Regressão Linear

Rafael Tenfen

Data de entrega: 28/05/2021

Descrição da atividade

O objetivo desta atividade é aplicar as técnicas de regressão linear. A atividade é dividida em três partes:

1. Adivinhe a correlação
2. Análise da relação entre variáveis
3. Análise de regressão com dados experimentais

Algumas recomendações:

- Se você não estiver habituado com R Markdown, acostume-se a processar com frequência o documento, usando o botão **Knit**. Isso permitirá que eventuais erros no documento ou no código R sejam identificados rapidamente, pouco depois de terem sido cometidos, o que facilitará sua correção. Na verdade, é uma boa ideia você fazer isso **agora**, para garantir que seu ambiente esteja configurado corretamente. Se você receber uma mensagem de erro do tipo *Error in library(foo)*, isso significa que o pacote `foo` não está instalado. Para instalar um pacote, execute o comando `install.packages("foo")` no Console, ou clique em *Tools -> Install Packages*.
- Após concluir a atividade, você deverá submeter no Moodle um arquivo ZIP contendo:
  - o arquivo fonte .Rmd;
  - a saída processada (PDF ou HTML) do arquivo .Rmd;
  - a imagem da Parte 1 e os arquivos de dados referentes à Parte 3, que são necessários para o processamento do .Rmd.

Configuração

Nesta atividade, nenhuma configuração é necessária por padrão, mas você pode usar o bloco abaixo para incluir o que julgar necessário.

```
# insira seus comandos de configuração aqui
```

Parte 1: Adivinhe a Correlação

Nesta parte você irá exercitar a sua familiaridade com o coeficiente de correlação de Pearson, que é uma medida da correlação linear entre duas variáveis. Para isso, basta jogar *Guess the Correlation*, um jogo online em que você faz pontos ao adivinhar de forma aproximada o coeficiente de correlação para um dado gráfico de dispersão. O jogo está disponível em `http://guessthecorrelation.com/`. Tire uma selfie onde

