

# Lista 3 - Tópicos em ML

Luben M. C. Cabezas

Reinaldo C. Anacleto

Primeiramente, importando bibliotecas que serão utilizadas:

```
# bibliotecas do R
library(ggplot2)
library(MASS)
library(BART)
library(purrr)
library(dplyr)
library(FNN)
library(quantregForest)
```

## Exercício 1

Importando o conjunto de dados de proteínas:

```
# importando os dados
protein_data <- read.csv("data/CASP.csv")
# visualizando numero de variaveis, observações e tipo de cada variavel
protein_data |> str()
```

```
'data.frame':  45730 obs. of  10 variables:
 $ RMSD: num  17.28 6.02 9.28 15.85 7.96 ...
 $ F1  : num  13558 6192 7726 8425 7461 ...
 $ F2  : num  4305 1623 1726 2368 1737 ...
 $ F3  : num  0.318 0.262 0.223 0.281 0.233 ...
 $ F4  : num  162.2 53.4 67.3 67.8 52.4 ...
 $ F5  : num  1872791 803447 1075648 1210472 1021020 ...
 $ F6  : num  215.4 87.2 81.8 109.4 94.5 ...
 $ F7  : num  4288 3329 2981 3248 2814 ...
```

```
$ F8 : int 102 39 29 70 41 15 70 74 39 26 ...
$ F9 : num 27 38.5 38.8 39.1 39.9 ...
```

Dividindo o conjunto em treino, calibração e teste. O conjunto de calibração será usado apenas para os métodos conformal:

```
set.seed(45)
# Dividindo em treino e teste
n_train_val <- floor(0.9 * nrow(protein_data))
train_val_ind <- sample(seq_len(nrow(protein_data)), n_train_val)
train_val_data <- protein_data[train_val_ind, ]
test_data <- protein_data[-train_val_ind, ]

# dividindo em treino e calibração
n_cal <- floor(0.5 * nrow(train_val_data))
train_cal_ind <- sample(seq_len(nrow(train_val_data)), n_cal)
calib_data <- train_val_data[train_cal_ind, ]
train_data <- train_val_data[-train_cal_ind, ]

# Transformando em matrizes
# matriz de features
X_train <- train_data |> dplyr::select(-1) |> as.matrix()
X_calib <- calib_data |> dplyr::select(-1) |> as.matrix()
X_test <- test_data |> dplyr::select(-1) |> as.matrix()

# features padronizados para o scale
# media e desvio padrao
mean_train <- colMeans(X_train)
sd_train <- sqrt(apply(X_train, 2, var))
X_train_scale <- X_train |> scale(center = mean_train, scale = sd_train)
X_calib_scale <- X_calib |> scale(center = mean_train, scale = sd_train)
X_test_scale <- X_test |> scale(center = mean_train, scale = sd_train)

# resposta
y_train <- train_data |> pull(1)
y_calib <- calib_data |> pull(1)
y_test <- test_data |> pull(1)

# matrizes para intervalos preditivos bayesiano
X_train_b <- train_val_data |> dplyr::select(-1) |> as.matrix()
y_train_b <- train_val_data |> pull(1)
n_test <- nrow(X_test)
```

## Item 1

```
# tunando o KNN num grid de K
# k_grid <- round(seq(1,50,length.out=10))

#erro <- rep(NA,length(k_grid))

#for (ii in seq_along(k_grid)) {
#  predito <- knn.reg(train=X_train_scale,
#                    y=y_train,
#                    k=k_grid[ii])$pred
#  erro[ii] <- mean((predito-y_train)^2)
# }

best_k <- 10
#k_grid[which.min(erro)]

# predizendo para o conjunto de calibração
predito1 <- knn.reg(train=X_train_scale,
                   test=X_calib_scale,
                   y=y_train,
                   k=best_k)$pred

# predizendo para o conjunto de teste
predito2 <- knn.reg(train=X_train_scale,
                   test=X_test_scale,
                   y=y_train,
                   k=best_k)$pred

# calculando o conformal score
h_rs <- abs(predito1 - y_calib)

# estimando o corte
t1 <- quantile(h_rs, probs = 0.95)
cat("Corte estimado para o Regression Split: ", t1)
```

