# BKN 599 - Final Exam

FirstName LastName

## Data Description

We will use the ICARE rehabilitation data (Winstein, Jama 2016). The goal is to predict the in an arm and hand impairment score (Upper extremity Fugl Meyer) after training FM2 as a function of multiple baseline variables including baseline FM1.

Note 1: these data are not publicly available, so PLEASE DO NOT DISTRIBUTE outside of this class.
Note 2: the actual data set has actually many more observations but we deleted the rows with missing data

### Variables

Y = `FM2` is the dependent variable

X = all baseline/demographic data are the independent variables - the meaning of the variables are given below

`FM1` = hand Fugl Meyer at baseline

`FM2` = Fugl Meyer after training

`CHAMchallenge` = a motivation question in the confidence in arm and hand test

`ave_CAHM` = average confidence in arm and hand test

`EQ_index` = generic health status questionnaire

`SIS_hand` = Stroke impact scale of hand function

`RNLIadj` = A reintegration to normal living index

`NIHtot` = NIH stroke impact scale ; A brief assessment of physical function post-stroke

`log_mean_time_MA_PA` = Time on the Wolf motor function; more affected hand

`log_mean_time_LA_PA` = Time on the Wolf motor function; less affected hand

`grip_MA` = grip strength, more affected

`grip_LA` = grip strength, less affected

`dose_hours` = actual dose of training

`onset_to_rand` = time since stroke at start of trial

`age_at_rand` = age at start of trial

`old_stroke` = whether participants had a stroke or not prior

# Required Libraries

```
library(ggplot2)
library(dplyr)
library(glmnet)
library(tidyr)
library(magrittr)
library(tree)
library(randomForest)
```

# Load the data

```
load("data_final.Rda")
str(data_final)
```

```
## 'data.frame':    214 obs. of  17 variables:
##  $ pid                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ CHAMchallenge      : int  100 50 50 100 90 100 20 100 90 75 ...
##  $ ave_CAHM           : num  27 10.5 52.5 94.5 23.5 ...
##  $ EQ_index           : num  0.167 0.705 0.748 0.843 0.813 0.589 0.827 0.832 0.437 0.843 ...
##  $ SIS_hand           : int  0 0 10 40 20 0 30 65 25 45 ...
##  $ RNLIadj            : num  46.4 39.1 61.8 85.5 78.2 ...
##  $ NIHtot             : int  4 5 6 3 4 8 8 1 3 4 ...
##  $ log_mean_time_MA_PA: num  3.465 3.989 0.987 0.607 2.278 ...
##  $ log_mean_time_LA_PA: num  1.111 1.54 0.82 0.385 0.548 ...
##  $ grip_MA            : num  2 1.33 8.67 36.67 4.67 ...
##  $ grip_LA            : num  36.7 18.7 32.3 52 53.3 ...
##  $ dose_hours         : int  32 31 28 0 0 0 30 31 0 33 ...
##  $ onset_to_rand      : int  79 43 70 17 63 75 52 15 14 36 ...
##  $ age_at_rand        : num  57.9 74.4 61.7 34.8 58.1 ...
##  $ old_stroke         : num  1 0 0 1 0 1 0 0 0 1 ...
##  $ FM1                : num  34 28 39 56 39 28 44 55 49 32 ...
##  $ FM2                : num  42 45 36 64 45 36 46 60 62 42 ...
##  - attr(*, "na.action")= 'omit' Named int [1:5] 22 36 92 169 213
##   ..- attr(*, "names")= chr [1:5] "22" "36" "92" "169" ...
```

To get rid of `pid`:

```
data_final = data_final[,-1]
```

# Question 1 (10 points)

### Q1. Interpreting and plotting a linear model output (10 points)

Create a binary variable by comparing `log_mean_time_MA_PA` to its mean. Make a regression model with two predictors: `FM1` and this new binary variable and their interactions. Compute the two slopes and intercepts from the model output. Using these values plot the regression lines of FM2 as a function of FM1 for the two

cases when log_mean_time_MA_PA is high and low. Add the data for the two cases with different marker colors. Add a complete legend.

*Note: Remember that you should create only one plot, so do not split the data for separate subplots.*

```
data <- data_final %>%
  mutate(log_mean_time_MA_PA_binary = factor(ifelse(
    log_mean_time_MA_PA>mean(log_mean_time_MA_PA),
    "High", "Low"))
    )

str(data)
```

```
## 'data.frame':    214 obs. of  17 variables:
##  $ CHAMchallenge             : int  100 50 50 100 90 100 20 100 90 75 ...
##  $ ave_CAHM                  : num  27 10.5 52.5 94.5 23.5 ...
##  $ EQ_index                  : num  0.167 0.705 0.748 0.843 0.813 0.589 0.827 0.832 0.437 0.843 ...
##  $ SIS_hand                  : int  0 0 10 40 20 0 30 65 25 45 ...
##  $ RNLIadj                   : num  46.4 39.1 61.8 85.5 78.2 ...
##  $ NIHtot                    : int  4 5 6 3 4 8 8 1 3 4 ...
##  $ log_mean_time_MA_PA       : num  3.465 3.989 0.987 0.607 2.278 ...
##  $ log_mean_time_LA_PA       : num  1.111 1.54 0.82 0.385 0.548 ...
##  $ grip_MA                   : num  2 1.33 8.67 36.67 4.67 ...
##  $ grip_LA                   : num  36.7 18.7 32.3 52 53.3 ...
##  $ dose_hours                : int  32 31 28 0 0 0 30 31 0 33 ...
##  $ onset_to_rand             : int  79 43 70 17 63 75 52 15 14 36 ...
##  $ age_at_rand               : num  57.9 74.4 61.7 34.8 58.1 ...
##  $ old_stroke                : num  1 0 0 1 0 1 0 0 0 1 ...
##  $ FM1                       : num  34 28 39 56 39 28 44 55 49 32 ...
##  $ FM2                       : num  42 45 36 64 45 36 46 60 62 42 ...
##  $ log_mean_time_MA_PA_binary: Factor w/ 2 levels "High","Low": 1 1 2 2 1 1 1 2 2 2 ...
```