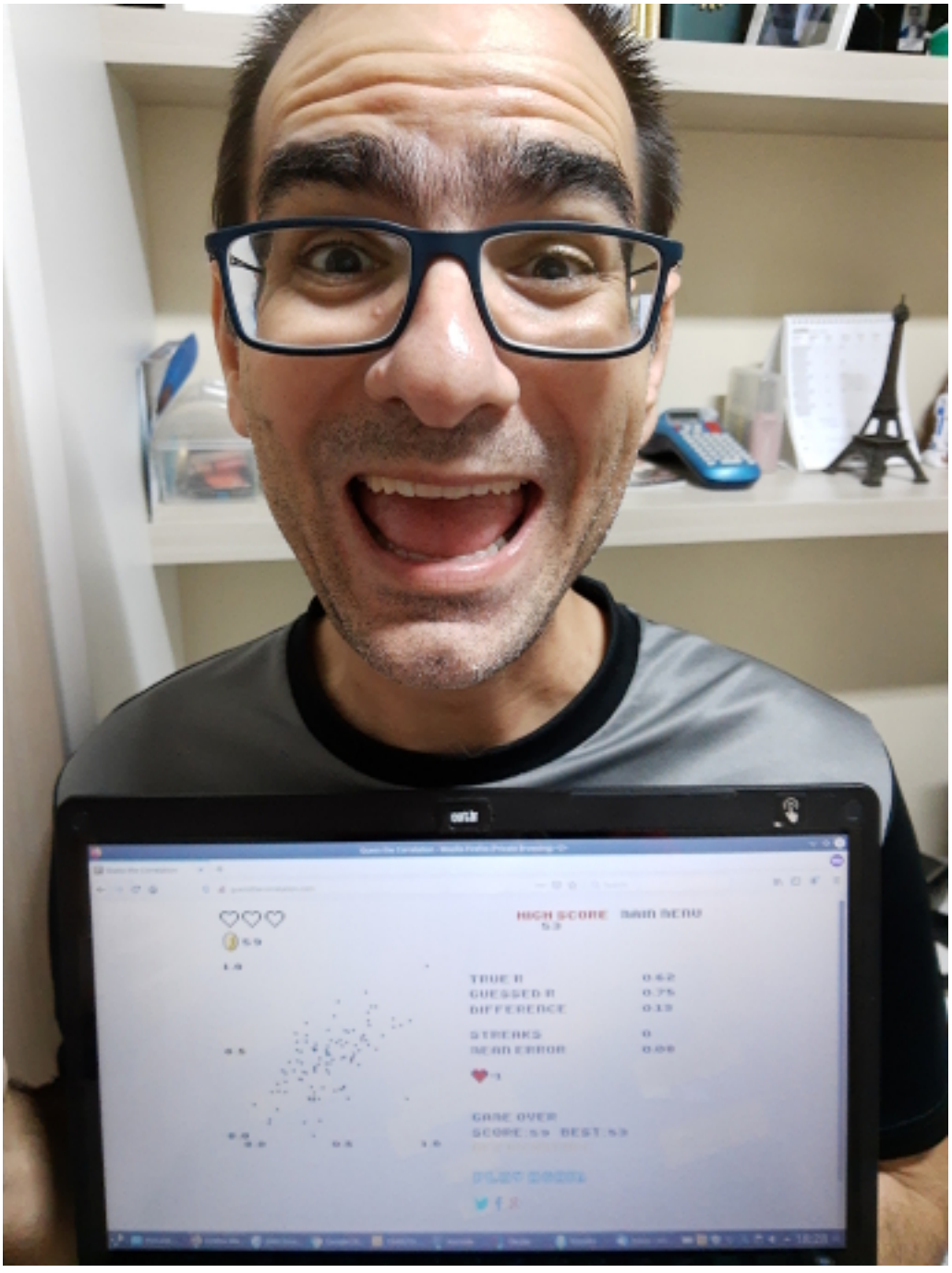
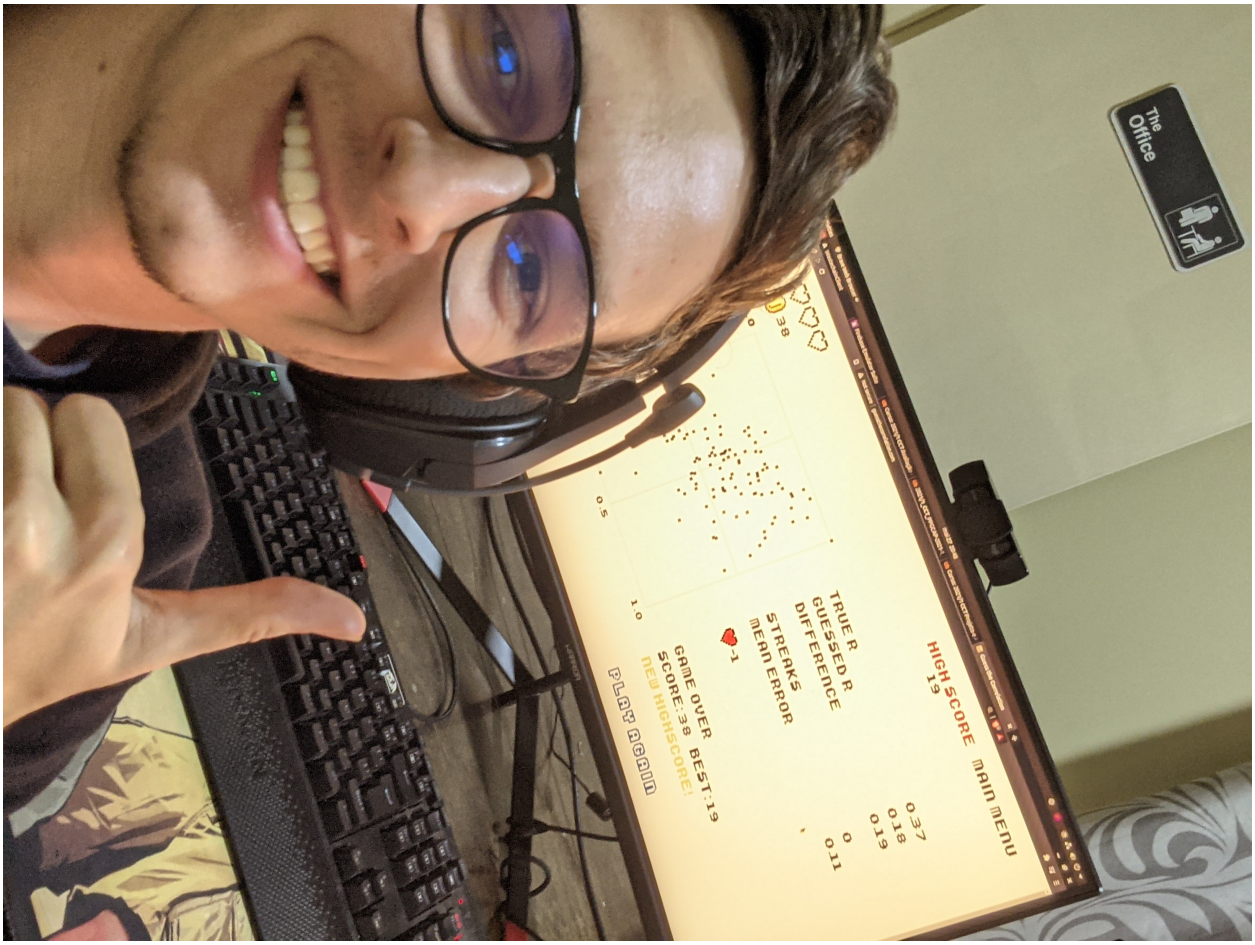


Figure 1: Guess the Correlation

apareçam **seu rosto e seu resultado** no jogo (**só vale escore de 25 ou mais!**), e insira-a no lugar do meu resultado de 59 pontos abaixo:

38pts! 

