

# Análise da Base de Dados Adult

## Sobre a base de dados

O conjunto de dados utilizado é chamado “Adult” e foi derivado do banco de dados do censo dos EUA. Foi preparado por Barry Becker a partir do censo de 1994. O propósito desse conjunto de dados é prever se uma pessoa ganha mais ou menos de US\$ 50.000 por ano.

Link do conjunto de dados: <https://archive.ics.uci.edu/ml/datasets/Adult>

## Carregando os Dados do arquivo

```
database <- read.csv(file = "Adult.CSV")  
  
colnames(database) <- c("age", "workclass", "fnlwgt", "education", "education_num", "marital_status", "occupation", "relationship", "race", "sex", "capital_gain", "capital_loss", "hours_per_week", "native_country", "income")  
  
#Printar a base de dados  
database
```

Table 1: Adult Database - Part 1

age	workclass	fnlwgt	education	education_num	marital_status	occupation
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty

Table 2: Adult Database - Part 2

relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
Not-in-family	White	Male	2174	0	40	United-States	<=50K
Husband	White	Male	0	0	13	United-States	<=50K
Not-in-family	White	Male	0	0	40	United-States	<=50K
Husband	Black	Male	0	0	40	United-States	<=50K
Wife	Black	Female	0	0	40	Cuba	<=50K

## 1. Identificação do atributo alvo (saída)

A base de dados possui 32561 exemplos e 15 atributos sendo seu atributo alvo “income”

```
target_attribute <- "income"
```

## 2. Identificação dos tipos de dados dos atributos de entrada (quantitativo, qualitativo)

Identificar os tipos de dados em uma base de dados é essencial para entender a natureza das informações presentes. Na base de dados analisada, os dados podem ser classificados nas seguintes categorias:

- Categóricos: Representam características com valores qualitativos.
- Numéricos: Representam valores numéricos.

```
#Para classificação dos atributos como int ou char  
str(database)
```

```
## 'data.frame': 32562 obs. of 15 variables:  
## $ age : int 39 50 38 53 28 37 49 52 31 42 ...  
## $ workclass : chr "State-gov" "Self-emp-not-inc" "Private" "Private" ...  
## $ fnlwgt : int 77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...  
## $ education : chr "Bachelors" "Bachelors" "HS-grad" "11th" ...  
## $ education_num : int 13 13 9 7 13 14 5 9 14 13 ...  
## $ marital_status: chr "Never-married" "Married-civ-spouse" "Divorced" "Married-civ-spouse" ...  
## $ occupation : chr "Adm-clerical" "Exec-managerial" "Handlers-cleaners" "Handlers-cleaners" ...  
## $ relationship : chr "Not-in-family" "Husband" "Not-in-family" "Husband" ...  
## $ race : chr "White" "White" "White" "Black" ...  
## $ sex : chr "Male" "Male" "Male" "Male" ...  
## $ capital_gain : int 2174 0 0 0 0 0 0 14084 5178 ...  
## $ capital_loss : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ hours_per_week: int 40 13 40 40 40 40 16 45 50 40 ...  
## $ native_country: chr "United-States" "United-States" "United-States" "United-States" ...  
## $ income : chr "<=50K" "<=50K" "<=50K" "<=50K" ...
```

```
#classificação manual  
data_types <- c(  
  age = "quantitativo",  
  workclass = "qualitativo",  
  fnlwgt = "quantitativo",  
  education = "qualitativo",  
  education_num = "quantitativo",  
  marital_status = "qualitativo",  
  occupation = "qualitativo",  
  relationship = "qualitativo",  
  race = "qualitativo",  
  sex = "qualitativo",  
  capital_gain = "quantitativo",  
  capital_loss = "quantitativo",  
  hours_per_week = "quantitativo",  
  native_country = "qualitativo",  
  income = "qualitativo")  
data_types
```

Table 3: Tipos de dados dos atributos de entrada

Atributo	Classificação
age	quantitativo
workclass	qualitativo
fnlwgt	quantitativo
education	qualitativo
education_num	quantitativo
marital_status	qualitativo
occupation	qualitativo
relationship	qualitativo
race	qualitativo
sex	qualitativo
capital_gain	quantitativo
capital_loss	quantitativo
hours_per_week	quantitativo
native_country	qualitativo
income	qualitativo

### 3. Identificação da escala de dados dos atributos de entrada (nominal, ordinal, intervalar, racional)

A identificação da escala de dados é relevante para entender a natureza e a magnitude dos valores presentes em uma base de dados. A escala dos dados pode ser dividida em quatro categorias principais:

- Nominal: Categorizados em diferentes grupos ou categorias, sem qualquer ordem ou hierarquia específica.
- Ordinal: Possuem uma ordem ou hierarquia específica, mas a diferença entre os valores não é necessariamente uniforme.
- Intervalar: Expressos em valores numéricos com uma diferença uniforme entre eles.
- Racional: Representam um valor numérico absoluto.

```
data_scales <- c(
  age = "Racional",
  workclass = "Nominal",
  fnlwgt = "Racional",
  education = "Ordinal",
  education_num = "Ordinal",
  marital_status = "Nominal",
  occupation = "Nominal",
  relationship = "Nominal",
  race = "Nominal",
  sex = "Nominal",
  capital_gain = "Racional",
  capital_loss = "Racional",
  hours_per_week = "Racional",
  native_country = "Nominal",
  income = "Nominal"
```

```
)  
data_scales
```

Table 4: Identificação da escala de dados

Atributo	Escala
age	Racional
workclass	Nominal
fnlwgt	Racional
education	Ordinal
education_num	Ordinal
marital_status	Nominal
occupation	Nominal
relationship	Nominal
race	Nominal
sex	Nominal
capital_gain	Racional
capital_loss	Racional
hours_per_week	Racional
native_country	Nominal
income	Nominal

## 4. Exploração dos dados através de medidas de localidade

As medidas de localidade, fornecem informações sobre o valor central dos dados, no qual ajudam a entender onde a maioria dos dados está concentrado. Essas medidas ajudam a entender onde a maioria dos dados está concentrada. Incluem:

- Média
- Moda
- Mediana

```
numeric_columns1 <- sapply(database, is.numeric)  
summary(database[, numeric_columns1])
```

```
##      age          fnlwgt      education_num      capital_gain  
##  Min.   :17.00    Min.   : 12285    Min.   : 1.00    Min.   : 0  
##  1st Qu.:28.00    1st Qu.: 117827   1st Qu.: 9.00    1st Qu.: 0  
##  Median :37.00    Median : 178356   Median :10.00    Median : 0  
##  Mean   :38.58    Mean   : 189778   Mean   :10.08    Mean   :1078  
##  3rd Qu.:48.00    3rd Qu.: 237051   3rd Qu.:12.00    3rd Qu.: 0  
##  Max.   :90.00    Max.   :1484705   Max.   :16.00    Max.   :99999  
##  NA's   :1         NA's   :1         NA's   :1         NA's   :1  
##      capital_loss      hours_per_week  
##  Min.   : 0.0        Min.   : 1.00  
##  1st Qu.: 0.0        1st Qu.:40.00  
##  Median : 0.0        Median :40.00  
##  Mean   : 87.3       Mean   :40.44  
##  3rd Qu.: 0.0        3rd Qu.:45.00  
##  Max.   :4356.0      Max.   :99.00  
##  NA's   :1         NA's   :1
```

```

Modes <- function(x) {
  ux <- unique(x)
  tab <- tabulate(match(x, ux))
  ux[tab == max(tab)]
}

for (i in 1:ncol(database)) {
  mod_val <- Modes(database[,i])
  print(mod_val)
}

```

```

## [1] 36
## [1] "Private"
## [1] 123011 164190 203488
## [1] "HS-grad"
## [1] 9
## [1] "Married-civ-spouse"
## [1] "Prof-specialty"
## [1] "Husband"
## [1] "White"
## [1] "Male"
## [1] 0
## [1] 0
## [1] 40
## [1] "United-States"
## [1] "<=50K"

```

## 5. Exploração dos dados através de medidas de espalhamento

As medidas de espalhamento são utilizadas para avaliar a dispersão ou variabilidade dos dados em um conjunto de observações. Elas fornecem informações sobre o quanto distantes os valores estão uns dos outros e da medida central dos dados. Incluem

- Variância
- Desvio Padrão
- Amplitude

```

spread_measures <- function(x) {
  c(variance = var(x, na.rm = TRUE), sd = sd(x, na.rm = TRUE), range = range(x, na.rm = TRUE))
}

# Identifica quais colunas são numéricas.
numeric_columns2 <- sapply(database, is.numeric)
# Aplica a função spread_measures a cada coluna numérica para calcular a variância, desvio padrão e amplitude
sapply(database[, numeric_columns2], spread_measures)

##          age      fnlwgt education_num capital_gain capital_loss
## variance 186.06140 11140797792       6.61889 54542539.178   162376.9378
## sd        13.64043     105550       2.57272    7385.292     402.9602
## range1    17.00000     12285       1.00000     0.000       0.00000
## range2    90.00000    1484705      16.00000   99999.000     4356.0000

```

```

##          hours_per_week
## variance      152.45900
## sd            12.34743
## range1        1.00000
## range2        99.00000

```

## 6. Exploração dos dados através de medidas de distribuição

As medidas de distribuição são utilizadas para analisar a forma e a natureza da distribuição dos dados em um conjunto de observações. Essas medidas fornecem informações sobre a simetria, o achatamento e a forma geral da distribuição dos dados. Incluem:

- Assimetria
- Curtose

```

# Funções para calcular assimetria e curtose
calc_skewness <- function(x) {
  n <- length(x)
  mean_x <- mean(x, na.rm = TRUE)
  sd_x <- sd(x, na.rm = TRUE)
  skewness <- sum(((x - mean_x) / sd_x) ^ 3, na.rm = TRUE) / n
  return(skewness)
}

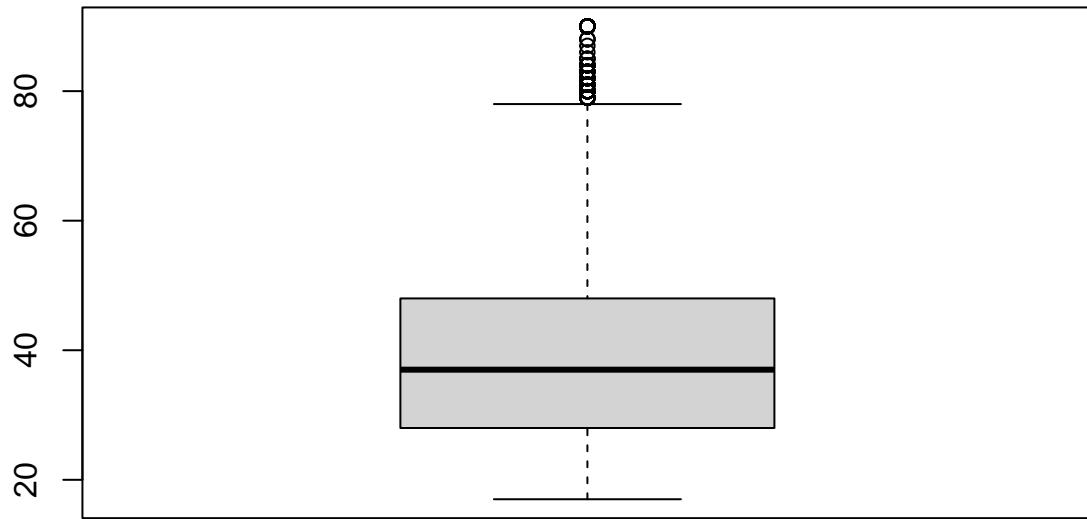
calc_kurtosis <- function(x) {
  n <- length(x)
  mean_x <- mean(x, na.rm = TRUE)
  sd_x <- sd(x, na.rm = TRUE)
  kurtosis <- sum(((x - mean_x) / sd_x) ^ 4, na.rm = TRUE) / n - 3
  return(kurtosis)
}

numeric_columns3 <- sapply(database, is.numeric)

# Plotando boxplots para cada variável numérica
boxplot(database$age, main = "Age Boxplot")

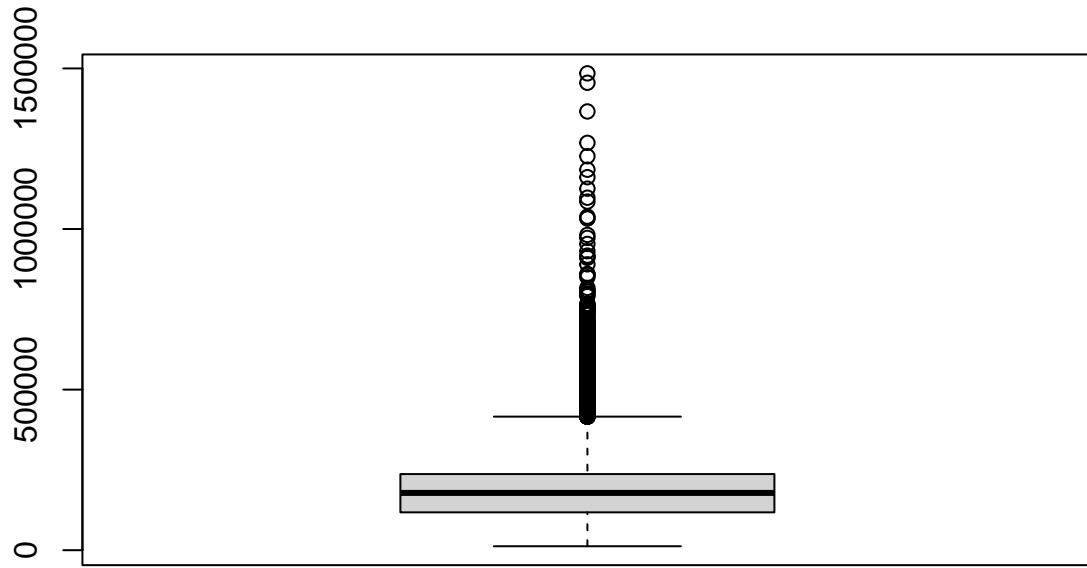
```

