BIOS 6312: Modern Regression Analysis

Andrew J. Spieker, Ph.D.

Assistant Professor of Biostatistics
Vanderbilt University Medical Center

Set 6 supplementary slides for `R` enthusiasts

Version: 03/11/2021

Examples for Set 6

Examples for R enthusiasts:

- Predicting DSST with LASSO (Slide 617)
- Natural cubic spline for height and FEV (Slide 651)
- AUC for diabetes (Slide 665)

Examples for Set 6

Examples for R enthusiasts:

- **Predicting DSST with LASSO (Slide 617)**
- Natural cubic spline for height and FEV (Slide 651)
- AUC for diabetes (Slide 665)

DSST with LASSO

Reading in the MRI data:

- Read in data:

```
mri.data <- read.csv("mri.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

DSST with LASSO

Elastic net for GLMs:

- Many use `glmnet` to perform penalized regression.

```
library("glmnet")
```

DSST with LASSO

Split the data:

- Split data into training and test set:

```
N <- dim(mri.data)[1]
set.seed(111)
idx <- sample(1:N, size = floor(N/2), replace = FALSE)
mri.train <- mri.data[idx,]
mri.test <- mri.data[-idx,]
```

DSST with LASSO

Split the data:

- Formulate training set:

```
XY.train <- model.matrix(~ age + male + factor(race) + weight +
                           height + packyrs + yrsquit + alcoh +
                           physact + factor(chf) + diabetes +
                           atrophy + factor(numinf) + volinf + dsst,
                           data = mri.train)

Y.train <- as.numeric(XY.train[,dim(XY.train)[2]])
X.train <- XY.train[,2:(dim(XY.train)[2] - 1)]
```

DSST with LASSO

Cross-validation:

- Cross-validation on training data (also extract the optimal penalty).
- Note that $\alpha = 0$ corresponds to LASSO.

```
set.seed(4)
zz <- cv.glmnet(x = X.train, y = Y.train,
                family = "gaussian",
                alpha = 1,
                nfolds = 10,
                standardize = TRUE)

lmin <- zz$lambda.min
```

DSST with LASSO

Training error:

- Generate and display training error. This will not match the Stata results because the seeds are different.

```
Y.pred.train <- predict(zz, newx = X.train, s = lmin)
rmse.train <- sqrt(mean((Y.pred.train - Y.train)^2))
```

```
> rmse.train
[1] 10.1
```

DSST with LASSO

Test error:

- Generate test data; generate and display test error. This will not match the Stata results because the seeds are different.

```
XY.test <- matrix(model.matrix(~ age + male + factor(race) + weight +
                                 height + packyrs + yrsquit + alcoh +
                                 physact + factor(chf) + diabetes +
                                 genhlth + ldl + alb + crt + plt + sbp +
                                 aai + fev + atrophy + factor(numinf) +
                                 volinf + dsst, data = mri.test), ncol = 30)

Y.test <- as.numeric(XY.test[,dim(XY.test)[2]])
X.test <- XY.test[,2:(dim(XY.test)[2] - 1)]

Y.pred.test <- predict(zz, newx = X.test, s = lmin)

rmse.test <- sqrt(mean((Y.pred.test - Y.test)^2))

> rmse.test
[1] 11.9
```

Examples for Set 6

Examples for R enthusiasts:

- *Predicting DSST with LASSO (Slide 617)*
- **Natural cubic spline for height and FEV (Slide 651)**
- AUC for diabetes (Slide 665)

Height and FEV

Reading in the FEV data:

- Read in data:

```
fev.data <- read.csv("fev.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

Height and FEV

Natural cubic splines:

- You're free to use my function:

```
ncs.R <- function(x, knots, stub = "b")
{
  N <- length(x); P <- length(knots)
  zP <- knots[P]; zP.1 <- knots[P - 1]
  bmat <- matrix(0, nrow = N, ncol = P - 1)
  bmat[,1] <- as.numeric(x)
  nms <- c(paste(stub, 1, sep = ""))
  for (j in 1:(P - 2))
  {
    zp <- knots[j]
    dp.num <- pmax(0, (x - zp)^3) - pmax(0, (x - zP)^3)
    dp <- dp.num/(zP - zp)
    dP.1.num <- pmax(0, (x - zP.1)^3) - pmax(0, (x - zP)^3)
    dP.1 <- dP.1.num/(zP - zP.1)
    bmat[,j + 1] <- dp - dP.1
    nms <- c(nms, paste(stub, j + 1, sep = ""))
  }
  bmat <- data.frame(cbind(1, bmat))
  names(bmat) <- c(paste(stub, 0, sep = ""), nms)
  return(bmat)
}
```

Height and FEV

Parameters for R function: `ncs.R`

- `x`: variable on which you seek to create spline basis functions.
- `knots`: values of the knots.
- `stub`: stub for variable name (helpful if you want to create basis functions for more than one variable and still be able to distinguish between them later).