Parte 2: Análise da Relação entre Variáveis

Uma etapa preliminar em uma análise de regressão é estabelecer se as variáveis consideradas efetivamente possuem a relação esperada pelo modelo de regressão. Para isso, utiliza-se o ferramental da análise exploratória de dados.

O arquivo `variaveis.dat` possui cinco pares de variáveis (`x1-y1`, `x2-y2`, ..., `x5-y5`). Nesta parte da atividade, você deve determinar para quais desses pares parece possível construir um modelo de regressão linear. Use o bloco R abaixo para realizar as análises de dados, e nas respostas indique, **para cada par x-y**, se seria viável uma regressão ou não, **justificando sua resposta**. Algumas observações:

- **não é necessário construir o modelo**, apenas realizar a análise exploratória dos dados;
- em cada par, considere sempre y como variável de resposta e x como variável preditora.

Análise e respostas

```r
pares <- read.table("variaveis.dat", header = TRUE)
```

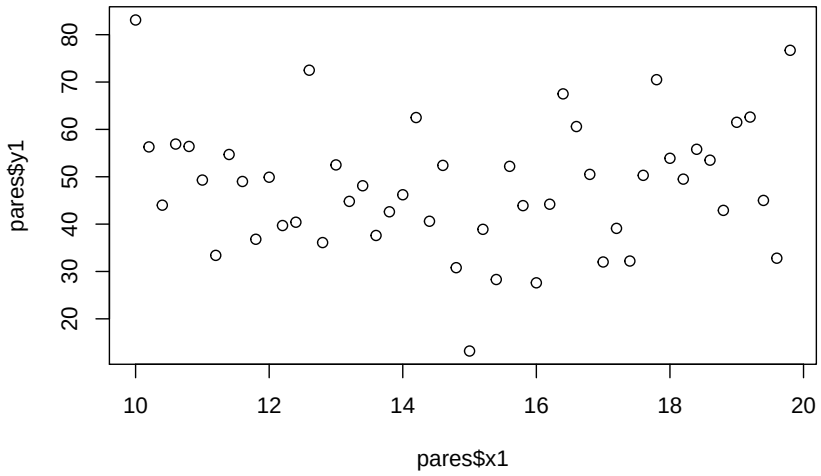
Respostas aqui

Parte 2 - Relacionamento entre os pares

- Adotado 95% de nível de confiança

x1 - y1

```
plot(pares$x1, pares$y1)
```




```
(cor.x1y1 = cor(pares$x1, pares$y1))
```

```
## [1] 0.01115994
```

```
(test.x1y1 = cor.test(pares$x1, pares$y1))
```

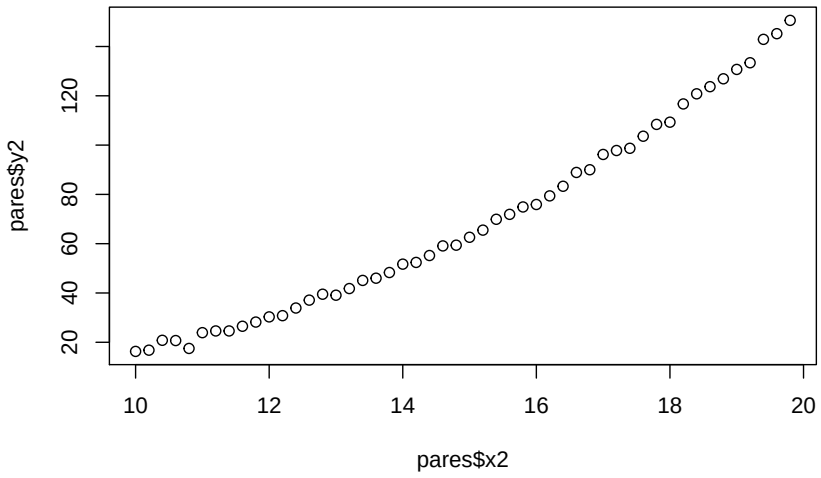
```
##
##  Pearson's product-moment correlation
##
## data:  pares$x1 and pares$y1
## t = 0.077323, df = 48, p-value = 0.9387
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2680203  0.2886111
## sample estimates:
##        cor
## 0.01115994
```

```
#summary(test.x1y1)
```

- **Resposta x1 - y1** A correlação entre os dados é significativamente baixa 0.011. Além da análise dos dados apresentar alta dispersão e o p-value 0.939 é maior que um alfa de 0.05 para um nível de confiaça de 95%, assim não seria viável construir um modelo de regressão.

x2 - y2

```
plot(pares$x2, pares$y2)
```




```
(cor.x2y2 = cor(pares$x2, pares$y2))
```

```
## [1] 0.9857316
```

