BB503/BB602 - R Training - Week XIII

Ege Ulgen

Simulated Data

Consider the following linear regression model:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{1i} X_{2i} + Z_i$$

We'll simulate Y values for n = 1000 subjects using the following configuration:

- $\beta_0 = 1$, $\beta_1 = 0.8$, $\beta_2 = -1$, $\beta_3 = 0.3$
- $X_{1i} \sim Normal(20, 5)$
- $X_{2i} \sim Bernoullu(p = 0.45)$
- $Z_i \sim N(0, 0.5)$

```r
n <- 1000

B0 <- 1
B1 <- 0.8
B2 <- -1
B3 <- 0.3

set.seed(123)
X1 <- rnorm(n, mean = 20, sd = sqrt(5))
X2 <- sample(c(0, 1), size = n, replace = TRUE)
Z <- rnorm(n, mean = 0, sd = sqrt(0.5))

Y_sim <- B0 + B1 * X1 + B2 * X2 + B3 * X1 * X2 + Z

simulated_df <- data.frame(Y = Y_sim, X1 = X1, X2 = X2)
```

We'll compare the estimated mean squared errors (MSEs) for two models (below) using 3 different approaches:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{1i} X_{2i} + Z_i$$

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + Z_i$$

The holdout method

```r
set.seed(123)
train_idx <- sample(1:nrow(simulated_df), nrow(simulated_df) * .8)

train_df <- simulated_df[train_idx, ]
test_df <- simulated_df[-train_idx, ]
```

```r
# fit model on training
fit1 <- lm(Y ~ X1 + X2, data = train_df)
fit2 <- lm(Y ~ X1 + X2 + X1 * X2, data = train_df)

# predict on test
preds1 <- predict(fit1, newdata = test_df)
preds2 <- predict(fit2, newdata = test_df)
# calculate MSE
MSE_holdout1 <- mean((test_df$Y - preds1) ^ 2)
MSE_holdout2 <- mean((test_df$Y - preds2) ^ 2)
```

K-Fold Cross Validation

```r
### 10-fold CV
K <- 10
set.seed(123)
# shuffle
idx <- sample(1:n, n)
kfold_df <- simulated_df[idx, ]
kfold_df <- cbind(kfold_df, fold = rep(1:K, each = 1000 / K))

all_MSE_kfold1 <- all_MSE_kfold2 <- c()
for (i in 1:K) {
    train_df <- kfold_df[kfold_df$fold != i, ]
    test_df <- kfold_df[kfold_df$fold == i, ]

    # fit model on training
    fit1 <- lm(Y ~ X1 + X2, data = train_df)
    fit2 <- lm(Y ~ X1 + X2 + X1 * X2, data = train_df)

    # predict on test
    preds1 <- predict(fit1, newdata = test_df)
    preds2 <- predict(fit2, newdata = test_df)
    # calculate MSE
    MSE1 <- mean((test_df$Y - preds1)^2)
    MSE2 <- mean((test_df$Y - preds2)^2)
    all_MSE_kfold1 <- c(all_MSE_kfold1, MSE1)
    all_MSE_kfold2 <- c(all_MSE_kfold2, MSE2)
}
MSE_kfold1 <- mean(all_MSE_kfold1)
MSE_kfold2 <- mean(all_MSE_kfold2)
```

Leave-one-out Cross Validation (LOOCV)

```r
### =1000-fold CV
K <- nrow(simulated_df)
set.seed(123)
# shuffle
idx <- sample(1:n, n)
LOO_df <- simulated_df[idx, ]
LOO_df <- cbind(LOO_df, fold = rep(1:K, each = 1000 / K))
```

```r
all_MSE_LOO1 <- all_MSE_LOO2 <- c()
for (i in 1:K) {
    train_df <- LOO_df[LOO_df$fold != i, ]
    test_df <- LOO_df[LOO_df$fold == i, ]

    # fit model on training
    fit1 <- lm(Y ~ X1 + X2, data = train_df)
    fit2 <- lm(Y ~ X1 + X2 + X1 * X2, data = train_df)

    # predict on test
    preds1 <- predict(fit1, newdata = test_df)
    preds2 <- predict(fit2, newdata = test_df)
    # calculate MSE
    MSE1 <- mean((test_df$Y - preds1) ^ 2)
    MSE2 <- mean((test_df$Y - preds2) ^ 2)
    all_MSE_LOO1 <- c(all_MSE_LOO1, MSE1)
    all_MSE_LOO2 <- c(all_MSE_LOO2, MSE2)
}
MSE_LOO1 <- mean(all_MSE_LOO1)
MSE_LOO2 <- mean(all_MSE_LOO2)
```

