Midterm.R

niles

2022-02-06

```r
# Open data
input = read.csv("ais.csv")

# Remove NAs
input = input[complete.cases(input), ]

# Check for factors
(l <- sapply(input, function(x) is.factor(x)))
```

```
## Sport  Bfat   Sex    Ht    Wt   LBM   RCC   WCC    Hc    Hg  Ferr   BMI   SSF
##  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

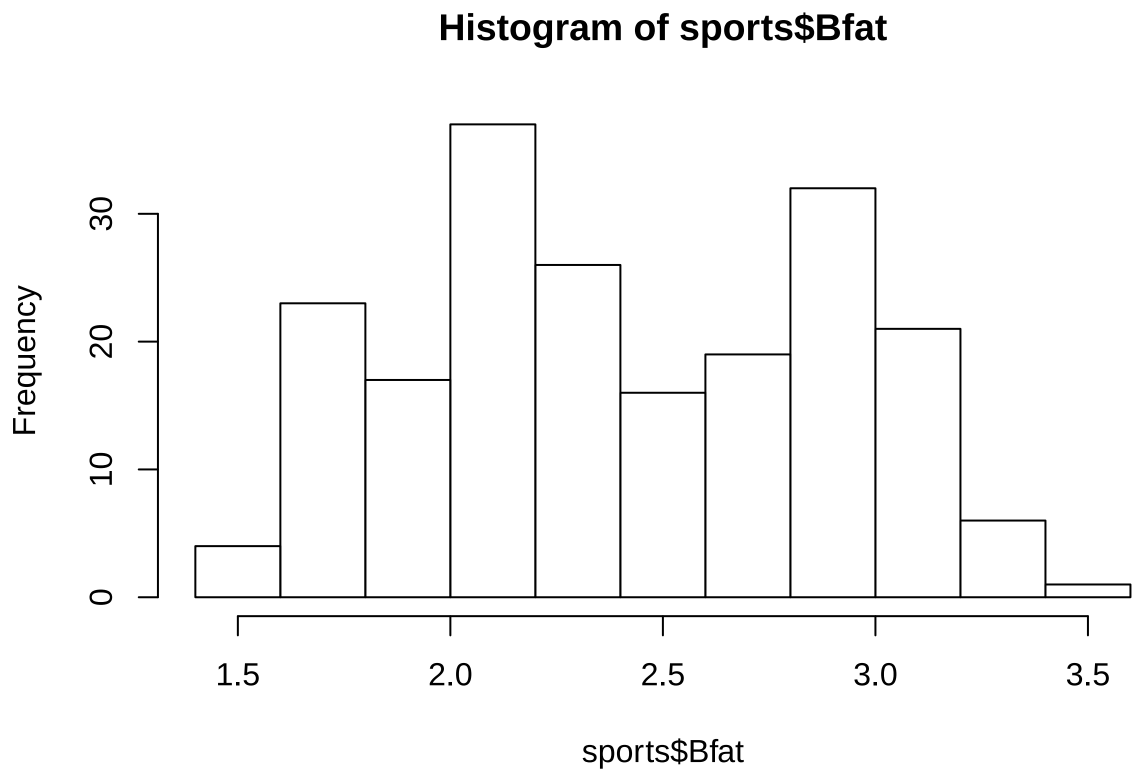
```r
# Set up data set
sports = input[,c(2, 3:13)]
dim(sports)
```

```
## [1] 202  12
```

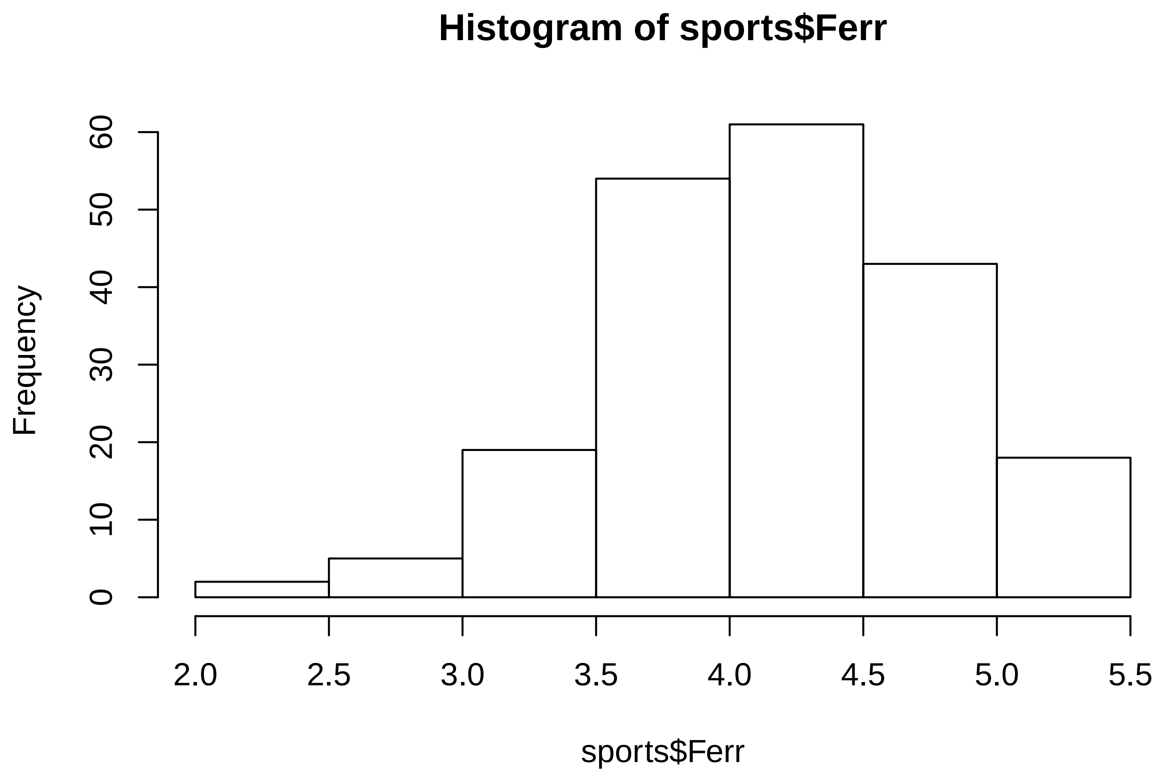
```r
n = dim(sports)[1]
names(sports)
```

```
##  [1] "Bfat" "Sex"  "Ht"   "Wt"   "LBM"  "RCC"  "WCC"  "Hc"   "Hg"   "Ferr"
## [11] "BMI"  "SSF"
```

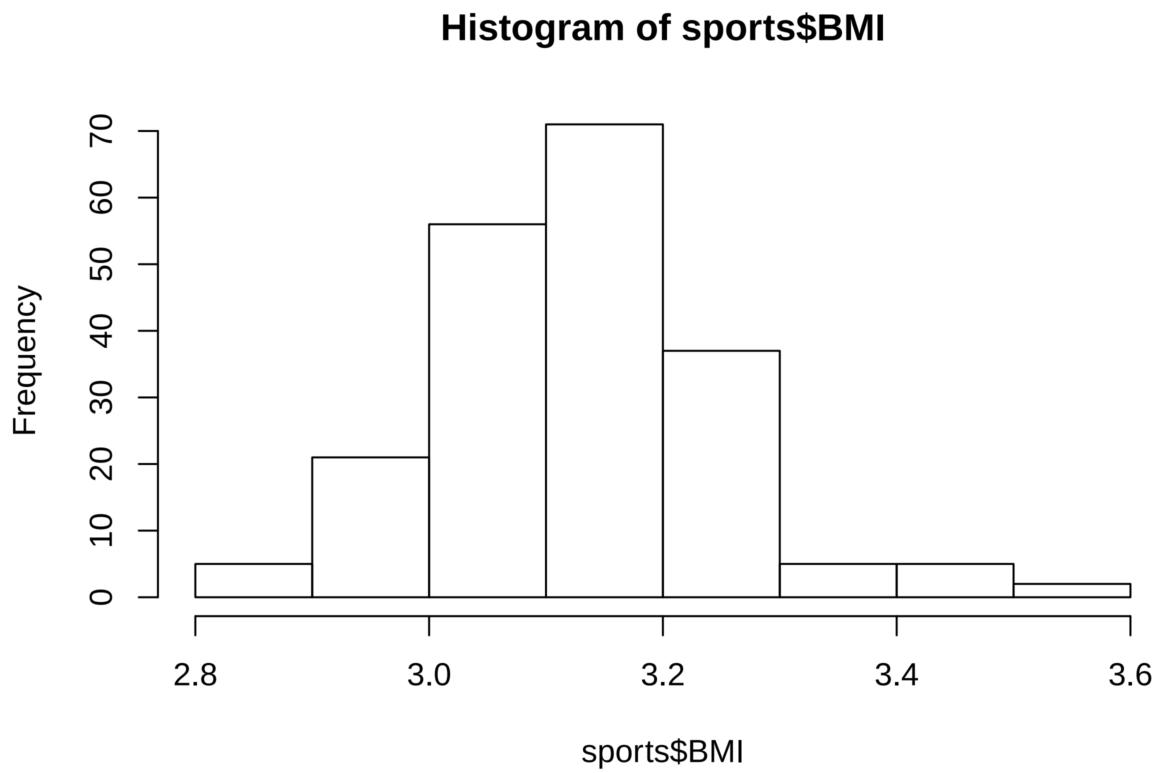
```r
# Set sex to be a factor variable
sports$Sex = as.factor(sports$Sex)