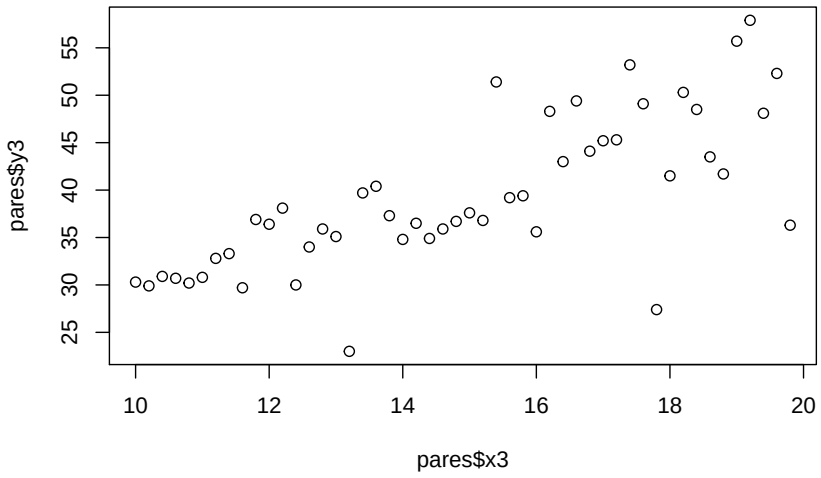
```
(test.x2y2 = cor.test(pares$x2, pares$y2))
```

```
##
##  Pearson's product-moment correlation
##
## data:  pares$x2 and pares$y2
## t = 40.573, df = 48, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9748631 0.9919201
## sample estimates:
##       cor
## 0.9857316
```

- **Resposta x2 - y2** A correlação entre os dados é significativamente alta 0.986. Além da análise exploratória apresentar baixa dispersão entre os dados e o p-value 0 é menor que o alfa 0.05 para um nível de confiaça de 95%, assim possuindo um grau significativo elevado assim sendo viável construir um modelo de regressão.

x3 - y3

```
plot(pares$x3, pares$y3)
```




```
(cor.x3y3 = cor(pares$x3, pares$y3))
```

```
## [1] 0.7428404
```

```
(test.x3y3 = cor.test(pares$x3, pares$y3))
```

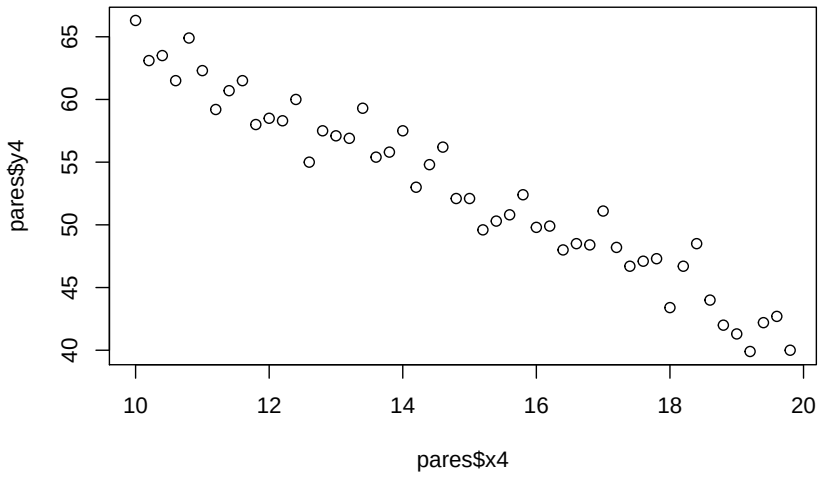
```
##
##  Pearson's product-moment correlation
##
## data:  pares$x3 and pares$y3
## t = 7.6875, df = 48, p-value = 6.553e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5855698 0.8462175
## sample estimates:
##       cor
## 0.7428404
```

```
#sd(pares$x3)
#sd(pares$y3)
```

- **Resposta x3 - y3** A correlação entre os dados é relativamente alta 0.743, não é aquilo tudo mas da pra considerar correlaçao de alguma forma entre os dados. A análise exploratória dos dados apresenta uma dispersão aceitável e o p-value 0 é menor que o alfa 0.05 para um nível de confiaça de 95%. Assim, acredito ser viável construir um modelo de regressão.

x4 - y4

```
plot(pares$x4, pares$y4)
```




```
(cor.x4y4 = cor(pares$x4, pares$y4))
```

```
## [1] -0.972956
```

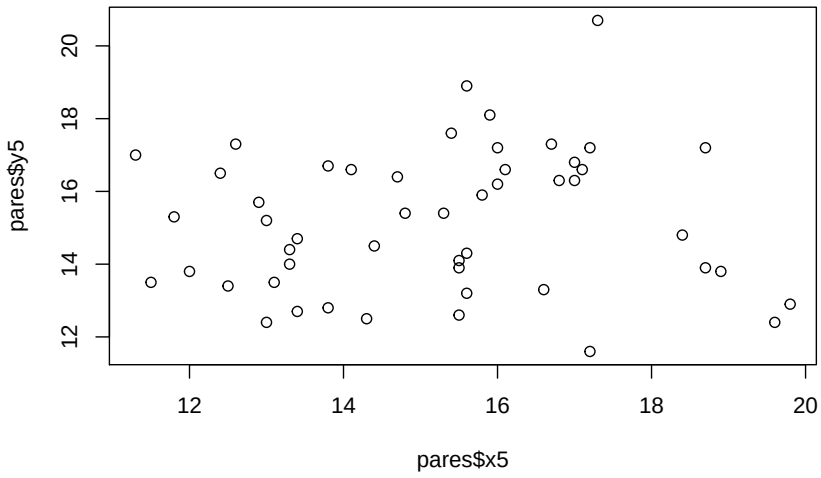
```
(test.x4y4 = cor.test(pares$x4, pares$y4))
```

```
##
##  Pearson's product-moment correlation
##
## data:  pares$x4 and pares$y4
## t = -29.182, df = 48, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9846427 -0.9525884
## sample estimates:
##       cor
## -0.972956
```