Final Comparison

```r
# Model 1
MSE_holdout1; MSE_kfold1; MSE_LOO1
```

```
## [1] 0.58105

## [1] 0.57947

## [1] 0.5792
```

```r
# Model 2
MSE_holdout2; MSE_kfold2; MSE_LOO2
```

```
## [1] 0.48435

## [1] 0.48459

## [1] 0.48606
```

The bootstrap

```r
### Bootstrap estimate of correlation betwen PSA and vol, median PSA, median vol
prca_df <- read.csv("../data/prostate_cancer.csv")

library(boot)

boot_func <- function(data, indices, cor.type = "spearman"){
  dt <- data[indices,]
  res <- c(cor(dt[, 1], dt[,2], method=cor.type),
           median(dt[,1]),
           median(dt[,2]))
  return(res)
}
```

```r
bs_res <- boot(prca_df, boot_func, R = 10000)
bs_res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = prca_df, statistic = boot_func, R = 10000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1*  0.70019 -0.0054947     0.05867
## t2* 13.33000 -0.1033030     1.79052
## t3*  4.26310 -0.1045216     0.52661
```

```r
# bootstrap realizations
head(bs_res$t)
```

```
##         [,1]   [,2]   [,3]
## [1,] 0.68348 13.066 3.1899
## [2,] 0.64957 14.585 4.2207
## [3,] 0.81697 13.330 3.7062
## [4,] 0.69220 13.330 4.7588
## [5,] 0.57132 14.732 4.6646
## [6,] 0.77362 16.610 4.3929
```

```r
# bias
colMeans(bs_res$t)
```

```
## [1]  0.69469 13.22670  4.15858
```

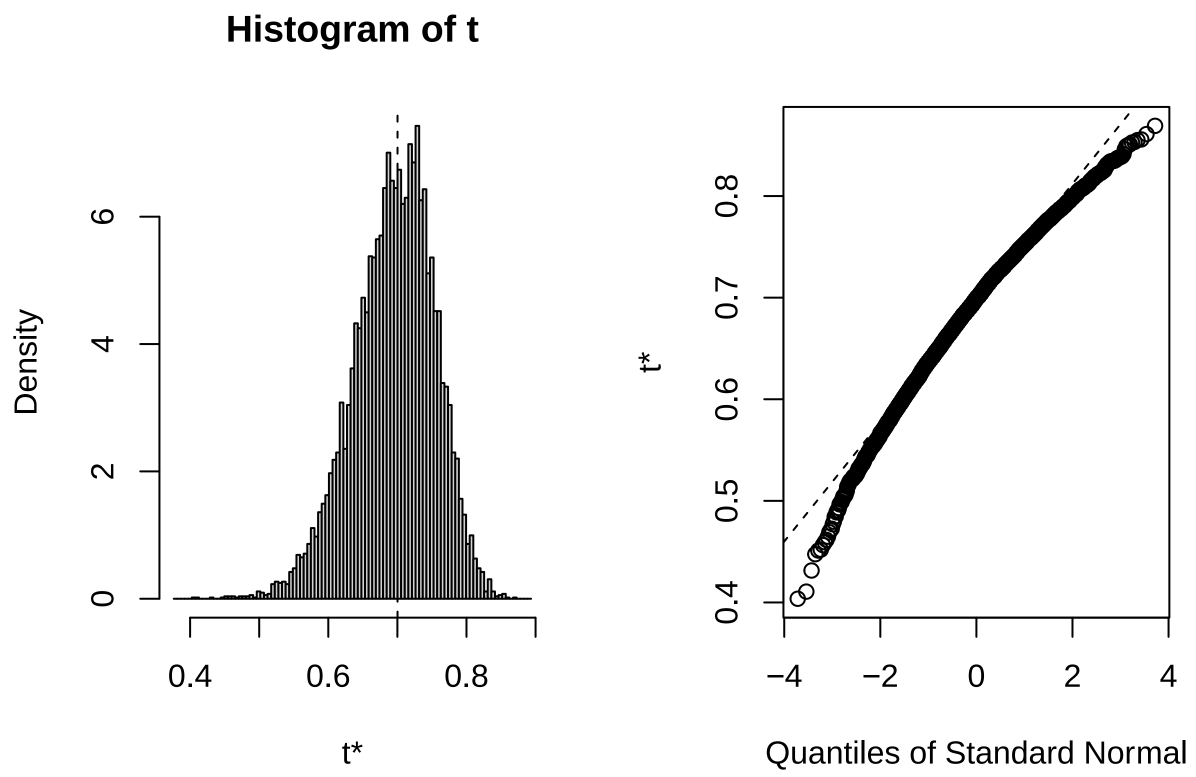
```r
colMeans(bs_res$t) - bs_res$t0
```

```
## [1] -0.0054947 -0.1033030 -0.1045216
```

