leve que o *bootstrap*; não faz suposições de distribuições; e para além disto, pode ser combinado com o *bootstrap*.

Algoritmo para o Método Jackknife

1. Seja $X_1, ..., X_n$ uma amostra aleatória de uma população $X \sim F(\theta)$, com F e θ desconhecidos. Consideremos ainda $T = g(X_1, ..., X_n)$ um estimador de θ.
2. Consideremos $x_1, ..., x_n$ uma realização da amostra aleatória e $t = g(x_1, ..., x_n)$ um estimador de θ.
3. Sejam
$$x_2, x_3, ..., x_n, \qquad x_1, x_3, ..., x_n, \qquad ..., \qquad x_1, ..., x_{n-1}$$
as n amostras de jackknife (cada uma com tamanho $n-1$) e
$$t_1^* = g(x_2, x_3, ..., x_n), \qquad ..., \qquad t_n^* = g(x_1, ..., x_{n-1})$$
as n estimativas jackknife de θ.
4. Por fim, consideramos
$$t_{jack} = \frac{1}{n}\sum_{i=1}^{n} t_i^* = \bar{t}^*$$
(estimativa jackknife de θ)

Para além disto, tem-se que

$$bias(T) \approx bias_{jack} = (n-1)(t_{jack} - t)$$
$$V(T) \approx var_{jack} = \frac{n-1}{n}\sum_{i=1}^{n}(t_i^* - t_{jack})^2$$

Apesar de ter algumas vantagens, este método só é eficaz para funções suaves, podendo falhar na estimativa da variância, por exemplo, caso isto não aconteça. Para além disto, o erro padrão costuma ser ligeiramente maior que o obtido com bootstrap, e os intervalos de confiança também não são tão satisfatórios.

Exercício 1

Let $X \sim F$ such that $E(X) = \mu$ with μ unknown. Further, let $X1, ..., Xn \sim_{iid} X$ and $T = g(X_1, ..., X_n)$ be an estimator of μ. Show that when $T = \overline{X}$ then: (i) the jackknife estimator of μ, say T_{jack}, coincides with T; and (ii) $V(T_{jack}) = \frac{n-1}{n}\sum_{i=1}^{n}(T_i^* - T_{jack})^2$ simplifies to $\frac{S^2}{n} = V(\overline{X})$.

Como foi visto no início do capítulo, temos que

$$T_{jack} = \overline{T}^* = \frac{1}{n}\sum_{i=1}^{n} T_i^*$$

Pela hipótese ficamos com

$$T_{jack} = \frac{1}{n}\sum_{i=1}^{n} \overline{X}_i^*,$$

onde $\overline{X}_i^* = \frac{1}{n-1}(X_1 + ... + X_{i-1} + X_{i+1} + ... + X_n)$ representa o valor médio sem a i-ésima observação.

Assim,

$$\begin{aligned} T_{jack} &= \frac{1}{n}(\frac{1}{n-1}((X_2 + ... + X_n) \quad + \quad (X_1 + X_3 + ... + X_n) \quad + \quad ... \quad + \quad (X_1 + ... + X_{n-1}))) \\ &= \frac{1}{n}(\frac{1}{n-1}((n-1)(X_1 + ... + X_n)) \\ &= \frac{1}{n}(X_1 + ... + X_n) \\ &= \overline{X} = T \end{aligned}$$

Analisando agora a variância do estimador jackknife, temos que

$$\begin{aligned} V(T_{jack}) &= \frac{n-1}{n}\sum_{i=1}^{n}(T_i^* - T_{jack})^2 = \\ &= \frac{n-1}{n}\sum_{i=1}^{n}(\overline{X}_i^* - \overline{X})^2 = \\ &= \frac{n-1}{n}[(\frac{1}{n-1}(X_2 + ... + X_n) - \overline{X})^2 + ... + (\frac{1}{n-1}(X_1 + ... + X_{n-1}) - \overline{X})^2] = \\ &= \frac{n-1}{n} \cdot \frac{1}{(n-1)^2}[((X_2 + ... + X_n) - (n-1)\overline{X})^2 + ... + ((X_1 + ... + X_{n-1}) - (n-1)\overline{X})^2] = \\ &= \frac{1}{n(n-1)}[(X_2 + ... + X_n - n\overline{X} + \overline{X})^2 + ... + (X_1 + ... + X_{n-1} - n\overline{X} + \overline{X})^2] \end{aligned}$$

Como $n\overline{X} = X_1 + ... + X_n$, ficamos com

$$\begin{aligned} V(T_{jack}) &= \frac{1}{n(n-1)}[(-X_1 + \overline{X})^2 + ... + (-X_n + \overline{X})^2] = \\ &= \frac{1}{n(n-1)}\sum_{i=1}^{n}(X_i - \overline{X})^2 = \\ &= \frac{1}{n}S^2 = V(\overline{X}) \end{aligned}$$

Bootstrap

O Bootstrap é um método estatístico que utiliza uma amostra de um único conjunto de dados para criar várias amostras simuladas. Este processo permite calcular erros-padrão, construir intervalos de confiança e realizar testes de hipóteses para vários tipos de amostras.

Os métodos de bootstrap são geralmente usados sempre que a distribuição de uma população é desconhecida, ou não é totalmente especificada. Assim, a amostra é a única informação disponível sobre a população.

Existem dois tipos de Bootstrap:

1. Bootstrap Paramétrico - assume que os dados vêm de uma distribuição conhecida com parâmetros desconhecidos;
2. Bootstrap Não-Paramétrico - não faz quaisquer suposições distributivas sobre a população.

Estimativa do enviesamento e variância

Seja $X_1, ..., X_n$ uma amostra aleatória de uma população $X \sim F(\theta)$, com F e θ desconhecidos. Consideremos também $T = g(X_1, ..., X_n)$ um estimador de θ. Nestas condições podemos enunciar o seguinte princípio:

Princípio de Bootstrap

Seja $x_1, ..., x_n$ uma realização da amostra aleatória anteriormente descrita e $t = g(x_1, ..., x_n)$ um estimador de θ.

Seja ainda $\widehat{F_e}$ a distribuição empírica dos dados (i.e., a distribuição de reamostragem).

Por fim, consideramos $x_1^*, ..., x_n^*$ uma "réplica" dos dados obtidos de $\widehat{F_e}$ e $t^* = g(x_1^*, ..., x_n^*)$ uma estimativa de θ calculada a partir da "réplica".

Então, o princípio do Bootstrap afirma que:

$$\widehat{F_e} \approx F$$

$$(T - \theta) \approx_d (T^* - t)$$

Se considerarmos B igual ao número de "réplicas" que pretendemos, temos que a estimativa Bootstrap de θ é dada por

$$t_{boot} = \frac{1}{B} \sum_{b=1}^{B} t_b^* \tag{1}$$

Uma vez que, $(T - \theta)) \approx_d (T^* - t)$:

$$bias(T) = E(T - \theta) \approx_d E(T^* - t) \approx \frac{1}{B} \sum_{b=1}^{B} (t_b^* - t) = \frac{1}{B} \sum_{b=1}^{B} t_b^* - t = t_{boot} - t \tag{2}$$

$$V(T - \theta) \approx_d V(T^* - t) \approx V(T^*) \approx \frac{1}{B - 1} \sum_{b=1}^{B} (t_b^* - t_{boot})^2 \tag{3}$$

Algoritmo para o Método Bootstrap

1. A partir das observções $x_1, .., x_n$ calcular $t = g(x_1, .., x_n)$, onde $T = g(X_1, ..., X_n)$ é um estimador de θ
2. Gerar uma "réplica" $x_1^*, ..., x_n^*$ a partir de $\hat{F}$
3. Determinar $t^* = g(x_1*, ..., x_n*)$
4. Repetir os dois últimos passos tantas vezes quanto o número de "réplicas"pretendidas (B) de forma a obter $t_1^*, ..., t_B^*$
5. Usar $t_1^*, .., t_B^*$ para calcular:
   (a) O estimador de θ, $t_{boot} = \frac{1}{B} \sum_{b=1}^{B} t_b^*$
   (b) $bias(T) = E(T - \theta) \approx t_{boot} - t$
   (c) $V(T - \theta) \approx \frac{1}{B-1} \sum_{b=1}^{B} (t_b^* - t_{boot})^2$

6. Se $|\widehat{bias}|/\widehat{SE}(T) > 0.25$, relatar a estimativa corrigida do viés de θ, $\tilde{t} = t - (t_{boot} - t) = 2t - t_{boot}$; se não reporta-se t
7. Reportar $\widehat{SE}(T)$

Intervalos de Confiança

De forma geral, em estatística, intervalo de confiança (IC) é uma estimativa intervalar de um parâmetro populacional desconhecido.

Assim, estudaram-se 3 formas de calcular o intervalo de confiança por Bootstrap de um qualquer θ:

1. Intervalo de confiança **normal**, assume que a distribuição T segue uma distribuição normal. Sendo t um estimador de θ
$$IC_{(1-\alpha)}(\theta) = [t - \widehat{SE}(T)Z_{\alpha/2}, t + \widehat{SE}(T)Z_{\alpha/2}]$$
2. **Fundamental** (ou básico), usa o princípio de que $(T - \theta) \approx_d (T^* - t)$
$$IC_{(1-\alpha)}(\theta) = [t - a*_{1-\alpha/2}, t + a*_{1-\alpha/2}]$$
3. **Percentil**, este método usa diretamente a distribuição de estimativas de θ para calcular o intervalo de confiança, calculando os quantis empíricos de T^*
$$IC_{(1-\alpha)}(\theta) = [a*_{\alpha/2}, t + a*_{1-\alpha/2}]$$

O intervalo de confiança normal, terá uma cobertura aproximadamente correta se a distribuição Bootstrap se assemelhar a uma distribuição normal. No entanto, se a distribuição Bootstrap for assimétrica, o intervalo de confiança terá uma cobertura diferente daquela inicialmente pretendida, especialmente se for uma assimetria substancial. A distribuição Bootstrap pode ser analisada através de um histograma.

O intervalo de confiança *Bootstrap* fundamental (ou básico) é um método simples e razoavelmente preciso, desde que T seja não enviasado e simétrico.

O intervalo de confiança Bootstrap percentil é um IC mais simples, mas pode não ser tão preciso se houver distorção ou enviesamento na distribuição de $T - \theta$, ou se o tamanho da amostra for pequeno. Se a distribuição de T^* se aproximar de uma normal, então os intervalos normais e percentuais serão quase idênticos.