## Age Boxplot



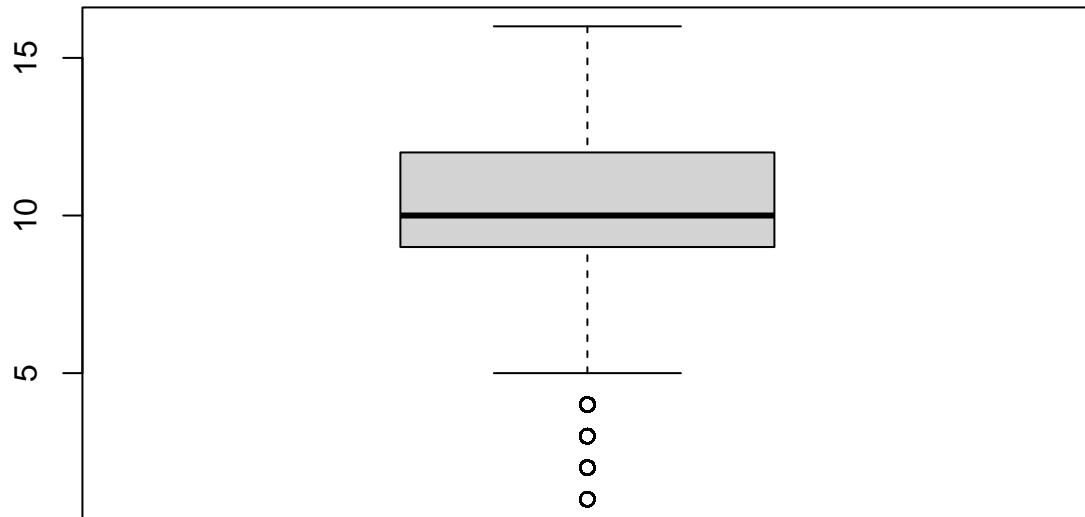
```
boxplot(database$fnlwgt, main = "Fnlwgt Boxplot")
```

### Fnlwgt Boxplot



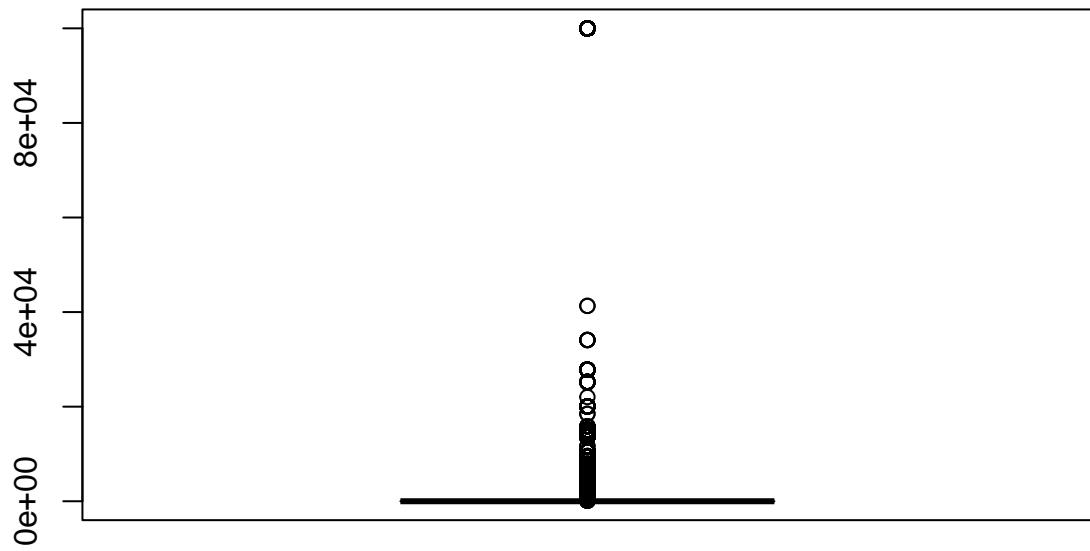
```
boxplot(database$education_num, main = "Education_num Boxplot")
```

## **Education\_num Boxplot**



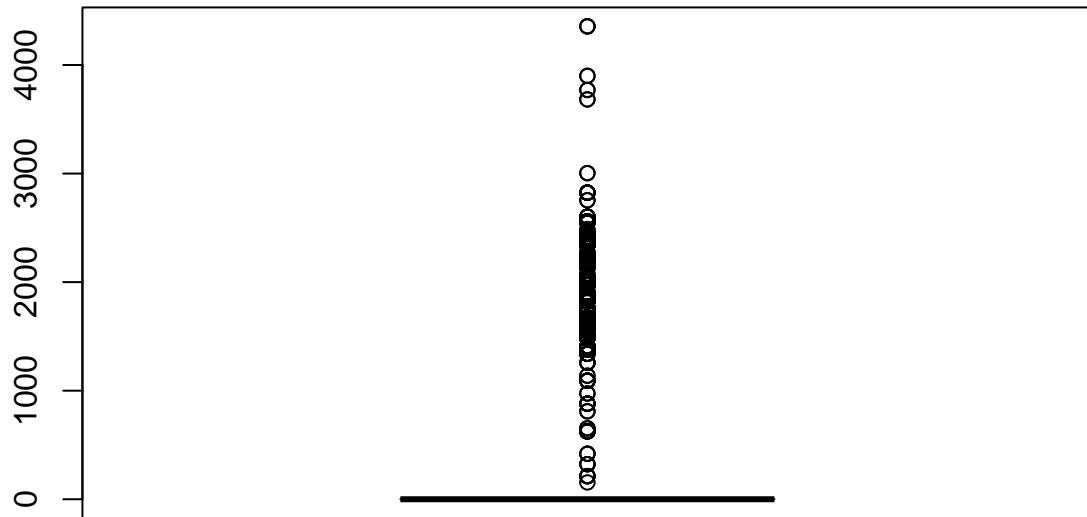
```
boxplot(database$capital_gain, main = "capital_gain Boxplot")
```

### **capital\_gain Boxplot**



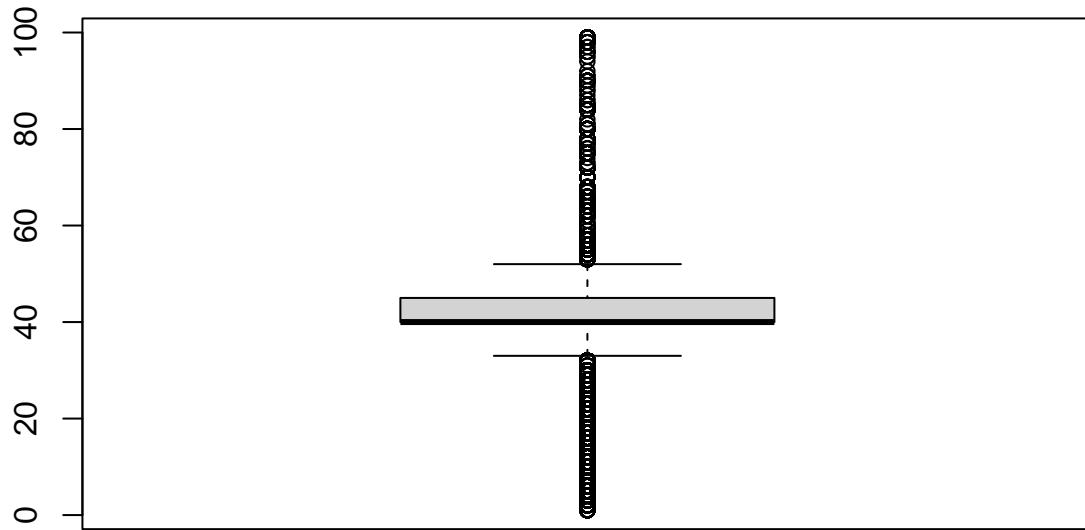
```
boxplot(database$capital_loss, main = "capital_loss Boxplot")
```

### **capital\_loss Boxplot**



```
boxplot(database$hours_per_week, main = "hours_per_week Boxplot")
```

## hours\_per\_week Boxplot

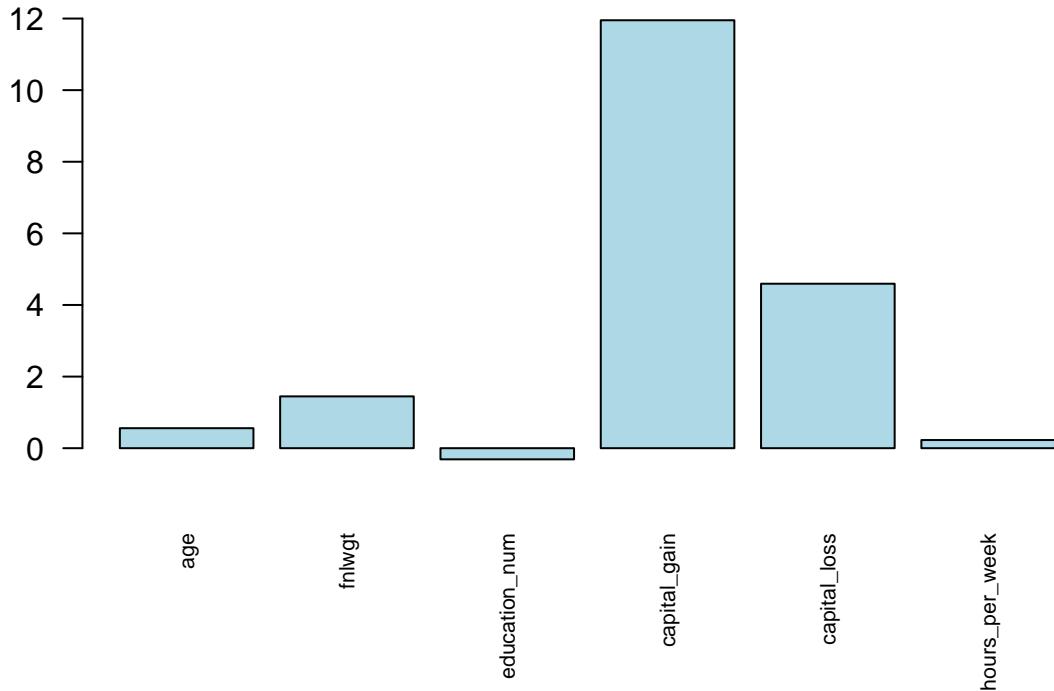


```
# Calculando assimetria e curtose
skewness_values <- sapply(database[, numeric_columns3], calc_skewness)
kurtosis_values <- sapply(database[, numeric_columns3], calc_kurtosis)

# Criando um dataframe para plotagem
metrics_df <- data.frame(
  Variable = names(skewness_values),
  Skewness = skewness_values,
  Kurtosis = kurtosis_values
)

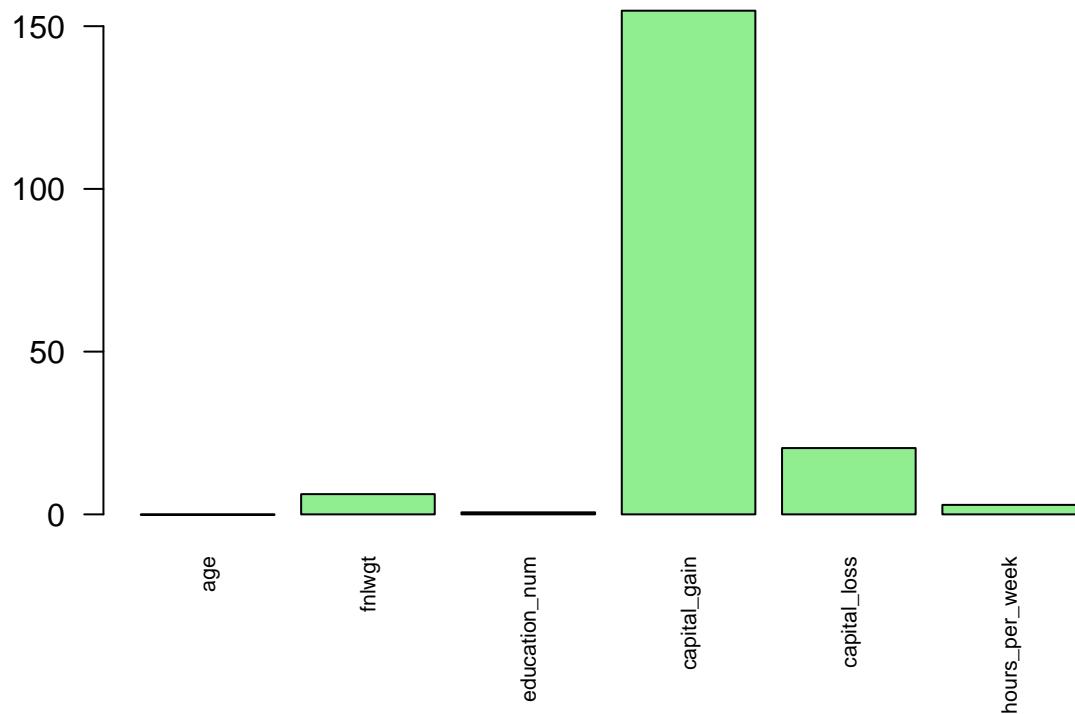
# Gráfico de assimetria
barplot(metrics_df$Skewness, names.arg = metrics_df$Variable,
        main = "Assimetria das Variáveis Numéricas",
        col = "lightblue", ylim = c(min(metrics_df$Skewness) - 1, max(metrics_df$Skewness) + 1),
        cex.names = 0.7, las = 2)
```

