Bikes_Rent.R

rodrigolima82

2019-05-30

```
# Coleta e Transformação de Dados

# Este código contém comandos para filtrar e transformar os dados de aluguel de bikes,
# dados que estão em nosso dataset.

source("src/Tools.R")
bikes <- read.csv("bikes.csv", sep = ",", header = TRUE, stringsAsFactors = FALSE )

# Selecionar as variáveis que serão usadas
cols <- c("dteday", "mnth", "hr", "holiday",
          "workingday", "weathersit", "temp",
          "hum", "windspeed", "cnt")

# Criando um subset dos dados
bikes <- bikes[, cols]

# Transformar o objeto de data
bikes$dteday <- char.toPOSIXct(bikes)

# Esta linha acima gera dois valores NA
# Esta linha abaixo corrige
bikes <- na.omit(bikes)

# Normalizar as variaveis preditoras numericas
cols <- c("temp", "hum", "windspeed")
bikes[, cols] <- scale(bikes[, cols])

#str(bikes)
#View(bikes)

# Criar uma nova variável para indicar dia da semana (workday)
bikes$isWorking <- ifelse(bikes$workingday & !bikes$holiday, 1, 0)

# Adicionar uma coluna com a quantidade de meses, o que vai ajudar a criar o modelo
bikes <- month.count(bikes)

# Criar um fator ordenado para o dia da semana, comecando por segunda-feira
# Neste fator eh convertido para ordenado numérico para ser compativel com os tipos de dados do Azure M
bikes$dayWeek <- as.factor(weekdays(bikes$dteday))


############ ATENÇÃO ############

# ==> Analise o dataframe bikes.
# Se os nomes dos dias da semana estiverem em português na coluna bikes$dayWeek,
# execute o Bloco1 abaixo, caso contrátio, execute o Bloco2 com os nomes em inglês.
# Execute um bloco ou o outro.
```

```
str(bikes$dayWeek)
```

```
##  Factor w/ 7 levels "Friday","Monday",..: 3 3 3 3 3 3 3 3 3 3 ...
```

```
# Bloco1
# Se o seu sistema operacional estiver em português, execute o comando abaixo.
#bikes$dayWeek <- as.numeric(ordered(bikes$dayWeek,
#                                    levels = c("segunda",
#                                               "terça",
#                                               "quarta",
#                                               "quinta",
#                                               "sexta",
#                                               "sábado",
#                                               "domingo")))

# Bloco2
# Se o seu sistema operacional estiver em inglês, execute o comando abaixo.
bikes$dayWeek <- as.numeric(ordered(bikes$dayWeek,
                                    levels = c("Monday",
                                               "Tuesday",
                                               "Wednesday",
                                               "Thursday",
                                               "Friday",
                                               "Saturday",
                                               "Sunday")))

# Agora os dias da semana devem estar como valores numéricos
# Se estiverem como valores NA, volte e verifique se você seguiu as instruções acima.
str(bikes$dayWeek)
```

```
##  num [1:17377] 6 6 6 6 6 6 6 6 6 6 ...
```

```
str(bikes)
```

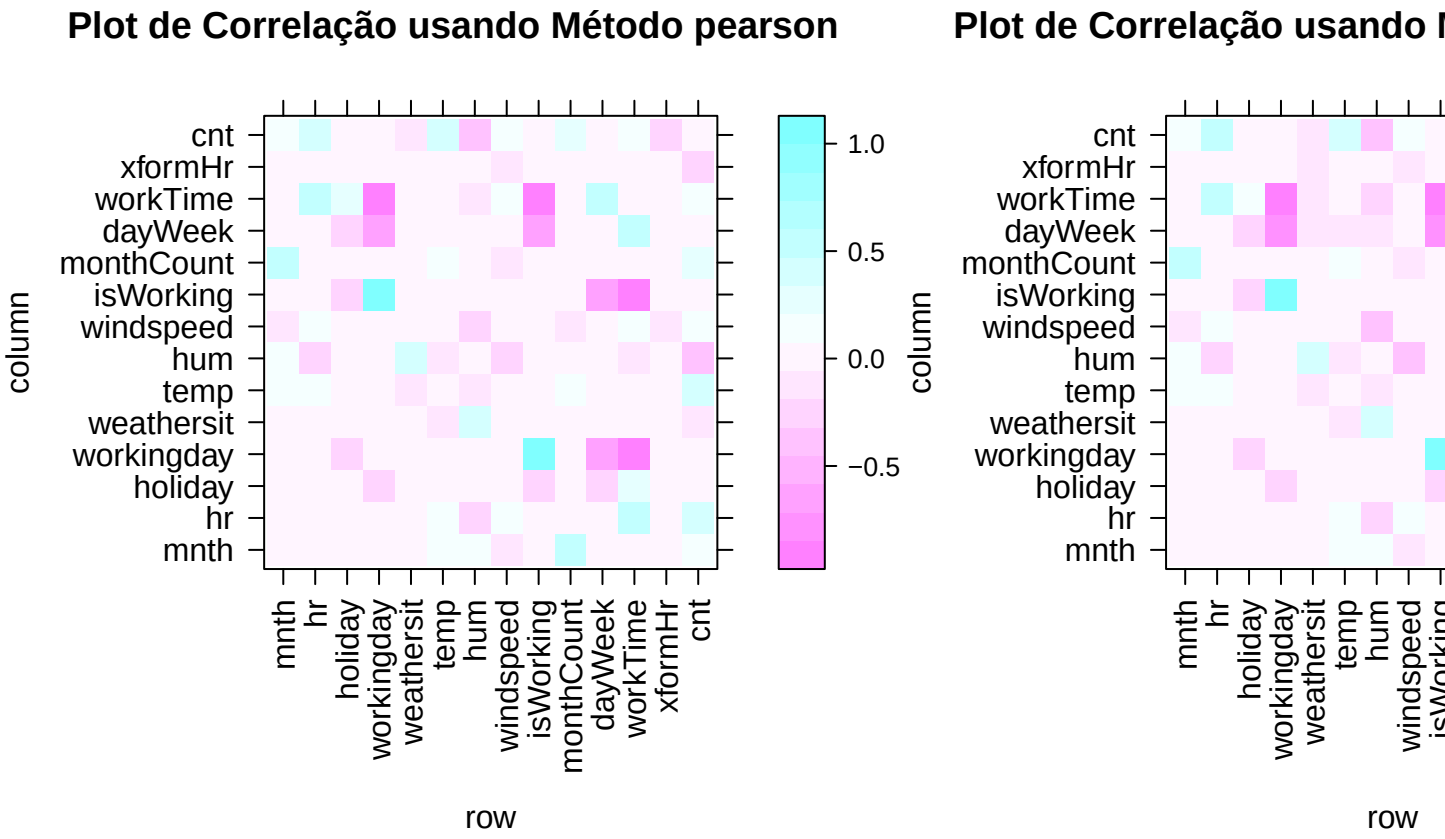
```
## 'data.frame':    17377 obs. of  13 variables:
##  $ dteday    : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01 01:00:00" ...
##  $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hr        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weathersit: int  1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  -1.33 -1.44 -1.44 -1.33 -1.33 ...
##  $ hum       : num  0.947 0.895 0.895 0.636 0.636 ...
##  $ windspeed : num  -1.55 -1.55 -1.55 -1.55 -1.55 ...
##  $ cnt       : int  16 40 32 13 1 1 2 3 8 14 ...
##  $ isWorking : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthCount: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ dayWeek   : num  6 6 6 6 6 6 6 6 6 6 ...
##  - attr(*, "na.action")= 'omit' Named int  6803 15692
##   ..- attr(*, "names")= chr  "6803" "15692"
```

```
# Adiciona uma variável com valores únicos para o horário do dia em dias de semana e dias de fim de sem
# Com isso diferenciamos as horas dos dias de semana, das horas em dias de fim de semana
bikes$workTime <- ifelse(bikes$isWorking, bikes$hr, bikes$hr + 24)

# Transforma os valores de hora na madrugada, quando a demanda por bibicletas é praticamente nula
```

```r
bikes$xformHr <- ifelse(bikes$hr > 4, bikes$hr - 5, bikes$hr + 19)

# Adiciona uma variável com valores únicos para o horário do dia para dias de semana e dias de fim de s
# Considerando horas da madrugada
bikes$xformWorkHr <- ifelse(bikes$isWorking, bikes$xformHr, bikes$xformHr + 24)

# str(bikes)
# View(bikes)
# O trabalho até aqui também é chamado de Feature Engineering ou Engenharia de Atributos


# Análise de Correlação

# Este código contém comandos para análise de correlação.
#View(bikes)

# Definindo as colunas para a análise de correlação
cols <- c("mnth", "hr", "holiday", "workingday",
          "weathersit", "temp", "hum", "windspeed",
          "isWorking", "monthCount", "dayWeek",
          "workTime", "xformHr", "cnt")

# Métodos de Correlação
# Pearson - coeficiente usado para medir o grau de relacionamento entre duas variáveis com relação line
# Spearman - teste não paramétrico, para medir o grau de relacionamento entre duas variaveis
# Kendall - teste não paramétrico, para medir a força de dependência entre duas variaveis

# Vetor com os métodos de correlação
metodos <- c("pearson", "spearman")

# Aplicando os métodos de correlação com a função cor()
cors <- lapply(metodos, function(method)
  (cor(bikes[, cols], method = method)))

#head(cors)

# Preprando o plot
require(lattice)
```

```
## Loading required package: lattice
```

```r
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot( levelplot(x,
                  main = paste("Plot de Correlação usando Método", labs),
                  scales = list(x = list(rot = 90), cex = 1.0)) )
}

# Mapa de Correlação
Map(plot.cors, cors, metodos)
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

```
# Análise de Série Temporal

# Este código contém comandos para análise de série temporal
# Avaliando a demanda por aluguel de bikes ao longo do tempo
# Construindo um time series plot para alguns determinados horários
# em dias úteis e dias de fim de semana.
times <- c(7, 9, 12, 15, 18, 20, 22)

# Time Series Plot
tms.plot <- function(times){
  ggplot(bikes[bikes$workTime == times, ], aes(x = dteday, y = cnt)) +
    geom_line() +
    ylab("Numero de Bikes") +
    labs(title = paste("Demanda de Bikes as ", as.character(times), ":00", sep = "")) +
    theme(text = element_text(size = 20))
}