Regressão Linear

Consideremos o modelo de regressão linear

$$Y_i = \beta_0 + \beta_1 x_i + \xi_i, \quad \xi_i \underset{i.i.d}{\sim} F(.) : E(\xi_i) = 0 \quad \& \quad V(\xi_i) = \sigma^2, \quad i = 1, ..., n,$$

com $Y = (y_1, ..., y_n)$ a variável de resposta, $x = (x_1, ..., x_n)$ a covariância, $\beta = (\beta_0, \beta_1)$ o vetor dos parâmetros desconhecidos, $\xi = (\xi_1, ..., \xi_n)$ os erros aleatórios e σ^2 uma variância finita desconhecida. Para este modelo, visto que os ξ_i são variáveis aleatórias, os Y_i também o serão. Relativamente aos parâmetros "fixos" desconhecidos β_0, β_1, estes serão estimados minimizando a soma dos erros quadráticos, ou seja, minimizando a seguinte expressão

$$\sum_{i=1}^{n} \xi_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 x_i)^2$$

As estimativas que serão obtidas a partir da minimização da expressão de cima seguindo este método são chamadas **estimativas dos quadrados mínimos**. Estas são respetivamente para β_0 e β_1

$$\hat{\beta}_1 = \frac{S_{xY}}{S_{xx}}, \qquad \hat{\beta}_0 = \overline{Y} - \hat{\beta}_1 \overline{x}$$

onde

$$S_{xx} = \sum_{i=1}^{n} (x_i - \overline{x})^2 = \sum_{i=1}^{n} x_i^2 - n\overline{x}^2$$

e

$$S_{xY} = \sum_{i=1}^{n} (Y_i - \overline{Y})(x_i - \overline{x}) = \sum_{i=1}^{n} Y_i(x_i - \overline{x}) = \sum_{i=1}^{n} x_i Y_i - n\,\overline{x}\,\overline{Y}$$

A soma residual dos quadrados é dada por

$$RSS = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 = S_{YY} - \hat{\beta}_i^2 S_{xx}$$

onde

$$S_{YY} = \sum_{i=1}^{n} (Y_i - \overline{Y})^2 = \sum_{i=1}^{n} Y_i^2 - n\overline{Y}^2$$

O coeficiente de determinação do modelo compara a soma residual dos resíduos dos modelos $Y_i = \beta_0 + \beta_1 x_i + \xi_i$ e $Y_i = \beta_0 + \xi_i$ refletindo assim quanto é que x contribui para explicar Y e é dado por

$$R^2 = 1 - \frac{RSS}{S_{YY}} = \hat{\beta}_i{}^2 \frac{S_{xx}}{S_{YY}} \quad \in [0, 1]$$

Um estimador não enviesado da variância do erro σ^2 é $\hat{\sigma}^2 = \frac{RSS}{n-2}$

Existem pressupostos que temos de verificar antes de começarmos a inferir informação sobre os nossos dados. Um destes é garantir a normalidade dos resíduos. Assim, quando temos que $\xi_i \sim N(0, \sigma^2)$

- Os estimadores do erro quadrático mínimo de β_0, β_1, sendo estes $\hat{\beta}_0, \hat{\beta}_0$, coincidem com os da máxima verosimilhança.
- O estimador da variância do erro é $\frac{(n-2)\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-2}$
- $\hat{\beta}_1 \sim N\left(\beta_1, \frac{\sigma^2}{S_{xx}}\right)$ de onde tiramos a variável pivotal

$$T = \frac{\hat{\beta}_1 - \beta_1}{\sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}} = \sqrt{S_{xx}} \frac{\hat{\beta}_1 - \beta_1}{\hat{\sigma}} \sim t_{n-2}$$

- $\hat{\beta}_0 \sim N\left(\beta_0, \frac{\sigma^2}{nS_{xx}} \sum_{i=1}^{n} x_i^2\right)$ de onde tiramos a variável pivotal

$$T = \frac{\hat{\beta}_0 - \beta_0}{\sqrt{\frac{\hat{\sigma}^2}{S_{xx}} \sum_{i=1}^{n} x_i^2}} = \sqrt{\frac{nS_{xx}}{\sum_{i=1}^{n} x_i^2}} \frac{\hat{\beta}_0 - \beta_0}{\hat{\sigma}} \sim t_{n-2}$$

Agora iremos demonstrar como calcularemos os intervalos de confiança para o parâmetro fixo β_1, que será o pedido para analisarmos no trabalho. O análogo poderia ser feito para os outros parâmetros, β_0 *e* σ^2. Assim, temos que a seguinte expressão define os limites do intervalo.

$$IC_{(1-\alpha)*100\%}(\beta_1) \equiv \left]\hat{\beta}_1 - t_{n-2;\alpha/2}\sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}; \hat{\beta}_1 + t_{n-2;\alpha/2}\sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}\right[,$$

onde $t_{n-2;\alpha/2}$ representa o $1 - \alpha/2$ quantil da distribiução t_{n-2}. O teste de hipóteses bilateral para β_1 consiste em:

- Hipóteses: $H_0 : \beta_1 = a \quad vs \quad H_1 : \beta_1 \neq a$
- Estatística de teste: $T = \sqrt{S_{xx}}\frac{\hat{\beta}_1 - a}{\hat{\sigma}} \underset{H_0}{\sim} t_{n-2}$
- Decisão: Rejeitamos H_0 ao nível de significância de α se $p - value < \alpha$, onde

  $$p\text{-}value = 2 * min\{P(T > t_{obs}), P(T < t_{obs})\}$$

  e t_{obs} é o valor da estatística de teste baseada em $x_1, ..., x_n$ realizações de uma amostra aleatória $X_1, ..., X_n$

Usando o bootstrap em modelos de regressão podemos obter estimativas para o viés e variância dos estimadores $\hat{\beta_1}, \hat{\beta_0}$ *e* $\hat{\sigma^2}$ dos respetivos parâmetros desconhecidos bem como intervalos de confiança para estes mesmos parâmetros.

O referido acima pode ser feito ou com *bootstrap* dos pares $z_i = (y_i, x_i)$ ou com *bootstrap* dos resíduos ξ_i. Tendo em conta estes dois métodos, estes são assimptóticamente equivalentes caso o modelo esteja correto e podem haver ligeiras diferenças das estimativas caso a amostra de dados com que estejamos a trabalhar seja pequena. Relativamente ao *bootstrap* dos pares, este é menos sensível caso hajam violações das assunções do modelo, isto é, caso haja heterocedasticidade, este vai ter uma melhor performance comparativamente ao dos resíduos. Caso queiramos à mesma utilizar *bootstrap* dos resíduos, poderemos usar uma variante deste método, o *wild bootstrap*.

O método para fazermos o ***bootstrapping* dos pares** $z_i = (y_i, x_i)$ é implementado da seguinte maneira:

- Primeiramente fazermos fit do modelo da regressão linear aos nossos dados para estimarmos $\beta = \beta_0, \beta_1$, isto é, $\widetilde{\beta}$=$\widetilde{\beta_0}, \widetilde{\beta_1}$
- Gerar B amostras de *bootstrap* de $z = (x, y)$, isto é, (x^b, y^b) com $b = 1, ..., B$
- Fazermos fit do modelo da regressão linear a cada uma das amostras de *bootstrap* (x^b, y^b) para estimarmos $\beta = \beta_0, \beta_1$, isto é, $\widetilde{\beta^b}$=$\widetilde{\beta_0^b}, \widetilde{\beta_1^b}$
- Calcular as variâncias e viéses estimados dos estimadores e verificar se é necessário corrigir o viés nas estimativas originais.
- Reportar o valor original ou corrigido caso seja o caso das estimativas $\widetilde{\beta}$ bem como a estimativa do erro $SE(\hat{\beta})$

O método para fazermos o ***bootstrapping* dos resíduos** é implementado da seguinte maneira:

- Estimar os parâmetros desconhecidos β_0, β_1 *e* σ^2 dada a amostra original $(\widetilde{\beta_0}, \widetilde{\beta_1} \ \ e \ \ \widetilde{\sigma}^2)$ e calcular os resíduos

  $$r_i = y_i - \widetilde{\beta_0} - \widetilde{\beta_1}\, x_i, \quad i = 1, ..., n$$

  Com

  $$\widetilde{\sigma}^2 = \frac{\sum_{i=1}^{n} \left(y_i - \widetilde{\beta_0} - \widetilde{\beta_1}\, x_i \right)^2}{n-2}$$

- Gerar B amostras de *bootstrap*, $(r_1^{(b)}, ..., r_n^{(b)})$, dos resíduos $r_1, ..., r_n$ e calcular para cada um

  $$y_i^{(b)} = \widetilde{\beta_0} + \widetilde{\beta_1}\, x_i + r_i^{(b)}, \quad i = 1, ..., n$$

- Para as B amostras de *bootstrap* $\{(x_i, y_i^{(b)}\}, \quad i = 1, ..., n$, fazer o fit do modelo de regressão linear para obtermos uma estimativa $(\beta_0^{(b)}, \beta_1^{(b)})$ de $\beta = (\beta_0, \beta_1)$
- Calcular as variâncias e vieses dos estimadores e verificar se é necessário corrigir o viés das estimativas originais.
- Reportar o valor original ou corrigido caso seja o caso das estimativas $\widetilde{\beta}$ bem como a estimativa do erro $SE(\hat{\beta})$

Exercício 2

Consider the observed sample referring to the survival times of some electrical component pertaining to a car assembly factory.

1766 884 2420 695 1825 1014 2183 2586 627 965
1577 2195 1354 1325 1552 1299 71 3725 1354 159

2.a)

Let μ refer to the mean survival time of that component. Use the non-parametric bootstrap ($B = 10000$ samples) to test the hypotheses

$$H0 : \mu \leq 1020 \quad vs \quad H1 : \mu > 1020,$$

at the 10% significance level. Would it be possible to perform this test using an exact test? If so, do it and compare the results.

Para avaliar se é possível aplicar um teste exacto ao conjunto de dados faremos, primeiramente, um teste à normalidade dos dados apresentados.

```
set.seed(1234)
B = 10000  # samples
x = c(1766, 884, 2420, 695, 1825, 1014, 2183, 2586, 627, 965, 1577, 2195, 1354, 1325,
    1552, 1299, 71, 3725, 1354, 159)
# Teste à normalidade dos dados

obsvOrdered = x[order(x)]
obsvOrdered
```

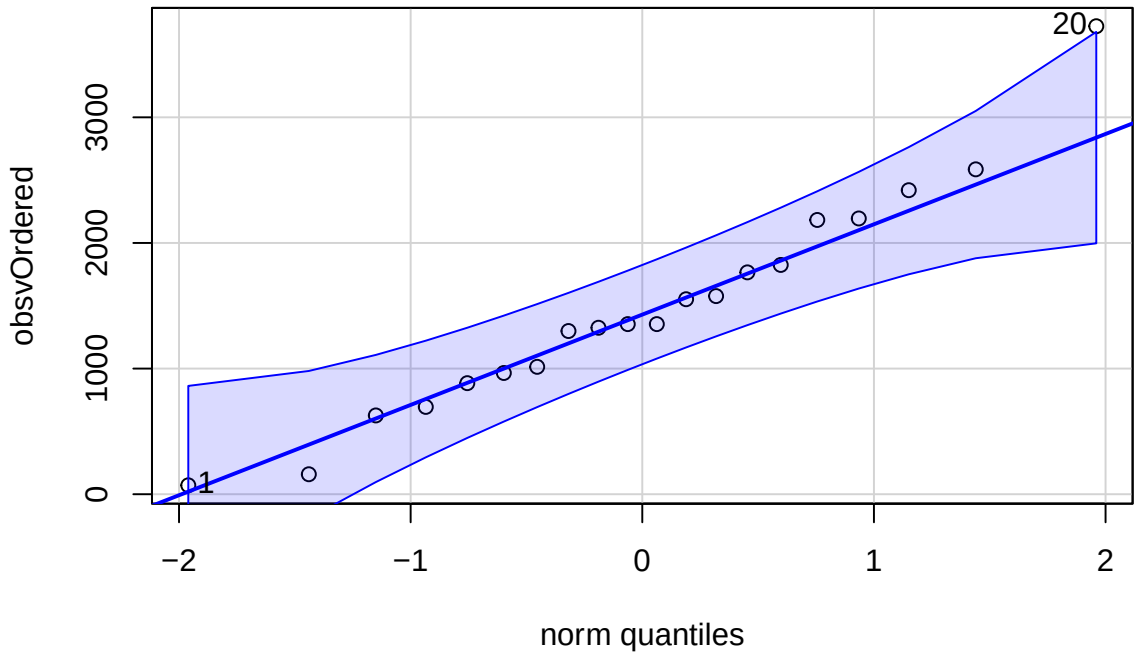
```
##  [1]   71  159  627  695  884  965 1014 1299 1325 1354 1354 1552 1577 1766 1825
## [16] 2183 2195 2420 2586 3725
```

```
probsI = (1:length(obsvOrdered) - 0.5)/length(obsvOrdered)
quantisI = qnorm(probsI)
par(pty = "s")
```

```
## Warning: package 'car' was built under R version 4.1.2
```

```
## Loading required package: carData
```

```
qqp(obsvOrdered, distribution = "norm")
```




```
## [1] 20  1
```

```
# Não rejeitamos a hipótese nula, isto é, de que a população segue uma
# distribuição normalidade
shapiro.test(x)$p.value
```

```
## [1] 0.6202625
```

