Introduction to R and Linear Regression

Alexander McLain

January 26, 2016

bioconductor.

First let's look at a data source from bioconductor. The following will install all core packages and update all installed packages:

```
# install.packages("BiocManager")
# BiocManager::install(c("ALL","limma"))
library("ALL")
data("ALL")
ALL
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 12625 features, 128 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 01005 01010 ... LAL4 (128 total)
##   varLabels: cod diagnosis ... date last seen (21 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
##   pubMedIds: 14684422 16243790
## Annotation: hgu95av2
```

We can looks at the results of molecular biology testing for the 128 samples:

```
ALL$mol.biol
```

```
##   [1] BCR/ABL  NEG      BCR/ABL  ALL1/AF4 NEG      NEG      NEG      NEG
##   [9] NEG      BCR/ABL  BCR/ABL  NEG      E2A/PBX1 NEG      BCR/ABL  NEG
##  [17] BCR/ABL  BCR/ABL  BCR/ABL  BCR/ABL  NEG      BCR/ABL  BCR/ABL  NEG
##  [25] ALL1/AF4 BCR/ABL  ALL1/AF4 NEG      ALL1/AF4 BCR/ABL  NEG      BCR/ABL
##  [33] NEG      BCR/ABL  BCR/ABL  ALL1/AF4 NEG      BCR/ABL  BCR/ABL  BCR/ABL
##  [41] NEG      E2A/PBX1 BCR/ABL  NEG      NEG      NEG      BCR/ABL  p15/p16
##  [49] ALL1/AF4 BCR/ABL  BCR/ABL  NEG      E2A/PBX1 NEG      NEG      NEG
##  [57] BCR/ABL  BCR/ABL  NEG      NEG      ALL1/AF4 NEG      ALL1/AF4 NEG
##  [65] BCR/ABL  NEG      NEG      NEG      NEG      NEG      BCR/ABL  ALL1/AF4
##  [73] BCR/ABL  NEG      E2A/PBX1 NEG      BCR/ABL  BCR/ABL  NEG      NEG
##  [81] NEG      NEG      BCR/ABL  NEG      BCR/ABL  BCR/ABL  BCR/ABL  ALL1/AF4
##  [89] NEG      NEG      BCR/ABL  NEG      BCR/ABL  BCR/ABL  E2A/PBX1 NEG
##  [97] NUP-98   NEG      NEG      NEG      NEG      NEG      NEG      NEG
## [105] NEG      NEG      NEG      NEG      NEG      NEG      NEG      NEG
## [113] NEG      NEG      NEG      NEG      NEG      NEG      NEG      NEG
## [121] NEG      NEG      NEG      NEG      NEG      NEG      NEG      NEG
```