require(ggplot2)
```

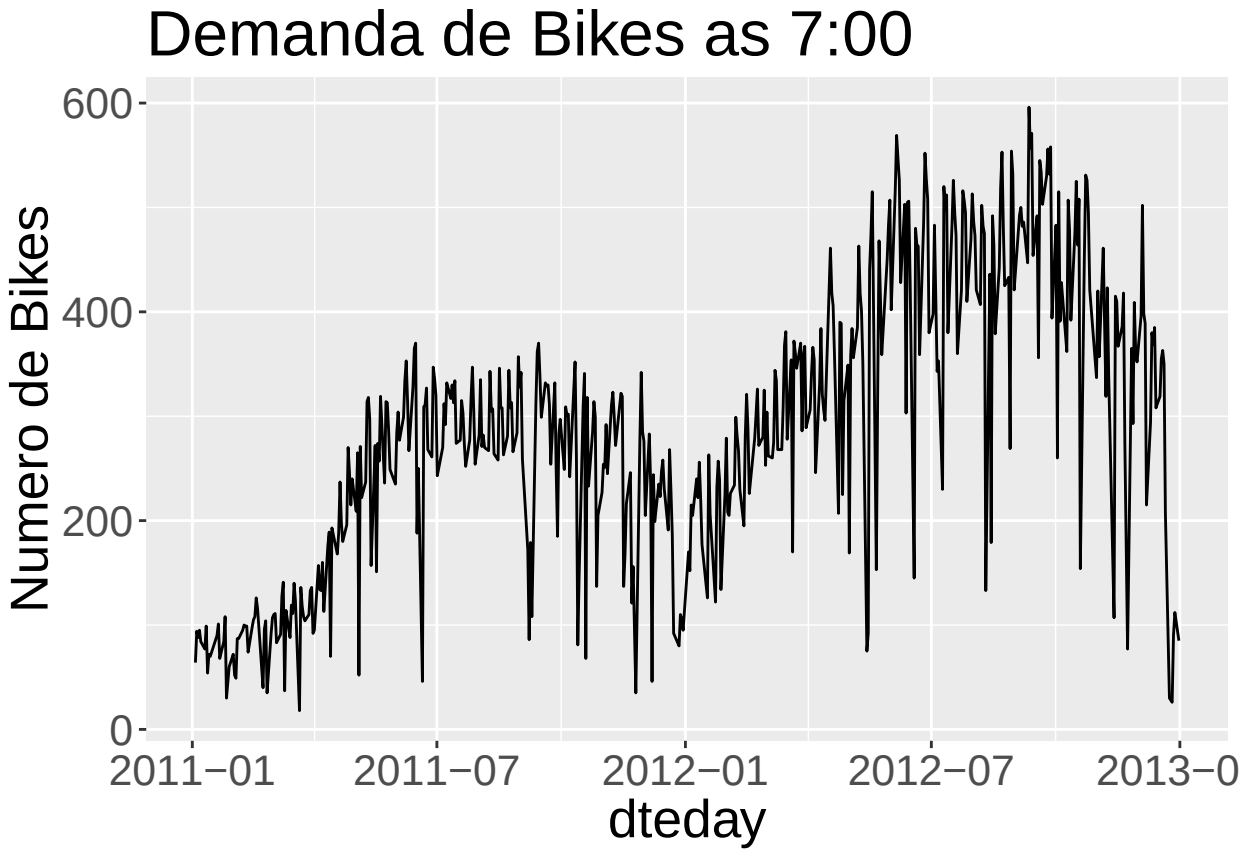
```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
```

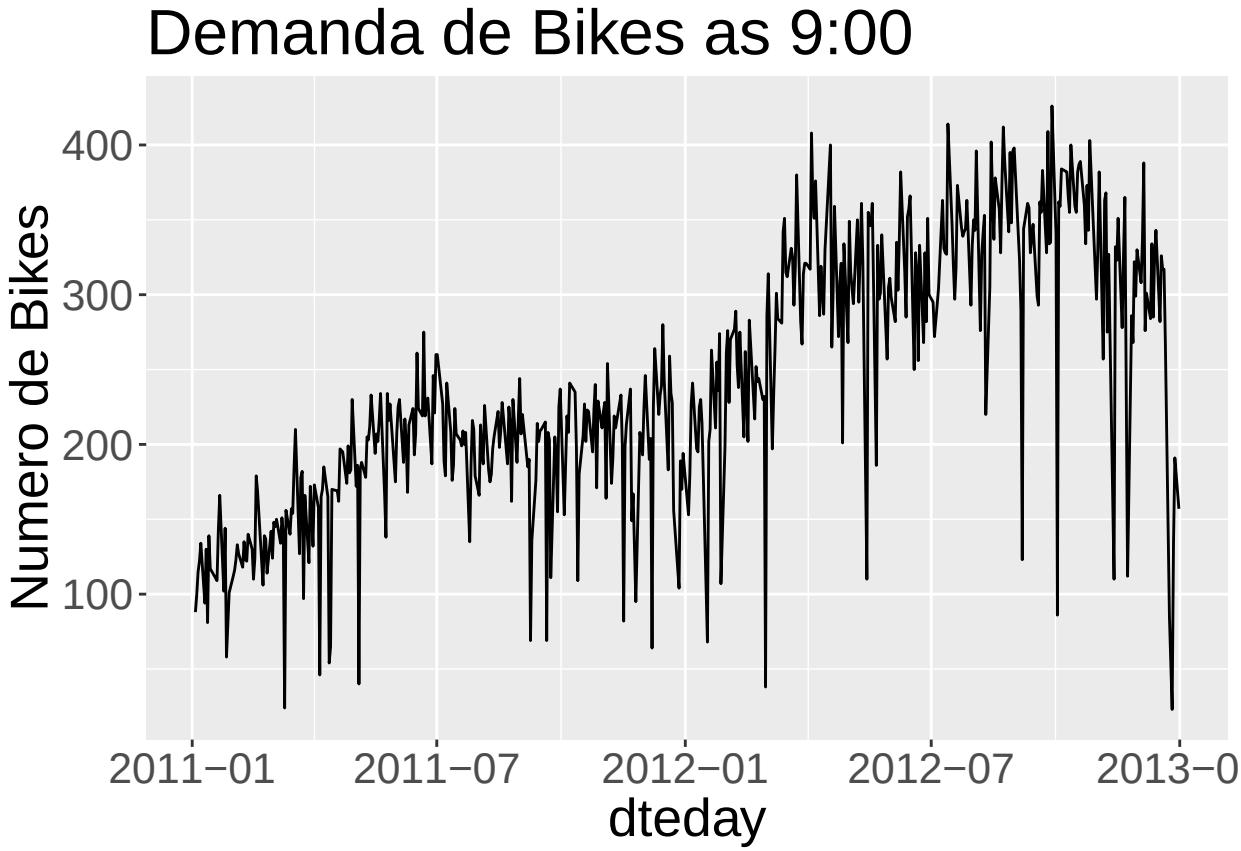
```
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
lapply(times, tms.plot)
```

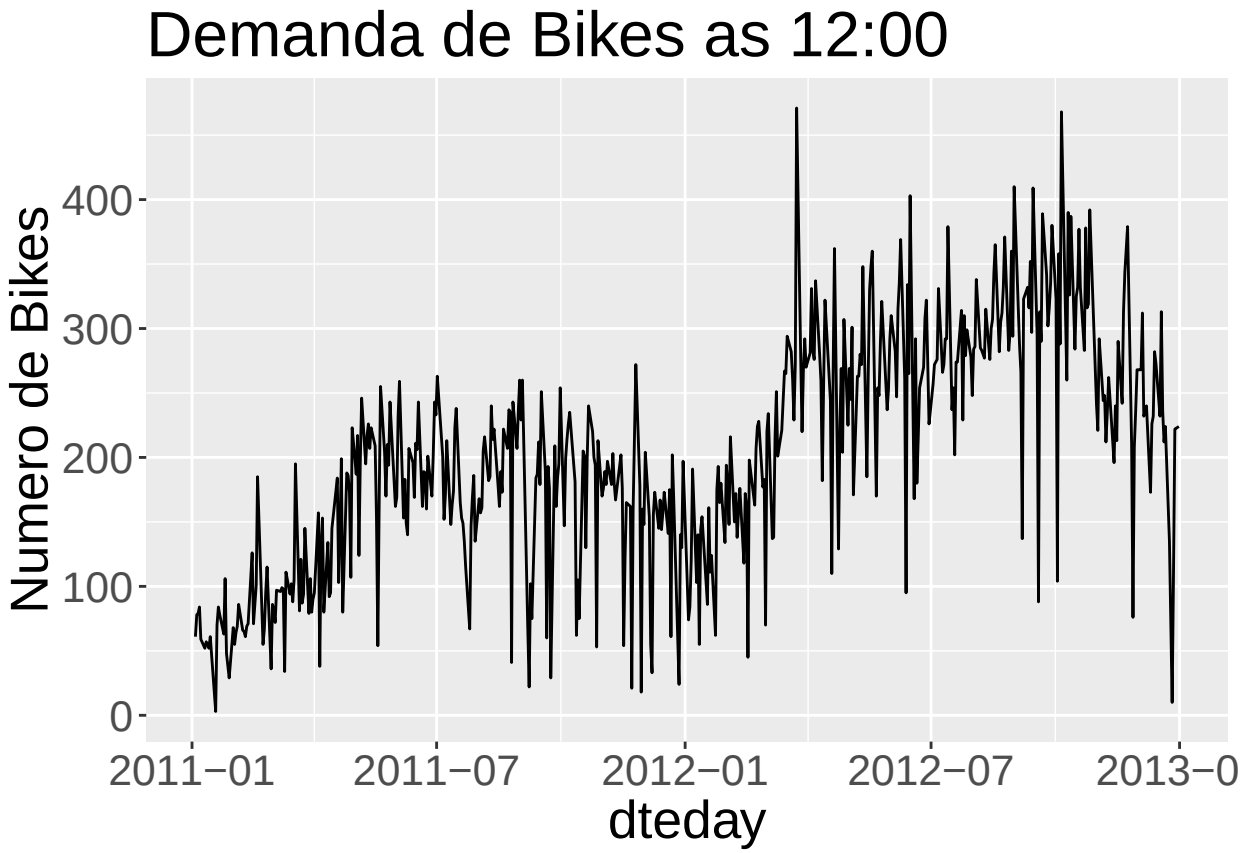
```
## [[1]]
```



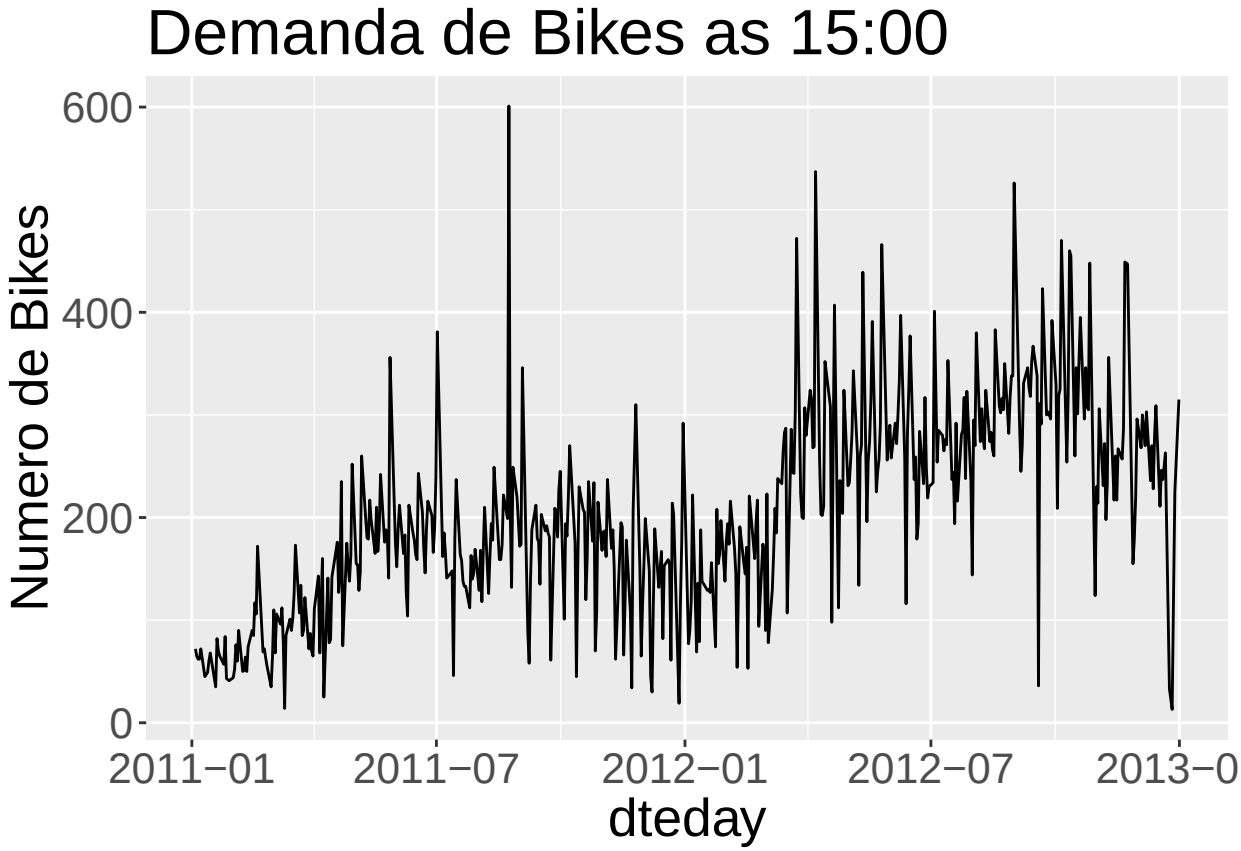

```
##
## [[2]]
```


Demanda de Bikes as 9:00
Numero de Bikes
400
300
200
100
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

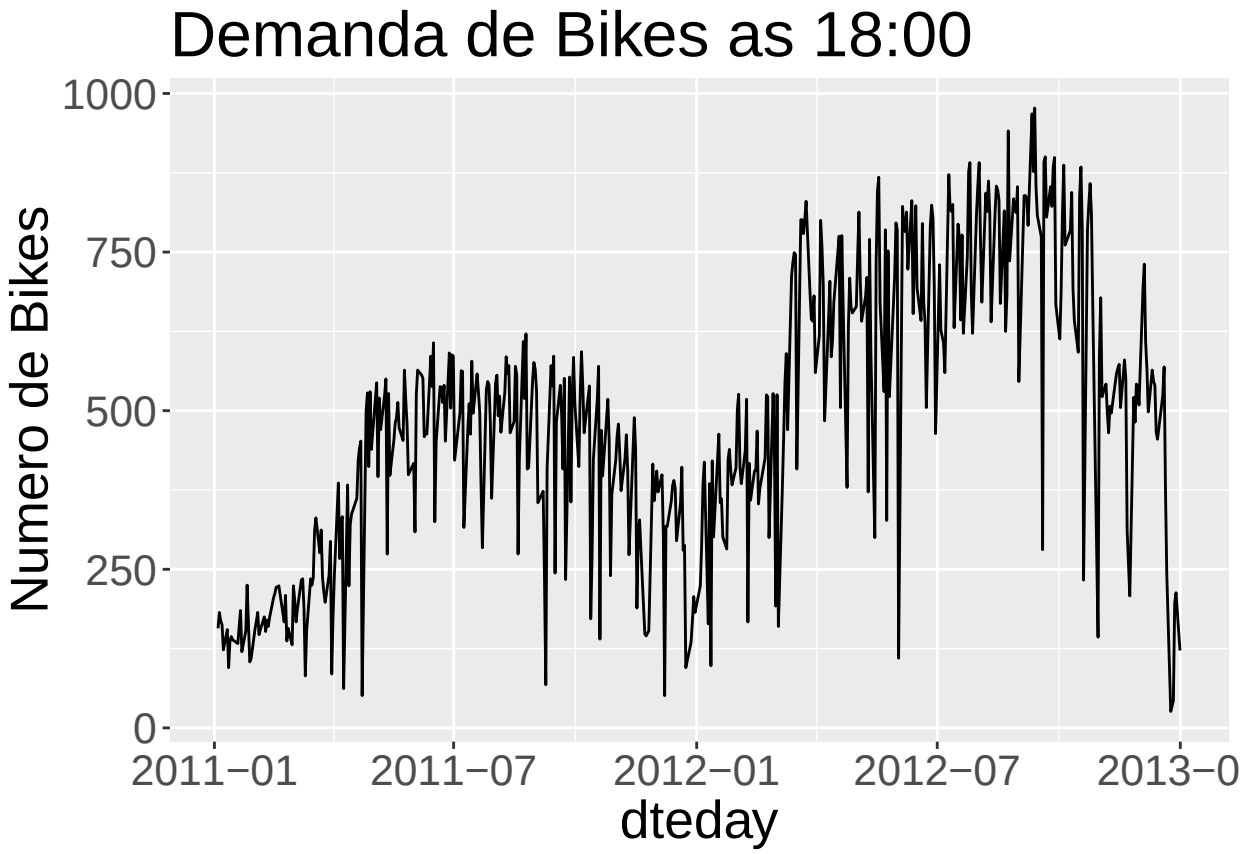
```
##
## [[3]]
```


Demanda de Bikes as 12:00
Numero de Bikes
400
300
200
100
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

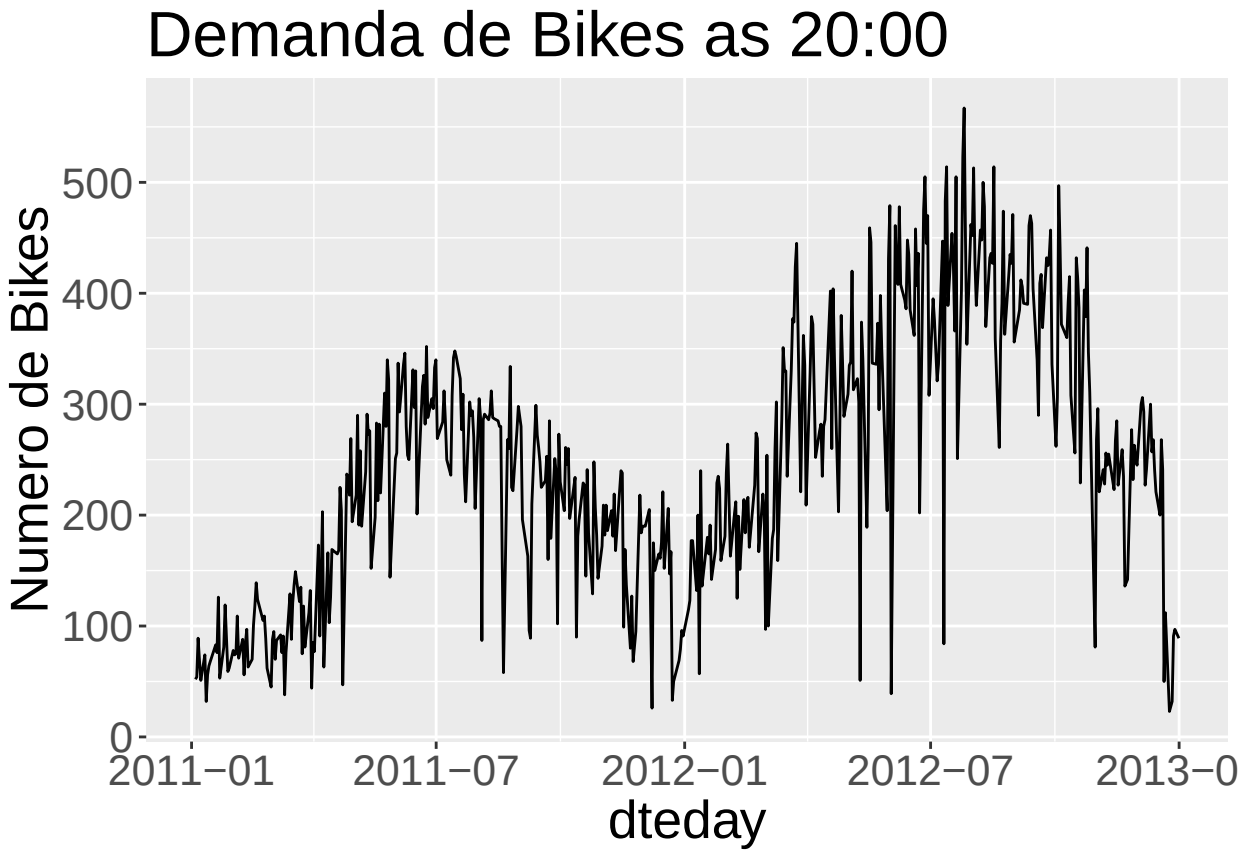
```
##
## [[4]]
```


Demanda de Bikes as 15:00
Numero de Bikes
600
400
200
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

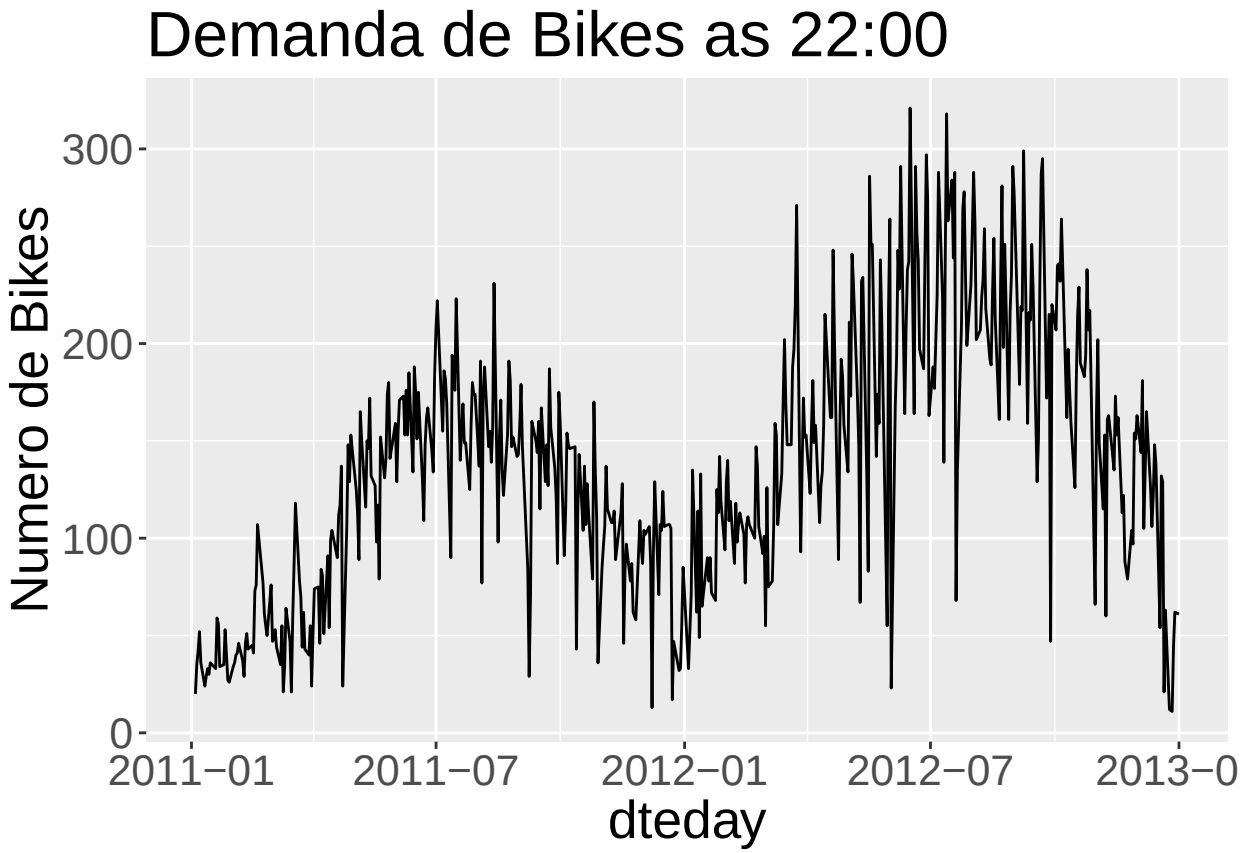
```
##
## [[5]]
```


Demanda de Bikes as 18:00
Numero de Bikes
1000
750
500
250
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

