norwayCars

Venda de carros da Noruega

```
library(dplyr)
```

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

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

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

```
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

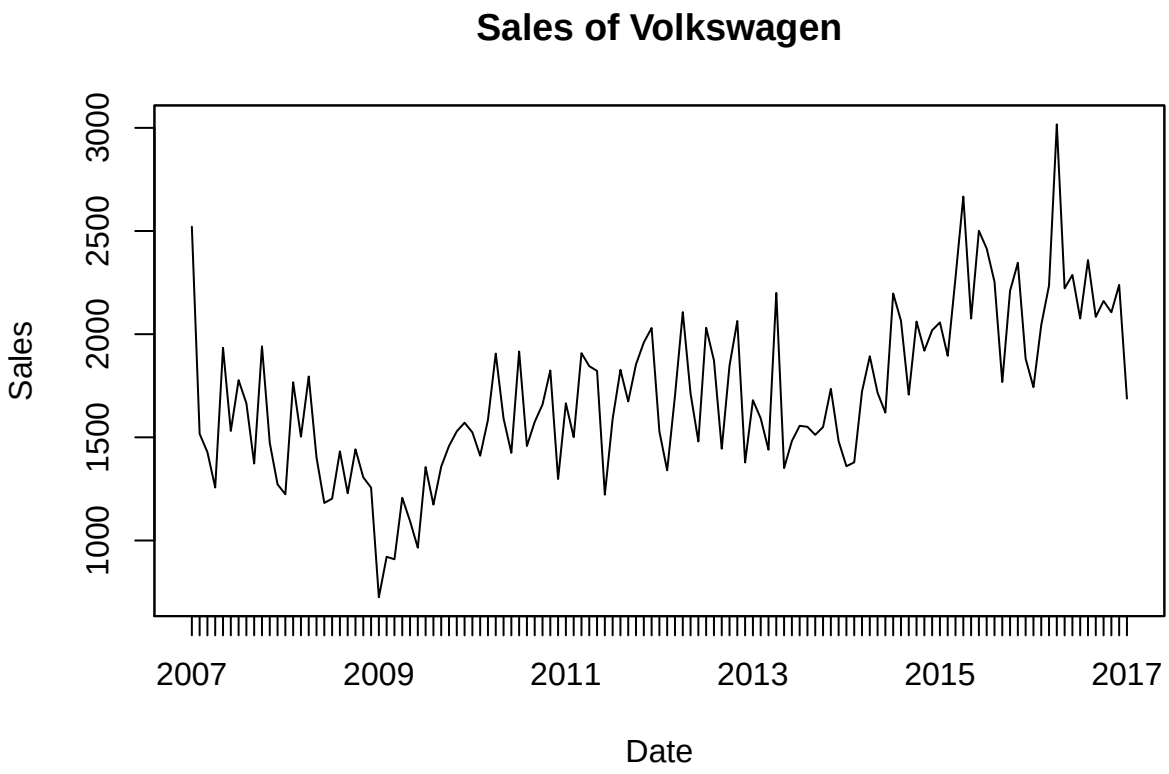
```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(ggplot2)
```

Os dados serão os dados de venda de carros da marca Volkswagen do período de 2007 a 2017.

```
data <- read.csv("norway_new_car_sales_by_make.csv")
car_brand <- "Volkswagen"
data <- data %>% dplyr::filter(data$Make == car_brand)

carSales <- zooreg(data$Quantity, frequency =  12, start = c(2007, 1))
plot(carSales, xlab = "Date", ylab = "Sales", main = "Sales of Volkswagen")
```



Nosso objetivo é obter um modelo para dado a observação de 2 anos prever o mês seguinte, para conseguirmos avaliar o erro do modelo com os dados observados, iremos percorrer uma janela de 2 anos ao longo do período e calcularemos o erro com a previsão do mês seguinte.

```
trainSales <- vector("list", 96)
testSales <- NA
startDate <- as.yearmon("2007-01-01")
endDate <- as.yearmon("2008-12-01")
finalDate <- as.yearmon("2017-01-01")
i <- 1
while(endDate < finalDate){
  trainSales[[i]] <- window(carSales, start = startDate, end=endDate)
  startDate <- startDate + 1/12
  endDate <- endDate + 1/12
  testSales[i] <- window(carSales, start = endDate, end = endDate)
  i <- i +1
}

ModelMAPE <- NA
```

Tendência polinomial

Uma etapa inicial, considerando uma formulação simples, é modelar a série como $Z_t = T_t + a_t$ sendo T_t um polimônio em função do tempo. Iremos considerar três situações distintas, uma função linear, uma função com termo quadrático e uma função com termo cúbico.