- **Resposta x4 - y4** A correlação entre os dados é significativamente alta -0.973. A análise entre os dados apresenta baixa dispersão e o p-value 0 é menor que o alfa 0.05 para um nível de confiaça de 95%, assim possuindo um grau significativo elevado e é viável construir um modelo de regressão.

x5 - y5

```
plot(pares$x5, pares$y5)
```




```
(cor.x5y5 = cor(pares$x5, pares$y5))
```

```
## [1] 0.07415827
```

```
(test.x5y5 = cor.test(pares$x5, pares$y5))
```

```
##
##  Pearson's product-moment correlation
##
## data:  pares$x5 and pares$y5
## t = 0.5152, df = 48, p-value = 0.6088
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2084931  0.3453768
## sample estimates:
##        cor
## 0.07415827
```

- **Resposta x5 - y5** A correlação entre os dados é baixa 0.074. A análise dos dados apresentar alta dispersão e o p-value 0.609 é maior que o alfa 0.05 para um nível de confiaça de 95%, assim não seria viável construir um modelo de regressão.

Parte 3: Análise de Regressão com Dados Experimentais

Para essa comparação iremos usar tempos de execução medidos pelo script Python `fibo.py`. Esse script mede o tempo necessário para encontrar N vezes o quinquagésimo termo da série de Fibonacci. O script varia o valor de N de acordo com três parâmetros na linha de comando: ~ - o valor inicial de N; - o valor final de N; - o passo de incremento.

O quarto parâmetro na linha de comando é quantas repetições são realizadas para cada valor testado; se o parâmetro não for especificado, cada valor de N é mensurado apenas uma vez. Por exemplo, o comando

```
python3 fibo.py 100 200 25 2
```

1200000 - 1000000 - 120000 executa o script com N valendo 100, 125, 150, 175 e 200, com duas repetições para cada valor de N.

O script pode ser executado no RStudio Cloud. Na janela inferior esquerda, normalmente usada para o console, há uma aba Terminal, na qual você pode executar comandos do Linux.

Os passos deste experimento são os seguintes:

1. Execute o script `fibo.py`. Os parâmetros devem ser ajustados a seu critério, respeitando as seguintes considerações:
   - o valor inicial de N deve produzir um tempo de execução superior a 0,5 s (em testes no RStudio Cloud, 100.000 mostrou-se um valor satisfatório, mas verifique);
   - o valor final deve ser pelo menos 10 vezes o valor inicial;
   - pelo menos oito valores distintos de N (incluindo o inicial e o final) devem ser usados;
   - devem ser realizadas pelo menos quatro medições para cada valor de N.
2. Especifique um modelo de regressão para o tempo de execução em função de N e ajuste-o aos dados coletados no passo 1. Qual a equação de regressão obtida?
3. Verifique a qualidade do ajuste e faça o diagnóstico dos resíduos do modelo. Determine se o modelo é adequado ou não.
4. Use o modelo para obter predições para quatro valores de N **diferentes** dos usados no script, sendo **dois valores dentro do intervalo** mensurado para N e **dois valores fora** desse intervalo (por exemplo, considerando o exemplo de execução do script, onde o intervalo era 100-200, os valores poderiam ser 51, 123, 171 e 500). Além das estimativas pontuais, obtenha também os intervalos de confiança de 95% para as predições.
5. Execute novamente o script para obter uma medida para cada um dos valores de N usados no item anterior. Calcule os erros $e_i = y_i - \hat{y}_i$ correspondentes a essas medidas, e verifique se as medidas estão dentro dos intervalos de confiança obtidos no item 4.
6. De acordo com o modelo, qual o valor de N correspondente a um tempo de execução de 23 segundos? Execute o script para obter o tempo de execução para esse valor de N, e determine o erro e_i dessa medida.

Lembre-se que os dados experimentais devem ser salvos em arquivos para que sua análise seja reproduzível. Para facilitar essa tarefa, o script já gera a saída em um formato apropriado; você pode redirecionar a saída do script para um arquivo (por exemplo, `python3 fibo.py 100 200 25 2 >fibo.dat`) ou simplesmente criar o arquivo de dados no próprio editor do RStudio Cloud (crie um novo arquivo texto e cole a saída do script). Para as execuções dos passos 5 e 6 você pode gerar um único arquivo de dados.