```
##
## [[6]]
```

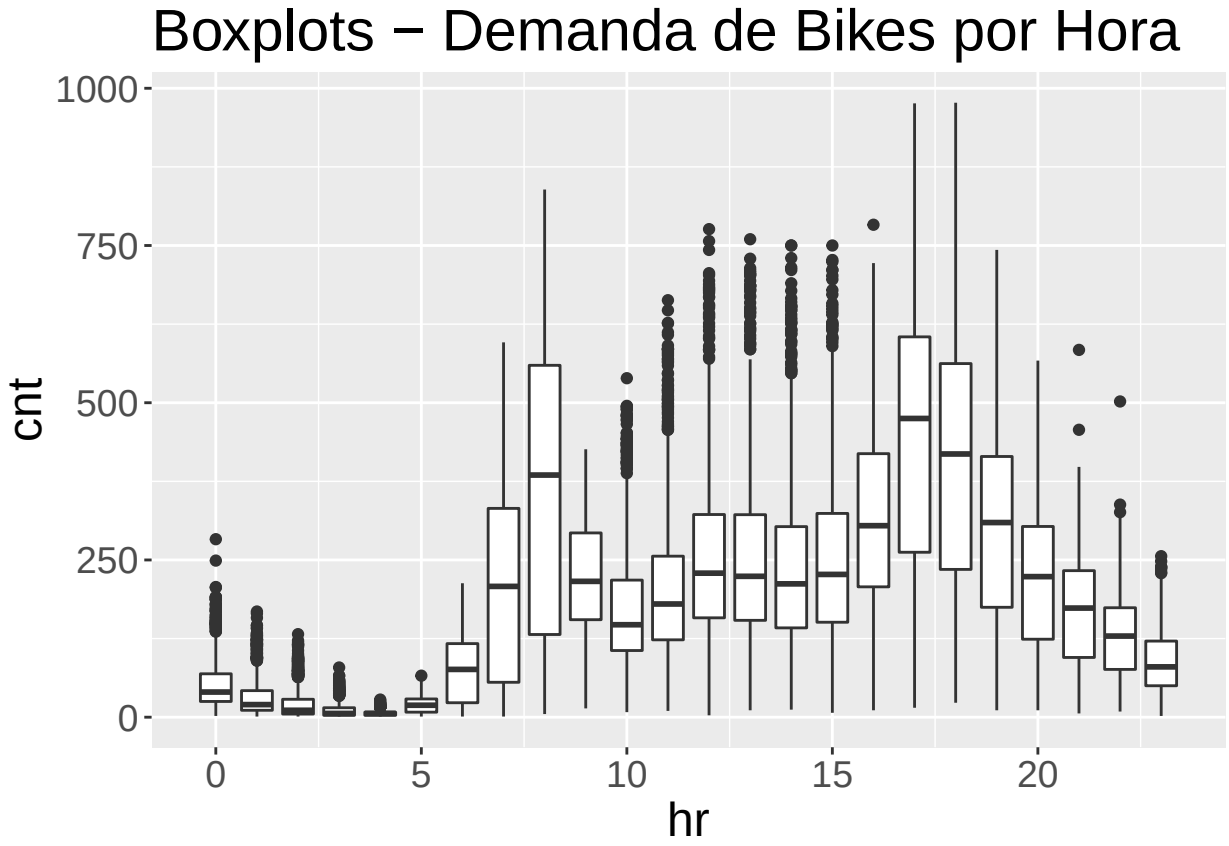

Demanda de Bikes as 20:00
Numero de Bikes
0
100
200
300
400
500
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

```
##
## [[7]]
```

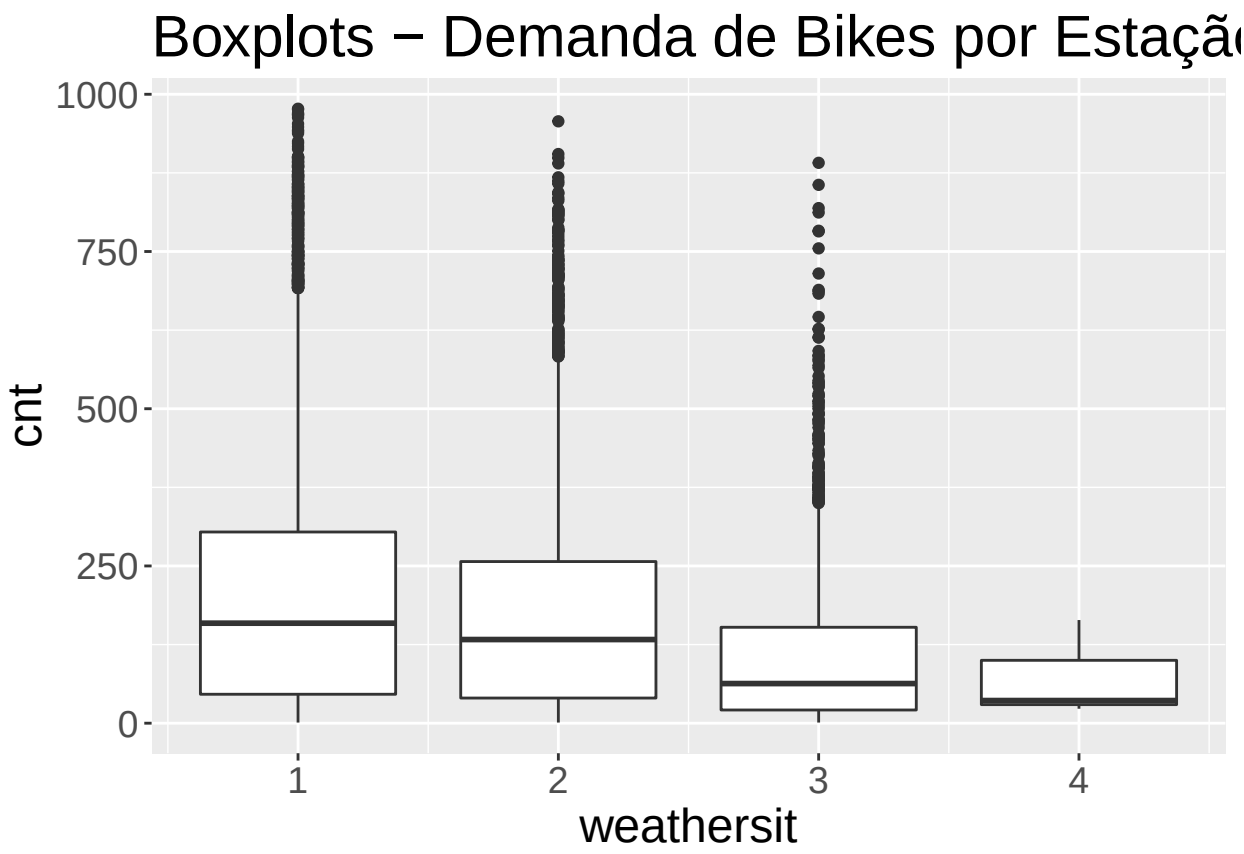



```
# Analisando BoxPlots

# Este código contém comandos para análise de variáveis usando BoxPlots
# Convertendo a variável dayWeek para fator ordenado e plotando em ordem de tempo
bikes$dayWeek <- fact.conv(bikes$dayWeek)

# Demanda de bikes x potenciais variáveis preditoras
labels <- list("Boxplots - Demanda de Bikes por Hora",
               "Boxplots - Demanda de Bikes por Estação",
               "Boxplots - Demanda de Bikes por Dia Útil",
               "Boxplots - Demanda de Bikes por Dia da Semana")

xAxis <- list("hr", "weathersit", "isWorking", "dayWeek")

# Função para criar os boxplots
plot.boxes  <- function(X, label){
  ggplot(bikes, aes_string(x = X, y = "cnt", group = X)) +
    geom_boxplot( ) +
    ggtitle(label) +
    theme(text = element_text(size = 18))
}