```r
#polynomial tendency
t <- rep(1:length(trainSales[[1]]))
predictions1 <- NA
predictions2 <- NA
predictions3 <- NA
for(n in seq_along(trainSales)){
  fit1 <- lm(trainSales[[n]]~t)
  fit2 <- lm(trainSales[[n]]~poly(t,2))
  fit3 <- lm(trainSales[[n]]~poly(t,3))
  predictions1[n] <- predict(fit1, newdata = data_frame(t = 25))[1]
  predictions2[n] <- predict(fit2, newdata = data_frame(t = 25))[1]
  predictions3[n] <- predict(fit3, newdata = data_frame(t = 25))[1]
}
```

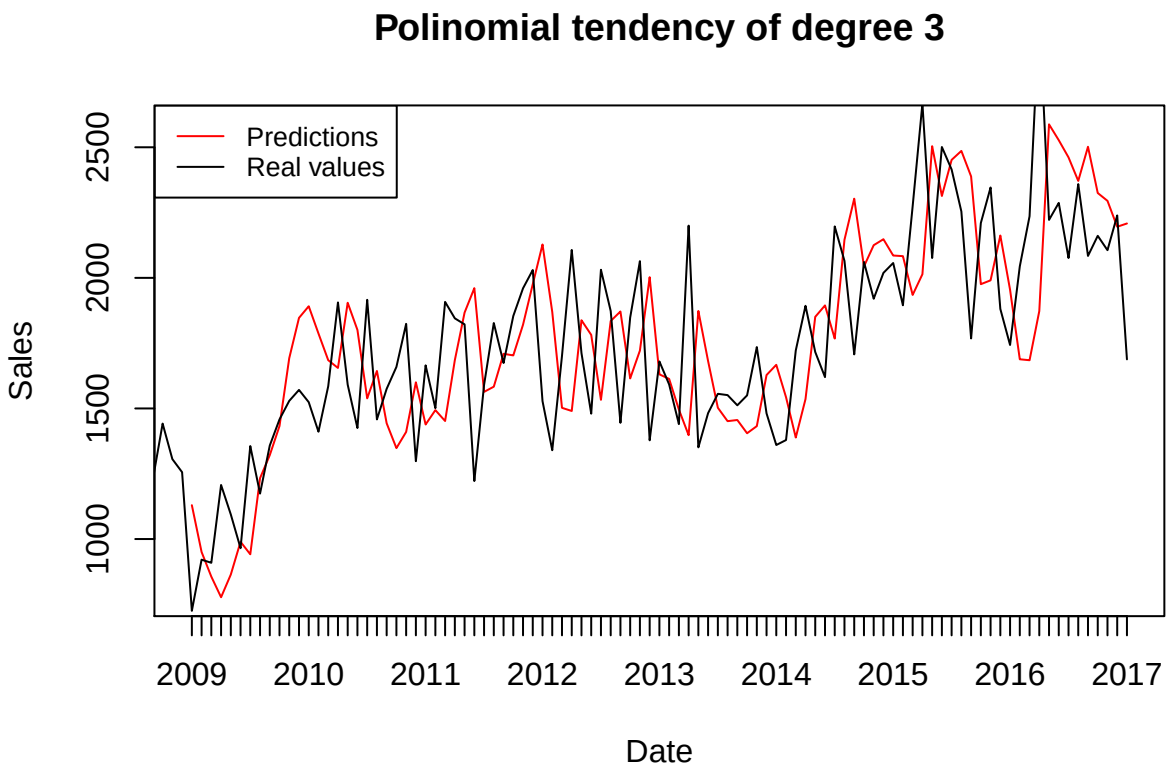
```
## Warning: ‘data_frame()‘ is deprecated as of tibble 1.1.0.
## Please use ‘tibble()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

```r
MAPE <- mean(abs((testSales-predictions1)/testSales))
ModelMAPE["Poly1T"] <- MAPE
MAPE <- mean(abs((testSales-predictions2)/testSales))
ModelMAPE["Poly2T"] <- MAPE
MAPE <- mean(abs((testSales-predictions3)/testSales))
ModelMAPE["Poly3T"] <- MAPE
print(ModelMAPE)
```

```
##              Poly1T    Poly2T    Poly3T
##        NA 0.1467764 0.1363978 0.1578839
```

O melhor resultado foi o obtido com o polinômio de terceiro grau, dessa forma, vamos visualizar o modelo resultante:

```r
# visualization of fitted model
predictions <- NA
for(i in seq_along(trainSales)){
  fit <- lm(trainSales[[i]]~poly(t, 3))
  predictions[i] <- predict(fit, newdata = data_frame(t= 25))[1]
}
plot(x = index(carSales)[(121-96):121], y = predictions, type = 'l',
     col = 'red', xlab =  "Date", ylab = "Sales", main = "Polinomial tendency of degree 3")
lines(carSales)
legend("topleft", legend=c("Predictions", "Real values"), col=c('red','black'), lty = 1:1, cex=0.8)
```



O modelo não é capaz de capturar a tendência presente nos dados.

Variáveis sazonais categóricas

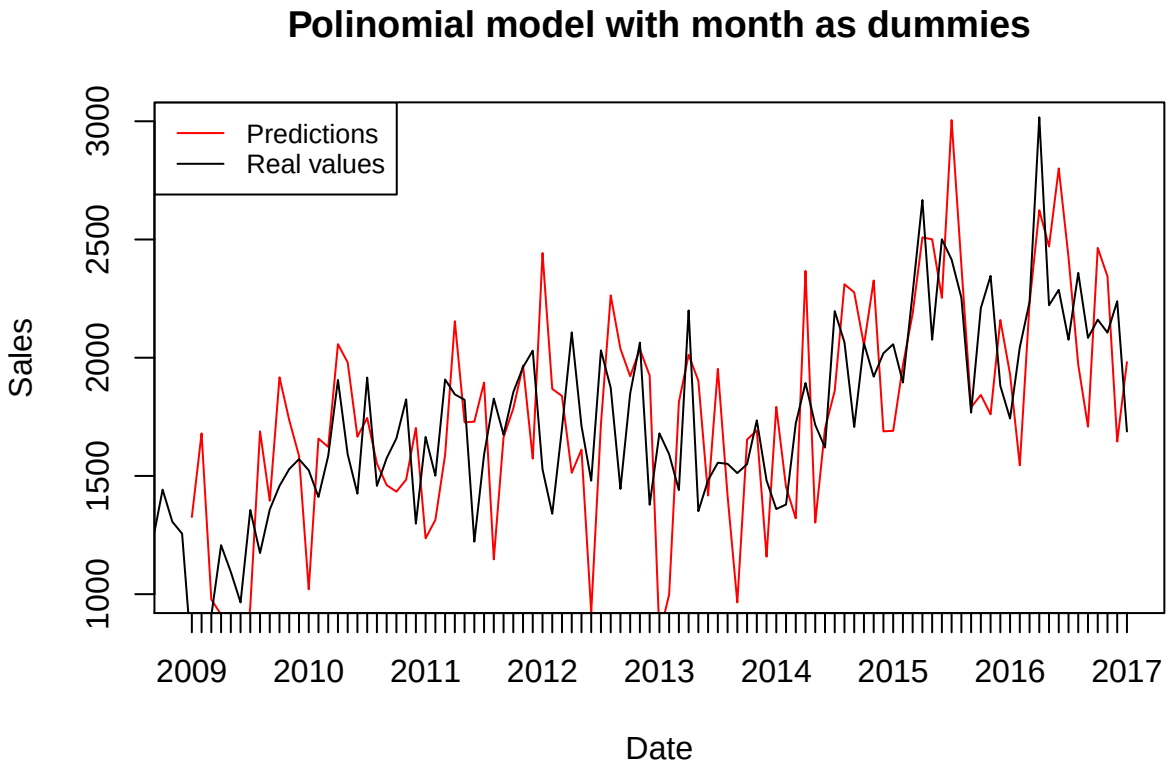
Uma segunda forma de modelar a série de forma linear é utilizar cada um dos meses como uma variável categórica, isto é, um conjunto de 11 variáveis i tal que $i = 1$ se o mês da observação é o mêss i (um dos meses é utilizado como valor de referência).

```
Q <- factor(rep(1:12, length.out = 121))
predictions <- NA
for(n in seq_along(trainSales)){
  Qn <- Q[n:(n+23)]
  Qnew <- Q[(n:24)]
  fit <- lm(trainSales[[n]]~Qn+poly(t,3))
  predictions[n] <- predict(fit,  newdata = data_frame(t = 25, Qn = Qnew))[1]
}
MAPE <- mean(abs((testSales-predictions)/testSales))
ModelMAPE["SazonalDummy"] <- MAPE
print(ModelMAPE)
```

```
##                   Poly1T       Poly2T       Poly3T SazonalDummy
##           NA    0.1467764    0.1363978    0.1578839    0.2002887
```