Pela análise visual e pelo teste de shapiro, temos condições para assumir que a população assume uma distribuição normal. Assim, poderemos aplicar um teste exacto:

```
#Teste Exacto
n=length(x)
mu0 = 1020; sd0 = sd(x); t.obs = (mean(x)-mu0)/(sd0/sqrt(n))
p_value_exacto=pt(q=t.obs, df=n-1, lower.tail=FALSE);p_value_exacto
```

```
## [1] 0.01457727
```

```
mean(x)
```

```
## [1] 1478.8
```

```
n=length(x)
#computing the observed value of the test statistic
mu0 = 1020; sd0 = sd(x); t.obs = (mean(x)-mu0)/(sd0/sqrt(n))
# non-parametric bootstrap
B=5000; set.seed(123); t.star=numeric(B)
z=x-mean(x)+mu0 # this transformation imposes H0 on Fhat
```

```r
for(i in 1:B){
  z.star = sample(z,n,replace=T)
  sd.z.star = sd(z.star)
  t.star[i] = (mean(z.star)-mu0)/(sd.z.star/sqrt(n))
}
# computing the p.value of the test
p.value <- sum(t.star>t.obs)/B; p.value
```

```
## [1] 0.007
```

Podemos ver que o p-valor, tanto no teste exacto como no teste utilizando o *Bootstrap*, é inferior a α, para os níveis de significância usuais, logo rejeitamos a hipótese nula.

2.b)

Compute the 90% bootstrap pivotal and percentile confidence intervals for μ. Plot the histogram of the B bootstrap estimates of μ. Which CI do you think is more adequate?

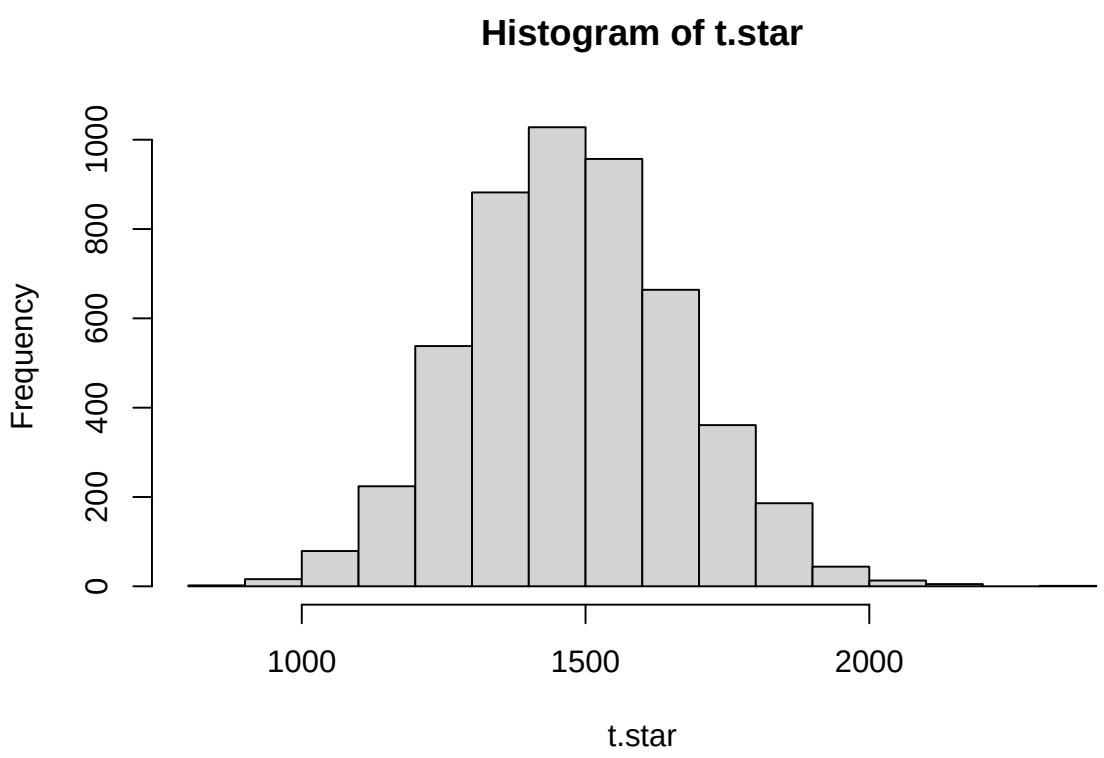
```r
t=mean(x)
t.star = numeric(B)
for(i in 1:B){ x.boot = sample(x,length(x),replace = T)
t.star[i] = mean(x.boot)}
# the bootstrap estimate of the mean is
t.boot = mean(t.star);
# computing the basic/pivotal 90% bootstrap CI
deltastar = t.star - t
d = quantile(deltastar, c(0.05,0.95))
ci = t - c(d[2], d[1])
names(ci) <- c("5%","95%");ci
```

```
##       5%      95%
## 1158.600 1785.562
```

```r
# computing the percentile 90% bootstrap CI
d = quantile(t.star, c(0.05,0.95));d
```

```
##       5%      95%
## 1172.037 1799.000
```

```r
hist(t.star)
```



Pelo histograma apresentado, podemos ver que a distribuição de Bootstrap apresenta uma distribuição aproximada à normal. Quanto aos intervalos de confiança, apesar dos máximos e mínimos dos intervalos serem diferentes o valor intervalar dos mesmos é bastante semelhante, pelo que os dois intervalos são adequados à estimação de μ.

2.c)

Research the literature for the bootstrap bias corrected and accelerated (BCa) confidence interval. Thoroughly present and discuss the BCa CI. Compute the 90% **bootstrap BCa CI for** μ**.**

This car assembly factory needs that all these components are replaced after 1100 hours of service. Let

$$T = \text{number of survival hours of a component.}$$

One is interested in estimating the proportion of components that live more than 1100 hours, i.e., one wishes to estimate $p = P(T > 1100)$**. It is known that**

$$X = \text{number of components that live more than 1100 hours in n inspected components, where } p = P(T > 1100) \text{ is the probability of a success, has distribution} \sim Bin(n, p)$$

Apesar de simples de implementar, o *"percentil interval"* tem duas limitações. Primeiro, não usa a estimativa do conjunto de dados original, é baseado nas "réplicas" obtidas pelo Bootstrap. Segundo, não ajusta a distorção da distribuição do Bootstrap.

Ora, o método BCa corrige o enviesamento e a distorção do parâmetro estimado pelo Bootstrap, incorporando um factor de correção do enviasamento (bias-correction factor), z_0. Este está relacionado com a proporção de estimativas de bootstrap que são menores do que a estatística observada.

O factor de aceleração (acceleration factor), $\hat{a}$, estima a taxa de "mudança" do erro padrão de $\hat{\theta}$ em relação ao parâmetro verdadeiro θ.

Assim, para o cálculo dos pontos limites do intervalo deste método temos:

$$\hat{z}_0 = \phi^{-1}(\frac{\#\{\widehat{\theta^*} < \hat{\theta}\}}{B})$$

Onde ϕ^{-1}é a função inversa da função de distribuição cumulativa normal padrão.

O factor de aceleração, $\hat{a}$, é estimado através do método *Jackknife*. Para cada uma das reamostragens do *jackknife*, obtemos $\widehat{\theta}(-i)$, $i = 1, ..., n$. A média destas estimativas é:

$$\widehat{\theta_{(.)}} = \sum_{i=1}^{n} \frac{\widehat{\theta_{-i}}}{m}$$

Então o factor de acelaração é:

$$\hat{a} = \frac{1}{6} \frac{\sum_{i=1}^{n} (\widehat{\theta_{(.)}} - \widehat{\theta_{(-i)}})^3}{\left(\sum_{i=1}^{n} (\widehat{\theta_{(.)}} - \widehat{\theta_{(-i)}})^2 \right)^{3/2}}$$

Com os valores de $\hat{z}_0$ e $\hat{a}$ podemos obter os valores do intervalo da seguinte forma:

$$\alpha_1 = \left\{ \hat{z}_0 + \frac{\hat{z}_0 + z^{\alpha/2}}{1 - \hat{a}(\widehat{z_0} + z^{\alpha/2})} \right\}$$

$$\alpha_2 = \left\{ \hat{z}_0 + \frac{\hat{z}_0 + z^{1-\alpha/2}}{1 - \hat{a}(\widehat{z_0} + z^{1-\alpha/2})} \right\}$$

```
library(boot);
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
#Função incorporada no Rstudio
TT <- function(data,indices){return(mean(data[indices,]))}
boot.mean <- boot(data=as.data.frame(x),statistic = TT,R=B)
boot.ci(boot.out = boot.mean,conf=0.90, type = "bca",)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.mean, conf = 0.9, type = "bca")
##
## Intervals :
## Level       BCa
## 90%   (1203, 1831 )
## Calculations and Intervals on Original Scale
```

```
z0 <- qnorm(mean(t.star<t.boot))
alpha=0.1
u <- c(alpha/2, 1-alpha/2)
zu <- qnorm(u)
n = length(x)
```

```r
# preparing for jackknifing
t1 = mean(x); t.star1 = numeric(n)
for(i in 1:n){ t.star1[i] = mean(x[-i]) }
# computing the estimates
t.jack = mean(t.star1)
uu<-(n-1)*t.jack-t.star1
acc<-sum(uu*uu*uu)/(6*(sum(uu*uu))^1.5)
tt <- pnorm(z0+ (z0+zu)/(1-acc*(z0+zu)))
confpoints <- quantile(x=t.star,probs=tt,type=1);confpoints
```

```
## 6.685128% 96.47769%
##    1202.3    1830.1
```

Podemos observar que houve um deslocamento do intervalo de confiança, comparando com os intervalos de confiança calculados anteriormente, em direção do valor "verdadeiro" da média. Note-se que o valor intervalar se manteve igual e apenas houve um deslocamento deste intervalo.

2.d)

Show that $P = \frac{X}{n}$ is an unbiased and consistent estimator of p. Estimate p and $SE(P)$.