Map(plot.boxes, xAxis, labels)
```

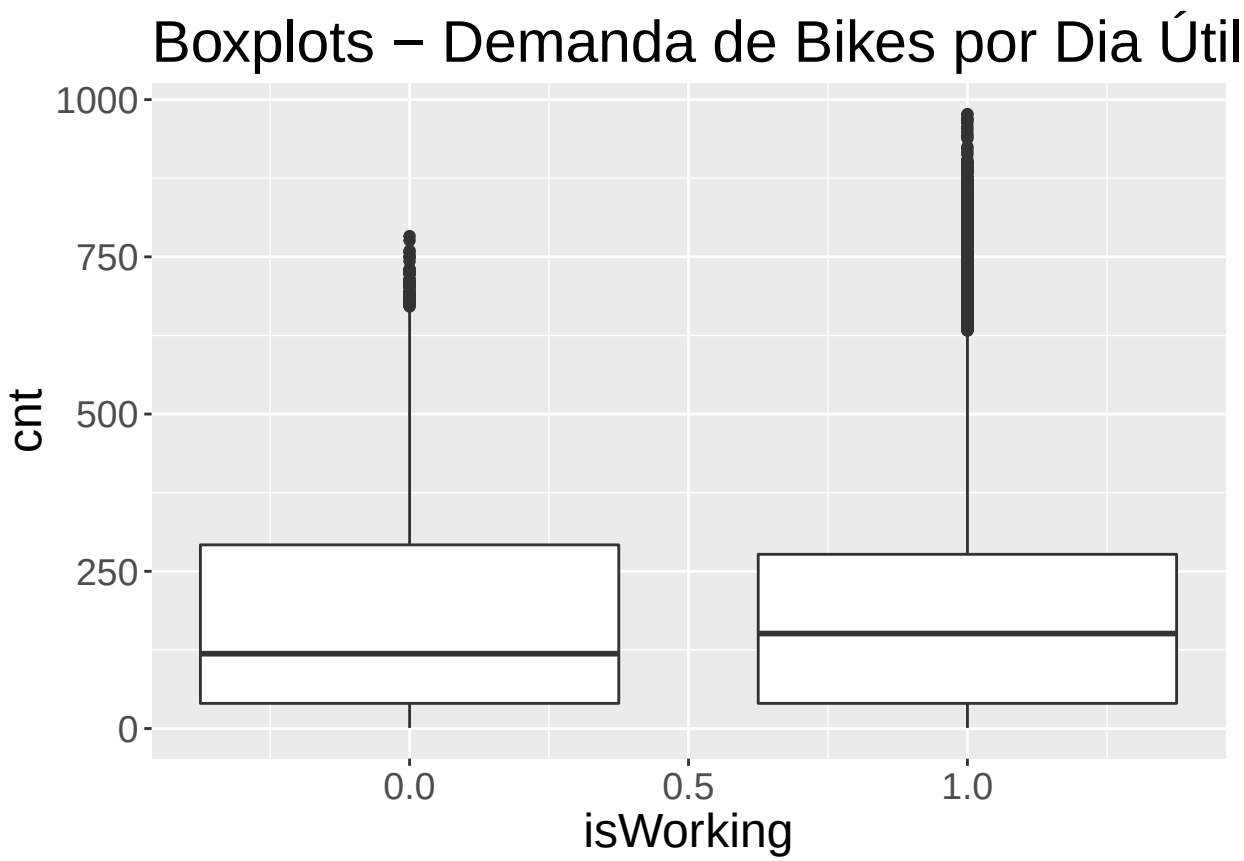
```
## [[1]]
```


Boxplots − Demanda de Bikes por Hora
1000
750
500
250
0
cnt
0
5
10
15
20
hr

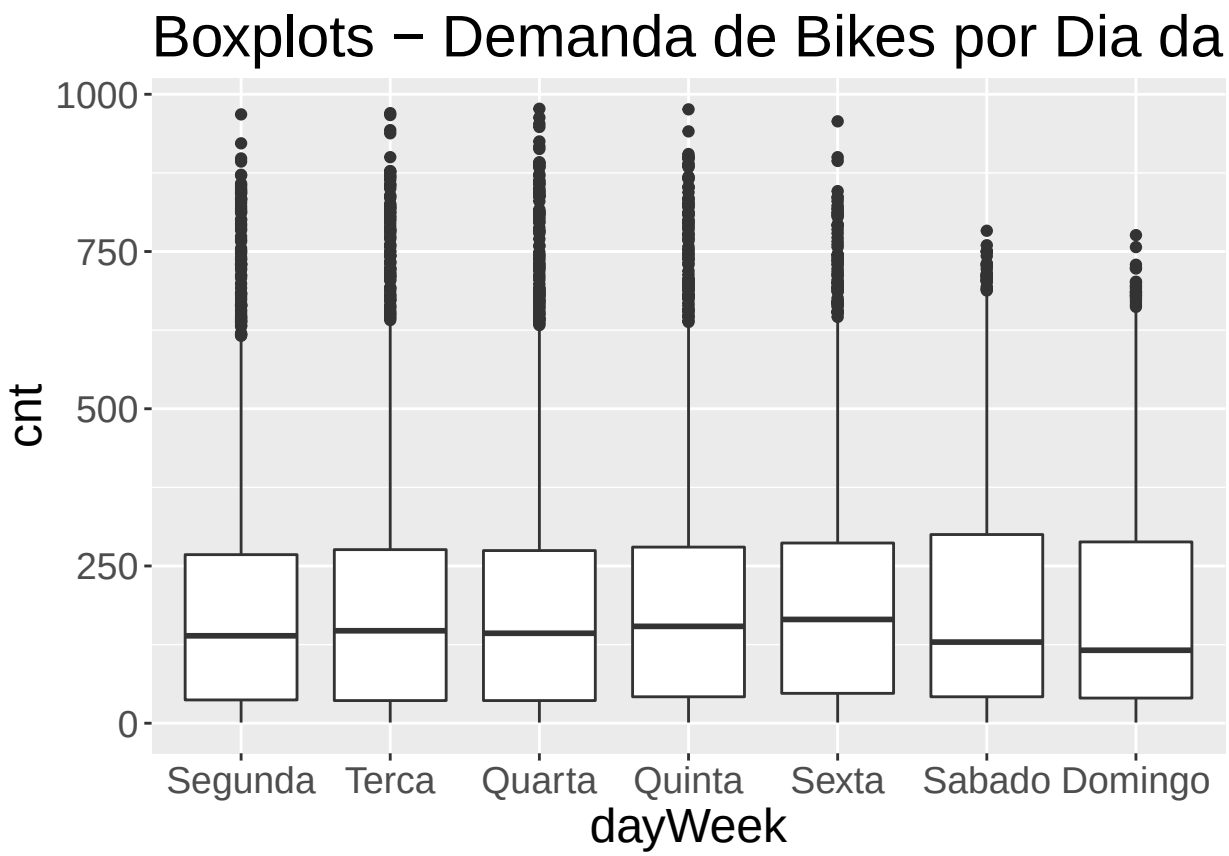
```
##
## [[2]]
```


Boxplots − Demanda de Bikes por Estaçã
1000
750
500
250
0
cnt
1
2
3
4
weathersit

```
##
## [[3]]
```

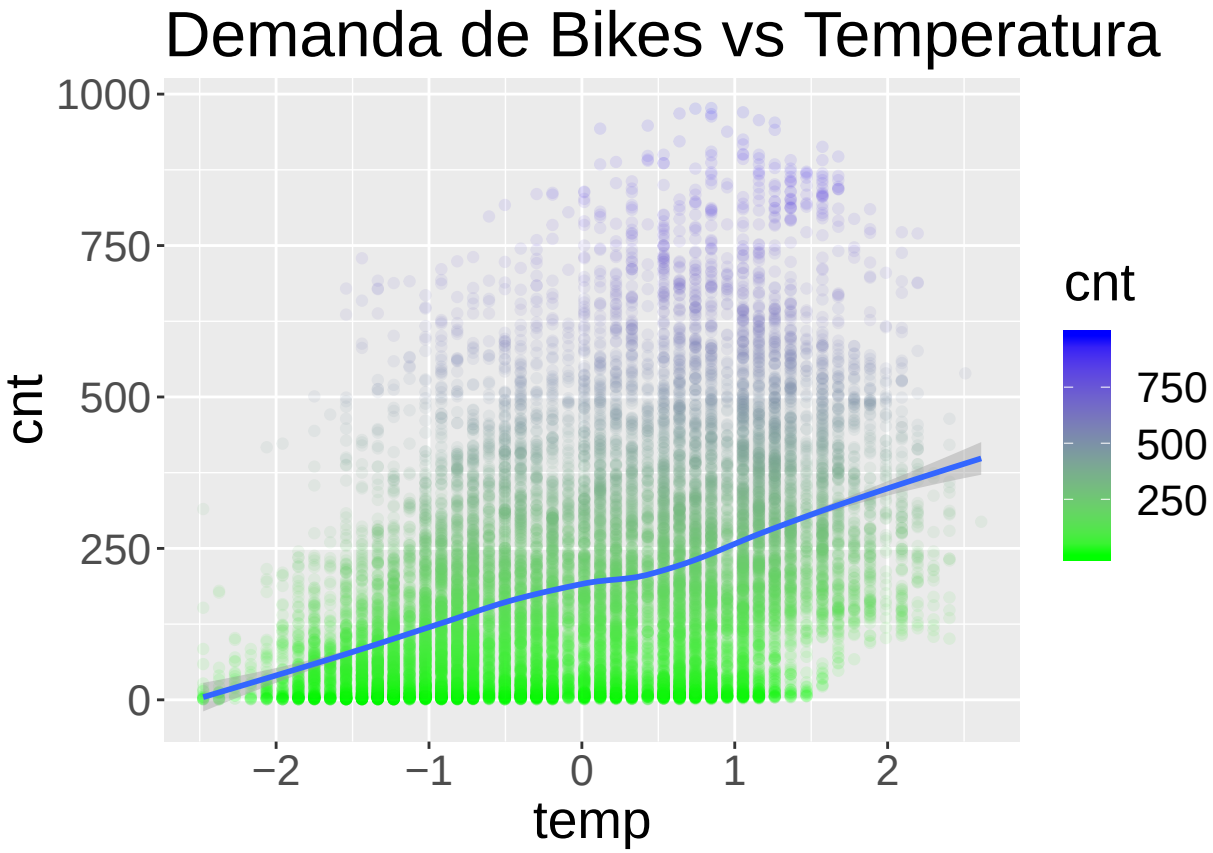

Boxplots − Demanda de Bikes por Dia Útil
1000
750
500
250
0
cnt
0.0
0.5
1.0
isWorking

```
##
## [[4]]
```

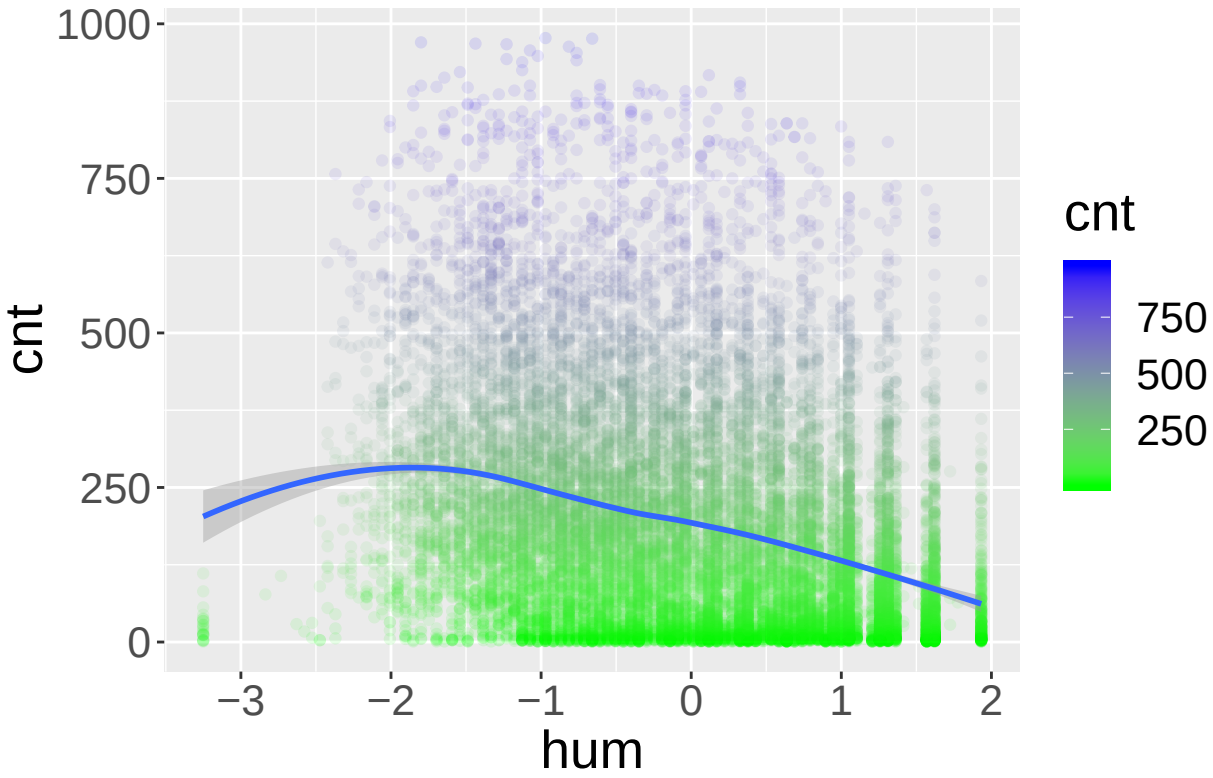



```
# Analisando Density Plots

# Este código contém comandos para análise de variáveis usando Density Plots
# Visualizando o relacionamento entre as variáveis preditoras e demanda por bike
labels <- c("Demanda de Bikes vs Temperatura",
            "Demanda de Bikes vs Humidade",
            "Demanda de Bikes vs Velocidade do Vento",
            "Demanda de Bikes vs Hora")

xAxis <- c("temp", "hum", "windspeed", "hr")

# Função para os Density Plots
plot.scatter <- function(X, label){
  ggplot(bikes, aes_string(x = X, y = "cnt")) +
    geom_point(aes_string(colour = "cnt"), alpha = 0.1) +
    scale_colour_gradient(low = "green", high = "blue") +
    geom_smooth(method = "loess") +
    ggtitle(label) +
    theme(text = element_text(size = 20))
}

