Rodando modelos de regressão linear no R

César Lemos, Bsc. Matemática

Abstract

Utilizando modelos de regressão (linear e log-log) para gerar previsões para a taxa de desemprego.

Contents

1 Pacotes Usados

```
library(sidrar)
library(readr)
library(caret)
library(forecast)
library(xtable)
library(tidyverse)
library(scales)
library(gridExtra)
```

2 Coleta de Dados

```
### Importando os dados

# Xreg
xreg <- read_csv2(choose.files()) # Importando o dado através do choose.file
xreg <- ts(xreg[,-1], start = c(2012,03), frequency = 12) # Transformando em séries temporais

# Dados brutos do PNAD
pnad <- get_sidra(api='/t/6318/n1/all/v/1641/p/all/c629/all')

# Coletando a PEA no PNAD
pea = pnad$Valor[pnad$`Condição em relação à força de trabalho e condição de ocupação (Código)`==32386]

# Coletando a População Desocupada no PNAD
desocupada = pnad$Valor[pnad$`Condição em relação à força de trabalho e condição de ocupação (Código)`==

# Criando a variável Desemprego
desemprego <- ts(desocupada/pea*100, start = c(2012,03), frequency = 12)

# Compilando os dados
data <- ts.intersect(desemprego, xreg)
colnames(data) <- c('desemprego', 'icd', 'iaemp', 'iie', 'google',
                    'ibc', 'selic')
```

3 Tratando os Dados

```
# Verificando missing value
sapply(data, function(x) sum(is.na(x)))
```

```
## desemprego        icd      iaemp        iie     google        ibc
##          0          0          0          0          0          1
##      selic
##          0
```

```r
tail(data) # Verificando os últimos registros para tentar localizar o missing value
```

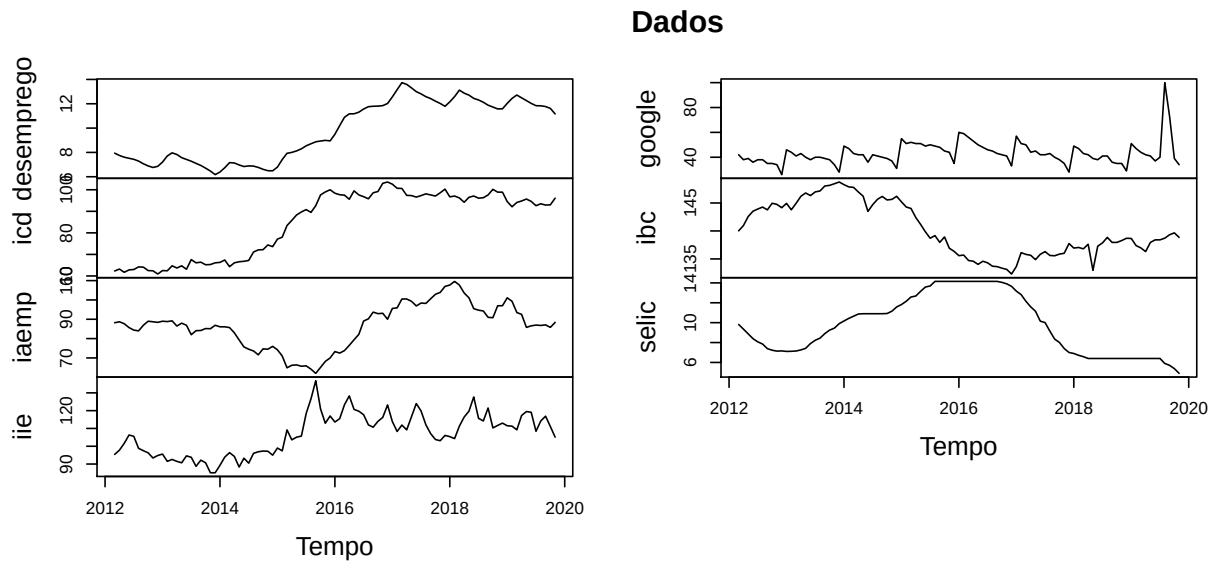
```
##          desemprego  icd iaemp   iie google    ibc selic
## Jun 2019   12.03114 94.6  86.6 119.1     37 138.41  6.40
## Jul 2019   11.84046 92.6  87.0 108.4     40 138.41  6.40
## Aug 2019   11.83201 93.5  86.8 114.2    100 138.71  5.90
## Sep 2019   11.77162 92.9  87.1 116.9     73 139.32  5.71
## Oct 2019   11.62083 93.0  85.8 111.1     39 139.66  5.38
## Nov 2019   11.16213 96.1  88.4 105.1     34     NA  4.90
```

```r
# Forecast com auto.arima para completar missing values
ibc.fcast <- forecast(auto.arima(data[,6], max.p = 4, max.q = 4, seasonal = F),
                      h = 1, level = 40)$lower # Escolhendo o limite inferior
                                               # do intervalo de confiança da projeção do IBC

data[nrow(data),6] <- ibc.fcast # Inserindo o valor projetado do IBC no lugar do missing value
tail(data) # Checando os valores
```

```
##          desemprego  icd iaemp   iie google     ibc selic
## Jun 2019   12.03114 94.6  86.6 119.1     37 138.410  6.40
## Jul 2019   11.84046 92.6  87.0 108.4     40 138.410  6.40
## Aug 2019   11.83201 93.5  86.8 114.2    100 138.710  5.90
## Sep 2019   11.77162 92.9  87.1 116.9     73 139.320  5.71
## Oct 2019   11.62083 93.0  85.8 111.1     39 139.660  5.38
## Nov 2019   11.16213 96.1  88.4 105.1     34 138.848  4.90
```