Obtivémos um resultado praticamente o mesmo do que o anterior, visualizando o resultado:

```
# visualization of fitted model
predictions <- NA
for(n in seq_along(trainSales)){
  Qn <- Q[n:(n+23)]
   Qnew <- Q[(n:24)]
  fit <- lm(trainSales[[n]]~Qn+poly(t,3))
  predictions[n] <- predict(fit,  newdata = data_frame(t = 25, Qn = Qnew))[1]
}
plot(x = index(carSales)[(121-96):121], y = predictions, type = 'l',
     col = 'red', xlab =  "Date", ylab = "Sales", main = "Polinomial model with month as dummies", ylim
lines(carSales)
legend("topleft", legend=c("Predictions", "Real values"), col=c('red','black'), lty = 1:1, cex=0.8)
```



Loess

Para o modelo com Loess, criamos uma nova coluna nos dados que junta o mês com o ano:

```
data<-transform(data, Yearmon = as.yearmon(paste(Year, Month, sep = "-")))
data$Date<-as.Date(data$Yearmon)
```

Com isso, criamos a timeseries dos dados, com a frequência igual a 12 indicando os anos

```
cmmts <- ts(data$Quantity, frequency=12, start=c(2007,1))
cmmts
```

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2007 2521 1517 1428 1257 1934 1531 1777 1665 1373 1941 1473 1272
## 2008 1224 1767 1503 1795 1402 1182 1203 1432 1229 1442 1306 1256
## 2009  725  921  909 1207 1094  965 1356 1174 1358 1458 1529 1571
## 2010 1524 1411 1585 1906 1591 1425 1916 1458 1575 1659 1824 1298
## 2011 1665 1501 1908 1845 1822 1222 1590 1827 1674 1854 1960 2030
## 2012 1528 1340 1701 2107 1712 1480 2031 1873 1445 1848 2064 1378
## 2013 1680 1592 1440 2200 1351 1482 1556 1551 1512 1550 1735 1481
## 2014 1360 1379 1722 1893 1716 1620 2197 2065 1707 2061 1920 2019
## 2015 2057 1895 2274 2667 2076 2501 2415 2254 1768 2210 2346 1881
## 2016 1743 2044 2236 3017 2222 2287 2076 2359 2084 2161 2106 2239
## 2017 1688
```

Para cada janela de 25 meses, decompomos a série temporal com o loess, através da função STL e calculamos a quantidade para o próximo mês com a função forecast:

```
predictions <- NA
for(i in 0:96){
  ano_0 <- 2007+i%/%12
  mes_0 <- 1+i%%12
  ano_1 <- ano_0+2
  mes_1 <- mes_0
  cmmts_w <- window(cmmts, start=c(ano_0,mes_0), end = c(ano_1,mes_1))
  fitstl<-stl(cmmts_w, s.window = "period")
  etsfc<-forecast(fitstl, method = "ets", h=1)
  predictions[i] <- etsfc$mean
}
ModelMAPE["Loess"] <- mean(abs((testSales[-1]-predictions)/testSales[-1]))
```

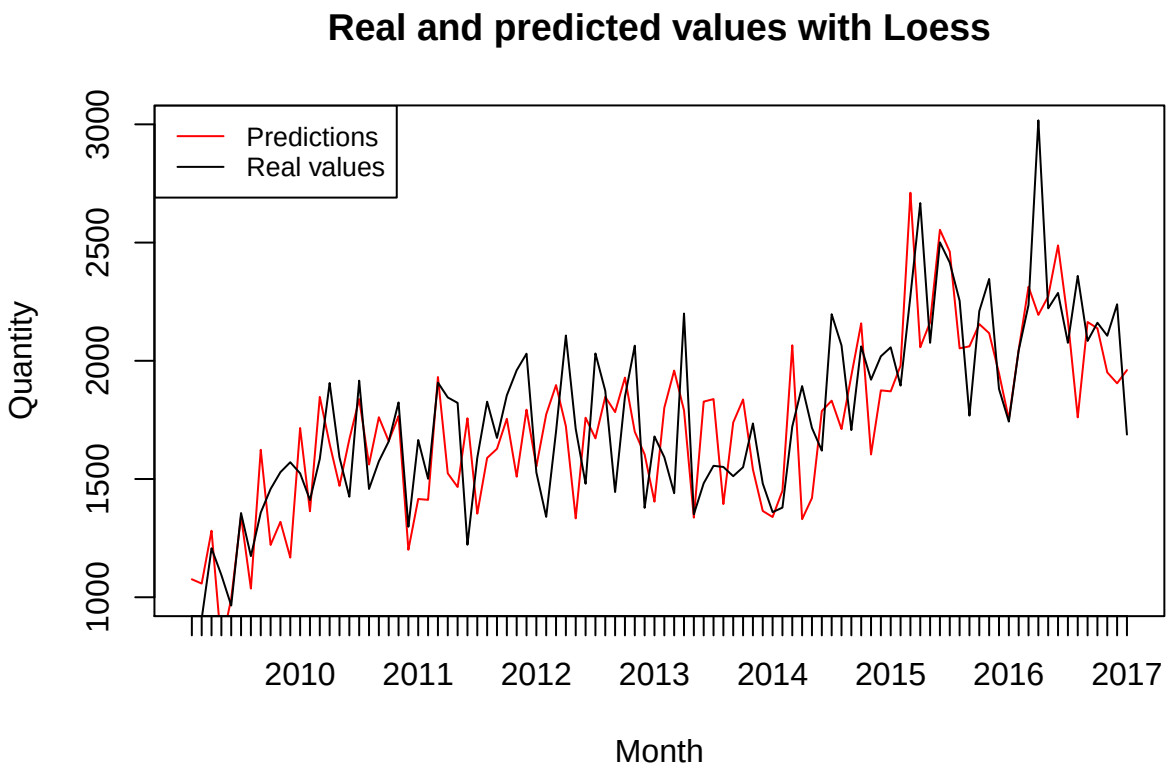
Utilizamos a janela de 25 meses, ao invés de 24, por conta da função STL solicitar dados que sejam maiores que duas vezes a frequência da série temporal.

O MAPE utilizando loess é o menor até então:

```
ModelMAPE
```

```
##                   Poly1T       Poly2T       Poly3T SazonalDummy        Loess
##           NA    0.1467764    0.1363978    0.1578839    0.2002887    0.1240543
```

```
plot(x = index(carSales)[(121-95):121], y = predictions, type = "l",xlab="Month", ylab="Quantity", col=
lines(carSales[(121-95):121])

legend("topleft", legend=c("Predictions", "Real values"), col=c('red','black'), lty = 1:1, cex=0.8)
```

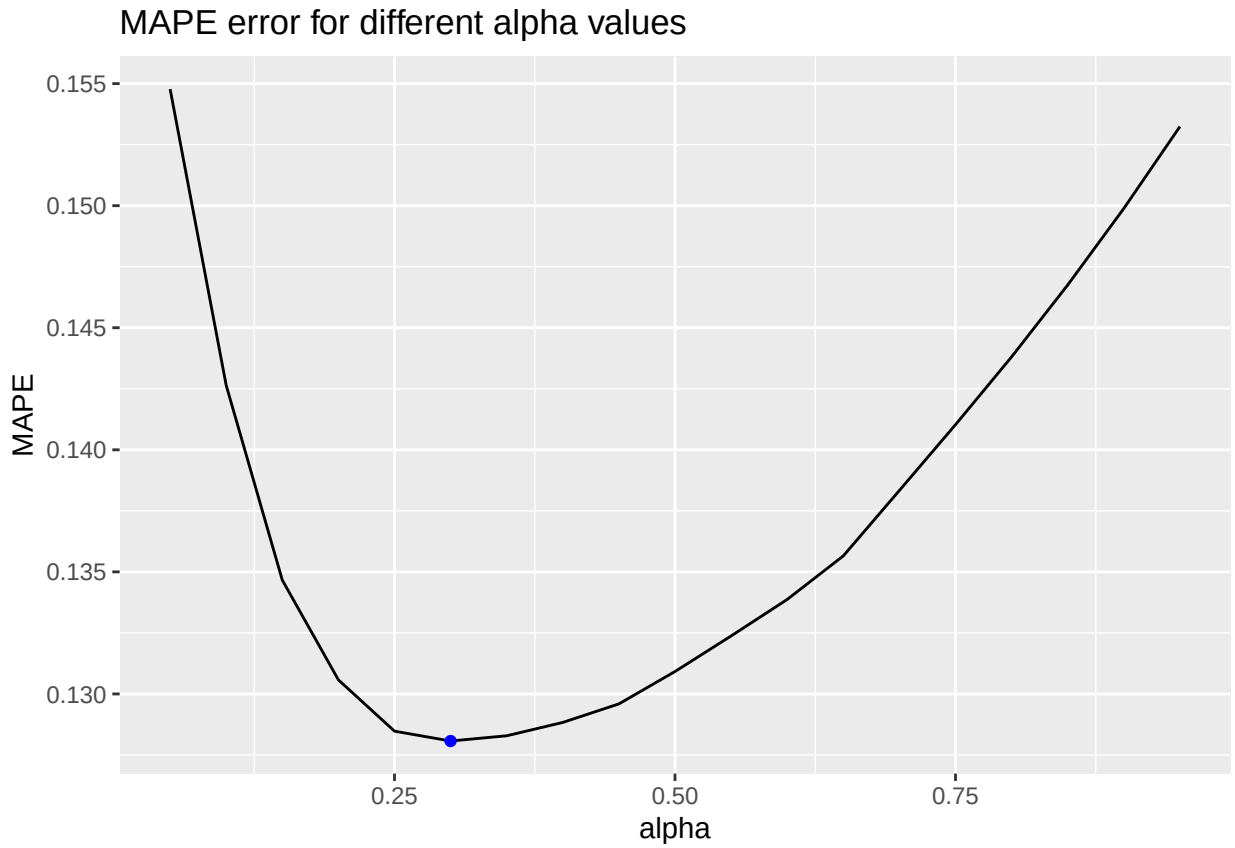


Suavização exponencial simples

Esse método não considera tendência e também não considera sazonalidade, ele utiliza de um hiperparâmetro que irá ser otimizado para aquele que minimiza o MAPE.

```
#SIMPLE EXPONENTIAL SMOOTHING
#TUNNING ALPHA PARAMETER
alpha <- seq(0.05, 0.95, by = .05)
MAPE <- NA
for(i in seq_along(alpha)){
  predictions <- NA
  for(j in seq_along(trainSales)){
    fit <- ses(trainSales[[j]], alpha = alpha[i], h = 1)
    predictions[j] <- fit$mean[1]
  }
  MAPE[i] <- mean(abs((testSales-predictions)/testSales))
}

alpha.err <- tibble(alpha, MAPE)
alpha.min <- filter(alpha.err, MAPE == min(MAPE))
ModelMAPE["ses"] <- alpha.min$MAPE