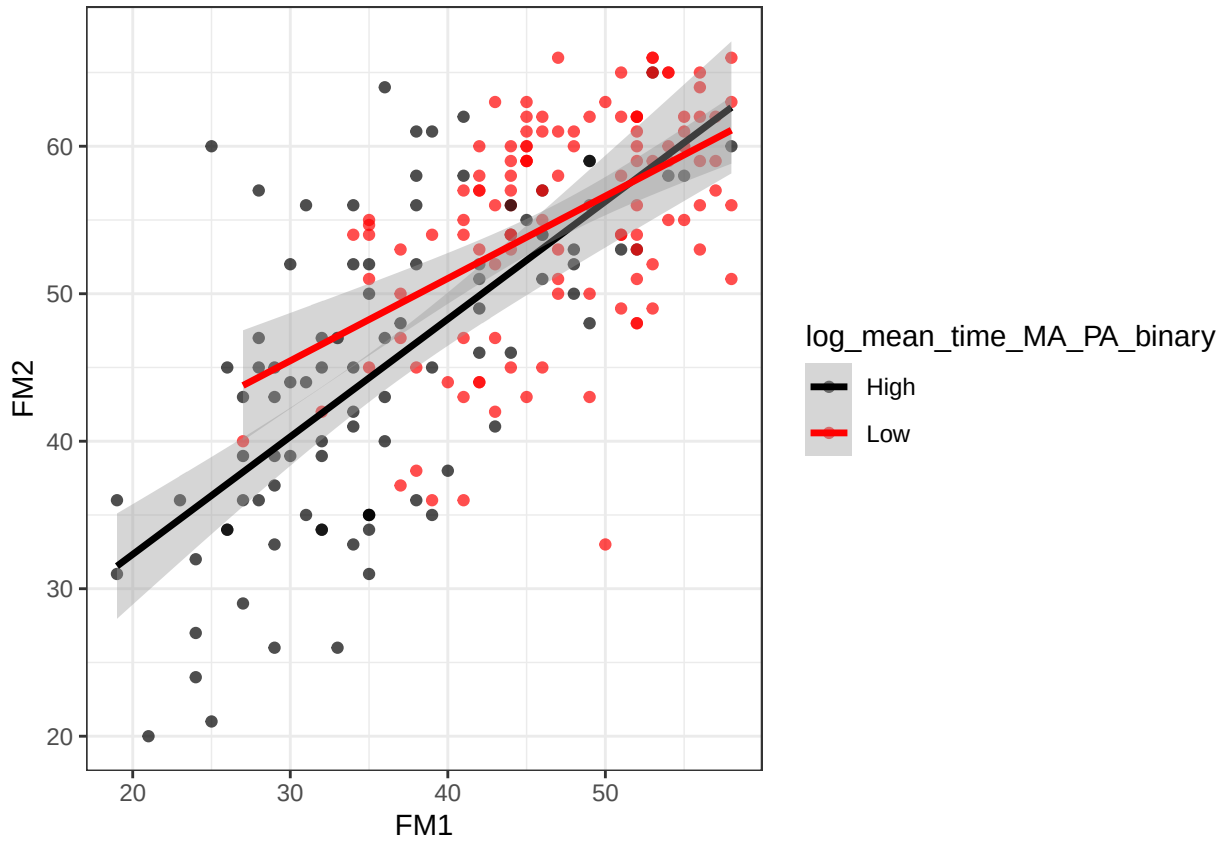
```
lm.fit <- lm(FM2 ~ FM1*log_mean_time_MA_PA_binary, data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = FM2 ~ FM1 * log_mean_time_MA_PA_binary, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.6222  -4.6434   0.8322   5.0729  23.6830
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      16.38011    3.15245   5.196 4.81e-07 ***
## FM1                               0.79748    0.08543   9.335  < 2e-16 ***
## log_mean_time_MA_PA_binaryLow    12.32239    5.67292   2.172   0.0310 *
## FM1:log_mean_time_MA_PA_binaryLow -0.23908    0.13092  -1.826   0.0693 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.24 on 210 degrees of freedom
```

```
## Multiple R-squared:  0.5112, Adjusted R-squared:  0.5042
## F-statistic: 73.21 on 3 and 210 DF,  p-value: < 2.2e-16
```

Now if we were to ignore the p-values and draw the regression lines we would simply get:

```
ggplot(data, aes(x=FM1, y=FM2, col=log_mean_time_MA_PA_binary)) +
  geom_point(alpha=.7) +
  geom_smooth(method = "lm", formula = y~x, size = 1.2) +
  scale_color_manual(values = c("black", "red")) +
  theme_bw()
```



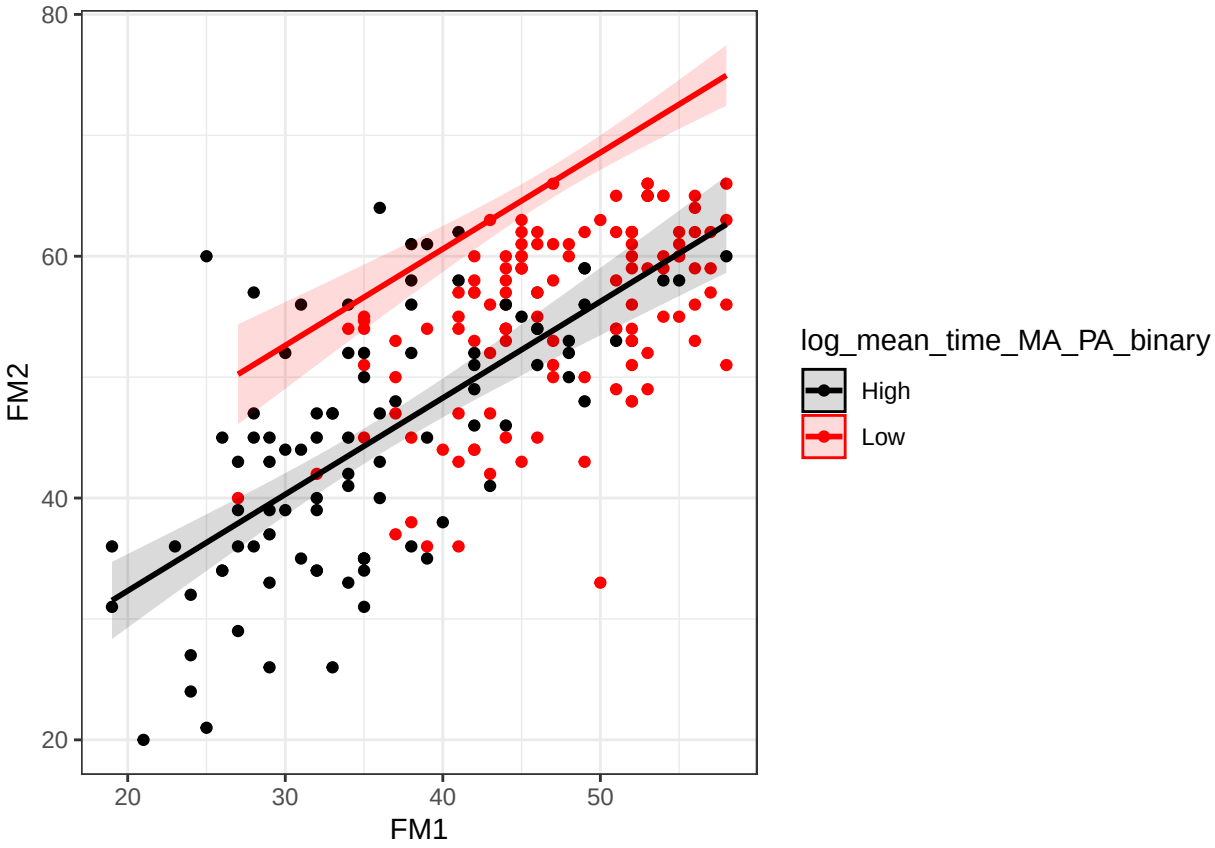

However, we see in the linear model that the slope interaction term is not significant; thus we will see what the new curves are going to look like if we set `FM1:log_mean_time_MA_PA_binaryLow = 0` or equivalently `lm.fit$coefficients[4] = 0`.

```
# Get the coefficients and zero the ones that are not significant
# (p-values > .05)
coeff <- coef(summary(lm.fit))
lm.fit$coefficients[coeff[,4]>.05] = 0

predslm = predict(lm.fit, interval = "confidence")
datlm = cbind(data, predslm)


ggplot(datlm, aes(FM1, y = FM2, color = log_mean_time_MA_PA_binary) ) +
  geom_point() +
```

```
geom_ribbon( aes(ymin = lwr, ymax = upr,
                 fill = log_mean_time_MA_PA_binary, color = NULL), alpha = .15) +
geom_line( aes(y = fit), size = 1) +
scale_color_manual(values = c("black", "red")) +
scale_fill_manual(values = c("black", "red")) +
theme_bw()
```




