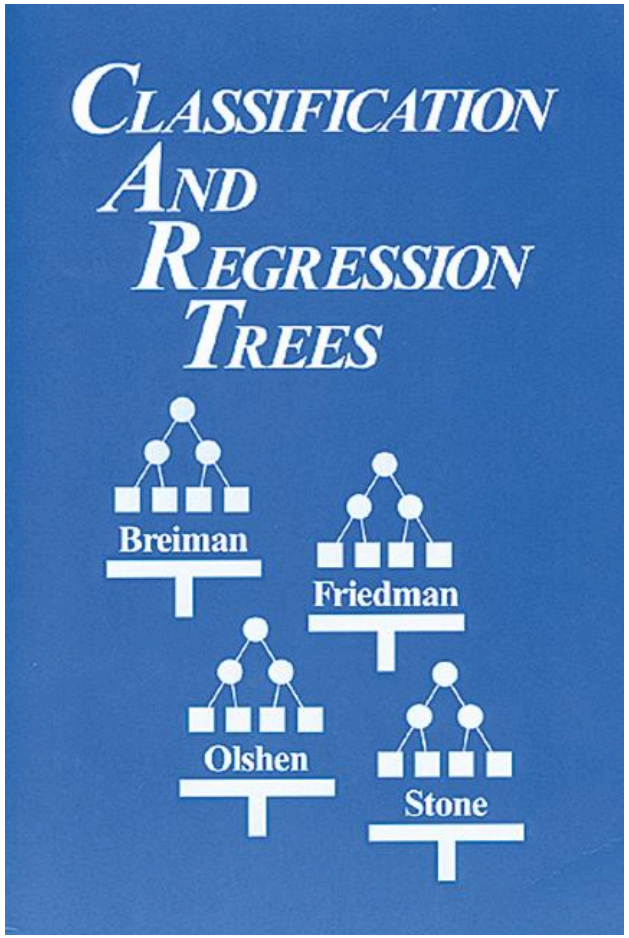
Árvore de Regressão e Classificação

Tiago Mendonça dos Santos

tiagoms.com
tiagomendonca
tiagoms1@insper.edu.br

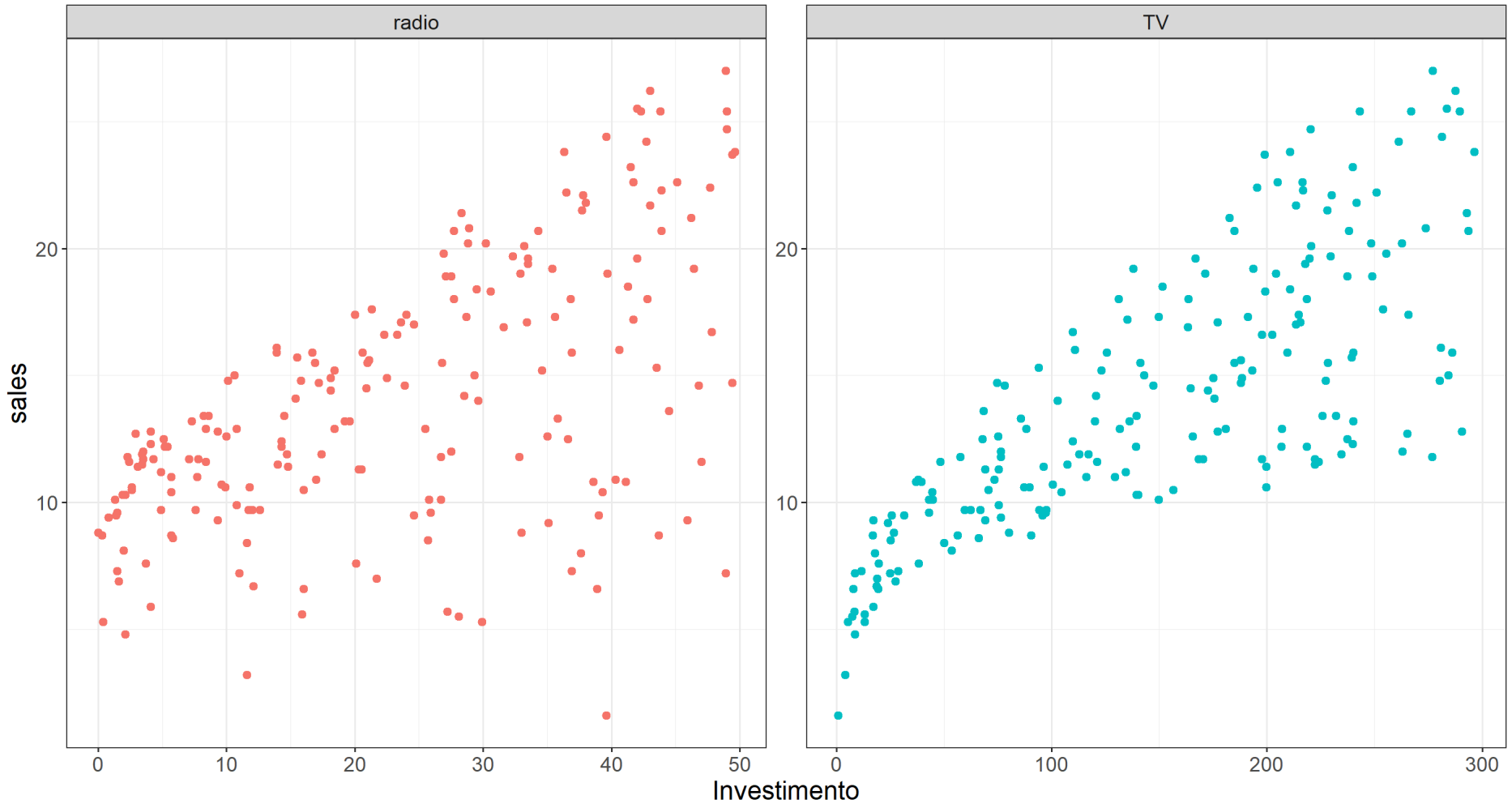
Introdução

Introdução



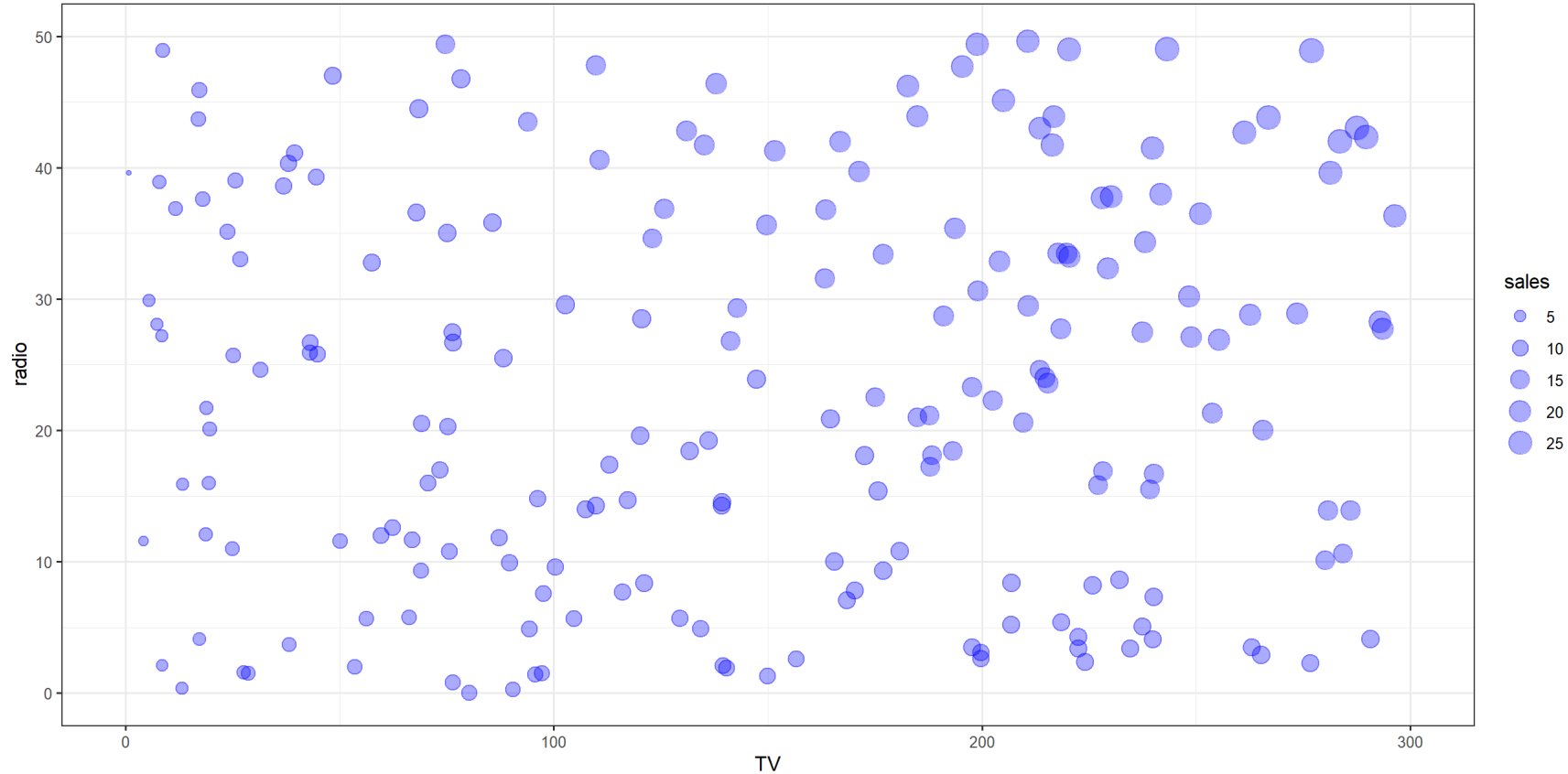
Leo Breiman *Statistical Modeling: The Two Cultures*

Advertising[1]



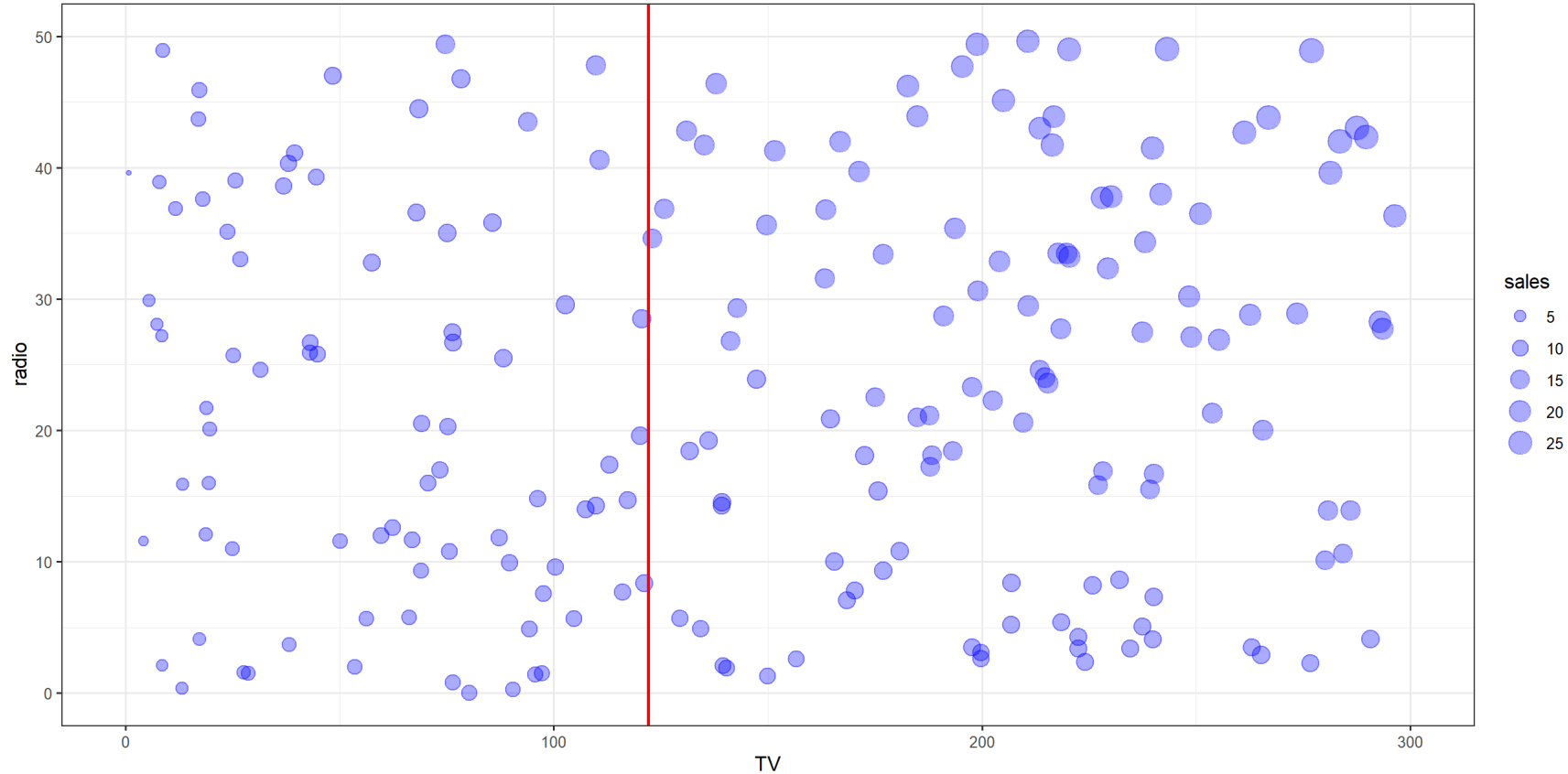
[1] dados retirados do livro *An Introduction to Statistical Learning with Applications in R*.

Árvore de Regressão



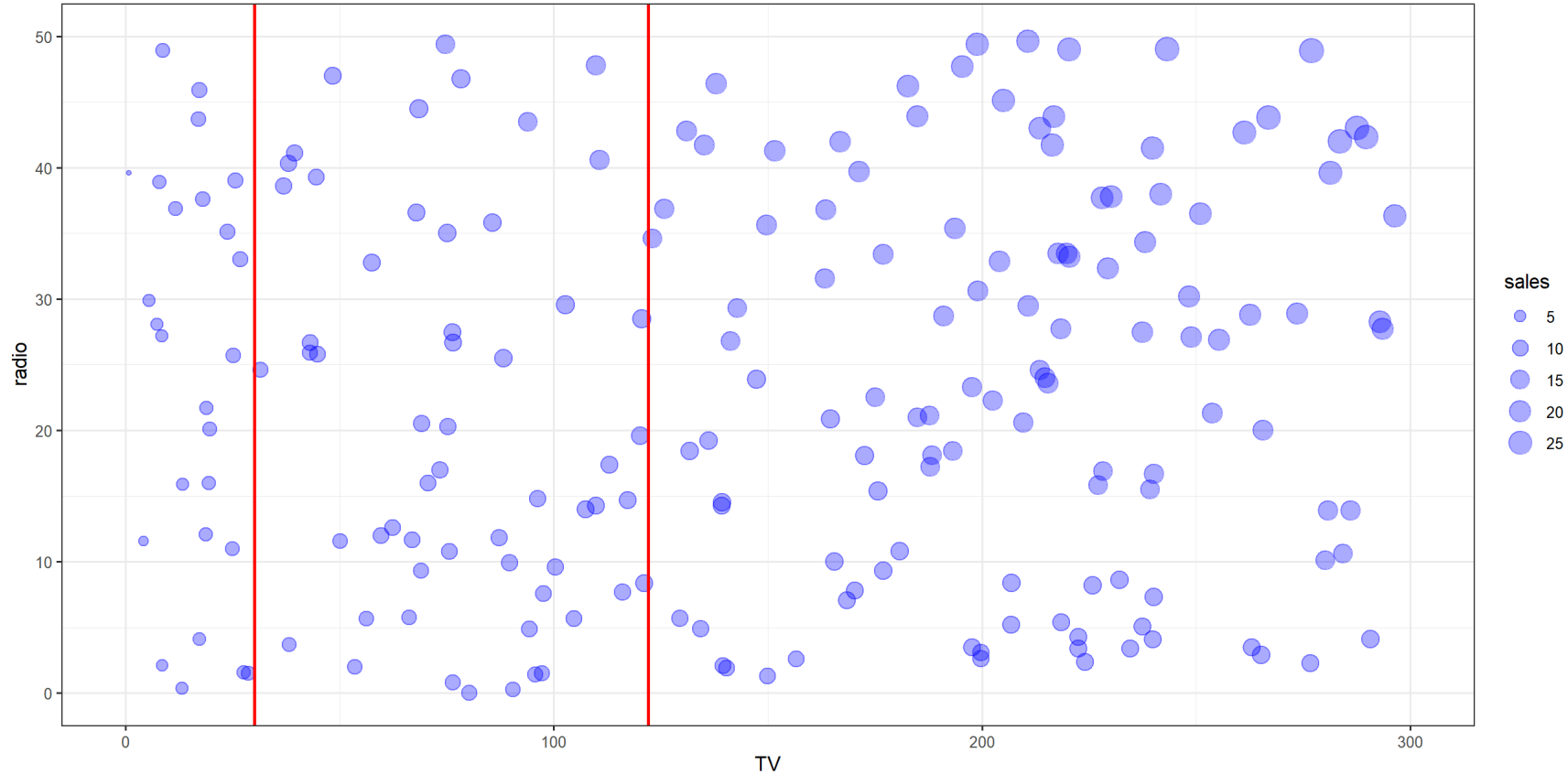
objetivo: criar regiões homogêneas.

Árvore de Regressão



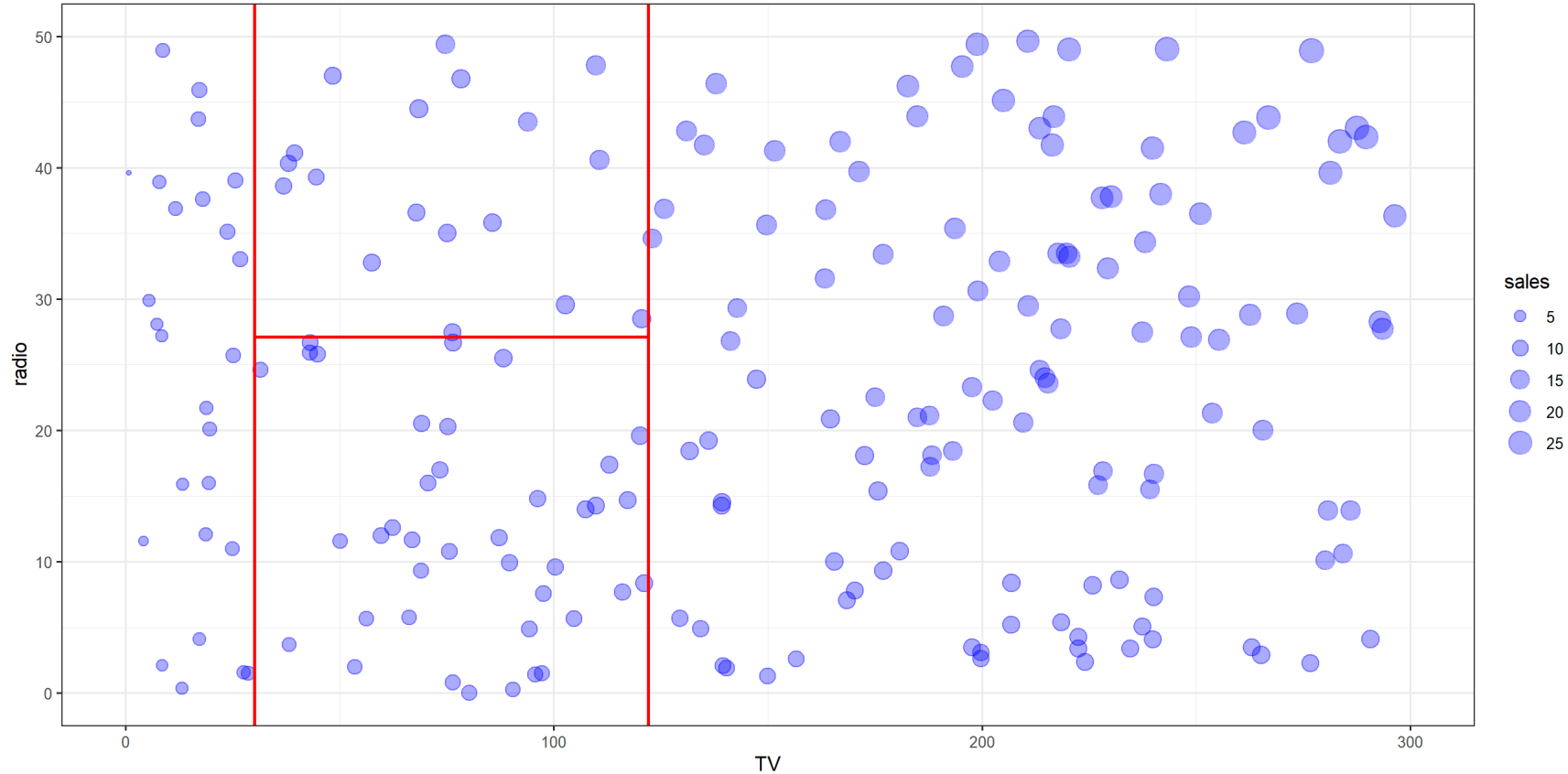
objetivo: criar regiões homogêneas.