ggplot() +
  geom_line(data = alpha.err, aes(x = alpha, y = MAPE)) +
```

```
  geom_point(data = alpha.min, aes(x = alpha, y = MAPE), color = 'blue') +
  ggtitle("MAPE error for different alpha values")
```

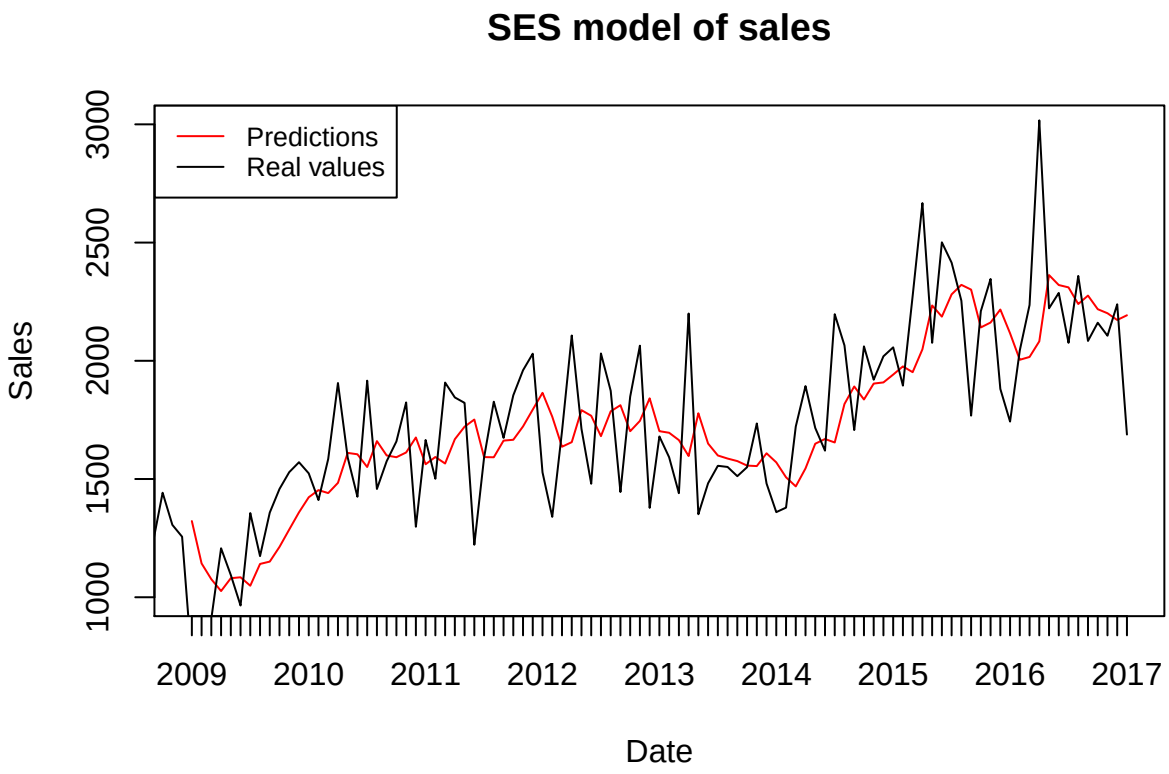



```
ModelMAPE
```

```
##                   Poly1T       Poly2T       Poly3T SazonalDummy        Loess
##           NA    0.1467764    0.1363978    0.1578839    0.2002887    0.1240543
##          ses
##    0.1280711
```

O valor ótimo encontrado para o alpha foi de 0.3 com um MAPE = 0.128, dessa forma, vamos criar o modelo com o alpha estimado e visualizar:

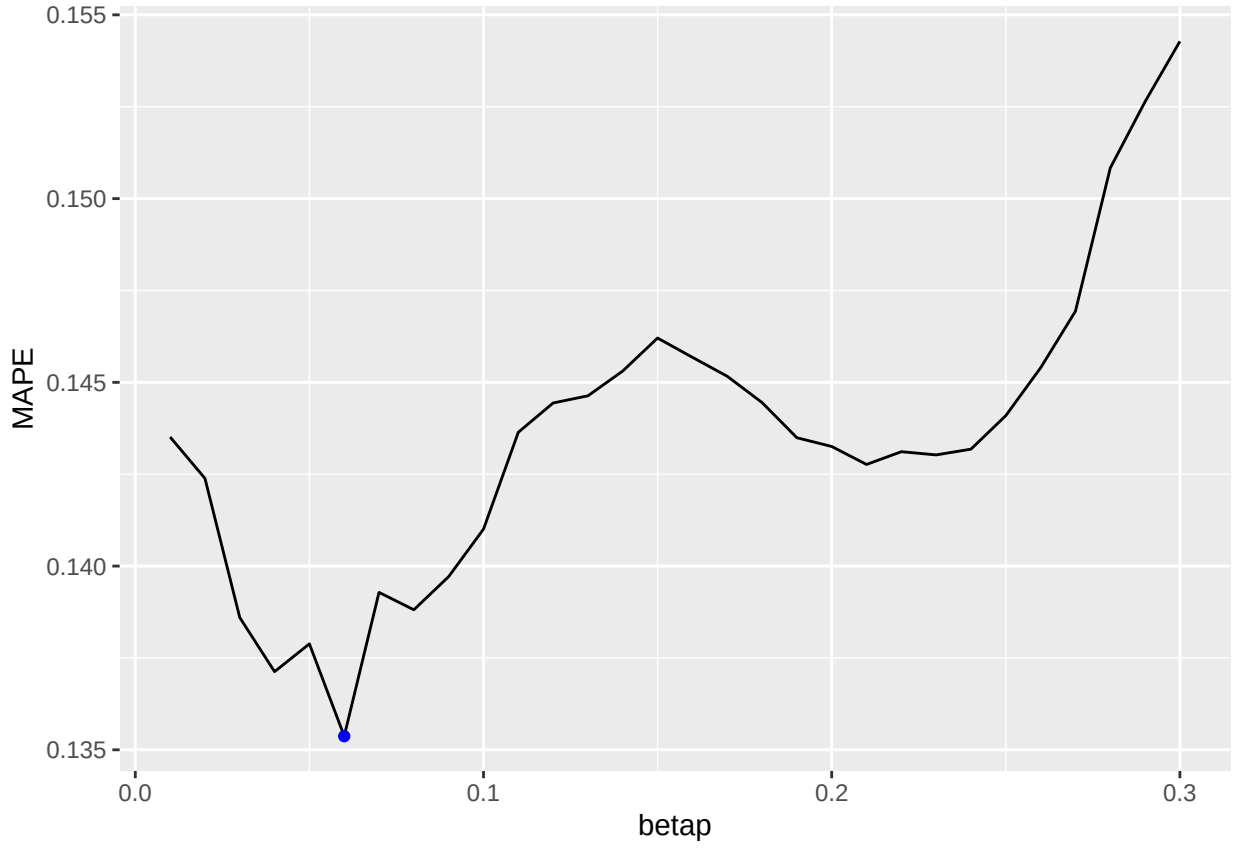
```
# visualization of fitted model
predictions <- NA
for(i in seq_along(trainSales)){
  fit <- ses(trainSales[[i]], alpha = 0.3, h = 1)
  predictions[i] <- fit$mean[1]
}


plot(x = index(carSales)[(121-96):121], y = predictions,
     type = 'l', col = 'red', xlab = "Date", ylab = "Sales", main = "SES model of sales", ylim = c(1000
lines(carSales)
legend("topleft", legend=c("Predictions", "Real values"), col=c('red','black'), lty = 1:1, cex=0.8)
```



Método de Holt

Nesse segundo modelo, método de Holt, também é considerada a existência de uma tendência nos dados.