Height and FEV

Natural cubic splines:

- Create matrix of basis functions (appending original variables for convenience of coding).

```
hmat <- ncs.R(fev.data$height, knots = c(50, 60, 70))
hmat$FEV <- fev.data$fev
hmat$height <- fev.data$height
```

Height and FEV

Natural cubic splines:

- Regression with basis splines included; extract coefficients.

```
zz.ncs <- lm(FEV ~ b1 + b2, data = hmat)
cfs <- coef(zz.ncs)
```

Height and FEV

Adding natural cubic splines to a plot:

- You're free to use my function:

```
line.ncs <- function(range, knots, coefs, col = "blue", lwd = 2, lty = 1)
{
  N <- length(range); P <- length(knots)
  zP <- knots[P]; zP.1 <- knots[P - 1]
  bmat <- matrix(0, nrow = N, ncol = P - 1)
  bmat[,1] <- as.numeric(range)
  for (j in 1:(P - 2))
  {
    zp <- knots[j]
    dp.num <- pmax(0, (range - zp)^3) - pmax(0, (range - zP)^3)
    dp <- dp.num/(zP - zp)
    dP.1.num <- pmax(0, (range - zP.1)^3) - pmax(0, (range - zP)^3)
    dP.1 <- dP.1.num/(zP - zP.1)
    bmat[,j + 1] <- dp - dP.1
  }
  bmat <- cbind(1, bmat)
  prdct <- bmat %*% coefs
  lines(range, prdct, col = col, lwd = lwd, lty = lty)
}
```

Height and FEV

Parameters for R function: `line.ncs`

- `range`: all x-values you want to use to generate the curve.
- `knots`: values of the knots (must be same as for basis spline generation).
- `coefs`: coefficients from regression model

Height and FEV

Natural cubic splines for height and FEV:

- Plot data and add splines (should match plot from notes).

```
plot(hmat$height, hmat$FEV, frame.plot = FALSE,
     xlab = "Height (in)", ylab = "FEV (L)",
     ylim = c(0,6), cex = 0.75, pch = 20,
     col = "gray40", main = "Natural cubic spline")

rng <- seq(min(fev.data$height),
           max(fev.data$height), 0.1)
line.ncs(rng, knots = c(50, 60, 70),
         coefs = cfs, lwd = 3)
```

Examples for Set 6

Examples for R enthusiasts:

- *Predicting DSST with LASSO (Slide 617)*
- *Natural cubic spline for height and FEV (Slide 651)*
- **AUC for diabetes (Slide 665)**

Predicting diabetes in MRI data

Reading in the MRI data:

- Read in data:

```
mri.data <- read.csv("mri.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

Predicting diabetes in MRI data

Split the data:

- Split data into training and test set; append indices:

```
N <- dim(mri.data)[1]
set.seed(111)
idx <- sample(1:N, size = floor(N/2), replace = FALSE)
mri.data$split <- 0
mri.data$split[idx] <- 1
mri.train <- mri.data[idx,]
mri.test <- mri.data[-idx,]
```

Predicting diabetes in MRI data

Train the model:

- Logistic regression model in training set:

```
zz.diab <- glm(diabetes ~ age + male + packyrs +
                  physact + weight + height,
                  family = binomial(link = "logit"),
                  data = mri.train)
```

Predicting diabetes in MRI data

Generate predictions:

- Create full data set for purposes of generating predictions:

```
full.mat <- data.frame(model.matrix(~ age + male + packyrs + physact +
                                      weight + height + diabetes + split,
                                      data = mri.data))
```

Predicting diabetes in MRI data

AUC: Scaled Mann-Whitney *U*-statistic

- You're welcome to use my function, `lroc.R`:

```
lroc.R <- function(Y, Y.hat, split)
{
  Y.hat.00 <- Y.hat[Y == 0 & split == 0]
  Y.hat.10 <- Y.hat[Y == 1 & split == 0]
  Y.hat.01 <- Y.hat[Y == 0 & split == 1]
  Y.hat.11 <- Y.hat[Y == 1 & split == 1]
  N00 <- length(Y.hat.00); N10 <- length(Y.hat.10)
  N01 <- length(Y.hat.01); N11 <- length(Y.hat.11)
  U0 <- wilcox.test(Y.hat.00, Y.hat.10)$statistic
  U1 <- wilcox.test(Y.hat.01, Y.hat.11)$statistic
  AUC0 <- U0/(N00 * N10)
  if (AUC0 < 0.5) {AUC0 <- 1 - AUC0}
  AUC1 <- U1/(N01 * N11)
  if (AUC1 < 0.5) {AUC1 <- 1 - AUC1}
  return(c(Train.AUC = as.numeric(AUC0),
           Test.AUC = as.numeric(AUC1)))
}
```

Predicting diabetes in MRI data

Parameters for R function: `lroc.R`

- `Y`: true outcome.
- `Y.hat`: predicted value.
- `split`: indicator of training (0) and test (1) set.

Predicting diabetes in MRI data

Predictive ability:

- Training and test AUC. Will not match Stata output as the seed is different.

```
lroc.R(Y = full.mat$diabetes,
      Y.hat = predict(zz.diab, type = "response", newdata = full.mat),
      split = full.mat$split)
```

```
## Train.AUC  Test.AUC
##     0.580     0.681
```