

# ANÁLISE DOS DADOS COM R

Funções e Modelos

James R. Hunter, PhD

Retrovirologia, EPM, UNIFESP

2023-10-17



# FUNÇÕES & MODELOS



# MAS, PRIMEIRO ...

De novo, Jim ...? 🤔

# WORKING WITH STRINGS

```
1 #| label: string_setup
2 #| echo: false
3
4 peng_raw <- read_csv("penguins_raw.csv")
```

# penguins\_raw -> penguins

- Variável *Species*
- Formato *raw*: Adelie Penguin (*Pygoscelis adeliae*)
- Só quero o nome *Adelie*
- Como converter?

# SOLUÇÃO #1

# FORÇA BRUTA

- for Loop
- sem funções de Tidyverse

```
1 peng_mod <- clean_names(peng_raw)
2
3 peng_mod$species_mod <- "" # create a new name variable
4
5 for (i in 1:nrow(peng_raw)) {
6   if (substr(peng_mod$species[i], 1, 6) == 'Adelie'){
7     peng_mod$species_mod[i] <- "ADELIE"
8   } else {
9     if (substr(peng_mod$species[i], 1, 9) == 'Chinstrap'){
10      peng_mod$species_mod[i] <- "CHINSTRAP"
11    } else {
12      if (substr(peng_mod$species[i], 1, 6) == 'Gentoo'){
13        peng_mod$species_mod[i] <- "GENTOO"
14      }
15    }
16  }
17 }
18 ht(peng_mod$species_mod)
```

\$HEAD

```
[1] "ADELIE" "ADELIE" "ADELIE" "ADELIE" "ADELIE"
```

\$TAIL

```
[1] "CHINSTRAP" "CHINSTRAP" "CHINSTRAP" "CHINSTRAP" "CHINSTRAP"
```

# SOLUÇÃO #2



# TIDYVERSE - `str_split_i()`

- Adelie Penguin (*Pygoscelis adeliae*)
- `str_split_i()` divide o *string* em partes cada vez que encontra o caracter que indica onde divide (aqui espaço - " ")
- Retorna o elemento indicado pelo parâmetro `i`
  - Nome da especie
- Várias versões da função `str_split_x()`

```
str_split_i(string, pattern, i)
```

```
1 species_mod2 <- str_split_i(string = peng_raw$Species,  
2                             pattern = " ",  
3                             i = 1)  
4  
5 species_mod2[c(1:2, 343:344)]
```

```
[1] "Adelie"      "Adelie"      "Chinstrap"  "Chinstrap"
```

# FUNÇÕES

- Todos os comandos em R são funções
- Programados por desenvolvedores de R ou dos pacotes

```
1 geom_bar
function (mapping = NULL, data = NULL, stat = "count", position = "stack",
  ..., just = 0.5, width = NULL, na.rm = FALSE, orientation = NA,
  show.legend = NA, inherit.aes = TRUE)
{
  layer(data = data, mapping = mapping, stat = stat, geom = GeomBar,
    position = position, show.legend = show.legend, inherit.aes =
inherit.aes,
    params = list2(just = just, width = width, na.rm = na.rm,
      orientation = orientation, ...))
}
<bytecode: 0x1420c8408>
<environment: namespace:ggplot2>
```

# FUNÇÕES – GANHAR EFICIÊNCIA

- Todas essas vezes que repetir um bloco de código
  - Repetir um gráfico com parâmetros diferentes
  - Mesmo para um modelo
  - Evitar copiar-colar com erros
- **VSS:** Se você vai copiar/colar código mais de 2 vezes
  - Use uma função

# EXEMPLO

- Temos tibble com 5 variáveis numéricas **a - e**

```
1 set.seed(42)
2 ex <- tibble(a = rnorm(5),
3              b = rnorm(5),
4              c = rnorm(5),
5              d = runif(5),
6              e = runif(5))
7 ex
```

# A tibble: 5 × 5

	a	b	c	d	e
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1.37	-0.106	1.30	0.738	0.833
2	-0.565	1.51	2.29	0.811	0.00733
3	0.363	-0.0947	-1.39	0.388	0.208
4	0.633	2.02	-0.279	0.685	0.907
5	0.404	-0.0627	-0.133	0.00395	0.612

# O QUE QUEREMOS FAZER

- Colocar todas as variáveis numa escala de 0 até 1
  - Mesma coisa que `scales::rescale` faz
    - Mas fazemos aqui por copiar/colar

```
1 ex |> mutate(  
2   a = (a - min(a, na.rm = TRUE)) /  
3     (max(a, na.rm = TRUE) - min(a, na.rm = TRUE)),  
4   b = (b - min(b, na.rm = TRUE)) /  
5     (max(b, na.rm = TRUE) - min(a, na.rm = TRUE)),  
6   c = (c - min(c, na.rm = TRUE)) /  
7     (max(c, na.rm = TRUE) - min(c, na.rm = TRUE)),  
8   d = (d - min(d, na.rm = TRUE)) /  
9     (max(d, na.rm = TRUE) - min(d, na.rm = TRUE)),  
10 )
```

# A tibble: 5 × 5

	a	b	c	d	e
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	0	0.733	0.909	0.833
2	0	0.801	1	1	0.00733
3	0.479	0.00568	0	0.476	0.208
4	0.619	1.05	0.302	0.844	0.907
5	0.501	0.0215	0.342	0	0.612

## Todos são entre 0 e 1?

```
1 ex |> mutate(  
2   a = (a - min(a, na.rm = TRUE)) /  
3     (max(a, na.rm = TRUE) - min(a, na.rm = TRUE)),  
4   b = (b - min(b, na.rm = TRUE)) /  
5     (max(b, na.rm = TRUE) - min(a, na.rm = TRUE)),  
6   c = (c - min(c, na.rm = TRUE)) /  
7     (max(c, na.rm = TRUE) - min(c, na.rm = TRUE)),  
8   d = (d - min(d, na.rm = TRUE)) /  
9     (max(d, na.rm = TRUE) - min(d, na.rm = TRUE)),  
10 )
```

# A tibble: 5 × 5

	a	b	c	d	e
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	0	0.733	0.909	0.833
2	0	0.801	1	1	0.00733
3	0.479	0.00568	0	0.476	0.208
4	0.619	1.05	0.302	0.844	0.907
5	0.501	0.0215	0.342	0	0.612

Todos são entre 0 e 1?

Tem erro em b: copiou **a** em **min(a)** invés de b

**Funções ajudam evitar esse tipo de erro**

# CONSTRUÇÃO DE UMA FUNÇÃO



# PASSO 1 - ESTUDAR A COMPUTAÇÃO

- *O que quer fazer?*
  - Criar uma nova escala para uma variável
- *O que é o cálculo que quer empenha?*
  - $(a - \min(a, na.rm = TRUE)) / (\max(a, na.rm = TRUE) - \min(a, na.rm = TRUE))$
  - O que varia nesta expressão?
  - Essa vai ser o argumento que nós vamos dar para função
  - Pode dar para ele um nome abstrato, vamos dizer x
  - $(x - \min(x, na.rm = TRUE)) / (\max(x, na.rm = TRUE) - \min(x, na.rm = TRUE))$






## 4 ELEMENTOS DE UMA FUNÇÃO

- **Nome:** como vamos chamar esta função (`rescale01`)
- **Keyword:** `function( )`
- **Argumento(s):** valores que variam quando a função é chamada (`x`)
- **Corpo:** o código que executa o cálculo
- **Template para uma função**

```
1 nome <- function(argumentos) {  
2   corpo  
3 }
```

# NOSSO EXEMPLO

```
1 rescale01 <- function(x){
2   (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) -
3     min(x, na.rm = TRUE))
4 }
```

Data		
 ex	5 obs. of 5 variables	
Values		
gr	"/Users/jameshunter/Documents/MAD/Bioinformatica_...	
Functions		
.Last.value	function (x)	
ht	function (d, m = 5, n = m)	
rescale01	function (x)	