```
betap <- seq(0.01, 0.3, by = 0.01)
MAPE <- NA
for(k in seq_along(betap)){
  predictions <- NA
  for(j in seq_along(trainSales)){
    fit <- holt(trainSales[[j]], beta = betap[k], h = 1, type = "Additive")
    predictions[j] <- fit$mean[1]
  }
  MAPE[k] <- mean(abs((testSales-predictions)/testSales))
}

beta.err <- data_frame(betap, MAPE)
beta.min <- filter(beta.err, MAPE == min(MAPE))
ModelMAPE["HoltAdd"] <- beta.min$MAPE

ggplot() +
  geom_line(data = beta.err, aes(x = betap, y = MAPE)) +
  geom_point(data = beta.min, aes(x = betap, y = MAPE), color = 'blue')
```

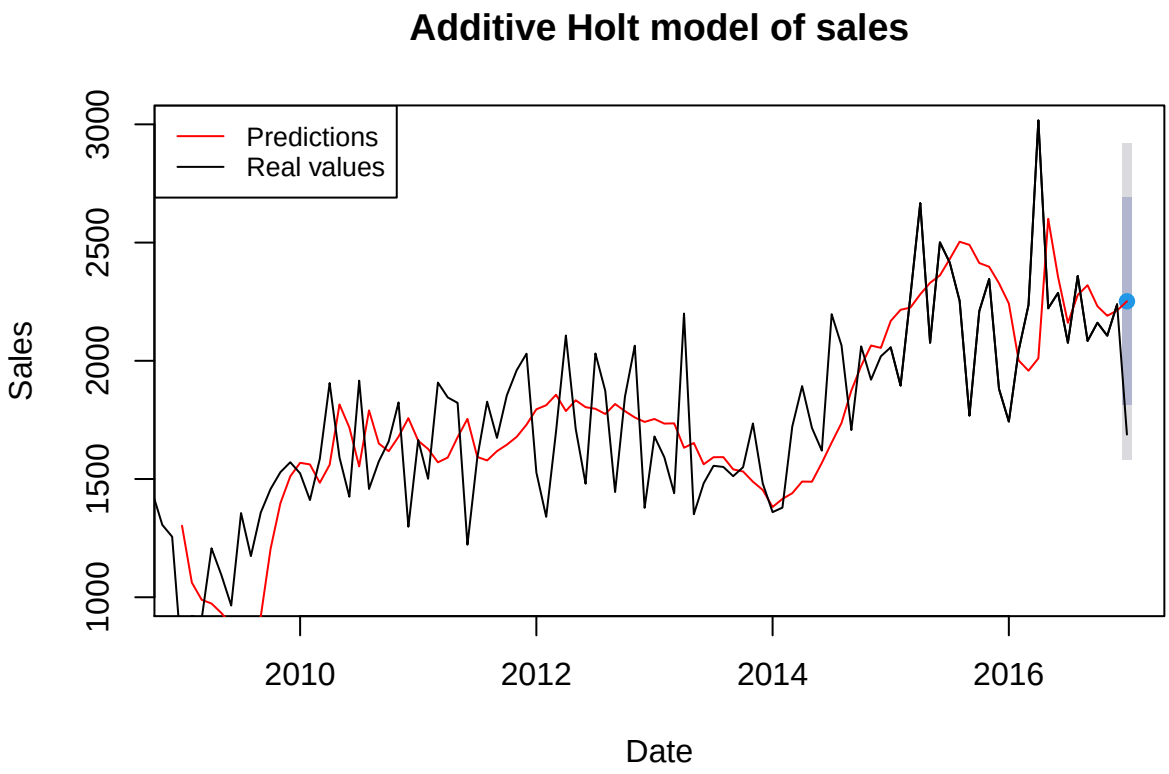



```
print(ModelMAPE)
```

```
##                   Poly1T       Poly2T       Poly3T SazonalDummy        Loess
##           NA    0.1467764    0.1363978    0.1578839    0.2002887    0.1240543
##          ses      HoltAdd
##    0.1280711    0.1353717
```

Obtemos um MAPE mínimo de 0.135 para o valor de beta = 0.06. Vamos visualizar as predições para este modelo:

```
# visualization of fitted model
predictions <- NA
for(i in seq_along(trainSales)){
  fit <- holt(trainSales[[i]], beta = 0.06, h = 1, type = "Additive")
  predictions[i] <- fit$mean[1]
}
plot(fit, xlim = c(startDate - 6, finalDate), ylim = c(1000, 3000),
     xlab = "Date", ylab = "Sales", main = "Additive Holt model of sales")
lines(x = index(carSales)[(121-96):121], y = predictions, col = 'red')
lines(carSales)
legend("topleft", legend=c("Predictions", "Real values"), col=c('red','black'), lty = 1:1, cex=0.8)
```



Holt Winters

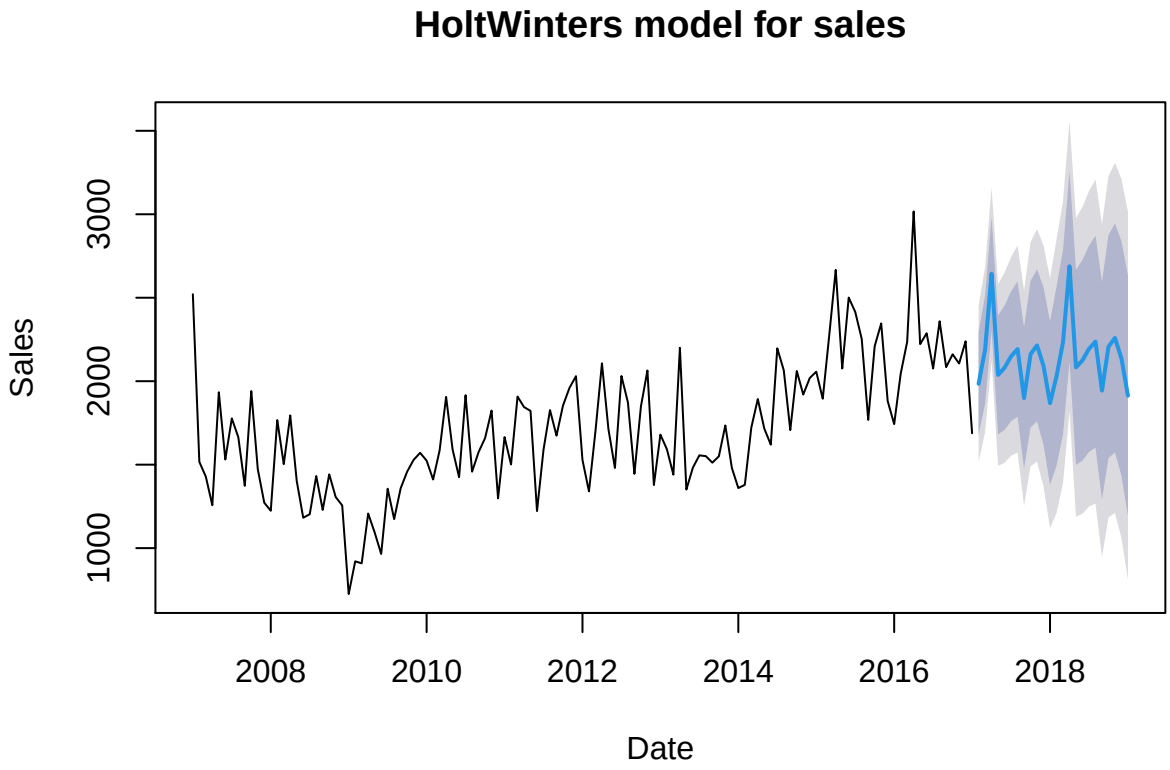
Por fim, este modelo considera que os dados possuem tendência e sazonalidade, dessa forma, utiliza de 3 parâmetros que devem ser otimizados. Para não percorrermos uma combinação muito grande de parâmetros, vamos utilizar do método automático de otimização da biblioteca.

