Estatística Computacional

Universidade Federal da Bahia

Gilberto Pereira Sassi

Tópico 8

Pacotes usados nessa semana

```
set.seed(12345)
library(readxl)
library(readODS)
library(nortest)
library(qqplotr)
library(ggthemes)
library(patchwork)
library(modelr)
library(randtests)
library(broom)
library(glue)
# instale com remotes::install_github("gilberto-sassi/statBasics")
library(statBasics)
library(bootstrap)
library(infer)
library(tidyverse)
```

Bootstrap, Jackknife e Validação Cruzada

Bootstrap

Ideia: Usar a funções de distribuição acumulada empírica para imitar geração de valores da população.

- Nenhuma suposição paramétrica é feita sobre a população
- Usamos a função de distribuição acumulada empírica para imitar a geração de amostras ~~pseudo-~~aleatórias
- Gerar B amostras *bootstrap*: $\mathbf{x}_1, \ldots, \mathbf{x}_B$
- Criamos B replicações *bootstrap*: $\hat{\theta}_b^\star = t(\mathbf{x}_b^\star), b = 1, \ldots, B$
- Usamos $\hat{\theta}^{\star b}, b = 1, \ldots, B$ para estudar a distribuição de $\hat{\theta}$
- Usar B tão grande quanto for possível
- Geralmente para aproximar a variância de estimador, e intervalos de confiança

Variância de estimador – bootstrap

Seja $T = g(X_1, \ldots, X_n)$ um estimador θ com $X_i \sim F$, e $V_F(T)$ é a variança de T.

Ideia: Aproximamos $V_F(T)$ com $V_{\hat{F}}(T)$, e aproximamos $V_{\hat{F}}(T)$ com simulação de monte carlo.

Note que:

1. $\hat{F}$ é uma Função de Distribuição Acumulada de uma variável aleatória discreta
2. $\hat{F}$ tem suporte $\{x_1, \ldots, x_n\}$, com função de probabilidade $f(x_i) = \frac{1}{n}, i = 1, \ldots, n$
3. Por causa de 1. e 2., para gerar uma amostra de $\hat{F}$ basta sortear valores de $\{x_1, \ldots, x_n\}$ com reposição.

Recomendação: usar $50 \leq B \leq 200$.

Variância de estimador – bootstrap

Lembre que $\bar{X} \sim N(\mu, \frac{\sigma^2}{n})$. Como exemplo, vamos assumir que $\mu = 5$, $\sigma^2 = 2$ e $n = 25$.

```
amostra <- rnorm(25, mean = 5, sd = sqrt(2))
amostras_bootstrap <- seq_len(200) |>
  map(~ sample(amostra, length(amostra), replace = T))
replicacoes_boostrap <- amostras_bootstrap |> map_dbl(~ mean(.x))
variancia_bowley <- var(replicacoes_boostrap)
variancia_bowley
```

```
## [1] 0.0730862
```

Variância de estimador – bootstrap

Vamos estimar a variância do estimador do coeficiente de Bowley, $B = \frac{q_3 - 2 \cdot q_2 + q_1}{q_3 - q_1}$, em que q_1, q_2 e q_3 são o primeiro, segundo e o terceiro quartil, para a variável `hp` do conjunto de dados `mtcars`.

```
dados <- read_csv2("data/raw/mtcars.csv")
bowley <- function(x) {
  q1 <- quantile(x, probs = 0.25)
  q2 <- quantile(x, probs = 0.5)
  q3 <- quantile(x, probs = 0.75)
  (q3 - 2 * q2 + q1) / (q3 - q1)
}
variancia_bowley <- seq_len(200) |>
  map(~ sample(dados$hp, length(dados$hp), replace = T)) |>
  map_dbl(~ bowley(.x)) |>
  var()
variancia_bowley
```

```
## [1] 0.1532168
```

Vício do estimador – bootstrap

Seja $T = g(X_1, \ldots, X_n)$ um estimador θ com $X_i \sim F$, e $V_F(T)$ é a variança de T.

Ideia: Aproximamos $E_F(T)$ com $E_{\hat{F}}(T)$, e aproximamos $E_{\hat{F}}(T)$ com simulação de monte carlo.

Consequentemente, podemos aproximar do vício T por $\widehat{bias} \approx \bar{\hat{\theta}}^{\star} - \hat{\theta}$, em que $\bar{\hat{\theta}}^{\star}$ é uma aproximação de $E_F(T)$.