Árvore de Regressão



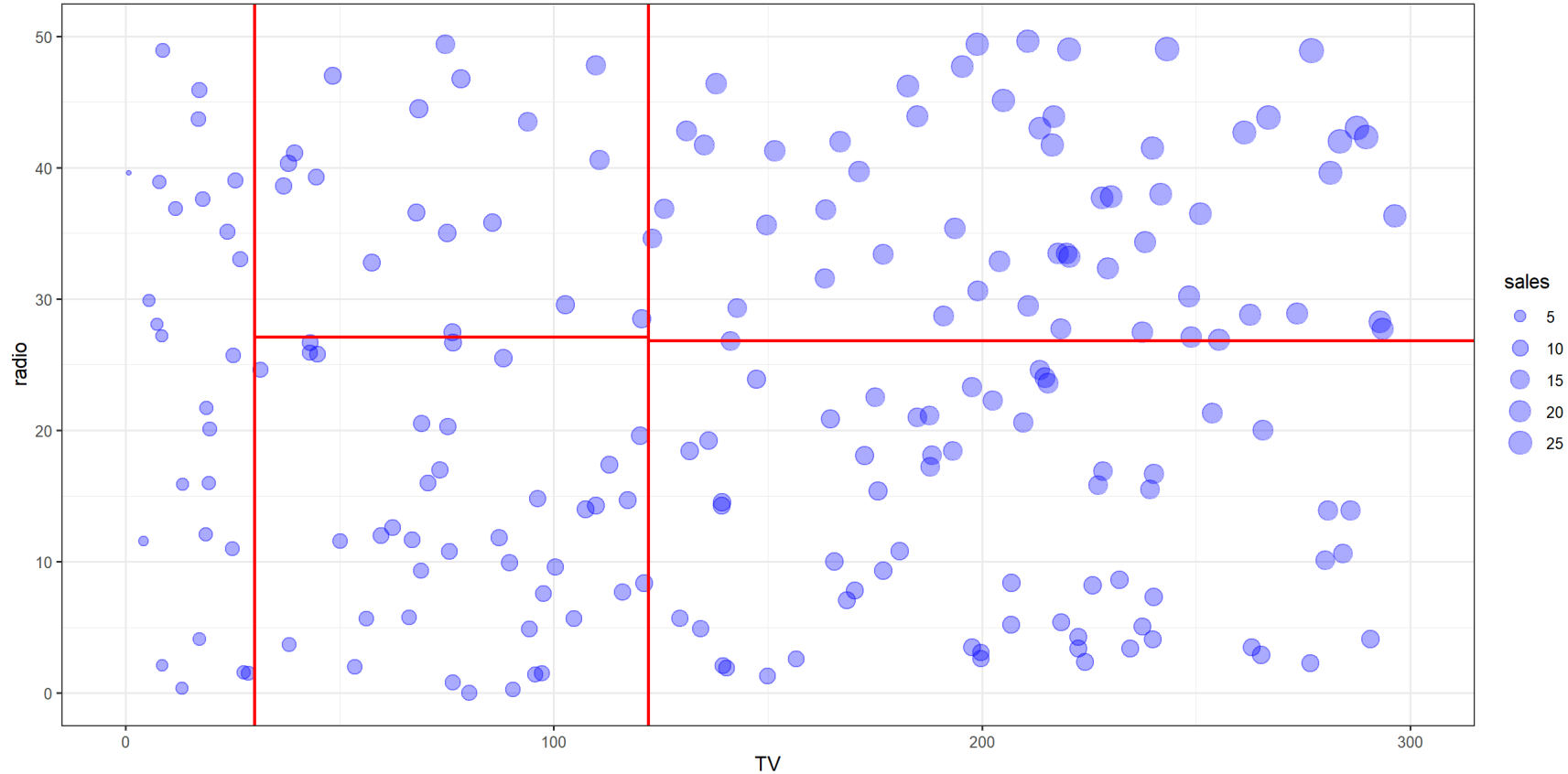
objetivo: criar regiões homogêneas.

Árvore de Regressão



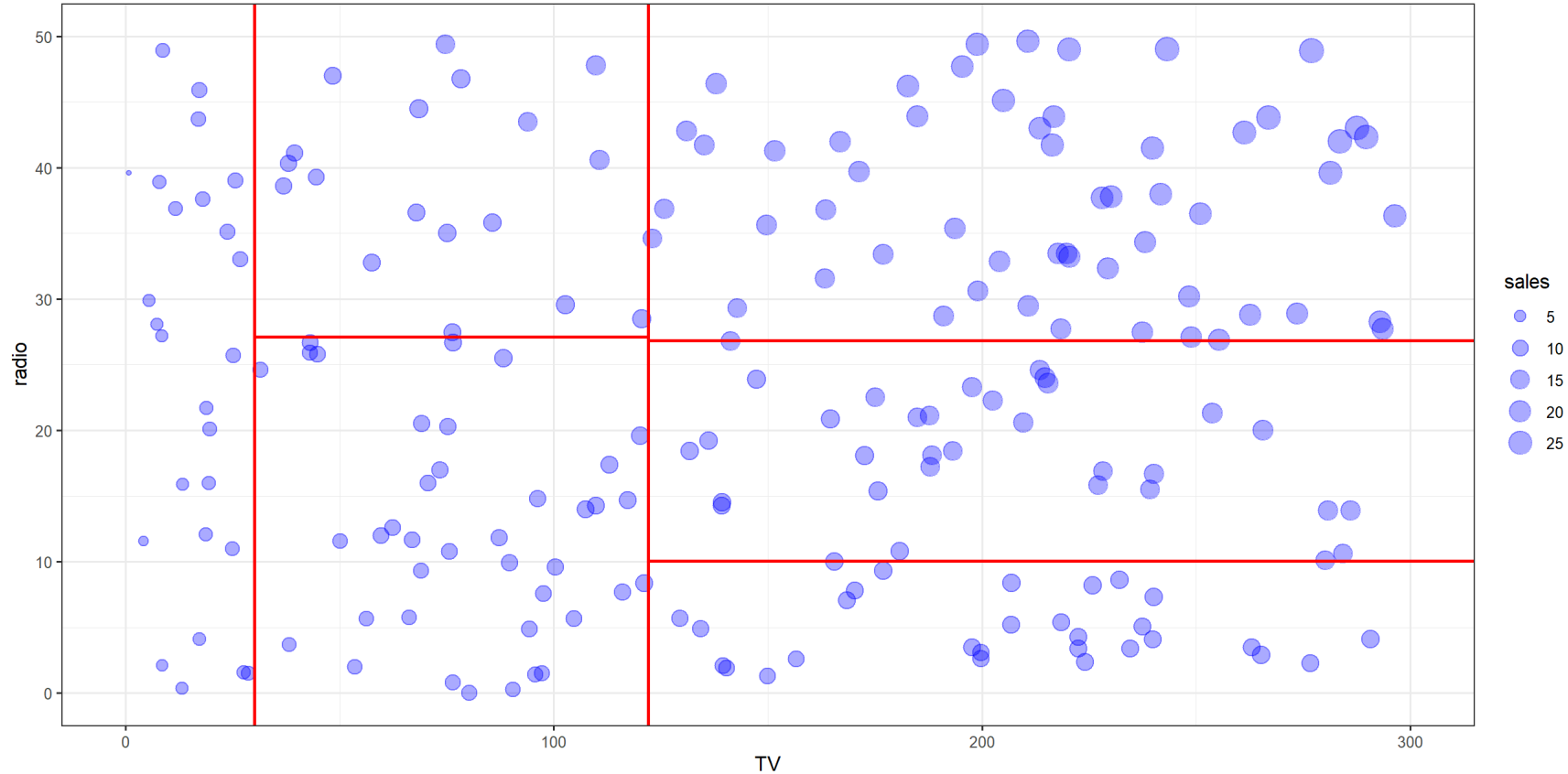
objetivo: criar regiões homogêneas.

Árvore de Regressão



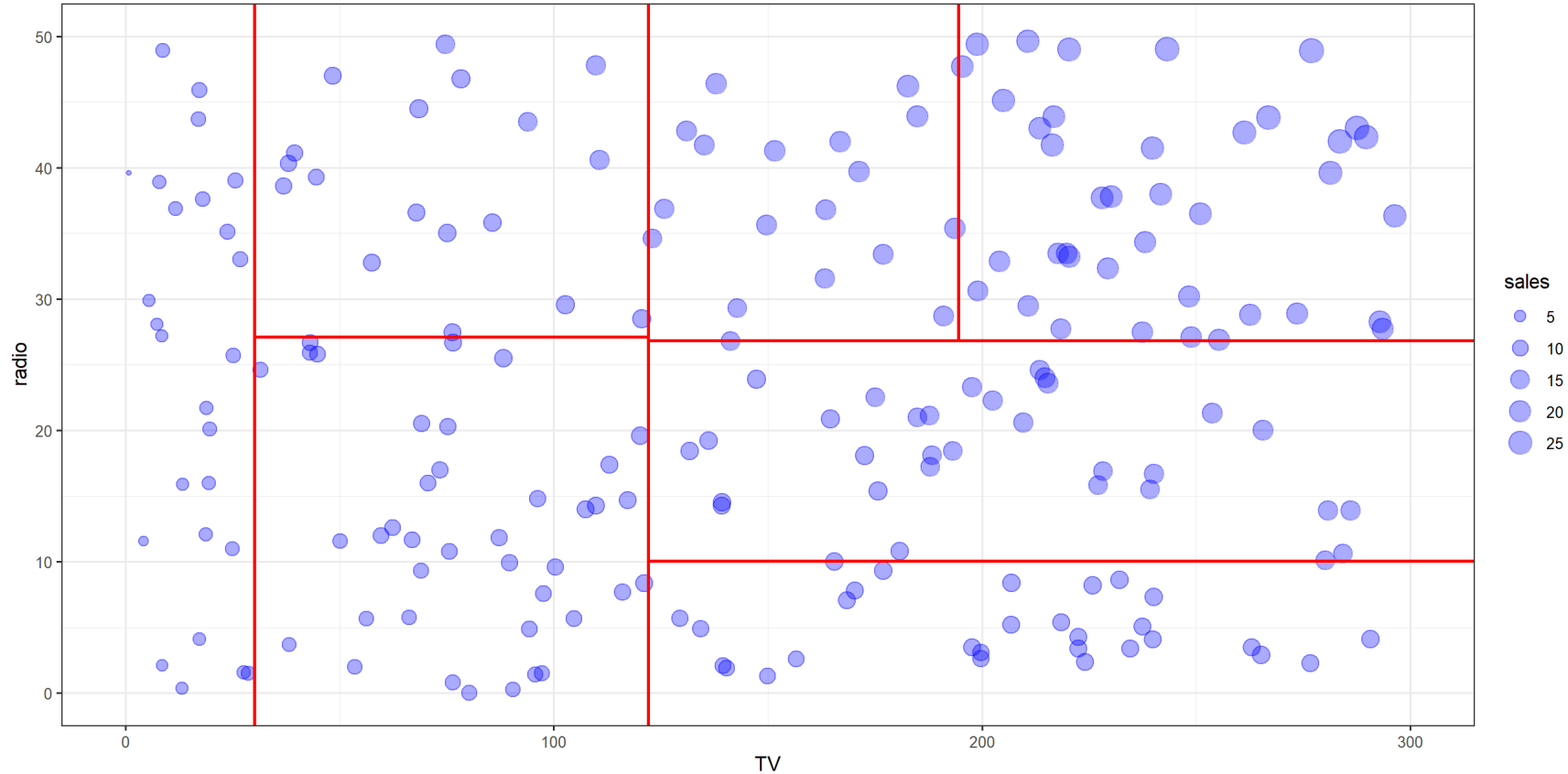
objetivo: criar regiões homogêneas.

Árvore de Regressão



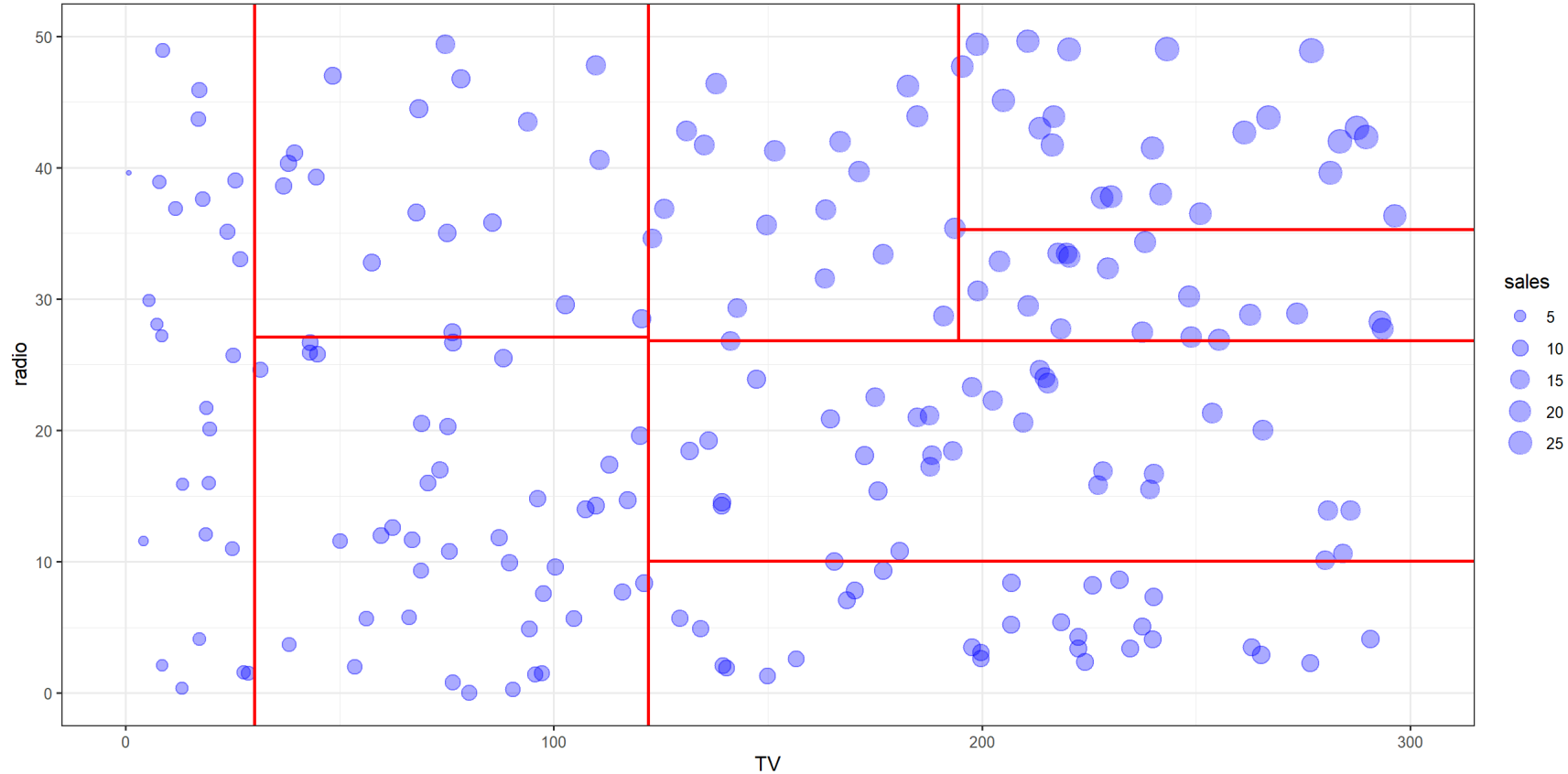
objetivo: criar regiões homogêneas.

Árvore de Regressão



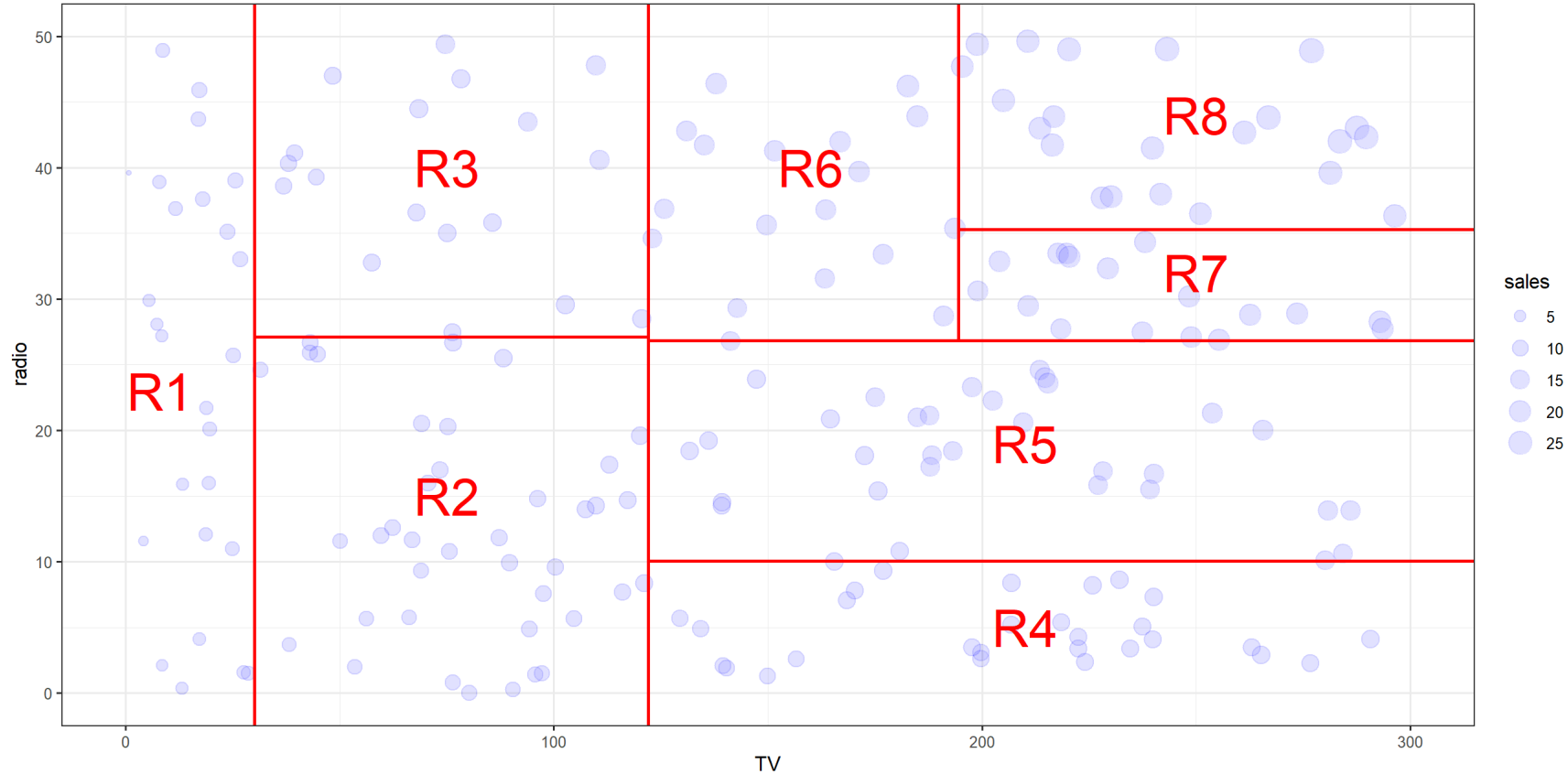
objetivo: criar regiões homogêneas.