Corte estimado para o Regression Split: 9.205615

```
# Estimando o corte para o weighted e rho
predito31 <- knn.reg(train=X_train_scale,
```

```

        test=X_train_scale,
        y=y_train,
        k=best_k)$pred #  $E[Y|X]$  (Treinamento)

# desvio absoluto para estimar o rho
y_ro <- abs(predito31 - y_train)
# estimando rho para o conjunto de calibracao
predito3 <- knn.reg(train=X_train_scale,
                   test=X_calib_scale,
                   y=y_ro,
                   k=best_k)$pred #  $E[Z|X]$ 

# estimando quantil para o conjunto de calibracao
h_w <- (abs(predito1 - y_calib))/predito3
t2 <- quantile(h_w, probs = 0.95)

# estimando rho para o conjunto de teste
predito4 <- knn.reg(train=X_train_scale,
                   test=X_test_scale,
                   y=y_ro,
                   k=best_k)$pred #  $E[Z|X]$ 
cat("Corte estimado para o Weighted: ", t2)

```

Corte estimado para o Weighted: 4.06167

```

qrf_model <- quantregForest(
  x = X_train,
  y = y_train,
  n_threads = 4,
  ntree = 50,
  nodesize = 10,
  sampsize = 30
)

quantis <- c(0.05/2, 1 - (0.05/2))
pred_qrf <- predict(qrf_model, newdata =X_calib , what = quantis)

y_pred_05 <- pred_qrf[, 1] # Quantil de 2.5%
y_pred_95 <- pred_qrf[, 2] # Quantil de 97.5%

```

```

h_cqr <- pmax(y_pred_05 - y_calib, y_calib - y_pred_95)
t3<-quantile(h_cqr,probs = 0.95)

pred_qrf2 <- predict(qrf_model, newdata =X_test , what = quantis)
cat("Corte estimado para o CQR: ", t3)

```

Corte estimado para o CQR: 0.33796

```

# obtendo limitest inferiores e superiores de cada metodo
# reg-split
lower_bound_rs <- predito2 - t1
upper_bound_rs <- predito2 + t1
# weighted
lower_bound_w <- predito2 - (t2*predito4)
upper_bound_w <- predito2 + (t2*predito4)
# CQR
lower_bound_cqr <- pred_qrf2[, 1] - t3
upper_bound_cqr <- pred_qrf2[, 2] + t3

# obtendo tamanho medio dos intervalos
data.frame(
  "Região Preditiva" = c("Reg-split", "Weighted", "CQR"),
  "Largura" = c(
    mean(upper_bound_rs- lower_bound_rs),
    mean(upper_bound_w - lower_bound_w),
    mean(upper_bound_cqr - lower_bound_cqr)),
  "SE*2" = c(
    2*sqrt(var(upper_bound_rs - lower_bound_rs)/n_test),
    2*sqrt(var(upper_bound_w - lower_bound_w)/n_test),
    2*sqrt(var(upper_bound_cqr - lower_bound_cqr)/n_test)
  )
)

```

Região.Preditiva	Largura	SE.2
Reg-split	18.41123	0.0000000
Weighted	17.37975	0.3801737
CQR	17.88776	0.0613971

## Item 2

```
sum(y_test>=predito2-t1 & y_test<=predito2+t1)/length(predito2)
```

```
[1] 0.9553903
```

```
sum(y_test>=(predito2-predito4*t2) & y_test<=(predito2+predito4*t2))/length(predito2)
```

```
[1] 0.9536409
```

```
sum(y_test>=(pred_qrf2[,1]-t3) & y_test<=(pred_qrf2[,2]+t3))/length(y_test)
```

```
[1] 0.9547343
```

## Item 3

## Item 4

Para esse exemplo, utilizaremos o conjunto de dados de treino completo. Utilizaremos o pacote BART no R, obtendo uma amostra de 1000 da posteriori para obter as regiões preditivas para Y: Agora ajustando o BART:

```
# Set seed for reproducibility
set.seed(686)
# Fit BART model
post <- wbart(X_train_b, y_train_b, X_test, ndpost = 1000)
```

```
*****Into main of wbart
*****Data:
data:n,p,np: 41157, 9, 4573
y1,yn: 0.207192, 5.652192
x1,x[n*p]: 8300.110000, 34.444600
xp1,xp[np*p]: 13558.300000, 29.811800
*****Number of Trees: 200
*****Number of Cut Points: 100 ... 100
*****burn and ndpost: 100, 1000
```