Temos que $\frac{X}{n}$ é um estimador imparcial de p, uma vez que

$$E\left(\frac{X}{n}\right) = \sum_{x=0}^{n} \frac{x}{n} \binom{n}{x} p^x (1-p)^{n-x} =$$

$$= \frac{p}{n} \sum_{x=0}^{n} \binom{n}{x} p^{x-1} (1-p)^{n-x} =$$

$$= \frac{p}{n} \sum_{x=1}^{n} \frac{n!x}{x!(n-x)!} p^{x-1} (1-p)^{n-x} =$$

$$= p \sum_{x=0}^{n-1} \frac{(n-1)!}{x!(n-1-x)!} p^x (1-p)^{(n-1-x)} = p$$

```r
estimador_P=mean(x>1100);estimador_P
```

```
## [1] 0.65
```

```r
#standard erro
var_p<-var(x>1100)/length(x)
Standard_error=sqrt(var_p);Standard_error
```

```
## [1] 0.1094243
```

$$\hat{p} = 0.65$$

$$SE(P) = 0.1094243$$

2.e)

Describe and discuss in detail the non-parametric bootstrap and jackknife techniques. Use both approaches ($B = 10000$ samples in the case of the bootstrap) to estimate the variance, standard error and bias of P. Compare the results. Check whether there is need to correct the original estimate of p for bias and if such report the corrected estimate of p.

```r
set.seed(1234)
B = 10000  # samples
# bootstraping B samples from the observed sample and computing the bootstrap
# mean for each
t.star = numeric(B)
for (i in 1:B) {
    x.boot = sample(x, length(x), replace = T)
    t.star[i] = mean(x.boot > 1100)
}
# the bootstrap estimate of the mean is
t.boot = mean(t.star)
# estimating the SE of the sample mean estimator
se.T.boot = sqrt(var(t.star))
# an estimate of the bias of the estimator is computed as
bias.T <- t.boot - estimador_P
# variance of the estimate of P
var.T.boot = var(t.star)
var.T.boot1 = (1/(B - 1)) * sum((t.star - t.boot)^2)

boot.res = cbind(t.boot, bias.T, se.T.boot)

######################## jackknife ######################

# inputing the data
n = length(x)
# preparing for jackknifing
t.star_jack = numeric(n)
for (i in 1:n) {
    t.star_jack[i] = mean(x[-i] > 1100)
}
# computing the estimates
t.jack = mean(t.star_jack)
se.jack = sqrt((n - 1) * mean((t.star_jack - t.jack)^2))
bias.jack = (n - 1) * (t.jack - estimador_P)
# displaying the results
jack.res = cbind(t.jack, bias.jack, se.jack)


results <- rbind(jack.res, boot.res)
colnames(results) <- c("estimate", "bias", "SE")
rownames(results) <- c("jackknife", "bootstrap")
results
```

```
##           estimate     bias        SE
## jackknife 0.650000 0.000000 0.1094243
## bootstrap 0.650315 0.000315 0.1060532
```

$$t_{boot} = 0.6503$$
$$t_{jack} = 0.6500$$
$$Se_{boot} = 0.1060$$
$$SE_{jack} = 0.1094$$

Comparando o estimador Bootstrap com o estimador Jackknife, para $\hat{p}$ e $SE(P)$, calculados na alínea anterior, concluímos que não é necessário corrigirmos este estimador uma vez que os resultados são praticamente iguais.

2.f)

The jackknife-after-bootstrap technique provides a way of measuring the uncertainty associated with the bootstrap estimate $\widehat{SE}(T)$ (T some estimator of interest). Research the literature for this technique and apply it in order to estimate the standard error of the bootstrap estimate of $SE(P)$ obtained in (d).

Neste método, tal como o nome sugere, primeiro realizamos *Bootstrap* e só depois aplicamos o método Jackkniffe. Assim, esta técnica permite medir a incerteza associada à estimativa de bootstrap $\widehat{SE}(T)$.

```
set.seed(1234);
n<-20
t.star2 = numeric(B)
for(i in 1:B){ x.boot2 = sample(x,length(x),replace = T)
t.star2[i] = mean(x.boot2>1100)}

t.star_jack = numeric(n)
for(i in 1:n){ t.star_jack[i] = sqrt(var(t.star2[-i]))}
sd(t.star2)
```

```
## [1] 0.1060532
```

```
sqrt((n-1)*mean((t.star_jack-mean(t.star_jack))^2))
```

```
## [1] 2.650698e-05
```

Assim, a incerteza associada à estimativa de bootstrap $\widehat{SE}(T)$ é igual a $2.6507e-05$.

Exercício 3

Consider the following data

x 34.00 28.00 31.00 28.00 30.0 27.0 32.0 25.0 34.0 34.00 29.0 26.00 24 33.00

y 23.44 7.95 17.04 9.57 16.9 9.3 16.2 3.2 24 19.02 11.2 7.32 3 18.63

3.a)

Graphically inspect that there is a linear trend in the data. Comment on the linear trend. Fit a linear regression model to your data in R presenting and commenting in detail all the summary results referring to the fitted model (returned by the R function summary()). In addition,

- **plot the data versus the fitted line; and**
- **use the R built-in function confint() and report a 90% CI for the slope parameter.**

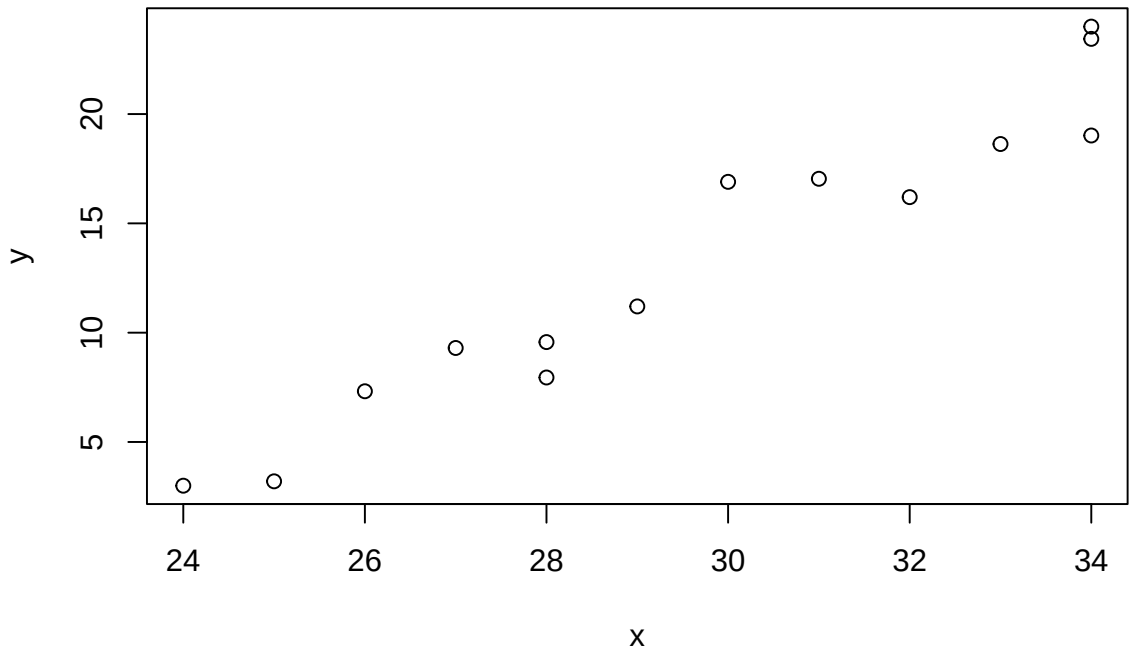
Considerando os dados que temos para X e Y, temos como objetivo ajustar um modelo de regressão linear aos dados, isto é, ajustar o modelo explicado acima

$$Y_i = \beta_0 + \beta_1 x_i + \xi_i$$

Para fazermos o ajuste do modelo aos dados que temos, precisamos primeiramente de validar o pressuposto da linearidade. Para realizarmos esta observação, representámos graficamente a variável Y em função da variável X. O gráfico resultante encontra-se abaixo e podemos observar que estes estão linearmente relacionados.

```
x = c(34, 28, 31, 28, 30, 27, 32, 25, 34, 34, 29, 26, 24, 33)
y = c(23.44, 7.95, 17.04, 9.57, 16.9, 9.3, 16.2, 3.2, 24, 19.02, 11.2, 7.32, 3, 18.63)
dataframe <- data.frame(x, y)

# representar dados graficamente
plot(x, y)
```



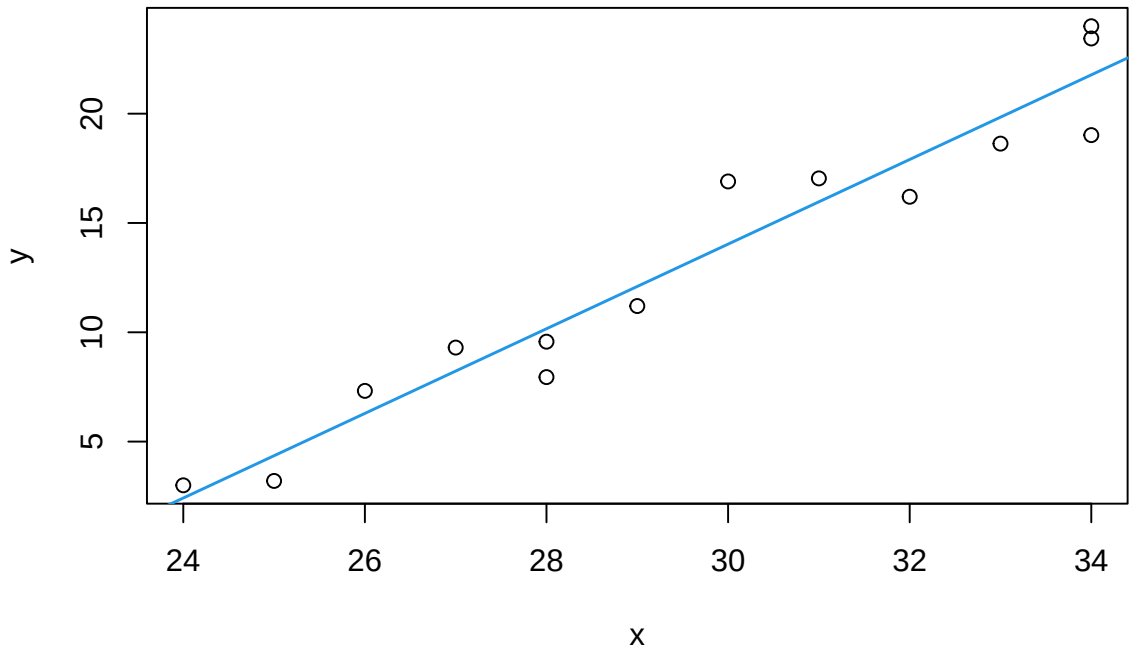
Para além de observarmos esta linearidade graficamente, também calculámos a correlação entre as duas variáveis. Este valor calculado foi de 0.968. Deste modo, é possivel afirmarmos que há uma relação linear entre as variáveis e existe uma correlação positiva entre estas pelo que validamos o pressuposto de linearidade, podendo assim ajustar um modelo de regressão linear aos dados. Abaixo representamos também a linha de regressão ajustada aos dados.

```
plot(x,y)
#correlacao entre variaveis
cor(x, y)
```

```
## [1] 0.9680854
```

```
#ajustar o modelo de reg linear aos nossos dados
fit = lm(y~x, dataframe)

# linha de regress ajustada
abline(fit,col=4,lwd=1.5)
```



Agora iremos verificar os resultados dados pela função *summary()*, que apresentam um resumo dos resultados relativos ao fit do modelo aos dados.

```
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x, data = dataframe)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.75498 -1.19528 -0.00411  1.07442  2.86795
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -44.0400     4.3152  -10.21 2.87e-07 ***
## x             1.9357     0.1447   13.38 1.42e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.802 on 12 degrees of freedom
## Multiple R-squared:  0.9372, Adjusted R-squared:  0.932
## F-statistic: 179.1 on 1 and 12 DF,  p-value: 1.424e-08
```

```
#parametros fixos beta0 e beta1
betas = fit$coefficients;betas
```

```
## (Intercept)           x
##  -44.039972    1.935734
```

```r
sigma = summary(fit)$sigma; sigma #1.802
```

```
## [1] 1.802288
```

```r
r2ajustado = summary(fit)$adj.r.square; r2ajustado #0.931
```

```
## [1] 0.931955
```

Os parâmetros fixos β_0 e β_1 são -44.04 e 1.94 respetivamente, o parâmetro aleatório é 1.80 e o R^2 ajustado é 0.93. Assim, o modelo pode ser escrito como

$$Y_i = -44.04 + 1.94x_i + 1.80$$

Uma vez que validámos todas as suposições (de linearidade e normalidade) e obtivemos um R^2 de 0.93, podemos afirmar que há uma forte relação linear entre X e Y. Como esta relação é positiva, quando X é alto, Y também o é. Uma vez que $\beta_1 = 1.94$ (declive da reta ajustada aos dados), o aumento de X numa unidade provocará um aumento de 1.94 no Y.

