Solution to Series 8

1. a) We first generate the data according to the provided code:

```
> # for replication
> set.seed(1)
> n <- 30
> p <- 50
> # relevant covariate
> x_true <- sample(c(0:1),size = n,replace = T)
> # noise covariates
> x <- matrix(sample(c(0:1),size=n*p,replace = T),ncol=p,nrow=n)
> # combination of the two
> x <- cbind(x_true, x)
> # response
> y <- ifelse(x[,1]==0, 0, sample(c(0:1), size = n, replace = T))
```

b) We then compute all chi-squared tests of marginal association between one of the covariates and the response y to obtain p-values:

```
> # this is has nothing to do with the code, just
> # to remove warnigns
> opt <- options("warn")
> options(warn = -1)
> # individual level
> alpha <- 0.05
> # Bonferroni level
> alpha_Bonferroni <- alpha/(p+1)
> # obtain pvalues
> # note here that we use the apply function in order to simplify the code
> pvalues <- apply(x, 2,function(i) chisq.test(x = y, y = i)$p.value)
> # Bonferroni correction
> rejection_Bonferroni <- pvalues < alpha_Bonferroni
> table(rejection_Bonferroni)

rejection_Bonferroni
FALSE
   51
```

We see from this output that no association is declared with Bonferroni correction.

c) In the second case we perform the Westfall Young permutation procedure on 1000 replicates, this gives us:

```
> # number of simulations
> nb <- 1000
> # vector storing the minimum pvalues
> min_pvalues <- numeric(nb)
> # main loop
> for(j in 1:nb) {
   y_perm <- y[sample(1:length(y), size = length(y), replace = F)]
   min_pvalues[j] <- min(apply(x, 2, function(i) chisq.test(x = y_perm,y = i)$p.value))
 }
> # getting the delta and doing the test
> delta <- quantile(min_pvalues, probs = 0.05, type = 1)
> rejection_WY <- pvalues < delta
> table(rejection_WY)

rejection_WY
FALSE  TRUE
   50     1
```

```
> which(rejection_WY)
x_true
     1
> # this is has nothing to do with the code, just
> # to remove warnigns
> options(warn = as.numeric(opt))
```

According to the Westfall Young procedure, we find one significant p-value corresponding to the single covariate with a "true" effect.

2. **a)** First we load the dataset, remove missing values and create folds:

```
> # reproducibility
> set.seed(1)
> # libs
> library(ISLR)
> library(leaps)
> #- split the data into k-fold
> Hitters <- na.omit(Hitters)
> k <- 10
> folds <- sample(cut(1:nrow(Hitters), breaks = 10, labels = F),nrow(Hitters), replace=F)
```

We will also use the following predict method:

```
>  predict.regsubsets <- function(object , newdata ,id ,...) {
     form  <- as.formula(object$call[[2]])
     mat   <- model.matrix(form ,newdata)
     coefi <- coef(object ,id=id)
     xvars <- names(coefi)
     return(mat[,xvars]%*%coefi)
  }
```

b) Then we can do the nested cross-validation looping on the hold-out fold:

```
> # inner and outer fold summaries
> cv.errors.inner  <- matrix(NA, k, 19, dimnames = list(NULL , paste(1:19)))
> cv.errors.outer <-  matrix(NA, k, 4)
> colnames(cv.errors.outer) <- c("cv","adjr2","cp","bic")
> # outer loop
> for(out in 1:k) {

  # current training and test sets
  train <- Hitters[folds!=out,]
  test  <- Hitters[folds==out,]

  # inner cross-validation folds
  inner.folds <- cut(sample(c(1:nrow(Hitters))[folds!=out],
                            size = sum(folds!=out),
                            replace = F),
                     breaks = 9, labels = F)

  # inner cross-validation
  for(inner in 1:9) {

    best.fit <- regsubsets(Salary~.,data=train[inner.folds!=inner,],
                           nvmax=19)

    # iteration over the 19 models
    for(i in 1:19){
      pred <- predict(best.fit, train[inner.folds==inner,], id=i)
      cv.errors.inner[inner,i] <- mean((train$Salary[inner.folds==inner]-pred)^2)
    }
  }

    # result aggegation for inner cross-validation
    cv.inner <- apply(cv.errors.inner, 2, mean, na.rm=T)
    best.cv.model <- which.min(cv.inner)

    # get the prediction errors for the best model selected by cv
    best.fit <- regsubsets(Salary~.,data=train,
                           nvmax=19)
    pred <- predict(best.fit, test, id=best.cv.model)
    cv.errors.outer[out,1] <- mean((test$Salary-pred)^2)

    # case of direct selection by criteria
    best.fit.criterion <- regsubsets(Salary~.,data=Hitters[!folds%in%c(out),],
                        nvmax=19)

    # get the summary of the fits
    s <- summary(best.fit.criterion)

    # obtain the best models wrt a given criterion
    best.adjr2 <- which.max(s$adjr2)
    best.cp    <- which.min(s$cp)
    best.bic   <- which.min(s$bic)

    # get the prediction errors for these best models
    pred=predict(best.fit.criterion, Hitters[folds%in%c(out),], id=best.adjr2)
    cv.errors.outer[out,2] <-  mean(( Hitters$Salary[folds%in%c(out)]-pred)^2)

    pred=predict(best.fit.criterion, Hitters[folds%in%c(out),], id=best.cp)
    cv.errors.outer[out,3] <- mean(( Hitters$Salary[folds==out]-pred)^2)

    pred=predict(best.fit.criterion, Hitters[folds%in%c(out),], id=best.bic)
    cv.errors.outer[out,4] <- mean(( Hitters$Salary[folds%in%c(out)]-pred)^2)
 }
```

