Stat542: Linear Regression

Prepare the Boston Housing Data

```
library(MASS)
data(Boston)
?Boston  # Check the description of the Boston data
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
##   lstat medv
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
## 4  2.94 33.4
## 5  5.33 36.2
## 6  5.21 28.7
```

```
dim(Boston)
```

```
## [1] 506  14
```

```
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

The data frame contains the following columns:

- crim: per capita crime rate by town.
- zn: proportion of residential land zoned for lots over 25,000 sq.ft.
- indus: proportion of non-retail business acres per town.
- chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- nox: nitrogen oxides concentration (parts per 10 million).
- rm: average number of rooms per dwelling.
- age: proportion of owner-occupied units built prior to 1940.
- dis: weighted mean of distances to five Boston employment centres.
- rad: index of accessibility to radial highways.
- tax: full-value property-tax rate per $10,000.
- ptratio: pupil-teacher ratio by town.
- black: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- lstat: lower status of the population (percent).
- **medv**: median value of owner-occupied homes in $1000s.

Change the response variable name to be "Y". Next take some transformations on Y and X's, suggested in the literature.

```
myData <- Boston
names(myData)[14] <- "Y"
iLog <- c(1, 3, 5, 6, 8, 9, 10, 14)
myData[, iLog] <- log(myData[, iLog])
myData[, 2] <- myData[, 2]/10
myData[, 7] <- myData[, 7]^2.5/10^4
myData[, 11] <- exp(0.4 * myData[, 11])/1000
myData[, 12] <- myData[, 12]/100
myData[, 13] <- sqrt(myData[, 13])
```

A quick summary of each column of myData

```
summary(myData)
```

```
##       crim               zn             indus              chas        
##  Min.   :-5.0640   Min.   : 0.000   Min.   :-0.7765   Min.   :0.00000  
##  1st Qu.:-2.5005   1st Qu.: 0.000   1st Qu.: 1.6467   1st Qu.:0.00000  
##  Median :-1.3606   Median : 0.000   Median : 2.2711   Median :0.00000  
##  Mean   :-0.7804   Mean   : 1.136   Mean   : 2.1602   Mean   :0.06917  
##  3rd Qu.: 1.3021   3rd Qu.: 1.250   3rd Qu.: 2.8959   3rd Qu.:0.00000  
##  Max.   : 4.4884   Max.   :10.000   Max.   : 3.3229   Max.   :1.00000  
##       nox                rm             age                 dis        
##  Min.   :-0.9545   Min.   :1.270   Min.   : 0.001432   Min.   :0.1219  
##  1st Qu.:-0.8007   1st Qu.:1.772   1st Qu.: 1.360301   1st Qu.:0.7420  
##  Median :-0.6199   Median :1.826   Median : 5.287613   Median :1.1655  
##  Mean   :-0.6100   Mean   :1.832   Mean   : 5.060790   Mean   :1.1880  
##  3rd Qu.:-0.4716   3rd Qu.:1.891   3rd Qu.: 8.583922   3rd Qu.:1.6464  
##  Max.   :-0.1381   Max.   :2.172   Max.   :10.000000   Max.   :2.4954  
##       rad             tax           ptratio           black       
##  Min.   :0.000   Min.   :5.231   Min.   :0.1545   Min.   :0.0032  
##  1st Qu.:1.386   1st Qu.:5.631   1st Qu.:1.0536   1st Qu.:3.7538  
##  Median :1.609   Median :5.799   Median :2.0390   Median :3.9144  
##  Mean   :1.868   Mean   :5.931   Mean   :2.1501   Mean   :3.5667  
##  3rd Qu.:3.178   3rd Qu.:6.501   3rd Qu.:3.2292   3rd Qu.:3.9623  
##  Max.   :3.178   Max.   :6.567   Max.   :6.6342   Max.   :3.9690  
##      lstat             Y        
##  Min.   :1.315   Min.   :1.609  
##  1st Qu.:2.636   1st Qu.:2.835  
##  Median :3.370   Median :3.054  
##  Mean   :3.418   Mean   :3.035  
##  3rd Qu.:4.118   3rd Qu.:3.219  
##  Max.   :6.162   Max.   :3.912
```

Produce a pair-wise scatter plot. Caution: a big figure.