Note que:

1. $\hat{F}$ é uma Função de Distribuição Acumulada de uma variável aleatória discreta
2. $\hat{F}$ tem suporte $\{x_1, \ldots, x_n\}$, com função de probabilidade $f(x_i) = \frac{1}{n}, i = 1, \ldots, n$
3. Por causa de 1. e 2., para gerar uma amostra de $\hat{F}$ basta sortear valores de $\{x_1, \ldots, x_n\}$ com reposição.

Recomendação: usar $B \geq 400$.

Vício do estimador – bootstrap

Lembre que $\bar{X} \sim N(\mu, \frac{\sigma^2}{n})$. Como exemplo, vamos assumir que $\mu = 5$, $\sigma^2 = 2$ e $n = 25$.

```
amostra <- rnorm(25, mean = 5, sd = sqrt(2))
media_bootstrap <- seq_len(500) |>
  map(~ sample(amostra, length(amostra), replace = T)) |>
  map_dbl(~ mean(.x)) |>
  mean()
vicio <- media_bootstrap - mean(amostra)
vicio
```

```
## [1] 0.00457422
```

Vício do estimador – bootstrap

Vamos estimar o vício do estimador do coeficiente de Bowley, $B = \frac{q_3 - 2 \cdot q_2 + q_1}{q_3 - q_1}$, em que q_1, q_2 e q_3 são o primeiro, segundo e o terceiro quartil, para a variável `hp` do conjunto de dados `mtcars`.

```
dados <- read_csv2("data/raw/mtcars.csv")
amostra <- dados$hp
media_bootstrap <- seq_len(500) |>
  map(~ sample(amostra, length(amostra), replace = T)) |>
  map_dbl(~ bowley(.x)) |>
  mean()
vicio <- media_bootstrap - bowley(amostra)
vicio
```

```
##        75%
## -0.1102563
```

Intervalo de confiança – bootstrap

Método: intervalo de confiança `t`

Seja $T = g(X_1, \ldots, X_n)$ um estimador θ com $X_i \sim F$. O intervalo de confiança pode ser calculado por $IC(\theta, \gamma = 1 - \alpha) = \left(-t_{1-\frac{\alpha}{2};B-1} dp_{\hat{\theta}} + \hat{\theta}; t_{1-\frac{\alpha}{2};B-1} dp_{\hat{\theta}} + \hat{\theta}\right)$, em que $t_{1-\frac{\alpha}{2};B-1}$ é quantil de ordem $1 - \frac{alpha}{2}$ da distribuição t-Student com $B - 1$ graus de liberdade, e $dp_{\hat{\theta}}$ é desvio padrão bootstrap do estimador $\hat{\theta}$.

```
amostra <- read_csv2("data/raw/mtcars.csv")$hp
B <- 200
dp_boot <- seq_len(B) |>
  map(~ sample(amostra, length(amostra), replace = T)) |>
  map_dbl(~ bowley(.x)) |>
  sd()
quantil <- qt(0.975, df = B - 1)
glue("Limite inferior: {-quantil * dp_boot + bowley(amostra)}
Limite superior: {quantil * dp_boot + bowley(amostra)}")
```

```
## Limite inferior: -0.430562031357789
## Limite superior: 1.16110095351348
```

Intervalo de confiança – bootstrap

Método: percentil

Seja $T = g(X_1, \ldots, X_n)$ um estimador θ com $X_i \sim F$. O intervalo de confiança pode ser calculado por $IC(\theta, \gamma = 1 - \alpha) = \left(\hat{q}_\star\left(\frac{\alpha}{2}\right); \hat{q}_\star\left(1 - \frac{\alpha}{2}\right)\right)$, em que $\hat{q}_\star\left(\frac{\alpha}{2}\right)$ é o quantil de ordem $\frac{\alpha}{2}$ e $\hat{q}_\star\left(1 - \frac{\alpha}{2}\right)$ é o quantil de ordem $1 - \frac{\alpha}{2}$ das replicações *bootstrap* $\theta^\star_1, \ldots, \theta^\star_B$.

```
amostra <- read_csv2("data/raw/mtcars.csv")$hp
B <- 200 # nolint
replicacoes_bootstrap <- seq_len(B) |>
  map(~ sample(amostra, length(amostra), replace = T)) |>
  map_dbl(~ bowley(.x))
glue("Limite inferior: {quantile(replicacoes_bootstrap, probs = 0.025)}
Limite superior: {quantile(replicacoes_bootstrap, probs = 0.975)}")
```

```
## Limite inferior: -0.853046218487395
## Limite superior: 0.832373032373033
```

Intervalo de confiança – bootstrap

Método: intervalo pivotal

No mundo ideial

Definição: Seja $\hat{\theta} = T_n = g(X_1, \ldots, X_n)$ um estimador θ. Chamamos $R_n = \hat{\theta} - \theta$ de pivô.

