Lista 1 - Mineração

Victor Alves Dogo Martins, RA: 744878 Ana Beatriz Alves Monteiro, RA: 727838
Larissa Torres, RA: 631914

03-07-2022

Item 1

Como orientado no enunciado deste item, foi feita a normalização da covariável 'PIB per capita' através da seguinte fórmula:

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

Computacionalmente, esse procedimento foi feito através do comando `mutate` abaixo, onde temos o comentário `# Normalizando a covariavel`. Além disso, também segue uma parcela do banco de dados após a normalização para fins de demonstração:

```
### Carregando Pacotes

library(tidyverse)
library(knitr)
library(kableExtra)

# Lendo dados

df <- readr::read_csv('worldDevelopmentIndicators.csv') |>
  select(-CountryName) |>
  rename(y=LifeExpectancy,
         x=GDPercapita) |>
  mutate(x = (x-min(x))/(max(x)-min(x))) # Normalizando a covariavel

# Mostrando dados

head(df) |>
  kable('latex',digits=4, align='cc',
        caption = 'Primeiras linhas do banco de dados após normalização') |>
  kable_styling(position="center",
                latex_options="HOLD_position")
```

Table 1: Primeiras linhas do banco de dados após normalização

y	x
60.5091	0.0042
51.4640	0.0505
77.3505	0.0362
69.9497	0.0710
76.9579	0.4000
76.0127	0.1093

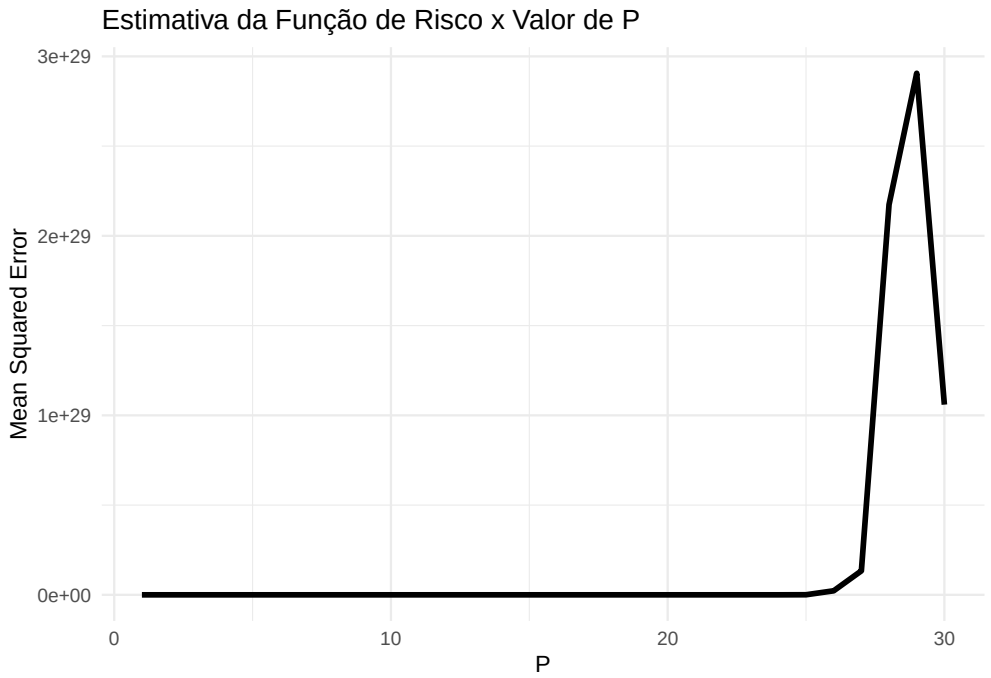
Item 2

```
# Criando lista com formulas de g(x) para cada valor de p

formulas <- list()

for (p in 1:30) {

  if(p==1){

    # Se p for igual a 1, a expressao mantem-se da forma abaixo

    formulas[[p]] <- "y~sin(2*pi*x)+cos(2*pi*x)"
  } else {

    # Para cada p maior do que 1, sua expressao g(x) sera dada pela expressao
    # do p anterior MAIS os seno e cosseno de 2*pi*x vezes o valor de p da
    # iteracao atual

    formulas[[p]] <- paste0(formulas[[p-1]],
                            "+sin(2*",p,
                            "*pi*x)+cos(2*",
                            p, "*pi*x)")
  }
}

# Definido tibble para guardarmos estimativas do risco das regressões de cada
# valor de p

tbl_result <- tibble(
  p=1:30,
  mse=as.double(1:30)
)

# Calculando erro quadrático médio para cada p via leave-one-out

for (p in 1:length(formulas)) {

  erros <- NULL

  model <- NULL
```