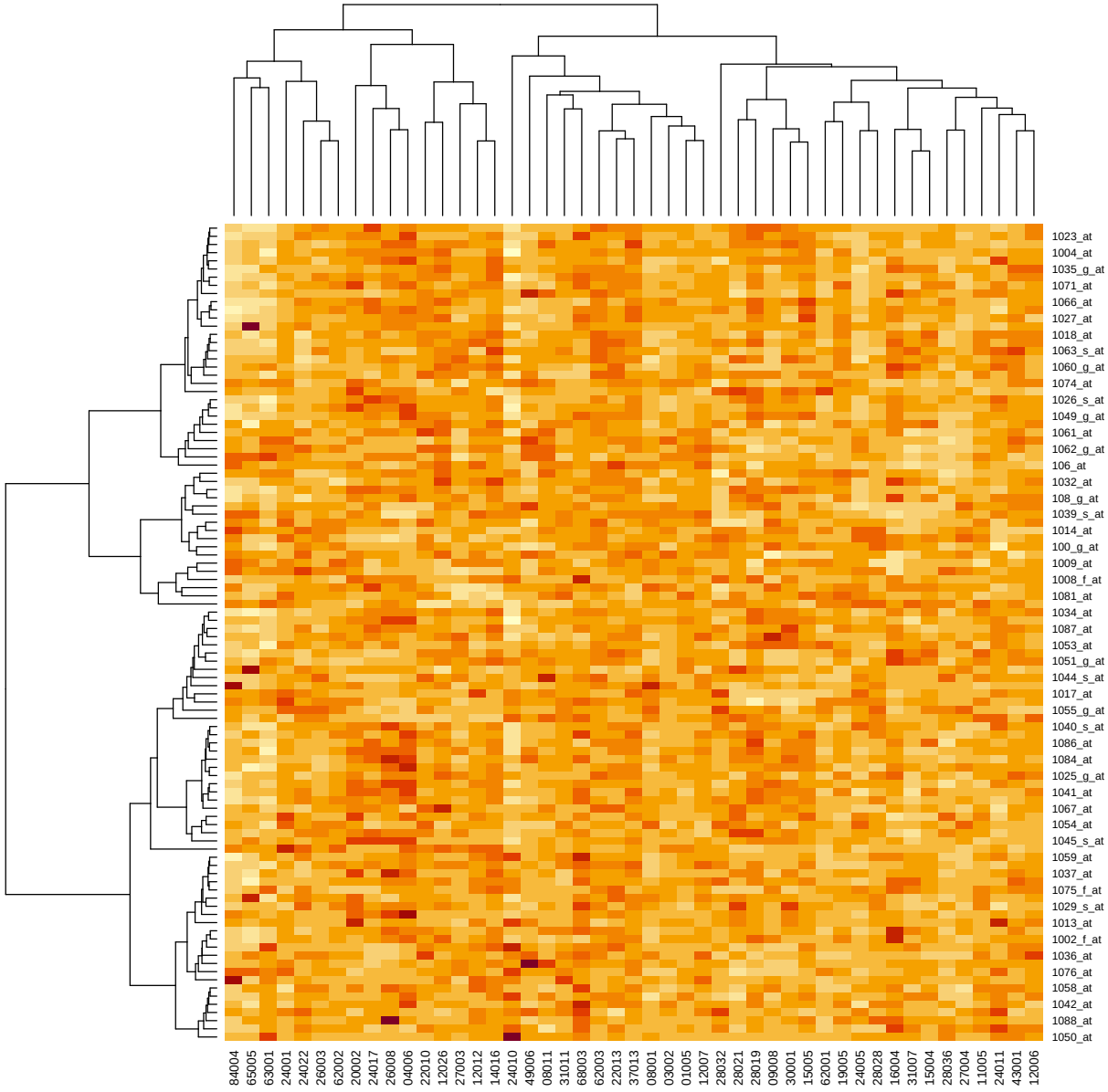
```
## Levels: ALL1/AF4 BCR/ABL E2A/PBX1 NEG NUP-98 p15/p16
```

For the purposes of this example, we are only interested in these two subgroups, so we will create a filtered version of the dataset using this as a selection criteria:

```
eset <- ALL[, ALL$mol.biol %in% c("BCR/ABL", "ALL1/AF4")]
```

The resulting variable, eset, contains just 47 samples - each with the full 12,625 gene expression levels.

```
heatmap(exprs(eset[1:100,]))
```



Below we're going to do some basic unadjusted variable selection and re-plot the heat map.

```
library(limma)
```