# There was some skewness with these variables, used log transformations to attempt
# to normalize
sports$Bfat = log(sports$Bfat-1)
sports$Ferr = log(sports$Ferr)
sports$BMI = log(sports$BMI)
sports$SSF = log(sports$SSF)

# Created histograms of variables to chceck for normality
hist(sports$Bfat)
```



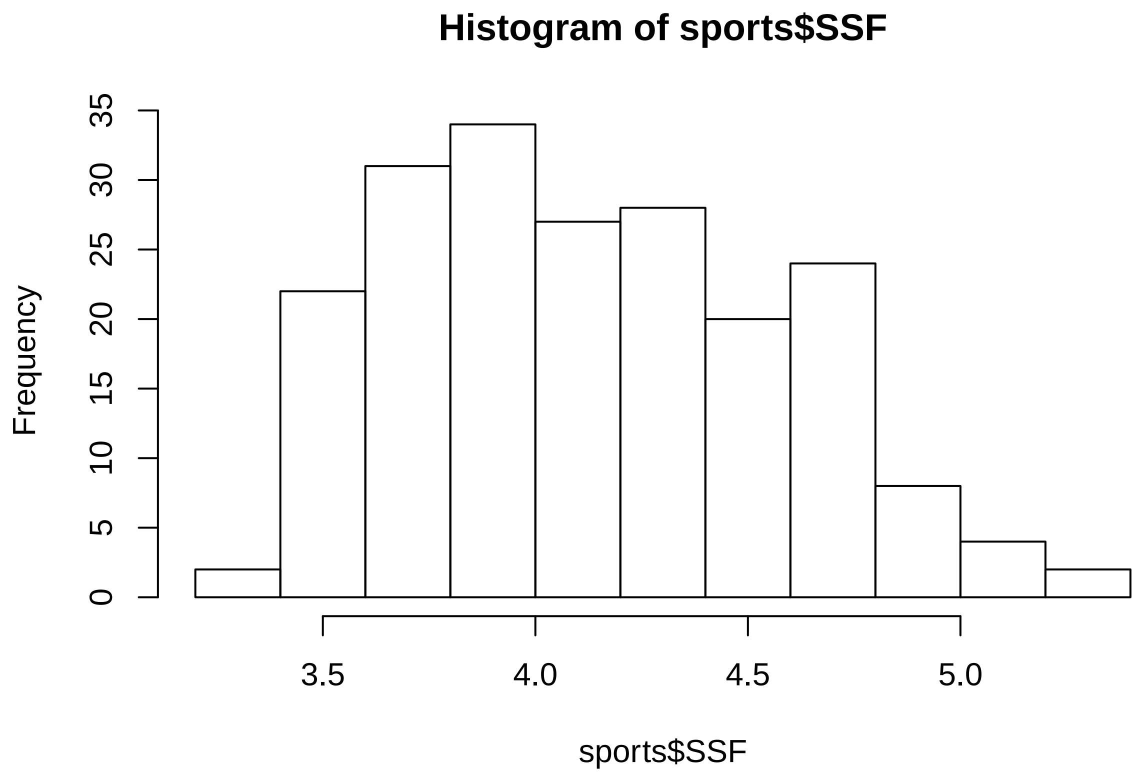