Map(plot.scatter, xAxis, labels)
```

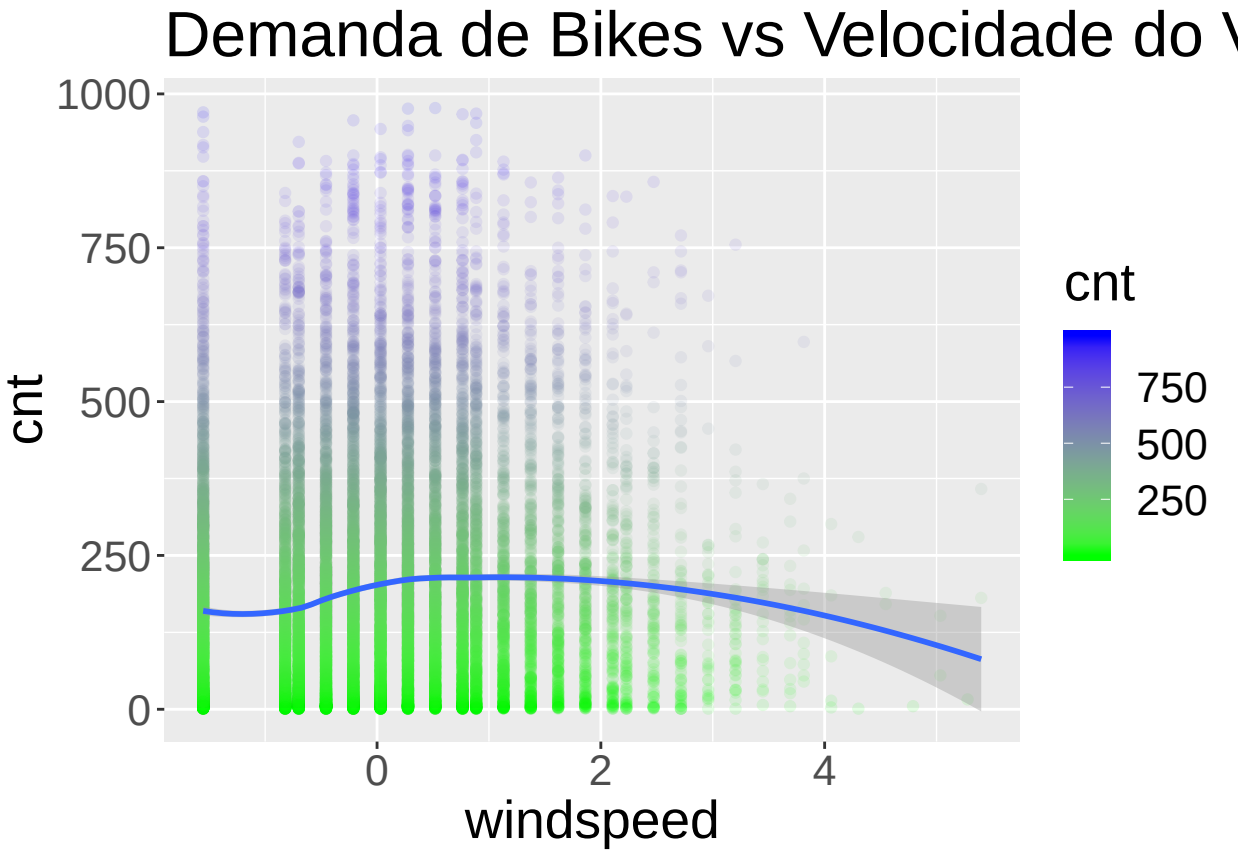
```
## $temp
```


Demanda de Bikes vs Temperatura
1000
750
500
250
0
cnt
−2
−1
0
1
2
temp
cnt
750
500
250

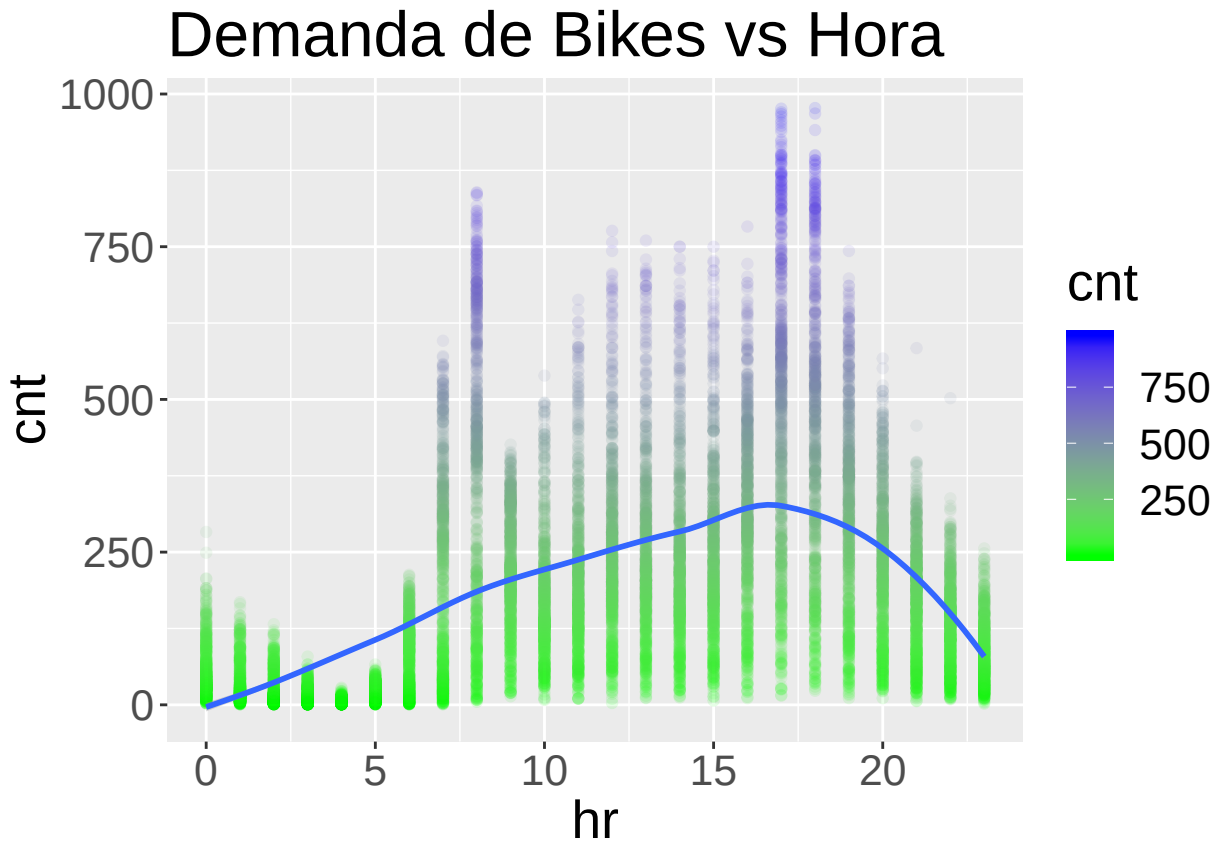
```
##
## $hum
```


Demanda de Bikes vs Humidade
1000
750
500
250
0
cnt
−3
−2
−1
0
1
2
hum
cnt
750
500
250

```
##
## $windspeed
```

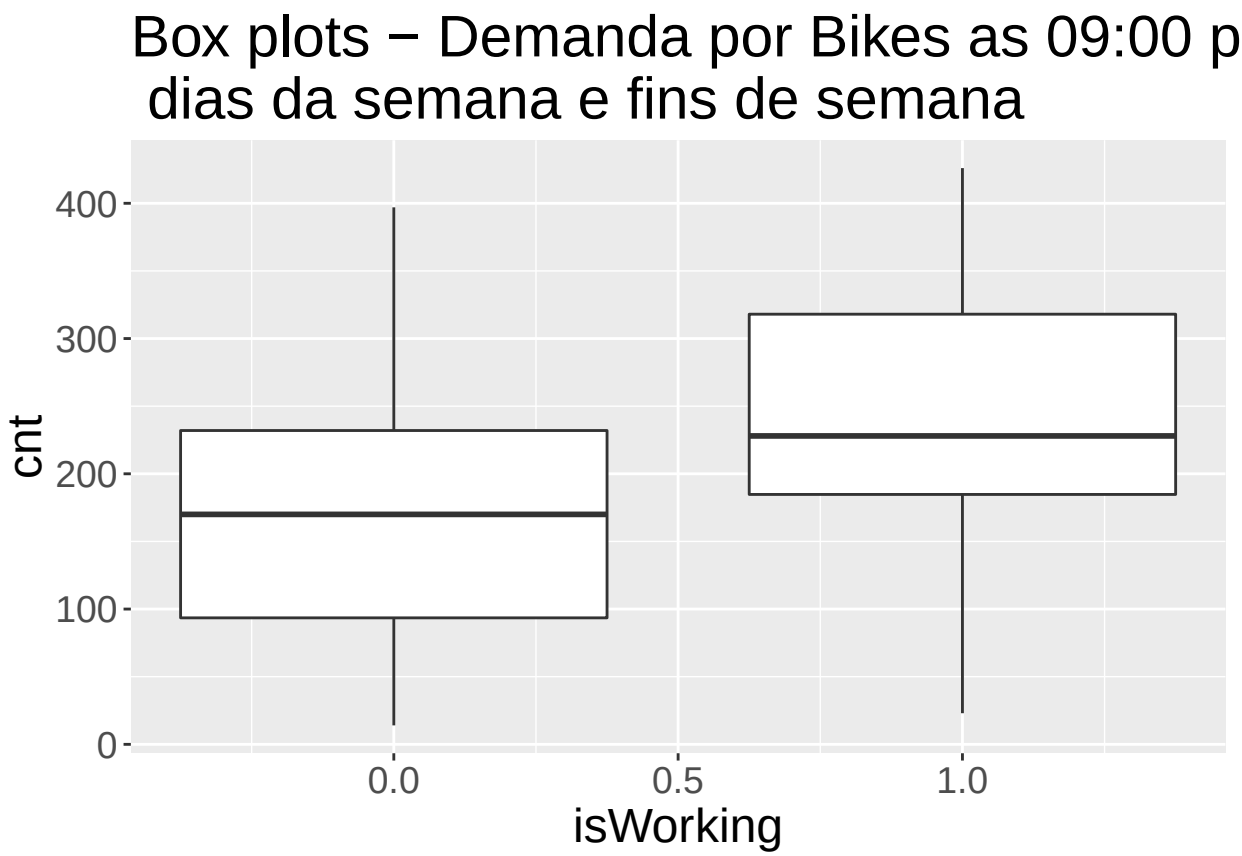

Demanda de Bikes vs Velocidade do V
1000
750
500
250
0
cnt
0
2
4
windspeed
cnt
750
500
250

```
##
## $hr
```

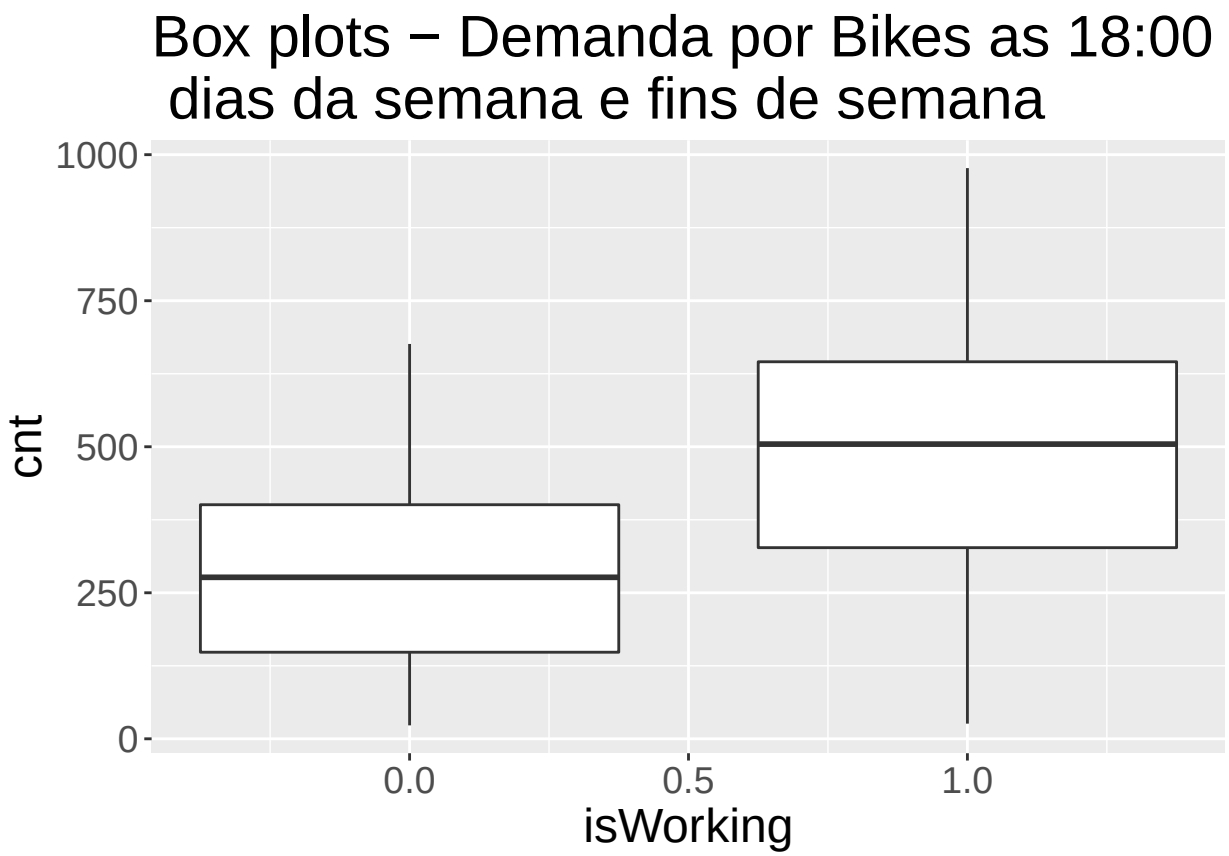



```
# Explorando a interação entre tempo e dia, em dias da semana e fins de semana
labels <- list("Box plots - Demanda por Bikes as 09:00 para \n dias da semana e fins de semana",
               "Box plots - Demanda por Bikes as 18:00  para \n dias da semana e fins de semana")

Times <- list(9, 18)

plot.box2 <- function(time, label){
  ggplot(bikes[bikes$hr == time, ], aes(x = isWorking, y = cnt, group = isWorking)) +
    geom_boxplot( ) + ggtitle(label) +
    theme(text = element_text(size = 18)) }

Map(plot.box2, Times, labels)
```

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


Box plots − Demanda por Bikes as 09:00 p
dias da semana e fins de semana
400
300
200
100
0
cnt
0.0
0.5
1.0
isWorking

```
##
## [[2]]
```




```
# Feature Selection

# Este código contém comandos para feature selection
dim(bikes)
```

```
## [1] 17377    16
```

```
any(is.na(bikes))
```

```
## [1] FALSE
```

```
# Criando um modelo para identificar os atributos com maior importância para o modelo preditivo
require(randomForest)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

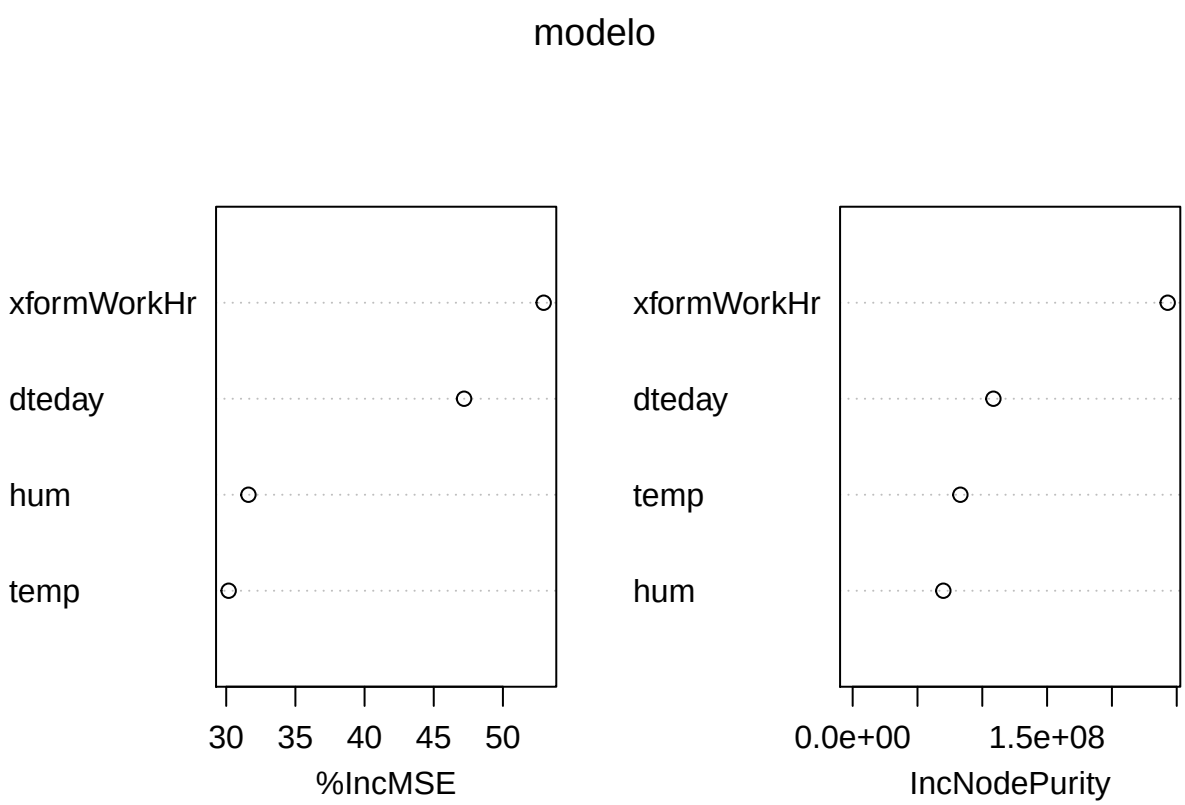
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Avalidando a importância de todas as variaveis
modelo <- randomForest(cnt ~ . ,
                       data = bikes,
                       ntree = 100,
```