```
autoModel <- HoltWinters(carSales)
print(autoModel)
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = carSales)
##
## Smoothing parameters:
##  alpha: 0.3241108
##  beta : 0.02532339
##  gamma: 0.4356731
##
## Coefficients:
##            [,1]
## a   2036.038476
## b      3.727398
## s1   -55.139433
```

```
## s2   145.770922
## s3   596.581750
## s4   -13.120264
## s5    26.002733
## s6    90.543842
## s7   130.233673
## s8  -166.944394
## s9    92.313871
## s10  141.239396
## s11   13.800433
## s12 -212.414353
```

```r
plot(forecast(autoModel), xlab = "Date", ylab =  "Sales", main = "HoltWinters model for sales")
```

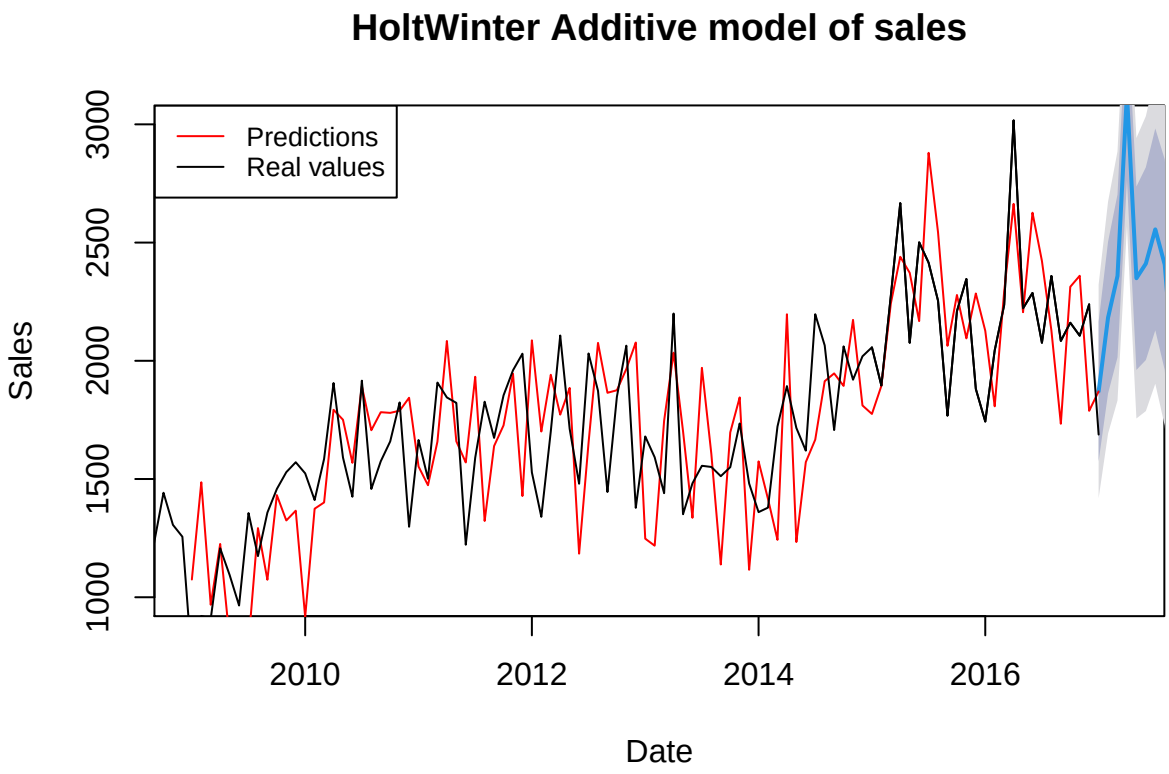


Observando então os parâmetros escolhidos, vamos percorrer diferentes valores para parâmetros próximos aos obtidos pelo método, escolhendo assim aquele que minimiza o MAPE.

```r
hwalpha <- seq(0.2, 0.5, length.out = 5)
hwbeta <- seq(0.001, 0.034, length.out = 5)
hwgamma <- seq(0.01, 0.5, length.out = 5)

MAPE <- array(dim = c(5, 5, 5))
for(i in seq_along(hwalpha)){
  for(j in seq_along(hwbeta)){
    for(k in seq_along(hwgamma)){
      predictions <- NA
```

```
      for(n in seq_along(trainSales)){
        fit <- HoltWinters(trainSales[[n]], alpha = hwalpha[i],
                           beta = hwbeta[j], gamma = hwgamma[k])
        predictions[n] <- predict(fit, n.ahead = 1)[1]
      }
      MAPE[i,j,k] <- mean(abs((testSales-predictions)/testSales))
    }
  }
}

MAPEmin <- 10000
bestParams <- NA
for(i in seq_along(hwalpha)){
  for(j in seq_along(hwbeta)){
    for(k in seq_along(hwgamma)){
      if(MAPEmin > MAPE[i, j, k]){
        MAPEmin <- MAPE[i, j, k]
        bestParams <- c(hwalpha[i], hwbeta[j], hwgamma[k])
      }
    }
  }
}