Árvore de Regressão



objetivo: criar regiões homogêneas.

Árvore de Regressão



objetivo: criar regiões homogêneas.

Previsão via estratificação do espaço de preditoras

- Dividimos o espaço das preditoras $(X_1, X_2, \ldots, X_p)$ em J regiões distintas e disjuntas
- Para cada observação que *cair* na região R_i fazemos a previsão com base na média das observações pertencentes a R_i

O objetivo é encontrar caixas $R_1, R_2, \ldots, R_J$ de forma a minimizar a seguinte quantidade:

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Previsão via estratificação do espaço de preditoras

Para todo j e s, definimos os seguintes conjuntos

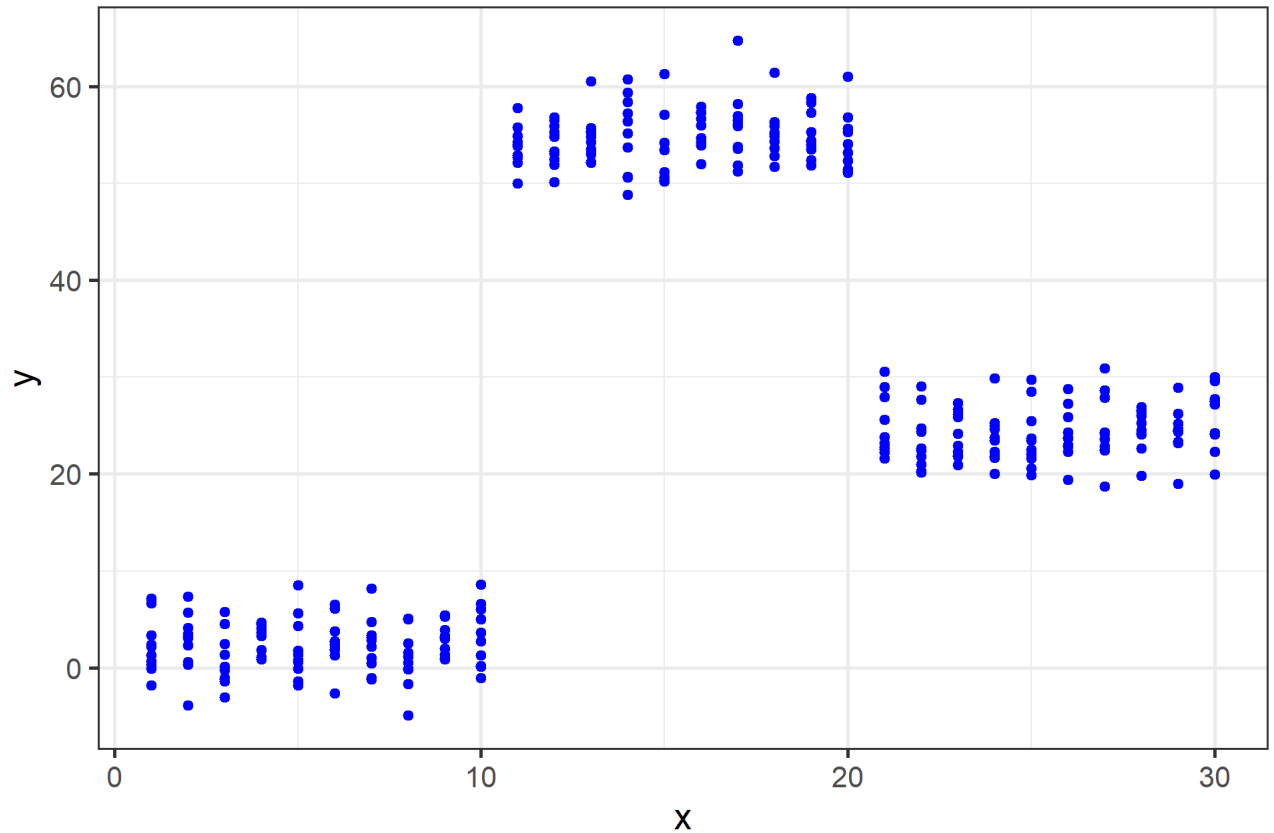
$$R_1(j,s) = \{X|X_j < s\} \text{ e } R_2(j,s) = \{X|X_j \geq s\},$$

em busca de j e s que minimizem a seguinte equação

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2.$$

Exemplo

$$Y = \beta_1 \mathbf{I}_{X \in R_1} + \beta_2 \mathbf{I}_{X \in R_2} + \beta_3 \mathbf{I}_{X \in R_3} + \epsilon, \text{ em que } \epsilon \sim N(0, 3^2)$$



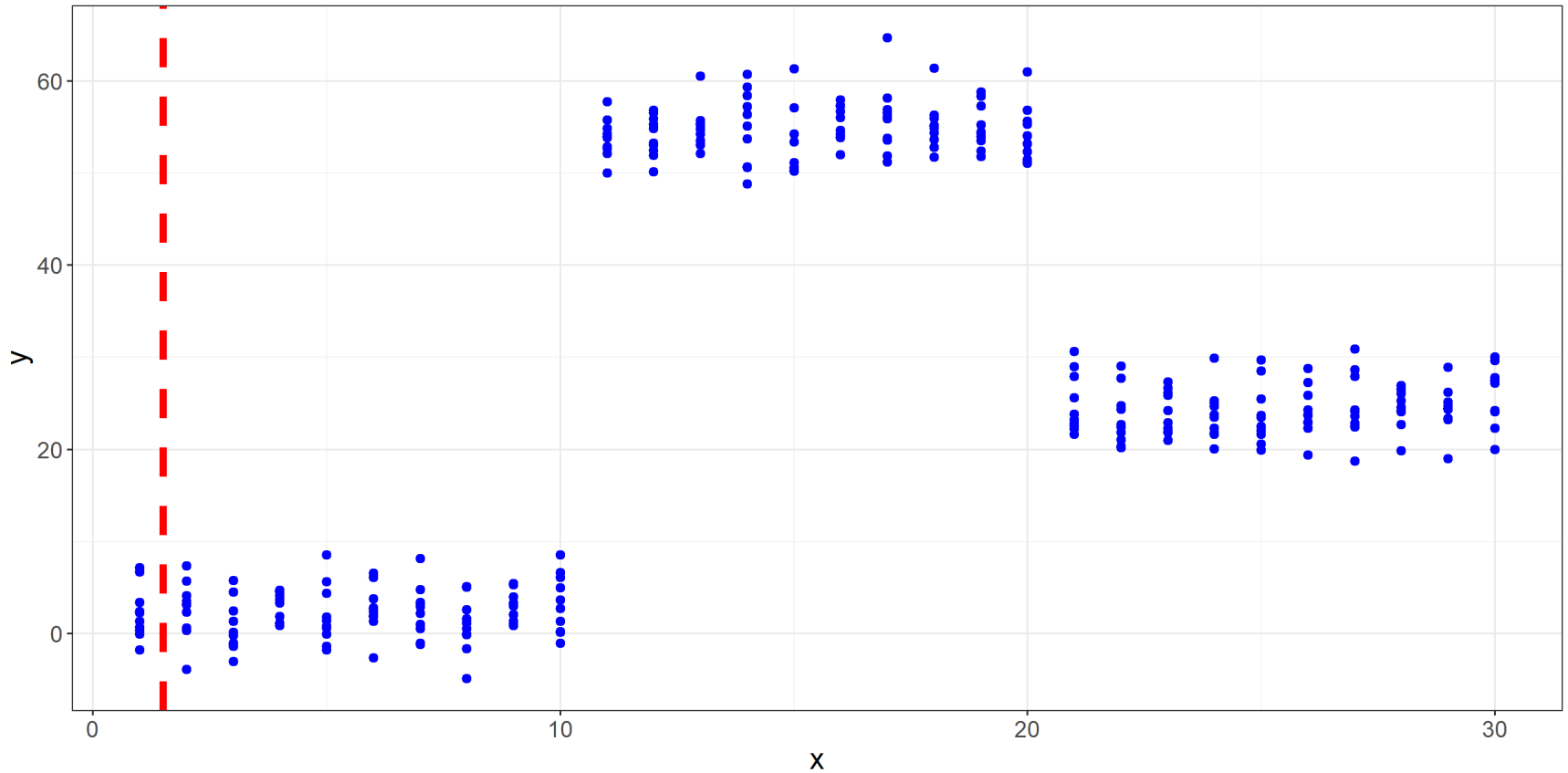
Exemplo

Para gerar os dados, utilizaremos o seguinte cenário:

```
set.seed(123)

beta <- rep(c(2, 55, 24), each = 100)

dados <- tibble(x = rep(1:30, each = 10),
                y = beta + rnorm(length(x), 0, 3))

dados %>%
  ggplot(aes(x, y)) +
    geom_point(color = "blue", size = 2)
```

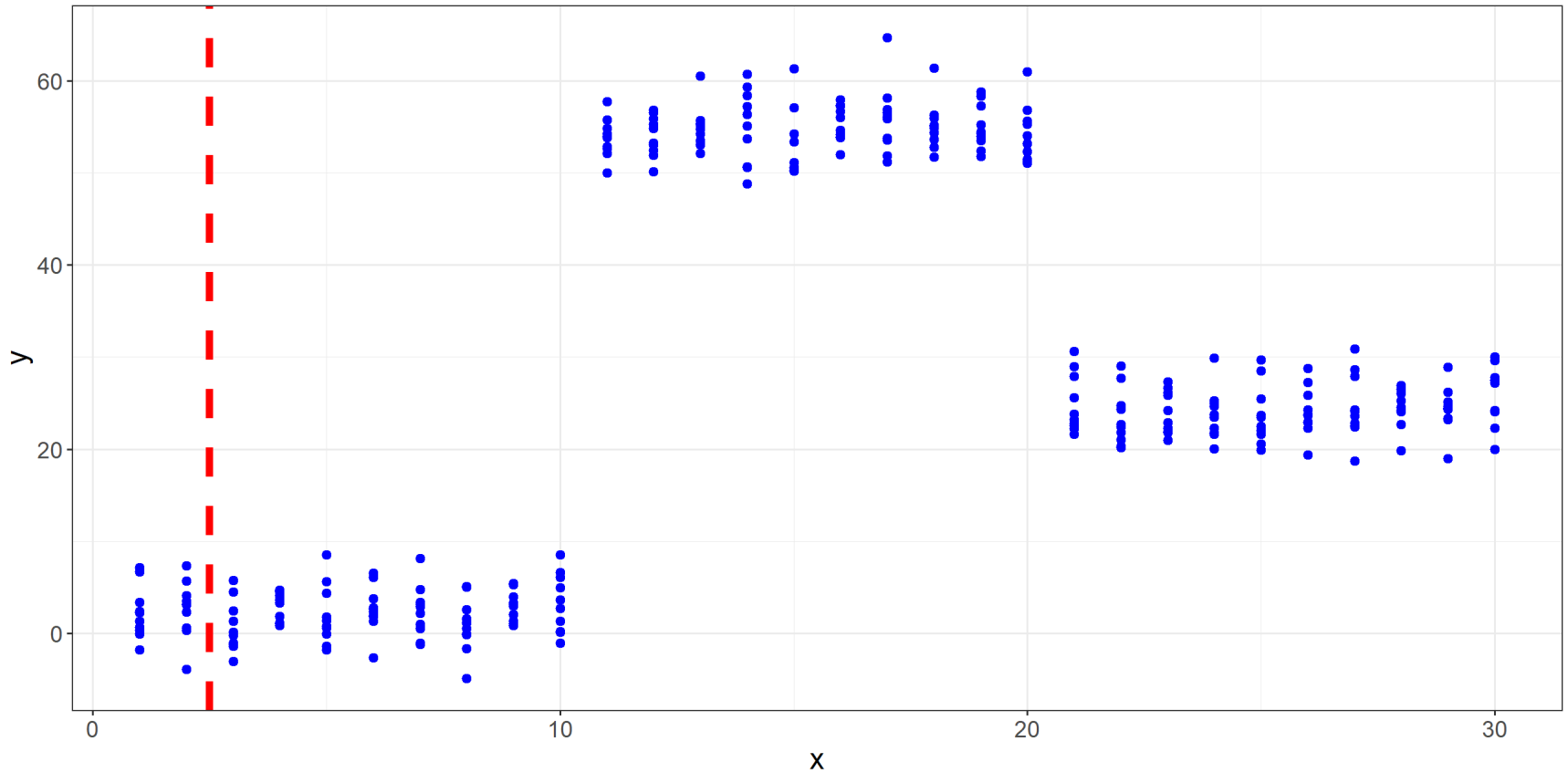
Exemplo

Vamos criar cortes, para todos os pontos intermediários de X, para avaliar a alteração na RSS.



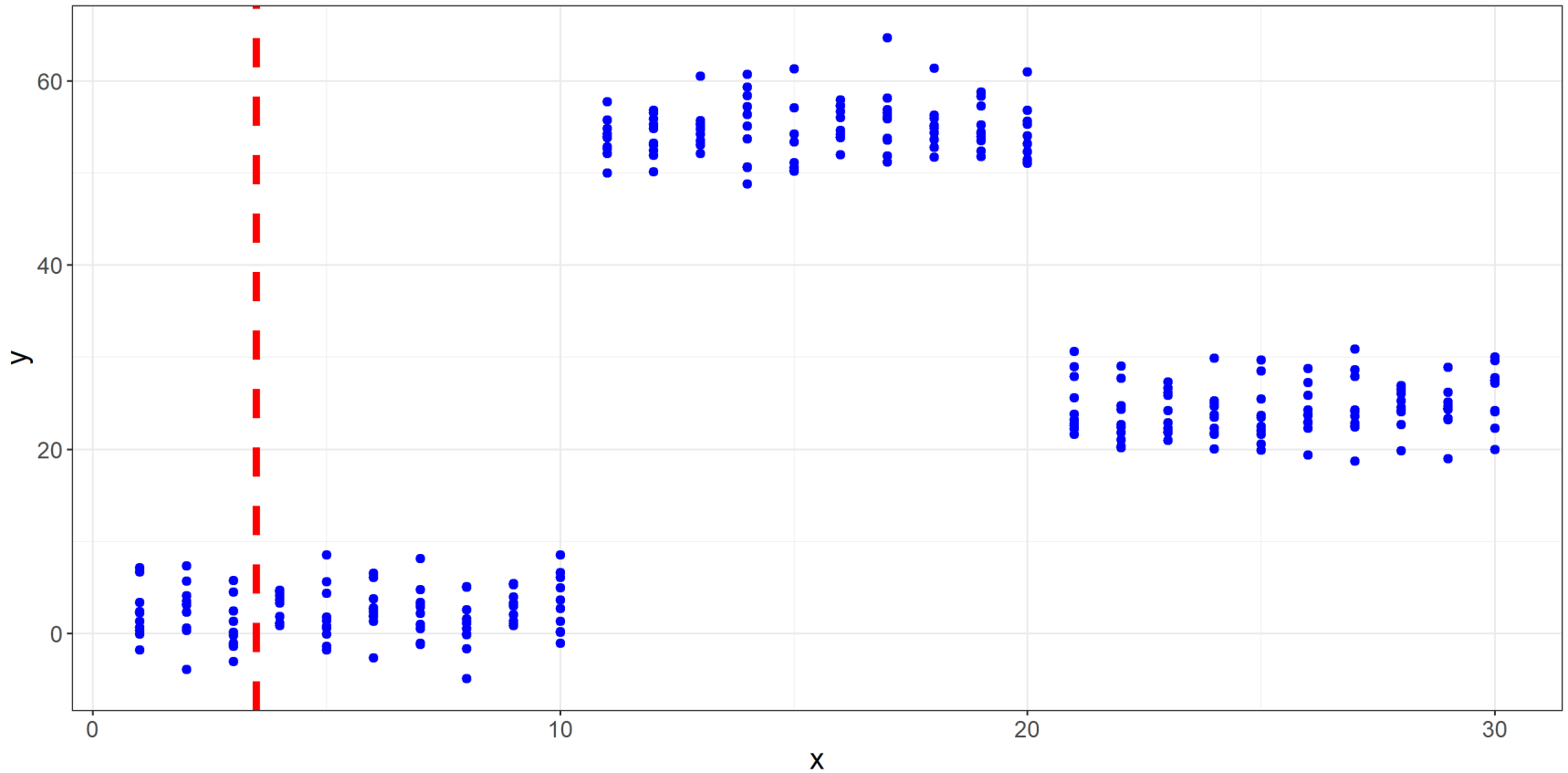
Exemplo

Vamos criar cortes, para todos os pontos intermediários de X, para avaliar a alteração na RSS.



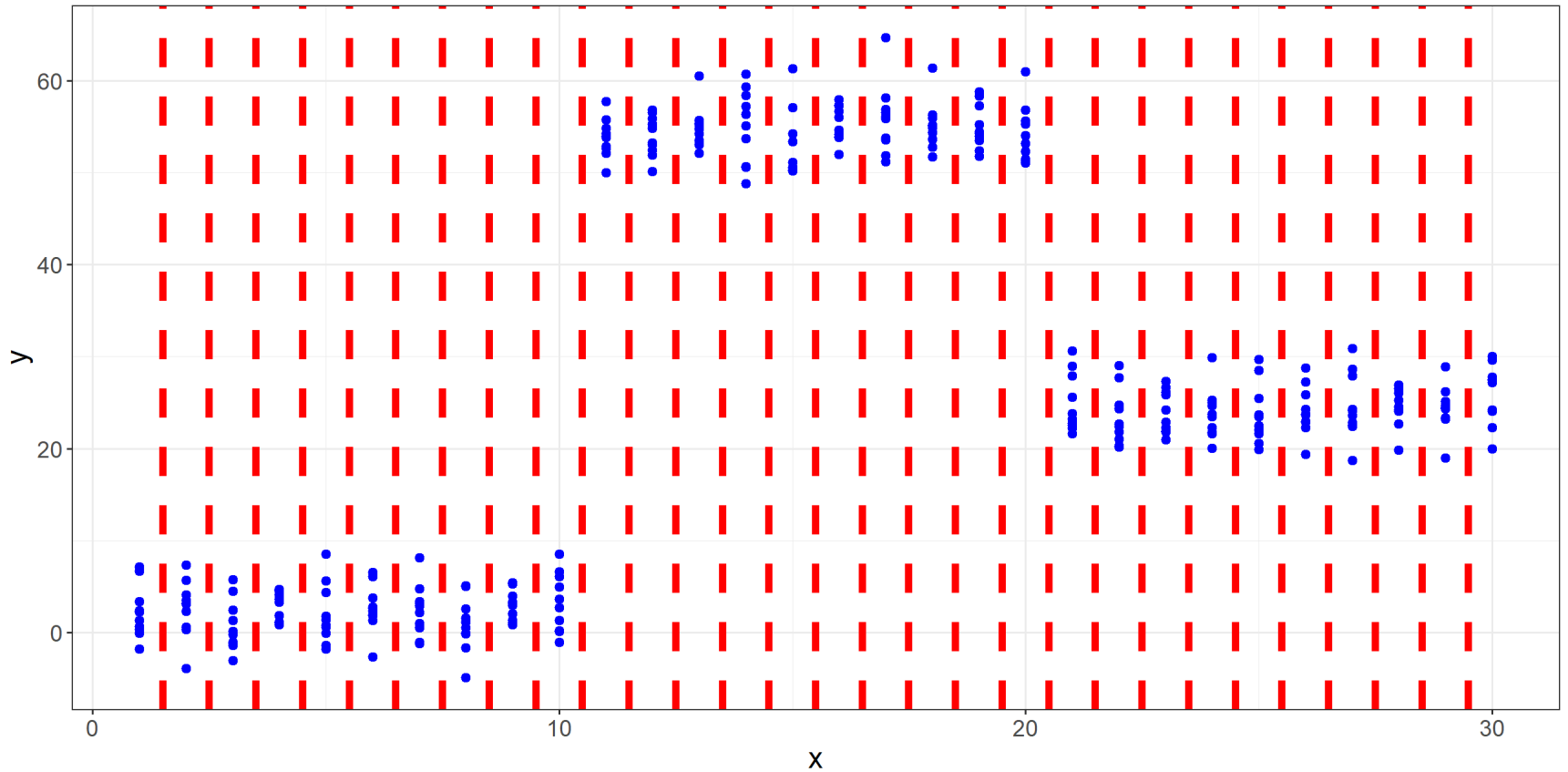
Exemplo

Vamos criar cortes, para todos os pontos intermediários de X, para avaliar a alteração na RSS.