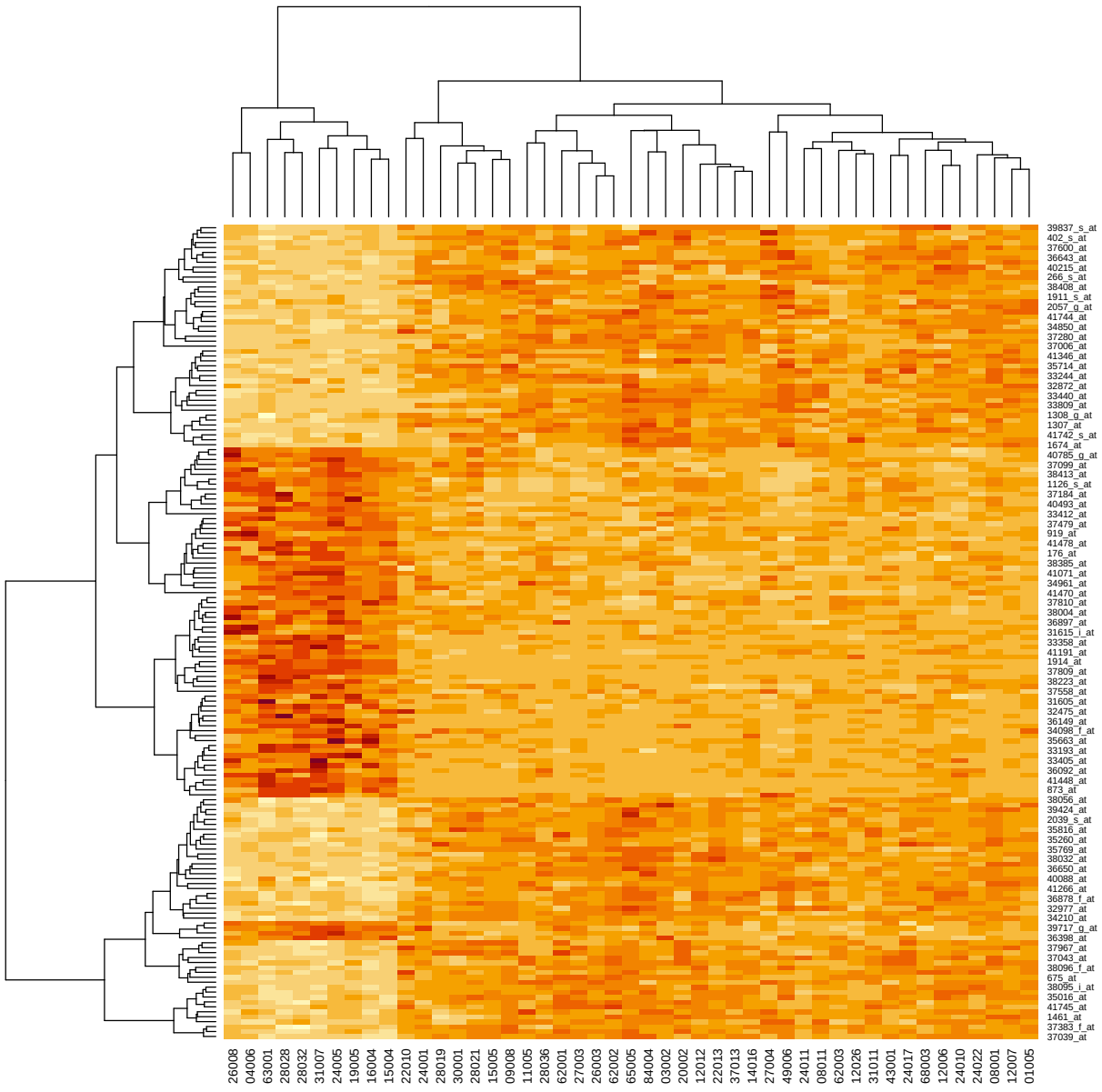
```
##
## Attaching package: 'limma'
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

```
f <- factor(as.character(eset$mol.biol))
design <- model.matrix(~f)
fit <- eBayes(lmFit(eset,design))

selected  <- p.adjust(fit$p.value[, 2]) <0.05
esetSel <- eset [selected, ]

heatmap(exprs(esetSel))
```



Some other good packages to be familar with are: `plyr`, `ggthemes`, `ggplot2`, `data.table`, `dplyr`, `Biobase`, `GEOmetadb`.

Body fat example

This example will use the bodyfat data from the textbook. First we will read in the data, then look at some summaries.

```
bf_dat <- read.csv("bodyfat2.csv")
bf_df <- data.frame(bf_dat)
head(bf_df)
```

density	bodyfat	age	weight	height	neck	chest	abdomen	hip	thigh	knee	ankle	biceps	forearm	wrist
1.0708	12.3	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
1.0502	20.9	24	210.25	74.75	39.0	104.5	94.4	107.8	66.0	42.0	25.6	35.7	30.6	18.8

Second, we'll look at the correlation matrix of the data:

```
round(cor(bf_df),2)
```

	density	bodyfat	age	weight	height	neck	chest	abdomen	hip	thigh	knee	ankle	biceps	forearm	wrist
density	1.00	-1.00	-0.29	-0.61	0.02	-0.49	-0.70	-0.81	-0.62	-0.56	-0.51	-0.27	-0.49	-0.36	-0.35
bodyfat	-1.00	1.00	0.29	0.61	-0.03	0.49	0.70	0.81	0.63	0.56	0.51	0.27	0.49	0.36	0.35
age	-0.29	0.29	1.00	-0.01	-0.25	0.11	0.18	0.23	-0.05	-0.20	0.02	-0.11	-0.04	-0.09	0.21
weight	-0.61	0.61	-0.01	1.00	0.49	0.83	0.89	0.89	0.94	0.87	0.85	0.61	0.80	0.63	0.73
height	0.02	-0.03	-0.25	0.49	1.00	0.32	0.23	0.19	0.37	0.34	0.50	0.39	0.32	0.32	0.40
neck	-0.49	0.49	0.11	0.83	0.32	1.00	0.78	0.75	0.73	0.70	0.67	0.48	0.73	0.62	0.74
chest	-0.70	0.70	0.18	0.89	0.23	0.78	1.00	0.92	0.83	0.73	0.72	0.48	0.73	0.58	0.66
abdomen	-0.81	0.81	0.23	0.89	0.19	0.75	0.92	1.00	0.87	0.77	0.74	0.45	0.68	0.50	0.62
hip	-0.62	0.63	-0.05	0.94	0.37	0.73	0.83	0.87	1.00	0.90	0.82	0.56	0.74	0.55	0.63
thigh	-0.56	0.56	-0.20	0.87	0.34	0.70	0.73	0.77	0.90	1.00	0.80	0.54	0.76	0.57	0.56
knee	-0.51	0.51	0.02	0.85	0.50	0.67	0.72	0.74	0.82	0.80	1.00	0.61	0.68	0.56	0.66
ankle	-0.27	0.27	-0.11	0.61	0.39	0.48	0.48	0.45	0.56	0.54	0.61	1.00	0.48	0.42	0.57
biceps	-0.49	0.49	-0.04	0.80	0.32	0.73	0.73	0.68	0.74	0.76	0.68	0.48	1.00	0.68	0.63
forearm	-0.36	0.36	-0.09	0.63	0.32	0.62	0.58	0.50	0.55	0.57	0.56	0.42	0.68	1.00	0.59
wrist	-0.35	0.35	0.21	0.73	0.40	0.74	0.66	0.62	0.63	0.56	0.66	0.57	0.63	0.59	1.00

Third, we'll fit a simple linear model to the data