```
pairs(myData, pch='.')
```

Fit a Linear Model

Fit a linear regression model using all the predictors.

```
lmfit <-  lm(Y ~ ., data = myData)
```

Check what have been returned by `lm`.

```
names(lmfit)  # What have been returned by "lm"?
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"         
##  [5] "fitted.values" "assign"        "qr"            "df.residual"  
##  [9] "xlevels"       "call"          "terms"         "model"
```

Check how to retrieve various LS results.

```
lmfit$residuals[1]
```

```
##          1 
## -0.2209709
```

```
length(lmfit$residuals)
```

```
## [1] 506
```

```
sqrt(sum(lmfit$residuals^2)/(506 - 14))  # residual standard error
```

```
## [1] 0.2007528
```

```
1 - sum(lmfit$residuals^2)/(var(myData$Y)*505)  # R-square
```

```
## [1] 0.7650004
```

```
lmfit$coef    # 13 regression cofficients including the intercept
```

```
##  (Intercept)         crim           zn        indus         chas 
##  4.176874035 -0.014606367  0.001391943 -0.012709368  0.109980144 
##          nox           rm          age          dis          rad 
## -0.283111884  0.421107840  0.006403368 -0.183154286  0.068361590 
##          tax      ptratio        black        lstat 
## -0.201832385 -0.040017441  0.044471934 -0.262615094
```

Print the summary of the LS results

```
summary(lmfit)
```

```
## 
## Call:
## lm(formula = Y ~ ., data = myData)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.9918 -0.1002 -0.0034  0.1117  0.7640 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  4.176874   0.379017  11.020  < 2e-16 ***
## crim        -0.014606   0.011650  -1.254 0.210527    
## zn           0.001392   0.005639   0.247 0.805121    
## indus       -0.012709   0.022312  -0.570 0.569195    
## chas         0.109980   0.036634   3.002 0.002817 ** 
## nox         -0.283112   0.105340  -2.688 0.007441 ** 
## rm           0.421108   0.110175   3.822 0.000149 ***
## age          0.006403   0.004863   1.317 0.188536    
## dis         -0.183154   0.036804  -4.977 8.97e-07 ***
## rad          0.068362   0.022473   3.042 0.002476 ** 
## tax         -0.201832   0.048432  -4.167 3.64e-05 ***
## ptratio     -0.040017   0.008091  -4.946 1.04e-06 ***
## black        0.044472   0.011456   3.882 0.000118 ***
## lstat       -0.262615   0.016091 -16.320  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.2008 on 492 degrees of freedom
## Multiple R-squared:  0.765,  Adjusted R-squared:  0.7588 
## F-statistic: 123.2 on 13 and 492 DF,  p-value: < 2.2e-16
```

Predict the price for two new houses.

```
newx <-  apply(myData, 2, median)
newx <-  rbind(newx, newx)
newx
```

```
##           crim zn    indus chas        nox       rm      age      dis
## newx -1.360641  0 2.271094    0 -0.6198967 1.825919 5.287613 1.165473
## newx -1.360641  0 2.271094    0 -0.6198967 1.825919 5.287613 1.165473
##           rad      tax ptratio  black    lstat        Y
## newx 1.609438 5.799093 2.03897 3.9144 3.370459 3.054001
## newx 1.609438 5.799093 2.03897 3.9144 3.370459 3.054001
```