Seja $H(x)$ a função de distribuição acumulada do pivô definido na definição acima, $H(x) = P\left(R_n \leq x\right)$, e considere

$$a = \hat{\theta} - H^{-1}\left(1 - \frac{\alpha}{2}\right),$$
$$b = \hat{\theta} - H^{-1}\left(\frac{\alpha}{2}\right),$$

em que $H^{-1}\left(\frac{\alpha}{2}\right)$ é o quantil de ordem $\frac{\alpha}{2}$ de R_n, e $H^{-1}\left(1 - \frac{\alpha}{2}\right)$ de ordem $1 - \frac{\alpha}{2}$ de R_n. Então, o intervalo de confiança é dado por

$$IC(\theta, \gamma = 1 - \alpha) = (a, b).$$

Note que

$$\begin{aligned} P\left(a \leq \theta \leq b\right) &= P\left(a - \hat{\theta} \leq \theta - \hat{\theta} \leq b - \hat{\theta}\right) \\ &= P\left(\hat{\theta} - b \leq \hat{\theta} - \theta \leq \hat{\theta} - a\right) \\ &= H(\hat{\theta} - a) - H(\hat{\theta} - b) \\ &= 1 - \frac{\alpha}{2} - \frac{\alpha}{2} \\ &= 1 - \alpha \end{aligned}$$

Problema: Difícil, ou mesmo impossível, estimar H.

Intervalo de confiança – bootstrap

Método: intervalo pivotal

Aproximando H: wConsidere $\hat{\theta}^\star_1, \ldots, \hat{\theta}^\star_B$ replicações bootstrap de $\hat{\theta}$, então podemos aproximar H por

$$\hat{H}(x) = \frac{\sum_{b=1}^{B} I\left(\hat{R}^\star_b \leq x\right)}{B},$$

em que $I\left(\hat{R}^\star_b \leq x\right) = \begin{cases} 1, & \hat{R}^\star_b \leq x \\ 0, & \hat{R}^\star_b > x \end{cases}$, e $\hat{R}^\star_b = \hat{\theta}^\star_b - \hat{\theta}$.

Seja $\hat{H}^{-1}(p)$ é o quantil de ordem p de $\{\hat{R}^\star_1, \ldots, \hat{R}^\star_B\} = \{\hat{\theta}_1 - \hat{\theta}, \ldots, \hat{\theta}_B - \hat{\theta}\}$, e note que $\hat{H}^{-1}(p) = \hat{q}^\star(p) - \hat{\theta}$ onde $\hat{q}^\star(p)$ é quantil de ordem p de $\{\hat{\theta}_1, \ldots, \hat{\theta}_B\}$.

$$\begin{cases} \hat{a} = \hat{\theta} - \hat{H}^{-1}\left(1 - \frac{\alpha}{2}\right) = \hat{\theta} - \left(\hat{q}^\star\left(1 - \frac{\alpha}{2}\right) - \hat{\theta}\right) = 2\hat{\theta} - \hat{q}^\star\left(1 - \frac{\alpha}{2}\right) \\ \hat{b} = \hat{\theta} - \hat{H}^{-1}\left(\frac{\alpha}{2}\right) = \hat{\theta} - \left(\hat{q}^\star\left(\frac{\alpha}{2}\right) - \hat{\theta}\right) = 2\hat{\theta} - \hat{q}^\star\left(\frac{\alpha}{2}\right) \end{cases}$$

Intervalo de confiança – bootstrap

Método: intervalo pivotal

```
amostra <- read_csv2("data/raw/mtcars.csv")$hp
B <- 200 # nolint
replicacoes_bootstrap <- seq_len(B) |>
  map(~ sample(amostra, length(amostra), replace = T)) |>
  map_dbl(~ bowley(.x))
glue("Limite inferior: {2 * bowley(amostra) - quantile(replicacoes_bootstrap, probs = 0.975)}
Limite superior: {2 * bowley(amostra) - quantile(replicacoes_bootstrap, probs = 0.025)}")
```

```
## Limite inferior: -0.0744129678786758
## Limite superior: 1.48592628257931
```

Convergência

É possível provar que $\lim_{n\to\infty} P(\hat{a} \leq \theta \leq \hat{b}) = 1 - \alpha$.