```
hist(sports$Ferr)
```



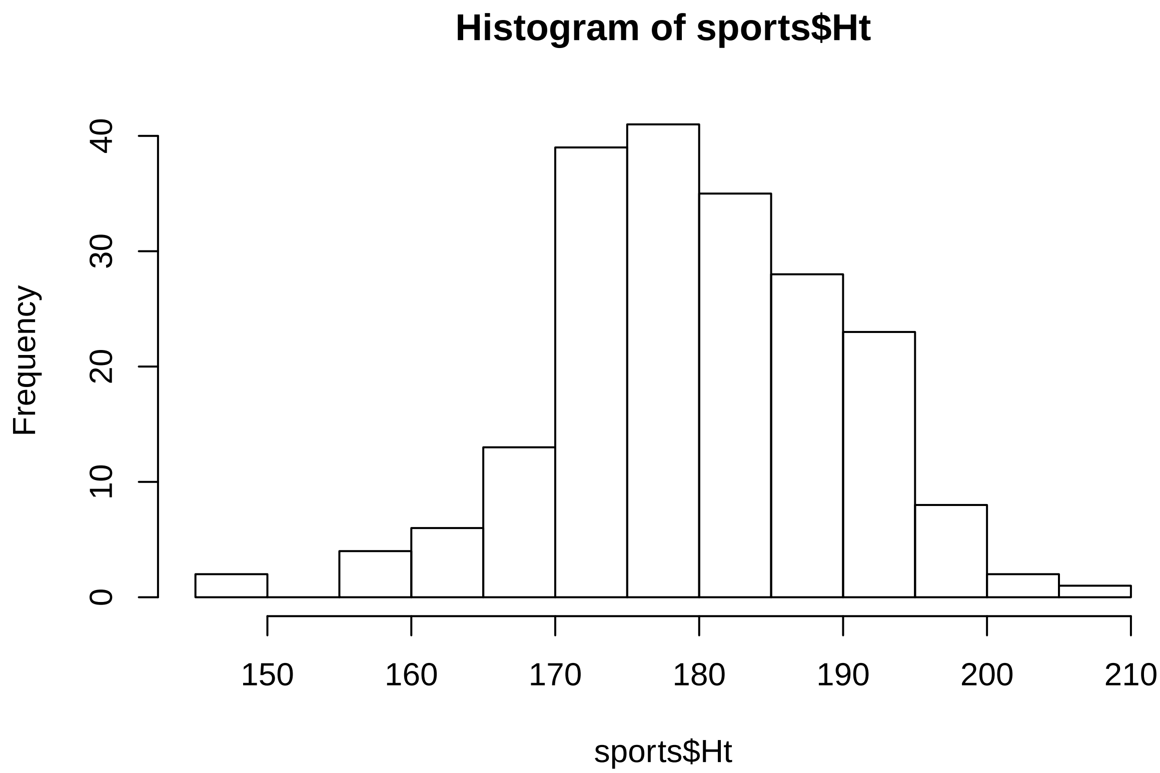

```
hist(sports$BMI)
```



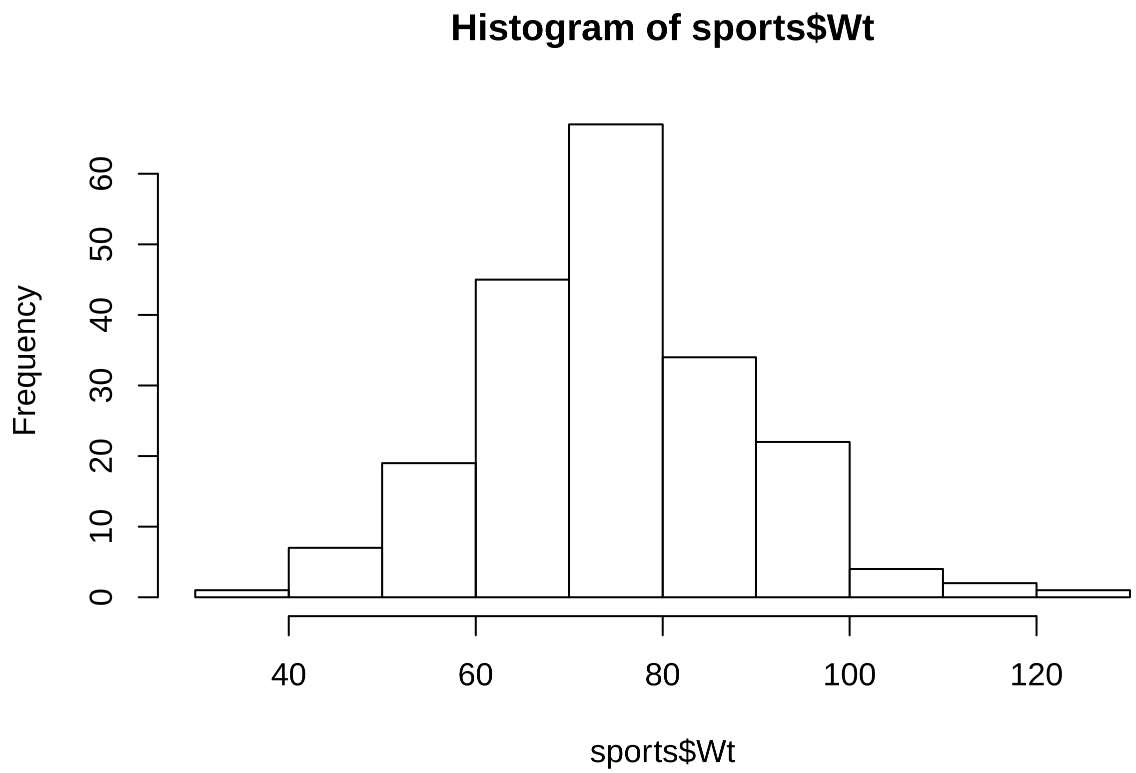

```
hist(sports$SSF)
```



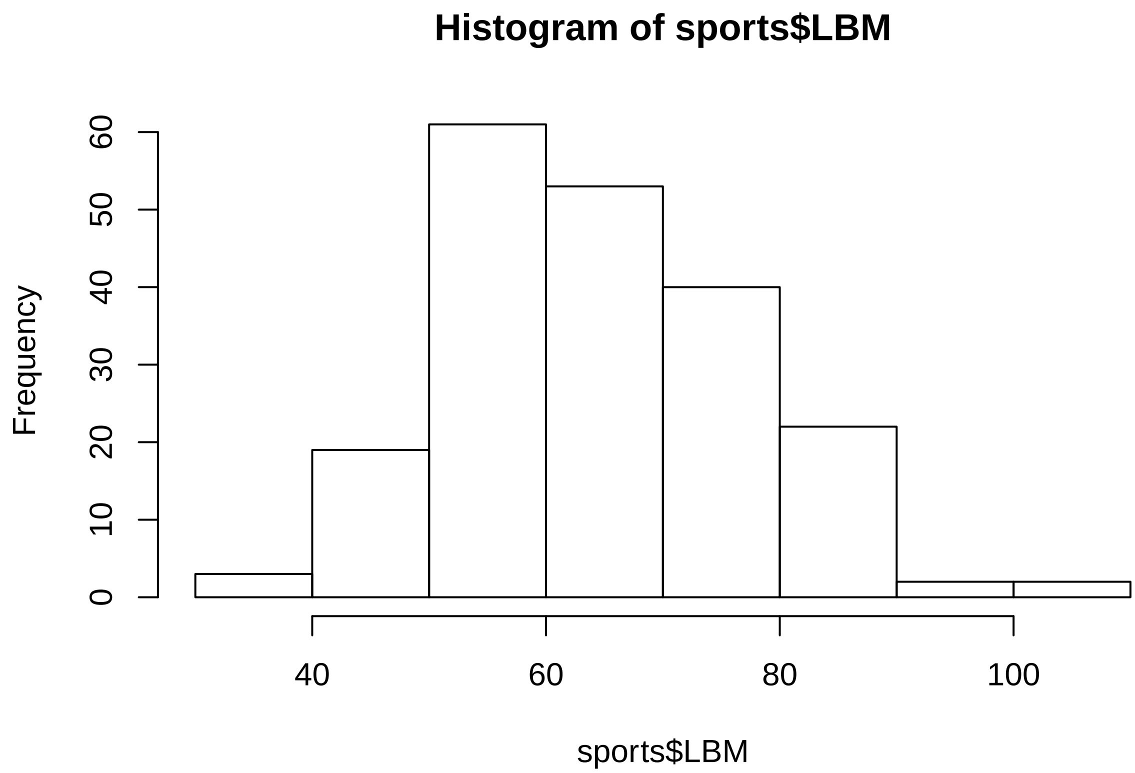

```r
hist(sports$Ht)
```



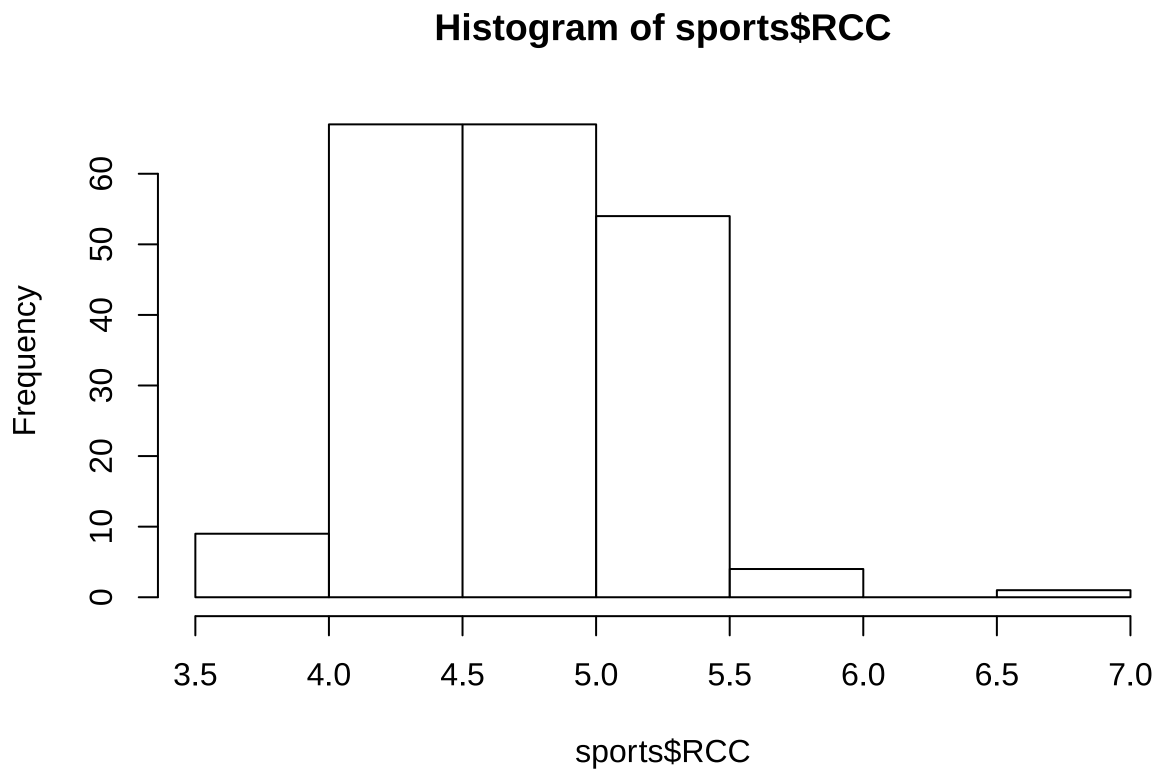

```
hist(sports$Wt)
```



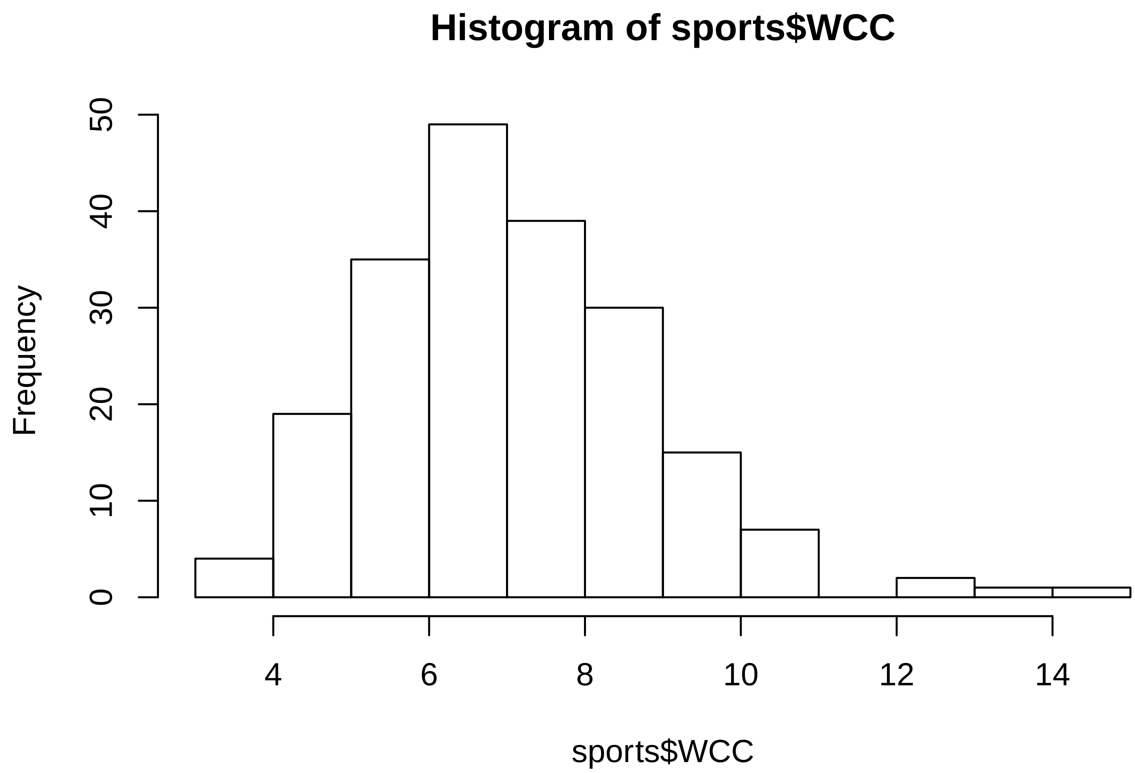

```
hist(sports$LBM)
```



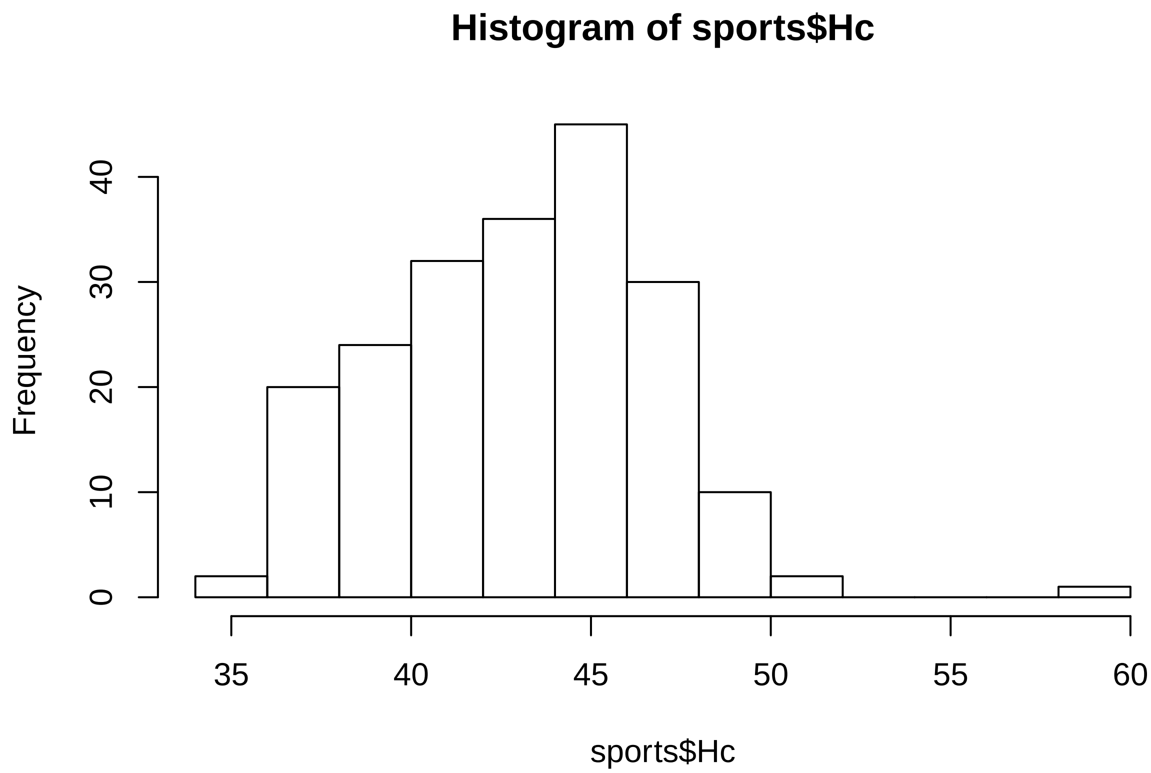

```r
hist(sports$RCC)
```



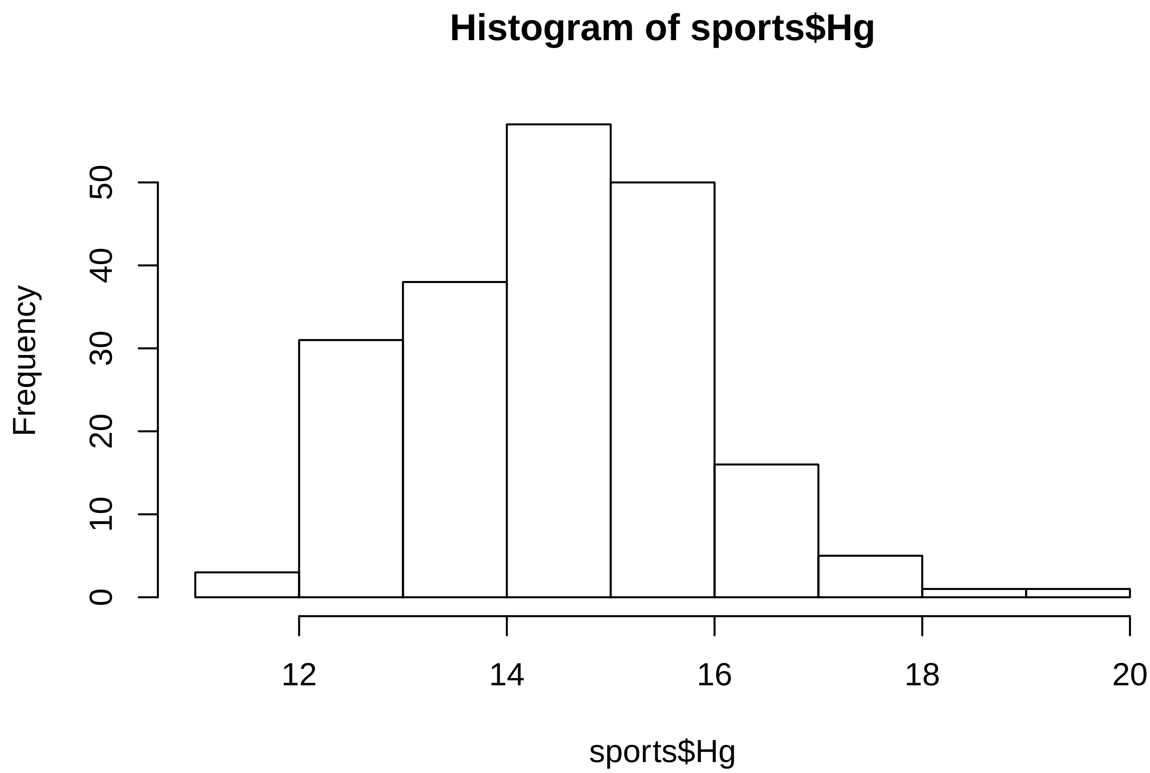

```
hist(sports$WCC)
```




```r
hist(sports$Hc)
```




```
hist(sports$Hg)
```




```r
# Create plots of complete dataset
plot(sports)

# Check to make sure there are no missing factor variables
(l <- sapply(sports, function(x) is.factor(x)))
```

```
##  Bfat   Sex    Ht    Wt   LBM   RCC   WCC    Hc    Hg  Ferr   BMI   SSF
## FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
# Run a quick regression and LASSO to get an idea of how the data looks
sports.lm = lm(Bfat ~ ., data = sports)
summary(sports.lm)
```