Análise e respostas

```r
# seu código R aqui
# python3 fibo.py 120000 1000000 5000 5 > part3.dat
# python3 fibo.py 90000 1300000 5000 5 > part3_12.dat
part3 <- read.table("part3.dat", header = TRUE)
#length(part3)
str(part3)
```

```
## 'data.frame':    885 obs. of  2 variables:
##  $ n    : int  120000 120000 120000 120000 120000 125000 125000 125000 125000 125000 ...
##  $ tempo: num  0.58 0.579 0.585 0.584 0.584 ...
```

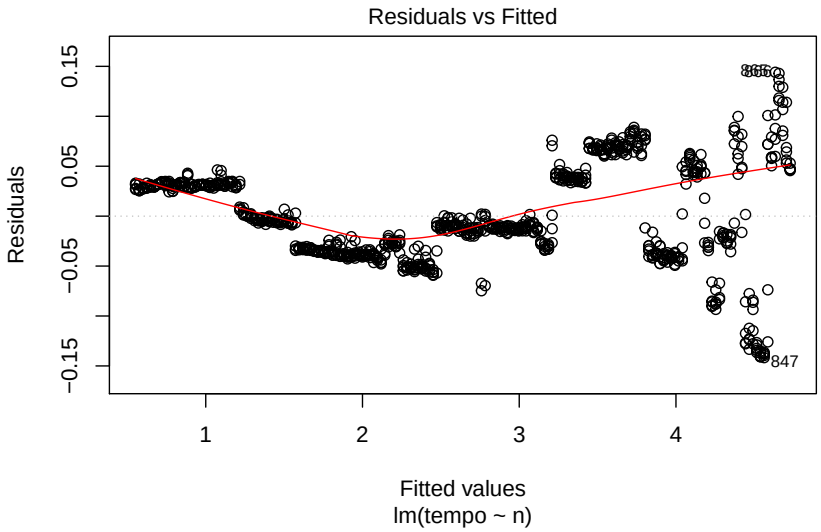
```
lmp3 = lm(tempo ~ n, data=part3)
lmp3
```

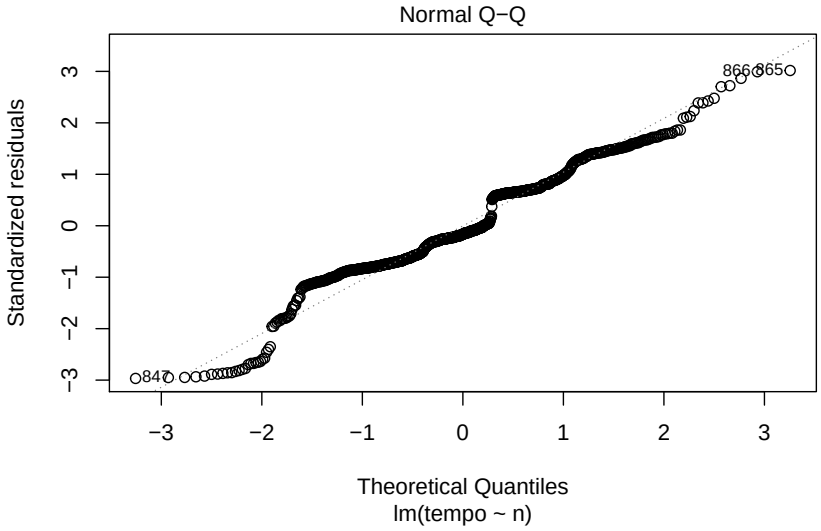
```
##
## Call:
## lm(formula = tempo ~ n, data = part3)
##
## Coefficients:
## (Intercept)            n
##  -1.712e-02    4.746e-06
```

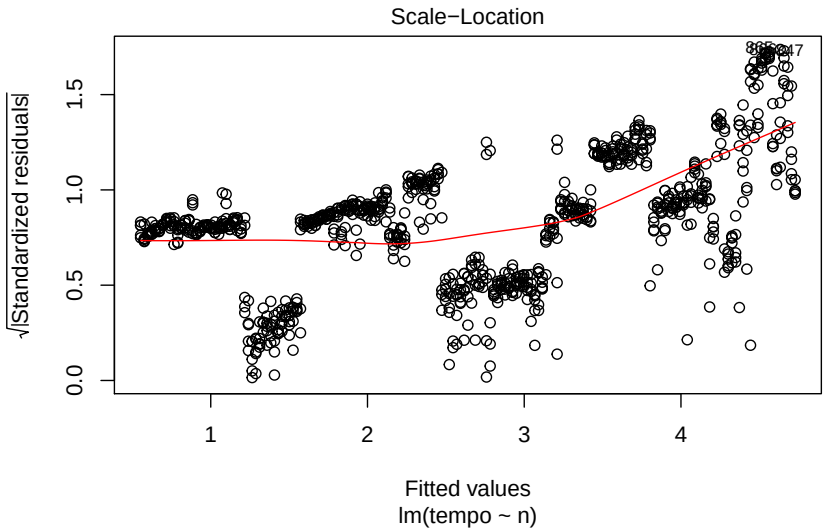
```
#str(lmp3)

(lmp3.summary = summary(lmp3))
```