## Question 2 (10 + 20 points)

**Q2. Estimating regression coefficients CI with a manually implemented bootstrap. (30 points:: 10 + 20)**

1. Estimate 95% confidence interval via the "typical" least square regression.

A. `set.seed(123)`. Fit a lasso model (with optimal lambda according to 1 SE) to predict `FM2` from the data.
B. Perform a "typical" regression model (`lm()`) to predict `FM2` using only the lasso-selected variables. Report the 95% confidence interval of the estimated coefficients using the R output (which uses the formulas in the text book).

2. Estimate 95% confidence interval via the bootstrap.

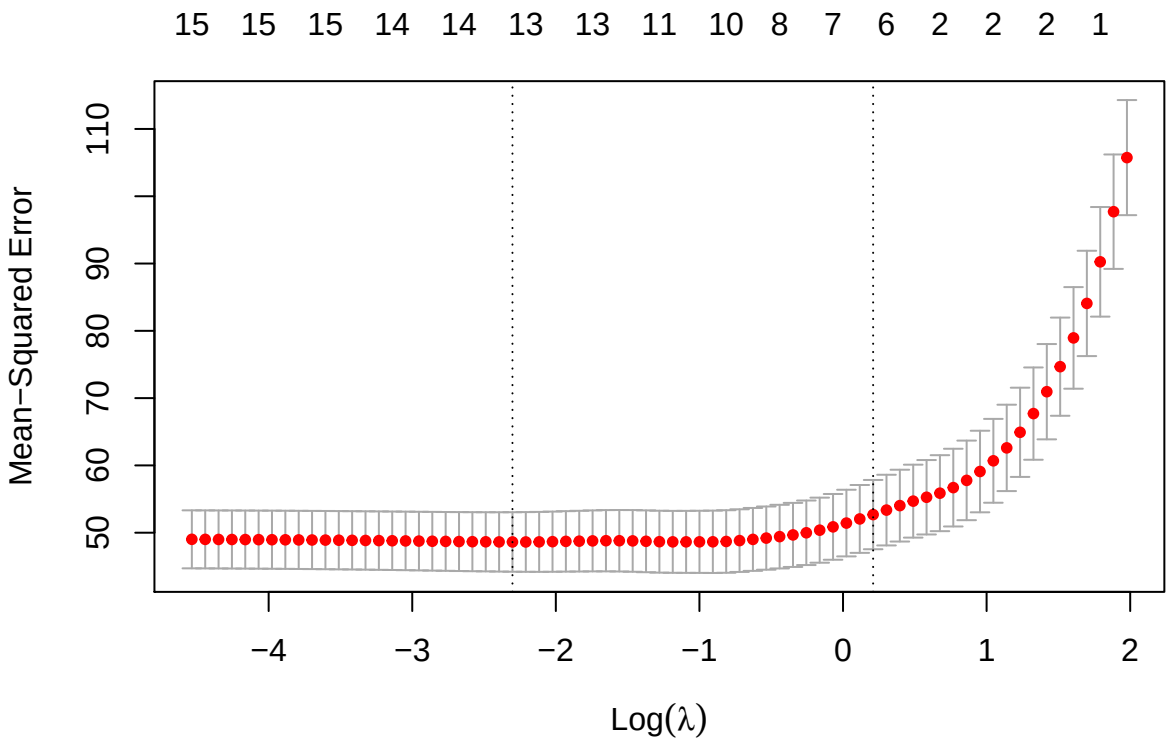
A. `set.seed(123)`. Now implement a bootstrap "manually" (i.e., write the code to do the sampling without any boot function from R) to estimate the 95% confidence interval of the parameters for the lasso-selected variables. Plot the histogram of the coefficient estimate distribution for `FM1`. Use percentiles to get the 95% CIs for all parameters. Make sure your results do not depend (too much) on your number of samples by generating many samples (report results for few, many, and even more samples). *Hint: Remember that in resampling for boostrapping, replacement is allowed, so use `sample(~,~, replace=T)`.

B. Compare the coefficient estimates and the CIs from both methods - `lm()` and regression and from the bootstrap. Discuss similarities/differences in your R notebook (note that any differences may be understood by performing regression diagnostics – remember the assumptions of regression `and notably how are computed the regression coefficient SE for regression using formulas in the textbook`)

### Q2.1 - A

```
x = model.matrix ( FM2~.,data_final)
y= data_final$FM2
```

```
# Finding best lambda
set.seed(123)
lasso_reg = cv.glmnet(x, y,alpha = 1)
plot(lasso_reg)
```



```r
bestlam =lasso_reg$lambda.1se

# Finding coefficients for best lambda
out.lasso <- glmnet(x, y, alpha=1)
lasso.coef <- predict (out.lasso , type = "coefficients", s = bestlam)
lasso.coef
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)         33.851305235
## (Intercept)          .
## CHAMchallenge        0.005927197
## ave_CAHM             .
## EQ_index             .
## SIS_hand             .
## RNLIadj              0.015732230
## NIHtot               .
## log_mean_time_MA_PA -1.702129410
## log_mean_time_LA_PA  .
## grip_MA              0.007451039
## grip_LA              .
## dose_hours           .
## onset_to_rand       -0.021135073
## age_at_rand          .
## old_stroke           .
## FM1                  0.463499511
```

```r
coeff<- lasso.coef[3:17,]


selected_vars = names(coeff [ coeff != 0 ] )

lasso_df = data_final[,c(selected_vars,"FM2")]
head(lasso_df)
```

```
##   CHAMchallenge RNLIadj log_mean_time_MA_PA grip_MA onset_to_rand FM1 FM2
## 1           100  46.363               3.465   2.000            79  34  42
## 2            50  39.090               3.989   1.333            43  28  45
## 3            50  61.818               0.987   8.666            70  39  36
## 4           100  85.454               0.607  36.666            17  56  64
## 5            90  78.181               2.278   4.666            63  39  45
## 6           100  61.818               3.499   6.000            75  28  36
```