Exemplo

Vamos criar cortes, para todos os pontos intermediários de X, para avaliar a alteração na RSS.



Exemplo

Vamos criar cortes, para todos os pontos intermediários de X, para avaliar a alteração na RSS.

- Como criar os pontos intermediários para os cortes? *Dica: unique(dados$x)*

```
cortes <- unique(dados$x)
cortes <- (cortes[-1] + cortes[-length(cortes)])/2
```

- Qual seria o desempenho do modelo *nulo* nesse caso?

```
RSS <- sum((dados$y - mean(dados$y))^2)
```

Exemplo

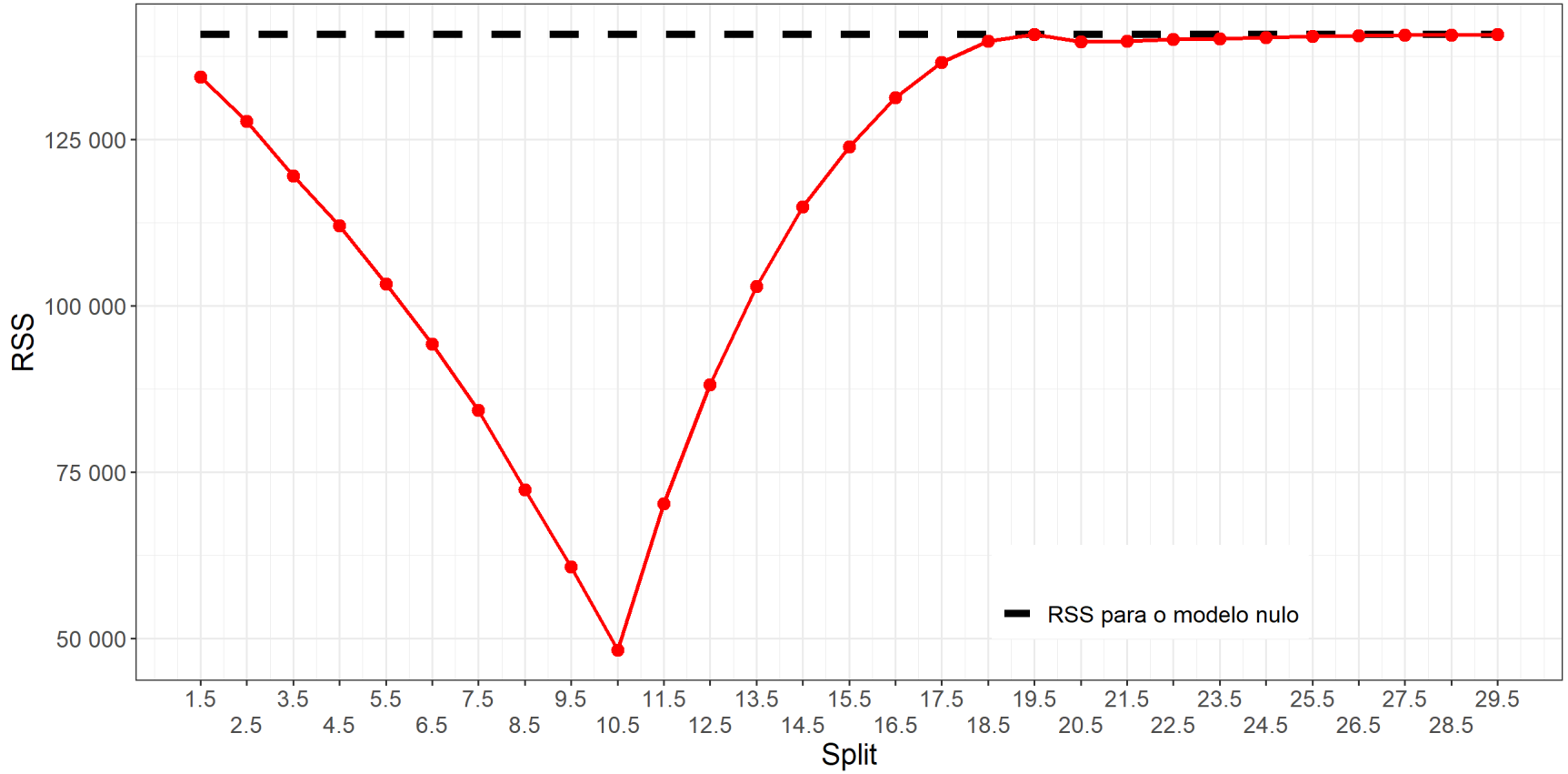
- Vamos criar a estrutura `resultados` para guardar os valores do RSS para cada corte:

```
resultados <- data.frame(c = cortes, RSS = NA)

for(i in 1:length(cortes)){

  RSS1 <- sum((dados$y[dados$x < cortes[i]] - mean(dados$y[dados$x < cortes[i]]))^2)
  RSS2 <- sum((dados$y[dados$x > cortes[i]] - mean(dados$y[dados$x > cortes[i]]))^2)

  resultados$RSS[i] <- RSS1 + RSS2

}
```

ou

```
rss <- function(c) {
  sum((dados$y[dados$x < c] - mean(dados$y[dados$x < c]))^2) +
    sum((dados$y[dados$x > c] - mean(dados$y[dados$x > c]))^2)
}

resultados %>%
  mutate(RSS = map_dbl(c, rss))
```

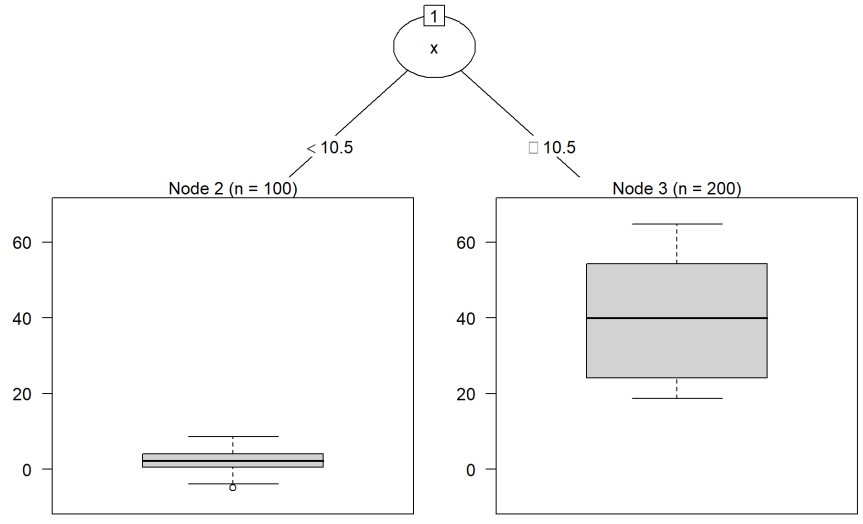
Exemplo

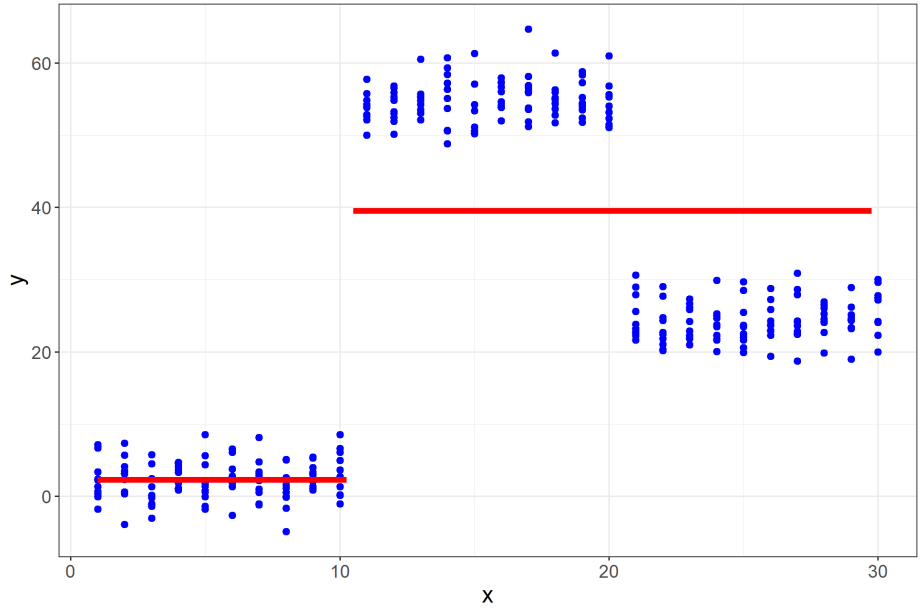


Qual será a primeira divisão/*split*?

Exemplo

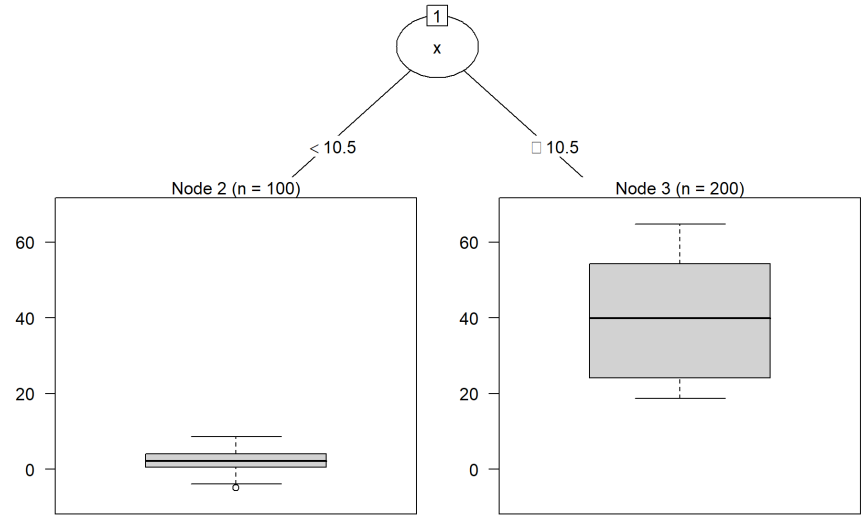
Para a primeira divisão, temos o seguinte cenário:

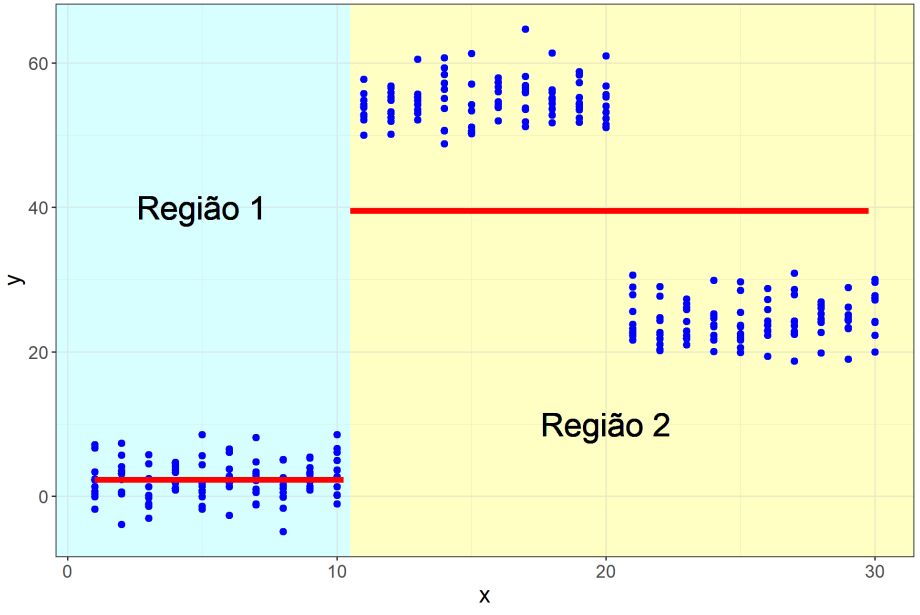




Exemplo

Para a primeira divisão, temos o seguinte cenário:





Exemplo

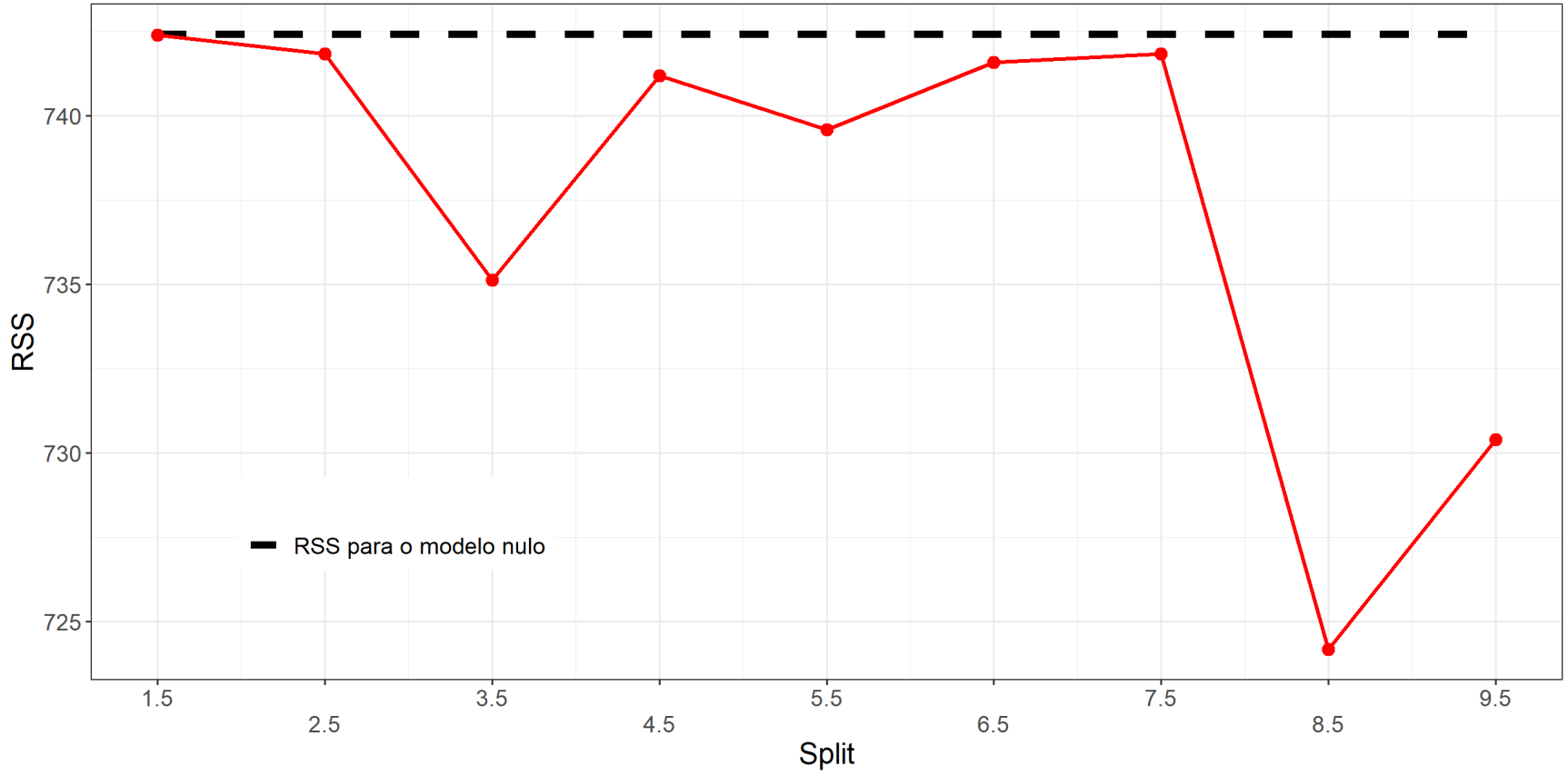
Agora repita o procedimento considerando dois blocos:

- Apenas as observações tais que $X < 10.5$
- Apenas as observações tais que $X > 10.5$

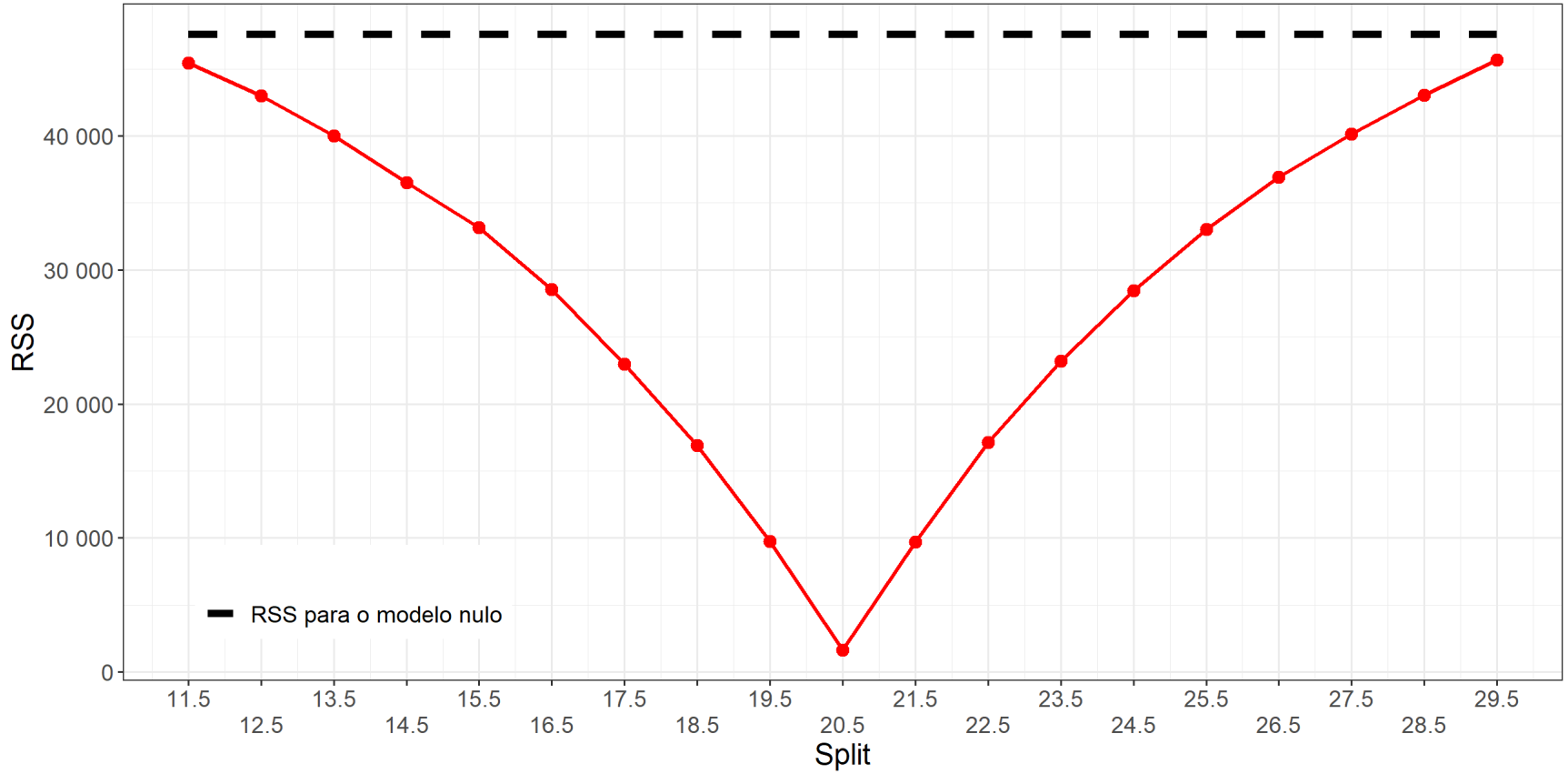
Você pode definir a medidas

```
# Primeira região
r1 <- dados %>%
        filter(x < 10.5)

# Segunda região
r2 <- dados %>%
        filter(x > 10.5)
```