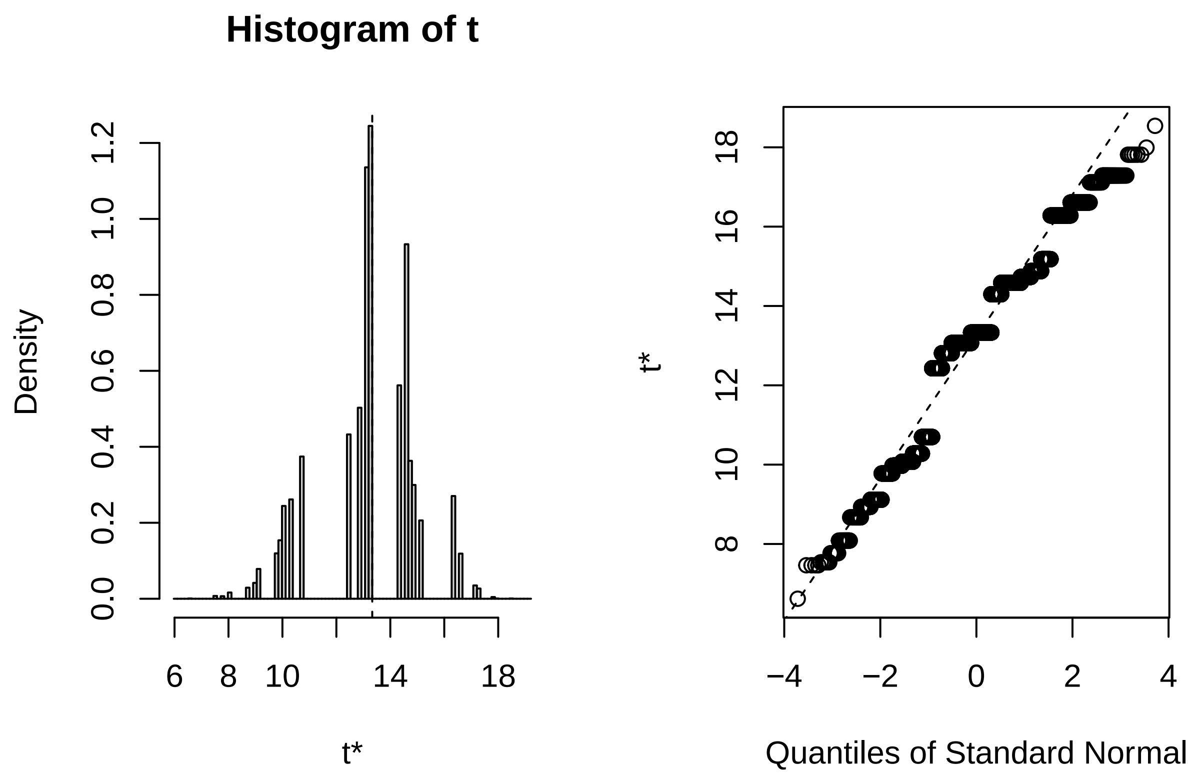
```r
# SE
apply(bs_res$t, 2, sd)
```

```
## [1] 0.05867 1.79052 0.52661
```