```
bf_mod <- lm(bodyfat~age + weight + height + neck + chest + abdomen +
               hip + thigh + knee + ankle + biceps + forearm + wrist,data = bf_df)
```

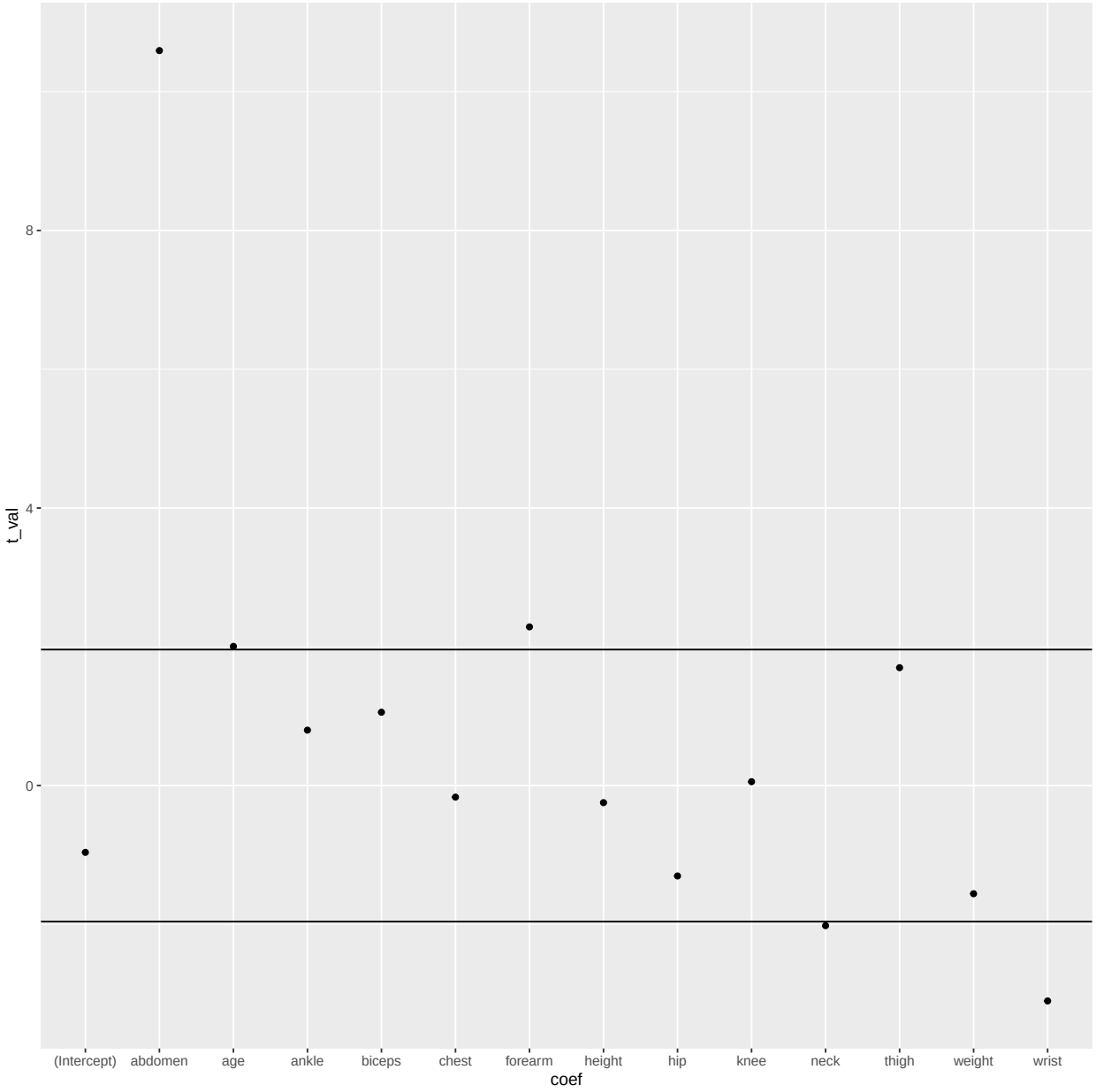
```
summary(bf_mod)
```

```
##
## Call:
## lm(formula = bodyfat ~ age + weight + height + neck + chest +
##     abdomen + hip + thigh + knee + ankle + biceps + forearm +
##     wrist, data = bf_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1966  -2.8824  -0.1111   3.1901   9.9979
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -21.35323   22.18616  -0.962  0.33680
## age           0.06457    0.03219   2.006  0.04601 *
## weight       -0.09638    0.06185  -1.558  0.12047
## height       -0.04394    0.17870  -0.246  0.80599
## neck         -0.47547    0.23557  -2.018  0.04467 *
## chest        -0.01718    0.10322  -0.166  0.86792
## abdomen       0.95500    0.09016  10.592  < 2e-16 ***
## hip          -0.18859    0.14479  -1.302  0.19401
## thigh         0.24835    0.14617   1.699  0.09061 .
## knee          0.01395    0.24775   0.056  0.95516
## ankle         0.17788    0.22262   0.799  0.42505
## biceps        0.18230    0.17250   1.057  0.29166
## forearm       0.45574    0.19930   2.287  0.02309 *
## wrist        -1.65450    0.53316  -3.103  0.00215 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.309 on 238 degrees of freedom
## Multiple R-squared:  0.7486, Adjusted R-squared:  0.7348
## F-statistic:  54.5 on 13 and 238 DF,  p-value: < 2.2e-16
```

```
t_vals <- data.frame(t_val = bf_mod$coefficients/(coef(summary(bf_mod))[,2]),
                     coef = names(bf_mod$coefficients))
```

Plot the resulting T-values:

```
ggplot(data=t_vals,aes(x=coef,y=t_val)) +geom_point() + geom_hline(yintercept=c(-1.96,1.96))
```




```
RSS_n <- mean(bf_mod$residuals^2)
RSS_n_p <- sum(bf_mod$residuals^2)/(
  nrow(bf_df) - length(bf_mod$coefficients))
```

Here

$$RSS\_n = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}$$

while

$$RSS\_n\_p = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n-p}$$

Let see how that compares to the CV estimate. To do this we will:

- create a function that will do K-fold CV sampling of the data ($K = 2, 3, \ldots, n$).
- execute a linear model for each of the K-fold samples

- estimate the prediction error for each of the K-fold samples

Here is the function to do the K-fold sampling:

```
CV_sampl <- function(data,K){
  n <- length(data[,1])
  L <- n/K
  if(is.integer(L)){
    cv_ids <- rep(1:K,each=L)
    }
  if(!is.integer(L)){
    cv_ids <- rep(1:(K-1),each=ceiling(L))
    cv_ids <- c(cv_ids,rep(K,n-(K-1)*ceiling(L)))
    }
  ids <- sample(cv_ids,n)
  Y <- list(data=data,ids=ids)
  return(Y)
}
```

Let's see it work.

```
CV_ids <- CV_sampl(bf_df,10)
CV_ids$ids[1:20]
```

```
##  [1]  4  3  2  4  9  7  9 10  1 10  6  7  4  1  9  3  5  1  2  2
```

Now to do the CV for each model:

```
#How many folds:
CV <- c(3,5,10,20,length(bf_df[,1]))

#Set the seed so we can replicate
set.seed(4)
PE_est <- c(RSS_n,RSS_n_p)
for(k in CV){
  #Get which group each subject is in.
  ids <-  CV_sampl(bf_df,k)$ids
  t_PE_est <- NULL
  for(j in 1:k){
    #Get jth leaning and test datasets, and estimate LM
    learning_data <- bf_df[ids!=j,]
    test_data <- bf_df[ids==j,]
    bf_mod_CV <- lm(bodyfat~age + weight + height + neck + chest +
                      abdomen + hip + thigh + knee + ankle + biceps +
                      forearm + wrist,data = learning_data)
    #Predict Y_hat for the new data.
    new_Yhat <- predict(bf_mod_CV,test_data)
    #Get the squared errors.
    new_RSS_n <- (test_data$bodyfat - new_Yhat)^2
    t_PE_est <- c(t_PE_est,new_RSS_n)
  }
  PE_est <- c(PE_est,mean(t_PE_est))
}
PE_est <- t(matrix(PE_est))
```

Now let's see all of the results:

	RSS_n	RSS_n_p	PE_CV 3	PE_CV 5	PE_CV 10	PE_CV 20	PE_CV 252
Estimate	17.5399	18.5717	21.3079	19.7441	20.8561	20.6022	20.2948

Testing Speed

In this portion we are going to test the speed of `R` in doing a simple linear regression by column.

First, let's read in the data:

```
Ex_dat <- read.csv("Example_data.csv")

Y <- Ex_dat[,1]
Z <- as.matrix(Ex_dat[,-1])
M <- dim(Z)[2]
dim(Z)
```

```
## [1]   400 10000
```

Now we're going to regress Y as a function of each Z_j while adjusting for the sum over all Z_k's. That is, we going to fit the model

$$Y_i = \beta_0 + \beta_{1j} Z_{ij} + \beta_2 (\sum_{k=1}^{10000} Z_{ik}) + \epsilon_i$$

This requires running $10,000$ separate models.

```
coef_mat <- NULL
StdErr <- NULL
Sigma2 <- NULL
Xp <- apply(Z,1,mean)

system.time(for(i in 1:M){
  t1 <- lm(Y~Z[,i]+Xp)
  coef_mat <- rbind(coef_mat,coef(t1))
  StdErr <- rbind(StdErr, sqrt(diag(vcov(t1))))
  Sigma2 <- c(Sigma2, sigma(t1)^2 )
})
```

```
##    user  system elapsed
##  12.132   0.422  12.667
```

Now, we're going to run this again using a package called `RcppArmadillo`. The function `LM_by_col.cpp` is available on the course website. This function does the same basic operation as the code above, but is coded in c++.

First, we have to compile the `LM_by_col.cpp` function so it is available.

```
library(Rcpp)
library(RcppArmadillo)
sourceCpp("LM_by_col.cpp")
```

Now we'll run all the models and compare the computation time.

```
system.time(LRcpp <- LM_by_col(Y, Z))
```

```
##    user  system elapsed
##   0.249   0.020   0.270
```

```
table(round(LRcpp$Coefficients,7) == round(coef_mat,7) )
```

TRUE
30000

```
table(round(LRcpp$StdErr,7) == round(StdErr,7) )
```

TRUE
30000