Teste de hipótese – bootstrap

Ideia: criar amostras *bootstrap* sob H_0.

- Para $H_0 : \mu = \mu_0$, considere a amostra sob H_0 dada por $\tilde{x}_i = x_i - \bar{x} + \mu_0$
- Para $H_0 : \sigma = \sigma_0$, considere a amostra sob H_0 dada por $\tilde{x}_i = \frac{x_i \sigma_0}{s}$, em que $s^2 = \frac{\sum_{i=1}^{n-1} (x_i - \bar{x})^2}{n-1}$
- Para $H_0 : Q(p) = m_0$, em que $Q(p)$ é o quantil de ordem p, considere a amostra sob H_0 dada por $\tilde{x}_i = x_i - q(p) + m_0$, onde $q(p)$ é o quantil de ordem p da amostra $x_1, \ldots, x_n$

Usamos $\tilde{x}_1, \ldots, \tilde{x}_n$ para teste de hipóteses de forma semelhante ao que vimos no tópico 7.

Procedimento de Neymann-Pearson – *bootstrap*

Ideia: Encontrar o valor crítico usando *bootstrap*.

1. Estabeleça H_0 e H_1
2. Escolha o nível de significância
3. Determine a região crítica e a estatística do teste
4. Gere uma amostra *bootstrap* usando $\tilde{x}_1, \ldots, \tilde{x}_n$
5. Compute a estatística do estatística para esta amostra *bootstrap*
6. Repita os passos **4.** e **5.** M vezes
7. Encontre o valor crítico, que é um quantil, usando os M valores das estatísticas
8. Calcule a estatística do teste para a amostra observada. Se ela estiver na região crítica, rejeitamos H_0

Procedimento de Neymann-Pearson – *bootstrap*

Exemplo

Considere a variável `hp` do conjunto de dados `mtcars`. Vamos testar $H_0 : \mu = 170$ e $H_1 : \mu \neq 170$. A região crítica é dada por $\{t \mid |t| > q_\alpha\}$, onde $t = \left|\frac{(\bar{x}-\mu_0)\cdot\sqrt{n}}{s}\right|$, e q_α é o quantil de ordem α da estatística t.

```
mu0 <- 170; B <- 200
amostra  <- read_csv2("data/raw/mtcars.csv")$hp
amostra_h0 <- amostra - mean(amostra) + mu0
reps_boot  <- seq_len(B) |>
  map(~ sample(amostra_h0, length(amostra_h0), replace = T)) |>
  map_dbl(~ (mean(.x) - mu0) * sqrt(length(.x)) / sd(.x)) |> abs()
q <- quantile(reps_boot, probs = 0.975)
estat <- abs((mean(amostra) - mu0) * sqrt(length(amostra)) / sd(amostra))
glue("Valores críticos: {q}. Estatística do teste: {estat}.
Decisão: {ifelse(estat > q, 'H1', 'H0')}.")
```

```
## Valores críticos: 2.15355936830683. Estatística do teste: 1.92342324055231.
## Decisão: H0.
```

valor-p – *bootstrap*

Ideia: Encontrar o valor-p usando *bootstrap*.

1. Estabeleça H_0 e H_1
2. Escolha o nível de significância
3. Calcule a estatística do teste para a amostra observada
4. Gere uma amostra *bootstrap* usando $\tilde{x}_1, \ldots, \tilde{x}_n$
5. Calcule a estatística do teste para esta amostra *bootstrap*
6. Repita os passos **4.** e **5.** M vezes
7. O valor-p é a proporção de estatísticas do teste no passo **6.** que são mais extremas que a estatística de teste calculada no passo **3.**

valor-p – *bootstrap*

Exemplo

Considere a variável `hp` do conjunto de dados `mtcars`. Vamos testar $H_0 : \mu = 170$ e $H_1 : \mu \neq 170$.

```
mu0 <- 170; B <- 200
amostra  <- read_csv2("data/raw/mtcars.csv")$hp
amostra_h0 <- amostra - mean(amostra) + mu0
estat <- abs((mean(amostra) - mu0) * sqrt(length(amostra)) / sd(amostra))
valor_p  <- seq_len(B) |>
  map(~ sample(amostra_h0, length(amostra_h0), replace = T)) |>
  map_dbl(~ (mean(.x) - mu0) * sqrt(length(.x)) / sd(.x)) |>
  map_dbl(~ abs(.x) > estat) |>
  mean()
glue("Decisão: {ifelse(valor_p < 0.05, 'H1', 'H0')}")
```

```
## Decisão: H0
```

Pacote `infer`

Pacote para construir intervalo de confiança e teste de hipóteses usando o estilo de codificação do `tidyverse`.