Exemplo - $R_1 = X < 10.5$



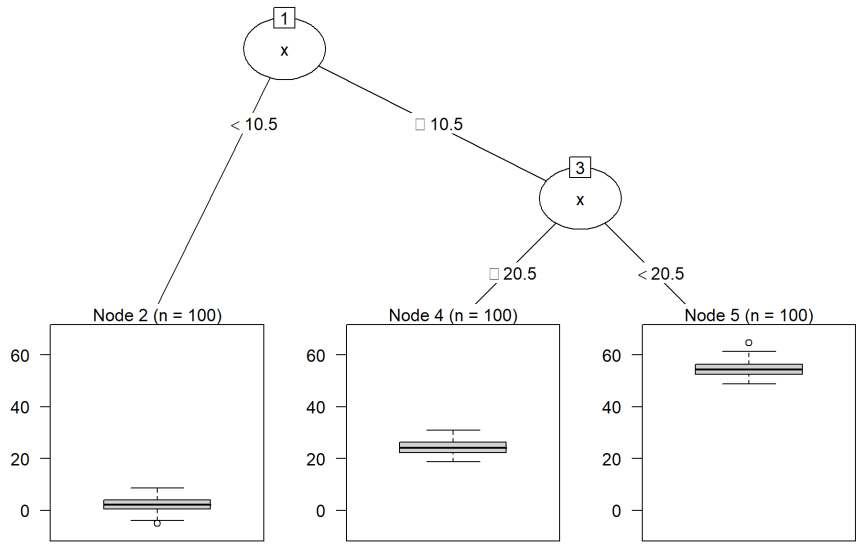
Exemplo - $R_2 = X > 10.5$

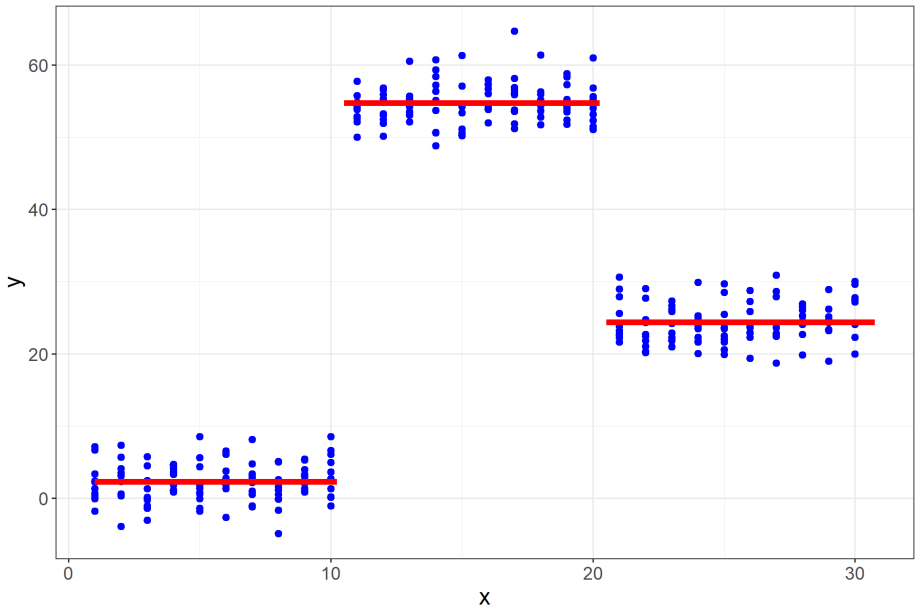


$$\text{err} = \sum (y - \bar{y})^2$$

Exemplo

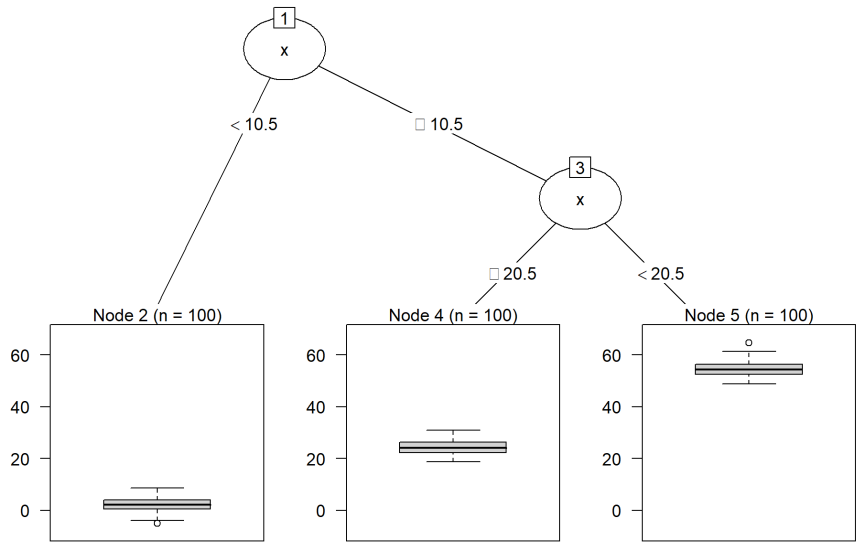
```
library(rpart)
library(partykit)

arvore <- rpart(y ~ x, dados)

plot_arvore <- as.party(arvore)

plot(plot_arvore)
```

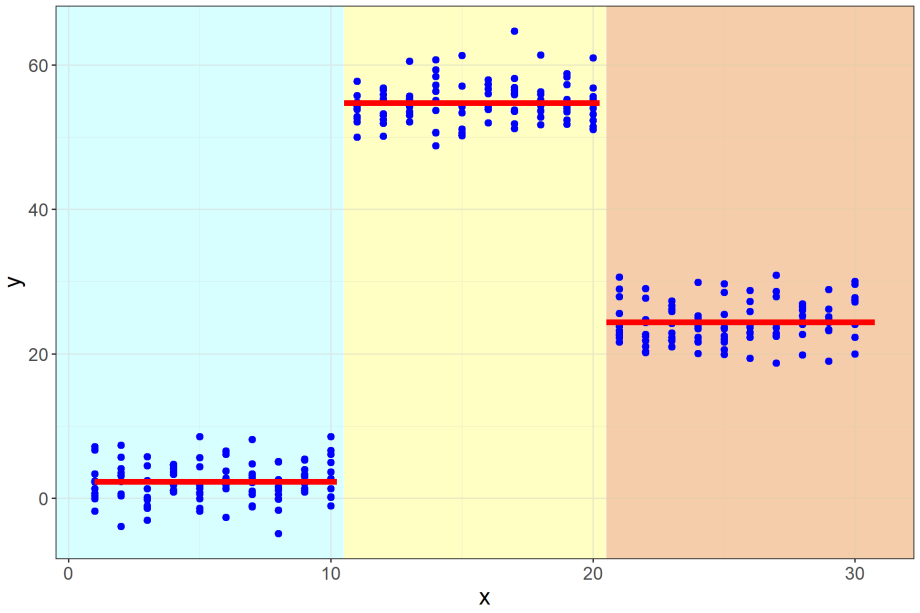




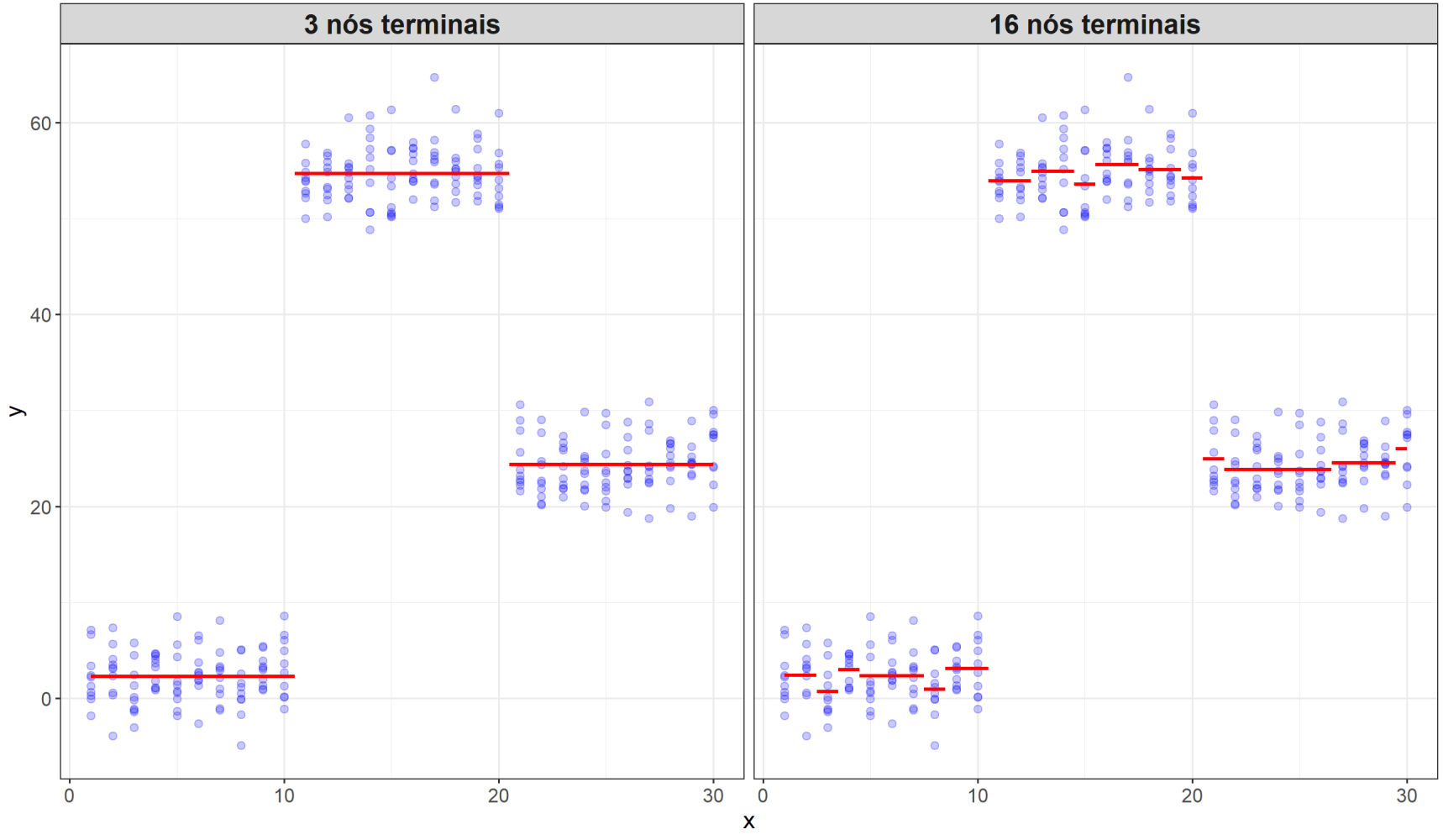
Exemplo

```
library(rpart)
library(partykit)

arvore <- rpart(y ~ x, dados)

plot_arvore <- as.party(arvore)

plot(plot_arvore)
```





Critério de parada



Critério de parada

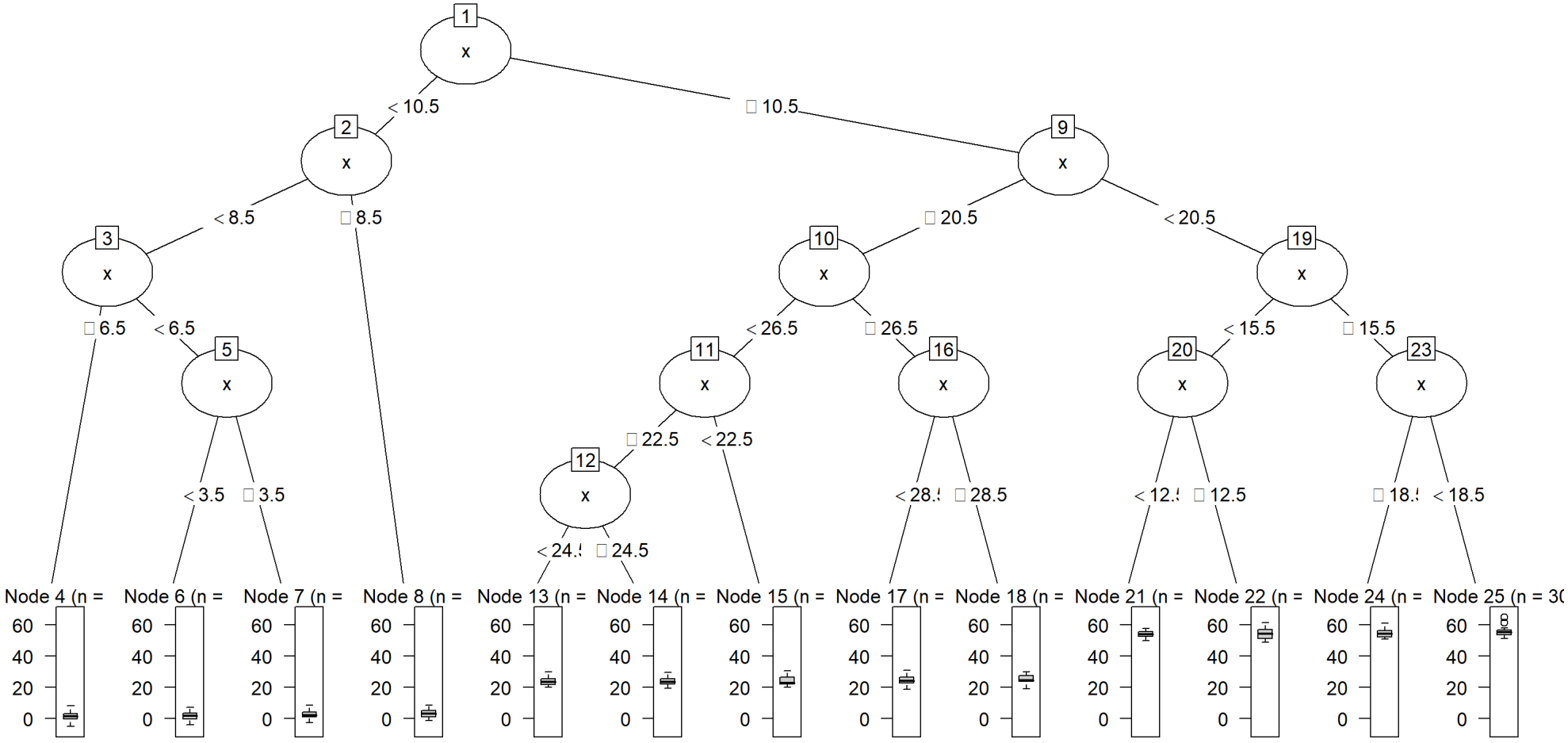
As funções utilizam critérios de parada para divisão dos nós. Por exemplo, verifique a documentação com `?rpart.control`

Alguns critérios são considerados para definir a complexidade da árvore:

- **minsplit:** número mínimo de observações para que se verifique a possibilidade de divisão
- **minbucket:** número mínimo de observações em um nó terminal
- **maxdepth:** profundidade máxima de uma árvore considerando a raíz como profundidade 0

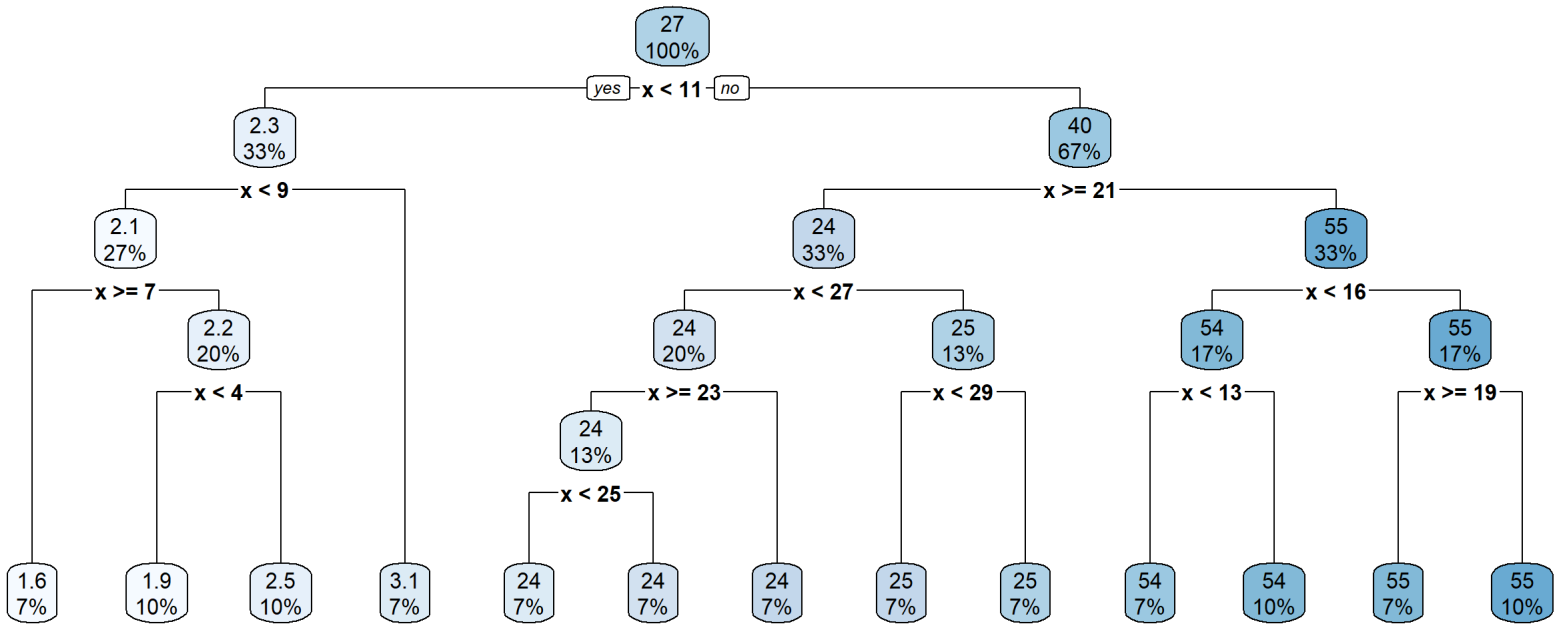
```
arvore <- rpart(y ~ x, dados, control = rpart.control(minsplit = 40, minbucket = 20, cp = 0))

arvore <- as.party(arvore)

plot(arvore)
```

Critério de parada



Critério de parada

```
library(rpart.plot)

arvore <- rpart(y ~ x, dados, control = rpart.control(minsplit = 40, minbucket = 20, cp = 0))

rpart.plot(arvore)
```



Para mais detalhes sobre as opções do pacote rpart.plot, veja http://www.milbo.org/doc/prp.pdf

Poda da Árvore

Poda da Árvore

O método apresentado anteriormente pode apresentar boa predição no conjunto de treinamento, mas é provável que haja um sobreajuste dos dados, levando a um fraco desempenho num conjunto de teste. Uma árvore menor, com menos divisões, pode reduzir a variância e melhorar a interpretação a custo de um aumento no viés.