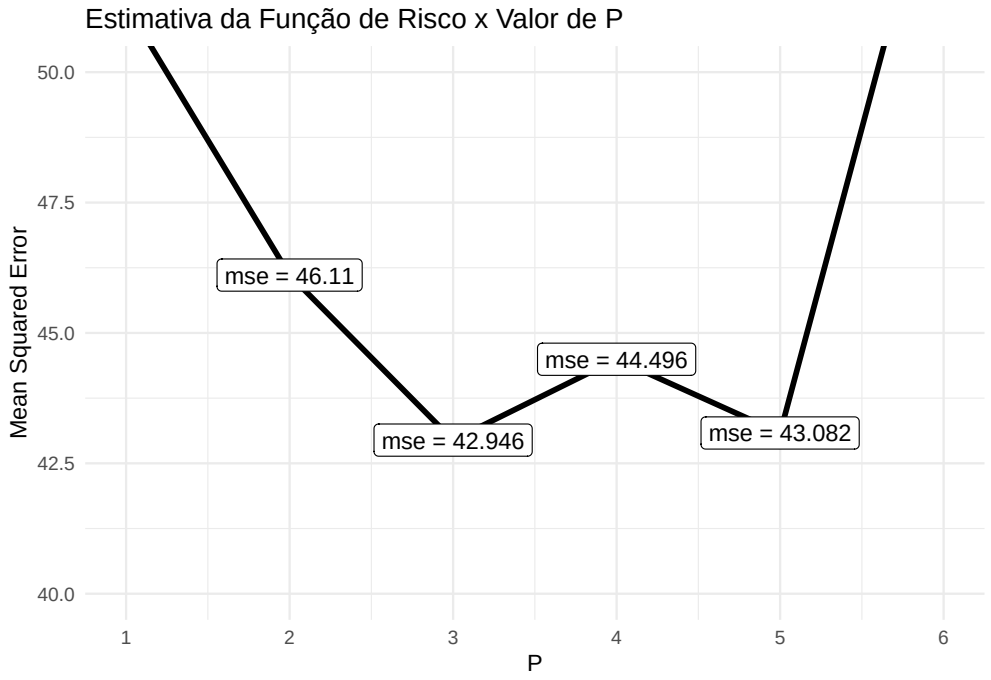
```r
  for (ii in 1:nrow(df)) {

    # Para o p atual, ajusta-se uma regressao retirando cada obs.
    # ii e utilizando o restante do banco de dados para validarmos
    # o modelo com a obs. ii retirada

    model <- lm(formulas[[p]], data=df[-ii,])

    erros <- c(erros,
               (df[ii,1] - predict(model, df[ii,]))^2)
  }

  # Fazendo a media dos erros dos ajustes para cada uma das observacoes
  # retiradas

  tbl_result[p,2] <- mean(unlist(erros))

}
```

Item 3

```r
tbl_result |>
  ggplot()+
  aes(x=p,y=mse)+
  geom_line(size=1.25)+
  theme_minimal()+
  labs(x = 'P', y = "Mean Squared Error",
       title='Estimativa da Função de Risco x Valor de P')
```