```
                        nodesize = 10,
                        importance = TRUE)

# Removendo variáveis colineares
modelo <- randomForest(cnt ~ . - mnth
                        - hr
                        - workingday
                        - isWorking
                        - dayWeek
                        - xformHr
                        - workTime
                        - holiday
                        - windspeed
                        - monthCount
                        - weathersit,
                        data = bikes,
                        ntree = 100,
                        nodesize = 10,
                        importance = TRUE)

# Plotando as variáveis por grau de importância
varImpPlot(modelo)
```




```
# Gravando o resultado
df_saida <- bikes[, c("cnt", rownames(modelo$importance))]

# Cria um modelo preditivo usando randomForest
```

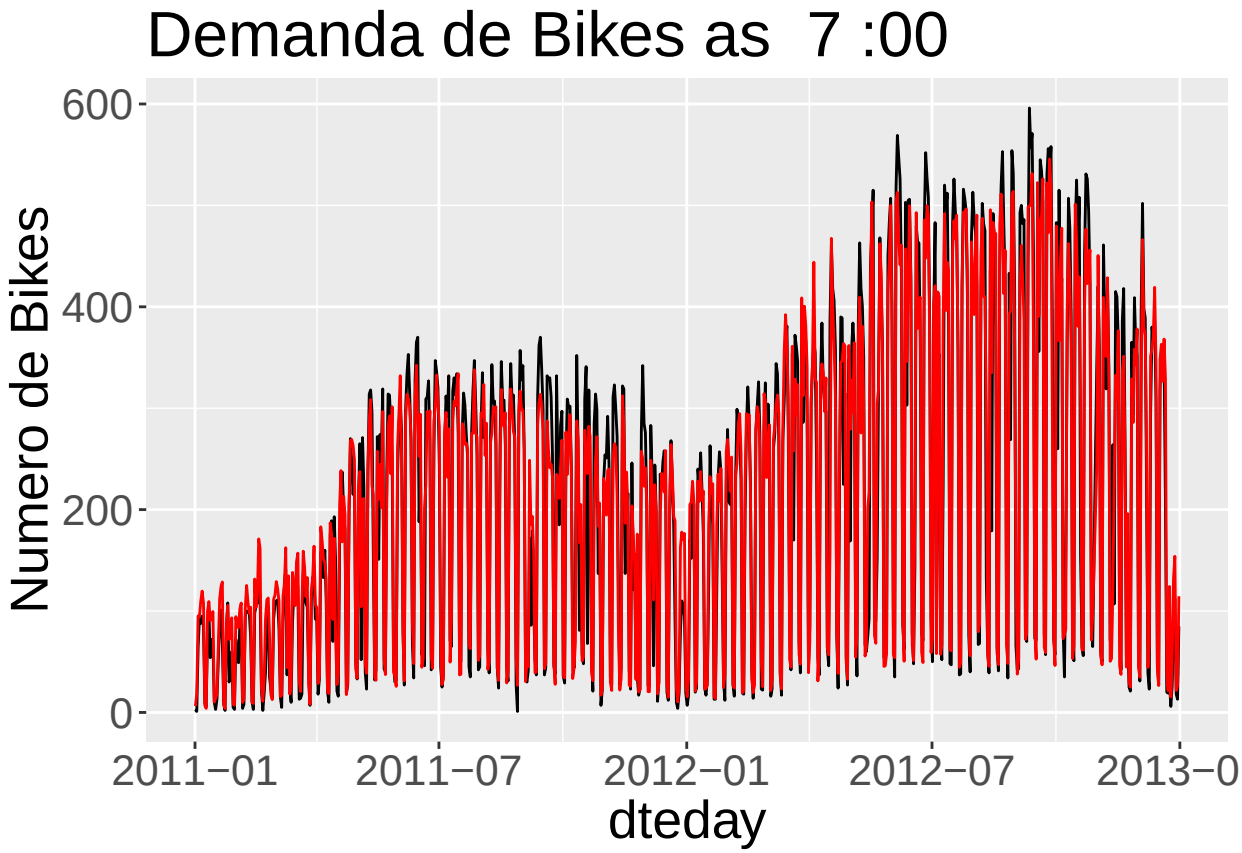
```r
# Função para tratar as datas
set.asPOSIXct <- function(inFrame) {
  dteday <- as.POSIXct(
    as.integer(inFrame$dteday),
    origin = "1970-01-01")

  as.POSIXct(strptime(
    paste(as.character(dteday),
          " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S"))
}

char.toPOSIXct <-   function(inFrame) {
  as.POSIXct(strptime(
    paste(inFrame$dteday, " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S")) }

require(randomForest)
model <- randomForest(cnt ~ xformWorkHr + dteday + temp + hum,
                      data = bikes, # altere o nome do objeto data para "dataset" de estiver trabalhand
                      ntree = 40,
                      nodesize = 5)
print(model)
```

```
##
## Call:
##  randomForest(formula = cnt ~ xformWorkHr + dteday + temp + hum,      data = bikes, ntree = 40, node
##                Type of random forest: regression
##                      Number of trees: 40
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 4018.184
##                     % Var explained: 87.79
```

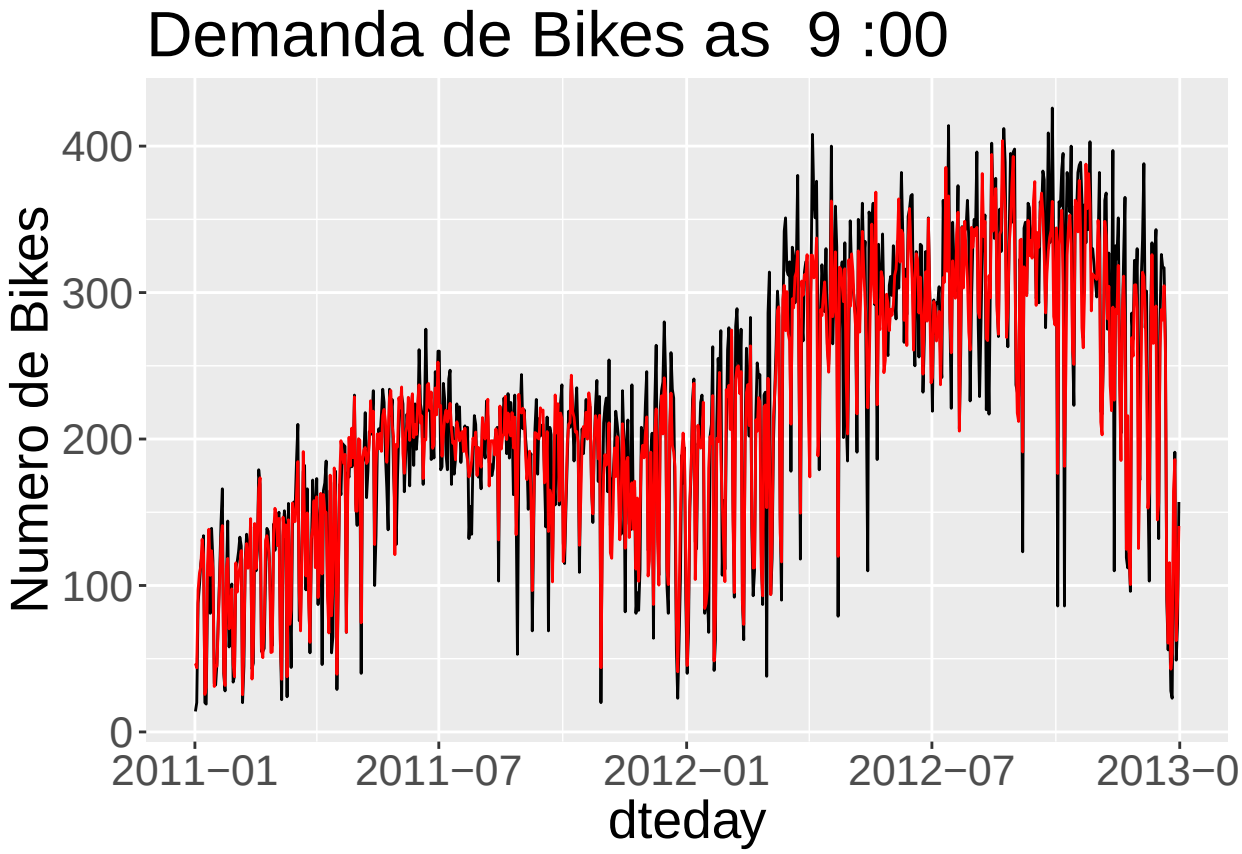
```r
# Score do modelo preditivo com randomForest

# Função para tratar as datas
set.asPOSIXct <- function(inFrame) {
  dteday <- as.POSIXct(
    as.integer(inFrame$dteday),
    origin = "1970-01-01")

  as.POSIXct(strptime(
    paste(as.character(dteday),
          " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
```

```
    "%Y-%m-%d %H:%M:%S"))
}

char.toPOSIXct <-   function(inFrame) {
  as.POSIXct(strptime(
    paste(inFrame$dteday, " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S")) }


require(randomForest)
scores <- data.frame(actual = bikes$cnt,
                     prediction = predict(model, newdata = bikes))


# Avaliação do modelo
source("src/Tools.R")
inFrame <- scores[, c("actual", "prediction")]
refFrame <- bikes

# Criando um dataframe
inFrame[, c("dteday", "monthCount", "hr", "xformWorkHr")] <- refFrame[, c("dteday", "monthCount", "hr",

# Nomeando o dataframe
names(inFrame) <- c("cnt", "predicted", "dteday", "monthCount", "hr", "xformWorkHr")

#  Time series plot mostrando a diferença entre valores reais e valores previstos
library(ggplot2)
inFrame <- inFrame[order(inFrame$dteday),]
s <- c(7, 9, 12, 15, 18, 20, 22)