## Q2.1 - B

```r
# Fitting linear model
lin_mod = lm(FM2~., lasso_df)
summary(lin_mod)
```

```
##
```

```
## Call:
## lm(formula = FM2 ~ ., data = lasso_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.5060  -4.0220   0.6447   4.2456  17.8227
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         30.62471    4.82566   6.346 1.37e-09 ***
## CHAMchallenge        0.03484    0.01864   1.869  0.06300 .
## RNLIadj              0.06663    0.02783   2.394  0.01757 *
## log_mean_time_MA_PA -2.07288    0.69192  -2.996  0.00307 **
## grip_MA              0.05329    0.06164   0.865  0.38830
## onset_to_rand       -0.06550    0.02315  -2.829  0.00513 **
## FM1                  0.45965    0.07894   5.822 2.19e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.868 on 207 degrees of freedom
## Multiple R-squared:  0.5664, Adjusted R-squared:  0.5538
## F-statistic: 45.06 on 6 and 207 DF,  p-value: < 2.2e-16
```

```
# Checking confidence intervals
confint(lin_mod, level=0.95)
```

```
##                           2.5 %      97.5 %
## (Intercept)         21.110967082 40.13845040
## CHAMchallenge       -0.001905412  0.07158818
## RNLIadj              0.011752841  0.12150262
## log_mean_time_MA_PA -3.436996814 -0.70875779
## grip_MA             -0.068237590  0.17482146
## onset_to_rand       -0.111148324 -0.01985430
## FM1                  0.304007562  0.61528250
```

## Q2.2 - A

```
# Bootstrap
set.seed(123)
datalist = matrix(rep(0,10000*7),  ncol = 7)

for(i in 1:10000){
  # sample data
  sample_df = lasso_df[sample(nrow(lasso_df), nrow(lasso_df),replace =TRUE),]
  samp_lin_mod = lm(FM2~., sample_df)
  datalist[i,] =   samp_lin_mod$coefficients
}
# boots_values <- dplyr::bind_rows(datalist)
boot.coeff <- data.frame(datalist) %>%
  set_colnames(names(coefficients(lin_mod)))
# Bootstrap mean and 95p confidence interval
colMeans(boot.coeff)
```

```
##         (Intercept)       CHAMchallenge             RNLIadj log_mean_time_MA_PA
##         30.86551320          0.03485211          0.06571602         -2.09231212
##             grip_MA       onset_to_rand                 FM1
##          0.05108533          -0.06588801          0.45756524
```

```
apply(boot.coeff, 2, function(x){mean(x)+c(-1.96,1.96)*sd(x)})
```

```
##      (Intercept) CHAMchallenge     RNLIadj log_mean_time_MA_PA     grip_MA
## [1,]    21.47917  -0.003738533 0.005905596          -3.4621430 -0.04961539
## [2,]    40.25186   0.073442750 0.125526445          -0.7224813  0.15178605
##      onset_to_rand       FM1
## [1,]   -0.11217759 0.3104196
## [2,]   -0.01959843 0.6047109
```

```
# linear model mean and 95p confidence interval
coef(lin_mod)
```

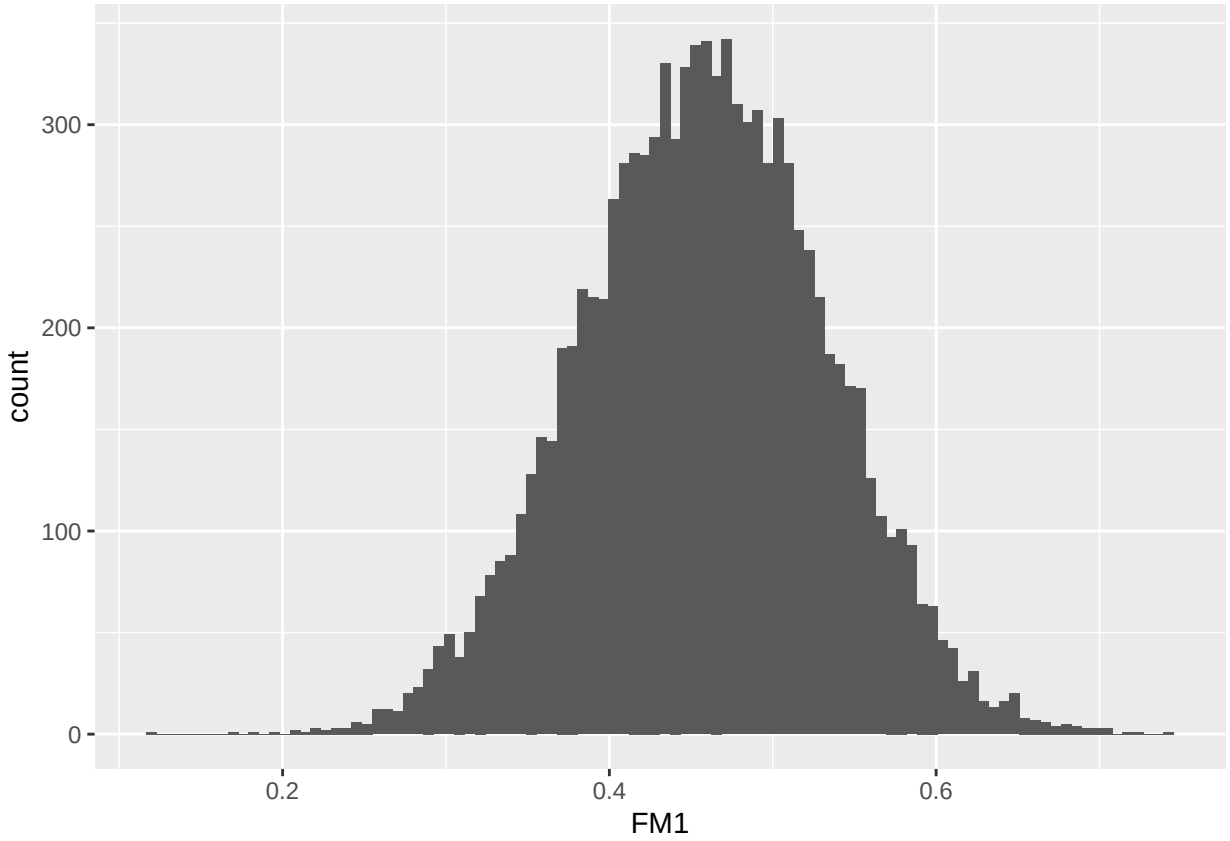
```
##         (Intercept)       CHAMchallenge             RNLIadj log_mean_time_MA_PA
##         30.62470874          0.03484138          0.06662773         -2.07287730
##             grip_MA       onset_to_rand                 FM1
##          0.05329193          -0.06550131          0.45964503
```

```
confint(lin_mod, level=0.95)
```

```
##                            2.5 %      97.5 %
## (Intercept)         21.110967082 40.13845040
## CHAMchallenge       -0.001905412  0.07158818
## RNLIadj              0.011752841  0.12150262
## log_mean_time_MA_PA -3.436996814 -0.70875779
## grip_MA             -0.068237590  0.17482146
## onset_to_rand       -0.111148324 -0.01985430
## FM1                  0.304007562  0.61528250
```

Plotting the histogram of the parameter distribution for FM1 resulted from Bootstrapping:

```
ggplot(boot.coeff)+
  geom_histogram(aes(x=FM1), bins=100)
```




**Q2.2 - B**

# Question 3 (25 + 5)

**Q3. Trees and random forest. (30 points: 25 + 5)**

- Divide the data into a training and a test set of 20% of the data assigned to the test. Use `sample(nrow(.), 0.2*nrow(.))`.

1.

a. Predict `FM2` from the data using a pruned tree via cross-validation. Plot the tree. Discuss your results.
b. Predict `FM2` using bagging. Discuss your results (including variable importance)
c. Predict `FM2` using random forest. Discuss your results.
d. Plot the variable importance for random forest. Compare with the pruned tree and discuss your results.