Para cada valor de α há uma subárvore correspondente $\mathrm{T} \subset \mathrm{T}_0$

$$\sum_{m=1}^{|\mathrm{T}|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|\mathrm{T}|,$$

em que $|\mathrm{T}|$ indica o número de nós terminais da árvore T.

Algoritmo para Construção de uma Árvore[1]

1. Utilize divisão binária recursiva para crescer uma árvore no conjunto de treinamento, parando apenas quando cada nó terminal tiver menos que um número mínimo de observações.

2. Aplique *cost complexity pruning* a árvore para obter uma sequência de melhores subárvores em função de α.

3. Utilize validação cruzada com k lotes para escolher α. Isto é, divida as observações de treinamento em k lotes. Para cada $k = 1, \ldots, K$:

   a) Repita os passos 1 e 2 em todos dados, com exceção dos dados no k-ésimo lote.

   b) Avalie o erro de predição nos dados do k-ésimo lote em função de α.

   Calcule a média dos resultados para cada valor de α e escolha α que minimiza o erro de predição.

4. Retorne a subárvore no passo 2 que corresponda ao valor escolhido de α.

[1] retiradas do livro *An Introduction to Statistical Learning with Applications in R*.

Cp

```
arvore <- rpart(y ~ x, dados, control = rpart.control(cp = 0.0002))
```

```
arvore$cptable
```

CP	nsplit	rel error	xerror	xstd
0.6567972	0	1.0000000	1.0073460	0.0432838
0.3263065	1	0.3432028	0.3472893	0.0163484
0.0002103	2	0.0168964	0.0172533	0.0014131
0.0002000	3	0.0166861	0.0180810	0.0014928

cp: complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. For instance, with anova splitting, this means that the overall R-squared must increase by cp at each step. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially,the user informs the program that any split which does not improve the fit by cp will likely be pruned off by cross-validation, and that hence the program need not pursue it.

Exemplo

```
arvore$cptable
```

CP	nsplit	rel error	xerror	xstd
0.6567972	0	1.0000000	1.0073460	0.0432838
0.3263065	1	0.3432028	0.3472893	0.0163484
0.0002103	2	0.0168964	0.0172533	0.0014131
0.0002000	3	0.0166861	0.0180810	0.0014928

```
rel1 <- sum((dados$y - mean(dados$y))^2)

rel2 <- sum((dados$y[dados$x < 10.5] - mean(dados$y[dados$x < 10.5]))^2) +
        sum((dados$y[dados$x >= 10.5] - mean(dados$y[dados$x >= 10.5]))^2)

rel2/rel1 # relative error
```

```
## [1] 0.3432028
```

```
1 - rel2/rel1
```

```
## [1] 0.6567972
```

Exemplo

```
arvore$cptable
```

CP	nsplit	rel error	xerror	xstd
0.6567972	0	1.0000000	1.0073460	0.0432838
0.3263065	1	0.3432028	0.3472893	0.0163484
0.0002103	2	0.0168964	0.0172533	0.0014131
0.0002000	3	0.0166861	0.0180810	0.0014928

```
rel1 <- sum((dados$y - mean(dados$y))^2)

rel3 <- sum((dados$y[dados$x < 10.5] - mean(dados$y[dados$x < 10.5]))^2) +
          sum((dados$y[dados$x >= 20.5] - mean(dados$y[dados$x >= 20.5]))^2) +
          sum((dados$y[dados$x  < 20.5 & dados$x > 10.5] -
                 mean(dados$y[dados$x  < 20.5 & dados$x > 10.5]))^2)

rel3/rel1 # relative error
```

```
## [1] 0.01689639
```

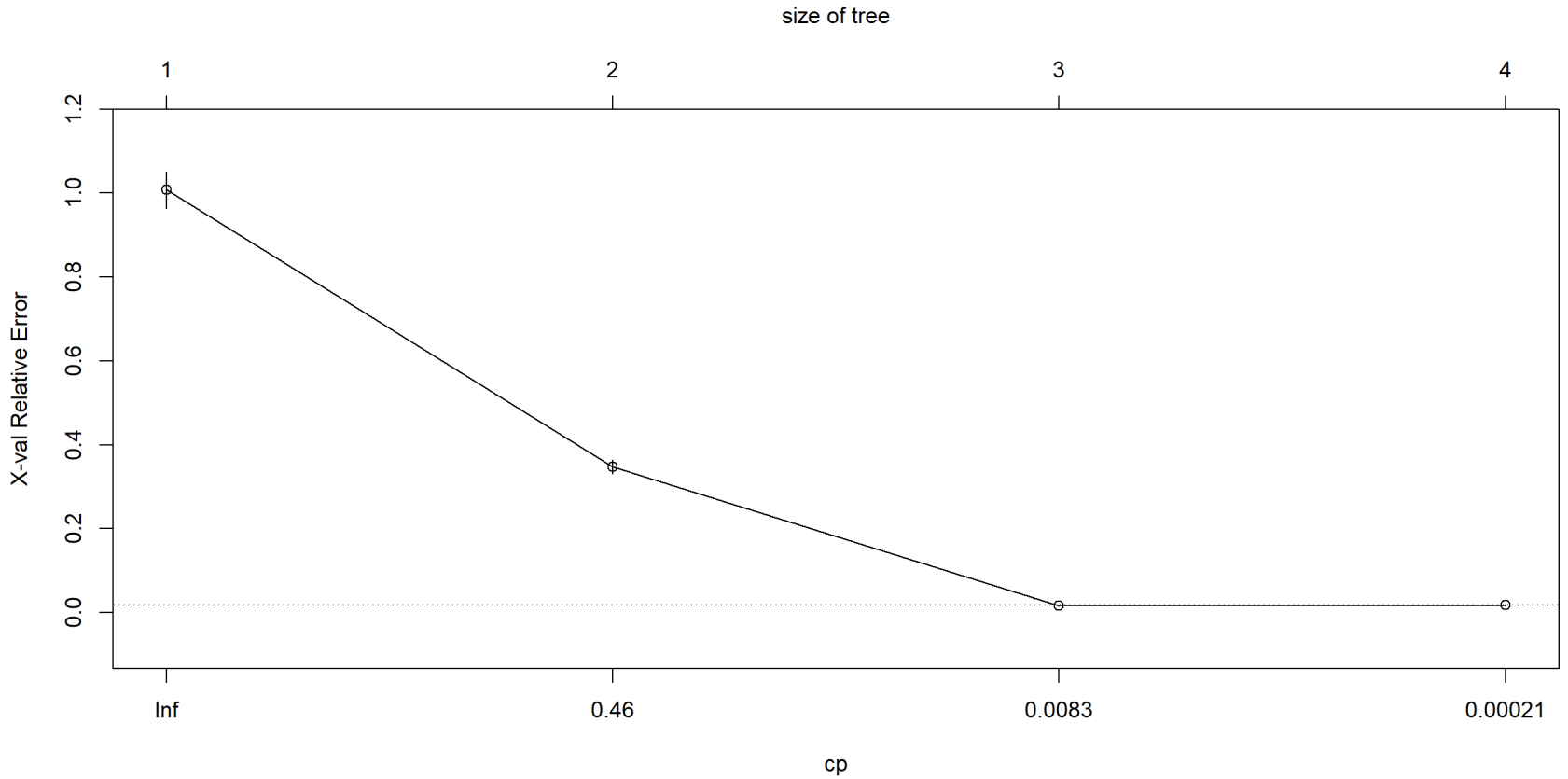
```
rel2/rel1 - rel3/rel1
```

```
## [1] 0.3263065
```

Cp

A função a seguir apresentar um gráfico dos resultados da validação cruzada obtida a partir do objeto `rpart`.

```
plotcp(arvore)
```



Aplicação

Credit[1]

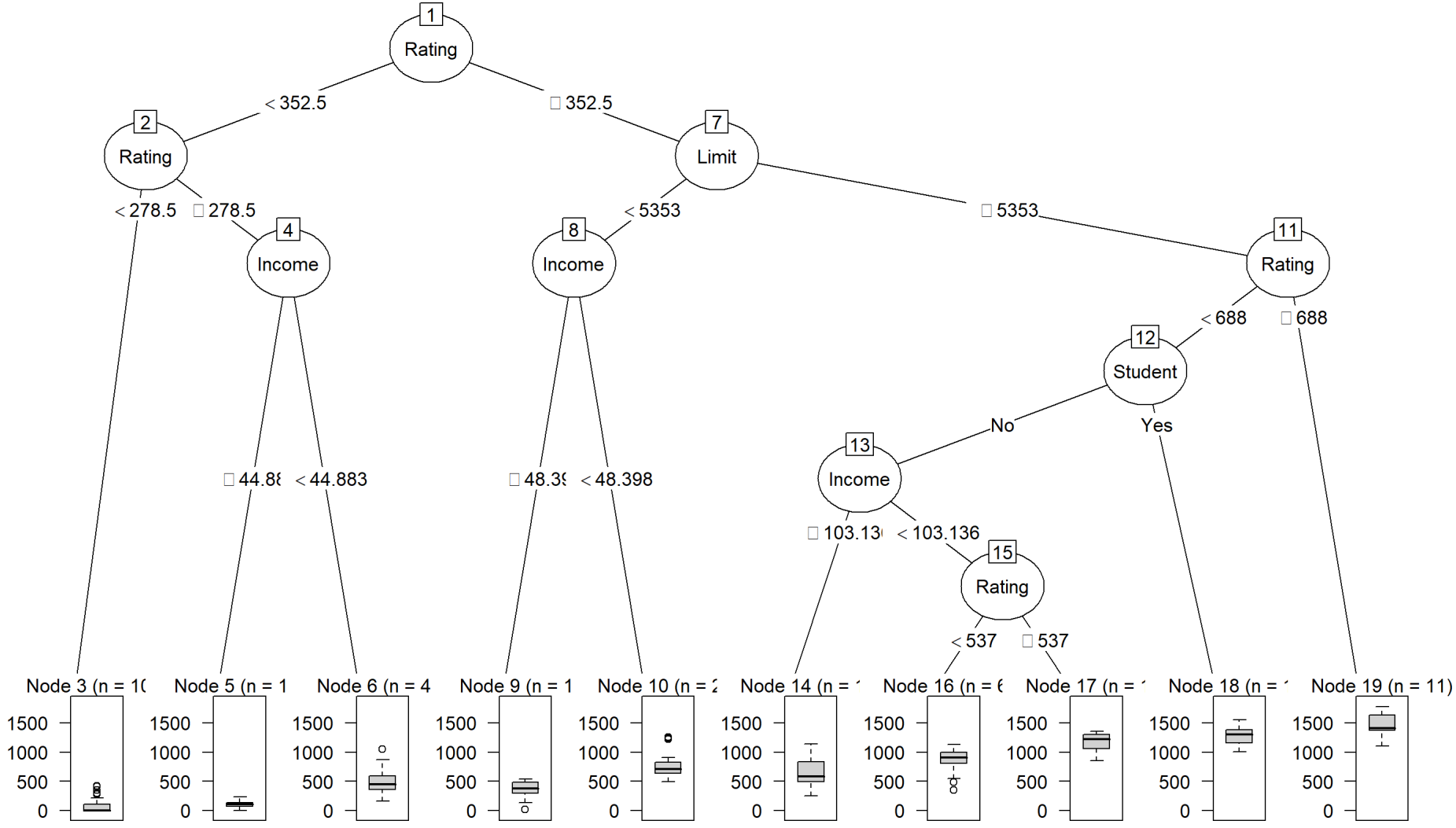
- **ID**
- **Income**: renda (em $10,000)
- **Limit**: limite de crédito
- **Rating**: rating de crédito
- **Cards**: número de cartões de crédito
- **Age**: idade em anos
- **Education**: anos de escolaridade
- **Gender**: Male / Female
- **Student**: Yes / No
- **Married**: Yes / No
- **Ethnicity** : African American / Asian / Caucasian
- **Balance**: saldo médio do cartão de crédito

[1] dados contidos no pacote **ISLR**.

Credit

```
library(rsample)
library(rpart)
library(partykit)
library(ISLR)

set.seed(21)

splits <- initial_split(Credit, prop = .8)

tr   <- training(splits)
test <- testing(splits)

arvore <- rpart(Balance ~ . -ID, data = tr)

arvore <- as.party(arvore)

plot(arvore)
```

Credit

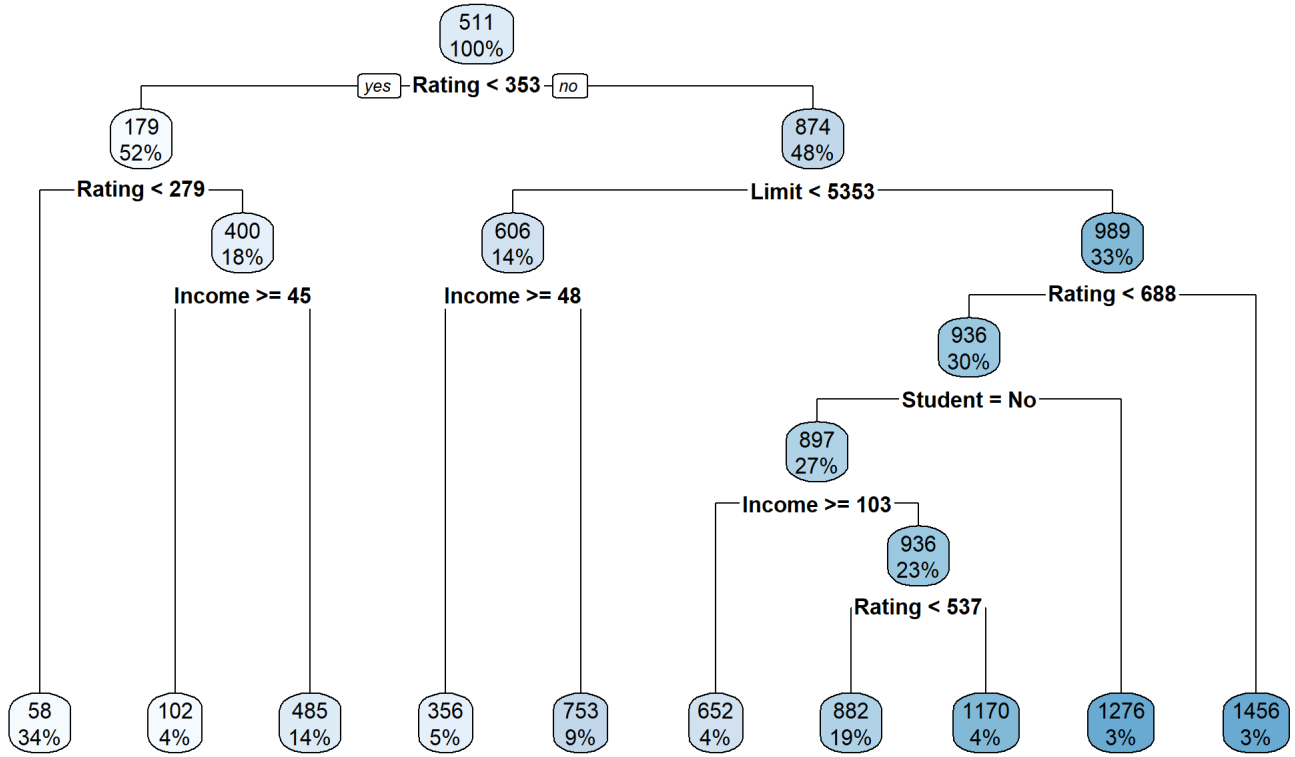

1
Rating
< 352.5
352.5
2
Rating
7
Limit
< 278.5
278.5
4
Income
8
Income
< 5353
5353
11
Rating
< 688
688
12
Student
No
Yes
13
Income
15
Rating
103.13
< 103.136
< 537
537
44.8
< 44.883
48.3
< 48.398
Node 3 (n = 1
Node 5 (n = 1
Node 6 (n = 4
Node 9 (n = 1
Node 10 (n = 2
Node 14 (n =
Node 16 (n =
Node 17 (n =
Node 18 (n =
Node 19 (n = 11)
1500
1000
500
0

Credit

```
library(rsample)
library(rpart)
library(partykit)
library(ISLR)

set.seed(21)

splits <- initial_split(Credit, prop = .8)

tr   <- training(splits)
test <- testing(splits)

arvore <- rpart(Balance ~ . -ID, data = tr)

rpart.plot(arvore, roundint = FALSE)
```

Credit

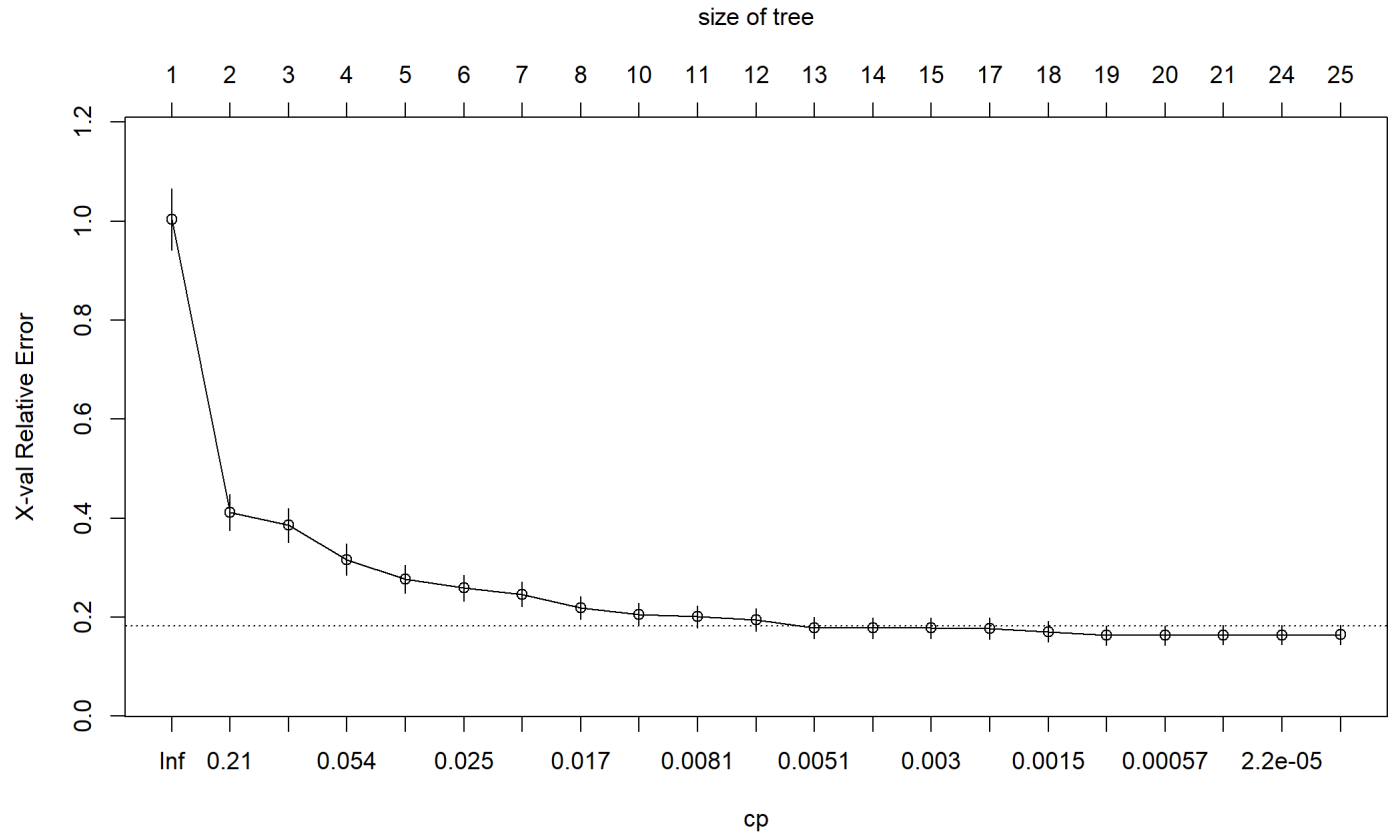
valor predito para nó / porcentagem pertencente ao nó