We can see which of the method is selected (the one with lowest expected test error):

```
> # mean
> apply(cv.errors.outer, 2, mean)
```

```
      cv    adjr2       cp      bic
124560.4 117774.3 121199.2 126561.7
```

We can also look at the standard errors (on the estimates of expected test MSE) to see if the differences may be significant or not

```
> # sd
> apply(cv.errors.outer, 2, sd)
```

```
      cv    adjr2       cp      bic
62463.12 54736.26 58158.47 59268.61
```

We see a huge variance of expected test MSE across the different folds for each method, clearly the results do not appear to be significant.

c) Finally we fit again the whole procedure to the dataset with the Mallow's Cp criterion:

```
> # fit on all the dataset
> best.fit <- regsubsets(Salary~.,data=Hitters, nvmax=19)
> # best model according to adjusted R-squared
> best.adjr2 <- which.min(s$cp)
> # coefficients of the best selected model
> coef(best.fit, id = best.adjr2)
```

```
 (Intercept)        AtBat         Hits        Walks
 162.5354420   -2.1686501    6.9180175    5.7732246
      CAtBat        CRuns         CRBI       CWalks
```

```
  -0.1300798    1.4082490    0.7743122   -0.8308264
   DivisionW      PutOuts      Assists
-112.3800575    0.2973726    0.2831680
```

3. **a)** First we load the needed packages and the dataset:

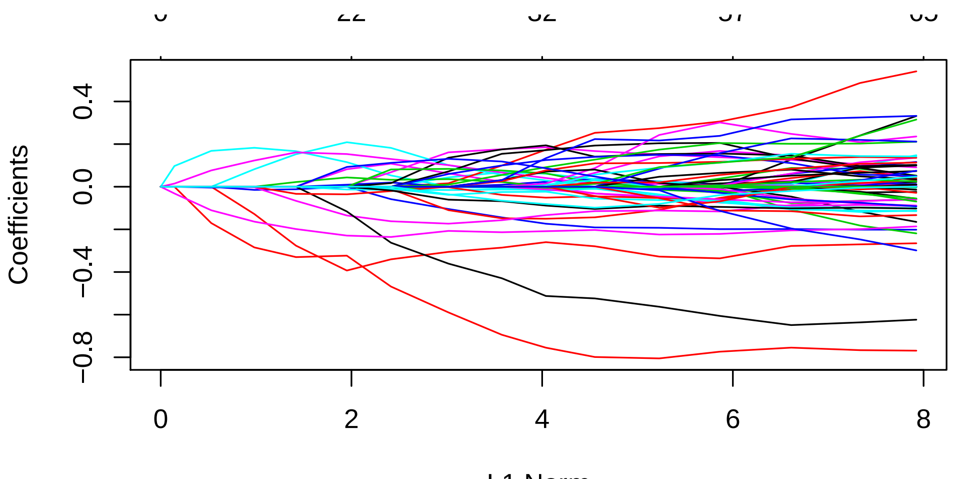
```
> # load the data
> library(hdi)
> data("riboflavin")
> # load the package for ridge and lasso regression
> library(glmnet)
```

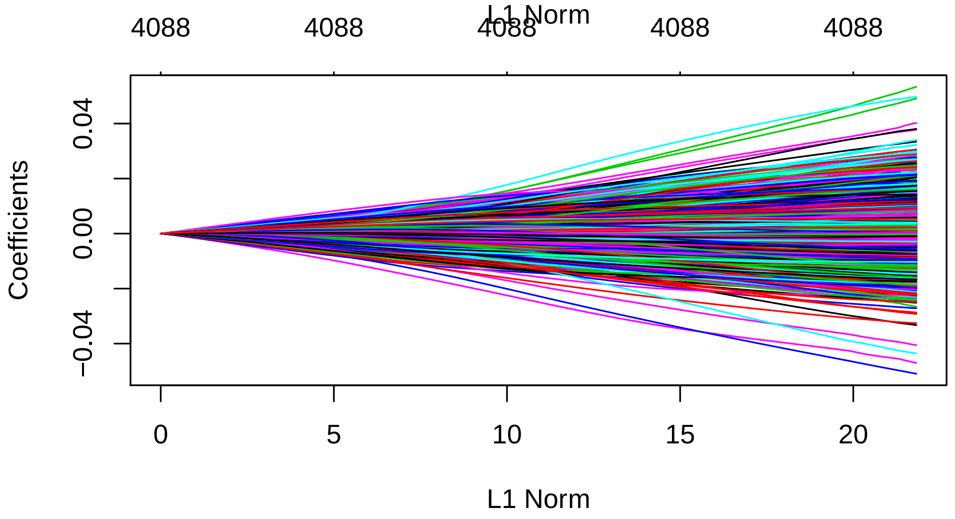
Then we can perform both ridge and lasso regression of y on x for the specified grid of lambda values:

```
> # grid of length values
> grid <- 10^seq(10, -2, length =100)
> # ridge regression (alpha = 0)
> ridge.mod <- glmnet(x      = riboflavin$x,
                     y      = riboflavin$y,
                     alpha  = 0,
                     lambda = grid)
> # lasso regression (alpha = 1)
> lasso.mod <- glmnet(x      = riboflavin$x,
                     y      = riboflavin$y,
                     alpha  = 1,
                     lambda = grid)
```