2. Predict `FM2` using the lasso model of Q2 with the same test set. Then, Compare the MSE on the test set of the four different methods.

3. Predict `FM2` using `a lasso model selected based on cross-validation and using the same test set`. Then, Compare the MSE on the test set of the four different methods.

*Hint: Recall that prediction is about investigating how well a model performs on new data, i.e., the test set here*

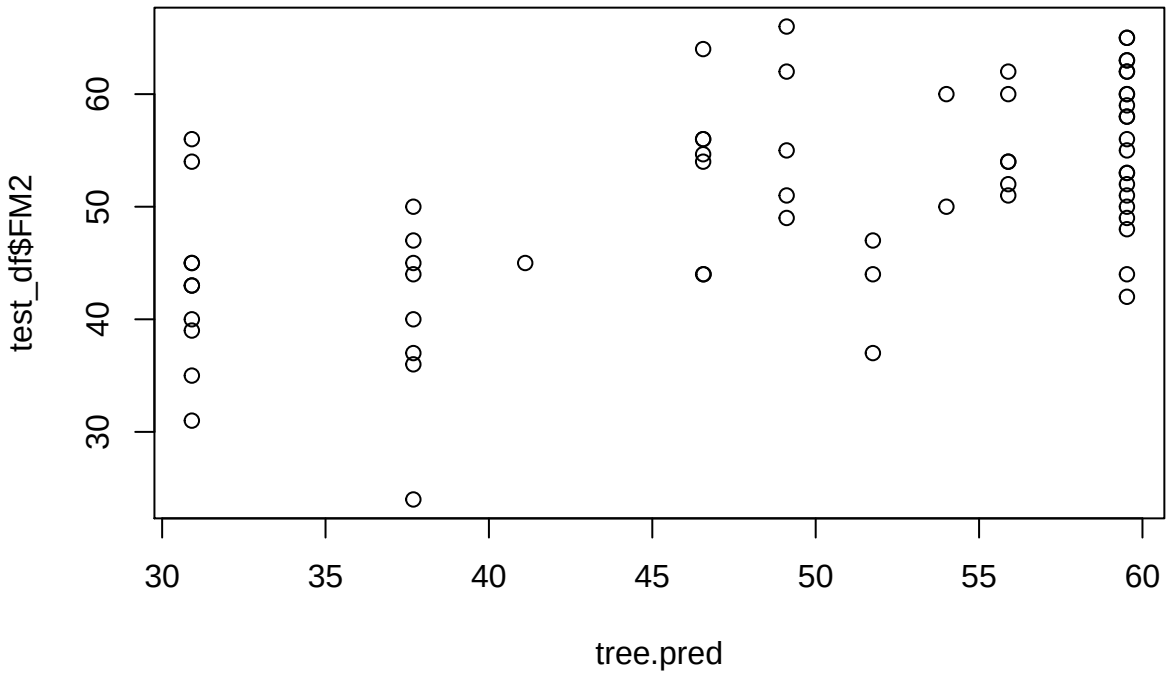
## Q3.1 - A

### Tree-based Regression

First split the data into train/test sets.

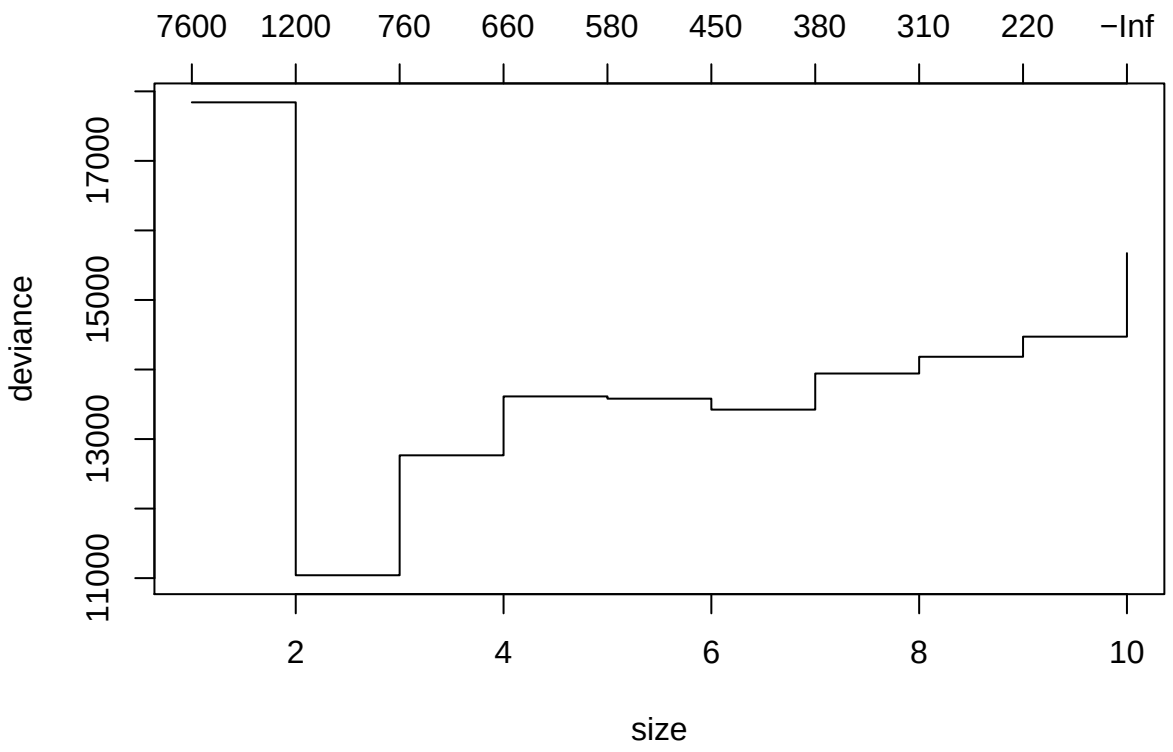
```
# Creating test and train sets
test_index =sample(nrow(data_final), 0.3*nrow(data_final))
test_df = data_final[test_index,]
train_df = data_final[-test_index,]

# Train tree and predict MSE
tree.FM2 =tree(FM2~. , train_df)
tree.pred=predict(tree.FM2 ,test_df )
plot(tree.pred ,test_df$FM2)
```