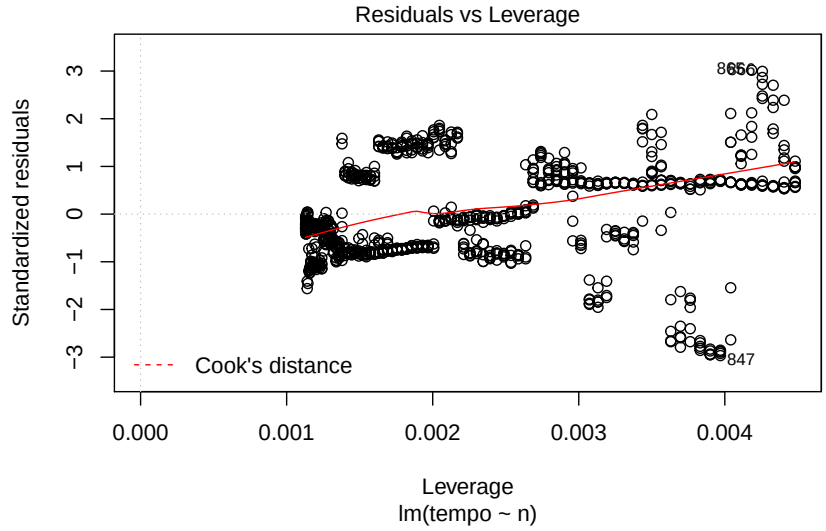
```
##
## Call:
## lm(formula = tempo ~ n, data = part3)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.141692 -0.033848 -0.007993  0.033530  0.144072
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.712e-02  3.875e-03  -4.417 1.12e-05 ***
## n            4.746e-06  6.296e-09 753.811  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04785 on 883 degrees of freedom
## Multiple R-squared:  0.9984, Adjusted R-squared:  0.9984
## F-statistic: 5.682e+05 on 1 and 883 DF,  p-value: < 2.2e-16
```

```
#str(lmp3.summary)

#par(mfrow=c(2,2));
plot(lmp3)
```

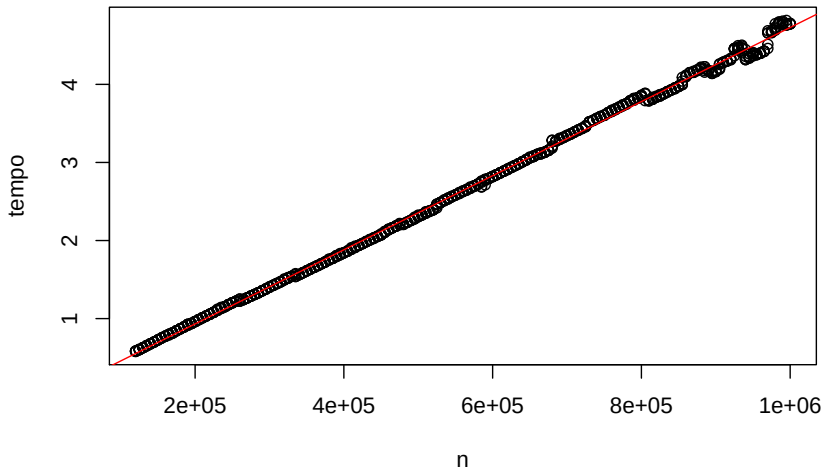

Residuals vs Fitted
Residuals
0.15
0.05
−0.05
−0.15
865
847
1
2
3
4
Fitted values
lm(tempo ~ n)