Agora iremos usar a função do R *confint()* para determinar o intervalo de confiança para o declive, para um nível de significância $\alpha = 0.1$.

```r
#IC a 90% para o declive
confint(fit, level = 0.90)[2,]
```

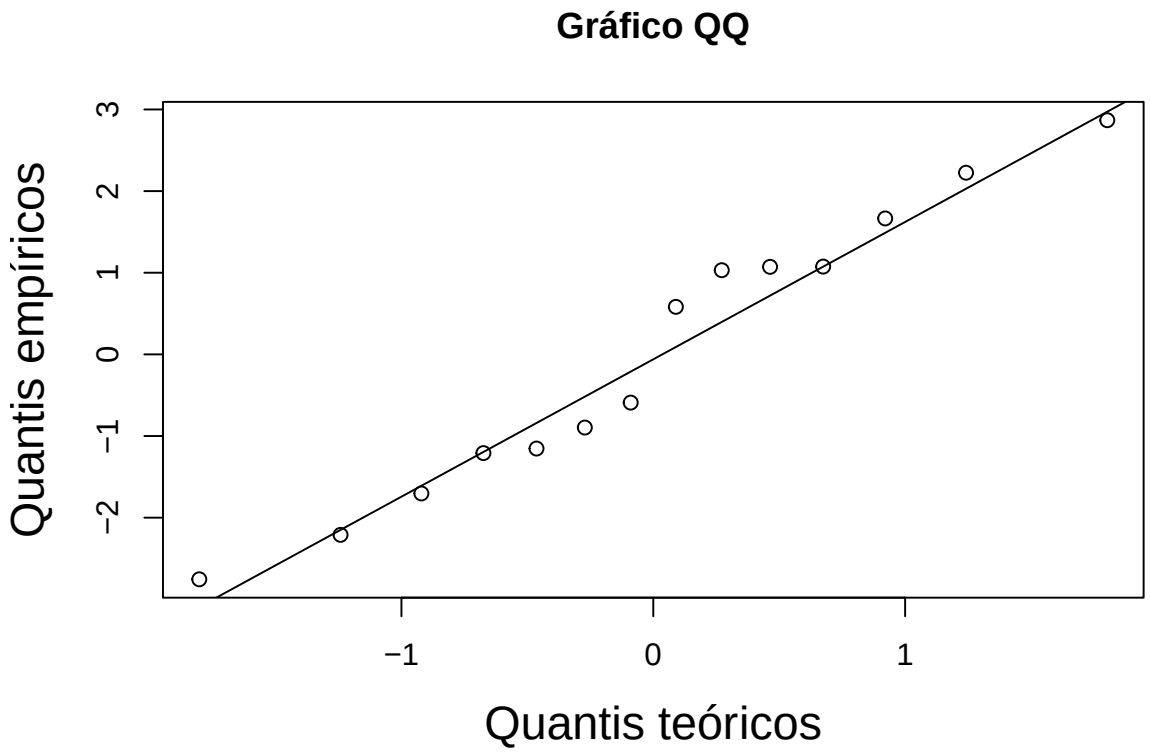
```
##      5 %     95 %
## 1.677902 2.193566
```

3.b)

Carefully check for the linear model's underlying assumptions - use both visual inspection and adequate statistical tests (at the 5% level) to check the assumptions. Does the fitted model validate all the underlying assumptions?

De seguida, iremos validar as suposições de normalidade dos resíduos, bem como a homocedasticidade do erro e a independência das observações. Para validarmos a normalidade dos resíduos (validar que $\xi_i \sim N(0, \sigma^2)$), usámos a representação do gráfico QQ e o teste de normalidade de *Shapiro-Wilk*.

```r
# Gráfico QQ (quantis empíricos vs quantis teóricos)
qqnorm(fit$residuals, main="Gráfico QQ", xlab="Quantis teóricos",
       ylab="Quantis empíricos",cex.lab=1.5)
qqline(fit$residuals)
```




```
# Teste Shapiro wilk (hipotese nula corresponde as amostras serem de uma pop normal)
shapiro.test(fit$residuals)$p.value
```

```
## [1] 0.6804204
```

O gráfico QQ serve para analisarmos o ajustamento dos dados à distribuição normal. Estes relacionam os quantis empíricos com os quantis teóricos que se esperaria observar caso as observações fossem mesmo provenientes de uma distribuição normal, sendo que o ajustamento a esta distribuição será melhor quanto mais linear for a disposição dos pontos (quantis empíricos proporcionais a quantis teóricos). Ao analisarmos o gráfico acima, percebemos que os pontos estão de facto alinhados com a reta e apresentam então uma disposição linear.

Relativamente ao teste de *Shapiro-Wilk*, este testa a hipótese nula de que uma amostra $x_1, ..., x_n$ foi retirada de uma população com distribuição normal. Ao fazermos este teste, obtivémos um *p-value* de 0.680. Visto que estávamos a usar $\alpha = 0.05$, não existem evidências estatísticas para rejeitar a normalidade dos resíduos, validando assim esta suposição (porque *p-value* $> \alpha$).

Para validarmos a homocedasticidade do erro (σ^2 é igual para todos os erros $\xi_i (i = 1, ..., n)$, realizámos o teste de *Breush-Pagan*. Este representa um teste contra a heterocedasticidade e a hipótese nula corresponde à igualdade das variâncias dos erros.

```
# Teste de Breush-Pagan (hipotese nula corresponde as variâncias do erros são iguais)
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.1.2
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

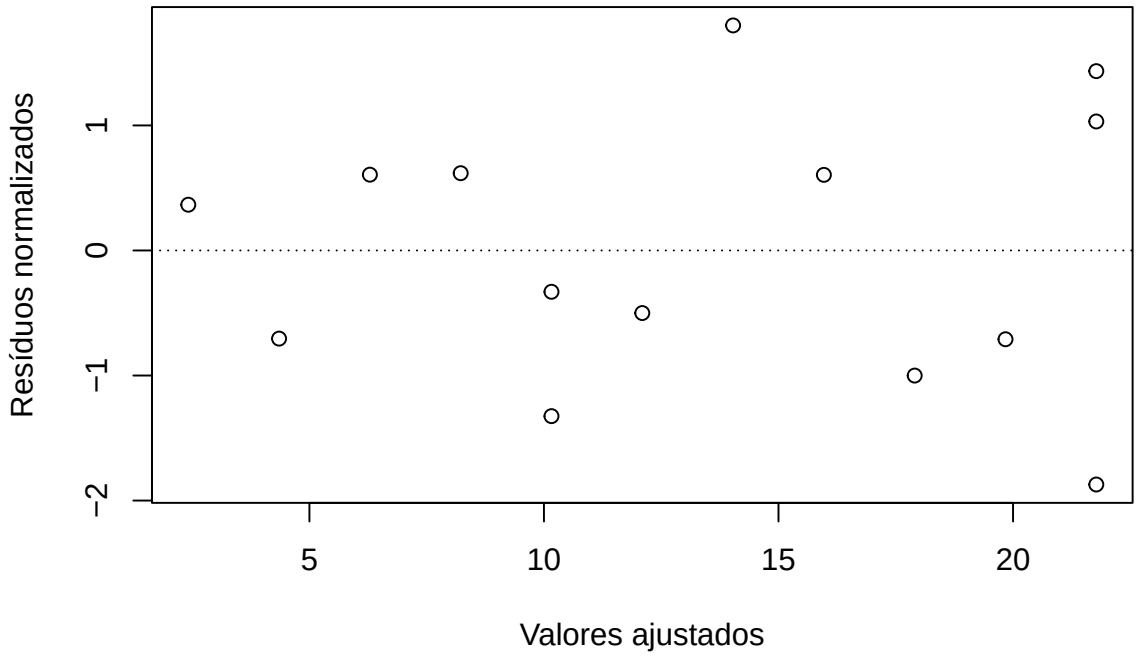
```
bptest(fit)$p.value
```

```
##         BP
## 0.06224839
```

Para este teste, o *p-value* obtido foi de 0.062. Tendo em conta que o nível de significância é de 95%, a hipótese não é rejeitada. No entanto, caso escolhessemos um nível de significância de 90%, a hipótese já poderia ser rejeitada. Assim, com o nível de significância pedido, é possível validar a homocedasticidade do erro.

Finalmente iremos validar a independência das observações. Para realisarmos isto, iremos representar o gráfico dos resíduos normalizados em função dos valores ajustados de forma a avaliar se os valores dos resíduos estão ou não aleatoriamente distribúidos em torno do valor 0 e se o valor absoluto dos resíduos não é afetado pelo valor de X.

```
# Validação da independência das observações
# Gráfico dos resíduos normalizados vs. valores ajustados
library(MASS)
plot(fit$fitted.values,studres(fit), xlab="Valores ajustados",
     ylab="Resíduos normalizados",cex.lab=1)
abline(h=0,lty=3)
```



Analisando o gráfico, observamos que os valores estão aleatoriamente distribuídos à volta do 0 e não variam com os valores ajustados o que nos leva a validar que as observações são independentes.

3.c)

Use the bootstrap of the pairs (with $B = 10000$) to

- **estimate the bias and standard error of the slope parameter estimator; check if there's need to correct the original estimate and if so report the corrected estimate;**
- **construct a pivotal 90% CI for the slope parameter; compare it with the CI obtained in (a).**

Agora iremos usar o método de *Bootstrap* dos pares para estimar a variância e o viés do estimador do parâmetro do declive. Para realizar este método, é necessário gerar B amostras de *Bootstrap* de $z = (x, y)$, (x^b, y^b) (com $b = 1, ..., B$), ajustar o modelo de regressão linear a cada amostra (x^b, y^b) de forma a podermos estimar $\beta^b = (\beta_0^b, \beta_1^b)$ e calcular a variância e o viés. Neste caso, iremos apenas fazer este procedimento para o declive.

```
#bootstrap dos pares
beta1 = fit$coefficients[2]
B = 10000
n = nrow(dataframe)
#guardar coeficientes beta1 (declive) p cada bootstrap
betaDecs = numeric(B)


for(i in 1:B){
  #amostrar aleatoriamente c replacement n amostras de um vetor com tamanho N
  idx = sample(1:n, n, replace = TRUE)

  #selecionar vals de x e y para os indexes obtidos da amostra aleatoria
  xBeta = x[idx]
  yBeta = y[idx]

  #ajustar modelo a estes vals de x e y
  fitBeta = lm(yBeta~xBeta)

  #guardar coeficiente beta1 (declive) do vetor beta
  betaDecs[i] = fitBeta$coefficients[2]
}

#variancia e vies p beta1
matrixBiasVar = matrix(c(mean(betaDecs), var(betaDecs), mean(betaDecs)-beta1), 1,3)
colnames(matrixBiasVar) = c("Media", "Variancia", "Bias")
rownames(matrixBiasVar) = "beta1"
matrixBiasVar
```

```
##          Media  Variancia         Bias
## beta1 1.936533 0.01970112 0.0007990865
```

Com isto obtemos valores de média de 1.94, variância de 0.019 e viés de 0.0041 relativamente ao declive.

Agora iremos calcular o intervalo de confiança pivotal para o declive a 90%. Começaremos por calcular $\beta_{1B}^* - \beta_1$ para as B amostras geradas pelo bootstrap e determinar quais os quantis empíricos de $\beta_1^* - \beta_1(a^*)$. O intervalo de confiança pelo declive é dado pela expressão

$$IC_{90\%}(\beta_1) = \left[\beta_1 - a^*_{1-\alpha/2}, \beta_1 - a^*_{\alpha/2}\right]$$

```
beta1
```

```
##        x
## 1.935734
```

```r
#diferenca entre beta1 obtido por bootstrap e beta1 obtido pelo fit do modelo aos dados
difBetas = betaDecs - beta1