# TESTES SIMPLES

```
1 test1 <- c(0, 5, 10)
2 rescale01(test1)
```

```
[1] 0.0 0.5 1.0
```

```
1 test2 <- c(0, 5, NA, 10)
2 rescale01(test2)
```

```
[1] 0.0 0.5 NA 1.0
```

# APLICAR A FUNÇÃO PARA NOSSO EXEMPLO

```
1 ex |> mutate(  
2   a = rescale01(a),  
3   b = rescale01(b),  
4   c = rescale01(c),  
5   d = rescale01(d),  
6 )
```

# A tibble: 5 × 5

	a	b	c	d	e
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	0	0.733	0.909	0.833
2	0	0.761	1	1	0.00733
3	0.479	0.00540	0	0.476	0.208
4	0.619	1	0.302	0.844	0.907
5	0.501	0.0204	0.342	0	0.612

# INDIREÇÃO

- O que acontece quando uma variável é o argumento de uma função que fica **dentro** de uma outra função
- Usar `pacdemo` para ilustrar
- Queremos saber a média da carga viral por grupo
  - Uma *média por grupos* em geral
  - Vamos usar `dplyr::summarize()` para obter a média

```
1 pacdemo_mini <- readxl::read_excel("../pac_demo.xlsx") |>
2   mutate(logcv = log10(copias_cv)) |>
3   select(sexo, logcv)
```

# SE FAÇO DIRETAMENTE, OK

```
1 pacdemo_mini |>
2   group_by(sexo) |>
3   summarize(media = mean(logcv))
```

```
# A tibble: 2 × 2
  sexo      media
<chr>    <dbl>
1 Feminino  4.27
2 Masculino 4.13
```

# DENTRO DE UMA FUNÇÃO

```
1 group_mean <- function(data, group_var, mean_var){
2   data |>
3     group_by(group_var) |>
4     summarize(media = mean(mean_var))
5 }
6 group_mean(pacdemo_mini, sexo, logcv)
```

```
Error in `group_by()` :
! Rest group by variables found in `data`.
• Column `group_var` is not found.
Backtrace:
1. global group_mean(pacdemo_mini, sexo, logcv)
4. dplyr::group_by.data.frame(data, group_var)
```

- `summarize()` está procurando uma variável chamada `group_var`
  - Invés de `sexo` - o conteúdo de `group_var`
  - Faz parte de *tidy evaluation* - maneira e ordem em que funções são avaliadas



# COMO SUPERAR ESSE ERRO

- *Embracing* (abraços)
  - Literamente colocando chaves (*braces*) em volta da variável
  - `var` torna `{{ var }}`

```
1 group_mean <- function(data, group_var, mean_var){
2   data |>
3     group_by({{ group_var }}) |>
4     summarize(media = mean({{ mean_var }}), .groups = "drop")
5 }
6 group_mean(pacdemo_mini, sexo, logcv)
```

```
# A tibble: 2 × 2
  sexo      media
<chr>    <dbl>
1 Feminino  4.27
2 Masculino 4.13
```

# FUNÇÕES *HELPER* (ASSISTÊNCIA)

- Quer usar partes de uma função para fazer resumos paralelos de alguns dados
- *Helper Function*
- Como `summarytools::descr` faz

```
1 mini_summary <- function(data, var){
2   data |>
3     summarise(
4       min = min({{ var }}, na.rm = TRUE),
5       max = max( {{var }}, na.rm = TRUE),
6       med = median({{ var }}, na.rm = TRUE)
7     )
8 }
9 mini_summary(pacdemo_mini, logcv)
```

```
# A tibble: 1 × 3
  min    max    med
<dbl> <dbl> <dbl>
1  1.92  5.94  4.22
```

```
1 mini_summary(ex, c)
```

```
# A tibble: 1 × 3
  min    max    med
<dbl> <dbl> <dbl>
1 -1.39  2.29 -0.133
```

# QUANDO DEVEMOS USAR FUNÇÕES

- Questão Prática
- Quando copiar/colar pode introduzir erros
- Pode ter funções chamando funções
  - Cada função faz uma tarefa específica
  - Utilidade aqui para *embracing*
- Programação Funcional
  - *Subset* de linguagem especial que facilita mais complexidade nas funções
  - Estilo que reduz quase a 0 uso de loops - funções vetorizadas

# APLICAR FUNÇÃO ÀS VÁRIAS COLUNAS

- Introdução a programação funcional

## 3 FUNÇÕES E FAMÍLIAS DAS FUNÇÕES

- `apply` - família
- `across()` - `dplyr`
- `purrr::map_x()` - família

# FAMÍLIA **apply**

- `apply()`, `lapply()`, `sapply()`, e `tapply()`
- Base R
- Aprender `apply()`
- Dados: `pacdemo`

```
1 pacdemo |>
2   slice(1:4) |>
3   gt()
```

sexo	idade	cv	cd4	cd8
Masculino	60	5200	898	1311
Masculino	73	1947	958	817
Feminino	51	480000	958	817
Masculino	50	257313	142	1009

# MÉDIA DAS VARIÁVEIS NÚMERICAS

- Pode fazer uma lista dos cálculos

```
1 (med_idade <- mean(pacdemo$idade))
```

```
[1] 51.2
```

```
1 (med_cv <- mean(pacdemo$cv))
```

```
[1] 90692.82
```

```
1 (med_cd4 <- mean(pacdemo$cd4))
```

```
[1] 513.9
```

```
1 (med_cd8 <- mean(pacdemo$cd8))
```

```
[1] 994.3
```

- Lembre a regra de não copiar/colar mais de 2 vezes
- Pode fazer um função que vai de coluna em coluna
- `apply()` já foi escrito



# apply()

- Template:

```
apply(X, MARGIN, FUN, ...)
```

- **X**: o objeto que estamos avaliando (**pacdemo**)
- **MARGIN**: colunas (2) ou fileiras (1)
- **FUN**: função a ser aplicada (sem parênteses)

# SOMA DOS VALORES NÚMERICOS

```
1 apply(pacdemo[,2:5], 2, sum)
```

idade	cv	cd4	cd8
2560	4534641	25695	49715

# MÉDIA DOS VALORES NÚMERICOS

- Levando em conta que pode ter um valor NA
- `mean(x, na.rm = TRUE)`
- Os ... no template

```
1 apply(pacdemo[,2:5], 2, mean, na.rm = TRUE)
```

idade	cv	cd4	cd8
51.20	90692.82	513.90	994.30

# TIDYVERSE - `dplyr::across()`

- Permite que você fazer um cálculo ou operação em várias colunas
  - Como `apply`, mas mais flexível
  - Facilita trabalho com grupos e com colunas na mesma operação
  - Como `apply`, expressa a função dentro da função `across()`
  - Pode ser aplicada a uma variedade das colunas de um conjunto grande
  - Funções anônimas
    - Equivalente em R para funções lambda ( $\lambda$ ) em Python e outras idiomas

# CASO MAIS SIMPLES

- Mesma operação que com `apply()`
- Template:

```
across(.cols, .fns, ...)
```

```
1 test <- pacdemo |>
2   summarise(sum = across(idade:cd8, sum),
3             mean = across(idade:cd8, mean))
4 test
```

```
# A tibble: 1 × 2
  sum$idade      $cv $cd4  $cd8 mean$idade      $cv $cd4  $cd8
  <dbl>      <dbl> <dbl> <dbl>      <dbl> <dbl> <dbl> <dbl>
1      2560 4534641 25695 49715      51.2 90693.  514.  994.
```