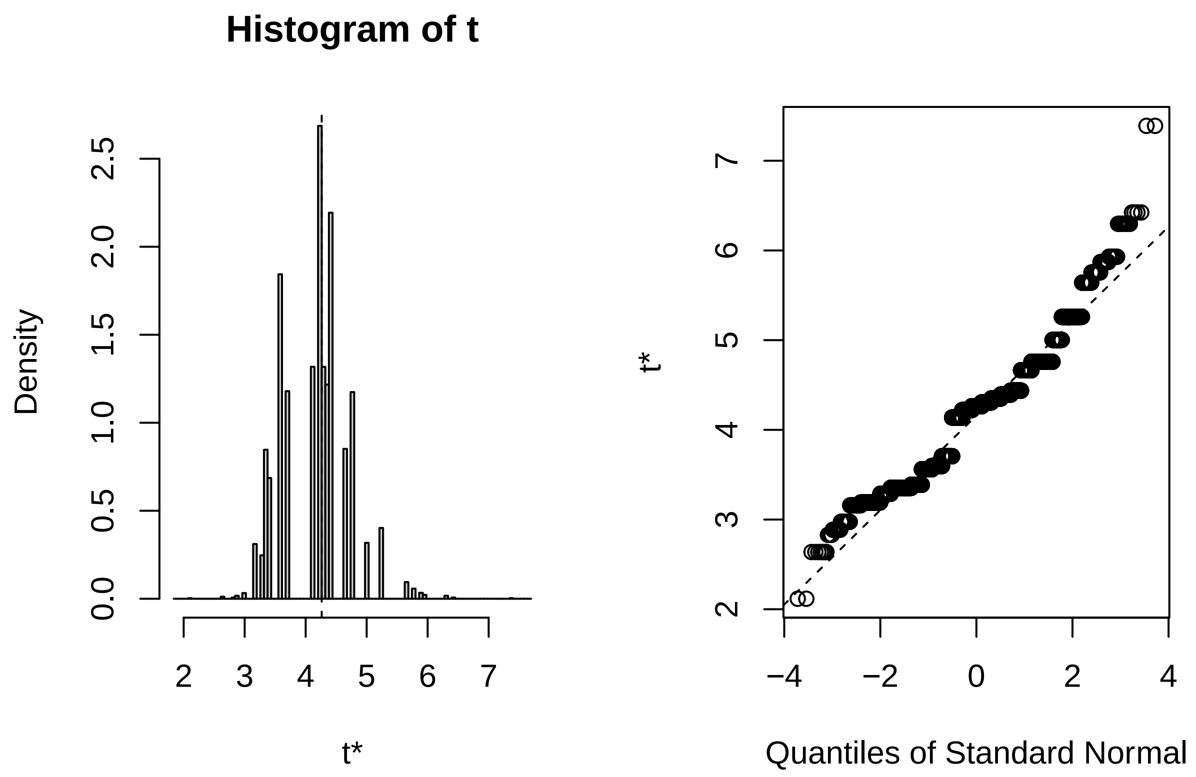
```r
## distribution of realizations
plot(bs_res, index = 1)
```



```
plot(bs_res, index = 2)
```



```r
plot(bs_res, index = 3)
```



```
?boot.ci
boot.ci(bs_res, index = 1, type = "basic")
```