O pacote `infer` tem quatro verbos principais:

1. `specify()`: permite especificar a variável ou as variáveis se tivermos interesse em regressão linear. Podemos usar fórmula no caso de regressão linear ou `response = variable` para especificar a variável de interesse;
2. `hypothesize()`: No caso de regressão linear usamos `null = "independence"`, e para um variável usamos `null = "point"` e informamos o valor da hipótese nula: `p`, `mu`. `med` e `sigma`;
3. `generate()`: Geração de amostras *bootstrap*. Informamos `reps = B`, o número de amostras *bootstrap*, e `type = "bootstrap"`;
4. `calculate()`: permite calcular a estatística ou sumários com o argumento `stat`. Os valores possíveis de `stat`: `mean`, `median`, `sum`, `sd`, `prop`, `count`, `diff in means`, `diff in medians`, `diff in props`, `Chisq`, `F`, `slope`, `correlation`, `t`, `z`, `ratio of props` e `odds ratio`;

Pacote `infer`

Após esses quatro verbos principais, podemos usar *visualizar*, *calcular o valor-p* e *construir um intervalo de confiança*:

- `visualize`: histograma da distribuição das replicações *bootstrap*
- `get_confidence_interval`: intervalo de confiança com argumentos:
  - `level`: coeficiente de confiança
  - `type`: tipo de intervalo de confiança *bootstrap*
    - `"se"`: intervalo de confiança `t`
    - `"percentil"`: intervalo de confiança percentil
    - `"bias-corrected"`: intervalo de confinça pivotal
- `get_p_value`: calcula o valor-p
  - `obs_stat`: valor observado da estatística (podemos usar `calculate` para isso)
  - `direction`: opções incluem `"two_sided"`, `"left"` (ou `"less"`) and `"right"` (ou `"greater"`)

Pacote infer

Vamos checar se a nota de matemática dos candidatos no ENEM na edição de 2019 da cidade de **Palmas** é maior que 600.

```
df_palmas <- read_csv2("data/raw/equipe_sample__Palmas.csv")

estat_obs <- df_palmas |>
  specify(response = NU_NOTA_MT) |>
  calculate(stat = "mean")

df_palmas |>
  specify(response = NU_NOTA_MT) |>
  hypothesize(null = "point", mu = 600) |>
  generate(reps = 200, type = "bootstrap") |>
  calculate(stat = "mean") |>
  get_p_value(obs_stat = estat_obs, direction = "greater")
```

```
## # A tibble: 1 × 1
##   p_value
##     <dbl>
## 1       1
```

Pacote infer

Vamos construir um intervalo de confiança com coeficiente de confiança $\gamma = 95\%$ para média da nota de matemática no ENEM na edição 2019.

Método: intervalo de confiança `t`

```
df_palmas <- read_csv2("data/raw/equipe_sample__Palmas.csv")

estat_obs <- df_palmas |>
  specify(response = NU_NOTA_MT) |>
  calculate(stat = "mean")

df_palmas |>
  specify(response = NU_NOTA_MT) |>
  generate(reps = 200, type = "bootstrap") |>
  calculate(stat = "mean") |>
  get_confidence_interval(level = 0.95, type = "se", point_estimate = estat_obs)
```

```
## # A tibble: 1 × 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     535.     548.
```

Pacote infer

Vamos construir um intervalo de confiança com coeficiente de confiança $\gamma = 95\%$ para média da nota de matemática no ENEM na edição 2019.

Método: percentil

```
df_palmas <- read_csv2("data/raw/equipe_sample__Palmas.csv")

df_palmas |>
  specify(response = NU_NOTA_MT) |>
  generate(reps = 200, type = "bootstrap") |>
  calculate(stat = "mean") |>
  get_confidence_interval(level = 0.95, type = "percentile")
```

```
## # A tibble: 1 × 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     535.     548.
```

Pacote infer

Vamos construir um intervalo de confiança com coeficiente de confiança $\gamma = 95\%$ para média da nota de matemática no ENEM na edição 2019.