Normal Q−Q
Standardized residuals
3
2
1
0
−1
−2
−3
866
865
847
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(tempo ~ n)






```
plot(part3)
abline(lmp3, col='red')
```




```
diffResidxFitted = 100*resid(lmp3)/fitted(lmp3)
max(diffResidxFitted)
```

```
## [1] 5.976154
```

```
(lmp3.predict = predict(lmp3, newdata = data.frame(n=c(100000, 400000, 800000, 1200000)), conf.level=0.9
```

```
##         1         2         3         4
## 0.4574549 1.8811743 3.7794667 5.6777591
```

```
(tempo.400000 = part3$tempo[part3$n == 400000])
```

```
## [1] 1.84171 1.84331 1.84262 1.84096 1.84353
```

```
#tempo.400000.diff = diff(tempo.400000, lmp3.predict[2])
tempo.400000.diff = abs(tempo.400000 - lmp3.predict[2])
(tempo.800000 = part3$tempo[part3$n == 800000])
```

```
## [1] 3.84203 3.84017 3.85909 3.86171 3.85011
```

```
#tempo.800000.diff = diff(tempo.800000, lmp3.predict[3])
tempo.800000.diff = abs(tempo.800000 - lmp3.predict[3])

part312 <- read.table("part3_12.dat", header = TRUE)
(tempo.100000 = part312$tempo[part312$n == 100000])
```

```
## [1] 0.46315 0.46320 0.46335 0.46445 0.46355
```

```r
tempo.100000.diff = abs(tempo.100000 - lmp3.predict[1])
(tempo.1200000 = part312$tempo[part312$n == 1200000])
```

```
## [1] 5.56335 5.56422 5.56622 5.57196 5.56985
```

```r
tempo.1200000.diff = abs(tempo.1200000 - lmp3.predict[4])


(lmp3.predict.confinterval = lmp3.predict - (lmp3.predict * 0.95))
```

```
##          1          2          3          4
## 0.02287275 0.09405871 0.18897333 0.28388796
```

```r
lmp3.predict[4]
```

```
##        4
## 5.677759
```

```r
tempo.100000.diff < lmp3.predict.confinterval[1]
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```r
tempo.400000.diff < lmp3.predict.confinterval[2]
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```r
tempo.800000.diff < lmp3.predict.confinterval[3]
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```r
tempo.1200000.diff < lmp3.predict.confinterval[4]
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```r
(tempo.23s = (lmp3.summary$coefficients[1] - 23) / (-lmp3.summary$coefficients[2]))
```

```
## [1] 4850068
```

```r
predict(lmp3, newdata = data.frame(n=c(tempo.23s)), conf.level=0.95)
```

```
##  1
## 23
```

```r
tempo.23s
```

```
## [1] 4850068
```

```r
#tempo.23s =
part23s <- read.table("part3_4850068.dat", header = TRUE)
part23s
```

```
##         n    tempo
## 1 4850068 23.14015
## 2 4850068 23.35855
## 3 4850068 23.34468
## 4 4850068 23.35834
## 5 4850068 23.36985
```

```r
tempo.23s
```

```
## [1] 4850068
```

```r
(part23s.conf = (23 * 100) / part23s$tempo)
```

```
## [1] 99.39434 98.46502 98.52352 98.46590 98.41741
```

```r
(part23s.err = 100 - part23s.conf)
```

```
## [1] 0.6056573 1.5349840 1.4764820 1.5340987 1.5825947
```

Respostas aqui 1 - CLI: 120000 inicia em 0.54 e 1200000 termina em 5.4 - python3 fibo.py 120000 1000000 5000 5 > part3.dat

2 - Formula = tempo ~ n. Equaçao de regressão obtida: tempo = -0.0171 + 5×10^{-6} n

3 - O resíduo do modelo, está meio disperso como se fosse um cone pelo apresentado no gráfico, contudo ao mensurar a máxima diferença entre os resíduos e ajustes chega só a 5.98 %, e os resíduos na normal, ficou bacana como se fosse um normal ou no máximo 'light tailed' com pequenas mudanças nas caudas, o R-squared é bom e o modelo é significativo então meu veredito final é que o modelo é adequado.

- 4: N x Estimativa
- 100000 x 0.457s, Nível de Confiança 95%: 0.023s
- 400000 x 1.881s, Nível de Confiança 95%: 0.094s
- 800000 x 3.779s, Nível de Confiança 95%: 0.189s
- 1200000 x 5.678s, Nível de Confiança 95%: 0.284s
- Comparando as duas predições (400000, 800000) com as medicoes executadas
- 1.881 - 1.842, 1.843, 1.843, 1.841, 1.844 = 0.039, 0.038, 0.039, 0.04, 0.038
- 3.779 - 3.842, 3.84, 3.859, 3.862, 3.85 = 0.063, 0.061, 0.08, 0.082, 0.071
- Devo reconhecer que eu esperava uma diferença menor da predição que ja existia nos dados utilizados para especificação do modelo, mas em todos os dois casos, a predição fica dentro do intervalo de confiança de 95%

- 5: Acredito que o modelo se adaptou bem para os valores que não estavam contemplados nos dados coletados e todos foram validos para um Nível de confiança de 95%, porem se fosse para 99% nenhuma prediçao seria valida
- 100000: 0.457s 0.463, 0.463, 0.463, 0.464, 0.464 = 0.006, 0.006, 0.006, 0.007, 0.006 < Nível de Confiança 95%: TRUE, TRUE, TRUE, TRUE, TRUE
- 1200000: 5.678s - 5.563, 5.564, 5.566, 5.572, 5.57 = 0.114, 0.114, 0.112, 0.106, 0.108 < Nível de Confiança 95%: TRUE, TRUE, TRUE, TRUE, TRUE
- 6: De acordo com o modelo, o valor de N correspondente a 23s é de 4.8500679×10^6, Ja testando o mesmo N 4.8500679×10^6 no script `python3 fibo.py 4850068 4850068 5000 5`, gera os tempos de 23.14015, 23.35855, 23.34468, 23.35834, 23.36985 s. E os erros relacionado ao 23s são de 0.61, 1.53, 1.48, 1.53, 1.58 % respectivamente. O que é considerado algo bom utilizando um Nível de confiança de 95%, porém invalido para algumas medidas utilizando 99%.