## Assimetria das Variáveis Numéricas



```
# Gráfico de curtose
barplot(metrics_df$Kurtosis, names.arg = metrics_df$Variable,
        main = "Curtose das Variáveis Numéricas",
        col = "lightgreen", ylim = c(min(metrics_df$Kurtosis) - 1, max(metrics_df$Kurtosis) + 1),
        cex.names = 0.7, las = 2)
```

## Curtose das Variáveis Numéricas



## 7. Identificação e separação do conjunto de teste

A base de dados original possui 32561 exemplos e 15 atributos

A base de dados teste possui 16281 exemplos e 15 atributos

O que implica em uma base de dados com metade dos exemplos.

```
train_data <- read.csv(file = "Adult.CSV")
test_data <- read.csv(file = "adultTest.csv")

colnames(train_data) <- c("age", "workclass", "fnlwgt", "education", "education_num", "marital_status",
                         "occupation", "relationship", "race", "sex", "capital_gain", "capital_loss", "hours_per_week",
                         "income")

colnames(test_data) <- c("age", "workclass", "fnlwgt", "education", "education_num", "marital_status",
                        "occupation", "relationship", "race", "sex", "capital_gain", "capital_loss", "hours_per_week",
                        "income$income")

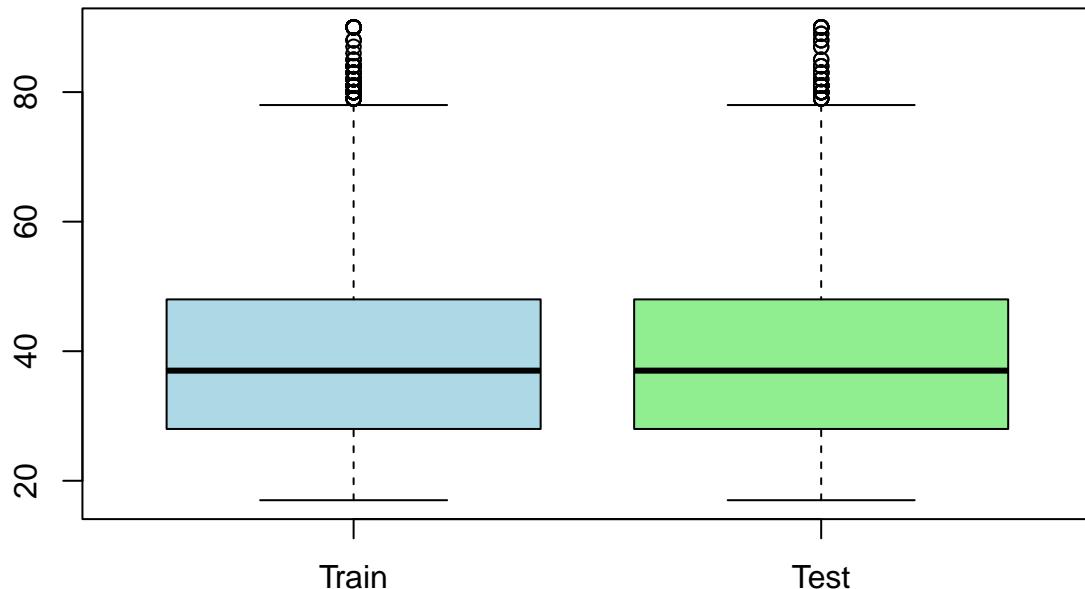
test_data$income <- gsub(">50K.", ">50K", test_data$income)
test_data$income <- gsub("<=50K.", "<=50K", test_data$income)

# Identificando colunas numéricas
numeric_columns <- sapply(train_data, is.numeric)

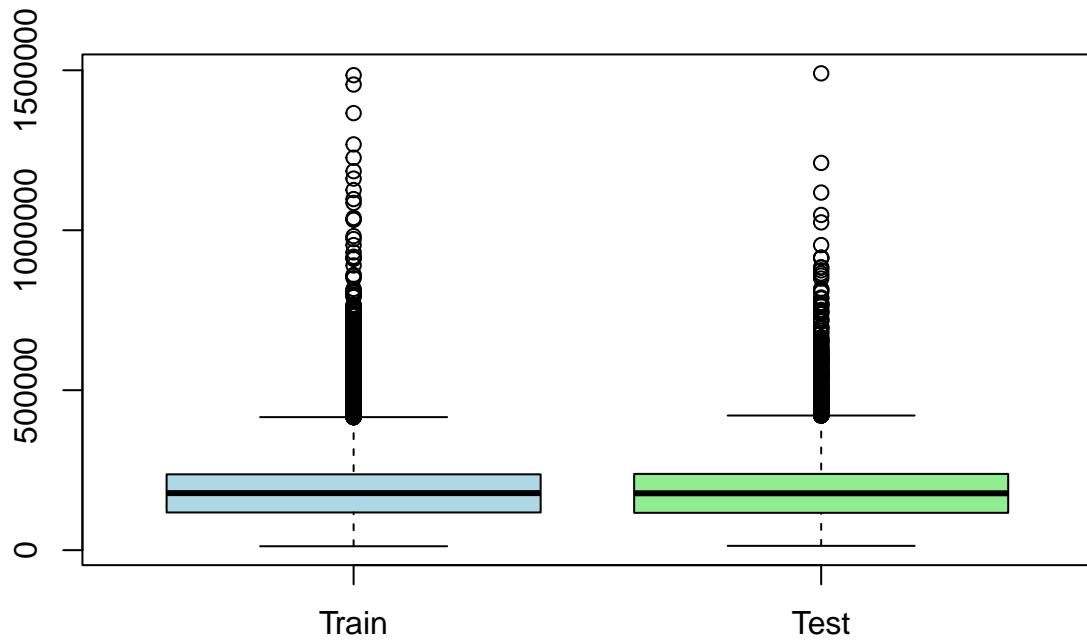
par(mfrow=c(1, 1))
for (col in names(train_data)[numeric_columns]) {
  boxplot(train_data[[col]], test_data[[col]],
          names = c("Train", "Test"),
          main = paste("Boxplot Comparativo para", col),
```

```
    col = c("lightblue", "lightgreen"))
}
```

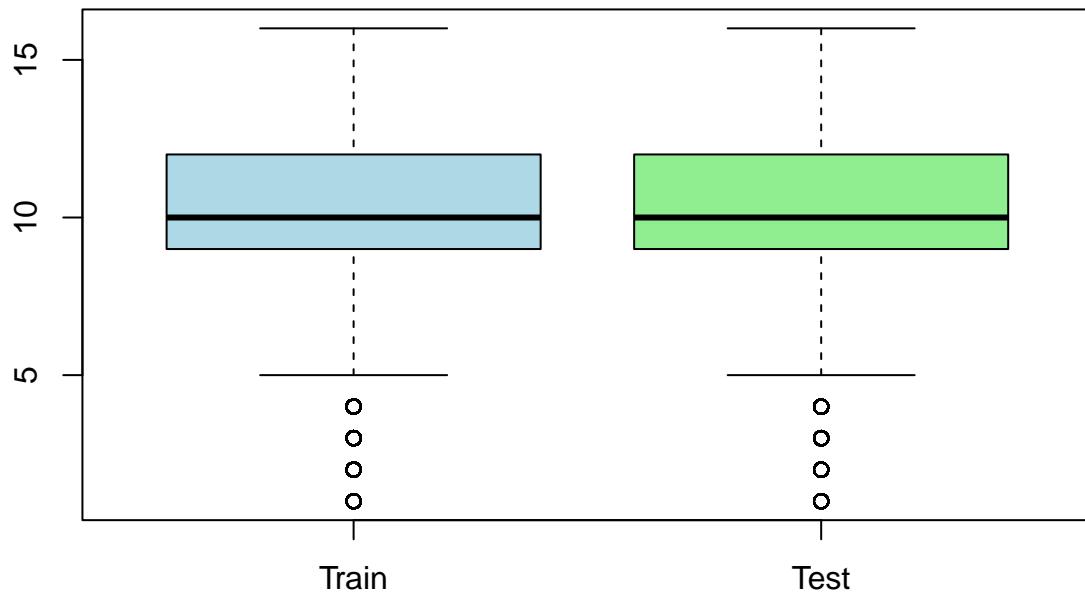
**Boxplot Comparativo para age**



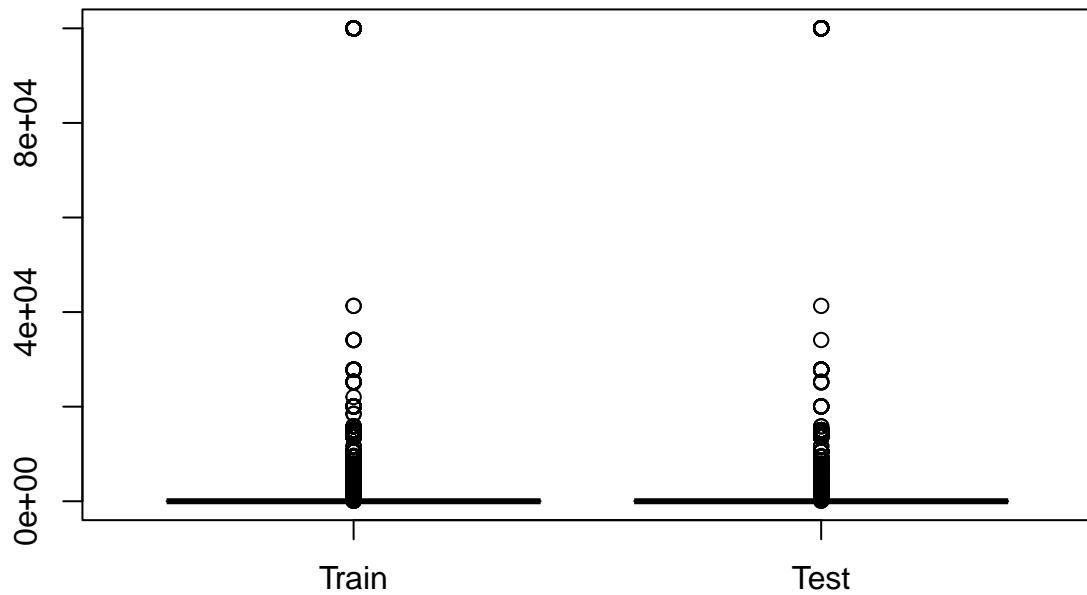
**Boxplot Comparativo para fnlwgt**



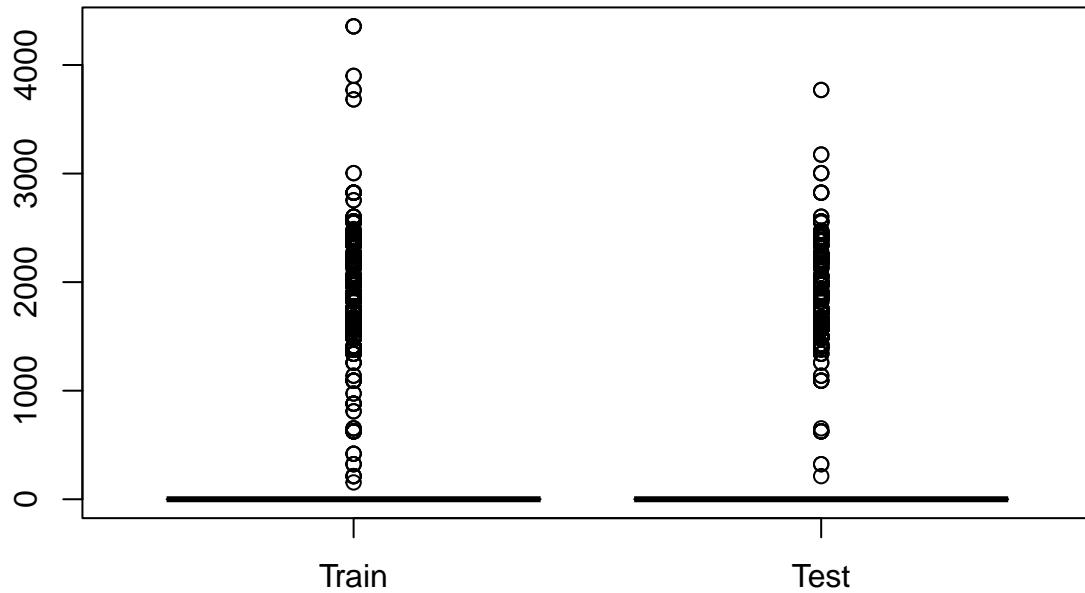
**Boxplot Comparativo para education\_num**