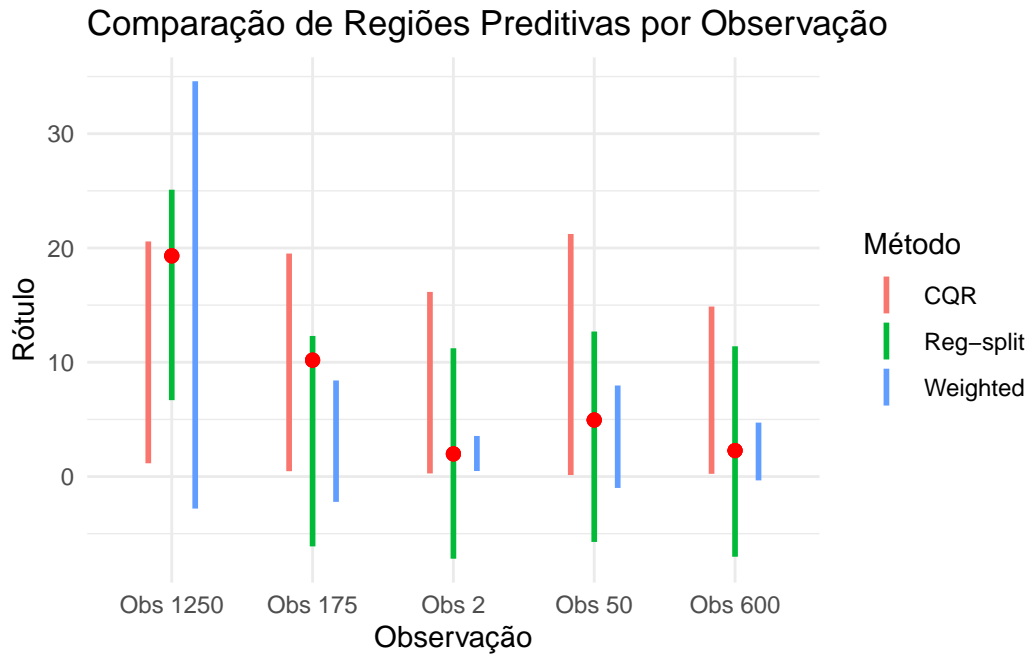


Figure 1: Intervalos preditivos conformais para as observações selecionadas. Pontos em vermelho representam o rótulo observado para cada observação.

```
*****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.371213,3.000000,5.253197
*****sigma: 5.193102
*****w (weights): 1.000000 ... 1.000000
*****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,9,0
*****nkeeptrain,nkeeptest,nkeeptestme,nkeepreedraws: 1000,1000,1000,1000
*****printevery: 100
*****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
```

MCMC

```
done 0 (out of 1100)
done 100 (out of 1100)
done 200 (out of 1100)
done 300 (out of 1100)
done 400 (out of 1100)
done 500 (out of 1100)
done 600 (out of 1100)
done 700 (out of 1100)
done 800 (out of 1100)
done 900 (out of 1100)
done 1000 (out of 1100)
```

```
time: 276s
check counts
trcnt,tecnt,temecnt,treedrawscnt: 1000,1000,1000,1000
```

Obtemos agora uma região preditiva simétrica e a região preditiva quantílica:

```
alpha <- 0.05
# Obtendo os lower e upper bounds da região simétrica para o conjunto de teste
means <- post$yhat.test.mean
std_dev <- sqrt(mean(post$sigma)^2 + apply(post$yhat.test, 2, sd)^2)
lower_bound_sym <- means - 1.96 * std_dev
upper_bound_sym <- means + 1.96 * std_dev

sigmas <- post$sigma[101:1100]
# Obtendo os lower e upper bounds da região baseada nos quantis
# encontrando quantil inferior usando monte carlo
y_new <- 1:length(y_test) |>
map(function(.x){
  y_sim <- post$yhat.test[, .x]
  return(
    rnorm(length(y_sim), mean = y_sim, sd = sigmas)
  )
}) |> unlist() |> matrix(nrow = 1000)

# grid em y
lower_bound_q <- apply(y_new, 2, quantile, probs = alpha/2)
upper_bound_q <- apply(y_new, 2, quantile, probs = (1 - alpha/2))
```