```
newx[, 4] <-  c(1,0)
newx
```

```
##           crim zn    indus chas        nox       rm      age      dis
## newx -1.360641  0 2.271094    1 -0.6198967 1.825919 5.287613 1.165473
## newx -1.360641  0 2.271094    0 -0.6198967 1.825919 5.287613 1.165473
##           rad      tax ptratio  black    lstat        Y
## newx 1.609438 5.799093 2.03897 3.9144 3.370459 3.054001
## newx 1.609438 5.799093 2.03897 3.9144 3.370459 3.054001
```

```
newx[, -14] %*% lmfit$coef[-1] + lmfit$coef[1]
```

```
##          [,1]
## newx 3.189603
## newx 3.079622
```

```
# or use the "predict" function, then new data should
# be a data frame.
row.names(newx) = NULL
newx <- data.frame(newx)
predict(lmfit, newdata = newx)
```

```
##        1        2
## 3.189603 3.079622
```

Rank Deficiency

If the design matrix (including the intercept) is not of full rank, the coefficient vector returned by R will have some elements to be NA. A column has its LS estimate to be NA means that it can be written as a linear combination of some columns listed before it, that is, this is a redundant column.

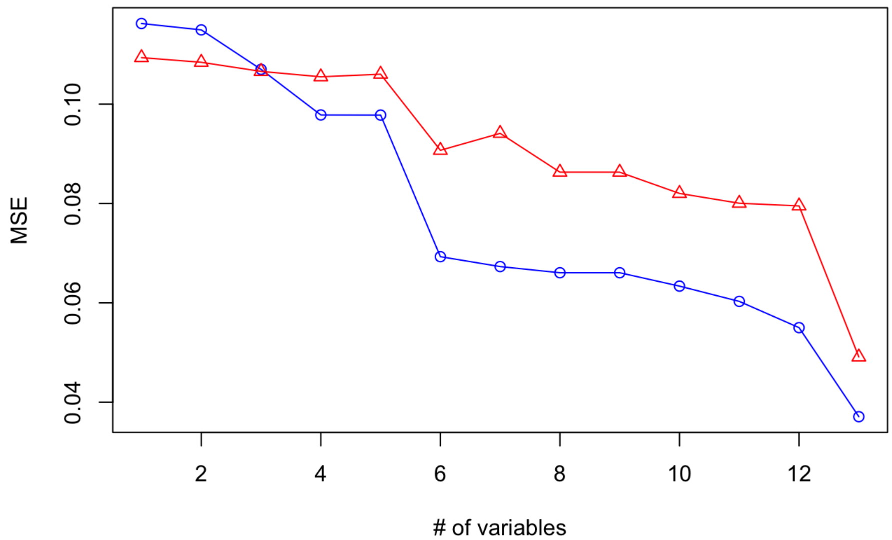
NA values do not mean error. It just means that in the LS fitting, R ignores the columns with NA coefficients. You can still use the fitted model to do prediction. The result should be the same as if you fit a linear regression model without those columns.

```
## Add a fake column named "junk"
myData$junk <-  myData$crim + myData$zn
tmp.lm <-  lm(Y ~ ., myData)
summary(tmp.lm)

## The fitted values (for the first 3 obs) are the same.
tmp.lm$fitted[1:3]
lmfit$fitted[1:3]

## remove the "junk" column
myData = myData[,-15]
```

Training Error vs Test Error

For linear regression models, when we add more and more variables, the training error (e.g., RSS or MSE) is always decreasing, but the test error (prediction error on an independent test data) is not necessary decreasing.