```
##
## Call:
## lm(formula = Bfat ~ ., data = sports)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.218786 -0.047220 -0.000482  0.054279  0.199241
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.237833   1.063001   0.224  0.82320
## Sex1         0.248749   0.027602   9.012  < 2e-16 ***
```

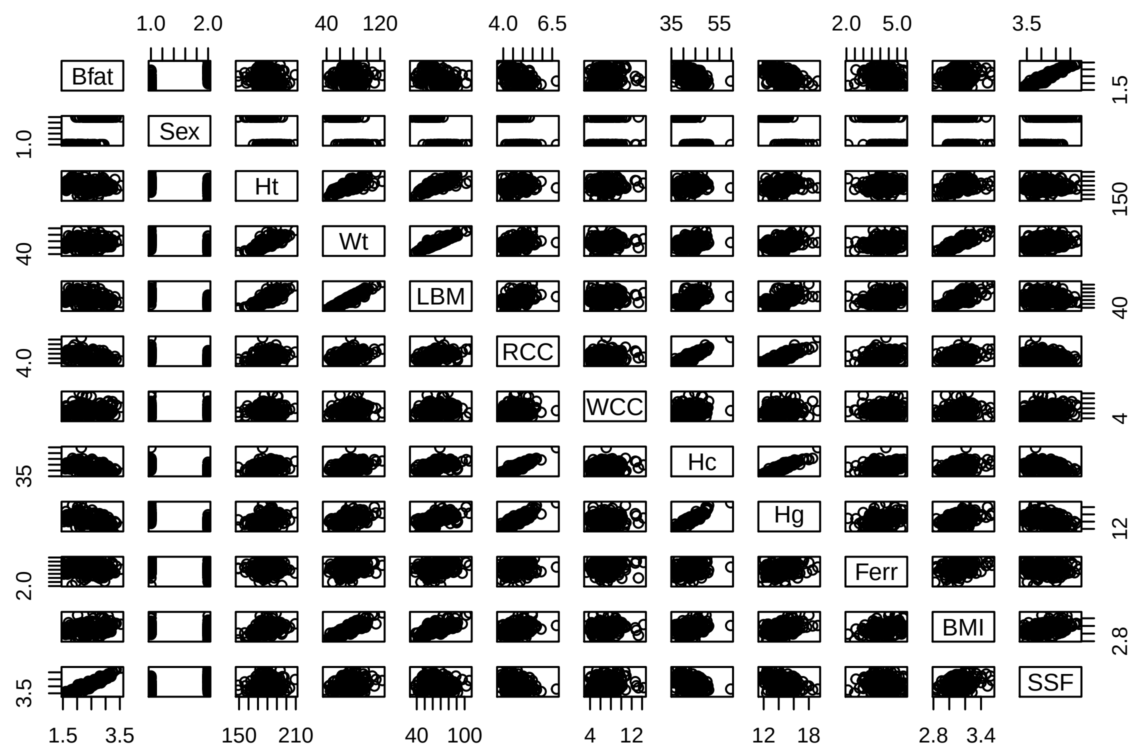
```
## Ht          -0.003257   0.002710  -1.202  0.23095
## Wt           0.022499   0.005256   4.281 2.95e-05 ***
## LBM         -0.021026   0.004209  -4.995 1.33e-06 ***
## RCC         -0.023962   0.033275  -0.720  0.47232
## WCC          0.004660   0.003318   1.405  0.16179
## Hc           0.016561   0.006234   2.657  0.00856 **
## Hg          -0.025331   0.014828  -1.708  0.08921 .
## Ferr         0.013694   0.010748   1.274  0.20418
## BMI         -0.311887   0.266619  -1.170  0.24355
## SSF          0.716210   0.044603  16.057  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08074 on 190 degrees of freedom
## Multiple R-squared:  0.9748, Adjusted R-squared:  0.9734
## F-statistic: 668.7 on 11 and 190 DF,  p-value: < 2.2e-16
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```



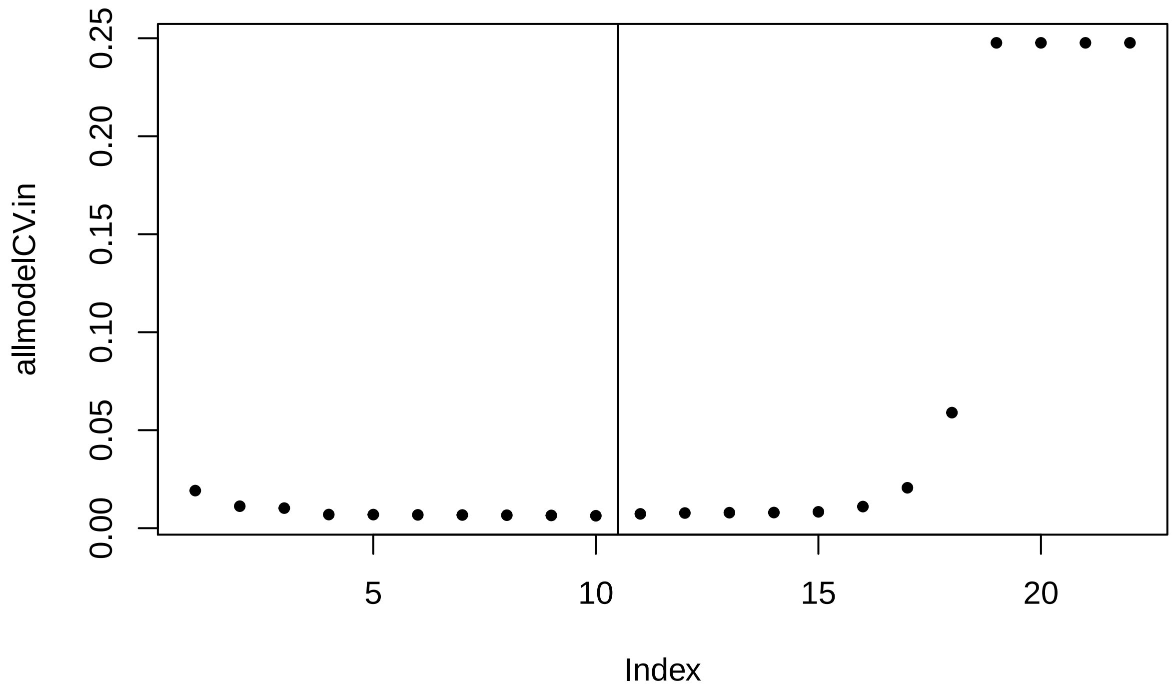
```
x = model.matrix(Bfat~.,data=sports)[,-1]
y = sports$Bfat
LASSOfit = glmnet(x,y, lmabda=0.01, alpha=1)



# specify which models to consider
# model list specification
LinModel1 = (Bfat ~ SSF)
LinModel2 = (Bfat ~ SSF + LBM)
LinModel3 = (Bfat ~ SSF + LBM + Wt)
LinModel4 = (Bfat ~ SSF + LBM + Wt + Sex)
LinModel5 = (Bfat ~ SSF + LBM + Wt + Sex + Ht)
LinModel6 = (Bfat ~ SSF + LBM + Wt + Sex + Ht + RCC)
LinModel7 = (Bfat ~ SSF + LBM + Wt + Sex + Ht + RCC + WCC)
LinModel8 = (Bfat ~ SSF + LBM + Wt + Sex + Ht + RCC + WCC + Hc)
LinModel9 = (Bfat ~ SSF + LBM + Wt + Sex + Ht + RCC + WCC + Hc + Hg)
LinModel10 = (Bfat ~ SSF + LBM + Wt + Sex + Ht + RCC + WCC +Hc + Hg + Ferr + BMI)
allLinModels = list(LinModel1, LinModel2, LinModel3, LinModel4, LinModel5, LinModel6, LinModel7, LinMode
nLinmodels = length(allLinModels)

# Specify LASSO models to consider
lambdalistLASSO = c(0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5)  # specifies LASSO m
nLASSOmodels = length(lambdalistLASSO)
nmodels = nLinmodels+nLASSOmodels

#######################################################
###### Validation set assessment of entire modeling process##########
######################################################

##### model assessment outer validation shell #####
fulldata.out = sports
k.out = 10
n.out = dim(fulldata.out)[1]
#define the split into training set (of size about 2/3 of data) and validation set (of size about 1/3)
groups.out = c(rep(1:k.out,floor(n.out/k.out))); if(floor(n.out/k.out) != (n.out/k.out)) groups.out = c
set.seed(8, sample.kind = "Rounding")
```