# ESTRUTURA DE `test`

```
1 str(test)
```

```
tibble [1 × 2] (S3: tbl_df/tbl/data.frame)
 $ sum : tibble [1 × 4] (S3: tbl_df/tbl/data.frame)
  ..$ idade: num 2560
  ..$ cv   : num 4534641
  ..$ cd4  : num 25695
  ..$ cd8  : num 49715
 $ mean: tibble [1 × 4] (S3: tbl_df/tbl/data.frame)
  ..$ idade: num 51.2
  ..$ cv   : num 90693
  ..$ cd4  : num 514
  ..$ cd8  : num 994
```

- *Tibbles*, *tibbles* em todos os lugares!
- `across()` retorna um *tibble* com uma coluna por
  - Cada coluna em `.cols`
  - Cada função em `.funs`

# OUTRA MANEIRA DE MOSTRA AS DUAS FUNÇÕES

```
1 pacdemo |>  
2 summarise(res = across(idade:cd8, c(sum, mean)))
```

```
# A tibble: 1 × 1  
  res$idade_1 $idade_2    $cv_1  $cv_2 $cd4_1 $cd4_2 $cd8_1 $cd8_2  
    <dbl>      <dbl>    <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>  
1      2560      51.2 4534641 90693. 25695   514. 49715   994.
```

# ACROSS() E FUNÇÕES ANÔNIMAS

- Se precisamos colocar outro argumento no *call* para `across()`
  - Como controlar para NAs nos dados: `na.rm = TRUE`
- Precisa usar uma função anônima para especificar a função



# FUNÇÃO ANÔNIMA

- Anônima porque não tem nome
- Função que pode ser jogado no lixo depois de uso
- `\(x)` seguido pelo código da função
  - `\(x) x + 1` - aumentar o valor de `x` por 1
  - `\(x) median(x, na.rm = TRUE)`

# APLICAR PARA **pacdemo** summarise

- Calcular média das colunas numéricas
  - Cuidando de chance de ter NA

```
1 pacdemo |>
2 summarise(res = across(idade:cd8, \(x) mean(x, na.rm = TRUE)))
```

# A tibble: 1 × 1

	res\$idade	\$cv	\$cd4	\$cd8
	<dbl>	<dbl>	<dbl>	<dbl>
1	51.2	90693.	514.	994.

# FAMÍLIA `purrr::map_()`

- Pacote `purrr` desenhado para fornecer funções para programação funcional
- `purrr::map()` transforma cada elemento de uma lista ou vetor por aplicação de uma função
  - Retorna uma lista
- `map_lgl()`, `map_int()`, `map_dbl()`, e `map_chr()` retornam um vetor do tipo indicado

```
1 # work with penguins
2
3 n_unique <- function(x) length(unique(x))
4
5 peng_mod |>
6   select(species, island, sex, ) |>
7   map(n_unique)
```

\$species

[1] 3

\$island

[1] 3

\$sex

[1] 3

```
1 peng_mod |>
2   select(species, island, sex, ) |>
3   map_int(n_unique)
```

species	island	sex
3	3	3

```
1 peng_mod |>
2   select(flipper_length_mm, body_mass_g, culmen_length_mm ) |>
3   map_dbl(\(x) mean(x, na.rm = TRUE))
```

flipper_length_mm	body_mass_g	culmen_length_mm
200.91520	4201.75439	43.92193

**SÓ O INÍCIO COM** **pu rrr**

# MODELOS

- Tirar conclusões sobre uma população baseado numa amostra
- Mesma ideia atrás de inferência em estatística
- Modelos de *machine learning* como grandes modelos estatísticos
- Também existem modelos de simulação
  - Replicar com matemática um processo cujas dimensões e regras são bem compreendidas

# ESTATÍSTICA VS. *MACHINE LEARNING*

- Testes estatísticos tendem de ser mais simples para aplicar e analisar
- Modelos de *machine learning* podem formar estruturas e previsões mais sofisticadas
- *Machine learning* mais certo que os modelos estatísticos?
  - Estudo de 2018 diz que não têm resultados melhores<sup>1</sup>
- Decisão para usar um modelo de *machine learning* depende de:
  - Necessidade
  - Sofisticação do modelo para estudo
  - Tamanho do amostra
  - Habilidade e experiência do pesquisador com o algoritmo de *machine learning*

1. Makridakis S, Spiliotis E, Assimakopoulos V. Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLoS ONE. 2018 Mar 27;13(3):e0194889.



# SABORES DE *MACHINE LEARNING*

- Existe uma variável dependente
  - **Supervisionado**
    - Supervisão por causa que o resultado do modelo pode ser avaliado em termos da realidade dos resultados observados
  - 2 subtipos
    - **Classificação** - Colocar cada caso em um grupo baseado no valores das variáveis independentes
    - **Regressão** - Determinar um valor de uma combinação das variáveis independentes
- Não existe uma variável dependente
  - **Não-supervisionado**
    - Explorar a estrutura dos casos e tentar agrupar eles em *clusters* dos casos
    - Também, olhar na estrutura das variáveis independentes com PCA

# REGRESSÃO LINEAR SIMPLES

# REGRESSÃO – HISTORIA

- Termo vem de eugenismo (*eugenics*) de Sir Francis Galton.
- Estudou alturas de famílias
- Observou que crianças de pais altos tendiam de ser mais baixas de que os pais e crianças de pais baixos tendiam de ser mais altas
- Chamou a tendência **regressão à média**
- Usaremos esses dados clássicos

# MÉTODO DE MÍNIMOS QUADRADOS

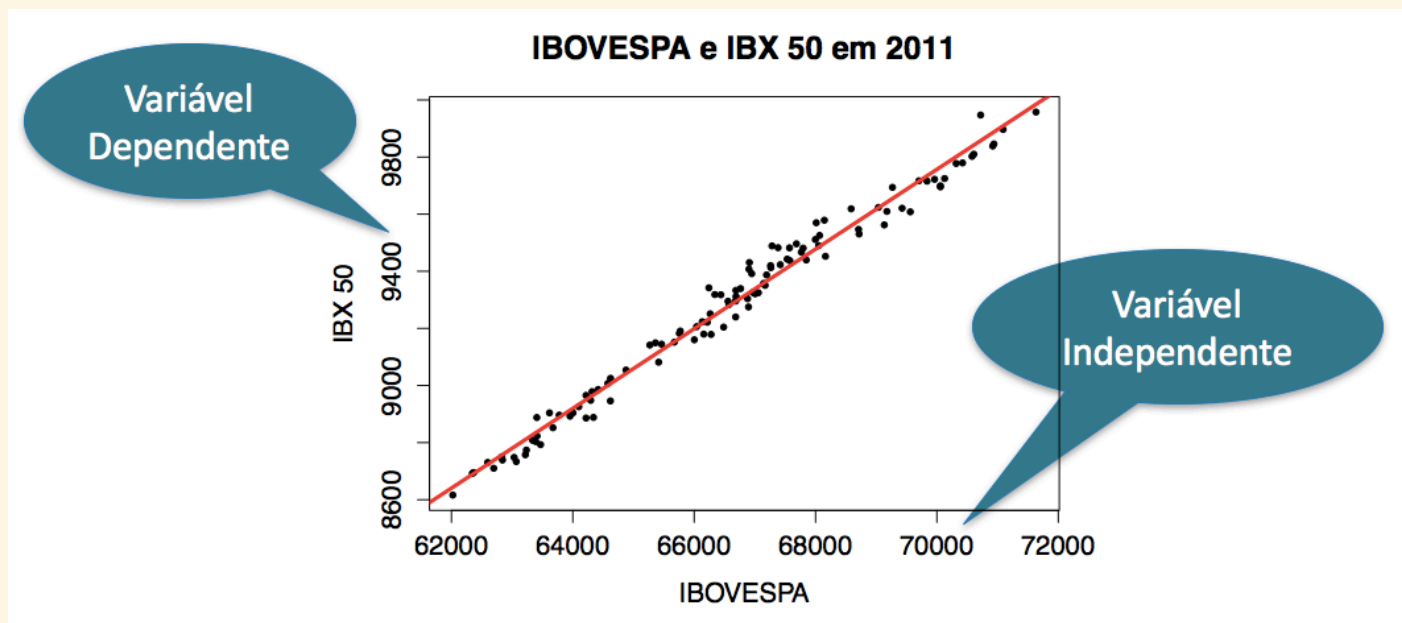
- Solucionamos com o método *Mínimos Quadrados*
  - Inventado por Carl Friedrich Gauss (1777 - 1855)
  - Método minimiza as divergências entre os valores lineares previstos e os valores dos dados
  - Consegue o melhor relação entre a variável de resultado e as variáveis prognósticas
- Por enquanto, vamos restringir o modelo para forma linear
  - Outras formas existem