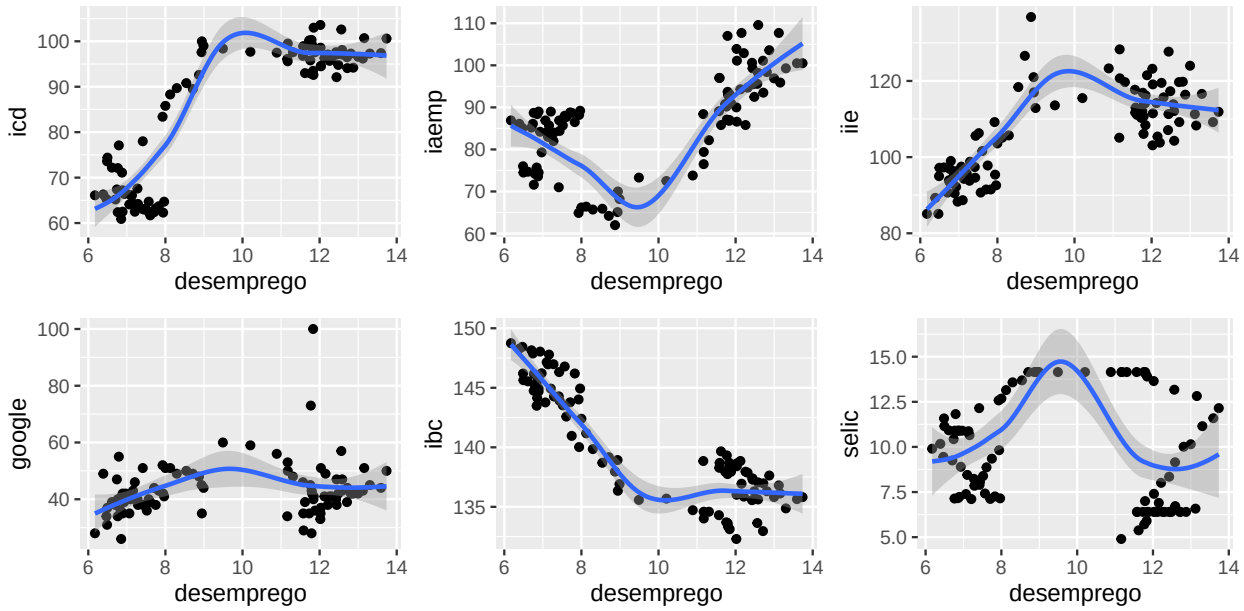
4 Plotando os gráficos das variáveis

```r
# Plotando todos os dados
plot(data, main = "Dados", xlab = "Tempo")
```