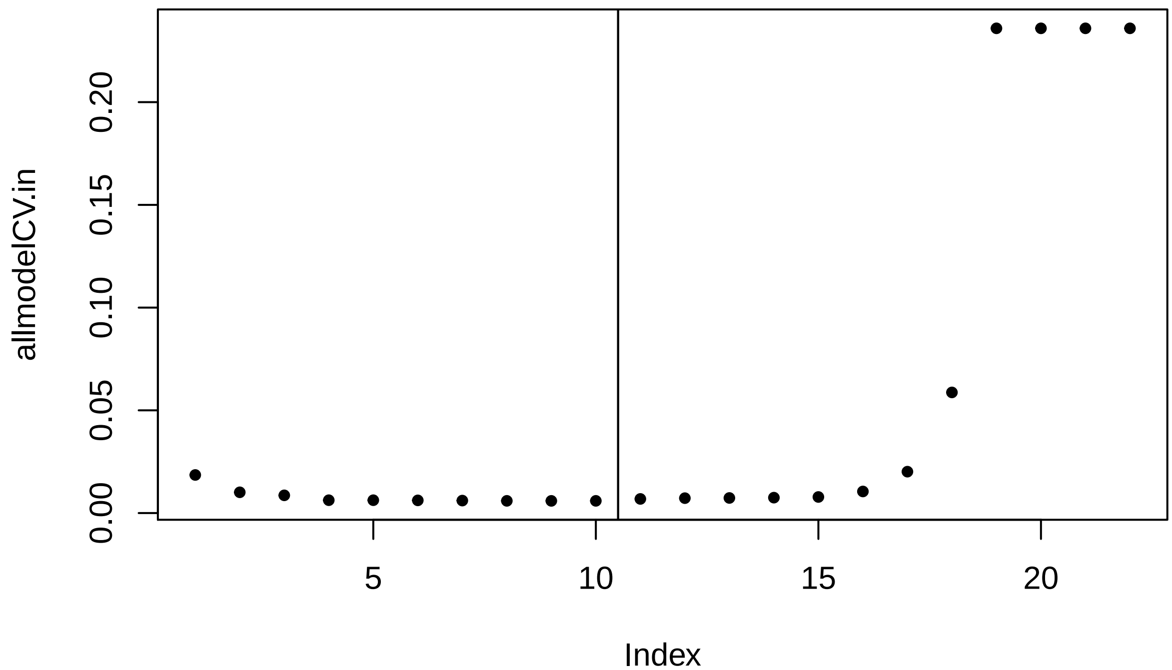
```
## Warning in set.seed(8, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

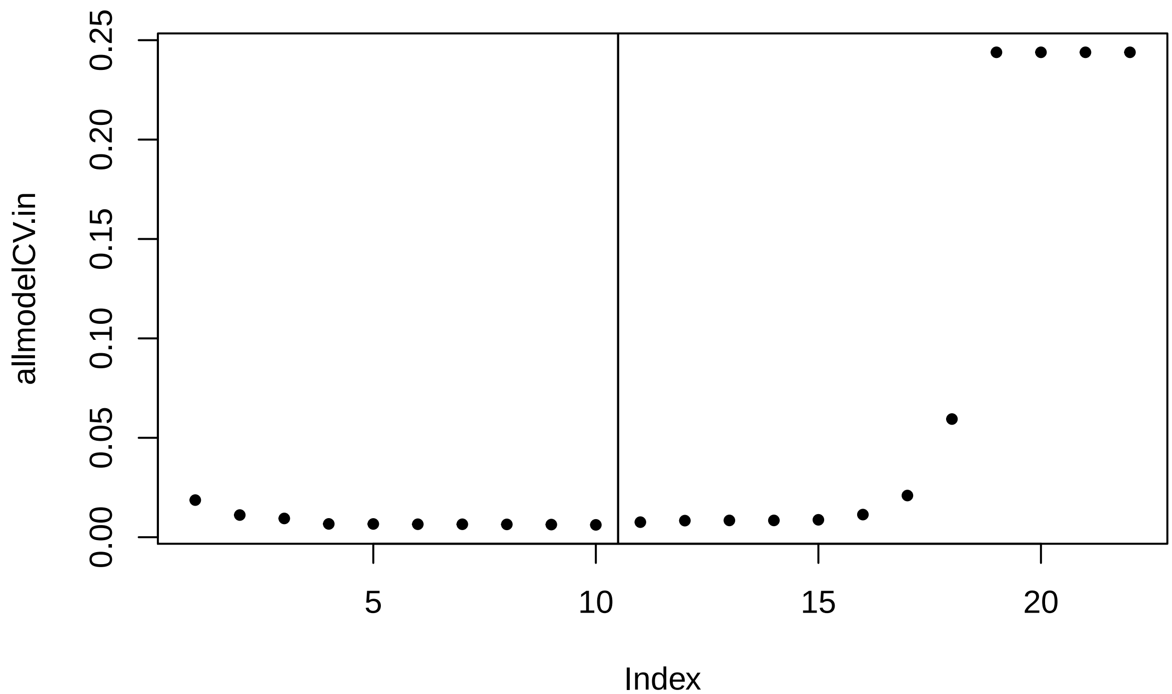
```
cvgroups.out = sample(groups.out,n.out)  #orders randomly, with seed (8)

# set up storage for predicted values from the double-cross-validation
allpredictedCV.out = rep(NA,n.out)
# set up storage to see what models are "best" on the inner loops
allbestmodels = rep(NA,k.out)


for (j in 1:k.out){

  # Just one split int training and validation sets
  groupj.out = (cvgroups.out == j)
```

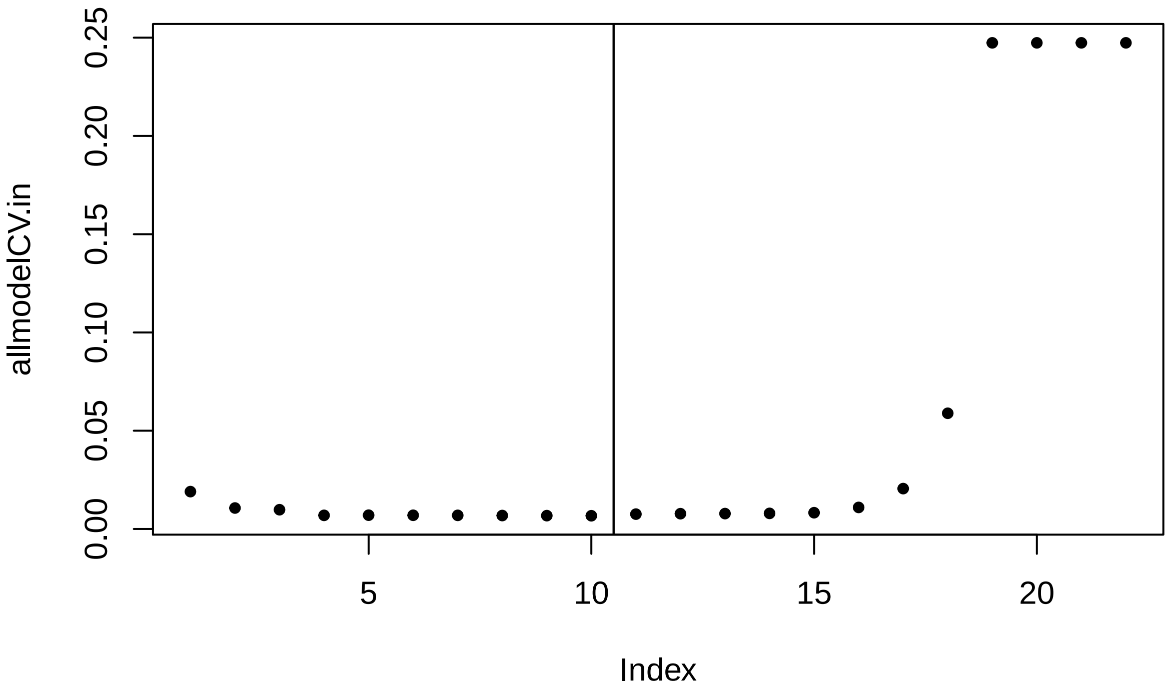
```r
traindata.out = sports[!groupj.out,]
trainx.out = model.matrix(Bfat ~ ., data=traindata.out)[,-1]
trainy.out = traindata.out[,1]
validdata.out = sports[groupj.out,]
validx.out = model.matrix(Bfat ~ ., data=validdata.out)[,-1]
validy.out = validdata.out[,1]

### entire model-fitting process ####
fulldata.in = traindata.out


################################
### Full modeling process ##
###############################

## we begin setting up the model-fitting process to use notation that will be
## useful later, inside a validation
n.in = dim(fulldata.in)[1]
x.in = model.matrix(Bfat ~ ., data = fulldata.in)[,-1]
y.in = fulldata.in[,1]

# number folds and groups for cross-validation for model selection
k.in = 10

# produce list of group labels
groups.in = c(rep(1:k.in, floor(n.in/k.in)))
if(floor(n.in/k.in) != (n.in/k.in)){
  groups.in = c(groups.in, 1:(n.in%%k.in))
}

# orders randomly with seed 8
cvgroups.in = sample(groups.in, n.in)

# check correct distribution
table(cvgroups.in)

# place holder for results
allmodelCV.in = rep(NA, nmodels)


##### cross-validation for model selection ##### reference - Lesson 2

# since linear regression does not have any automatic CV output,
# set up storage for predicted values from the CV splits, across all linear models
allpredictedCV.in = matrix(rep(NA,n.in*nLinmodels),ncol=nLinmodels)