```
# Go back to the Boston Housing Data
# Divide the data into training and test

# n: sample size
# col 1:p: predictors
# col (p+1): response (in this particular example)
n <- dim(myData)[1]
p <-  dim(myData)[2] - 1

ntrain <-  round(n*0.6)
train.id <-  sample(1:n, ntrain)
train.MSE <-  rep(0, p)
test.MSE <-  rep(0, p)

for(i in 1:p){
  myfit <-  lm(Y ~ ., myData[train.id, c(1:i, (p+1))])
  train.Y <-  myData[train.id, (p+1)]
  train.Y.pred <-  myfit$fitted
  train.MSE[i] <-  mean((train.Y - train.Y.pred)^2)

  test.Y <-  myData[-train.id, (p+1)]
  test.Y.pred <-  predict(myfit, newdata = myData[-train.id, ])
  test.MSE[i] <-  mean((test.Y - test.Y.pred)^2)
}

## type="n": don't plot; just set the plotting region
  plot(c(1, p), range(train.MSE, test.MSE), type="n",
     xlab="# of variables", ylab="MSE")
  points(train.MSE, col = "blue", pch = 1)
  lines(train.MSE, col = "blue", pch = 1)
  points(test.MSE, col = "red", pch = 2)
  lines(test.MSE, col = "red", pch = 2)
```



You can run the code above multiple times. In most cases, the blue line is below the red line (i.e., training error is better than test error), but sometimes, the red line could be below the blue line (i.e., test error is even better than the training error). In each iteration, you would see the blue line is always monotonically decreasing, but red line is not necessarily decreasing. Check the differences between the adjacent terms, which should be always negative for the blue line, but could have some positive terms for the red line.

```
diff(train.RSS) ## always negative
diff(test.RSS)  ## not always negative
```

Understand the LS Coefficient

How to interprete LS coefficients? For example, the coefficient for variable “rm” measures the average change of Y per room, with all other predictors held fixed.

Note that the result from SLR (regression with just one non-intercept predictor) might be different from the one from MLR. SLR suggests that “age” has a significant negative effect on housing price, while MLR suggests the opposite.

Such seemingly contradictory statements are caused by correlations among predictors. In this case, “age” has strong positive correlation with “crim” and “lsat”, two predictors with negative effect on Y; So in the joint model, the coefficient with “age” turns out to be positive, to correct the negative contribution that has already been introduced to the mdoel through the other two predictors.

```
summary(lm(Y~ age, myData))
```

```
##
## Call:
## lm(formula = Y ~ age, data = myData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.17860 -0.18806 -0.03558  0.17234  1.15041
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.314134   0.027682  119.72   <2e-16 ***
## age         -0.055252   0.004473  -12.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3585 on 504 degrees of freedom
## Multiple R-squared:  0.2324, Adjusted R-squared:  0.2309
## F-statistic: 152.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
round(cor(myData), dig=2)
```

```
##           crim    zn indus  chas   nox    rm   age   dis   rad   tax
## crim      1.00 -0.52  0.74  0.03  0.81 -0.32  0.70 -0.74  0.84  0.81
## zn       -0.52  1.00 -0.66 -0.04 -0.57  0.31 -0.53  0.59 -0.35 -0.31
## indus     0.74 -0.66  1.00  0.08  0.75 -0.43  0.66 -0.73  0.58  0.66
## chas      0.03 -0.04  0.08  1.00  0.08  0.08  0.07 -0.09  0.01 -0.04
## nox       0.81 -0.57  0.75  0.08  1.00 -0.32  0.78 -0.86  0.61  0.67
## rm       -0.32  0.31 -0.43  0.08 -0.32  1.00 -0.28  0.28 -0.21 -0.31
## age       0.70 -0.53  0.66  0.07  0.78 -0.28  1.00 -0.80  0.47  0.54
## dis      -0.74  0.59 -0.73 -0.09 -0.86  0.28 -0.80  1.00 -0.54 -0.60
## rad       0.84 -0.35  0.58  0.01  0.61 -0.21  0.47 -0.54  1.00  0.82
## tax       0.81 -0.31  0.66 -0.04  0.67 -0.31  0.54 -0.60  0.82  1.00
## ptratio  0.45 -0.35  0.45 -0.13  0.34 -0.32  0.38 -0.32  0.40  0.48
## black    -0.48  0.18 -0.33  0.05 -0.38  0.13 -0.29  0.32 -0.41 -0.43
## lstat     0.62 -0.45  0.62 -0.06  0.61 -0.64  0.64 -0.56  0.46  0.53
## Y        -0.57  0.36 -0.55  0.16 -0.52  0.61 -0.48  0.41 -0.43 -0.56
##          ptratio black lstat     Y
## crim        0.45 -0.48  0.62 -0.57
## zn         -0.35  0.18 -0.45  0.36
## indus       0.45 -0.33  0.62 -0.55
## chas       -0.13  0.05 -0.06  0.16
## nox         0.34 -0.38  0.61 -0.52
## rm         -0.32  0.13 -0.64  0.61
## age         0.38 -0.29  0.64 -0.48
## dis        -0.32  0.32 -0.56  0.41
## rad         0.40 -0.41  0.46 -0.43
## tax         0.48 -0.43  0.53 -0.56
## ptratio     1.00 -0.20  0.43 -0.51
## black      -0.20  1.00 -0.36  0.40
## lstat       0.43 -0.36  1.00 -0.83
## Y          -0.51  0.40 -0.83  1.00
```

Partial Regression Coefficients

Check how to retrieve the LS coefficient for "age" using Algorithm 3.1