ModelMAPE["HoltWinterAdd"] <- MAPEmin
```

```
#estimated values for final model
# visualization of fitted model
predictions <- NA
for(i in seq_along(trainSales)){
  fit <- HoltWinters(trainSales[[i]], alpha = bestParams[1],
                     beta = bestParams[2], gamma = bestParams[3])
  predictions[i] <- predict(fit, n.ahead = 1)[1]
}

plot(forecast(fit), xlim = c(as.yearmon("2009-01-01"), endDate + 1/4),
     xlab = "Date", ylab = "Sales", ylim = c(1000, 3000), main = "HoltWinter Additive model of sales")
lines(x = index(carSales)[(121-96):121], y = predictions, type = 'l', col = 'red')
lines(carSales)
legend("topleft", legend=c("Predictions", "Real values"), col=c('red','black'), lty = 1:1, cex=0.8)
```

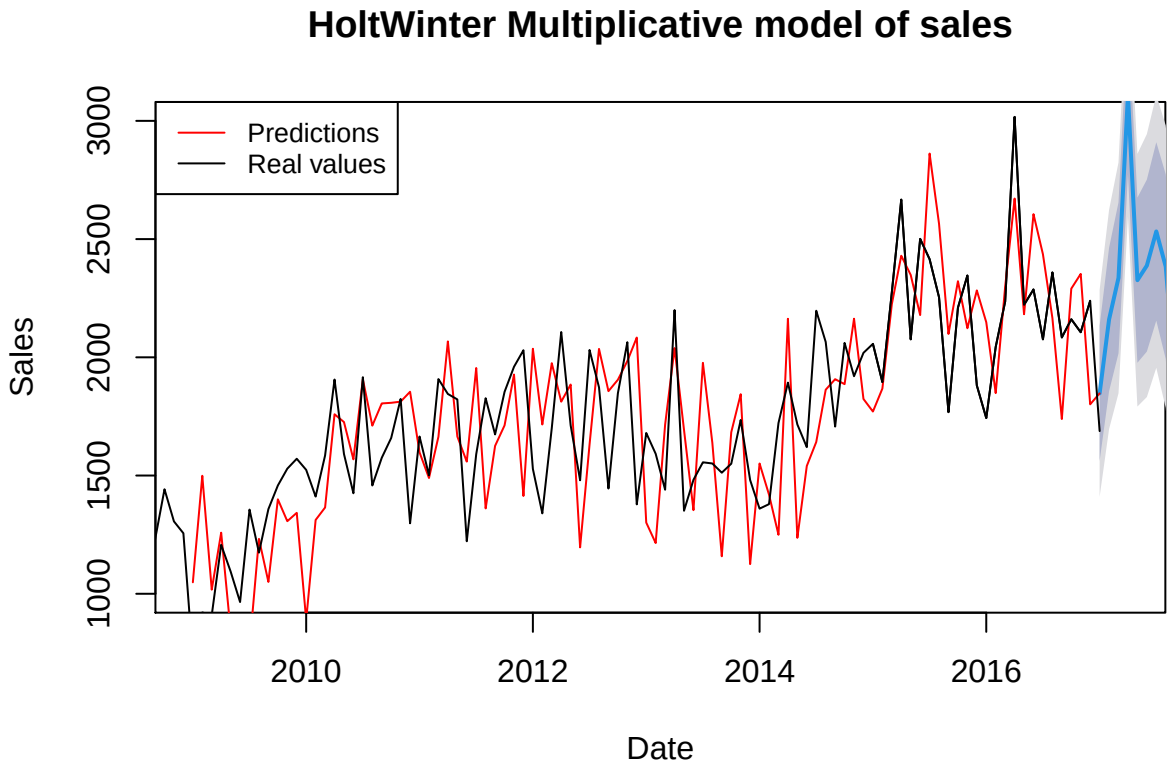


Finalizaremos tentando utilizar o modelo HoltWinters multiplicativo:

```
hwalpha <- seq(0.2, 0.5, length.out = 5)
hwbeta <- seq(0.001, 0.034, length.out = 5)
hwgamma <- seq(0.01, 0.5, length.out = 5)

MAPE <- array(dim = c(5, 5, 5))
for(i in seq_along(hwalpha)){
  for(j in seq_along(hwbeta)){
    for(k in seq_along(hwgamma)){
      predictions <- NA
      for(n in seq_along(trainSales)){
        fit <- HoltWinters(trainSales[[n]], alpha = hwalpha[i],
                           beta = hwbeta[j], gamma = hwgamma[k], seasonal = "multiplicative")
        predictions[n] <- predict(fit, n.ahead = 1)[1]
      }
      MAPE[i,j,k] <- mean(abs((testSales-predictions)/testSales))
    }
  }
}

MAPEmin <- 10000
bestParams <- NA
for(i in seq_along(hwalpha)){
  for(j in seq_along(hwbeta)){
    for(k in seq_along(hwgamma)){
```

```
      if(MAPEmin > MAPE[i, j, k]){
        MAPEmin <- MAPE[i, j, k]
        bestParams <- c(hwalpha[i], hwbeta[j], hwgamma[k])
      }
    }
  }
}

ModelMAPE["HoltWinterMul"] <- MAPEmin
```

```
#estimated values for final model
# visualization of fitted model
predictions <- NA
for(i in seq_along(trainSales)){
  fit <- HoltWinters(trainSales[[i]], alpha = bestParams[1],
                     beta = bestParams[2], gamma = bestParams[3])
  predictions[i] <- predict(fit, n.ahead = 1)[1]
}

plot(forecast(fit), xlim = c(as.yearmon("2009-01-01"), endDate + 1/4),
     xlab = "Date", ylab = "Sales", ylim = c(1000, 3000), main = "HoltWinter Multiplicative model of sa
lines(x = index(carSales)[(121-96):121], y = predictions, type = 'l', col = 'red')
lines(carSales)
legend("topleft", legend=c("Predictions", "Real values"), col=c('red','black'), lty = 1:1, cex=0.8)
```



Resultados

Comporando por final o valor do erro MAPE para cada um dos modelos considerados.

```
print(ModelMAPE)
```

```
##                    Poly1T        Poly2T        Poly3T  SazonalDummy
##            NA     0.1467764     0.1363978     0.1578839     0.2002887
##         Loess           ses       HoltAdd HoltWinterAdd HoltWinterMul
##     0.1240543     0.1280711     0.1353717     0.1552255     0.1538115
```