5 Plotando os gráficos de correlação entre desemprego e as demais variáveis

```
# Plotando as Correlações
corr1 <- ggplot(as.data.frame(data)) +
  geom_point(mapping = aes(x = desemprego, y = icd)) +
  geom_smooth(mapping = aes(x = desemprego, y = icd))

corr2 <- ggplot(as.data.frame(data)) +
  geom_point(mapping = aes(x = desemprego, y = iaemp)) +
  geom_smooth(mapping = aes(x = desemprego, y = iaemp))

corr3 <- ggplot(as.data.frame(data)) +
  geom_point(mapping = aes(x = desemprego, y = iie)) +
  geom_smooth(mapping = aes(x = desemprego, y = iie))

corr4 <- ggplot(as.data.frame(data)) +
  geom_point(mapping = aes(x = desemprego, y = google)) +
  geom_smooth(mapping = aes(x = desemprego, y = google))

corr5 <- ggplot(as.data.frame(data)) +
  geom_point(mapping = aes(x = desemprego, y = ibc)) +
  geom_smooth(mapping = aes(x = desemprego, y = ibc))

corr6 <- ggplot(as.data.frame(data)) +
  geom_point(mapping = aes(x = desemprego, y = selic)) +
  geom_smooth(mapping = aes(x = desemprego, y = selic))

grid.arrange(corr1, corr2, corr3, corr4, corr5, corr6, ncol=3)
```



6 Criando amostas de treino e teste

```
set.seed(1234) # Garantindo a reprodutibilidade do experimento

intrain <- createDataPartition(data[,1], p = 0.7, list = F)
treino <- as.data.frame(data[intrain,])
teste <- as.data.frame(data[-intrain,])
```

7 Criando o modelo de Regressão Linear

```
lm <- lm(desemprego~., data = treino)
summary(lm)
```

```
##
## Call:
## lm(formula = desemprego ~ ., data = treino)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79813 -0.32431 -0.00242  0.30542  1.05776
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.190512   7.480786   0.694  0.49041
## icd          0.080484   0.009277   8.676 3.05e-12 ***
## iaemp        0.083745   0.010236   8.181 2.15e-11 ***
## iie          0.033894   0.011113   3.050  0.00338 **
## google       0.020660   0.006327   3.265  0.00180 **
## ibc         -0.092036   0.039504  -2.330  0.02314 *
## selic       -0.111039   0.030929  -3.590  0.00066 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.474 on 61 degrees of freedom
## Multiple R-squared:  0.9662, Adjusted R-squared:  0.9629
## F-statistic: 290.6 on 6 and 61 DF,  p-value: < 2.2e-16
```

8 Criando o modelo de Regressão Log-Log

```
log.log <- lm(log(desemprego)~log(icd)+log(iaemp)+log(iie)+log(google)+log(ibc)+log(selic),
              data = treino)
summary(log.log)
```

```
##
## Call:
## lm(formula = log(desemprego) ~ log(icd) + log(iaemp) + log(iie) +
```

```
##     log(google) + log(ibc) + log(selic), data = treino)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.096757 -0.028210  0.002462  0.028153  0.093901
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.26323    3.48927   1.222 0.226479
## log(icd)     0.61118    0.06919   8.833 1.64e-12 ***
## log(iaemp)   0.64761    0.08013   8.082 3.18e-11 ***
## log(iie)     0.39356    0.12163   3.236 0.001963 **
## log(google)  0.13085    0.03130   4.180 9.47e-05 ***
## log(ibc)    -1.95079    0.53814  -3.625 0.000591 ***
## log(selic)  -0.12829    0.02805  -4.573 2.40e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04584 on 61 degrees of freedom
## Multiple R-squared:  0.972,  Adjusted R-squared:  0.9693
## F-statistic: 353.1 on 6 and 61 DF,  p-value: < 2.2e-16
```

9 Previsão

```
# Previsão com o modelo de regressão linear
lm_fcast <- forecast(lm, newdata = teste[,-1], level = 95)

# Previsão com o modelo de regressão log-log
log.log_fcast <- forecast(log.log, newdata = teste[,-1], level = 95)
```