Método: pivotal

```
df_palmas <- read_csv2("data/raw/equipe_sample__Palmas.csv")

estat_obs <- df_palmas |>
  specify(response = NU_NOTA_MT) |>
  calculate(stat = "mean")

df_palmas |>
  specify(response = NU_NOTA_MT) |>
  generate(reps = 200, type = "bootstrap") |>
  calculate(stat = "mean") |>
  get_ci(level = 0.95, type = "bias-corrected", point_estimate = estat_obs)
```

```
## # A tibble: 1 × 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     536.     550.
```

Jackknife

- **Ideia:** usar parte dos dados para estimar o vício e a variância do estimador.

Seja $\hat{\theta} = T(X_1, \ldots, X_n)$ um estimador de θ. Considere

- **Amostra Jackknife:** $\{x_1, \ldots, x_n\} - \{x_i\}$ (i-ésima amostra Jackknife)
- **Replicação Jackknife:** $T^{(i)} = T(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$:
  - **Aproximando o vício:** $\widehat{\text{vício}}_{jack}(T) = (n-1)(\bar{T} - \hat{\theta})$
  - **Aproximando o desvio padrão:** $\widehat{dp}_{jack}(T) = \left[\frac{n-1}{n} \sum_{i=1}^{n} (\bar{T} - T^{(i)})^2\right]^{\frac{1}{2}}$

onde $\bar{T} = \frac{1}{n} \sum_{i=1}^{n} T^{(i)}$.

Jackknife

Exemplo

Calcular o vício e desvio padrão do estimador da correlação para o conjunto de dados `LSAT` (Equivalente ao Exame OAB no EUA).

```
dados <- read_xlsx("data/raw/LSAT.xlsx")
tam <- nrow(dados)

replicacoes_jack <- seq_len(tam) |>
  map(\(i) dados |> dplyr::filter(seq_len(nrow(dados)) != i)) |>
  map_dbl(\(amostra) cor(amostra$LSAT, amostra$GPA))
glue("Vício: {(tam - 1) * (mean(replicacoes_jack) - cor(dados$LSAT, dados$GPA))}.")
```

```
## Vício: -0.00293859136468688.
```

```
glue("Desvio padrão: {sqrt((tam - 1)^2 * var(replicacoes_jack) / tam)}.")
```

```
## Desvio padrão: 0.0533478477089502.
```

`jackknife` com o pacote `bootstrap`

Exemplo

Calcular o vício e desvio padrão do estimador da correlação para o conjunto de dados `LSAT` (Equivalente ao Exame OAB no EUA).

```
dados <- read_xlsx("data/raw/LSAT.xlsx")
tam <- nrow(dados)
estimador <- \(index, var1, var2) cor(var1[index], var2[index])
resultado <- jackknife(seq_len(tam), estimador, dados$LSAT, dados$GPA)
glue("Vício: {resultado$jack.bias}.")
```

```
## Vício: -0.00293859136468688.
```

```
glue("Desvio padrão: {resultado$jack.se}.")
```

```
## Desvio padrão: 0.0533478477089502.
```

`jackknife` com o pacote `bootstrap`

Exemplo

Calcular o vício e o o desvio padrão do coeficiente de Bowley para a variável `hp` do conjunto de dados `mtcars`.

```
dados <- read_csv2("data/raw/mtcars.csv")
estimador <- \(x) {
  q1 <- quantile(x, probs = 0.25); q2 <- quantile(x, probs = 0.50)
  q3 <- quantile(x, probs = 0.75)
  (q3 - 2 * q2 + q1) / (q3 - q1)
}
resultado <- jackknife(dados$hp, estimador)
glue("Desvio padrão: {resultado$jack.se}")
```

```
## Desvio padrão: 0.210671622248335
```

```
glue("Vício: {resultado$jack.bias}")
```

```
## Vício: 0.47949904609858
```

Validação Cruzada

Ideia: Para novas observações, calculamos a média do desvio quadrático.

- Avaliar a *exatidão* do modelo
- Escolher um modelo adequado para o problema em análise entre vários competidores
- Usado para avaliar o erro de predição ou o erro de classificação

Imagine que temos um modelo $y = \hat{f}(\mathbf{x})$, e suponha que temos novas observações $(y_1, \mathbf{x}_1), \ldots, (y_m, \mathbf{x}_m)$. Então o erro de predição é dado por

$$EP = \frac{\sum_{i=1}^{m} \left(\hat{f}(\mathbf{x}_i) - y_i \right)^2}{m}$$

Problema: Geralmente, não é simples coletar novas observações $(y_1, \mathbf{x}_1), \ldots, (y_m, \mathbf{x}_m)$.