Credit

Insper

```
set.seed(202)

arvore <- rpart(Balance ~ . -ID, data = tr, control = rpart.control(xval = 10, cp = 0))

plotcp(arvore)
```



Credit

```
arvore$cptable
```

```
##              CP nsplit  rel error    xerror       xstd
## 1  5.999647e-01      0 1.00000000 1.0038235 0.06171279
## 2  7.326264e-02      1 0.40003534 0.4123000 0.03650042
## 3  6.959009e-02      2 0.32677270 0.3856467 0.03470900
## 4  4.137211e-02      3 0.25718260 0.3165198 0.03128219
## 5  2.625693e-02      4 0.21581049 0.2770190 0.02878403
## 6  2.303990e-02      5 0.18955356 0.2591704 0.02624438
## 7  1.997430e-02      6 0.16651366 0.2462795 0.02495521
## 8  1.376088e-02      7 0.14653935 0.2192381 0.02323385
## 9  8.125944e-03      9 0.11901760 0.2056220 0.02278371
## 10 7.995077e-03     10 0.11089166 0.2007967 0.02239886
## 11 6.231397e-03     11 0.10289658 0.1947724 0.02234241
## 12 4.242727e-03     12 0.09666518 0.1786618 0.02142486
## 13 3.347656e-03     13 0.09242246 0.1779581 0.02110724
## 14 2.725947e-03     14 0.08907480 0.1783227 0.02102300
## 15 2.671424e-03     16 0.08362290 0.1771199 0.02100557
## 16 8.328683e-04     17 0.08095148 0.1710620 0.02066650
## 17 5.925655e-04     18 0.08011861 0.1635298 0.01965982
## 18 5.412387e-04     19 0.07952605 0.1635494 0.01966195
## 19 8.370811e-05     20 0.07898481 0.1640419 0.01966582
## 20 5.925521e-06     23 0.07873122 0.1643790 0.01966589
## 21 0.000000e+00     24 0.07872530 0.1643980 0.01966649
```

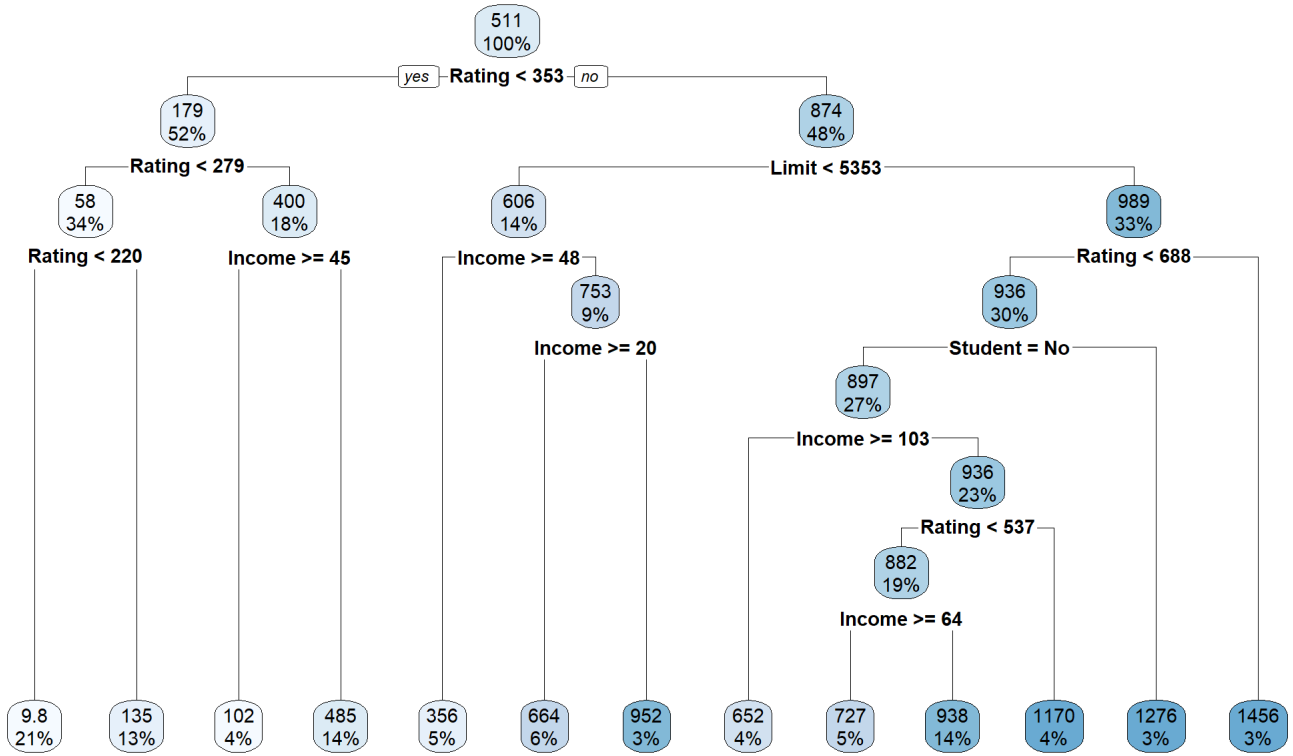
Credit

```
cp_ot <- arvore$cptable[which.min(arvore$cptable[,"xerror"]),"CP"]

cp_ot <- arvore$cptable %>%
          as_tibble() %>%
            filter(xerror == min(xerror))

# OU std

corte <- arvore$cptable %>%
          as_tibble() %>%
          filter(xerror == min(xerror)) %>%
          transmute(corte = xerror + xstd)


cp_ot <- arvore$cptable %>%
          as_tibble() %>%
          filter(xerror <= corte[[1]])
```

Credit

Insper

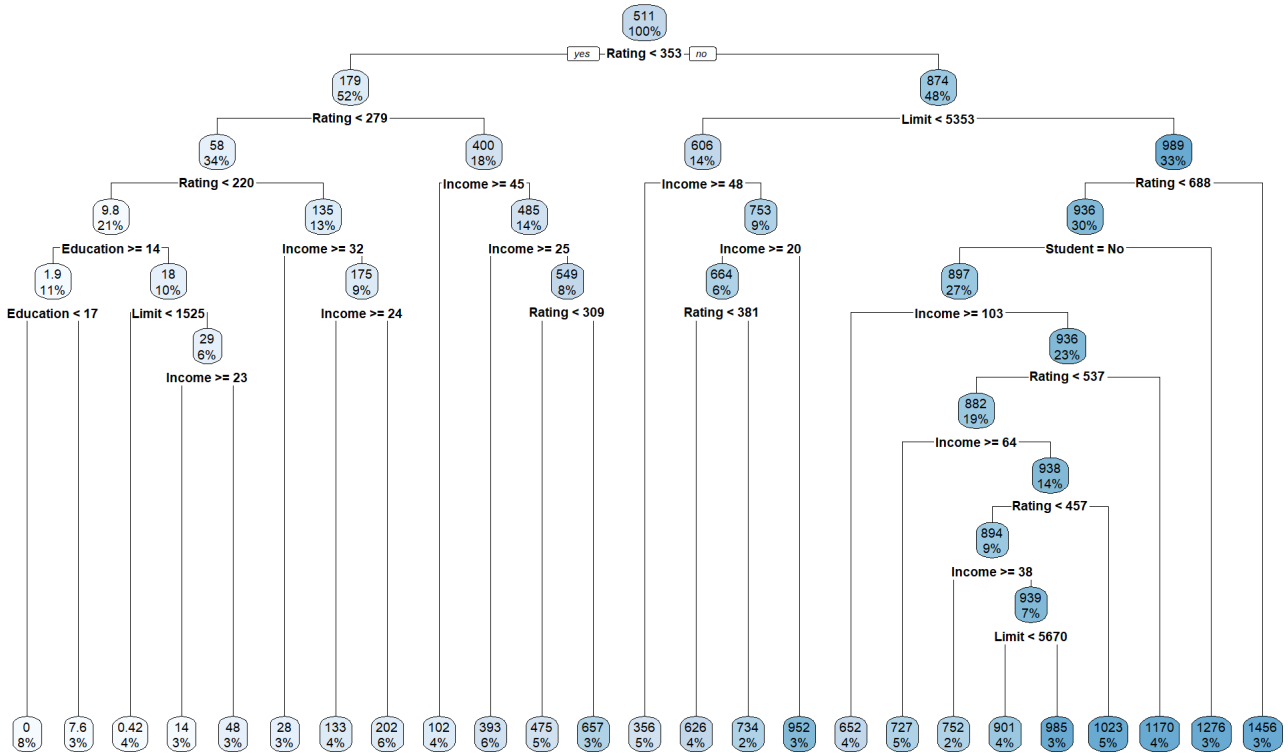
```
poda1 <- prune(arvore, cp = cp_ot$CP[1])

rpart.plot(poda1, roundint = FALSE)
```



Credit

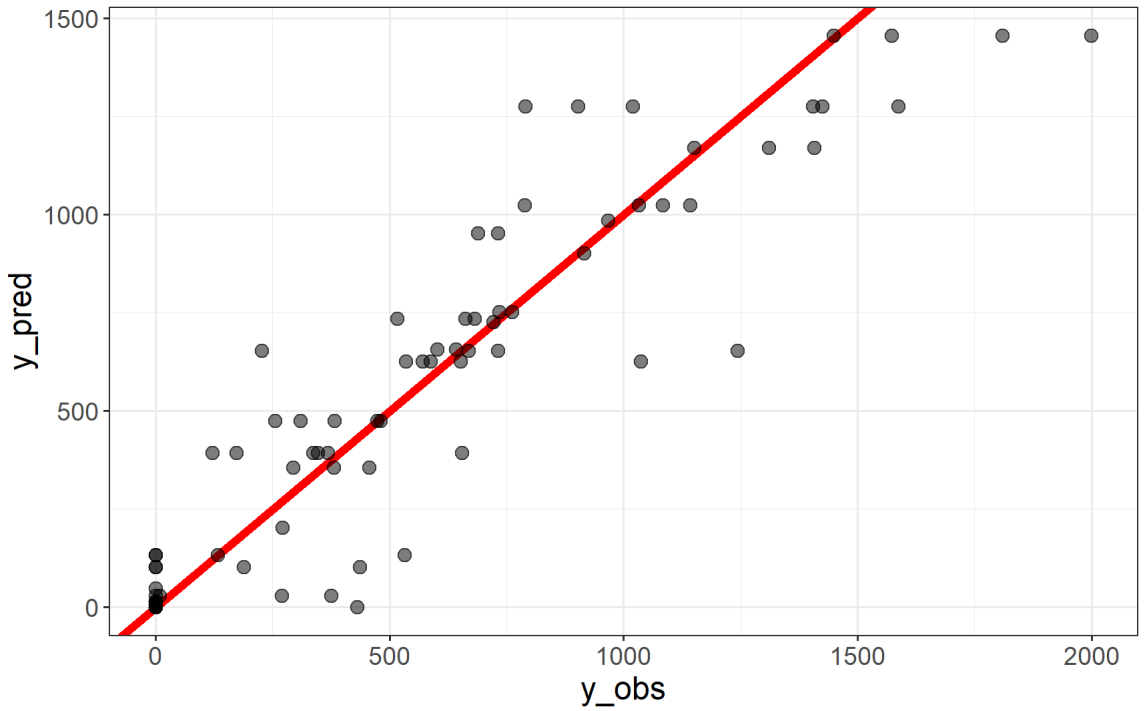
Insper

```
poda1 <- prune(arvore, cp = 0)

rpart.plot(poda1, roundint = FALSE)
```

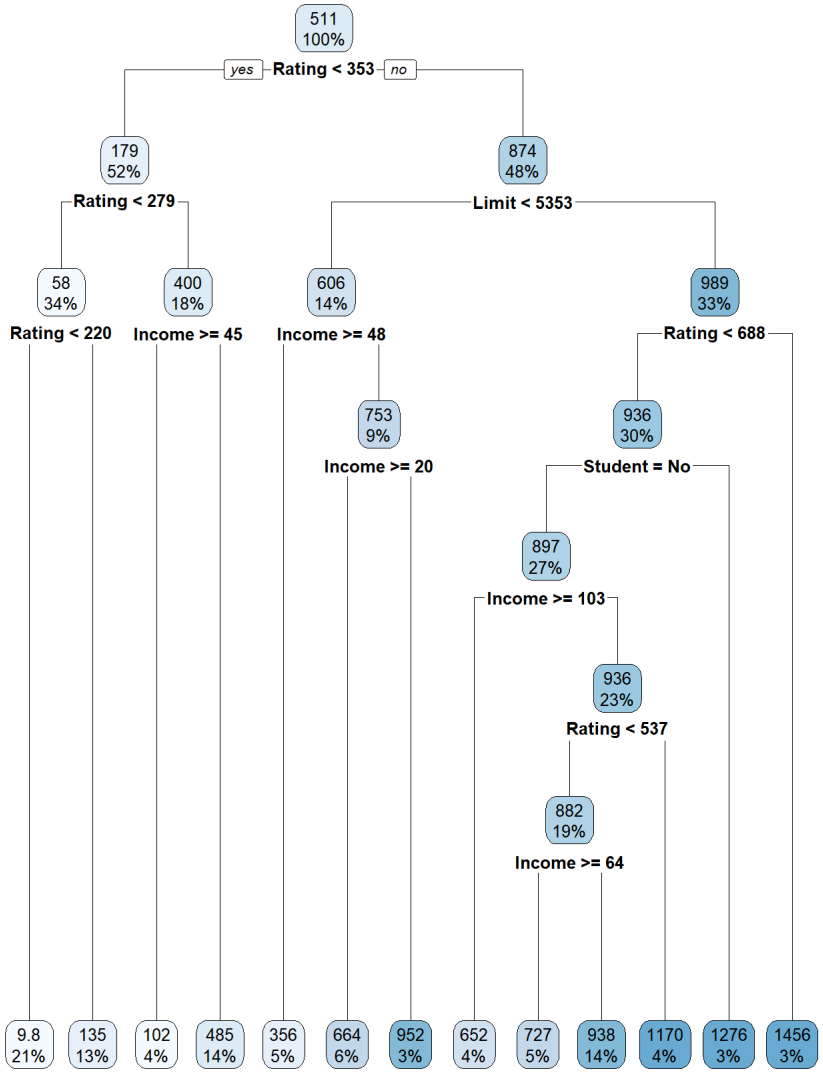


Credit

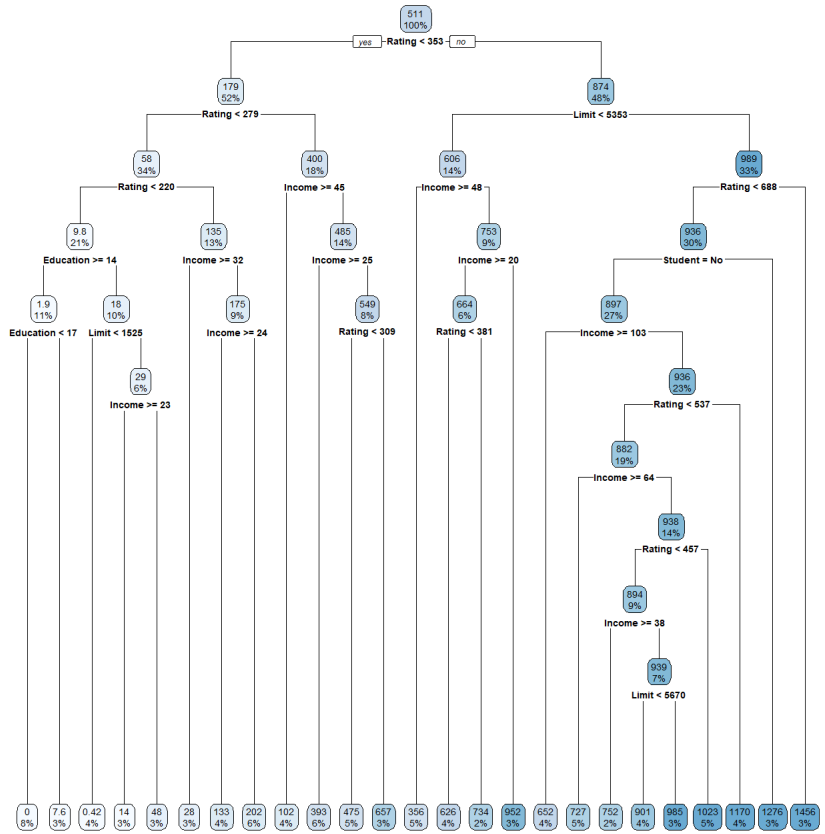
```
tibble(y_obs = test$Balance,
       y_pred = predict(poda1, newdata = test)) %>%
 ggplot(aes(y_obs, y_pred)) +
  geom_abline(slope = 1, intercept = 0, color = "red", size = 2) +
  geom_point(size = 3, alpha = .5)
```