#calculo dos quantis associados a estas diferencas
alpha = 0.1
a = quantile(difBetas, c(alpha/2, 1-alpha/2))

#ic pivotal a 90%
beta1Pivotal = beta1 - c(a[2],a[1])
beta1Pivotal
```

```
##      95%       5%
## 1.708070 2.167793
```

```r
#comparando c alinea a)
confint(fit, level = 0.90)[2,]
```

```
##      5 %     95 %
## 1.677902 2.193566
```

$$IC_{90\%}Pivotal = [1.711, 2.167]$$
$$IC_{90\%} = [1.678, 2.194]$$

Comparando os dois intervalos, pivotal e o calculado na alínea a) com a função *confint()* do R, percebemos que ambos os intervalos de confiança contêm o declive, mas que o intervalo de confiança pivotal é mais estreito (está contido no da alinea a).

3.d)

The wild bootstrap (there are several variants) is a bootstrap technique that has been shown to be more effective in the case of error heteroskedasticity than bootstrapping the pairs. Provided a detailed discussion of this method. Redo (c) using the wild bootstrap (with a variant of your choice).

Para obter uma estimativa através do wild bootstrap, primeiro definimos uma regressão de mínimos quadrados ordinários, e de seguida, multiplica-se cada valor dos resíduos por uma variável aleatória da forma

$$(1-\sqrt{5})/2, \text{ com probabilidade } (1+\sqrt{5})/(2\sqrt{5})$$
$$(1+\sqrt{5})/2, \text{ com probabilidade } 1-(1+\sqrt{5})/(2\sqrt{5}).$$

Tanto este método como o bootstrap dos pares proporcionam uma boa inferência com heterocedasticidade porém, como o wild bootstrap impõe uma restrição em cada amostra de bootstrap que seja idêntica às condições utilizadas para identificar o modelo ordinário dos mínimos quadrados será, em geral, mais preciso do que o bootstrap dos pares.

```r
library(lmboot)
```

```
## Warning: package 'lmboot' was built under R version 4.1.2
```

```r
wildBoot = wild.boot(formula = y~x, B=10000, data = NULL,  1234)
matrixBiasVarWild = matrix(c(mean(wildBoot$bootEstParam[,2]), var(wildBoot$bootEstParam[,2]), mean(wi
colnames(matrixBiasVarWild) = c("Media", "Variancia", "Bias")
rownames(matrixBiasVarWild) = "beta1"
matrixBiasVarWild
```

```
##          Media  Variancia          Bias
## beta1 1.935468 0.01653359 -0.0002661079
```

Otimização

Seja $X_1, ..., X_n$ uma amostra aleatória de uma população $X \sim F(\theta)$. Seja $f(x;\theta)$ a função de densidade de X, onde $\theta = (\theta_1, ..., \theta_p)$. Para cada concretização $(x_1, ..., x_n)$ dessa amostra aleatória, a função de verosimilhança é definida como a função de densidade conjunta de $(X_1, ..., X_n$ avaliada em $(x_1, ..., x_n)$, isto é

$$L(\theta) = L(x_1, ..., x_n; \theta) = f(x_1, ..., x_n; \theta).$$

Como $X_1, ...X_n$ são variáveis independentes e idênticamente distribuídas, então a função de verosimilhança passa a ser

$$L(\theta) = f(x_1, ..., x_n, \theta) \overset{indep.}{=} f_{X_1}(x_1;\theta) * ... * f_{X_n}(x_n;\theta) \overset{i.d}{=} \prod_{i=1}^{n} f(X_i;\theta).$$

Relativamente à função de log-verosimilhança esta seria simplesmente o logaritmo da função de verosimilhança. Por se tratar de um logaritmo sobre um produto, este passa à soma dos logaritmos, ficando assim como

$$l(\theta) = \sum_{i=1}^{n} log(f(x_i;\theta))$$

Finalmente, para a função score temos que esta é a derivada parcial da função da log-verosimilhança sobre a variável θ.

$$S(\theta) = \frac{\partial}{\partial\theta} l(\theta) = \left[\frac{\partial}{\partial\theta_1} l(\theta), ..., \frac{\partial}{\partial\theta_p} l(\theta)\right].$$

O estimador de máxima verosimilhança de θ é o que maximiza a função de verosimilhança falada em primeiro lugar.

$$\theta_{MLE} = arg \max_{\theta} L(\theta)$$

Este estimador tem 4 propriedades que o definem e são necessárias para este ser um estimador de máxima verosimilhança:

- **Consistência** - Sob certas condições (por exemplo famílias regulares), θ_{MLE} é um estimador consistente de θ, isto é, $\theta_{MLE} \xrightarrow{p} \theta$
- **Não Enviesamento Assimptótico** $\Rightarrow \lim_{n\to\infty} \mathbb{E}[\theta_{n,MLE}] = \theta$
- **Normalidade Assimptótica** $\Rightarrow$ Sob certas condições (por exemplo famílias regulares)

  $$\sqrt{n}(\theta_{MLE} - \theta) \xrightarrow{d} N(0, I(\theta)^{-1})$$

  , onde $I(\theta)$ é a matriz de informação de Fisher baseada numa observação.
- **Invariância** $\Rightarrow$ Se θ_{MLE} é o estimador de máxima verosimilhança de θ, então $h(\theta_{MLE})$ é o estimador de máxima verosimilhança de $h(\theta)$.

A Matriz Informação de Fisher é dada por

$$I_n(\theta) = \mathbb{E}[(\frac{\partial}{\partial\theta} l(\theta))(\frac{\partial}{\partial\theta} l(\theta))^T] =$$

$$= -\mathbb{E}[\frac{\partial^2}{\partial\theta\partial\theta^T} l(\theta)] =$$

$$= -\mathbb{E}\{H(\theta)\} = nI(\theta)$$

Para obter o MLE de $\theta = (\theta_1, ..., \theta_p)^t$ tendo uma conjunto $(x_1, ..., x_n)$, podemos proceder ao seguinte:

- Escrever a função de verosimilhança $L(\theta)$
- Escrever a função de log-verosimilhança $l(\theta) = log\ L(\theta)$

- Resolver a equação da função score,

$$S(\theta) = 0 \Leftrightarrow \frac{\partial}{\partial\theta} l(\theta) = 0$$

- Seja θ^* a solução da equação. Para verificar que é um máximo, temos de provar que:

$$s^T H(\theta^*) s < 0, \ \forall s \neq 0 \ \in \mathbb{R}^p$$

Existem situações em que não é possível obter analiticamente o MLE. Nesses casos podemos usar algoritmos iterativos para obter o MLE.

Notas:

- um valor de MLE θ^* tal que $l'(\theta^*) = 0$
- Maximizar a log-verosimilhança é igual a minimizar o seu simétrico, i.e.,

$$max \ l(\theta) \Leftrightarrow -min \ l(\theta)$$

- os procedimentos para otimizar a log-verosimilhança são geralmente iterativos e precisam de
  - um valor inicial $\theta^{(0)}$
  - uma equação de atualização

$$\theta^{(t+1)} = g(\theta^{(t)}), \ t = 0, 1, 2, ...$$

  - uma regra de paragem baseado num critério de convergência

Para avaliar a convergência do processo iterativo, vemos a mudança na transição de $\theta^{(t)}$ para $\theta^{(t+1)}$. Para o efeito, temos de:

- controlar a convergência avaliando a expressão $|\theta^{(t+1)} - \theta^{(t)}|$
- usamos $S(\theta^{(t+1)})$ para confirmar os resultados

Critérios de convergência:

- **Critério de Convergência Absoluto**: O processo iterativo acaba quando

$$|\theta^{(t+1)} - \theta^{(t)}| < \varepsilon.$$

  Onde ε é um valor de precisão constante.

- **Critério de Convergência Relativo**: O processo iterativo acaba quando

$$\frac{|\theta^{(t+1)} - \theta^{(t)}|}{|\theta^{(t)}|} < \varepsilon.$$

  Nos casos em que θ fica instável quando θ^t está perto de 0, usamos

$$\frac{|\theta^{(t+1)} - \theta^{(t)}|}{|\theta^{(t)}| + \varepsilon} < \varepsilon.$$

Método da Bisseção

Este método é um algoritmo que procura a raíz da função ($S(\theta) = 0$), caso as possamos encontrar e a função seja contínua. Usa como base o Teorema de Bolzano, com um intervalo]a,b[, cujo o qual vamos determinar em qual metade do intervalo a raíz está localizada. Faz isto iterativamente até ao critério de paragem.

Passos do algoritmo:

- Encontrar [a1,b1], com S contínuo e $S(a)S(b) < 0$
- $\theta^{(1)} = (a_1 + b_1)/2$ se $S(\theta^{(1)}) \neq 0$ (caso contrário, encontrámos a solução) $[a_2, b_2] = [a_1, \theta^{(1)}]$ se $S(a)S(\theta^{(1)}) < 0$ ou $[a_2, b_2] = [\theta^{(1)}, b_1]$ caso contrário
- Repetimos o passo anterior até $S(\theta^{(t)}) = 0$ ou a regra de paragem for validada

Newton-Raphson

O método de Newton-Raphson é um algoritmo baseado na expansão de Taylor para procurar as raízes da função.

- Seja $S \in C^2[a, b]$. Seja $\theta^{(0)} \in [a, b]$ uma aproximação da raíz de S (θ^*), tal que $S'(\theta^{(0)}) \neq 0$ e $|\theta^* - \theta^{(0)}|$ um valor pequeno
- Consideramos a expansão de Taylor de S em ordem 2 à volta de $\theta^{(0)}$

  $$S(\theta^*) \approx S(\theta^{(0)}) + (\theta^* - \theta^{(0)})S'(\theta^{(0)})$$

  que se torna em

  $$0 \approx S(\theta^{(0)}) + (\theta^* - \theta^{(0)})S'(\theta^{(0)})$$

  porque θ^* é a raíz de S
- Resolvendo a equação, teremos

  $$\theta^{(t+1)} = \theta^{(t)} - \frac{S(\theta^{(t)})}{S'(\theta^{(t)})}$$

  que converge para θ^* quando outras condições são verdadeiras (como por exemplo $|\theta^{(t+1)} - \theta^{(t)}| > \varepsilon$)

Fisher Scoring

O método de Fisher Scoring é uma variação do NR com objetivo de acelerar o procedimento da convergência.

- $-S'(\theta)$ é substituído pela Matriz Informação de Fisher $I(\theta)$
- a equação de atualização passa a ser:

  $$\theta^{(t+1)} = \theta^{(t)} + I(\theta^{(t)})^{-1}S(\theta^{(t)})$$

Newton-Raphson vs Fisher Scoring

- A Informação de Fisher pode ser difícil de deduzir
- O Fisher Scoring pode ser computacionalmente mais rápido
- Normalmente, o NR tem melhor performance perto do máximo
- O Fisher Scoring é menos dependente em valores específicos de dados

Exercício 4

4.a)

Derive the likelihood, log-likelihood and score functions (simplify the expressions as much as possible). Derive both the maximum likelihood estimator (MLE) and method of moments estimator (MME) of α and use them to estimate α. Why are ML estimators so attractive?

Seja $X \sim Pareto(1, \alpha)$ que tem como p.d.f

$$f(x; \alpha) = \frac{\alpha}{x^{\alpha+1}}, \quad x > 0 \quad x \geq 1.$$

A função de verosimilhança L é dada por:

$$L(\alpha) = \prod_{i=1}^{n} f(x_i, \alpha) = \prod_{i=1}^{n} \frac{\alpha}{x_i^{\alpha+1}} = \prod_{i=1}^{n} \alpha \frac{1}{x_i^{\alpha+1}} = \alpha^n \prod_{i=1}^{n} \frac{1}{x_i^{\alpha+1}}$$

Ao aplicarmos o logaritmo a esta função temos a log-verosimilhança l:

$$\begin{aligned}
l(\alpha) &= log\left(\alpha^n \prod_{i=1}^{n} \frac{1}{x_i^{\alpha+1}}\right) = \\
&= log(\alpha^n) + log\left(\prod_{i=1}^{n} \frac{1}{x_i^{\alpha+1}}\right) = \\
&= nlog(\alpha) + \sum_{i=1}^{n} log\left(\frac{1}{x_i^{\alpha+1}}\right) = \\
&= nlog(\alpha) + \sum_{i=1}^{n} log\left((x_i^{\alpha+1})^{-1}\right) = \\
&= nlog(\alpha) - \sum_{i=1}^{n} log(x_i^{\alpha+1}) = \\
&= nlog(\alpha) - (\alpha+1)\sum_{i=1}^{n} log(x_i)
\end{aligned}$$

Finalmente, a função score s é dada pela derivada parcial da função de log-verosimilhança:

$$\frac{\partial}{\partial\alpha} l(\alpha) = l'(\alpha) = \left(nlog(\alpha) - (\alpha+1)\sum_{i=1}^{n} log(x_i)\right)' = \frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i)$$

Agora, para termos o MLE (*maximum likelihood estimator*), igualamos a função score a 0 e no final comprovamos, com o sinal da segunda derivada da função de log-verosimilhança, se o candidato a estimador é de facto um estimador de máxima verosimilhança de α.

$$\hat{\alpha}_{MLE} = \underset{\alpha}{arg\max}\ L(\alpha) \Leftrightarrow l'(\alpha) = 0 \Leftrightarrow s(\alpha) = 0 \Leftrightarrow \frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i) = 0 \Leftrightarrow \frac{n}{\alpha} = \sum_{i=1}^{n} log(x_i) \Leftrightarrow \frac{n}{\sum_{i=1}^{n} log(x_i)} = \alpha$$

Verificando o sinal da primeira derivada da função score (segunda derivada da log-verosimilhança):

$$l''(\alpha) = s'(\alpha) = (\frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i))' = \left(\frac{n}{\alpha}\right)' - \left(\sum_{i=1}^{n} log(x_i)\right)' = -\frac{n}{\alpha^2} - 0 = -\frac{n}{\alpha^2} \quad < 0,\ \forall\alpha$$

Podemos confirmar que o candidato $\hat{\alpha}_{MLE} = \frac{n}{\sum_{i=1}^{n} log(x_i)}$ é o estimador de máxima verosimilhança de α e a estimativa obtida foi de 2.814.