```
y.star <-  lm(Y ~ ., data = subset(myData, select = -age))$res
age.star <-  lm(age ~ ., data = subset(myData, select = -Y))$res
tmpfit <-  lm(y.star ~ age.star)
```

The LS cofficient for "age" (from `lmfit`) is the same as the one from `tmpfit`. The residuals from the two LS models are also the same.

```
tmpfit$coef
```

```
##  (Intercept)     age.star
## 1.119402e-19 6.403368e-03
```

```
sum((lmfit$res - tmpfit$res)^2)
```

```
## [1] 2.713755e-29
```

F-test

Test a single predictor (in this case, F-test = t-test).

```
lmfit0 <- lm(Y ~ ., data = subset(myData, select = -age))
anova(lmfit0, lmfit)
```

```
## Analysis of Variance Table
##
## Model 1: Y ~ crim + zn + indus + chas + nox + rm + dis + rad + tax + ptratio +
##     black + lstat
## Model 2: Y ~ crim + zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + black + lstat
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    493 19.898
## 2    492 19.828  1  0.069876 1.7338 0.1885
```

Test multiple predictors.

```
lmfit0 <- lm(Y ~ ., data = myData[, -c(1:3)])
anova(lmfit0, lmfit)
```

```
## Analysis of Variance Table
##
## Model 1: Y ~ chas + nox + rm + age + dis + rad + tax + ptratio + black +
##     lstat
## Model 2: Y ~ crim + zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + black + lstat
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    495 19.929
## 2    492 19.828  3   0.10091 0.8346 0.4753
```

Collinearity

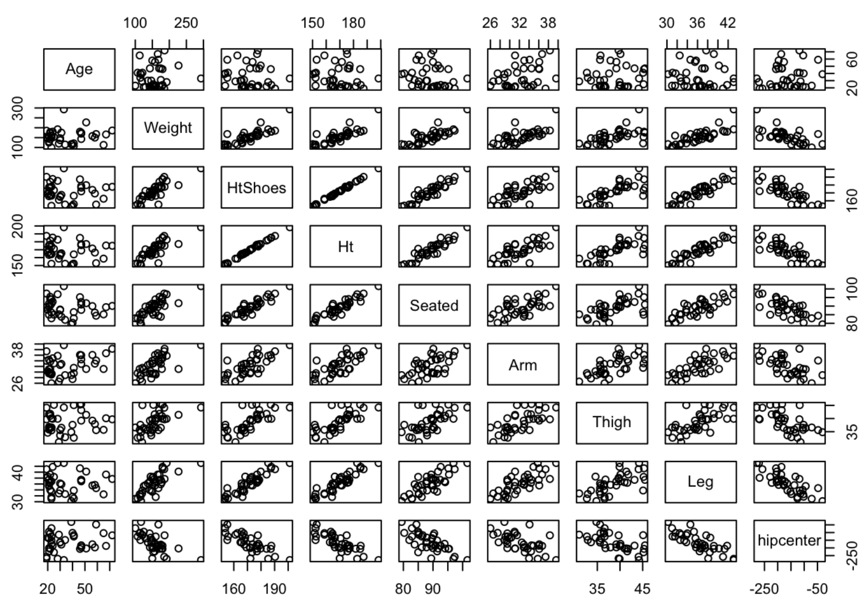
Check the Car Seat Position Data from `faraway` package.

Car drivers like to adjust the seat position for their own comfort. Car designers would find it helpful to know how different drivers will position the seat depending on their size and age. Researchers at the HuMoSim laboratory (http://humosim.org/) at the University of Michigan collected data on 38 drivers.

- Age:
- Weight:
- HtShoes: height with shoes in cm
- Ht: height without shoes in cm
- Seated: seated height in cm
- Arm: lower arm length in cm
- Thigh: thigh length in cm
- Leg: lower leg length in cm
- hipcenter: horizontal distance of the midpoint of the hips from a fixed location in the car in mm

The researchers were interested in determining if a relationship exists between hipcenter and the other variables. Due to the high correlations among the predictors, we see high R-square, significant overall F-test, but no individual variables are significant.