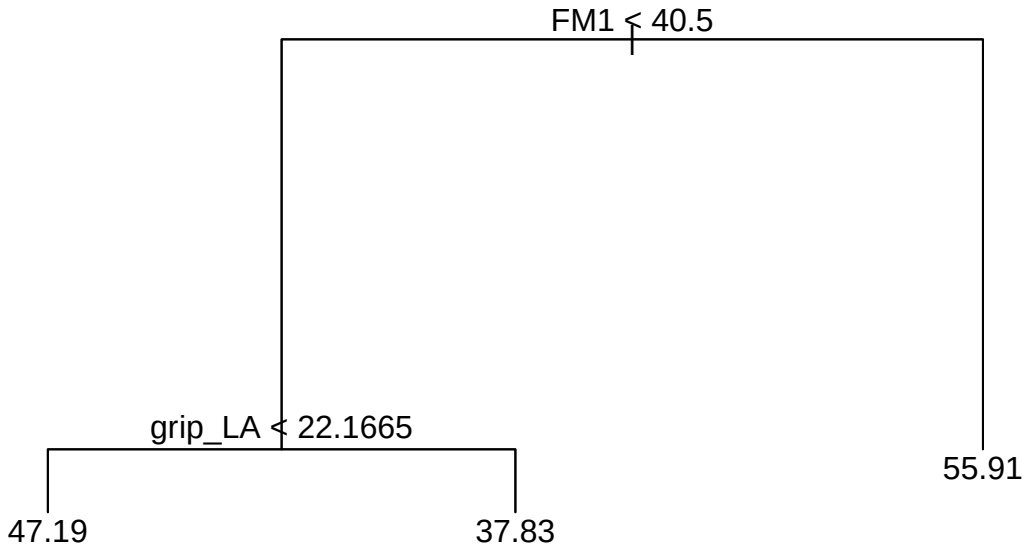



Prune the tree

```
set.seed(123)
cv.FM2 =cv.tree(tree.FM2 )
plot(cv.FM2)
```

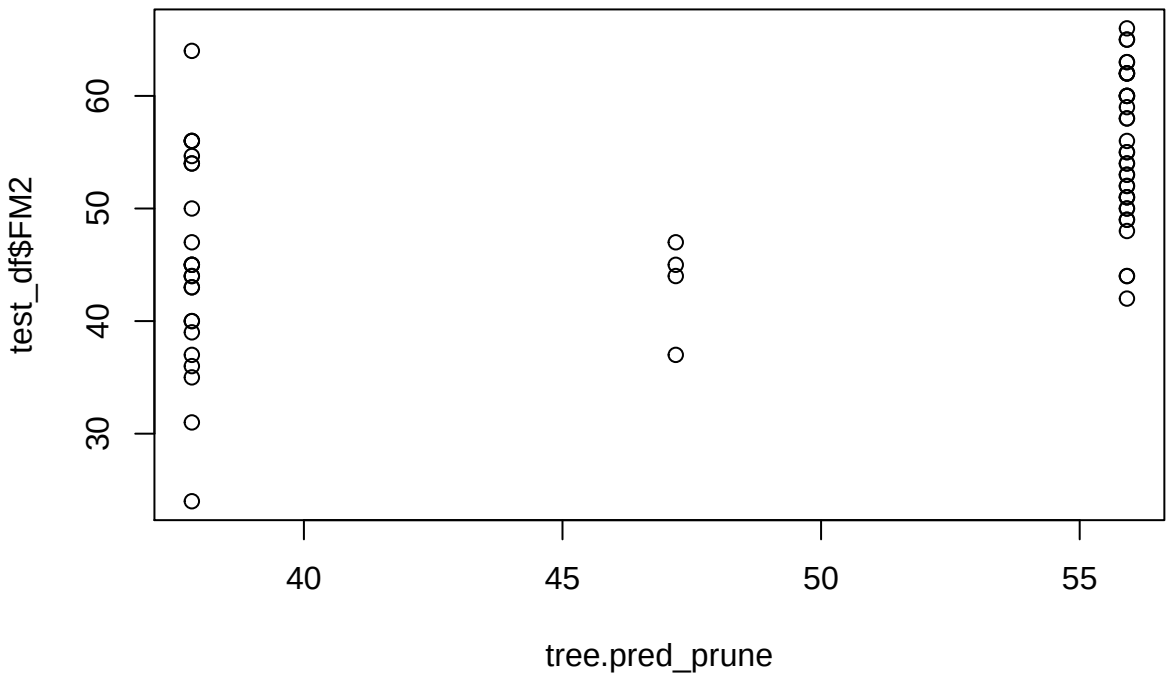



```
prune.FM2 =prune.tree (tree.FM2 ,best =3)
plot(prune.FM2 )
text(prune.FM2 ,pretty =0)
```




Test Error

```
tree.pred_prune =predict(prune.FM2, test_df)
plot(tree.pred_prune,test_df$FM2)
```




```
MSE_pruned_tree = mean((tree.pred_prune-test_df$FM2)^2)
MSE_pruned_tree
```

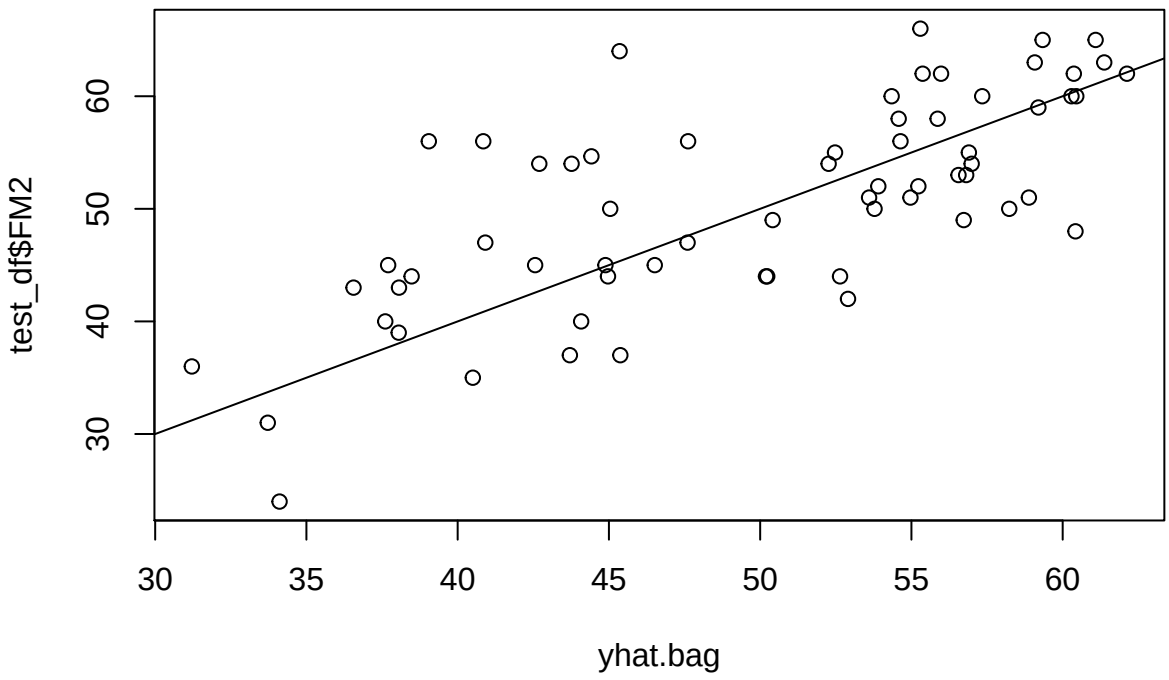
```
## [1] 75.17403
```

### Q3.1 - B

```
# Bagging
set.seed(123)
bag.FM2 =randomForest(FM2~.,data=train_df, mtry=15, importance =TRUE)
bag.FM2
```

```
##
## Call:
##  randomForest(formula = FM2 ~ ., data = train_df, mtry = 15, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 15
##
##           Mean of squared residuals: 60.74286
##                     % Var explained: 47.14
```

```
yhat.bag = predict (bag.FM2 ,newdata =test_df)
plot(yhat.bag , test_df$FM2)
abline (0,1)
```




```
MSE_bagging= mean(( yhat.bag -test_df$FM2)^2)
MSE_bagging
```

```
## [1] 44.24469
```