Tendo ambas as regiões, podemos a seguir calcular a cobertura empírica nesses casos:

```
cover_sym <- ((lower_bound_sym <= y_test) &
  (upper_bound_sym >= y_test)) |>
mean()

cover_q <- ((lower_bound_q <= y_test) &
  (upper_bound_q >= y_test)) |>
mean()

data.frame("Região Preditiva" = c("Simétrica", "Quantílica"),
  "Cobertura empírica" = c(cover_sym, cover_q) |> round(4))
```



Região.Preditiva	Cobertura.empírica
Simétrica	0.9488
Quantílica	0.9458

Percebe-se que ambas regiões são razoavelmente próximas da cobertura nominal 0.95, tendo porém uma leve sub-cobertura, principalmente a região quantílica que está um pouco mais distante do nível nominal que o intervalo simétrico. Podemos também observar pela Figura 2 as regiões preditivas estimadas no conjunto de teste para 5 diferentes valores.

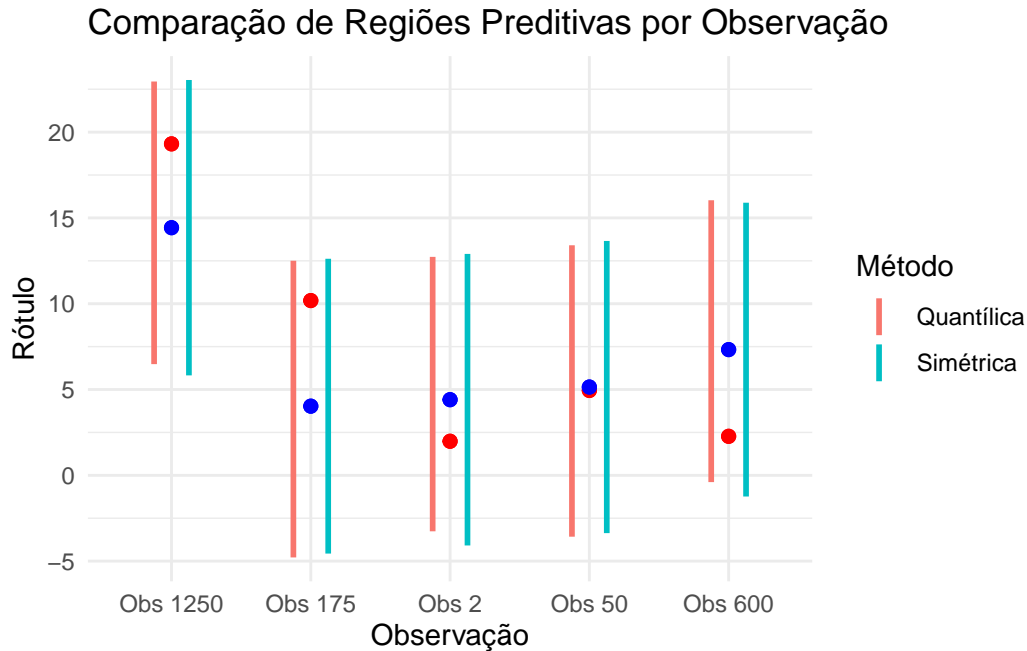


Figure 2: Intervalos preditivos bayesianos para as observações selecionadas. Pontos em vermelho representam o rótulo observado para cada observação enquanto pontos azuis representam a média a posteriori em cada caso.

Primeiramente visualiza-se nesses casos em particular que todos os intervalos preditivos contém o valor observado de  $Y$ . Percebemos também que há poucas diferenças entre os tipos de regiões, com ambas tendo tamanhos similares e razoavelmente largos. Além disso, as regiões quantílicas são visualmente apenas um pouco assimétricas em torno da média e um pouco mais curtas que as regiões simétricas. Podemos comparar adicionalmente a largura média das regiões através da seguinte tabela:

```
data.frame(
  "Região Preditiva" = c("Simétrica", "Quantílica"),
  "Largura" = c(
```

```

    mean(upper_bound_sym - lower_bound_sym),
    mean(upper_bound_q - lower_bound_q)),
  "SE*2" = c(
    2*sqrt(var(upper_bound_sym - lower_bound_sym)/n_test),
    2*sqrt(var(upper_bound_q - lower_bound_q)/n_test)
  )
)

```

Região.Preditiva	Largura	SE.2
Simétrica	17.10048	0.0062037
Quantílica	16.88489	0.0160593

Concluindo que de fato, as regiões quantílica são em média um pouco menos largas que as regiões simétricas.