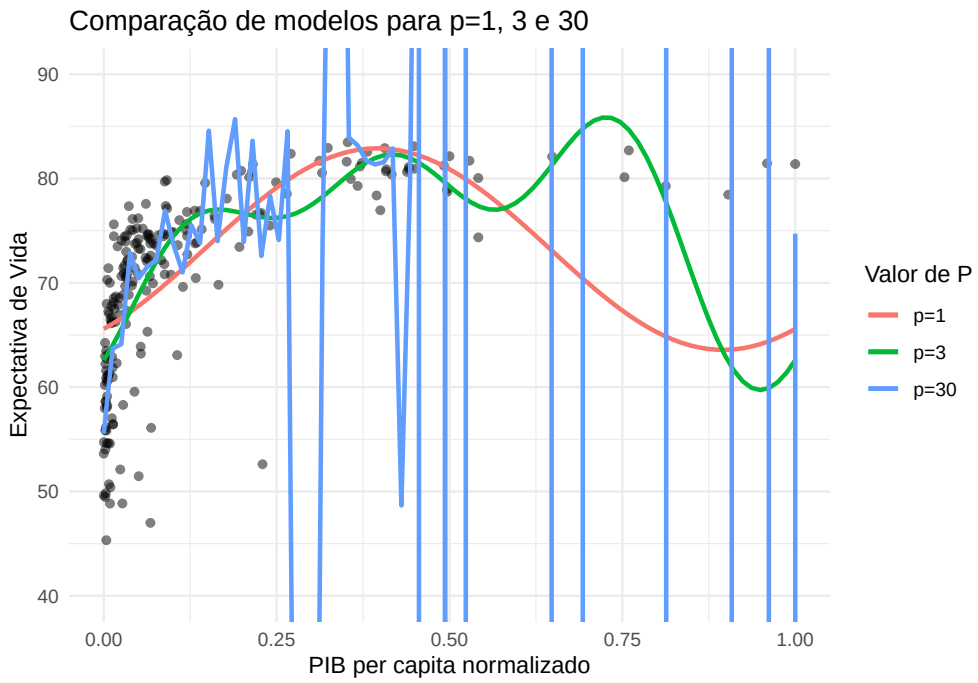
```r
tbl_result |>
  ggplot()+
  aes(x=p,y=mse)+
  geom_line(size=1.25)+
  geom_label(aes(label=paste0('mse = ', round(mse,3))))+
  coord_cartesian(ylim=c(40,50), xlim=c(1,6))+
  scale_x_continuous(breaks = 1:6)+
  theme_minimal()+
  labs(x = 'P', y = "Mean Squared Error",
       title='Estimativa da Função de Risco x Valor de P')
```



Item 4

```r
cores <- c('p=1'="#F8766D",
           'p=3'="#00BA38",
           'p=30'="#619BFF")

df |>
  ggplot()+
  aes(x=x,y=y)+
  geom_point(alpha=0.5)+
  geom_smooth(aes(color='p=1'),
              method = 'lm', se=FALSE, formula=formulas[[1]])+
  geom_smooth(aes(color='p=3'),
              method = 'lm', se=FALSE, formula=formulas[[3]])+
  geom_smooth(aes(color='p=30'),
              method = 'lm', se=FALSE, formula=formulas[[30]])+
  coord_cartesian(ylim=c(40,90))+
```

```
  theme_minimal()+
  labs(color='Valor de P',
       x='PIB per capita normalizado',
       y='Expectativa de Vida',
       title='Comparação de modelos para p=1, 3 e 30')
```



Item 5

```
# Criando tabela para armazenar valores preditos

predict_new <- tibble(
  y = rep(df$y, 3),
  pred = rep(0, 3*nrow(df)),
  p_fator = c(rep('p = 1', nrow(df)),
              rep('p = 3', nrow(df)),
              rep('p = 30', nrow(df)))
)

model <- NULL

for (ii in 1:nrow(df)) {

  #p=1

  model <- lm(formulas[[1]], data=df[-ii,]) # Ajuste do modelo sem obs. ii

  predict_new[ii,2] <- predict(model, df[ii,]) # Predizendo obs ii
```

```r
  #p=3

  model <- lm(formulas[[3]], data=df[-ii,]) # Ajuste do modelo sem obs. ii

  predict_new[ii+211,2] <- predict(model, df[ii,]) # Predizendo obs ii

  #p=30

  model <- lm(formulas[[30]], data=df[-ii,]) # Ajuste do modelo sem obs ii

  predict_new[ii+422,2] <- predict(model, df[ii,]) # Predizendo obs ii

}

library(gridExtra)

# grafico para p = 1

p1 <- predict_new[1:211,] |>
  ggplot()+
  aes(x=pred,y=y)+
  geom_point(color="#F8766D")+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  ggtitle('p = 1')

# grafico para p = 3

p3 <- predict_new[212:422,] |>
  ggplot()+
  aes(x=pred,y=y)+
  geom_point(color="#00BA38")+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  ggtitle('p = 3')

p30 <- predict_new[423:633,] |>
  ggplot()+
  aes(x=pred,y=y)+
  geom_point(color="#619BFF")+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  ggtitle('p = 30')

todos <- predict_new |>
  ggplot()+
  aes(x=pred,y=y, color=p_fator)+
```

```
  geom_point()+
  geom_abline(intercept =0 , slope = 1, size=1)+
  coord_cartesian(xlim=c(50,100),
                  ylim=c(50,100))+
  theme_minimal()+
  ggtitle('Todos os valores de p')

grid.arrange(p1, p3, p30, todos, ncol=2)
```