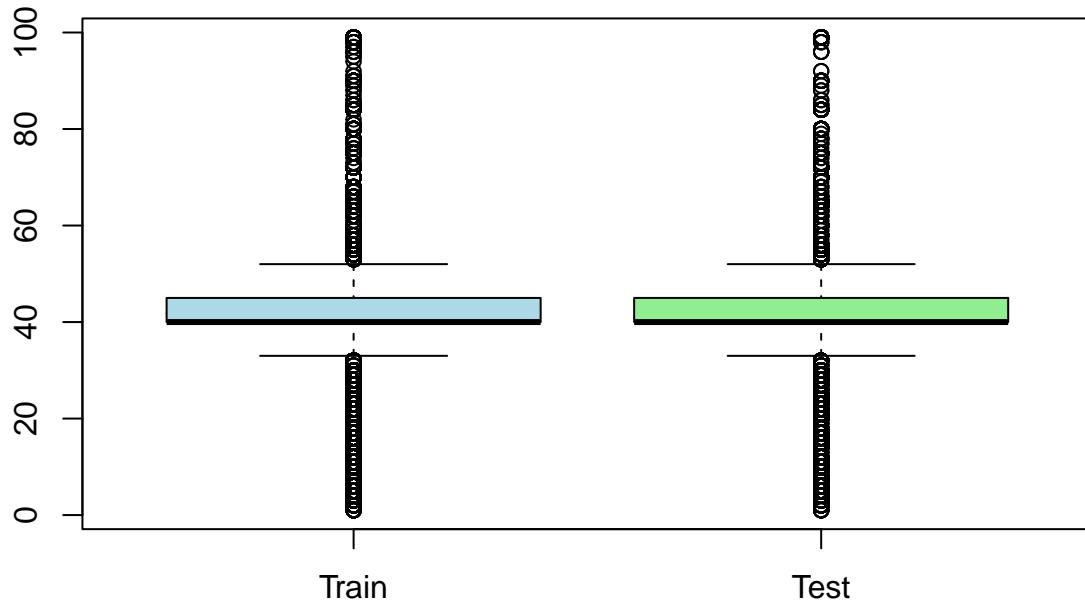
**Boxplot Comparativo para capital\_gain**



**Boxplot Comparativo para capital\_loss**



## Boxplot Comparativo para hours\_per\_week



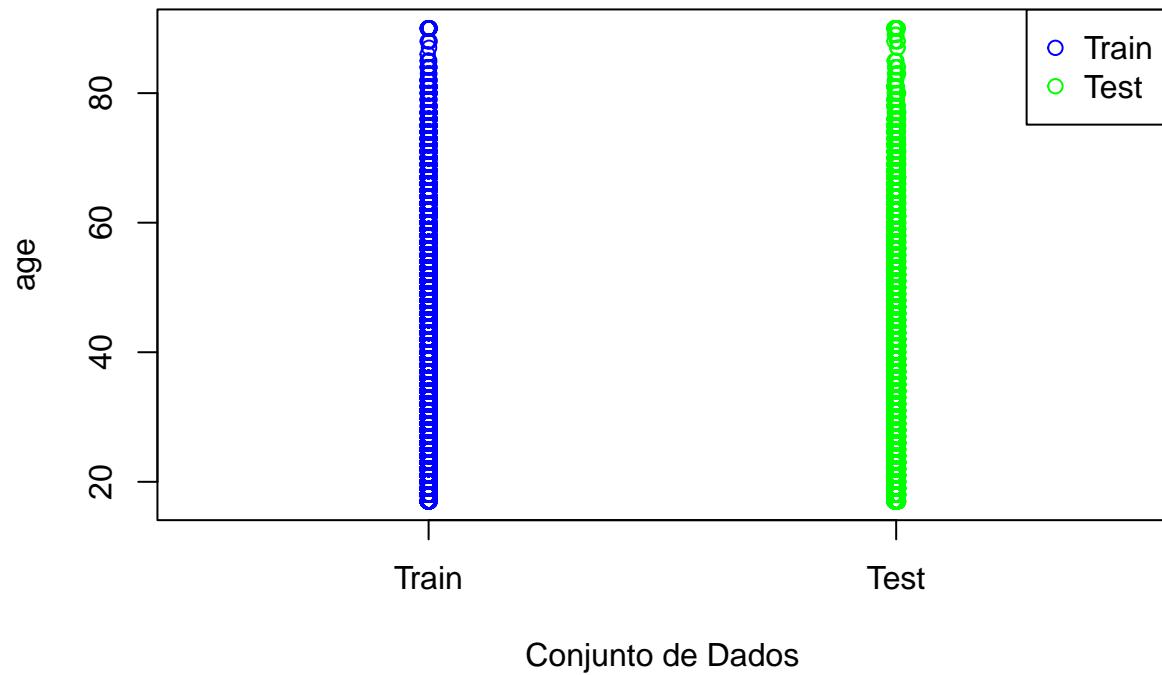
```
jitter_horizontal <- function(x) {
  jitter(x, factor = 0.1)
}

# Criando um loop para gerar gráficos de dispersão comparativos
for (col in names(train_data)[numeric_columns]) {
  plot(jitter_horizontal(rep(1, length(train_data[[col]]))), train_data[[col]],
       xlim = c(0.5, 2.5), xaxt = "n",
       main = paste("Gráfico de Dispersão Comparativo para", col),
       xlab = "Conjunto de Dados", ylab = col, col = "blue")

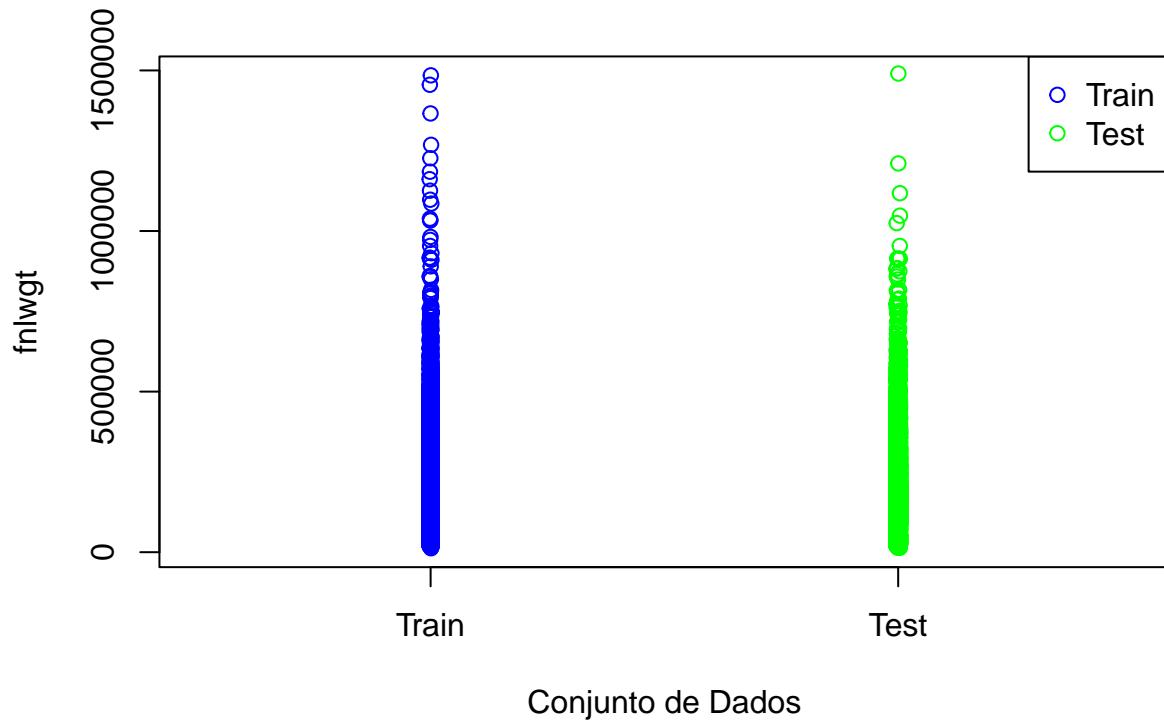
  points(jitter_horizontal(rep(2, length(test_data[[col]]))), test_data[[col]], col = "green")

  axis(1, at = 1:2, labels = c("Train", "Test"))
  legend("topright", legend = c("Train", "Test"), col = c("blue", "green"), pch = 1)
}
```

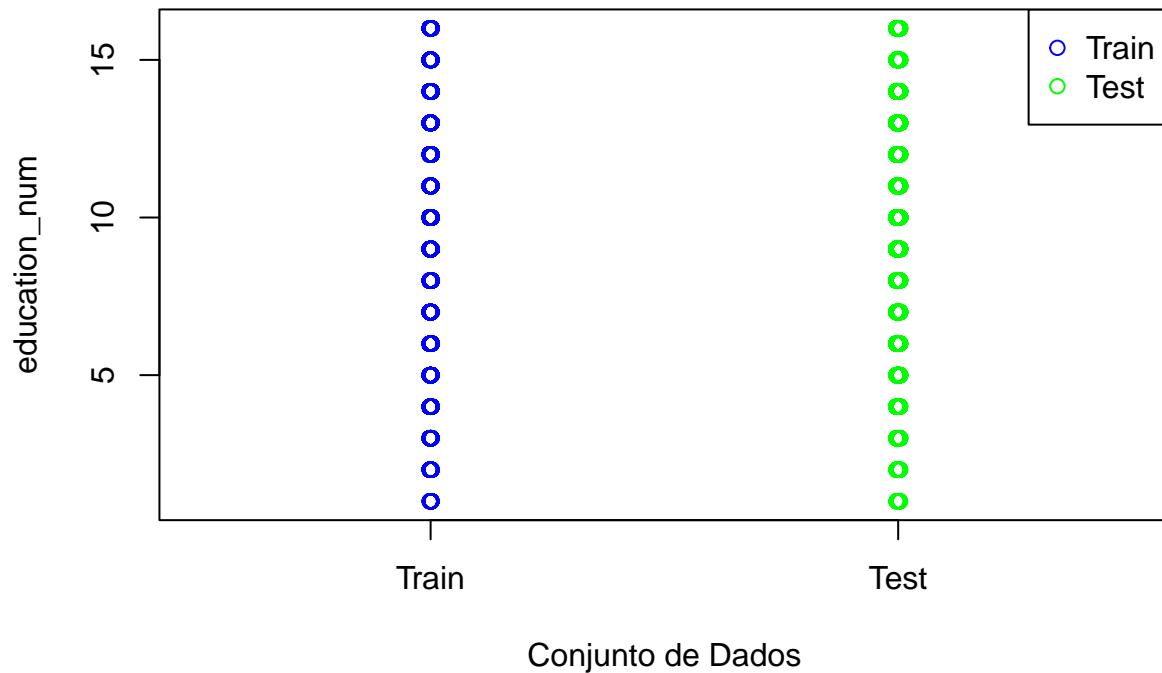
## Gráfico de Dispersão Comparativo para age