Poda com CP ótimo

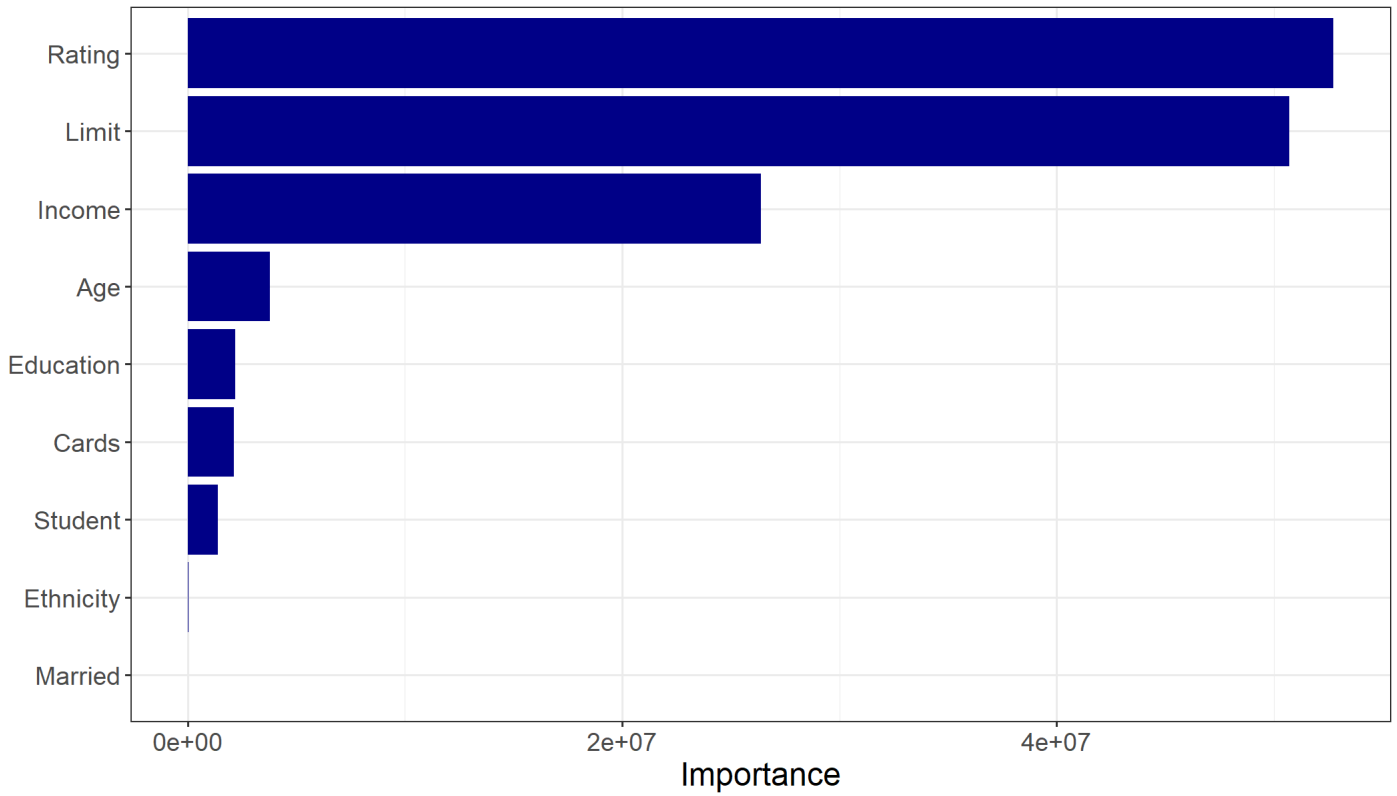


CP = 0



Árvore de Regressão

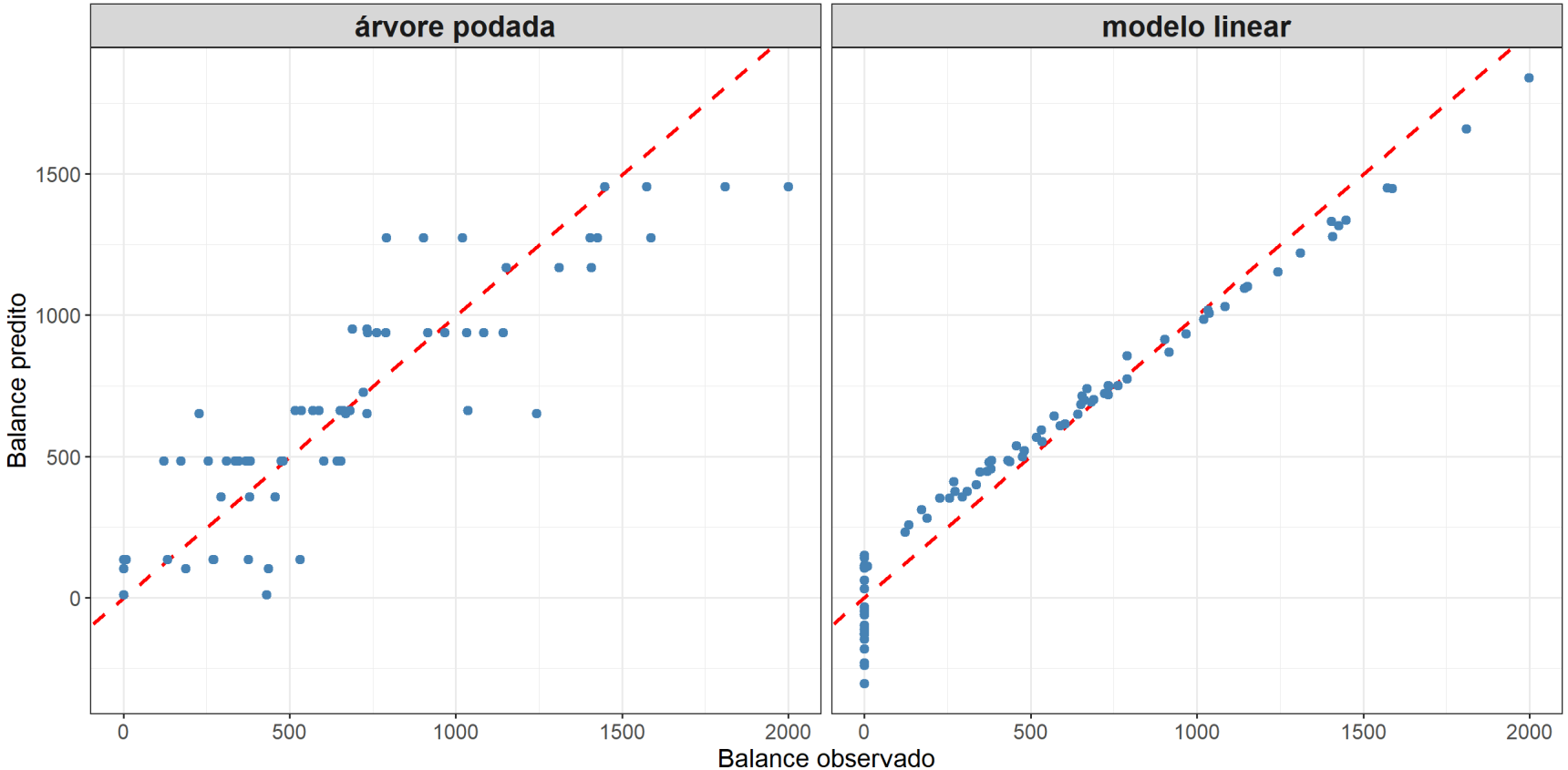
```
vip::vip(poda1, aesthetics = list(fill = "darkblue")) +
  theme_bw()
```



The relative importance of predictor x is the sum of the squared improvements over all internal nodes of the tree for which x was chosen as the partitioning variable; see Breiman, Friedman, and Charles J. Stone (1984) for details. (documentação do pacote vip).

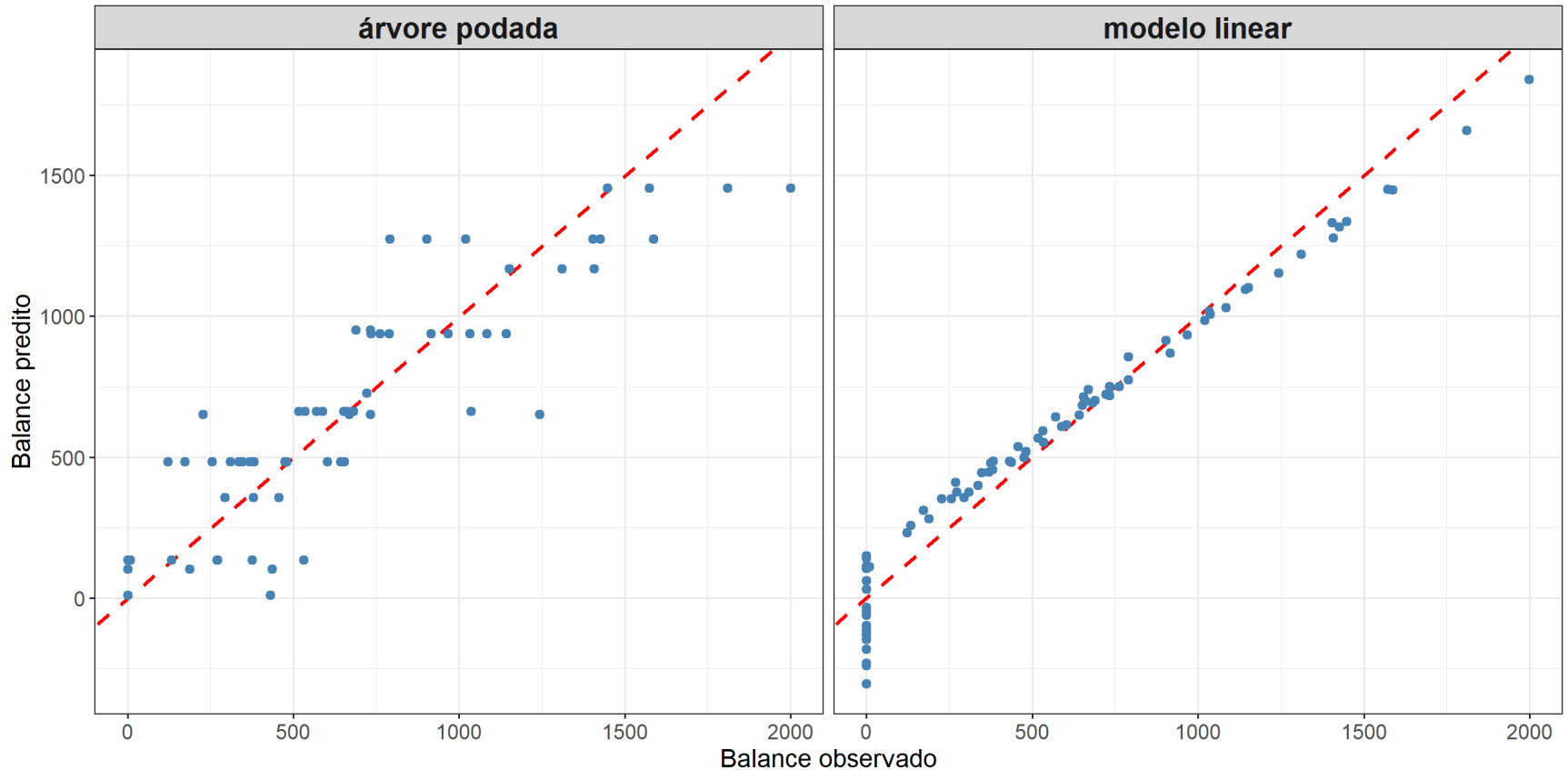
Comparação com modelo linear

Calcule o EQM (erro quadrático médio) no conjunto de teste para a árvore podada e para o modelo linear.



Comparação com modelo linear

Calcule o EQM (erro quadrático médio) no conjunto de teste para a árvore podada e para o modelo linear.



EQM(árvore) = 39.329 e EQM(modelo linear) = 9.792.

Árvore de Classificação

Medidas

Como medir uniformidade? ($\hat{p}_{mk}$ é a proporção de observações na m-ésima região que pertence a k-ésima classe)

Erro

$$\mathrm{E} = 1 - \max_k(\hat{p}_{mk})$$

Índice de Gini

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Cross-entropy ou **deviance**

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

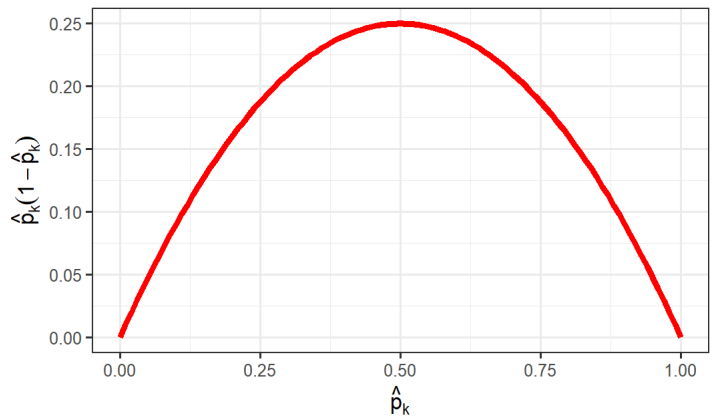
Medidas

Índice de Gini: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$

Classificação	Grupo 1	Grupo 2	Total
> corte	80	20	100
< corte	40	160	200
Total	120	180	300

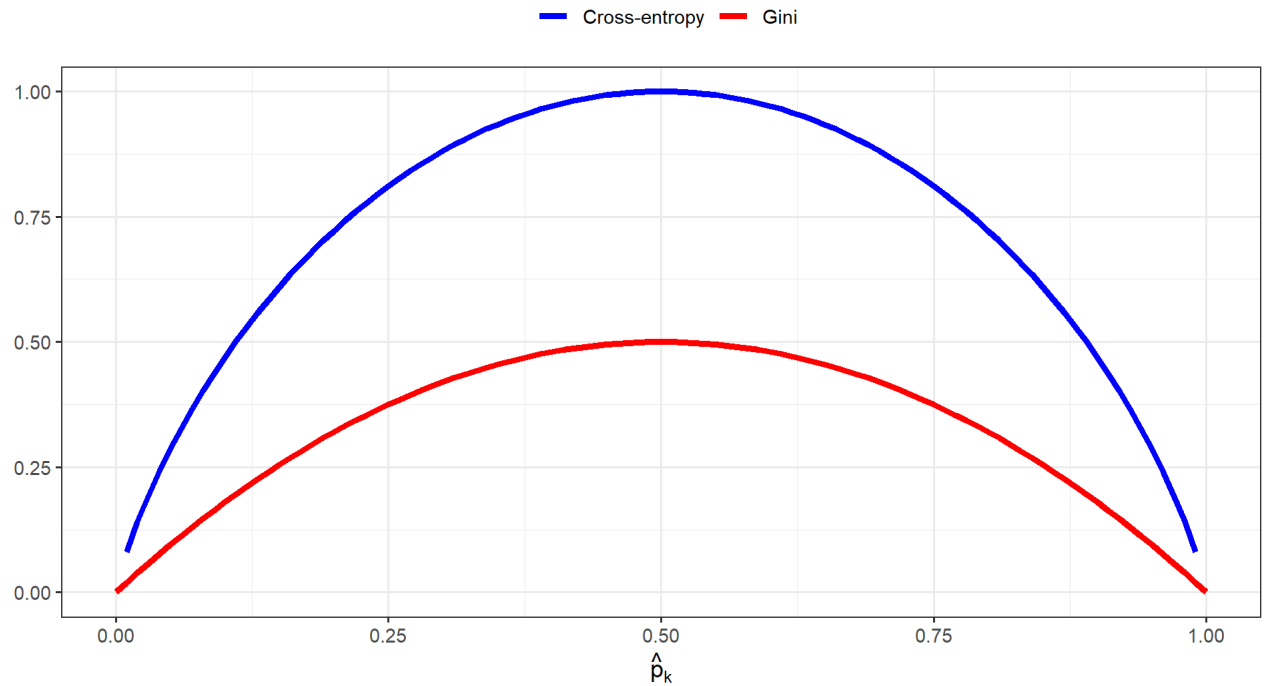
$$G(\text{antes}) = \frac{120}{300}\left(1 - \frac{120}{300}\right) + \frac{180}{300}\left(1 - \frac{180}{300}\right) = 0.48$$

$$G(\text{depois}) = \frac{100}{300} \times \left[\frac{80}{100}\left(1 - \frac{80}{100}\right) + \frac{20}{100}\left(1 - \frac{20}{100}\right)\right] + \frac{200}{300} \times \left[\frac{40}{200}\left(1 - \frac{40}{200}\right) + \frac{160}{200}\left(1 - \frac{160}{200}\right)\right] = 0.32$$



Árvore de Classificação

O índice de Gini e *cross-entropy* são utilizados para avaliar a qualidade de uma divisão/*split*. No entanto, o erro de classificação é mais indicado se a acurácia da previsão é o objetivo da poda da árvore.



Normalmente índice de Gini e *cross-entropy* levam a árvores similares. O índice de Gini apresenta um menor custo computacional.

Árvore de Classificação

Cost-complexity function

$$\mathrm{C}_\alpha(\mathrm{T}) = \sum_{m=1}^{|\mathrm{T}|}(1 - \hat{p}_{R_m}) + \alpha|\mathrm{T}|$$

Para cada valor de α escolhemos a subárvore que minimiza $\mathrm{C}_\alpha(\mathrm{T})$. O parâmetro α controla o *tradeoff* entre o tamanho da árvore e o ajuste. Usualmente α é escolhido por validação cruzada com 5 ou 10 lotes.

Default

```
dados <- ISLR::Default

fit <- rpart(default ~ ., dados)

head(predict(fit, type = "class"))
```

```
##  1  2  3  4  5  6
## No No No No No No
## Levels: No Yes
```

```
head(predict(fit, type = "prob"))
```

```
##          No        Yes
## 1 0.9823929 0.01760708
## 2 0.9823929 0.01760708
## 3 0.9823929 0.01760708
## 4 0.9823929 0.01760708
## 5 0.9823929 0.01760708
## 6 0.9823929 0.01760708
```

Vantagens e Desvantagens

Aspectos Positivos

- Fácil de explicar (muito mais que regressão linear)
- Podem ser apresentadas graficamente e facilmente interpretadas por pessoas que não são especialistas no assunto
- Tratam facilmente preditores qualitativos, sem a necessidade da criação de variáveis indicadoras / *dummies*
- Não é sensível a escala como outros métodos

Aspectos Negativos

- Uma pequena alteração nos dados pode causar uma grande alteração na árvore estimada (variância alta)
- Previsões baseadas em regiões retangulares
- Não apresentam desempenho preditivo tão bom quanto outros métodos

Aplicações

Boston Housing

- **crim** per capita crime rate by town.
- **zn** proportion of residential land zoned for lots over 25,000 sq.ft.
- **indus** proportion of non-retail business acres per town.
- **chas** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- **nox** nitrogen oxides concentration (parts per 10 million).
- **rm** average number of rooms per dwelling.
- **age** proportion of owner-occupied units built prior to 1940.
- **dis** weighted mean of distances to five Boston employment centres.
- **rad** index of accessibility to radial highways.
- **tax** full-value property-tax rate per $10,000.
- **ptratio** pupil-teacher ratio by town.
- **black** 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- **lstat** lower status of the population (percent).
- **medv** median value of owner-occupied homes in $1000s.

Boston Housing

```
library(MASS)

head(Boston)
```

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
1.46336	0	19.58	0	0.605	7.489	90.8	1.9709	5	403	14.7	1.73	50
1.83377	0	19.58	1	0.605	7.802	98.2	2.0407	5	403	14.7	1.92	50
1.51902	0	19.58	1	0.605	8.375	93.9	2.162	5	403	14.7	3.32	50
2.01019	0	19.58	0	0.605	7.929	96.2	2.0459	5	403	14.7	3.7	50
0.05602	0	2.46	0	0.488	7.831	53.6	3.1992	3	193	17.8	4.45	50
0.01381	80	0.46	0	0.422	7.875	32	5.6484	4	255	14.4	2.97	50
0.02009	95	2.68	0	0.4161	8.034	31.9	5.118	4	224	14.7	2.88	50
0.52693	0	6.2	0	0.504	8.725	83	2.8944	8	307	17.4	4.63	50
0.61154	20	3.97	0	0.647	8.704	86.9	1.801	5	264	13	5.12	50
0.57834	20	3.97	0	0.575	8.297	67	2.4216	5	264	13	7.44	50

Telcom Customer Churn

- **customerID** Customer ID
- **gender** Whether the customer is a male or a female
- **SeniorCitizen** Whether the customer is a senior citizen or not (1, 0)
- **Partner** Whether the customer has a partner or not (Yes, No)
- **Dependents** Whether the customer has dependents or not (Yes, No)
- **tenure** Number of months the customer has stayed with the company
- **PhoneService** Whether the customer has a phone service or not (Yes, No)
- **MultipleLines** Whether the customer has multiple lines or not (Yes, No, No phone service)
- **InternetService** Customer's internet service provider (DSL, Fiber optic, No)
- **OnlineSecurity** Whether the customer has online security or not (Yes, No, No internet service)
- **OnlineBackup** Whether the customer has online backup or not (Yes, No, No internet service)
- **DeviceProtection** Whether the customer has device protection or not (Yes, No, No internet service)

Telcom Customer Churn

- **TechSupport** Whether the customer has tech support or not (Yes, No, No internet service)
- **StreamingTV** Whether the customer has streaming TV or not (Yes, No, No internet service)
- **StreamingMovies** Whether the customer has streaming movies or not (Yes, No, No internet service)
- **Contract** The contract term of the customer (Month-to-month, One year, Two year)
- **PaperlessBilling** Whether the customer has paperless billing or not (Yes, No)
- **PaymentMethod** The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
- **MonthlyCharges** The amount charged to the customer monthly
- **TotalCharges** The total amount charged to the customer
- **Churn** Whether the customer churned or not (Yes or No)

Telcom Customer Churn

```
dados <- read_csv("dados/WA_Fn-UseC_-Telco-Customer-Churn.csv")

head(dados)
```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7590-VHVEG	Female	0	Yes	No	1	No	No phone service
5575-GNVDE	Male	0	No	No	34	Yes	No
3668-QPYBK	Male	0	No	No	2	Yes	No
7795-CFOCW	Male	0	No	No	45	No	No phone service
9237-HQITU	Female	0	No	No	2	Yes	No

Obrigado!

tiagoms.com

tiagomendonca

tiagoms1@insper.edu.br