lapply(s, function(s){
  ggplot() +
    geom_line(data = inFrame[inFrame$hr == s, ],
              aes(x = dteday, y = cnt)) +
    geom_line(data = inFrame[inFrame$hr == s, ],
              aes(x = dteday, y = predicted), color = "red") +
    ylab("Numero de Bikes") +
    labs(title = paste("Demanda de Bikes as ",
                       as.character(s), ":00", spe ="")) +
    theme(text = element_text(size = 20))
})
```

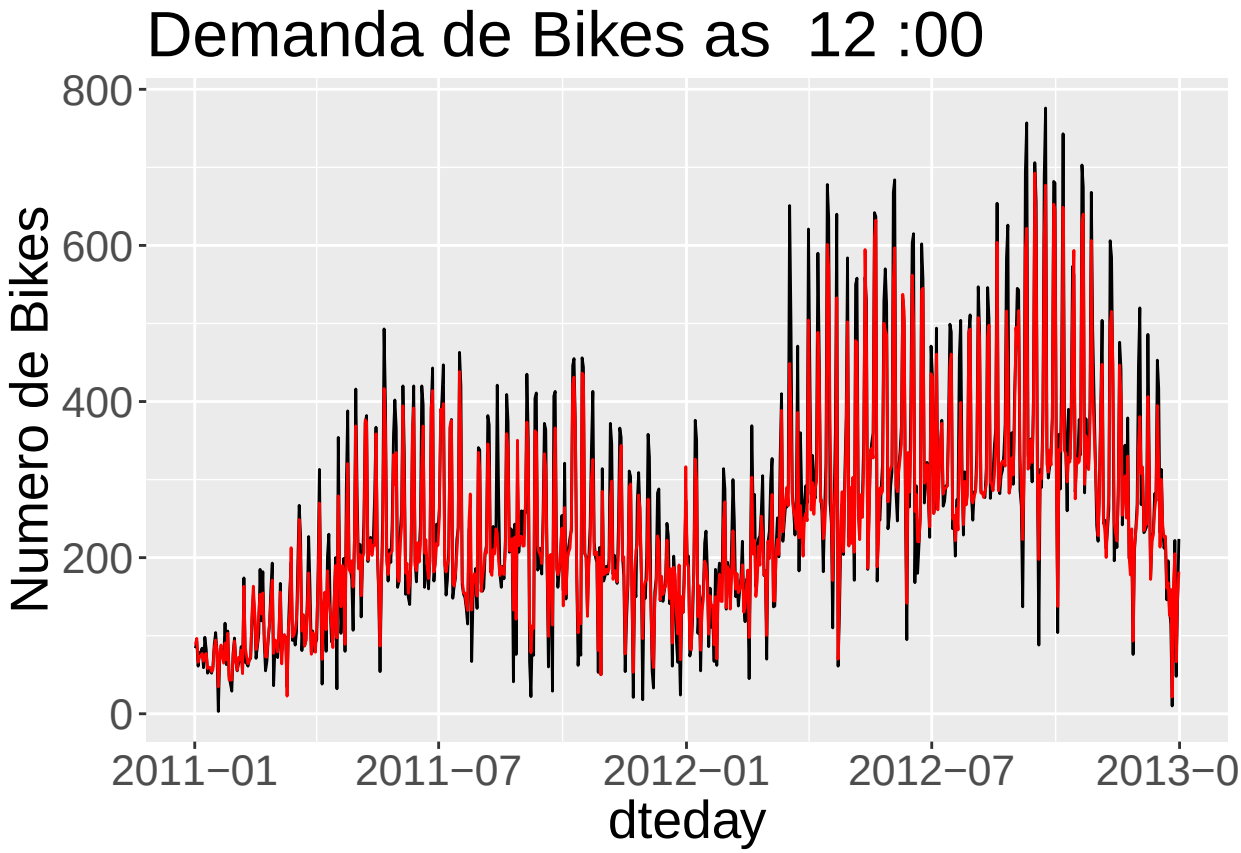
```
## [[1]]
```


Demanda de Bikes as 7 :00
Numero de Bikes
600
400
200
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

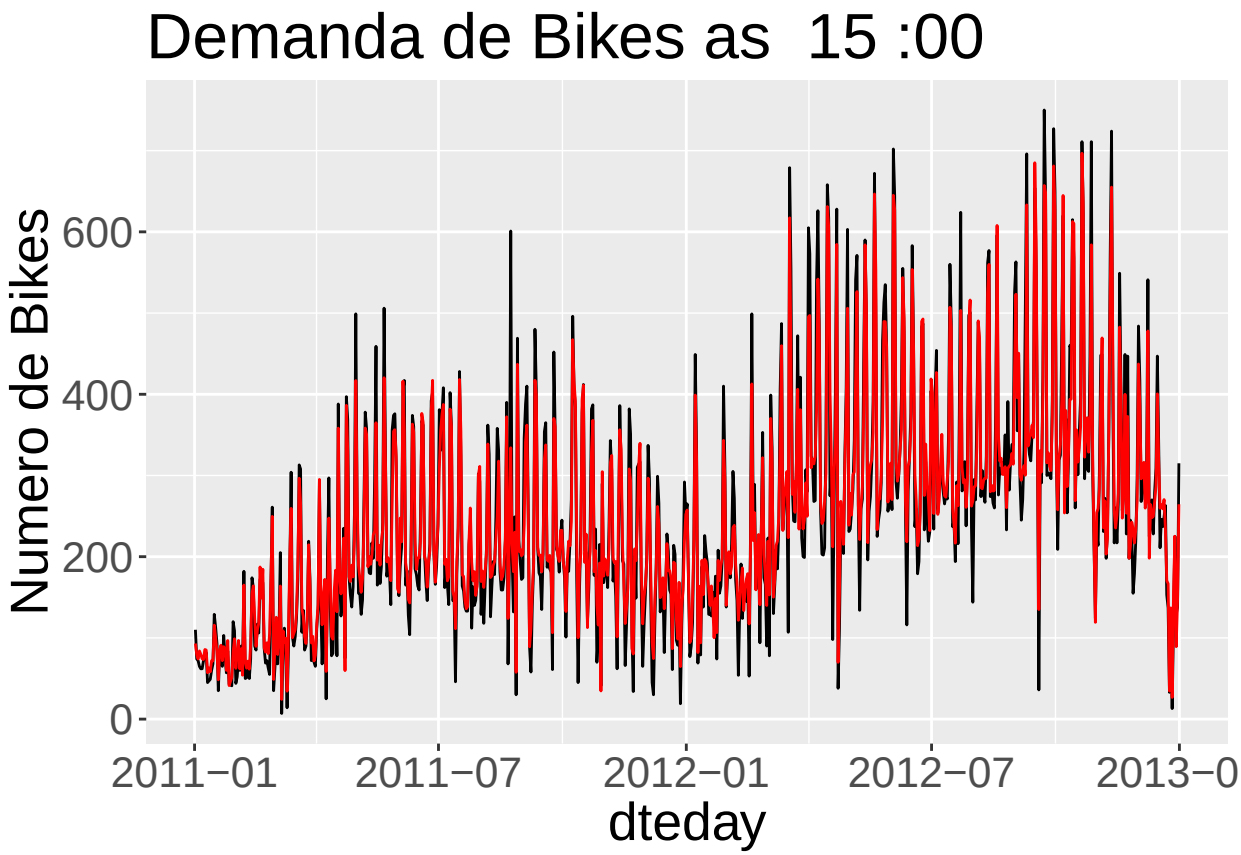
```
##
## [[2]]
```


Demanda de Bikes as 9 :00
Numero de Bikes
400
300
200
100
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

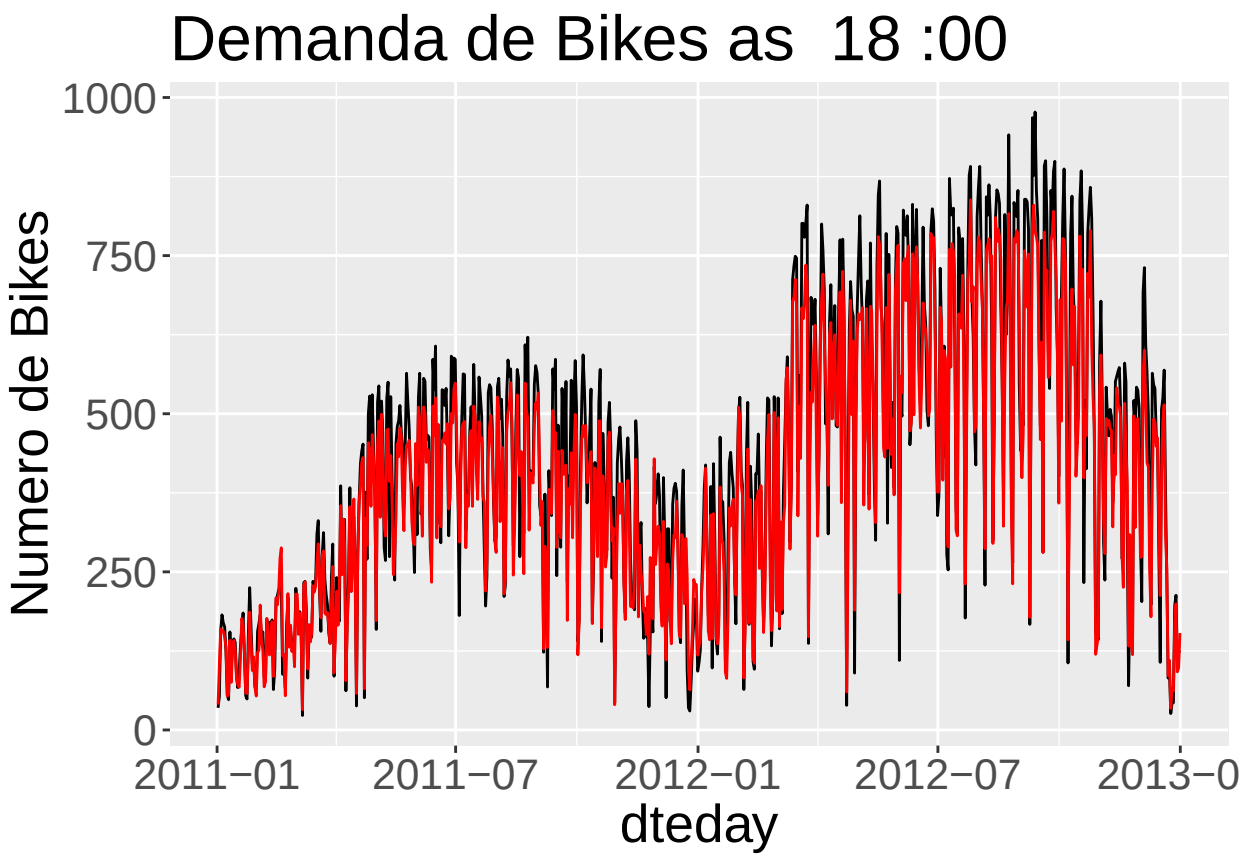
```
##
## [[3]]
```


Demanda de Bikes as 12 :00
Numero de Bikes
800
600
400
200
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

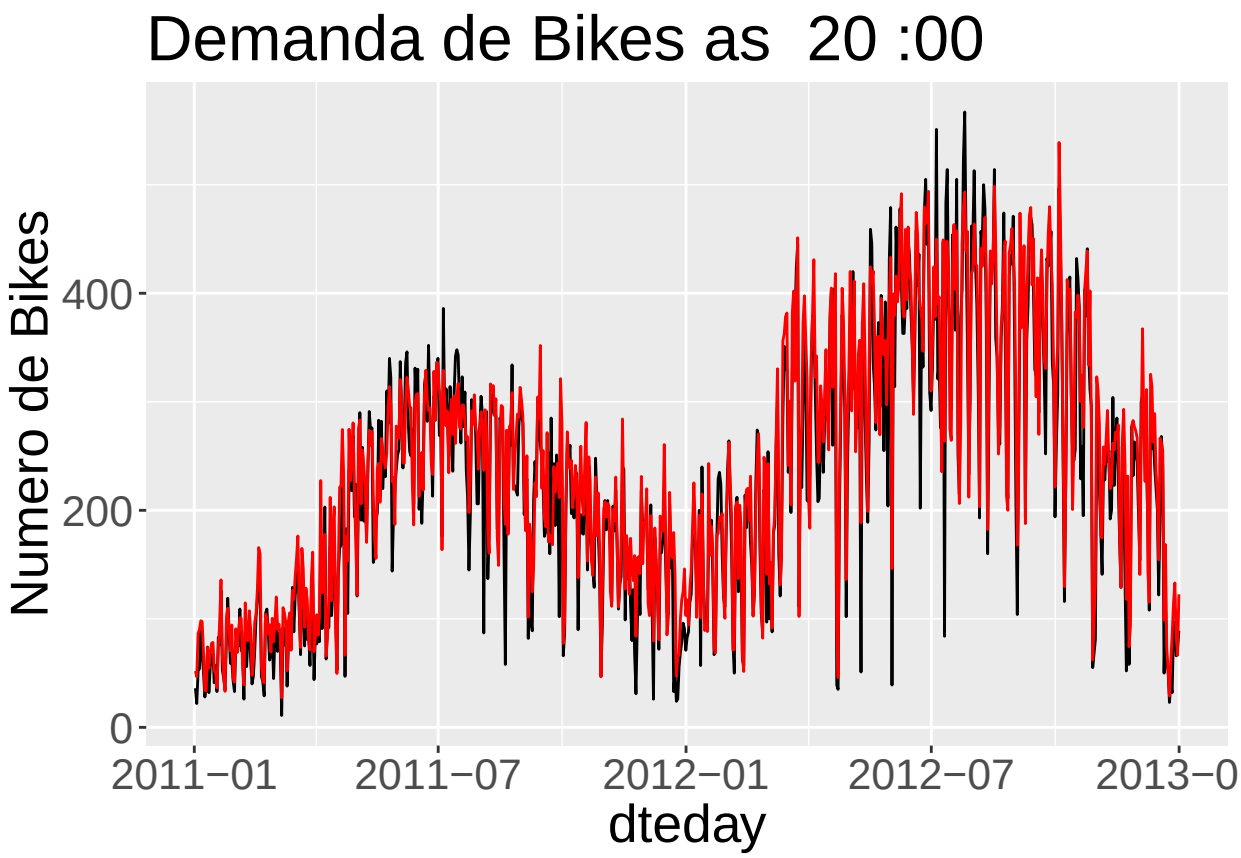
```
##
## [[4]]
```


Demanda de Bikes as 15 :00
Numero de Bikes
600
400
200
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

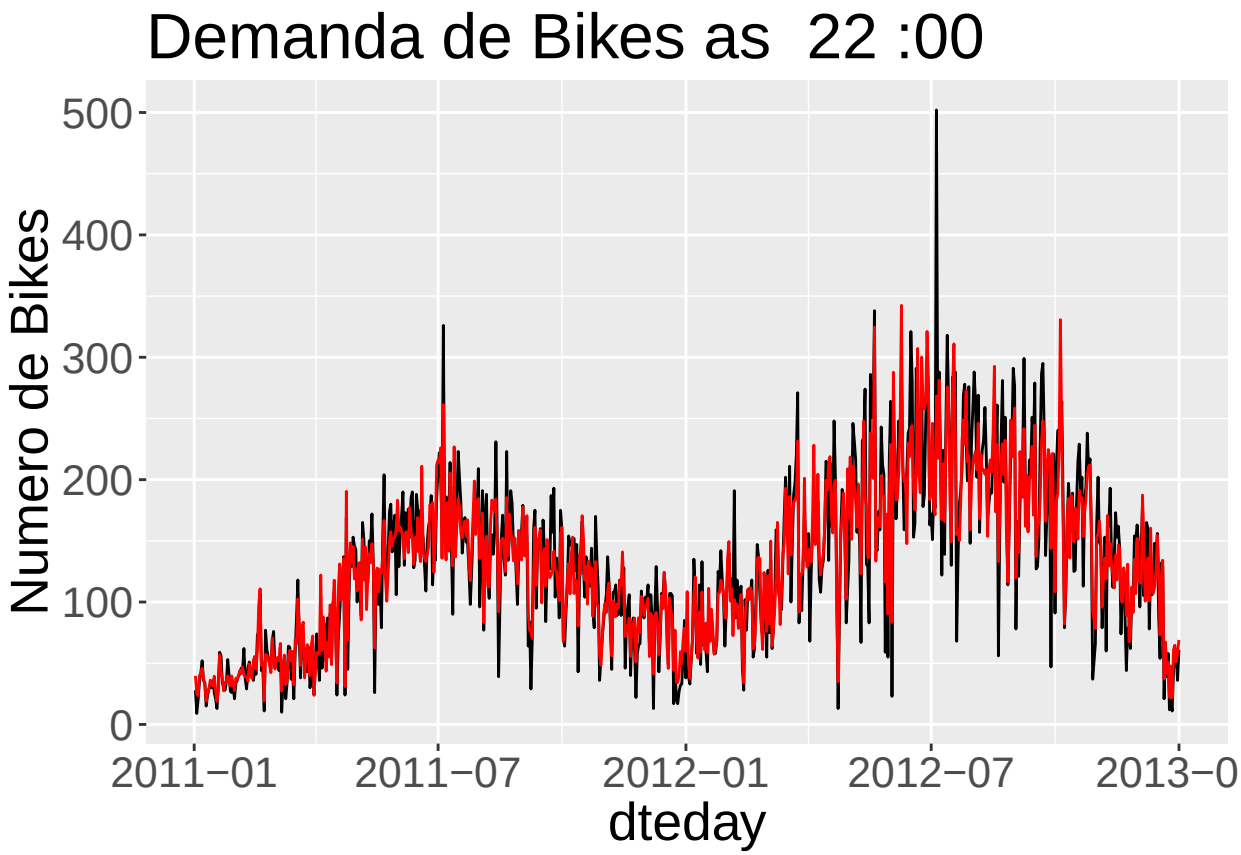
```
##
## [[5]]
```


Demanda de Bikes as 18 :00
Numero de Bikes
1000
750
500
250
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

```
##
## [[6]]
```


Demanda de Bikes as 20 :00
Numero de Bikes
400
200
0
2011−01
2011−07
2012−01
2012−07
2013−0
dteday