# PROPOSITO

Prever um resultado numa variável dependente baseado em uma ou mais variáveis independentes

- Uma – regressão linear *simples*
- Mais – regressão linear *múltipla*

# VISUALIZAÇÃO DE REGRESSÃO



# LINHA RETA

$$y = \beta_0 + \beta_1 x$$

- $\beta_1$  = inclinação da linha (*slope*)
- $\beta_0$  = intercepto (onde cruza o eixo  $y$ )
- Os dois parâmetros da regressão
- Com estes parâmetros, Mínimos Quadrados acha a reta que melhor prevê o valor da variável dependente dado o valor de independente

## “MELHOR” QUER DIZER “BOM”?

- Apesar de ser a melhor maneira de prever  $y$ , possível que não descreve bem  $y$
- **Bom** depende dos dados
- **Melhor** depende do método



# EQUAÇÃO DE REGRESSÃO

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

- $Y_i$  = valor de variável dependente
- $\beta_0$  = intercepto
- $\beta_1$  = inclinação da reta de regressão
- $X_i$  = valor da variável independente
- $\epsilon_i$  = termo de erro de cada caso

# EQUAÇÃO DE REGRESSÃO - ESTIMAÇÃO

$$\hat{Y}_i = b_0 + b_1X_i + e_i$$

- $\hat{Y}_i$  = valor de variável dependente (estimado)
- $b_0$  = intercepto
- $b_1$  = inclinação da reta de regressão
- $X_i$  = valor da variável independente
- $e_i$  = termo de erro de cada caso

## TERMO DE ERRO ( $\epsilon$ )

- Também chamado **resíduo**
- Responsável pela variabilidade em  $y$  que a reta não consegue explicar

# MÍNIMOS QUADRADOS

- Faz o cálculo que minimiza o quadrado da soma dos erros
- Erros = resíduos = diferenças entre o valor *observado* e o valor *esperado*

$$\min \sum (y_i - \hat{y}_i)^2$$

- $y_i$  = valor observado da variável dependente
- $\hat{y}_i$  = valor estimado da variável dependente

## BASTA DE TEORIA – EXEMPLO

- A base de dados de Galton sobre altura nas famílias
- Pergunta é se filhos/as são mais altos ou mais baixos de que os pais
- Mediu 898 filhos/as em 197 famílias
- Base de dados originais (em papel) fica na University College, London (UCL)

# VARIÁVEIS

```
1 galton <- readRDS(here::here("../galton.rds"))
2 str(galton)
```

```
'data.frame':  898 obs. of  6 variables:
 $ family: Factor w/ 197 levels "1","10","100",...: 1 1 1 1 108 108 108 108 123
123 ...
 $ father: num  78.5 78.5 78.5 78.5 75.5 75.5 75.5 75.5 75 75 ...
 $ mother: num  67 67 67 67 66.5 66.5 66.5 66.5 64 64 ...
 $ sex    : Factor w/ 2 levels "F","M": 2 1 1 1 2 2 1 1 2 1 ...
 $ height: num  73.2 69.2 69 69 73.5 72.5 65.5 65.5 71 68 ...
 $ nkids  : int   4 4 4 4 4 4 4 4 2 2 ...
```

- **height, father, mother** todos medem altura em polegadas

# FOCO EM PAIS E FILHOS

```
1 boys <- galton %>%  
2   filter(sex == "M") %>%  
3   select(-family, -mother, -sex, -nkids)  
4 glimpse(boys)
```

Rows: 465

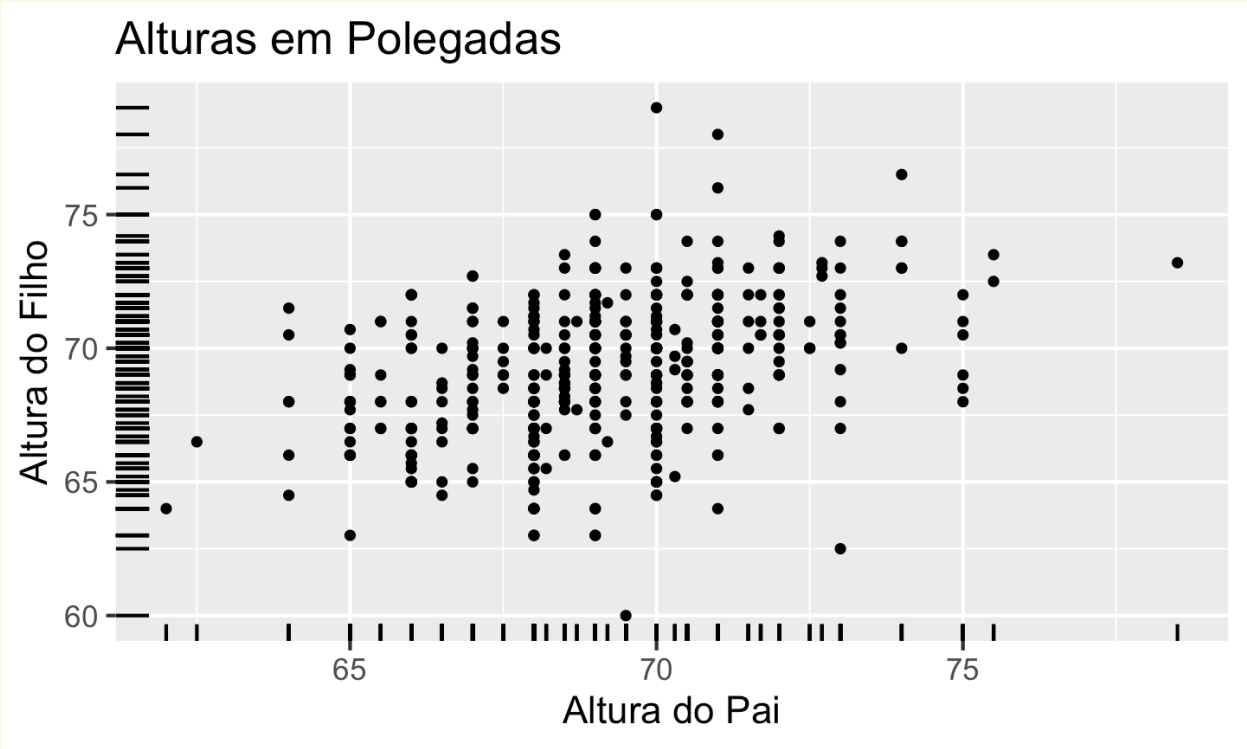
Columns: 2

\$ father <dbl> 78.5, 75.5, 75.5, 75.0, 75.0, 75.0, 75.0, 75.0, 75.0, 74.0,  
74....

\$ height <dbl> 73.2, 73.5, 72.5, 71.0, 70.5, 68.5, 72.0, 69.0, 68.0, 76.5,  
74....