#cycle through all folds:  fit the model to training data, predict test data,
# and store the (cross-validated) predicted values
for (i in 1:k.in)  {
  train.in = (cvgroups.in != i)
  test.in = (cvgroups.in == i)
  #fit each of the linear regression models on training, and predict the test
  for (m in 1:nLinmodels) {
```

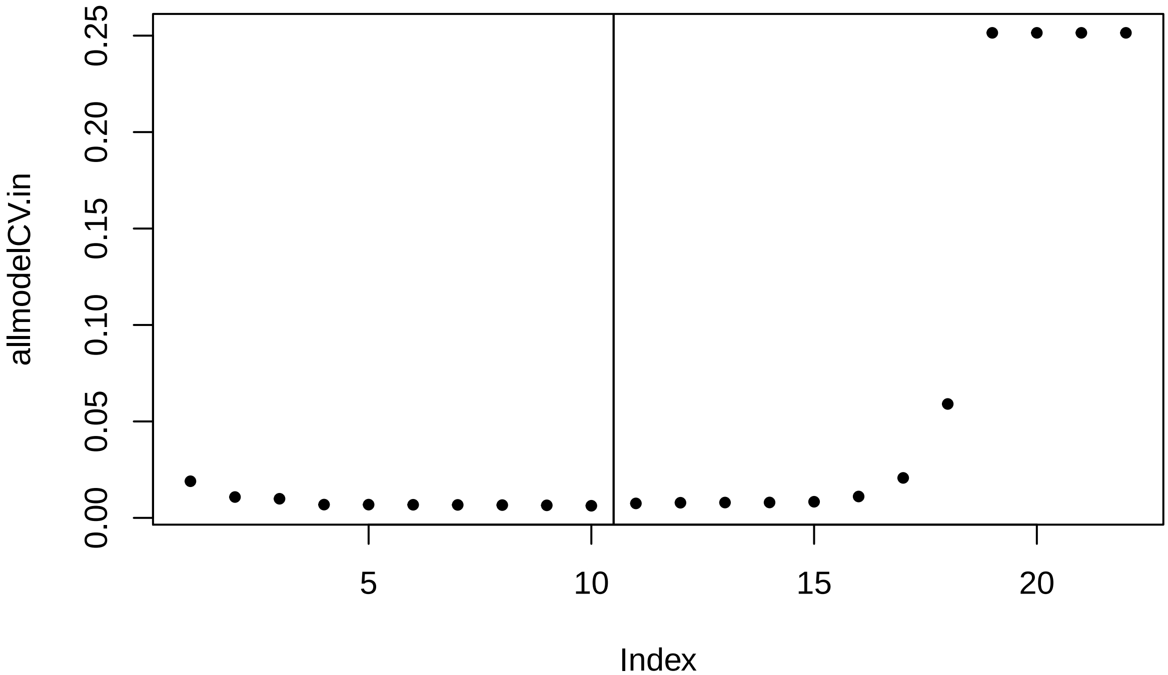
```
      lmfitCV.in = lm(formula = allLinModels[[m]],data=sports,subset=train.in)
      allpredictedCV.in[test.in,m] = predict.lm(lmfitCV.in,fulldata.in[test.in,])
    }
  }
  # compute and store the CV(10) values
  for (m in 1:nLinmodels) {
    allmodelCV.in[m] = mean((allpredictedCV.in[,m]-fulldata.in$Bfat)^2)
  }

  #LASSO cross-validation - uses internal cross-validation function
  cvLASSOglm.in = cv.glmnet(x.in, y.in, lambda=lambdalistLASSO, alpha = 1, nfolds=k.in, foldid=cvgroups

  # store CV(10) values, in same numeric order as lambda, in storage spots for CV values
  allmodelCV.in[(1:nLASSOmodels)+nLinmodels] = cvLASSOglm.in$cvm[order(cvLASSOglm.in$lambda)]

  # visualize CV(10) values across all methods
  plot(allmodelCV.in,pch=20); abline(v=c(nLinmodels+.5,nLinmodels+.5))

  bestmodel.in = (1:nmodels)[order(allmodelCV.in)[1]]  # actual selection
  # state which is best model and minimum CV(10) value
  bestmodel.in; min(allmodelCV.in)

  ### finally, fit the best model to the full (available) data ###
  if (bestmodel.in <= nLinmodels) {  # then best is one of linear models
    bestfit = lm(formula = allLinModels[[bestmodel.in]],data=fulldata.in)  # fit on all available data
    bestcoef = coef(bestfit)
  } else {  # then best is one of LASSO models
    bestlambdaLASSO = (lambdalistLASSO)[bestmodel.in-nLinmodels]
    bestfit = glmnet(x.in, y.in, alpha = 1,lambda=lambdalistLASSO)  # fit the model across possible lam
    bestcoef = coef(bestfit, s = bestlambdaLASSO) # coefficients for the best model fit
  }

  #############################
  ## End of modeling process ##
  #############################