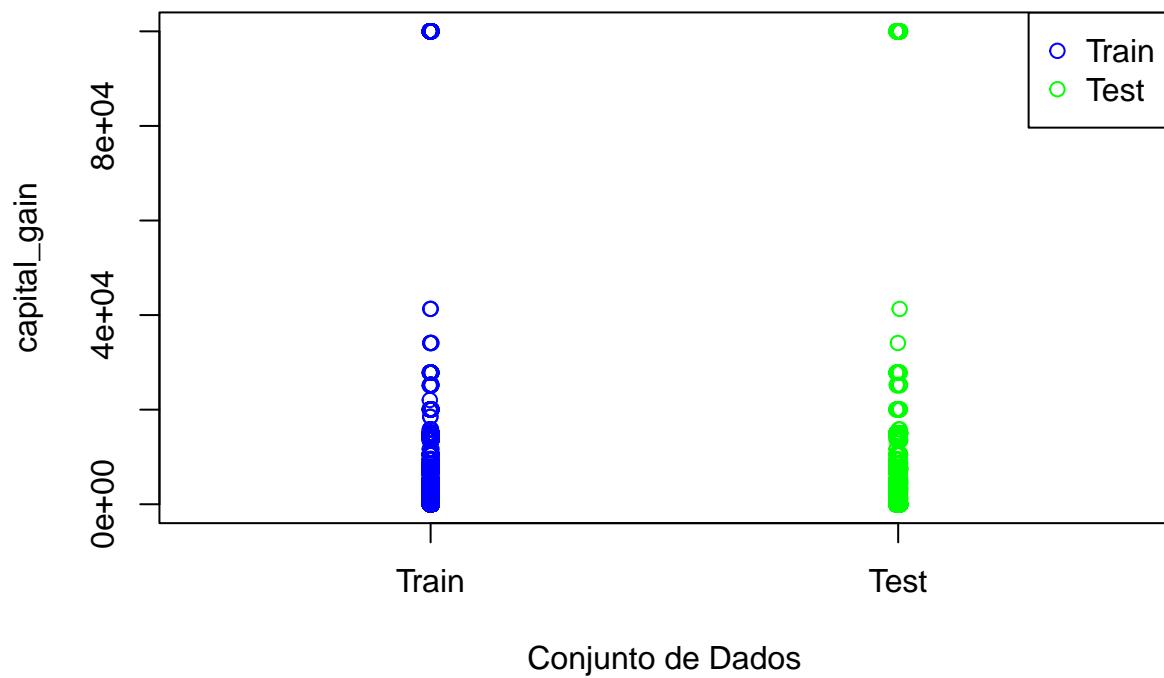
### Gráfico de Dispersão Comparativo para fnlwgt



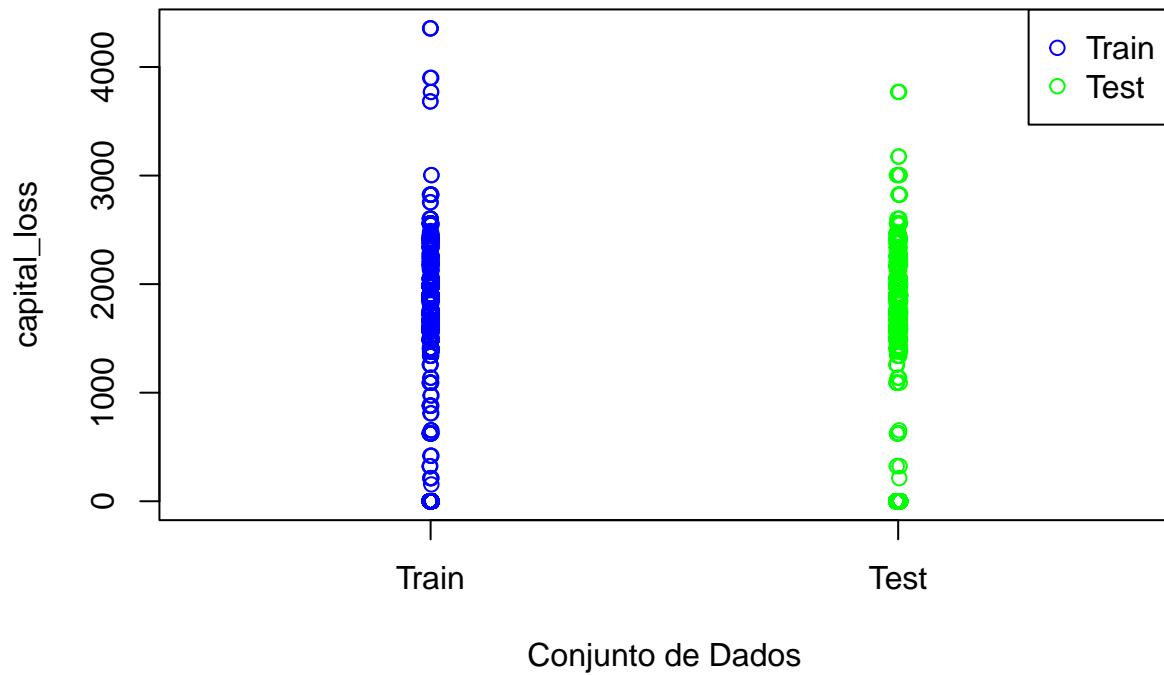
## Gráfico de Dispersão Comparativo para education\_num



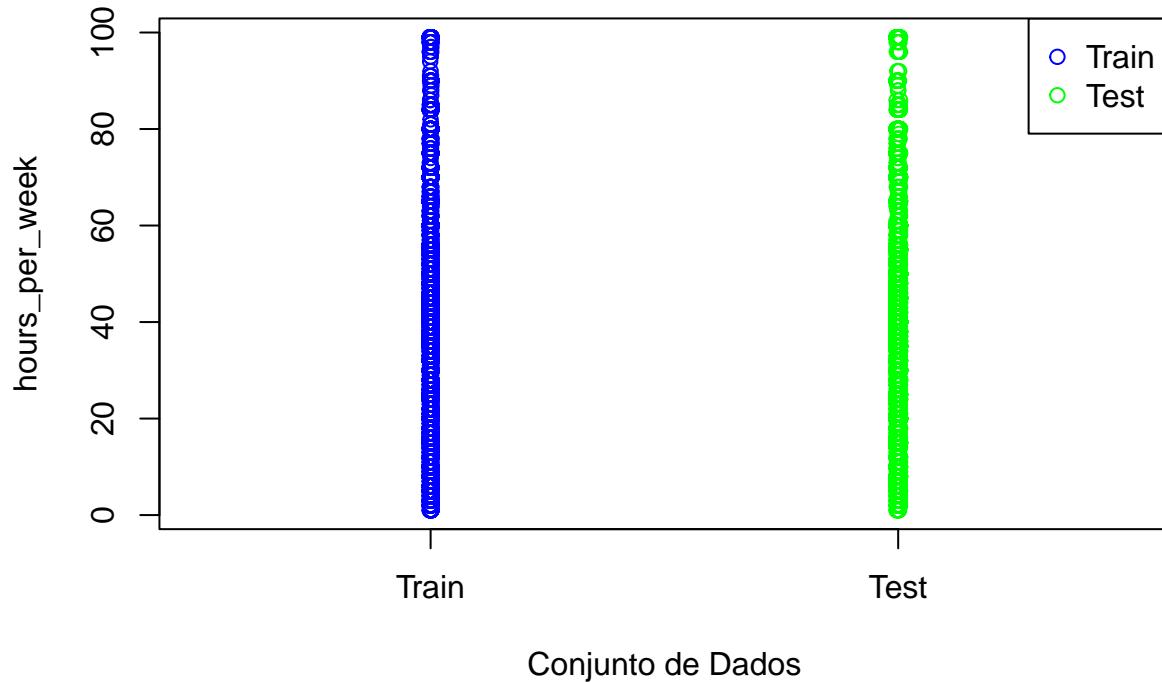
## Gráfico de Dispersão Comparativo para capital\_gain



## Gráfico de Dispersão Comparativo para capital\_loss



## Gráfico de Dispersão Comparativo para hours\_per\_week



Os boxplots mostram a distribuição dos dados por meio de quartis, mediana, outliers e intervalo interquartil. Ao comparar os boxplots, nota-se que as medidas estatísticas, como quartis e mediana, são próximas entre as bases original e de teste. Isso sugere que os atributos estão sendo preservados de forma consistente na base de teste, com valores semelhantes aos do conjunto original. Os gráficos de dispersão mostram a relação entre dois atributos, permitindo observar a dispersão dos pontos. Ao comparar os gráficos de dispersão, podemos identificar se a relação entre os atributos é preservada na base de teste. Se os pontos seguem uma tendência semelhante e mantêm uma relação coerente, isso indica que a base de teste reflete adequadamente as características do conjunto original. Conclui-se que a base de dados de testes mantém as mesmas características da base de dados original.

## 8. Identificação e eliminação de atributos não necessários

Foi removido o atributo fnlwgt, pois não é relevante para a tarefa de previsão em questão, e o atributo education-num, pois duplica as informações disponíveis no atributo education. Além disso, o atributo relationship foi removido, pois é um subgrupo do atributo marital\_status.

```
#é a cópia de education_num
train_data$education <- remove()
#Não é um dado significante para a base de dados
train_data$fnlwgt <- remove()
#Relationship é um subgrupo de marital_status
train_data$relationship <- remove()
```

```

test_data$education <- remove()
test_data$fnlwgt <- remove()
test_data$relationship <- remove()

```

## 9. Identificação e eliminação de exemplos não necessários

A única eliminação feita foi em uma linha na base de dados onde todos os atributos se encontram como “N/A”

```

#Achar o único exemplo onde não há nada nele
empty_rows <- !complete.cases(train_data)
#Printar a linha no qual ele se encontra
print(train_data[empty_rows, ])

```

```

##      age workclass education_num marital_status occupation race sex
## 32562    NA                      NA
##      capital_gain capital_loss hours_per_week native_country income
## 32562        NA            NA                 NA

```

```

#Remover a linha N/A
train_data <- train_data[-32562, ]

#Checando se há algum exemplo que está como NA na base de teste
empty_rows <- !complete.cases(test_data)
print(test_data[empty_rows, ])

```

```

## [1] age          workclass      education_num marital_status occupation
## [6] race          sex           capital_gain  capital_loss   hours_per_week
## [11] native_country income
## <0 linhas> (ou row.names de comprimento 0)

```

## 10 e 11. Análise e aplicação de técnicas de amostragem de dados e desbalanceamento

Para realizar uma análise ou aplicar técnicas de amostragem de dados primeiro é preciso verificar se a base de dados de treinamento e teste estão desbalanceados.

```

# Checar a distribuição da base de treinamento do atributo alvo
table(train_data$income)

```

```

##
##  <=50K    >50K
##  24720    7841

```

```

# Calcular proporções
prop.table(table(train_data$income))

```

```

## <=50K      >50K
## 0.7591904 0.2408096

#Checar a distribuição da base de teste do atributo alvo
table(test_data$income)

## <=50K      >50K
## 12434     3846

prop.table(table(test_data$income))

## <=50K      >50K
## 0.7637592 0.2362408

Como podemos ver ambas as bases estão desbalanceadas, e por isso vamos aplicar técnicas de amostragem de dados e desbalanceamento, a técnica de amostragem utilizada foi amostragem aleatória simples (Sem reposição de exemplos) e para técnicas de desbalanceamento foi utilizado a técnica de undersampling (Onde igualamos os números de casos da classe majoritária com o número de casos da classe minoritária).

# Função para realizar amostragem estratificada
amostra_estratificada <- function(data, target_col) {
  # Separar as classes
  class_min <- subset(data, data[[target_col]] == " >50K")
  class_maj <- subset(data, data[[target_col]] == " <=50K")

  # Número de exemplos na classe minoritária
  num_min <- nrow(class_min)

  # Amostragem aleatória da classe majoritária
  set.seed(123) # para reproduzibilidade
  class_maj_sampled <- class_maj[sample(nrow(class_maj), num_min, replace = FALSE), ]

  # Combinar as classes amostradas
  balanced_data <- rbind(class_min, class_maj_sampled)

  # Embaralhar os dados para evitar ordenação por classe
  balanced_data <- balanced_data[sample(nrow(balanced_data)),]

  return(balanced_data)
}

# Aplicar a função nas bases de dados
train_data_balanced <- amostra_estratificada(train_data, "income")
test_data_balanced <- amostra_estratificada(test_data, "income")

# Nova distribuição da base de treinamento
table(train_data_balanced$income)

## <=50K      >50K
##    7841     7841

```

```
prop.table(table(train_data_balanced$income))
```

```
##  
## <=50K >50K  
## 0.5 0.5
```

```
# Nova distribuição da base de teste  
table(test_data_balanced$income)
```

```
##  
## <=50K >50K  
## 3846 3846
```

```
prop.table(table(test_data_balanced$income))
```

```
##  
## <=50K >50K  
## 0.5 0.5
```

## 12. Limpeza de Dados

Para a limpeza de dados, foram realizadas as seguintes etapas:

### 1. Tratamento de valores ausentes: Foram substituídos os valores ausentes pela moda de cada coluna.

Nesse caso foi optado por substituir os valores ausentes pela moda de cada coluna, pois a moda é uma medida de tendência central que representa o valor mais frequente em um conjunto de dados. Substituir os valores ausentes pela moda ajuda a manter a consistência dos dados e a preservar a distribuição original.

```
# Substituir " ?" por NA  
train_data[train_data == " ?"] <- NA  
test_data[test_data == " ?"] <- NA  
  
# Imprimir a porcentagem de valores NA em cada coluna para verificar  
print(sapply(train_data, function(col) {  
  sum(is.na(col)) * 100 / length(col)  
}))
```

```
##          age      workclass education_num marital_status      occupation  
## 0.000000 5.638647 0.000000 0.000000 5.660146  
##          race       sex capital_gain capital_loss hours_per_week  
## 0.000000 0.000000 0.000000 0.000000 0.000000  
## native_country      income  
## 1.790486 0.000000
```

```
print(sapply(test_data, function(col) {  
  sum(is.na(col)) * 100 / length(col)  
}))
```

```

##          age    workclass education_num marital_status      occupation
## 0.000000 5.915233     0.000000     0.000000      5.933661
##          race       sex   capital_gain   capital_loss hours_per_week
## 0.000000 0.000000     0.000000     0.000000     0.000000
## native_country      income
##          1.683047     0.000000

# Função para calcular a moda
Modes <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Substituir NA pela moda em cada coluna do train_data
train_data <- data.frame(lapply(train_data, function(col) {
  if (any(is.na(col))) {
    col[is.na(col)] <- Modes(col[!is.na(col)])
  }
  return(col)
}))

# Substituir NA pela moda em cada coluna do test_data
test_data <- data.frame(lapply(test_data, function(col) {
  if (any(is.na(col))) {
    col[is.na(col)] <- Modes(col[!is.na(col)])
  }
  return(col)
}))

# Imprimir a porcentagem de valores NA em cada coluna para verificar
print(sapply(train_data, function(col) {
  sum(is.na(col)) * 100 / length(col)
}))

##          age    workclass education_num marital_status      occupation
## 0            0            0            0            0            0
##          race       sex   capital_gain   capital_loss hours_per_week
## 0            0            0            0            0            0
## native_country      income
## 0            0            0            0            0            0

print(sapply(test_data, function(col) {
  sum(is.na(col)) * 100 / length(col)
}))

##          age    workclass education_num marital_status      occupation
## 0            0            0            0            0            0
##          race       sex   capital_gain   capital_loss hours_per_week
## 0            0            0            0            0            0
## native_country      income
## 0            0            0            0            0            0

```

## 2. Remoção de duplicatas: Foram removidas as linhas duplicadas da base de dados.

A remoção de duplicatas é importante para garantir a integridade dos dados e evitar viés na análise. A presença de linhas duplicadas pode distorcer os resultados e levar a conclusões errôneas. Portanto, é essencial remover duplicatas para manter a qualidade dos dados.

```
# Verificar duplicatas no train_data
duplicated_train <- duplicated(train_data)
any_duplicated_train <- any(duplicated_train)
print(paste("Existem linhas duplicadas em train_data:", any_duplicated_train))
```

```
## [1] "Existem linhas duplicadas em train_data: TRUE"
```

```
# Verificar duplicatas no test_data
duplicated_test <- duplicated(test_data)
any_duplicated_test <- any(duplicated_test)
print(paste("Existem linhas duplicadas em test_data:", any_duplicated_test))
```

```
## [1] "Existem linhas duplicadas em test_data: TRUE"
```

```
# Remover linhas duplicadas do train_data
train_data <- train_data[!duplicated(train_data), ]
```

```
# Remover linhas duplicadas do test_data
test_data <- test_data[!duplicated(test_data), ]
```

```
# Verificar novamente duplicatas no train_data
duplicated_train <- duplicated(train_data)
any_duplicated_train <- any(duplicated_train)
print(paste("Existem linhas duplicadas em train_data:", any_duplicated_train))
```

```
## [1] "Existem linhas duplicadas em train_data: FALSE"
```

```
# Verificar novamente duplicatas no test_data
duplicated_test <- duplicated(test_data)
any_duplicated_test <- any(duplicated_test)
print(paste("Existem linhas duplicadas em test_data:", any_duplicated_test))
```

```
## [1] "Existem linhas duplicadas em test_data: FALSE"
```

## 3. Criação de grupos para valores categóricos: Foram agrupados os valores categóricos em grupos mais amplos para simplificar a análise.

A criação de grupos para valores categóricos é útil para reduzir a complexidade dos dados e facilitar a análise. Agrupar valores categóricos semelhantes em categorias mais amplas pode ajudar a identificar padrões e tendências nos dados. Além disso, a criação de grupos pode melhorar a interpretação dos resultados e simplificar a visualização dos dados.

```

#Criação de grupos para alojar grupos parecidos (WorkClass)
train_data$workclass <- as.character(train_data$workclass)

train_data$workclass[train_data$workclass == " Without-pay" |
                     train_data$workclass == " Never-worked"] <- " Unemployed"

train_data$workclass[train_data$workclass == " State-gov" |
                     train_data$workclass == " Local-gov"] <- " SL-gov"

train_data$workclass[train_data$workclass == " Self-emp-inc" |
                     train_data$workclass == " Self-emp-not-inc"] <- " Self-employed"

table(train_data$workclass)

```

```

## 
##      Federal-gov      Private  Self-employed      SL-gov      Unemployed
##            939           20256       3558          3276             21

```

```

test_data$workclass <- as.character(test_data$workclass)

test_data$workclass[test_data$workclass == " Without-pay" |
                     test_data$workclass == " Never-worked"] <- " Unemployed"

test_data$workclass[test_data$workclass == " State-gov" |
                     test_data$workclass == " Local-gov"] <- " SL-gov"

test_data$workclass[test_data$workclass == " Self-emp-inc" |
                     test_data$workclass == " Self-emp-not-inc"] <- " Self-employed"

table(test_data$workclass)

```

```

## 
##      Federal-gov      Private  Self-employed      SL-gov      Unemployed
##            469           10802       1874          1702             10

```

```

#Criação de grupos para alojar grupos parecidos (Marital Status)
table(train_data$marital_status)

```

```

## 
##      Divorced      Married-AF-spouse      Married-civ-spouse
##            4067                  23                13171
##      Married-spouse-absent      Never-married      Separated
##                    418                  8393                  1002
##      Widowed
##            976

```

```

train_data$marital_status <- as.character(train_data$marital_status)

train_data$marital_status[train_data$marital_status == " Married-AF-spouse" |
                     train_data$marital_status == " Married-civ-spouse" |
                     train_data$marital_status == " Married-spouse-absent"] <- " Married"

```

```

train_data$marital_status[train_data$marital_status == " Divorced" |
                           train_data$marital_status == " Separated" |
                           train_data$marital_status == " Widowed"] <- " Not-Married"
table(train_data$marital_status)

##
##      Married  Never-married    Not-Married
##      13612          8393        6045

table(test_data$marital_status)

##
##      Divorced      Married-AF-spouse      Married-civ-spouse
##      2073                  14                6875
##  Married-spouse-absent      Never-married      Separated
##      210                  4663                 498
##      Widowed
##      524

test_data$marital_status <- as.character(test_data$marital_status)

test_data$marital_status[test_data$marital_status == " Married-AF-spouse" |
                           test_data$marital_status == " Married-civ-spouse" |
                           test_data$marital_status == " Married-spouse-absent"] <- " Married"

test_data$marital_status[test_data$marital_status == " Divorced" |
                           test_data$marital_status == " Separated" |
                           test_data$marital_status == " Widowed"] <- " Not-Married"
table(test_data$marital_status)

##
##      Married  Never-married    Not-Married
##      7099          4663        3095

#Criação de grupos para alojar grupos parecidos (Native Country)
train_data$native_country <- as.character(train_data$native_country)

north_america <- c(" Canada", " Cuba", " Dominican-Republic", " El-Salvador", " Guatemala",
                     " Haiti", " Honduras", " Jamaica", " Mexico", " Nicaragua",
                     " Outlying-US(Guam-USVI-etc)", " Puerto-Rico", " Trinadad&Tobago",
                     " United-States")
asia <- c(" Cambodia", " China", " Hong", " India", " Iran", " Japan", " Laos",
         " Philippines", " Taiwan", " Thailand", " Vietnam")
south_america <- c(" Columbia", " Ecuador", " Peru")
europe <- c(" England", " France", " Germany", " Greece", " Holand-Netherlands",
            " Hungary", " Ireland", " Italy", " Poland", " Portugal", " Scotland",
            " Yugoslavia")
other <- c(" South")
train_data$native_country[train_data$native_country %in% north_america] <- " North America"
train_data$native_country[train_data$native_country %in% asia] <- " Asia"
train_data$native_country[train_data$native_country %in% south_america] <- " South America"

```

```

train_data$native_country[train_data$native_country %in% europe] <- "Europe"
train_data$native_country[train_data$native_country %in% other] <- "Other"

table(train_data$native_country)

##          Asia      Europe North America           Other South America
##        669         519       26664            80          118

test_data$native_country <- as.character(test_data$native_country)

north_america <- c(" Canada", " Cuba", " Dominican-Republic", " El-Salvador", " Guatemala",
                   " Haiti", " Honduras", " Jamaica", " Mexico", " Nicaragua",
                   " Outlying-US(Guam-USVI-etc)", " Puerto-Rico", " Trinadad&Tobago",
                   " United-States")
asia <- c(" Cambodia", " China", " Hong", " India", " Iran", " Japan", " Laos",
         " Philippines", " Taiwan", " Thailand", " Vietnam")
south_america <- c(" Columbia", " Ecuador", " Peru")
europe <- c(" England", " France", " Germany", " Greece", " Holand-Netherlands",
            " Hungary", " Ireland", " Italy", " Poland", " Portugal", " Scotland",
            " Yugoslavia")
other <- c(" South")

test_data$native_country[test_data$native_country %in% north_america] <- "North America"
test_data$native_country[test_data$native_country %in% asia] <- "Asia"
test_data$native_country[test_data$native_country %in% south_america] <- "South America"
test_data$native_country[test_data$native_country %in% europe] <- "Europe"
test_data$native_country[test_data$native_country %in% other] <- "Other"

table(test_data$native_country)

##          Asia      Europe North America           Other South America
##        310         259       14195            35          58

```

## Tabela de Dados (Treinamento) após a limpeza

Table 5: Adult Database - Train Data - Part 1

age	workclass	education_num	marital_status	occupation
39	SL-gov	13	Never-married	Adm-clerical
50	Self-employed	13	Married	Exec-managerial
38	Private	9	Not-Married	Handlers-cleaners
53	Private	7	Married	Handlers-cleaners
28	Private	13	Married	Prof-specialty

Table 6: Adult Database - Train Data - Part 2

race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
White	Male	2174	0	40	North America	<=50K
White	Male	0	0	13	North America	<=50K
White	Male	0	0	40	North America	<=50K
Black	Male	0	0	40	North America	<=50K
Black	Female	0	0	40	North America	<=50K

### Tabela de Dados (Teste) após a limpeza

Table 7: Adult Database - Test Data - Part 1

age	workclass	education_num	marital_status	occupation
38	Private	9	Married	Farming-fishing
28	SL-gov	12	Married	Protective-serv
44	Private	10	Married	Machine-op-inspct
18	Private	10	Never-married	Prof-specialty
34	Private	6	Never-married	Other-service

Table 8: Adult Database - Test Data - Part 2

race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
White	Male	0	0	50	North America	<=50K
White	Male	0	0	40	North America	>50K
Black	Male	7688	0	40	North America	>50K
White	Female	0	0	30	North America	<=50K
White	Male	0	0	30	North America	<=50K

## 13. Identificação e conversão dos tipos de dados

Para a identificação e conversão dos tipos de dados, foram realizadas a conversão de tipo de dados simbólicos para numéricos, a fim de facilitar a análise e a modelagem dos dados.

```
# Conversão dos valores de 'workclass' e 'marital_status'
table(train_data$workclass)

## 
##   Federal-gov      Private Self-employed        SL-gov      Unemployed
##       939           20256         3558            3276          21

train_data$workclass <- ifelse(train_data$workclass == "Federal-gov", 0, train_data$workclass)
train_data$workclass <- ifelse(train_data$workclass == "Private", 1, train_data$workclass)
train_data$workclass <- ifelse(train_data$workclass == "Self-employed", 2, train_data$workclass)
train_data$workclass <- ifelse(train_data$workclass == "SL-gov", 3, train_data$workclass)
train_data$workclass <- ifelse(train_data$workclass == "Unemployed", 4, train_data$workclass)

table(test_data$workclass)
```

```

##          Federal-gov      Private   Self-employed       SL-gov      Unemployed
##             469            10802        1874           1702            10

test_data$workclass <- ifelse(test_data$workclass == " Federal-gov", 0, test_data$workclass)
test_data$workclass <- ifelse(test_data$workclass == " Private", 1, test_data$workclass)
test_data$workclass <- ifelse(test_data$workclass == " Self-employed", 2, test_data$workclass)
test_data$workclass <- ifelse(test_data$workclass == " SL-gov", 3, test_data$workclass)
test_data$workclass <- ifelse(test_data$workclass == " Unemployed", 4, test_data$workclass)