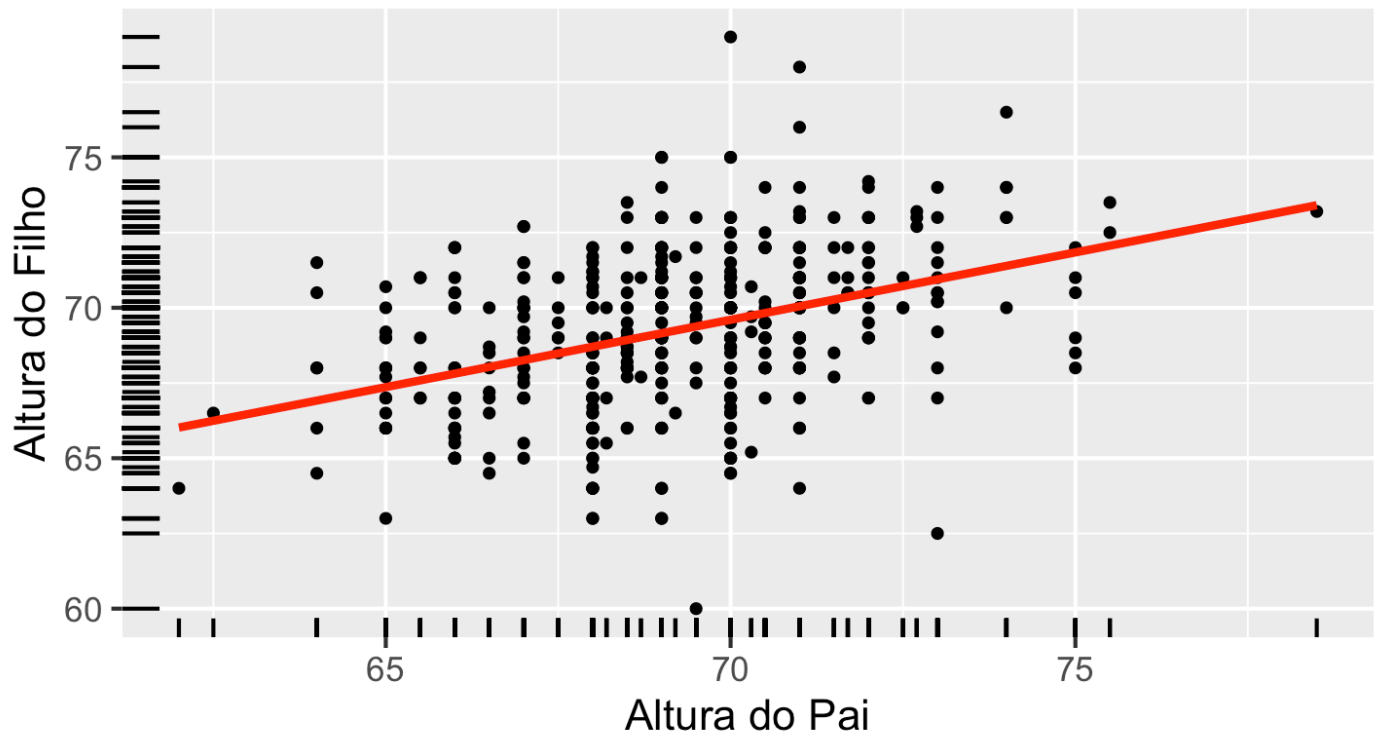
- **father** é a variável independente
- **height** é a variável dependente
- Queremos ver se a altura do pai prevê a altura do filho

# PAI/FILHO – GRÁFICO DE DISPERSÃO





## Alturas em Polegadas



# O QUE PODEMOS DIZER AGORA?

- Parece que mais altos os pais, mais altos os filhos
- Vamos olhar nas estatísticas descritivas das 2 variáveis
  - mais correlação

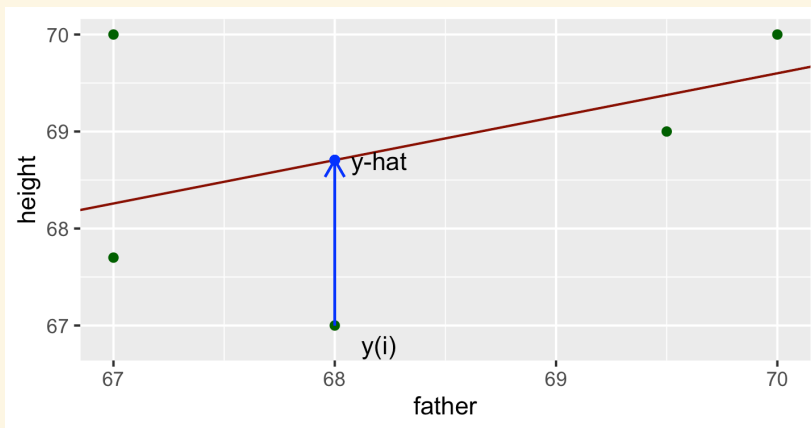
```
      vars    n  mean    sd median trimmed  mad min  max range  skew kurtosis
father    1 465 69.17 2.30   69.0   69.16 1.93  62 78.5  16.5  0.11    0.55
height    2 465 69.23 2.63   69.2   69.25 2.67  60 79.0  19.0 -0.03    0.29
      se
father 0.11
height 0.12
[1] "Coeficiente de Correlação: 0.391"
```

# O QUE É A “CORRELAÇÃO”?

- *Coefficiente de Correlação* mede o grau da associação linear entre 2 variáveis
- Sempre cai entre -1 e +1
  - -1 significa uma relação perfeitamente inversa (quando  $x$  sobe,  $y$  desce pela mesma proporção)
  - 0 significa que não existe uma relação linear entre as 2 variáveis
  - +1 significa uma relação perfeitamente positiva (quando  $x$  sobe,  $y$  sobe pela mesma proporção)
- V.S.S: quando tem correlação positiva, tem inclinação da linha de tendência positiva, e vice versa

# PARA CALCULAR A LINHA DE REGRESSÃO – O QUE QUEREMOS?

- Uma linha que minimiza a diferença entre  $y_i$  e  $\hat{y}$
- Precisamos trabalhar com o quadrado da diferença
  - para não ter uma soma de 0

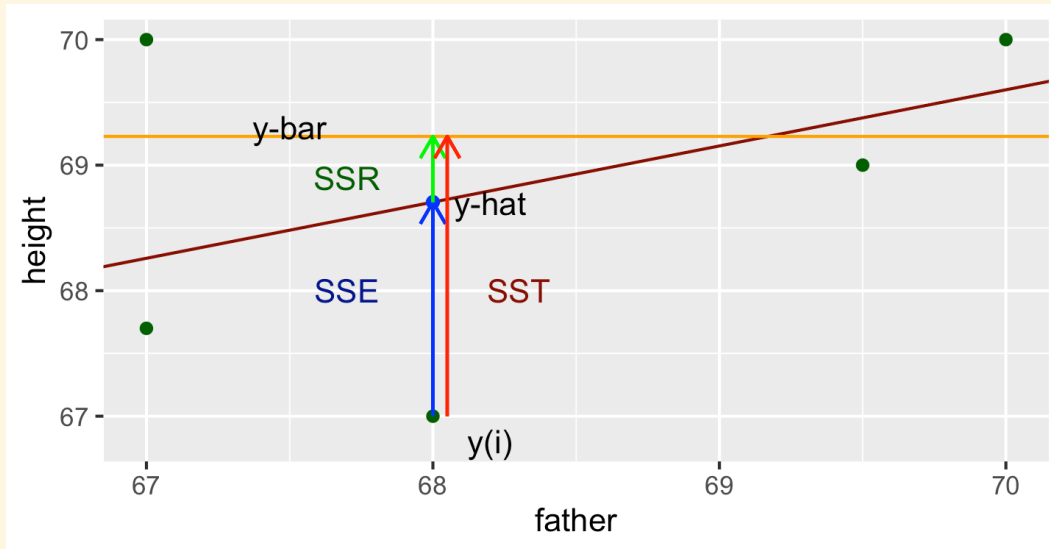


# SSE – UM COMPONENTE DO SOMA DE QUADRADOS (SST)

- $SST = SSE + SSR$
- SST – Total
- SSE – Relacionados aos Erros/Resíduos
- SSR – Relacionados/Explicados pela regressão