  allbestmodels[j] = bestmodel.in

  if(bestmodel.in <= nLinmodels) { # then best is one of linear models
    allpredictedCV.out[groupj.out] = predict(bestfit, validdata.out)
  } else { # then best is one of LASSO models
    allpredictedCV.out[groupj.out] = predict(bestfit, newx=validdata.out, s=bestlambdaLASSO)
  }
}
```

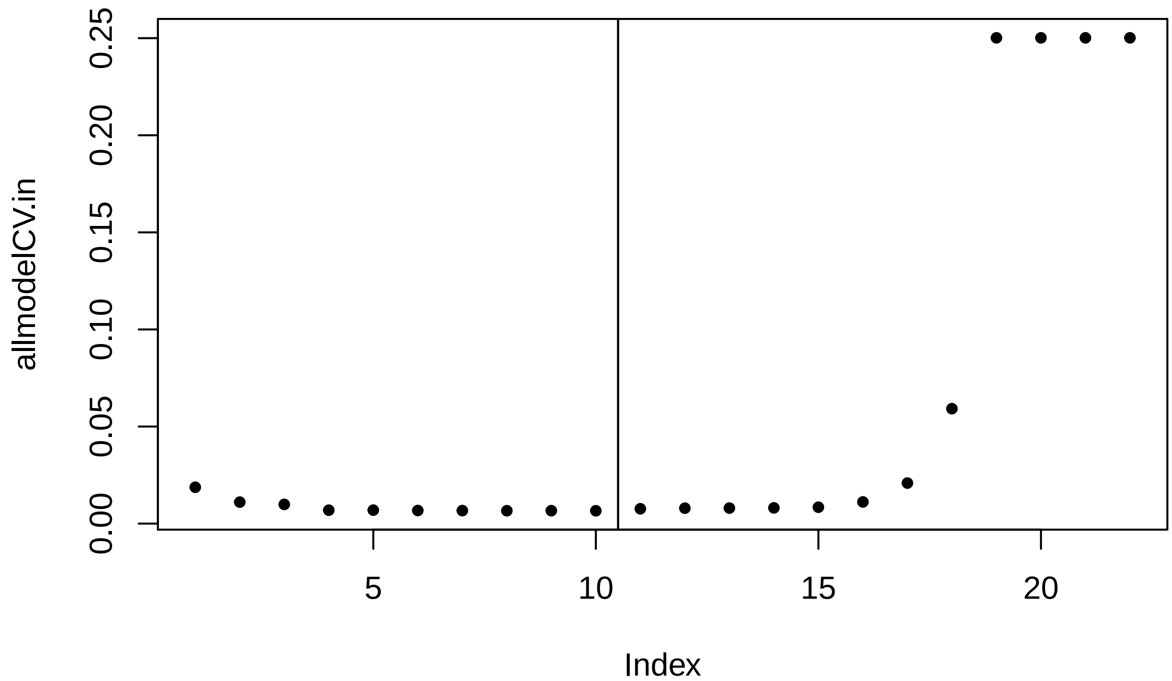

allmodelCV.in
0.00
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Index

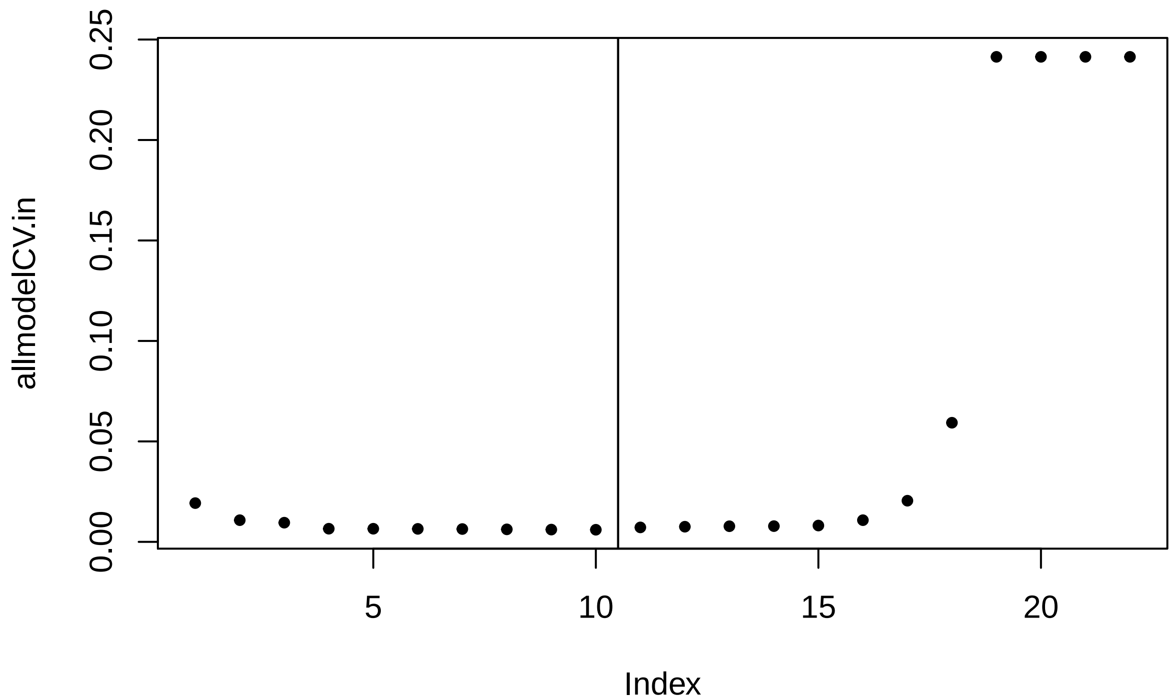

allmodelCV.in
0.00
0.05
0.10
0.15
0.20
5
10
15
20
Index

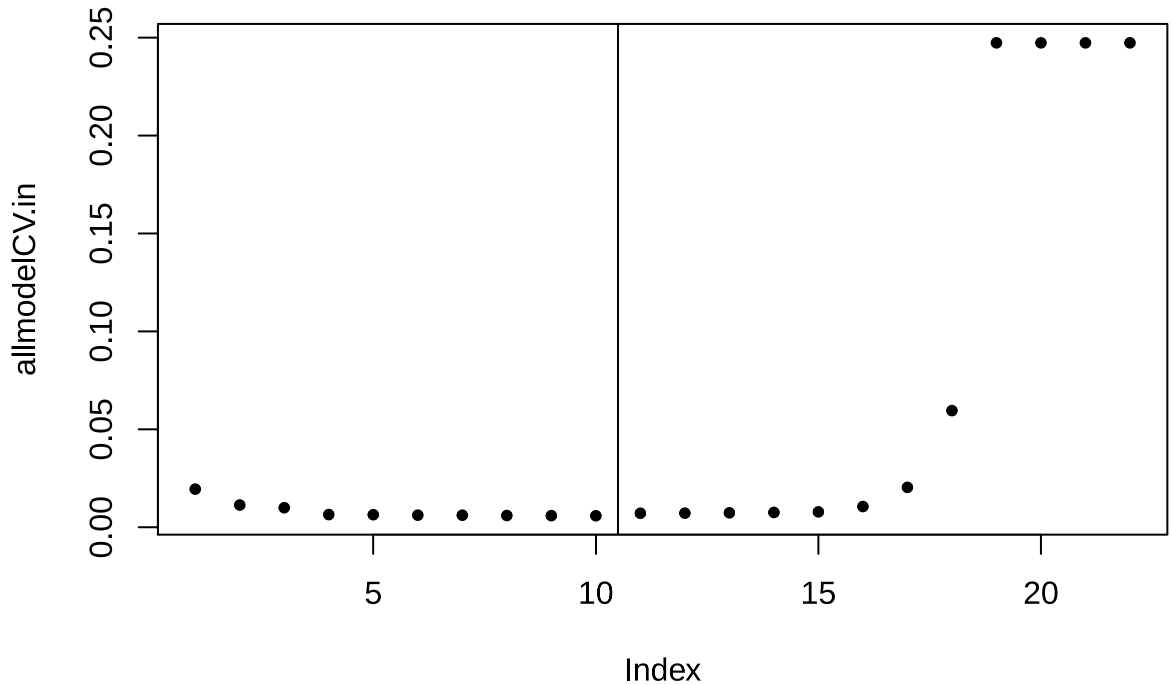

allmodelCV.in
0.00
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Index


allmodelCV.in
0.00
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Index

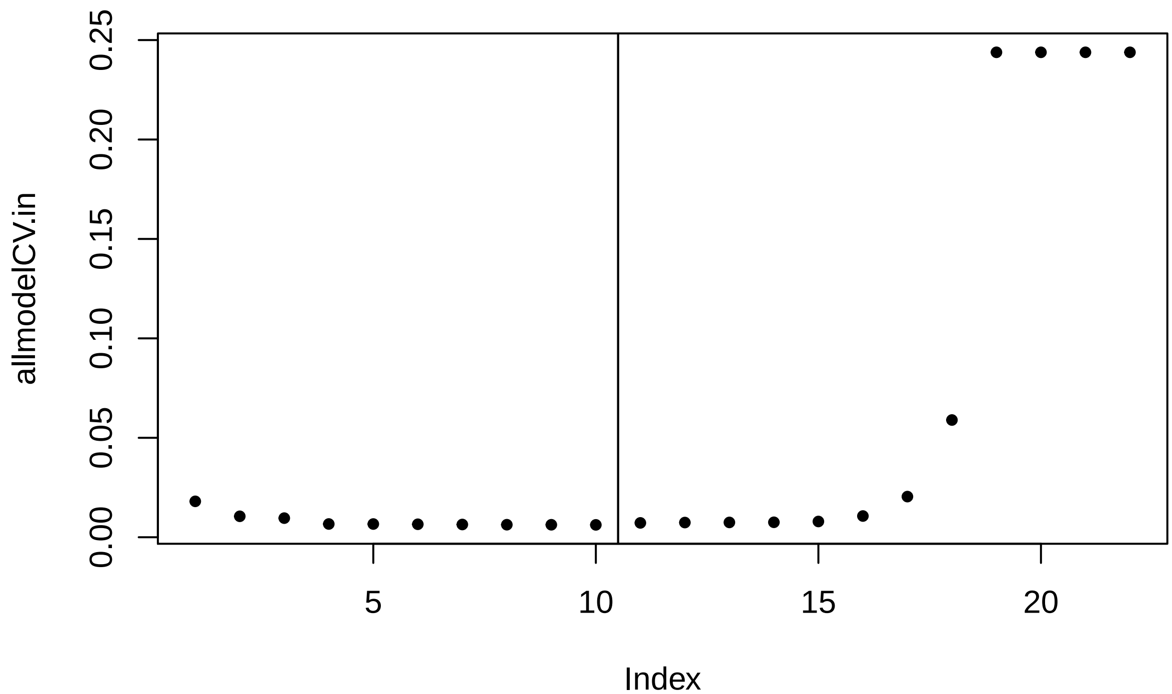

allmodelCV.in
0.00
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Index

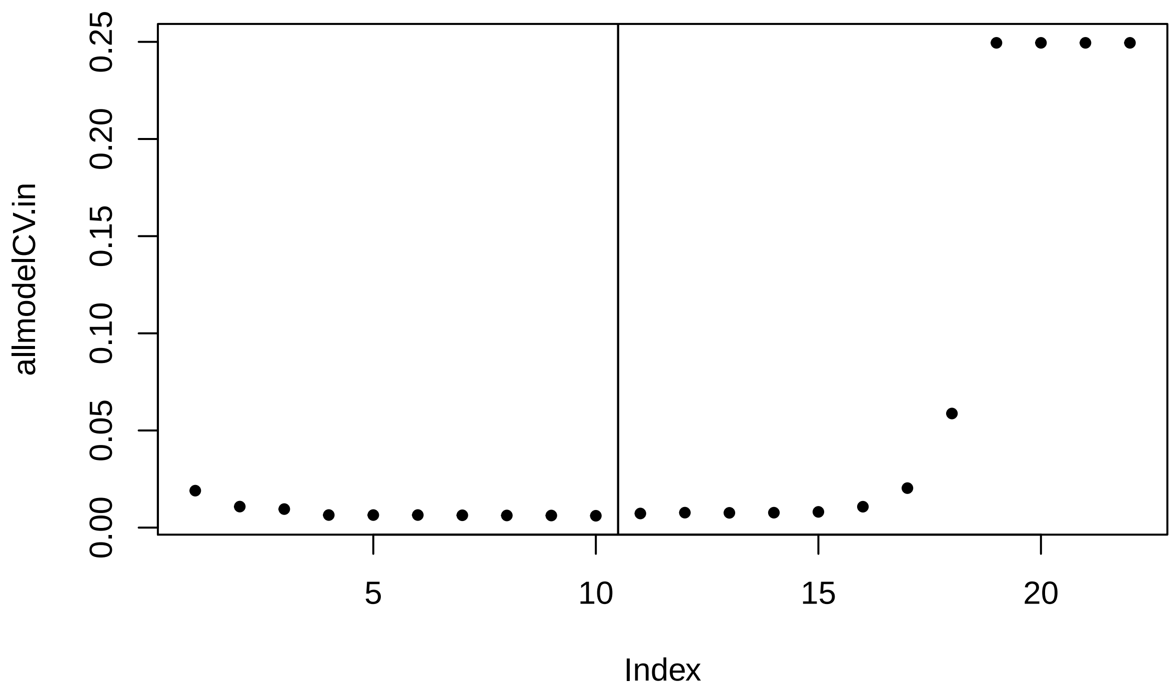

allmodelCV.in
0.00
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Index


allmodelCV.in
0.00
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Index



0.00
0.05
0.10
0.15
0.20
0.25
allmodelCV.in
5
10
15
20
Index


allmodelCV.in
0.00
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Index




```
# for curiosity, we can see the models that were "best" on each of the inner splits
allbestmodels
```

```
##  [1] 10  9 10 10 10 10 10 10 10 10
```

```
#assessment
y.out = fulldata.out$Bfat
CV.out = sum((allpredictedCV.out-y.out)^2)/n.out;
CV.out
```

```
## [1] 0.007325102
```

```
R2.out = 1-sum((allpredictedCV.out-y.out)^2)/sum((y.out-mean(y.out))^2);
R2.out
```

```
## [1] 0.9699189
```