```
##
## [[7]]
```




```
# Computando os resíduos
library(dplyr)
```

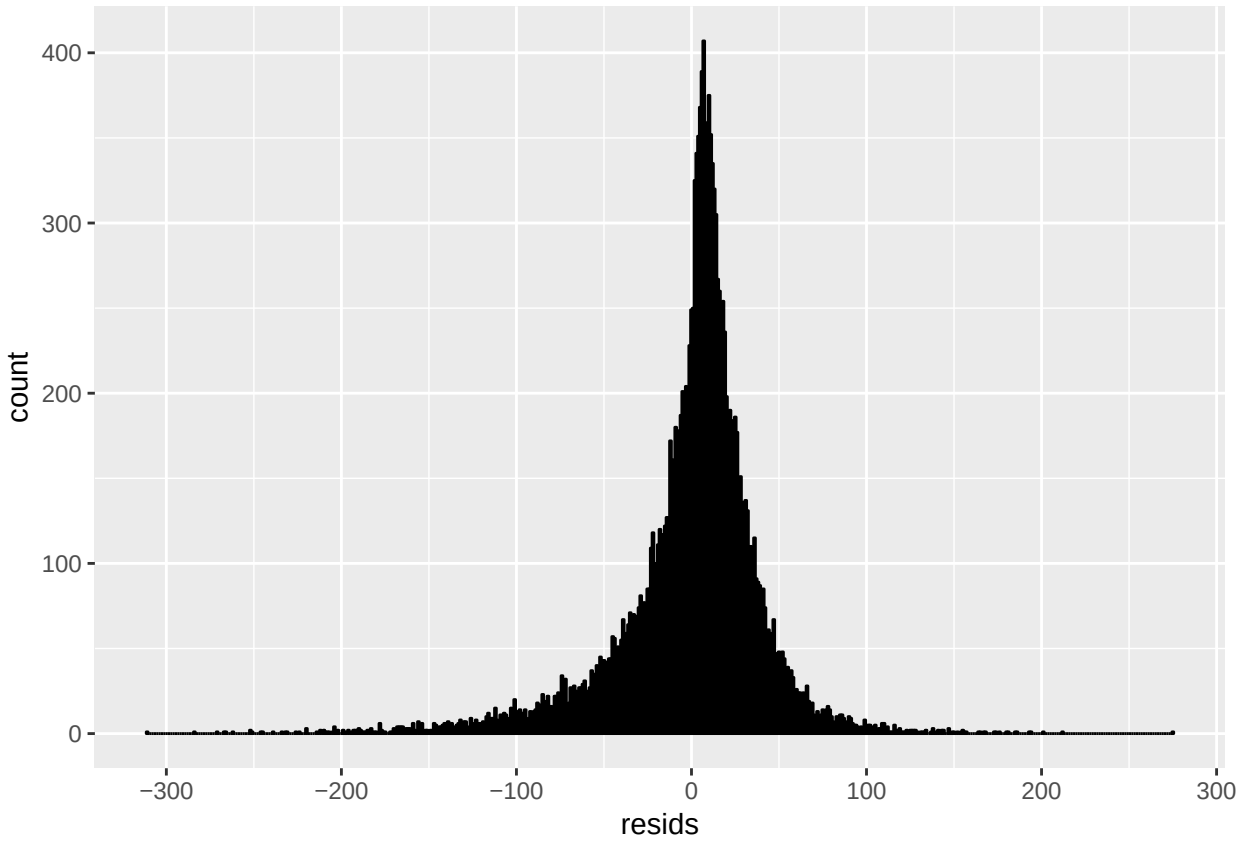
```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

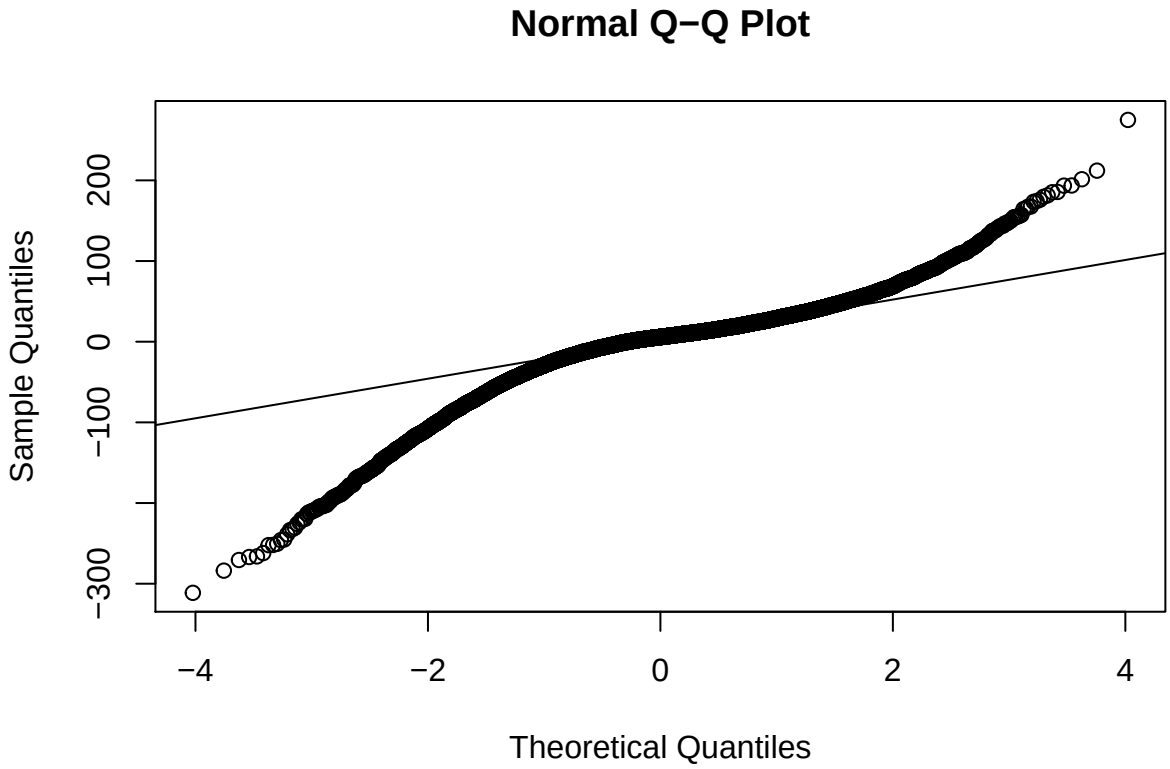
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
inFrame <-  mutate(inFrame, resids = predicted - cnt)

# Plotando os resíduos
ggplot(inFrame, aes(x = resids)) +
  geom_histogram(binwidth = 1, fill = "white", color = "black")
```

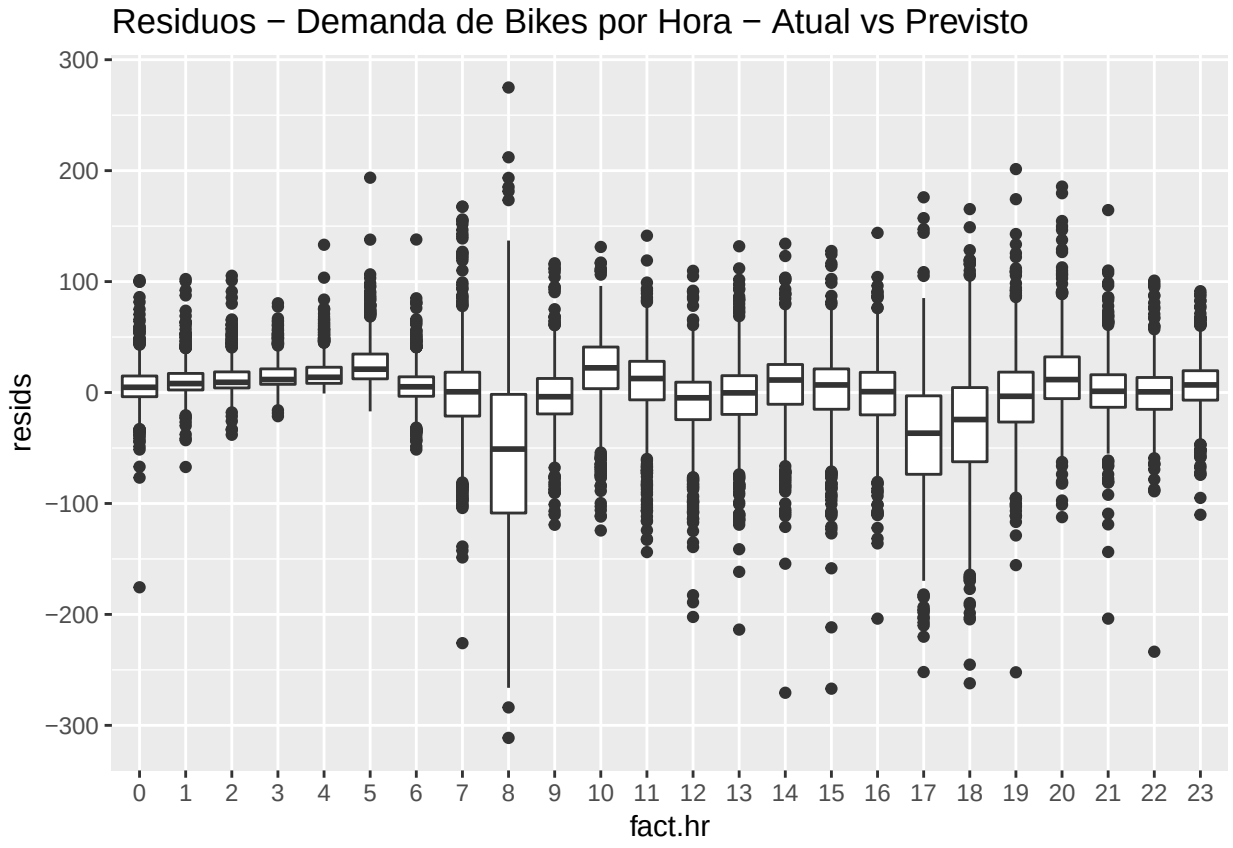



```
qqnorm(inFrame$resids)
qqline(inFrame$resids)
```

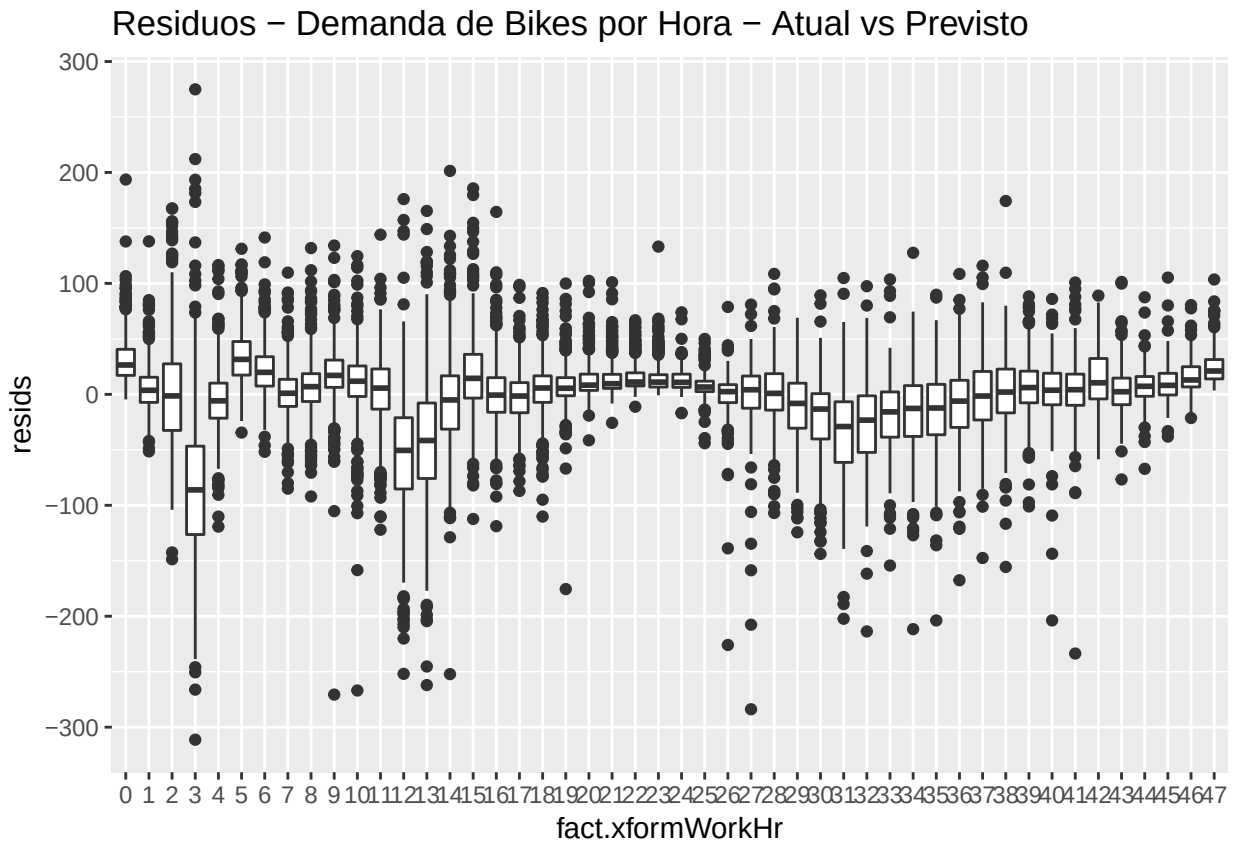


```
# Plotando os resíduos com as horas transformadas
inFrame <- mutate(inFrame, fact.hr = as.factor(hr),
                  fact.xformWorkHr = as.factor(xformWorkHr))
facts <- c("fact.hr", "fact.xformWorkHr")
lapply(facts, function(x){
  ggplot(inFrame, aes_string(x = x, y = "resids")) +
    geom_boxplot( ) +
    ggtitle("Residuos - Demanda de Bikes por Hora - Atual vs Previsto")})
```

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




```
##
## [[2]]
```




```
# Mediana dos resíduos por hora
evalFrame <- inFrame %>%
  group_by(hr) %>%
  summarise(medResidByHr = format(round(
    median(predicted - cnt), 2),
    nsmall = 2))

# Computando a mediana dos resíduos
tempFrame <- inFrame %>%
  group_by(monthCount) %>%
  summarise(medResid = median(predicted - cnt))

evalFrame$monthCount <- tempFrame$monthCount
evalFrame$medResidByMcnt <- format(round(
  tempFrame$medResid, 2),
  nsmall = 2)

print("Resumo dos residuos")
```

```
## [1] "Resumo dos residuos"
```

```
print(evalFrame)
```

```
## # A tibble: 24 x 4
##       hr medResidByHr monthCount medResidByMcnt
##    <int> <chr>              <dbl> <chr>
##  1     0 4.72                   1 4.11
##  2     1 8.10                   2 5.01
```

```
##  3     2 9.22                 3 6.52
##  4     3 11.71                4 9.08
##  5     4 13.75                5 4.01
##  6     5 21.04                6 5.61
##  7     6 5.19                 7 8.46
##  8     7 0.67                 8 8.25
##  9     8 -50.97               9 3.84
## 10     9 -3.79               10 5.04
## # ... with 14 more rows
```