# SST – O QUE REPRESENTA?

- A variância total é a diferença entre o valor do modelo para cada valor de X e a média dos valores da variável dependente ( $y$ )



# SOMA DOS QUADRADOS

- Referimos a esse soma dos quadrados que queremos minimizar como **SSE**
  - Error sum of squares
- SSE como componente da soma dos quadrados total
  - SSE — soma dos quadrados relacionados ao resíduo
  - SSR — soma dos quadrados relacionados a regressão
- Expressão de SSE

$$SSE = \sum_{i=1}^n (y_i - \hat{y})^2$$

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

## PARA DETERMINAR A FORMULA PARA $\beta_0$ E $\beta_1$

- Para minimizar a SSE (determinar a linha mais eficiente), precisamos usar cálculo
- Fazer a derivativo parcial com respeito a  $\beta_0$  e  $\beta_1$

$$\frac{\partial}{\partial \beta_0} SSE = \frac{\partial}{\partial \beta_1} SSE = 0$$

- Chamadas as equações normais
- Confiamos nos softwares para calcular os parâmetros da equação



# FUNÇÃO EM R

- Função `lm` (“linear model”)
- `lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)`
- Os importantes são `formula`, `data`, `subset`, `weights`, `na.action`

# ARGUMENTOS PARA `lm()`

- **formula**: onde mostra quais variáveis você está modelando
  - Variável dependente vem primeiro
  - Separada da independente(s) por " ~ "
  - Para os **boys**: `height ~ father`
- **data**: data frame ou tibble que contem as variáveis
- **subset, weights**: parâmetros que permitem que você customizar tratamento das variáveis
- **na.action**: como vai tratar os dados missing na base de dados

# FUNÇÃO APLICADA AOS PAIS E FILHOS

- Função `lm` produz uma lista de 12 itens em um formato especial

```
1 fit1 <- lm(height ~ father, data = boys)
2 summary(fit1)
```

Call:

```
lm(formula = height ~ father, data = boys)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-9.3774	-1.4968	0.0181	1.6375	9.3987

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	38.25891	3.38663	11.30	<2e-16 ***
father	0.44775	0.04894	9.15	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.424 on 463 degrees of freedom

Multiple R-squared: 0.1531, Adjusted R-squared: 0.1513

F-statistic: 83.72 on 1 and 463 DF, p-value: < 2.2e-16

## O QUE DIZ O MODELO

$$\hat{y} = 38.259 + 0.448x$$

- Se o pai tivesse 0 altura, o filho teria 38.259 polegadas de altura
  - Não faz sentido prático, mas estabelece a base para calculo de altura
  - Para cada polegada incremental da altura do pai, o filho seria 0.448 polegadas mais alto

# EXTRAIR OS VALORES DOS COEFICIENTES

## 1. Usar `broom::tidy`

```
1 broom::tidy(fit1) %>% knitr::kable()
```

term	estimate	std.error	statistic	p.val
(Intercept)	38.2589122	3.3866340	11.297032	
father	0.4477479	0.0489353	9.149788	

## 2. Usar `coef`

```
1 coef(fit1)
```

```
(Intercept)      father  
38.2589122      0.4477479
```

# PREVISÕES DE NOVOS VALORES

- Pode usar o modelo para prever novos valores da altura dos filhos
- Usar `broom::augment`