We can plot the coefficient path for each of these two regression models, i.e. the value of the estimated coefficient for each lambda value:

```
> # graphics parameter (two plots on the same pane)
> par(mfrow = c(2,1))
> # lasso path
> plot(lasso.mod)
> # ridge path
> plot(ridge.mod)
```





We clearly see a difference, for the ridge regresion there is no sparsity, but rather a shrinkage towards 0 for bigger lambda values, whereas for the lasso regression we clearly observe sparsity, i.e. some coefficients can be estimated to be $zero$ for some values of lambda.

b) As before, first we build the folds for the outer cross-validation, we will use 10-folds cross-validation

```
> # note here we propose an alternative for the nested folds where we keep the
> # same folds for outer and inner validation.
> set.seed(1)
> k <- 10
> folds <- cut(sample(1:k,
               nrow(riboflavin$x),
               replace=T), breaks = 10, labels = F)
> # vector of prediction errors
> error.lasso <- numeric(k)
> error.ridge <- numeric(k)
> for (i in 1:k) {

   # inner cross validation
   inner.folds <- folds[folds != i]

   # need to change levels of the folds
   # to always start from 1 for cv.glmnet
   inner.folds <- factor(inner.folds)
   levels(inner.folds) <- 1:(k-1)

   # run cross-validation for ridge (alpha = 0)
   optim.lambda.ridge <- cv.glmnet(x      = riboflavin$x[folds!=i,],
                                   y      = riboflavin$y[folds!=i],
                                   foldid = as.numeric(inner.folds),
                                      alpha  = 0)$lambda.min

   # run cross-validation for lasso (alpha = 1)
   optim.lambda.lasso <- cv.glmnet(x      = riboflavin$x[folds!=i,],
                                   y      = riboflavin$y[folds!=i],
                                   foldid = as.numeric(inner.folds),
                                   alpha  = 1)$lambda.min

   # ridge regression (alpha = 0)
   ridge.mod <- glmnet(x      = riboflavin$x[folds!=i,],
                       y      = riboflavin$y[folds!=i],
                       alpha  = 0,
                       lambda = grid)

   # lasso regression (alpha = 1)
   lasso.mod <- glmnet(x      = riboflavin$x[folds!=i,],
                       y      = riboflavin$y[folds!=i],
                       alpha  = 1,
                       lambda = grid)

   # prediction on hold-out fold
   lasso.pred <- predict(lasso.mod,
                         s = optim.lambda.lasso,
                         newx = riboflavin$x[folds==i,])
   ridge.pred <- predict(ridge.mod,
                         s = optim.lambda.ridge,
                         newx = riboflavin$x[folds==i,])

   error.ridge[i] <- mean((ridge.pred - riboflavin$y[folds==i])^2)
   error.lasso[i] <- mean((lasso.pred - riboflavin$y[folds==i])^2)
 }
```

We can finally compare the expected test MSE:

```
> # ridge
> mean(error.ridge)
[1] 0.2387204
> # lasso
> mean(error.lasso)
[1] 0.1910017
```

We see in this example that lasso regression tends to peform better in terms of expected test error according to the nested cross-validation.

c) At the end we fit again the best model (lasso) on the whole dataset:

```
> # selection of optimal lambda by cross validation on whole dataset
> optim.lambda.ridge <- cv.glmnet(x      = riboflavin$x,
                                  y      = riboflavin$y,
                                  nfolds = 10,
                                  alpha  = 1)$lambda.min
> # fit with the optimal lambda on whole dataset
> ridge.mod <- glmnet(x      = riboflavin$x,
                      y      = riboflavin$y,
                      alpha  = 1,
                      lambda = optim.lambda.lasso)
>
```

4. exercise 1:

a) best subset, since this is exhaustive search.

c) i. True (models are increasingly nested with the number of variables).
   ii. True (models are decreasingly nested with the number of variables).
   iii. False (not always true).
   iv. False (not always true).
   v. False (not always true).

exercise 3:

a) Steadily decrease.
b) Decrease initially, and then eventually start increasing in a U shape.
c) Steadily increase.
d) Steadily decrease.
e) Remain constant.