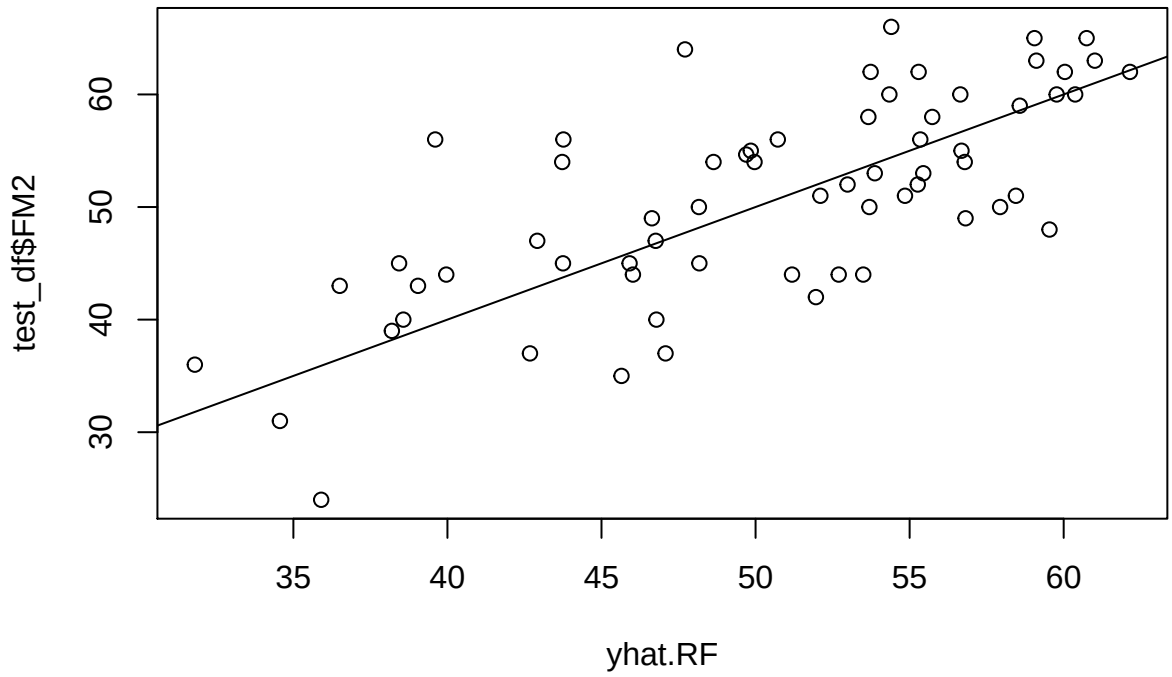
### Q3.1 - C

```
# Random Forest
set.seed(123)
RF.FM2 =randomForest(FM2~.,data=train_df, importance =TRUE)
RF.FM2
```

```
##
## Call:
##  randomForest(formula = FM2 ~ ., data = train_df, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
```

```
##
##           Mean of squared residuals: 56.55222
##                     % Var explained: 50.78
```

```
yhat.RF = predict (RF.FM2 ,newdata =test_df)
plot(yhat.RF , test_df$FM2)
abline (0,1)
```

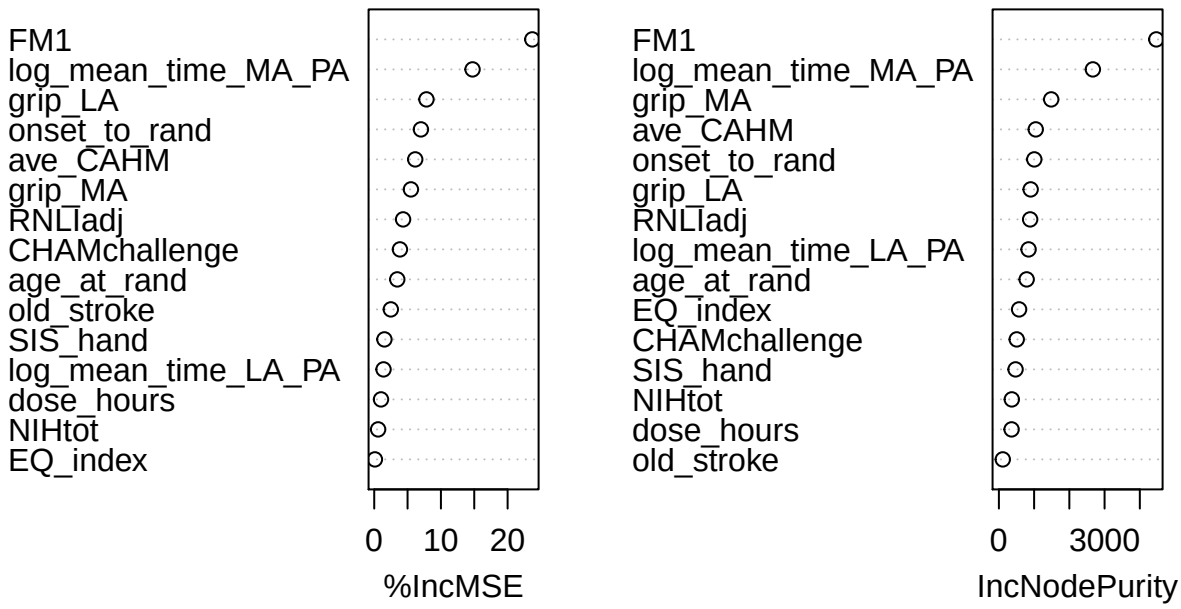



```
MSE_RF= mean(( yhat.RF -test_df$FM2)^2)
MSE_RF
```

```
## [1] 40.95762
```