```
1 fit1 %>% broom::augment(newdata = data_frame(father = 72))  
# A tibble: 1 × 2  
  father .fitted  
  <dbl>   <dbl>  
1     72    70.5
```

# **O QUE SIGNIFICA O MODELO? COMO INTERPRETAR ELE?**

# EXISTE RELAÇÃO ENTRE VARIÁVEIS INDEPENDENTE E DEPENDENTES?

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

- Se  $\beta_1$  (inclinação da linha) for 0, o que seria a equação?

$$Y_i = \beta_0 + \epsilon_i$$

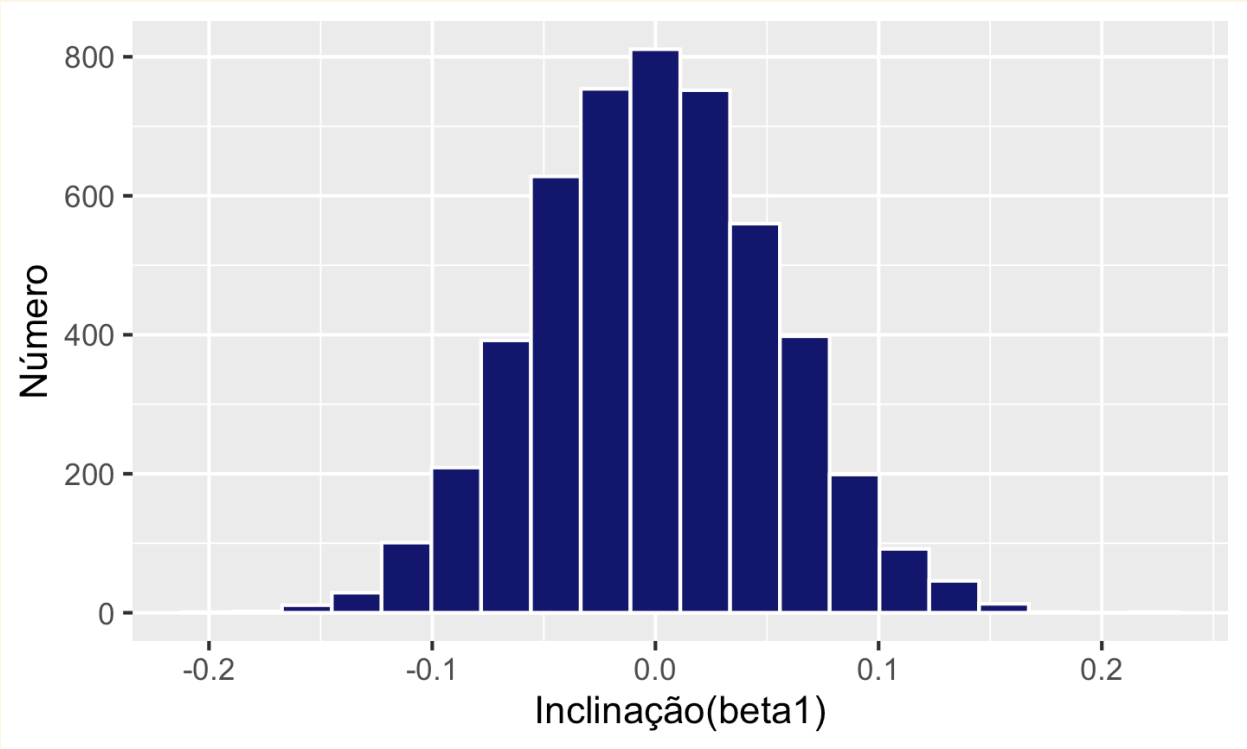
- X desaparece
- Relação entre Y e X não existe
  - Só tem intercepto e erro
- Faz possível teste eficiente de existência ou não de uma relação entre X e Y
- Cria uma hipótese nula de  $H_0 : \beta_1 = 0$



# TESTE DE HIPÓTESE NULA

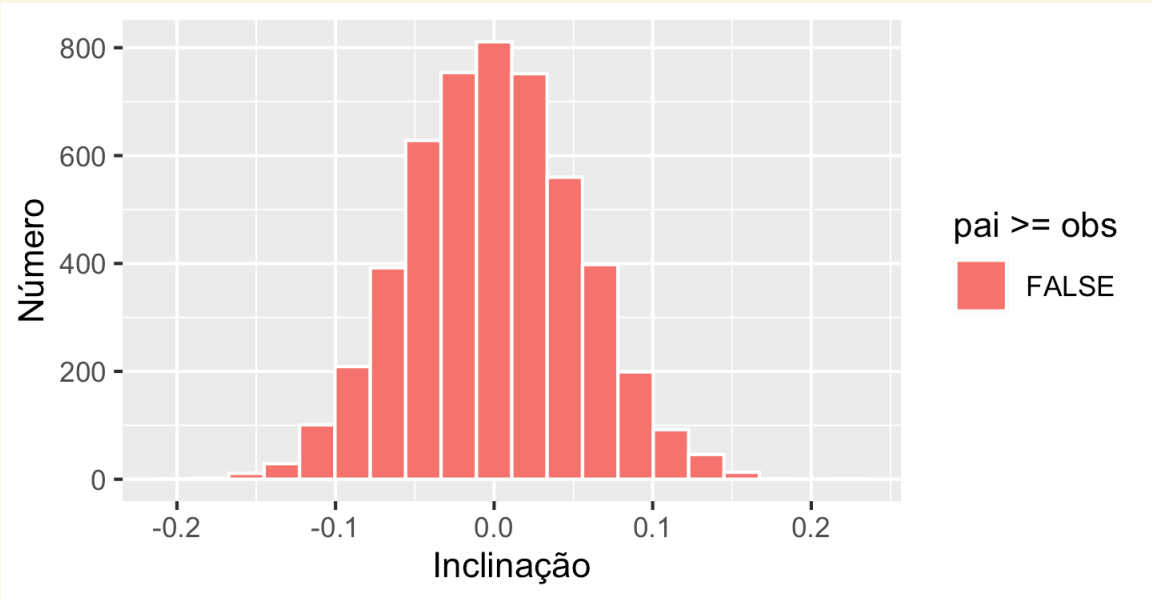
- Vamos fazer uma simulação de hipótese nula
- Se a nula é correta, qualquer altura do filho podia ter ocorrido com qualquer altura do pai.
- Podemos calcular o modelo de regressão 5.000 vezes com valores de todo a base de alturas dos filhos
- Como resultado, vamos focar nos valores da inclinação,  $\beta_1$
- Depois, nós vamos comparar nosso valor de  $\beta_1$  observado e ver onde cai na distribuição dos valores simulados

# HISTOGRAMA DAS INCLINAÇÃO DOS MODELOS



# HISTOGRAMA COM VALORES ABAIXO/ACIMA DO VALOR DA AMOSTRA

Número de simulações com  $\beta_{a1} \geq \text{obs}$ : 0



## O VALOR-P DA INCLINAÇÃO ( $\beta_1$ )

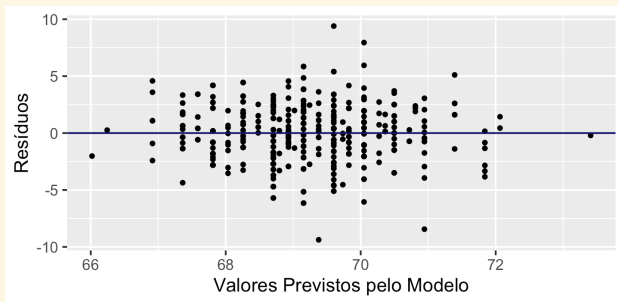
- Porque **nenhuma** das simulações produziu um valor superior ao observado (0.448)
  - Pode concluir que o valor-p deste teste é 0.
  - Não parece existir nenhuma chance que a inclinação = 0
- Assim, rejeitamos a hipótese nula e concluir que uma relação linear entre as alturas dos pais e filhos realmente existe.

# PREMISSAS DE REGRESSÃO LINEAR

1. Todas as variáveis independentes devem ter a mesma variância - Gráfico de resíduo deve evitar padrões indo de esquerda até direita
2. Todas as observações, resíduos e variáveis independentes: todos devem ser independentes - Gráfico de resíduo não deve mostrar um padrão sinuoso
3. Resíduos têm uma distribuição perto a normal - Gráfico “qq” dos resíduos padronizados - Indica que as variáveis têm distribuição normal multivariada
4. Variáveis independentes devem evitar *multicollinearity* - Ter correlações altas entre elas

# GRÁFICO DE RESÍDUOS

- Gráfico que mostra o valor previsto pelo modelo (“fitted value”) vs. o resíduo
- Uso da função `broom::augment()`
  - Eficiente para extrair os valores utilizados nos testes dos modelos



# IMPORTÂNCIA DOS RESÍDUOS

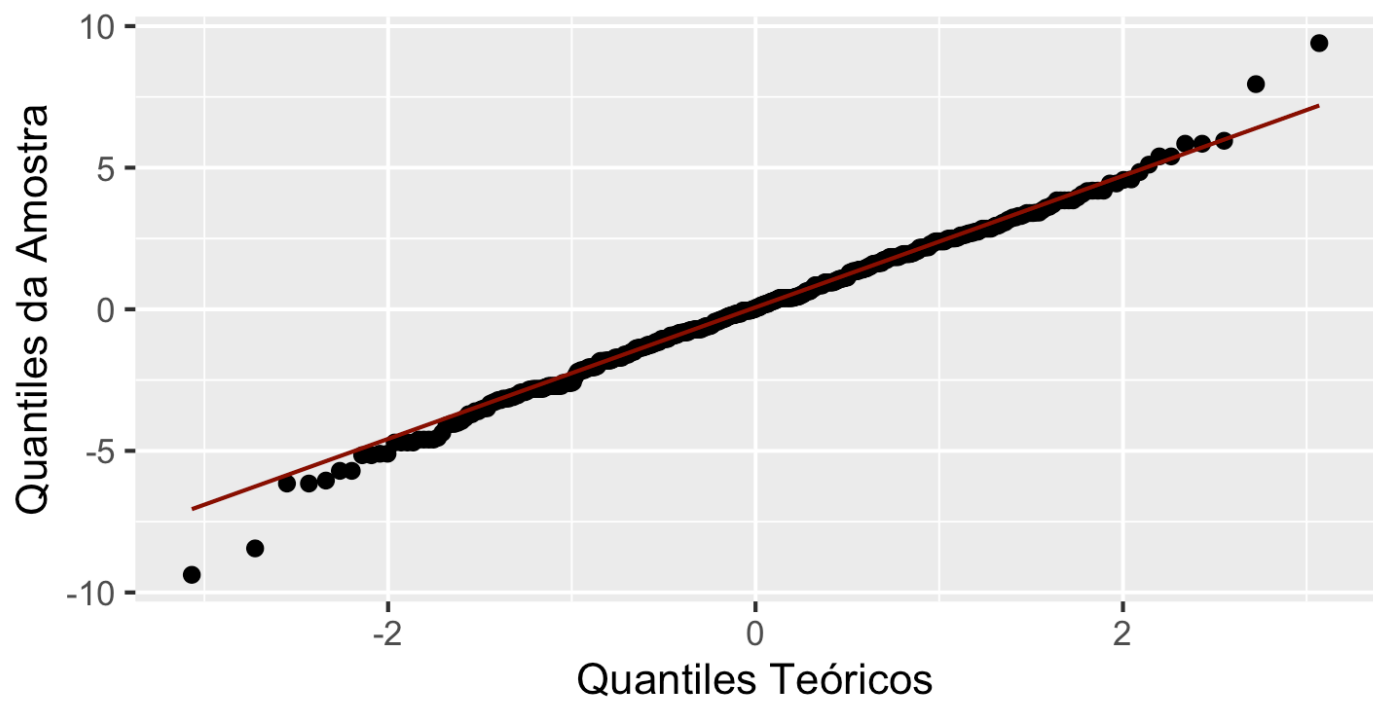
- Pode usar os erros/resíduos para verificar se as premissas da regressão foram respeitadas
- Não devem mostrar um padrão linear

# GRÁFICO Q-Q

- Verifica a normalidade dos resíduos
  - Mais perto a uma linha reta, melhor o “fit” com uma distribuição normal