```
vector4 = c(1.977866, 1.836622, 1.097168, 1.232889, 1.229526, 2.438342, 1.551389,
    1.300618, 1.068584, 1.183466, 2.179033, 1.535904, 1.3235, 1.458713, 1.013755,
    3.602314, 1.087067, 1.014013, 1.613929, 2.792161, 1.197081, 1.02143, 1.111531,
    1.131036, 1.064926)

n = length(vector4)

MLE = function(x) {

    res <- sum(log(x))

    return(n/res)
}
mleVar = MLE(vector4)
mleVar
```

```
## [1] 2.814179
```

Vamos agora determinar o estimador de α pelo método dos momentos. Neste sentido, tem-se

$$E(X) = \overline{X} \quad \iff \quad \frac{\alpha}{\alpha - 1} = \overline{X} \quad \iff \quad \alpha = \frac{\overline{X}}{\overline{X} - 1}$$

Note-se que se $X \sim Pareto(x_0, \alpha)$, então

$$E(X^k) = \frac{\alpha}{\alpha - k} x_0^k,$$

se $k < \alpha$ (ver [3], pag. 306).

Assim, a estimativa de α é dada por

```
MME = function(x){

  x_bar = mean(x)
  alpha = x_bar/(x_bar-1)

  return(alpha)
}
mmeVar=MME(vector4)
mmeVar
```

```
## [1] 2.913822
```

Os estimadores de máxima verosimilhança representam uma abordagem para problemas de estimação de parâmetros. Assim, pode ser aplicado nas situações que consistam neste problema. Estes têm propriedades que os tornam interessantes de usar ao invés de outros métodos de estimação. Mais especificamente, tornam-se estimadores de variância mínima não enviesados à medida que o tamanho da amostra aumenta. Não enviesados neste caso significa que se retirarmos um elevado número de amostras aleatórias com reposição de uma população, o valor médio do parâmetro estimado será, teoricamente, exatamente igual ao valor "real" da população. Relativamente à variância mínima, esta representa que o estimador terá uma variância muito baixa e por conseguinte, um intervalo de confiança mais "estreito" comparativamente ao outro tipo de estimadores. Estes também seguem distribuições aproximadamente normais que podem ser usadas para calcular intervalos de confiança e testes de hipóteses para os parâmetros. Por serem estimadores já bastante estudados e analisados, a maioria dos softwares de estatística e análise de dados já contêm as versões otimizadas destes algoritmos para que a complexidade computacional exigida seja a mínima possível.

4.b)

Noting that the Pareto distribution belongs to the exponential family, derive the Fisher information $I_n(\alpha)$. Use the Fisher information to estimate the variance of the MLE.

Uma vez que a distribuição de Pareto pertence a uma família exponencial, temos que $I_n(\theta)$ é dado pela expressão referida na parte teórica da secção.

Assim, pela alínea a), tem-se

$$I_n(\alpha) = -\mathbb{E}(\ell''(\alpha)) = \ = -\mathbb{E}(\frac{\partial}{\partial \alpha}\left[\frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i)\right]) = \ = -\mathbb{E}(-\frac{n}{\alpha^2}) = \ = \frac{n}{\alpha^2}$$

A variância do MLE é deduzida pela fórmula $Var(MLE) = \frac{1}{I_n(\alpha)}$, que no R fica:

```r
VAR = function(alpha, n){
  In=n/(alpha^2)
  return(1/In)
}
VAR(mleVar,n)
```

```
## [1] 0.3167842
```

que fica (com $\alpha = 2.814179$),

$$\frac{1}{\frac{n}{\alpha^2}} = 0.3167842$$

Assume herein that it was not possible to derive the MLE of α.

4.c)

Display graphically (side-by-side) the likelihood, log-likelihood and score functions in order to locate the ML estimate of α. Indicate an interval that contains the ML estimate.

De modo a representar graficamente as funções de verosimilhança, log-verosimilhança e score derivadas acima, iremos declará-las no R.

```r
# Função de verosimilhança
lik <- function(alpha){
  res = vector()
  for(a in alpha) {
    res = c(res, (a^n)*prod(1/(vector4^(a+1))))
  }
  return(res)
}

# Função log-verosimilhança
loglik <- function(alpha){
  res <- vector()
  for(a in alpha) {
    res <- c(res, n*log(a)-((a + 1) * sum(log(vector4))))
  }
  return(res)
}

# Função score
funScore <- function(alpha) {
  res <- vector()
  for(a in alpha) {
    res <- c(res, (n/a) - sum(log(vector4)))
  }
  return(res)
}
```

De seguida, representaremos então as 3 funções lado a lado.

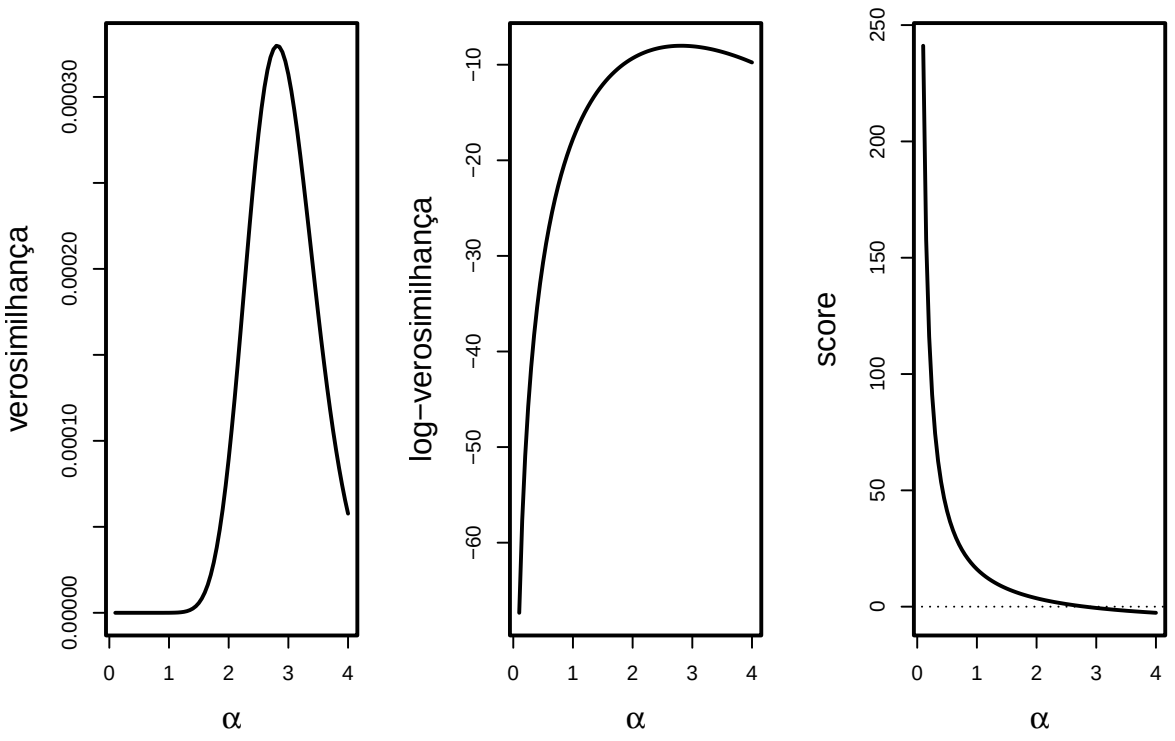
```r
#Representação gráfica
par(mfrow=c(1,3))
b <- seq(0.1,4,0.05)

plot(b,lik(b),ylab="verosimilhança", xlab=expression(alpha),lwd=2,type="l",
     cex.lab=1.5)
```

```
box(lwd=2)

plot(b,loglik(b),ylab="log-verosimilhança",xlab=expression(alpha),lwd=2,
     type="l",cex.lab=1.5)
box(lwd=2)

plot(b,funScore(b),ylab="score", xlab=expression(alpha),lwd=2,type="l",cex.lab=1.5)
abline(h=0,lty=3); box(lwd=2)
```



Vendo agora o traçado das funções podemos localizar que o seu máximo se encontra aproximadamente em 2.8. Um intervalo que contém a estimativa de máxima verosimilhança é o [1,4]. Calcular o máximo da função de verosimilhança é equivalente a calcular o máximo da função log-verosimilhança. Sabendo que esta tem a sua concavidade para baixo, vai ter o ponto máximo onde sua derivada é igual a zero (função score).

4.d)

Use the R function maxLik() from library maxLik to approximate the ML estimate of α. Feed maxLik() with the initial estimate of α given by the method of moments.

Para obtermos uma estimativa mais precisa para o valor de α iremos usar a função do R *maxLik()*. Ao aplicar este método à função de log-verosimilhança, usando o ponto máximo calculado de 2.8 como o ponto inicial, temos que o valor resultante é de 2.814. Como podemos verificar, o cálculo feito analiticamente para estimar o valor de *alpha* na alínea a) deu o mesmo valor, logo foi bem calculado.