### Q3.2

```
varImpPlot(RF.FM2)
```



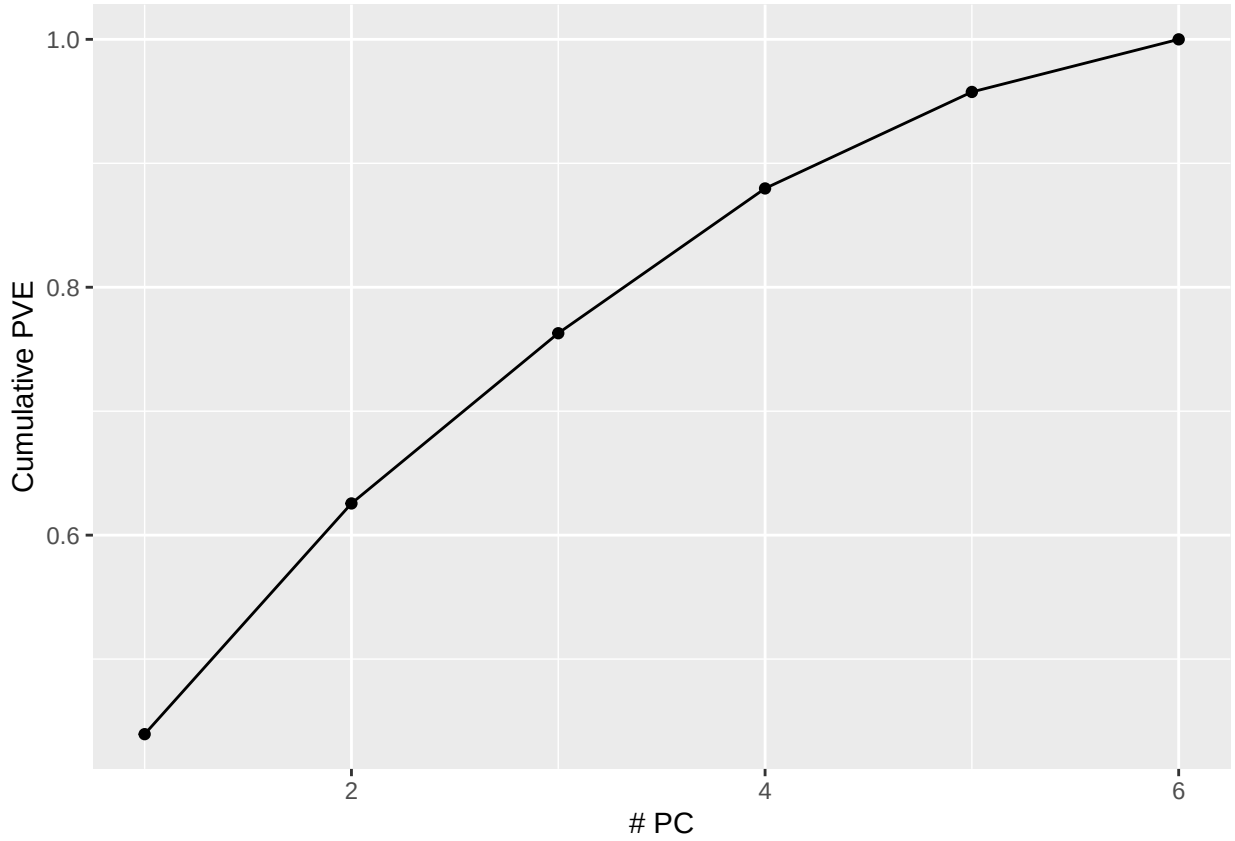

## Question 4 (10 + 10 + 10)

**Q4/ PCA and K-means of the lasso-selected predictor variables. (30 points: 10 + 10 + 10)**

1. Perform a PCA on all predictor variables **selected by the lasso in Q2**. Using a **95%** variable accounted for cut-off, how many PCs do you find? Plot the results with a biplot. Discuss your results.
2. Now, using K = 2, perform a K-means clustering on the predictor variables selected by the lasso.
3. Plot the K-means results in the PC1/PC2 axes. Discuss your results.

```
pr.out = prcomp(select(lasso_df, -"FM2"),scale = TRUE)
variance_PCS =(pr.out$sdev)^2/sum((pr.out$sdev)^2)
cumsum_PCA = cumsum(variance_PCS)

ggplot() +
  geom_line(aes(x = 1:length(cumsum_PCA), y = cumsum_PCA)) +
  geom_point(aes(x = 1:length(cumsum_PCA), y = cumsum_PCA)) +
  xlab("# PC") + ylab("Cumulative PVE")
```




```
# 5 PCs needed for 90% variance
```

```
ggbiplot::ggbiplot(pr.out)
```

```
## Warning: package 'plyr' was built under R version 4.0.5
```

```
## ------------------------------------------------------------------------------
```

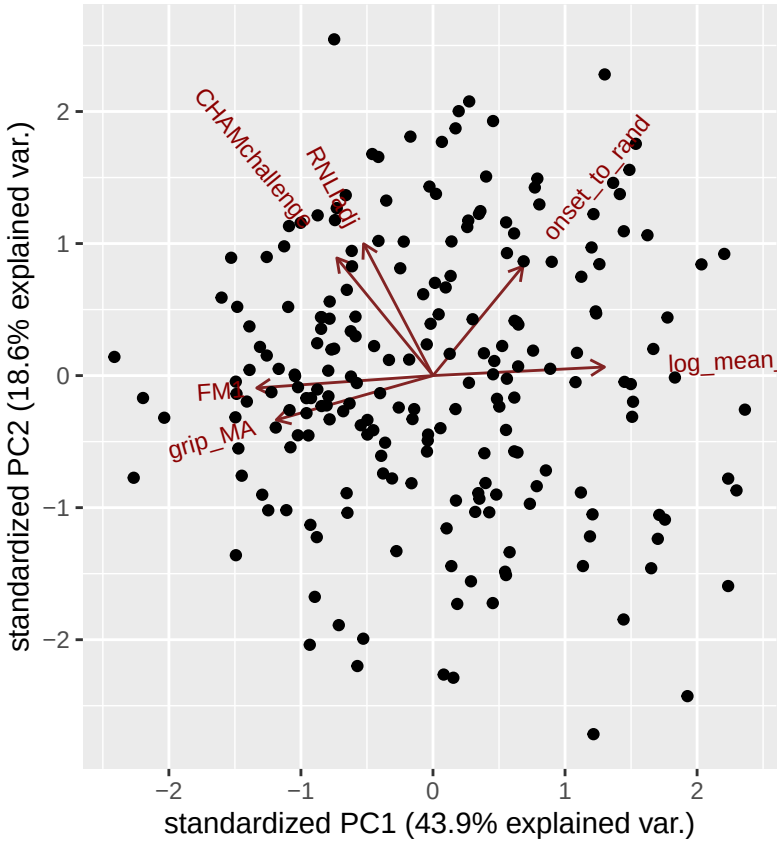
```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## Warning: package 'scales' was built under R version 4.0.5
```




### Kmeans clustering with k =2

```
km.out =kmeans (select(lasso_df, -"FM2"),2, nstart =20)
# Cluster assignment
km.out$cluster
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   2   2   1   1   1   2   1   1   1   1   2   1   1   2   2   2   1   1   1
##  21  23  24  25  26  27  28  29  30  31  32  33  34  35  37  38  39  40  41  42
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   2   1   1   1   1   2
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
##   1   1   2   1   1   1   1   2   1   1   2   2   1   2   2   1   1   2   1   1
##  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   1   2   2   2   1   2   1   1   2   1   1   1   1   1   1   1   2   2   1   1
##  83  84  85  86  87  88  89  90  91  93  94  95  96  97  98  99 100 101 102 103
##   1   1   1   1   2   1   1   1   2   1   1   1   1   1   1   1   1   2   2   1
## 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123
##   2   1   1   1   1   1   1   2   1   2   2   1   2   1   2   2   1   1   2   2
## 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
##   2   1   1   1   2   2   2   2   2   1   1   2   1   1   1   1   1   1   1   2
## 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163
##   1   1   1   1   1   2   2   1   1   2   2   1   1   2   2   1   1   2   1   1
## 164 165 166 167 168 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184
##   1   2   1   2   1   1   1   2   2   1   1   1   2   2   2   1   1   2   1   1
```

```
## 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204
##   1   2   2   1   1   2   1   1   1   1   1   1   1   1   1   2   2   1   2   1
## 205 206 207 208 209 210 211 212 214 215 216 217 218 219
##   2   1   2   2   2   2   2   1   1   1   2   2   2   1
```

### Plotting clusters

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.5
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_cluster(km.out, data = select(lasso_df, -"FM2"),
             # palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw())
```