```
1 grqq <- ggplot(data = mods, aes(sample = .resid))
2 grqq <- grqq + stat_qq()
3 grqq <- grqq + stat_qq_line(color = "darkred")
4 grqq <- grqq + labs(x = "Quantiles Teóricos",
5                     y = "Quantiles da Amostra")
```



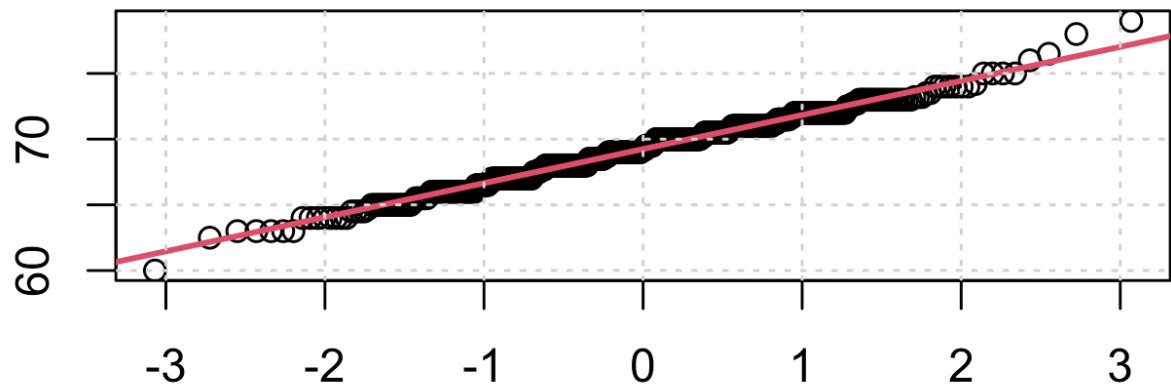


# GRÁFICOS Q-Q TAMBÉM DISPONÍVEL DIRETAMENTE EM BASE R

```
1 qqnorm(boys$height)
2 qqline(boys$height, col = 2, lwd = 2)
3 grid()
```

## Normal Q-Q Plot

Sample Quantiles



Theoretical Quantiles

# TESTE-F DAS VARIÂNCIAS DO MODELO

- Teste-F é um teste que verifica que as variâncias das variáveis são perto de iguais
- Utiliza a Distribuição F
  - Tem 2 graus de liberdade como parâmetros
- Serve como um teste de significância total de um modelo
- Produzido pela função `Summary` da função `lm`

# TESTE-F DO MODELO DAS ALTURAS PAI-FILHO

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	38.25891	3.38663	11.30	<2e-16	***
father	0.44775	0.04894	9.15	<2e-16	***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.424 on 463 degrees of freedom

Multiple R-squared: 0.1531, Adjusted R-squared: 0.1513

F-statistic: 83.72 on 1 and 463 DF, p-value: < 2.2e-16

# $R^2$ – COEFICIENTE DE DETERMINAÇÃO

- Medida de quanto a linha de regressão explica a variância em Y
- Relação entre a SSR e a SST

$$R^2 = \frac{SSR}{SST}$$

- Calculado pelo `lm`
  - visível em `Summary`
- Varia entre 0 e 1
- $\sqrt{R^2} = r$  (coeficiente de correlação)

# $R^2$

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	38.25891	3.38663	11.30	<2e-16	***
father	0.44775	0.04894	9.15	<2e-16	***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.424 on 463 degrees of freedom

Multiple R-squared: 0.1531, Adjusted R-squared: 0.1513

F-statistic: 83.72 on 1 and 463 DF, p-value: < 2.2e-16

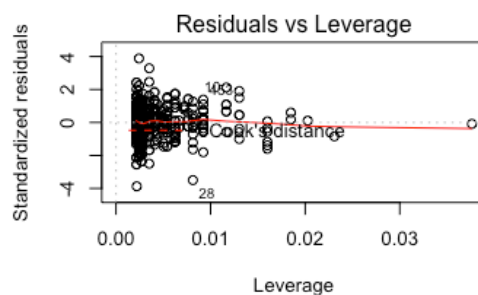
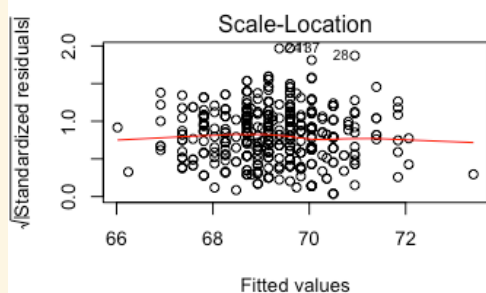
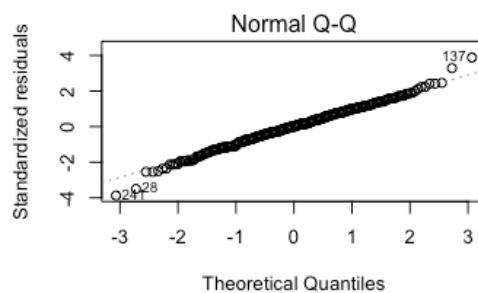
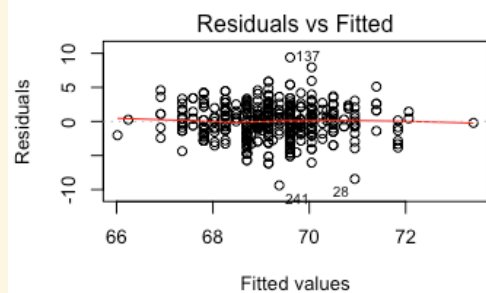
# SIGNIFICÂNCIA DE $R^2$

- Se 100% da variância ser explicado pela regressão
- $SSR = SST$
- $\therefore R^2 = SST/SST = 1$
- Variância completamente explicado pela regressão
- Em geral, o grau em que a regressão explica a variância no modelo



# DOIS GRÁFICOS MAIS AVANÇADOS

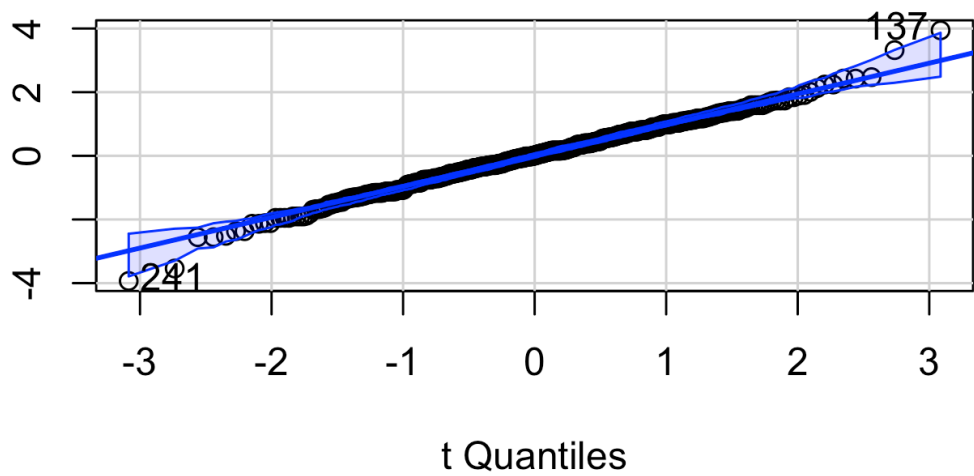
# FUNÇÃO `plot` PARA OBJETOS `lm`



# FUNÇÃO qqPlot ( ) DO PACOTE car

[1] 137 241

Studentized Residuals(fit1)



# REGRESSÃO LINEAR MÚLTIPLA - *MLR*

- Regressão quando tem mais de 1 variável independente
  - Mais de 1 covariado
- Mudança na Equação do Modelo de Regressão

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i + \dots + \beta_k X_i + \epsilon_i$$