```

```

train_data$occupation <- ifelse(train_data$occupation == " Priv-house-serv", 8, train_data$occupation)
train_data$occupation <- ifelse(train_data$occupation == " Prof-specialty", 9, train_data$occupation)
train_data$occupation <- ifelse(train_data$occupation == " Protective-serv", 10, train_data$occupation)
train_data$occupation <- ifelse(train_data$occupation == " Sales", 11, train_data$occupation)
train_data$occupation <- ifelse(train_data$occupation == " Tech-support", 12, train_data$occupation)
train_data$occupation <- ifelse(train_data$occupation == " Transport-moving", 13, train_data$occupation)






```

```

test_data$occupation <- ifelse(test_data$occupation == " Adm-clerical", 0, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Armed-Forces", 1, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Craft-repair", 2, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Exec-managerial", 3, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Farming-fishing", 4, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Handlers-cleaners", 5, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Machine-op-inspct", 6, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Other-service", 7, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Priv-house-serv", 8, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Prof-specialty", 9, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Protective-serv", 10, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Sales", 11, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Tech-support", 12, test_data$occupation)
test_data$occupation <- ifelse(test_data$occupation == " Transport-moving", 13, test_data$occupation)

# Conversão dos valores 'race'





```

```

train_data$race <- ifelse(train_data$race == " Amer-Indian-Eskimo", 0, train_data$race)
train_data$race <- ifelse(train_data$race == " Asian-Pac-Islander", 1, train_data$race)
train_data$race <- ifelse(train_data$race == " Black", 2, train_data$race)
train_data$race <- ifelse(train_data$race == " Other", 3, train_data$race)
train_data$race <- ifelse(train_data$race == " White", 4, train_data$race)






```

```

## Amer-Indian-Eskimo Asian-Pac-Islander           Black          Other
##                  159                 480           1521            135
##                  White
##                 12562

test_data$race <- ifelse(test_data$race == "Amer-Indian-Eskimo", 0, test_data$race)
test_data$race <- ifelse(test_data$race == "Asian-Pac-Islander", 1, test_data$race)
test_data$race <- ifelse(test_data$race == "Black", 2, test_data$race)
test_data$race <- ifelse(test_data$race == "Other", 3, test_data$race)
test_data$race <- ifelse(test_data$race == "White", 4, test_data$race)

# Conversão dos valores 'sex'
table(train_data$sex)

```

```

##
## Female     Male
##    9314    18736

train_data$sex <- ifelse(train_data$sex == "Male", 0, train_data$sex)
train_data$sex <- ifelse(train_data$sex == "Female", 1, train_data$sex)

table(test_data$sex)

```

```

##
## Female     Male
##    4947    9910

test_data$sex <- ifelse(test_data$sex == "Male", 0, train_data$sex)
test_data$sex <- ifelse(test_data$sex == "Female", 1, train_data$sex)

# Conversão dos valores 'native_country'

table(train_data$native_country)

```

```

##
##          Asia        Europe North America           Other   South America
##             669          519      26664                80            118

train_data$native_country <- ifelse(train_data$native_country == "North America", 0, train_data$native_country)
train_data$native_country <- ifelse(train_data$native_country == "Asia", 1, train_data$native_country)
train_data$native_country <- ifelse(train_data$native_country == "South America", 2, train_data$native_country)
train_data$native_country <- ifelse(train_data$native_country == "Europe", 3, train_data$native_country)
train_data$native_country <- ifelse(train_data$native_country == "Other", 4, train_data$native_country)

table(test_data$native_country)

```

```

##
##          Asia        Europe North America           Other   South America
##             310          259      14195                35            58

```

```

test_data$native_country <- ifelse(test_data$native_country == " North America", 0, test_data$native_country)
test_data$native_country <- ifelse(test_data$native_country == " Asia", 1, test_data$native_country)
test_data$native_country <- ifelse(test_data$native_country == " South America", 2, test_data$native_country)
test_data$native_country <- ifelse(test_data$native_country == " Europe", 3, test_data$native_country)
test_data$native_country <- ifelse(test_data$native_country == " Other", 4, test_data$native_country)

# Conversão dos valores 'income'
table(train_data$income)

## 
##   <=50K    >50K
##   20873    7177

train_data$income <- ifelse(train_data$income == "<=50K", 0, train_data$income)
train_data$income <- ifelse(train_data$income == ">50K", 1, train_data$income)

table(test_data$income)

## 
##   <=50K    >50K
##   11214    3643

test_data$income <- ifelse(test_data$income == "<=50K", 0, test_data$income)
test_data$income <- ifelse(test_data$income == ">50K", 1, test_data$income)

```

## Tabela de Dados (Treinamento) após a conversão

Table 9: Adult Database - Train Data - Part 1

age	workclass	education_num	marital_status	occupation
39	3	13	2	0
50	2	13	0	3
38	1	9	1	5
53	1	7	0	5
28	1	13	0	9

Table 10: Adult Database - Train Data - Part 2

race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
4	0	2174	0	40	0	0
4	0	0	0	13	0	0
4	0	0	0	40	0	0
2	0	0	0	40	0	0
2	1	0	0	40	0	0

## Tabela de Dados (Teste) após a conversão

Table 11: Adult Database - Test Data - Part 1

age	workclass	education_num	marital_status	occupation
38	1	9	0	4
28	3	12	0	10
44	1	10	0	6
18	1	10	2	9
34	1	6	2	7

Table 12: Adult Database - Test Data - Part 2

race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
4	0	0	0	50	0	0
4	0	0	0	40	0	1
2	0	7688	0	40	0	1
4	0	0	0	30	0	0
4	1	0	0	30	0	0

## 14. Análise e aplicação de alguma técnica para redução de dimensionalidade

O PCA é uma técnica de transformação linear que busca identificar as principais direções de variação nos dados e projetá-los em um novo espaço de menor dimensionalidade

```
head(train_data_balanced)
```

```
##      age      workclass education_num      marital_status      occupation
## 10985 19 Self-emp-not-inc          9 Never-married Adm-clerical
## 13876 44           Private          9            Divorced Adm-clerical
## 21531 55           Private         13 Married-civ-spouse Prof-specialty
## 4953 45 Self-emp-inc          10 Married-civ-spouse Exec-managerial
## 6017 65           Private          9 Married-civ-spouse        Sales
## 29355 32           Private         13            Divorced        Sales
##             race      sex capital_gain capital_loss hours_per_week
## 10985 Asian-Pac-Islander Female          0          0            30
## 13876 White Female          0          0            40
## 21531 White Male            0          0            60
## 4953 White Male            0          0            50
## 6017 White Male          9386          0            40
## 29355 White Female          0          0            30
##      native_country income
## 10985 United-States <=50K
## 13876 United-States <=50K
## 21531 United-States >50K
## 4953 United-States >50K
## 6017 United-States >50K
## 29355 United-States <=50K
```

```

# Normalização das variáveis numéricas
numeric_cols <- sapply(train_data_balanced, is.numeric)
train_data_num <- train_data_balanced[, numeric_cols]

# Transformação de variáveis categóricas em variáveis dummy
train_data_cat <- train_data_balanced[, !numeric_cols]

# Usar a função model.matrix para criar variáveis dummy
dummies <- model.matrix(~ . - 1, data = train_data_cat) # -1 para remover o intercepto

# Combinar dados normalizados e variáveis dummy
train_data_processed <- cbind(scale(train_data_num), dummies)

# Aplicar PCA
pca_result <- prcomp(train_data_processed, center = TRUE, scale. = TRUE)

# Resumo dos resultados da PCA
summary(pca_result)

```

```

## Importance of components:
##          PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation   1.85525 1.57777 1.43430 1.32317 1.28510 1.21362 1.20060
## Proportion of Variance 0.04302 0.03112 0.02572 0.02188 0.02064 0.01841 0.01802
## Cumulative Proportion 0.04302 0.07414 0.09986 0.12174 0.14238 0.16080 0.17881
##          PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation   1.16703 1.12182 1.11435 1.09840 1.07927 1.0769 1.0657
## Proportion of Variance 0.01702 0.01573 0.01552 0.01508 0.01456 0.0145 0.0142
## Cumulative Proportion 0.19584 0.21157 0.22709 0.24217 0.25673 0.2712 0.2854
##          PC15     PC16     PC17     PC18     PC19     PC20     PC21
## Standard deviation   1.06074 1.05167 1.04284 1.0393 1.03631 1.03365 1.03065
## Proportion of Variance 0.01406 0.01382 0.01359 0.0135 0.01342 0.01336 0.01328
## Cumulative Proportion 0.29949 0.31332 0.32691 0.3404 0.35384 0.36719 0.38047
##          PC22     PC23     PC24     PC25     PC26     PC27     PC28
## Standard deviation   1.02740 1.0238 1.01621 1.01274 1.00941 1.00660 1.00374
## Proportion of Variance 0.01319 0.0131 0.01291 0.01282 0.01274 0.01267 0.01259
## Cumulative Proportion 0.39366 0.4068 0.41968 0.43250 0.44523 0.45790 0.47049
##          PC29     PC30     PC31     PC32     PC33     PC34     PC35
## Standard deviation   1.00225 1.00176 1.00158 1.00138 1.00104 1.00101 1.00097
## Proportion of Variance 0.01256 0.01254 0.01254 0.01253 0.01253 0.01253 0.01252
## Cumulative Proportion 0.48305 0.49559 0.50813 0.52067 0.53319 0.54572 0.55824
##          PC36     PC37     PC38     PC39     PC40     PC41     PC42
## Standard deviation   1.00084 1.00083 1.00066 1.00062 1.00053 1.00037 1.00025
## Proportion of Variance 0.01252 0.01252 0.01252 0.01252 0.01251 0.01251 0.01251
## Cumulative Proportion 0.57076 0.58328 0.59580 0.60831 0.62083 0.63334 0.64584
##          PC43     PC44     PC45     PC46     PC47     PC48     PC49
## Standard deviation   1.0002 0.99964 0.99896 0.99840 0.99789 0.99741 0.99669
## Proportion of Variance 0.0125 0.01249 0.01247 0.01246 0.01245 0.01244 0.01242
## Cumulative Proportion 0.6583 0.67084 0.68331 0.69577 0.70822 0.72066 0.73307
##          PC50     PC51     PC52     PC53     PC54     PC55     PC56
## Standard deviation   0.99535 0.99438 0.99328 0.99131 0.98984 0.98600 0.98246
## Proportion of Variance 0.01238 0.01236 0.01233 0.01228 0.01225 0.01215 0.01207
## Cumulative Proportion 0.74546 0.75782 0.77015 0.78243 0.79468 0.80683 0.81890
##          PC57     PC58     PC59     PC60     PC61     PC62     PC63