Solução: particione a amostra original em duas partes:

- **Amostras de treinamento:** observações usadas para ajuste do modelo;
- **Amostras de teste:** observações usadas para teste do modelo.

Validação cruzada em K dobras: particione os dados K partitions de tamanhos aproximadamente iguais. Uma parte de cada participação é reservada para o ajuste da amostra, e outra parte de cada participação é reservada para calcular EP.

Procedimento para validação cruzada

1. Crie K partições da amostra. Assuma que $n = r \cdot K$, em que r é o tamanho da *amostra de teste* em cada participação.
2. Para uma participação, use $n - r$ observações para ajustar o modelo.
3. Para uma participação, use r observações para calcular EP.
4. Repita os passos 2) e 3) para todas as participações obtendo $EP_1, \cdots, EP_K$.
5. A validação cruzada é obtida através de $VC = \frac{\sum_{j=1}^{K} EP_j}{K}$.

Validação Cruzada

Vamos usar o conjunto de dados `carros.xlsx` com 50 observações e as seguintes variáveis: `vel` – velocidade em `mph`; `dist` – distância percorrida. Mais informações em `?cars`.

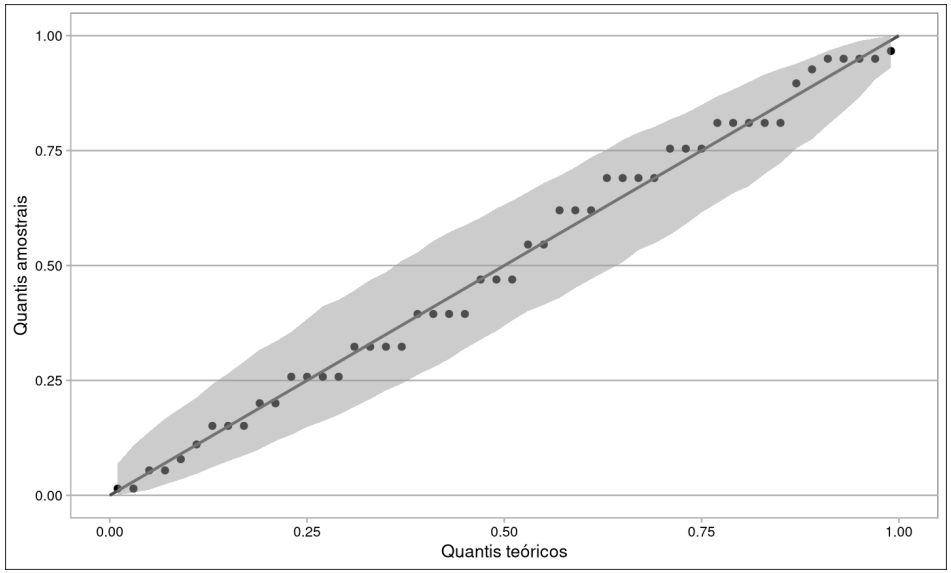
```
df_carros <- read_xlsx("data/raw/carros.xlsx")

# dist tem distribuição normal
ad.test(df_carros$vel)
```

```
##
##  Anderson-Darling normality test
##
## data:  df_carros$vel
## A = 0.26143, p-value = 0.6927
```

```
shapiro.test(df_carros$vel)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_carros$vel
## W = 0.97765, p-value = 0.4576
```

```
ggplot(df_carros, aes(sample = vel)) +
  stat_pp_point() +
  stat_pp_line() +
  stat_pp_band() +
  theme_calc() +
  labs(x = "Quantis teóricos", y = "Quantis amostrais")
```



Então podemos usar regressão linear para este modelo. Vamos usar *validação cruzada* para decidir entre três modelos:

1. $vel = 1 + \beta_1 dist + \epsilon$ em que $\epsilon \sim N(0, \sigma^2)$.
2. $vel = \beta_0 + \beta_1 dist + \beta_2 dist^2 + \epsilon$ em que $\epsilon \sim N(0, \sigma^2)$.
3. $vel = \beta_0 + \beta_1 dist + \beta_2 dist^2 + \beta_3 dist^3 + \epsilon$ em que $\epsilon \sim N(0, \sigma^2)$.