```
library(faraway)
data(seatpos)
pairs(seatpos)
```




```
summary(lm(hipcenter ~ . , data=seatpos))
```

```
##
## Call:
## lm(formula = hipcenter ~ ., data = seatpos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.827 -22.833  -3.678  25.017  62.337
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 436.43213  166.57162   2.620   0.0138 *
## Age           0.77572    0.57033   1.360   0.1843
## Weight        0.02631    0.33097   0.080   0.9372
## HtShoes      -2.69241    9.75304  -0.276   0.7845
## Ht            0.60134   10.12987   0.059   0.9531
## Seated        0.53375    3.76189   0.142   0.8882
## Arm          -1.32807    3.90020  -0.341   0.7359
## Thigh        -1.14312    2.66002  -0.430   0.6706
## Leg          -6.43905    4.71386  -1.366   0.1824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.72 on 29 degrees of freedom
## Multiple R-squared:  0.6866, Adjusted R-squared:  0.6001
## F-statistic:  7.94 on 8 and 29 DF,  p-value: 1.306e-05
```

```
## check pairwise correlation
round(cor(seatpos), dig=2)
```

```
##             Age Weight HtShoes    Ht Seated   Arm Thigh   Leg hipcenter
## Age        1.00   0.08   -0.08 -0.09  -0.17  0.36  0.09 -0.04      0.21
## Weight     0.08   1.00    0.83  0.83   0.78  0.70  0.57  0.78     -0.64
## HtShoes   -0.08   0.83    1.00  1.00   0.93  0.75  0.72  0.91     -0.80
## Ht        -0.09   0.83    1.00  1.00   0.93  0.75  0.73  0.91     -0.80
## Seated    -0.17   0.78    0.93  0.93   1.00  0.63  0.61  0.81     -0.73
## Arm        0.36   0.70    0.75  0.75   0.63  1.00  0.67  0.75     -0.59
## Thigh      0.09   0.57    0.72  0.73   0.61  0.67  1.00  0.65     -0.59
## Leg       -0.04   0.78    0.91  0.91   0.81  0.75  0.65  1.00     -0.79
## hipcenter  0.21  -0.64   -0.80 -0.80  -0.73 -0.59 -0.59 -0.79      1.00
```

If we remove some (almost) redundant variables, the LS results make much more sense.

```
summary(lm(hipcenter ~ Age + Weight + Ht + Seated, data=seatpos))
summary(lm(hipcenter ~ Ht, data=seatpos))
```