```

```

## Standard deviation      0.98076 0.97479 0.96903 0.96243 0.95724 0.95006 0.94462
## Proportion of Variance 0.01202 0.01188 0.01174 0.01158 0.01145 0.01128 0.01115
## Cumulative Proportion   0.83092 0.84280 0.85454 0.86612 0.87757 0.88885 0.90001
##                           PC64     PC65     PC66     PC67     PC68     PC69     PC70
## Standard deviation      0.93134 0.92042 0.91698 0.90125 0.86792 0.83493 0.80725
## Proportion of Variance 0.01084 0.01059 0.01051 0.01015 0.00942 0.00871 0.00815
## Cumulative Proportion   0.91085 0.92144 0.93195 0.94210 0.95152 0.96023 0.96838
##                           PC71     PC72     PC73     PC74     PC75     PC76     PC77
## Standard deviation      0.74874 0.72359 0.67805 0.62829 0.53159 0.41765 0.31770
## Proportion of Variance 0.00701 0.00654 0.00575 0.00493 0.00353 0.00218 0.00126
## Cumulative Proportion   0.97539 0.98193 0.98768 0.99261 0.99614 0.99832 0.99959
##                           PC78     PC79     PC80
## Standard deviation      0.18195 7.351e-15 6.179e-15
## Proportion of Variance 0.00041 0.000e+00 0.000e+00
## Cumulative Proportion   1.00000 1.000e+00 1.000e+00

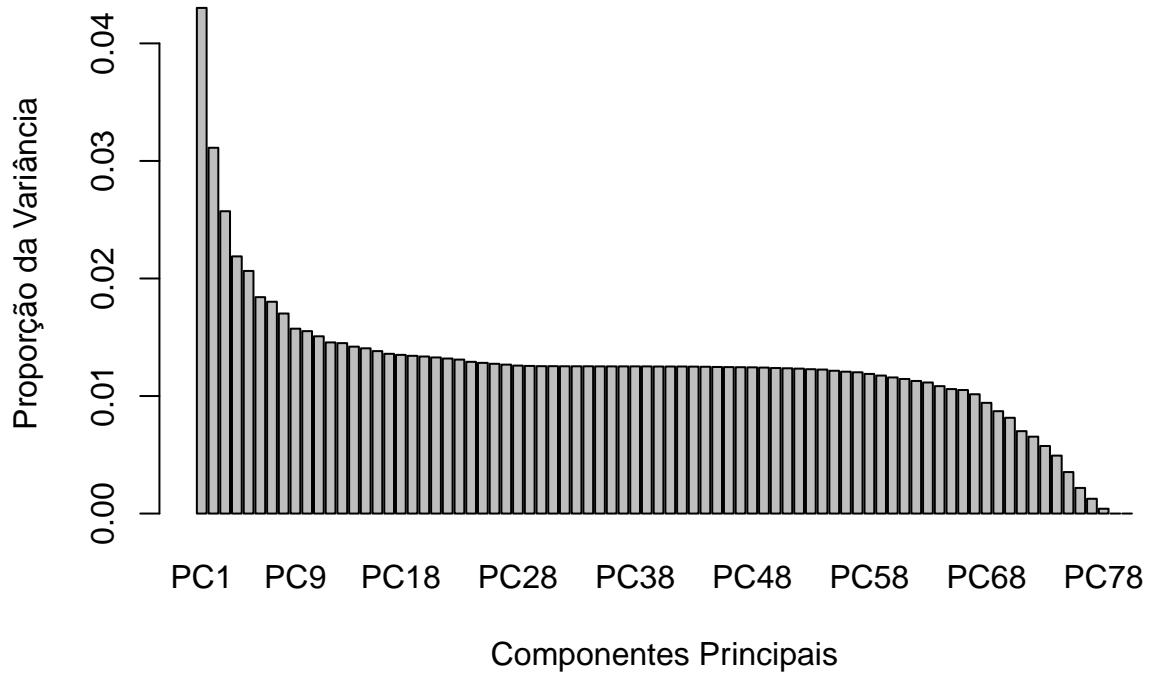
```

```

# Gráfico da variância explicada
variance_explained <- summary(pca_result)$importance[2, ]
barplot(variance_explained, main = "Proporção da Variância Explicada",
         xlab = "Componentes Principais", ylab = "Proporção da Variância")

```

## Proporção da Variância Explicada



```

# Cálculo do número de componentes que explicam 95% da variância
cumulative_variance <- cumsum(variance_explained)
num_components <- which(cumulative_variance >= 0.95)[1]

cat("Número de componentes principais que explicam 95% da variancia:", num_components)

```

```
## Numero de componentes principais que explicam 95% da variancia: 68
```

```
# Obter as componentes principais
pca_data <- as.data.frame(pca_result$x[, 1:num_components])

# Visualizar os dados transformados
head(pca_data)
```

```
##          PC1         PC2         PC3         PC4         PC5         PC6
## 10985  3.318954  2.77250698 -2.04832032  0.10663767  0.796084158 -1.8718603
## 13876  1.698503 -1.02492863 -0.45907788 -0.40241741  0.255213983  1.4018625
## 21531 -2.241859  0.07598404  0.01247386 -1.19097435  0.042216843  0.1929800
## 4953   -2.667352  0.03256877 -0.04744857  0.51909446 -0.008451272  0.4365713
## 6017   -1.496748 -0.47738334  1.18677980 -0.02194607 -0.597853594  0.9009233
## 29355  1.245542 -1.06152683 -0.20975170 -1.16506739  0.562511196  0.5570396
##          PC7         PC8         PC9         PC10        PC11        PC12
## 10985 -3.0682985  0.1935011  0.5208678 -1.8833194 -0.5345813  0.90397951
## 13876 -0.5294335  0.8029492  0.6376772 -1.2779277 -0.2019724  0.98715993
## 21531  0.8368687  0.2050584 -1.4194250 -0.2854555 -0.2088286  0.12904818
## 4953   -0.9104497 -1.9424199  2.4401188  1.2780569  0.5843696 -0.03716053
## 6017   -0.8029693 -0.1992145 -0.8247769  0.7603982  0.6582941  0.04375539
## 29355 -0.8959635 -0.5697008 -0.7422815  0.6555127  0.4847345  0.17826265
##          PC13        PC14        PC15        PC16        PC17
## 10985 -0.749769757  0.25976132  0.1552774558 -0.66188140 -0.23416138
## 13876 -0.960587677 -0.12144352 -0.0002059415 -0.60789390 -0.59837119
## 21531 -0.009209015  0.01035558 -0.4324687280 -0.04285794 -0.24128143
## 4953   0.473775498  0.11123599  0.0108926712 -0.66937141 -0.01727034
## 6017   -1.872339095 -0.37990013  0.3193238244  0.35483557  0.38586044
## 29355 -1.687910855 -0.32664067  0.4899657722  0.28151458  0.49984293
##          PC18        PC19        PC20        PC21        PC22        PC23
## 10985 -0.30554657 -0.48466776 -0.424392336 -0.01424211  0.42525586 -0.16390900
## 13876  0.19279759 -0.22326151 -0.158700974  0.07741712  0.31147774 -0.52836098
## 21531  0.11068540 -0.07815504  0.478667112 -0.19039183 -0.16000968  0.09483692
## 4953   0.89118229  0.12638185 -0.012453973  0.48338527 -0.23459400  0.07577108
## 6017   0.01660159  0.45165788 -0.006261253  0.50029686  0.00411575 -0.25864107
## 29355 -0.35980692  0.07175060 -0.604856553  0.29177290 -0.01267192 -0.34972185
##          PC24        PC25        PC26        PC27        PC28        PC29
## 10985  0.628762728  0.35998558  0.82987903 -0.09528562 -0.03708090 -0.061711589
## 13876  0.178253983  0.13552633  0.17940544 -0.29412657  0.05314705 -0.069728818
## 21531  0.007076688  0.06639581 -0.20176209 -0.07567153 -0.06295469  0.045870765
## 4953   -0.776765302 -0.12000785  0.16546592 -0.45611396 -0.29354625 -0.062901109
## 6017   0.393403450  0.03920787 -0.02567288  0.48417868  0.02995085 -0.063286876
## 29355  0.293201609 -0.18875513 -0.17207068  0.38375415 -0.13157335  0.009577616
##          PC30        PC31        PC32        PC33        PC34
## 10985  0.060987027  0.0755892885  0.05894441 -0.031886850 -0.01262061
## 13876  0.003589161  0.0002161545 -0.02283677  0.027197156 -0.02386341
## 21531 -0.029858847  0.0295054769 -0.09469785 -0.007393458 -0.04119321
## 4953   -0.015658132  0.0524881722  0.02220096 -0.043263199  0.05575181
## 6017   0.051658171 -0.0500650324 -0.04276844  0.027590569  0.01262167
## 29355  0.058964636  0.0203959660 -0.01877501  0.034750215 -0.01394271
##          PC35        PC36        PC37        PC38        PC39
## 10985  0.001620154  0.01125461  0.0022136253  0.008444167  0.035832902
## 13876  0.028012859  0.01370314  0.0255112656 -0.005968358 -0.012512276
## 21531  0.024744520  0.03229201  0.0090939422 -0.025271375 -0.008890666
```

```

## 4953 -0.063317978 -0.08888480  0.0184803933  0.008203390 -0.014046398
## 6017 -0.034218493 -0.01405125 -0.0000681203  0.003054551 -0.019652683
## 29355 -0.069107127  0.02562954 -0.0020276786  0.008112562  0.006871949
##          PC40        PC41        PC42        PC43        PC44
## 10985  0.004541627  0.009197097  1.506303e-02  0.02334100  0.130409037
## 13876 -0.003578200 -0.003812765 -2.032480e-03 -0.03227411 -0.012466750
## 21531  0.008452770  0.007429750 -2.041192e-02 -0.03285491  0.061505104
## 4953   0.006193002  0.003200244 -8.403024e-05 -0.01279923 -0.145750196
## 6017   -0.018865964 -0.017077473  1.104083e-02 -0.04725729 -0.091049032
## 29355 -0.023160458 -0.019934115  4.425658e-02 -0.03008786  0.008075751
##          PC45        PC46        PC47        PC48        PC49        PC50
## 10985 -0.075368421 -0.15423342  0.11240138  0.04256095  0.20136360 -0.01219385
## 13876 -0.075771713  0.01683160  0.06046817 -0.15090182 -0.04938821 -0.01651339
## 21531  0.016425505  0.06549947  0.02319072 -0.19532944 -0.12232717 -0.18208057
## 4953   0.002164421 -0.19778017 -0.01940760 -0.05559911  0.25513015  0.21651476
## 6017   -0.084308768  0.09998961 -0.26565506  0.11815645  0.21481229  0.33364902
## 29355 -0.189367944  0.15211167 -0.29031669  0.25395870  0.19944831  0.24528225
##          PC51        PC52        PC53        PC54        PC55        PC56
## 10985 -0.28928272  0.1752920  0.23709722 -0.7502519  0.45088098 -0.25511089
## 13876 -0.32178411  0.2232792 -0.01566803 -0.3099723  0.46926855 -0.54902791
## 21531  0.15813522  0.1872602  0.15311887  0.2496729 -0.03841195 -0.33497542
## 4953   -0.05796201 -0.4985684 -0.59499856 -0.2590766 -0.44666147  0.04397378
## 6017   0.10277755  0.3053885  0.19590228  0.2066194  0.23086139 -0.07769049
## 29355 -0.07355774  0.1287047  0.14587167  0.1601884  0.11794945 -0.07868770
##          PC57        PC58        PC59        PC60        PC61        PC62
## 10985 -0.19975524  0.10770508 -0.70556043 -0.1881239 -0.24317800 -1.2216620
## 13876  0.38916321 -0.02078182 -0.11919258 -0.5719069 -0.47547001 -0.7277097
## 21531  0.04755684  0.06695005  0.17431947  0.1392527  0.06393287  0.4410674
## 4953   0.38984078 -0.05490558  0.44406156  0.6299892 -1.31190028  0.6084690
## 6017   -0.11506412 -0.47248126 -0.05185699 -0.2952167  0.90050991 -0.2796433
## 29355 -0.01292286 -0.25091944 -0.06561728 -0.3404165  0.70218635  0.1129535
##          PC63        PC64        PC65        PC66        PC67        PC68
## 10985 -0.39768922 -0.1441893  0.8219270  0.54223898 -1.9226411  0.40504626
## 13876 -0.44993339 -0.3063297  0.5252418 -0.34649425 -0.1264649  0.02087312
## 21531 -0.05068197  0.3995391 -0.2738986 -0.76995591  0.6749537 -0.10074012
## 4953   -0.44101898  0.4164298  0.9918082 -0.94501533 -0.4227189  0.40269509
## 6017   -0.09624543  0.0362411 -0.1348001  0.20950031  0.2375012 -0.29290506
## 29355 -0.13893925  0.5481909 -0.3242977 -0.06196783 -0.3473413 -0.20915794

```

A análise da proporção da variância é fundamental na PCA porque ajuda a decidir quantos componentes principais manter. O objetivo é reduzir a dimensionalidade dos dados enquanto retém a maior parte da informação relevante. A regra geral é manter componentes que, juntos, expliquem uma porcentagem significativa da variância (como 90% ou 95%). Neste caso, o número de componentes principais que explicam 95% da variância é 68. Portanto, podemos reduzir a dimensionalidade dos dados de 80 para 68 mantendo a maioria da informação relevante.

## 15. Definição da técnica de validação a ser utilizada

Para a base de dados Adult, vamos utilizar a Validação Cruzada k-fold onde k = 10, vamos utilizar a técnica de k-fold para evitar o overfitting e proporcionar uma estimativa robusta do desempenho do modelo.

```
# Definindo o número de folds para a validação cruzada
k <- 10

# Função para realizar a validação cruzada
cross_val <- function(model, data, k) {
  folds <- cut(seq(1,nrow(data)), breaks=k, labels=FALSE)
  results <- list()

  for(i in 1:k) {
    # Segmentando os dados em treino e teste
    testIndexes <- which(folds == i, arr.ind=TRUE)
    test_data <- data[testIndexes, ]
    train_data <- data[-testIndexes, ]

    # Treinando o modelo
    model_fit <- model(trainData)

    # Previsões no conjunto de teste
    predictions <- predict(model_fit, testData[-ncol(testData)])

    # Armazenando resultados
    results[[i]] <- data.frame(Actual = testData[,ncol(testData)], Predicted = predictions)
  }

  return(results)
}
```