10 Avaliação

- Linear Model

```
print(accuracy(lm_fcast$mean, teste[,1]))
```

ME RMSE MAE MPE MAPE

Test set -0.02762727 0.5460426 0.4291997 -0.2451491 4.421073

- Log-Log Model

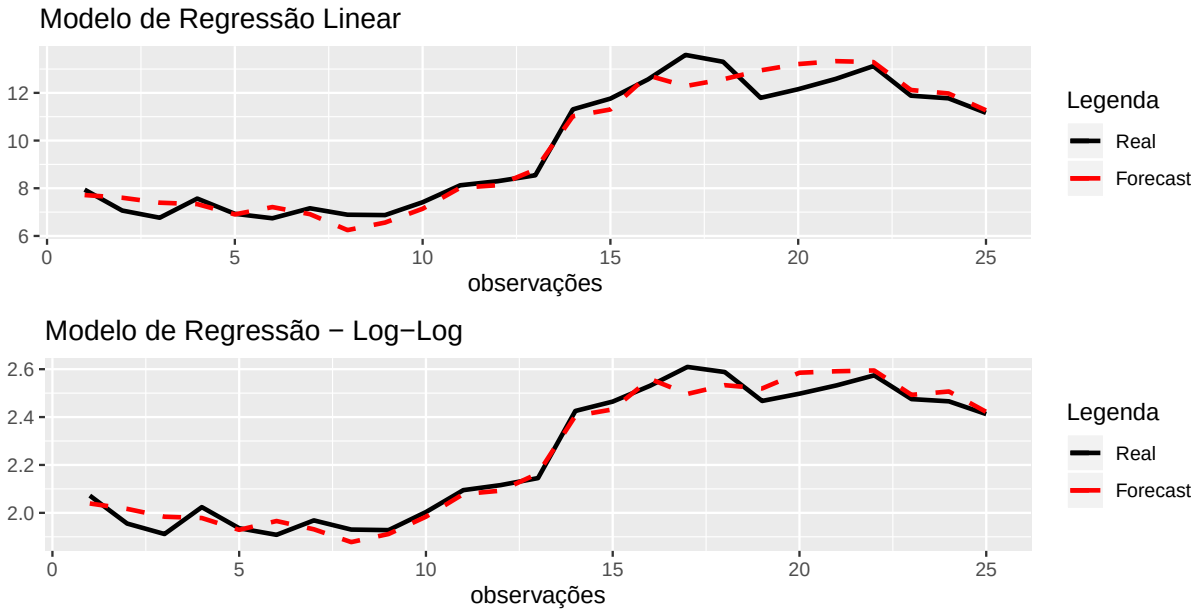
```
print(accuracy(log.log_fcast$mean, teste[,1]))
```

ME RMSE MAE MPE MAPE

Test set 7.488472 7.81937 7.488472 76.0667 76.0667

11 Plotando os gráficos das previsões

```
## Juntando os dados realizados do treino com o forecast e criando coluna índice
# para ser o eixo das abcissas

# Juntando os dados da regressão linear
dt1 <- as.data.frame(cbind(teste$desemprego, lm_fcast$mean))
dt1$id <- seq.int(nrow(dt1))
colnames(dt1) <- c("teste","forecast","id")

# Juntando os dados da regressão log-log
dt2 <- as.data.frame(cbind(log(teste$desemprego), log.log_fcast$mean))
dt2$id <- seq.int(nrow(dt2))
colnames(dt2) <- c("teste","forecast","id")

# Plotando os dados
lm_plot <- ggplot(dt1, aes(x=id)) +
  geom_line(mapping = aes(y = teste, color = "black"), size = 1) +
  geom_line(mapping = aes(y = forecast, color = "red"), size = 1, linetype = "dashed") +
  labs(title = "Modelo de Regressão Linear", x = "observações", y = "") +
  scale_color_identity(name = "Legenda",
                       breaks = c("black", "red"),
                       labels = c("Real","Forecast"),
                       guide = "legend")

log_plot <- ggplot(dt2, aes(x=id)) +
  geom_line(mapping = aes(y = teste, color = "black"), size = 1) +
  geom_line(mapping = aes(y = forecast, color = "red"), size = 1, linetype = "dashed") +
  labs(title = "Modelo de Regressão - Log-Log", x = "observações", y = "") +
  scale_color_identity(name = "Legenda",
                       breaks = c("black", "red"),
                       labels = c("Real","Forecast"),
                       guide = "legend")

grid.arrange(lm_plot, log_plot, nrow = 2)
```


Modelo de Regressão Linear
12
10
8
6
0
5
10
15
20
25
observações
Legenda
Real
Forecast

Modelo de Regressão − Log−Log
2.6
2.4
2.2
2.0
0
5
10
15
20
25
observações
Legenda
Real
Forecast