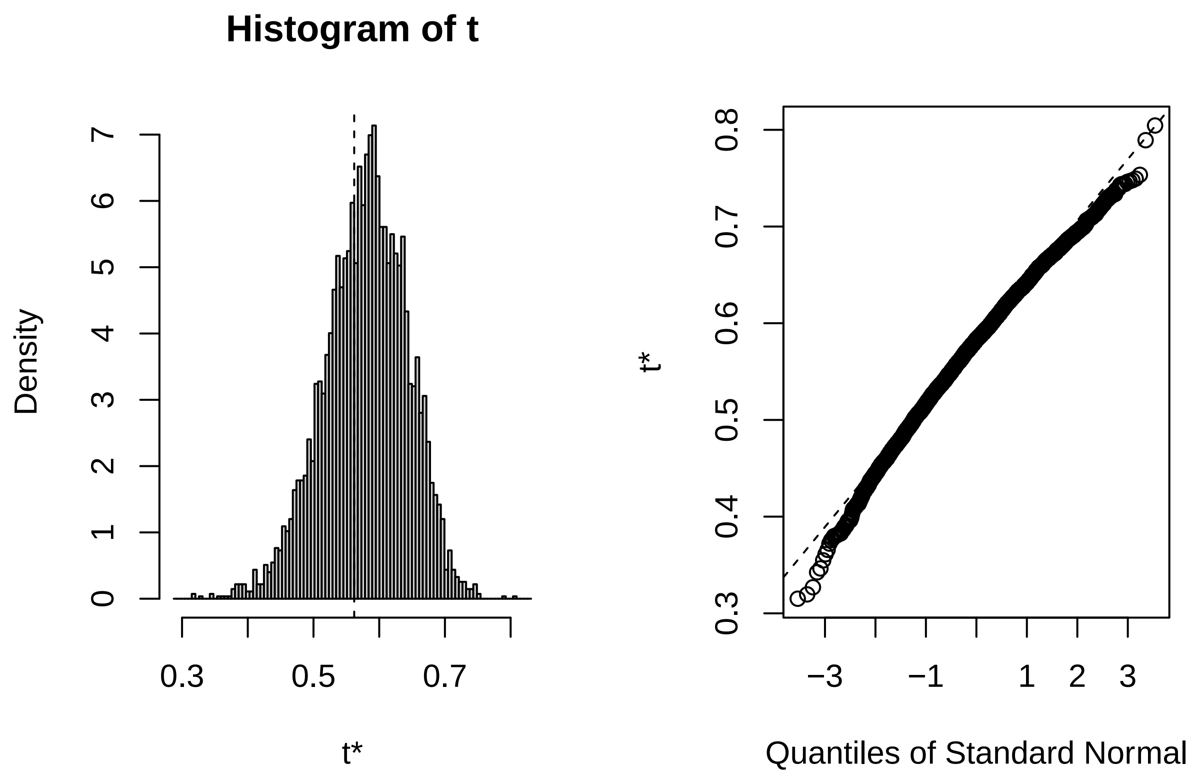
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bs_res, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%   ( 0.6036,  0.8321 )
## Calculations and Intervals on Original Scale
```

```
### Bootstrap estimate of adjusted R squared for the multiple regression model from week 11
prca_df$PSA <- log(prca_df$PSA)

prca_df$Gleason <- as.factor(prca_df$Gleason)
prca_df$invasion <- as.factor(prca_df$invasion)

boot_func2 <- function(data, indices, cor.type = "spearman"){
    dt <- data[indices,]
    tmp <- lm(dt[, 1] ~ dt[, 2] + dt[, 5] + dt[, 6] + dt[, 8])
    tmp <- summary(tmp)
    res <- tmp$adj.r.squared
    return(res)
}
```

```r
bs_res <- boot(prca_df, boot_func2, R = 5000)
bs_res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = prca_df, statistic = boot_func2, R = 5000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*  0.56194 0.017505    0.063359
```

```r
## distribution of realizations
plot(bs_res)
```



```r
boot.ci(bs_res, index = 1)
```

```
## Warning in boot.ci(bs_res, index = 1): bootstrap variances needed for
## studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
```

```
## boot.ci(boot.out = bs_res, index = 1)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.4203,  0.6686 )   ( 0.4311,  0.6764 )
##
## Level     Percentile            BCa
## 95%   ( 0.4475,  0.6928 )   ( 0.3904,  0.6620 )
## Calculations and Intervals on Original Scale
```