```
library(maxLik)
```

```
## Loading required package: miscTools
```

```
##
```

```
## Please cite the 'maxLik' package as:
## Henningsen, Arne and Toomet, Ott (2011). maxLik: A package for maximum likelihood estimation in R
##
## If you have questions, suggestions, or comments regarding the 'maxLik' package, please use a forum
## https://r-forge.r-project.org/projects/maxlik/
```

```
#dar como start a estimativa inicial de alpha dado pelo metodo dos momentos
#start = maximo da funcao loglikelihood
maxLik(loglik, start = 2.8)
```

```
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 2 iterations
## Return code 1: gradient close to zero (gradtol)
## Log-Likelihood: -8.016816 (1 free parameter(s))
## Estimate(s): 2.814179
```

```
#Assim, a estimativa de max verosimilhanca de alpha obtida usando a funcao do R foi 2.81
```

4.e)

Describe and discuss in detail the algorithms of bissection, Newton-Raphson and secant that enable, in particular, the approximation of the ML estimate of α. Implement those in R and use them and the sample above to estimate α - report all the iterations together with the error. Justify your choice of the initial estimates for each method and discuss the results.

Note: the secant method is a variation of the method of Newton-Raphson. It considers the update equations

$$\theta^{(t+1)} = \theta^{(t)} - S(\theta^{(t)})\frac{\theta^{(t)} - \theta^{(t-1)}}{S(\theta^{(t)}) - S(\theta^{(t-1)})}$$

where S is the score function whose zero we want to approximate. In particular, this method needs two initial estimates, which need to be carefully addressed in order for the method to converge.

O primeiro algoritmo de aproximação do MLE de α é a bisseção. Tal como explicado na teoria, pegamos num intervalo [a,b] onde a e b são simétricos, para podermos aplicar o teorema de Bolzano. Pelo gráfico do Score apresentado na alínea c), é fácil perceber que $f(2)f(4) < 0$, logo escolhemos o intervalo [2,4]. A função em R fica a seguinte:

```
bisection <- function(x,a,b,eps){
#x: amostras
#a: limite esquerdo do intervalo para o teorema de Bolzano
#b: limite direito do intervalo para o teorema de Bolzano
#eps: valor para regra de paragem
  alpha.it = vector(); alpha.it[1] = (a+b)/2
  k = 1; diff = 1
  diffs= vector()
  while(diff>eps){
    #se a multiplicação for < 0, diminuir b
    if(funScore(alpha.it[k])*funScore(a)<0){
      b = alpha.it[k]
      alpha.it[k+1] = (a+b)/2
    }
    #se a multiplicação for > 0, aumentar a
    else{if(funScore(alpha.it[k])*funScore(a)>0){
      a = alpha.it[k]
```

```
      alpha.it[k+1] = (a+b)/2
    #se a multiplicação for = 0, chegamos ao valor e a regra de paragem é ativada
    }else{alpha.it[k+1]=alpha.it[k]}
    }
    #atualização da diferença de iterações para a regra de paragem
    diff = abs(alpha.it[k+1]-alpha.it[k])
    diffs[k]=diff
    k = k+1
  }
  result = as.matrix(alpha.it)
  colnames(result)<-"iterations"
  rownames(result)<-1:length(alpha.it)
  retList<-list("res"=result,"error"=diffs)
  retList
}
#resultado da bisection
l1=bisection(vector4,2,4,0.000001)
m1=t(l1$res)
m1
```

```
##            1   2    3     4      5       6        7        8        9       10
## iterations 3 2.5 2.75 2.875 2.8125 2.84375 2.828125 2.820312 2.816406 2.814453
##                  11       12       13       14       15       16       17
## iterations 2.813477 2.813965 2.814209 2.814087 2.814148 2.814178 2.814194
##                  18       19       20       21
## iterations 2.814186 2.814182 2.81418 2.814179
```

Numa situação normal, para calcular o erro de um método, usaríamos a fórmula:

$$\frac{|valorAproximado - valorExato|}{valorExato} * 100$$

Mas, tendo em conta que nestes métodos estamos a assumir que não sabemos o *valorExato* do MLE, iremos demonstrar a diferença entre as várias iterações para cada método. O código seguinte demonstra esses resultados:

```
l1$error
```

```
##  [1] 5.000000e-01 2.500000e-01 1.250000e-01 6.250000e-02 3.125000e-02
##  [6] 1.562500e-02 7.812500e-03 3.906250e-03 1.953125e-03 9.765625e-04
## [11] 4.882812e-04 2.441406e-04 1.220703e-04 6.103516e-05 3.051758e-05
## [16] 1.525879e-05 7.629395e-06 3.814697e-06 1.907349e-06 9.536743e-07
```

Para o segundo algoritmo, Newton-Raphson, precisamos da primeira derivada da função score, já calculada na alínea a).

Considerando a sucessão

$$\alpha^{(t+1)} = \alpha^{(t)} \frac{S(alpha^{(t)})}{S(alpha^{(t)})},$$

podemos observar que esta converge para α^*.

O α^0 usado neste algoritmo é o MME, que foi calculado na primeira alínea.

```
prime <- function(alpha){
  out = numeric(length(alpha))
  if(length(alpha)==1){out =- n/alpha^2}
  if(length(alpha)!=1){
    for(i in 1:length(alpha)){out[i] = - n/alpha[i]^2}
```

```
  }
  return(out)
}

#método Newton-Raphson
NR <- function(x,alpha0,eps){
#x: amostras
#alpha0: primeiro valor do vetor a iterar. i.e., o MME
#eps: valor para regra de paragem
  alpha.it = vector()
  alpha.it[1] = alpha0
  k = 1
  diff = 1
  diffs=vector()
  #iterações
  while(diff>eps){
    alpha.it[k+1] = alpha.it[k]-funScore(alpha.it[k])/prime(alpha.it[k])
    diff = abs(alpha.it[k+1]-alpha.it[k])
    diffs[k]=diff
    k = k+1
  }
  result = as.matrix(alpha.it)
  colnames(result)<-"iterations"
  rownames(result)<-1:length(alpha.it)
  result
  retList<-list("res"=result,"error"=diffs)
  retList
}
#resultado do NR
l2=NR(vector4,mmeVar,0.000001)
m2=t(l2$res)
m2
```

```
##                   1        2        3        4        5
## iterations 2.913822 2.810651 2.814175 2.814179 2.814179
```

Com erro:

```
l2$error
```

```
## [1] 1.031711e-01 3.523681e-03 4.423136e-06 6.951772e-12
```

O algoritmo *Secant* é semelhante ao NR, mas neste caso usamos a sequência dada no enunciado, e precisamos de um intervalo de valores próximo do valor ótimo. Usando o intervalo [2,4] conseguimos obter um resultado semelhante (se não igual) aos outros métodos de aproximação.

```
secant <- function(x, alpha0, alpha1, eps) {
    # x: amostras alpha0: limite esquerdo do intervalo alpha1: limite direito
    # do intervalo eps: valor para regra de paragem
    alpha.it = vector()
    alpha.it[1] = alpha0
    alpha.it[2] = alpha1
    k = 2
    diff = 1
    diffs = vector()
    diffs[1] = diff
```

```
    # iterações
    while (diff > eps) {
        alpha.it[k + 1] = alpha.it[k] - funScore(alpha.it[k]) * ((alpha.it[k] - alpha.it[k -
            1])/(funScore(alpha.it[k]) - funScore(alpha.it[k - 1])))
        diff = abs(alpha.it[k + 1] - alpha.it[k])
        diffs[k] = diff
        k = k + 1
    }
    result = as.matrix(alpha.it)
    colnames(result) <- "iterations"
    rownames(result) <- 1:length(alpha.it)
    retList <- list(res = result, error = diffs)
    retList
}
# resultado da secant
l3 = secant(vector4, 2, 4, 1e-06)
m3 = t(l3$res)
```

Com erro:

```
l3$error
```

```
## [1] 1.000000e+00 8.427469e-01 4.876357e-01 1.621855e-01 1.671811e-02
## [6] 9.109698e-04 5.671144e-06 1.823774e-09
```

4.f)

Describe and discuss in detail the Fisher scoring method. Show, analytically, that the methods of Newton-Raphson and Fisher scoring coincide in this particular case. Implement it in R and use it and the sample above to estimate α - report all the iterations together with the error. Justify your choice of the initial estimates.

Para o Fisher Scoring, precisamos de calcular $I(\alpha)$:

$$I(\alpha) = -E(l''(\alpha)) = -E(-\frac{n}{\alpha^2}) = \frac{n}{\alpha^2}$$

O α^0 usado neste algoritmo, tal como no NR, é o MME. Com o uso da equação para o Fisher Scoring, temos em R:

```
#Fórmula do I(alpha)
Ialpha<-function(alpha){
  n/(alpha^2)
}

#método Fisher Scoring
fisherScore <- function(x,alpha0,eps){
#x: amostras
#alpha0: primeiro valor para o vetor a iterar, i.e., MME
#eps: valor para regra de paragem
  alpha.it = vector()
  alpha.it[1] = alpha0
  k = 1
  diff = 1
  diffs=vector()
  diffs[1]=diff
  #iterações
  while(diff>eps){
```

```
    alpha.it[k+1] = alpha.it[k]+(1/Ialpha(alpha.it[k]))*funScore(alpha.it[k])
    diff = abs(alpha.it[k+1]-alpha.it[k])
    k = k+1
    diffs[k]=diff
  }
  result = as.matrix(alpha.it)
  colnames(result)<-"iterations"
  rownames(result)<-1:length(alpha.it)
  retList<-list("res"=result,"error"=diffs)
  retList
}
#resultado do fisher score
l4=fisherScore(vector4,mmeVar,0.000001)
m4=t(l4$res)
m4
```

```
##                   1        2        3        4        5
## iterations 2.913822 2.810651 2.814175 2.814179 2.814179
```

Com erro:

```
l4$error
```

```
## [1] 1.000000e+00 1.031711e-01 3.523681e-03 4.423136e-06 6.951772e-12
```

Referências

[1] Ross, S. (2010). *A first course in probability.* 8-th Edition. Prentice Hall.

[2] Lourenço, V. M. (2021). *Computational Numerical Statistics, slides week5,6,7.* First Semester - FCT-UNL

[3] Gonçalves,E., Lopes, N.M. (2003). *ESTATÍSTICA - Teoria Matemática e Aplicações.* Escolar Editora 2003.

[4] Jung, K., Lee, J., Gupta, V., Cho R. (2019, 11 de outubro). *Comparison of Bootstrap Confidence Interval Methods for GSCA Using a Monte Carlo Simulation.* Frontiers.
Acedido a 07/12/2021, em https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02215/full

[5] Efron, B. (1987) *Better bootstrap confidence intervals (with Discussion).* Journal of the American Statistical Association, 82, 171–200.

[6] YouTube (2014, 10 de maio). *Asymptotic distribution of the maximum likelihood estimator(mle) - finding Fisher information*, Phil Chan.
Acedido a 07/12/2021, em https://www.youtube.com/watch?v=DyLWP5Zx934&ab_channel=PhilChan

[7] Peterson, J.L. *Estimating the Parameters of a Pareto Distribution.*
Acedido a 07/12/2021, em https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.966&rep=rep1&type=pdf

[8] Gomes, J. (2011). *Regressao Linear.*
Acedido a 12/12/2021, em https://moodle-arquivo.ciencias.ulisboa.pt/1415/pluginfile.php/101729/mod_resource/content/0/2011_Regress%C3%A3o%20Linear.pdf

[9] Brownstone, D., Valletta, R. (2001). *The Bootstrap and Multiple Imputations: Harnessing Increased Computing Power for Improved Statistical Tests.* Journal of Economic Perspective. Volume 15, Number 4.
Acedido a 12/12/2021, em https://pubs.aeaweb.org/doi/pdfplus/10.1257/jep.15.4.129

[10] Flachaire, E. (2007, 1 de outubro). *Bootstrapping heteroskedastic regression models: wild bootstrap vs. pairs bootstrap.*
Acedido a 12/12/2021, em https://halshs.archives-ouvertes.fr/halshs-00175910/document