Vamos usar o pacote `modelr` do *framework* `tidyverse` para calcular validação cruzada:

- `cross_kfold(data, k)`: cria K partições da amostra.
- `mse()`: calcula o EP.

Validação cruzada

```
particao <- crossv_kfold(df_carros, k = nrow(df_carros))

vc_linear <- particao$train |>
  map(~ lm(vel ~ 1 + dist, data = .x)) |>
  map2_dbl(particao$test, mse) |>
  mean()

vc_quadrado <- particao$train |>
  map(~ lm(vel ~ 1 + dist + I(dist^2), data = .x)) |>
  map2_dbl(particao$test, mse) |>
  mean()

vc_cubico <- particao$train |>
  map(~ lm(vel ~ 1 + dist + I(dist^2) + I(dist^3), data = .x)) |>
  map2_dbl(particao$test, mse) |>
  mean()

tab_vc <- tibble(
  modelo = c("Linear", "Quadrático", "Cúbico"),
  VC = c(vc_linear, vc_quadrado, vc_cubico)
)
```

modelo	VC
Linear	10,525
Quadrático	9,023
Cúbico	9,272

Validação cruzada

```
ajuste <- lm(vel ~ 1 + dist + I(dist^2), data = df_carros)
tidy(ajuste)
```

```
## # A tibble: 3 × 5
##   term        estimate std.error statistic     p.value
##   <chr>          <dbl>     <dbl>     <dbl>       <dbl>
## 1 (Intercept)  5.14     1.30          3.97 0.000244
## 2 dist         0.327    0.0547        5.98 0.000000286
## 3 I(dist^2)   -0.00153  0.000494     -3.09 0.00332
```

```
glance(ajuste)
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC deviance df.resi
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>       <
## 1     0.710         0.698  2.91      57.6 2.30e-13     2  -123.  254.  261.     397.
```

Validação cruzada

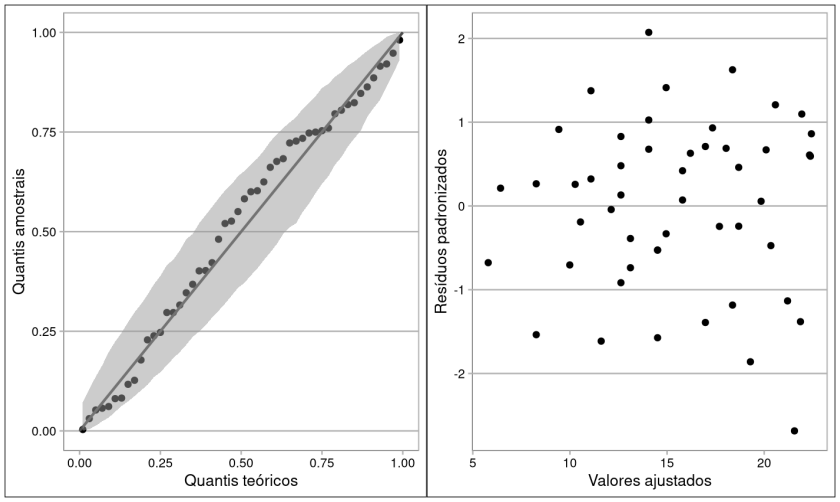
```
info_geral <- augment(ajuste)
shapiro.test(info_geral$.std.resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  info_geral$.std.resid
## W = 0.97958, p-value = 0.5345
```

```
runs.test(info_geral$.std.resid)
```

```
##
##  Runs Test
##
## data:  info_geral$.std.resid
## statistic = -0.57155, runs = 24, n1 = 25, n2 = 25, n = 50, p-value = 0.5676
## alternative hypothesis: nonrandomness
```

```
plot_qqnorm  <- ggplot(info_geral, aes(sample = .std.resid)) +
  stat_pp_point() +
  stat_pp_line() +
  stat_pp_band() +
  theme_calc() +
  labs(x = "Quantis teóricos", y = "Quantis amostrais")
plot_residuo <- ggplot(info_geral) +
  geom_point(aes(.fitted, .std.resid)) +
  theme_calc() +
  labs(y = "Resíduos padronizados", x = "Valores ajustados")
plot_qqnorm | plot_residuo
```



Validação cruzada

```
ggplot(info_geral) +
  geom_point(aes(x = seq_along(.std.resid), y = .std.resid)) +
  labs(x = "Ordem de observação", y = "Resíduos padronizados") +
  theme_calc()
```