Untitled

Jessica McPhaul

2024-03-31

```
library(aplore3)
library(ggplot2)
library(plotly)
library(RColorBrewer)
library(pheatmap)
library(cluster)
library(ggcorrplot)
library(dplyr)
library(tidyr)
library(sjPlot)
library(sjmisc)
```

Data Format: A data.frame with 500 rows and 18 variables such as:

priorfrac - If the patient previously had a fracture
age
weight
height
bmi
premeno
momfrac
armassist
smoke
raterisk
fracscore
fracture
bonemed - Bone medications at enrollment (1: No, 2: Yes)
bonemed_fu - Bone medications at follow-up (1: No, 2: Yes)
bonetreat - Bone medications both at enrollment and follow-up (1: No, 2: Yes)

```
head(glow_bonemed)
```

```
##   sub_id site_id phy_id priorfrac age weight height      bmi premeno
momfrac
## 1      1       1     14        No  62   70.3    158 28.16055      No
No
## 2      2       4    284        No  65   87.1    160 34.02344      No
No
```

```
## 3       3        6    305        Yes  88    50.8    157 20.60936       No
Yes
## 4       4        6    309         No  82    62.1    160 24.25781       No
No
## 5       5        1     37         No  61    68.0    152 29.43213       No
No
## 6       6        5    299        Yes  67    68.0    161 26.23356       No
No
##   armassist smoke raterisk fracscore fracture bonemed bonemed_fu bonetreat
## 1        No    No     Same         1       No      No         No        No
## 2        No    No     Same         2       No      No         No        No
## 3       Yes    No     Less        11       No      No         No        No
## 4        No    No     Less         5       No      No         No        No
## 5        No    No     Same         1       No      No         No        No
## 6        No   Yes     Same         4       No      No         No        No
```

Summary statistics for numeric variables

```
mysummary = glow_bonemed %>%
  select(age, weight, height, bmi, fracscore) %>%
  summarise_each(
    funs(min = min,
    q25 = quantile(., 0.25),
    median = median,
    q75 = quantile(., 0.75),
    max = max,
    mean = mean,
    sd = sd,
    variance= var))
# reshape it using tidyr functions
clean.summary = mysummary %>%
  gather(stat, val) %>%
  separate(stat, into = c("var", "stat"), sep = "_") %>%
  spread(stat, val) %>%
  select(var, min, max, mean, sd, variance)
print(clean.summary)
##         var       min       max      mean        sd   variance
```

```
## 1       age  55.00000  90.00000  68.56200  8.989537  80.811780
## 2       bmi  14.87637  49.08241  27.55303  5.973958  35.688178
## 3 fracscore   0.00000  11.00000   3.69800  2.495446   6.227251
## 4    height 134.00000 199.00000 161.36400  6.355493  40.392289
## 5    weight  39.90000 127.00000  71.82320 16.435992 270.141825
```

Summary statistics for categorical variables

```
summary(glow_bonemed %>% select(priorfrac, premeno, momfrac, armassist,
smoke, raterisk, fracture, bonemed, bonemed_fu, bonetreat))
```

```
##  priorfrac premeno   momfrac   armassist smoke        raterisk   fracture
##  No :374   No :403   No :435   No :312   No :465   Less   :167   No :375
##  Yes:126   Yes: 97   Yes: 65   Yes:188   Yes: 35   Same   :186   Yes:125
##                                                   Greater:147
##  bonemed   bonemed_fu bonetreat
##  No :371   No :361    No :382
##  Yes:129   Yes:139    Yes:118
##
```

No missing values

```
colSums(is.na(glow_bonemed))
```

```
##     sub_id    site_id     phy_id  priorfrac        age     weight
height
##          0          0          0          0          0          0
0
##        bmi    premeno    momfrac  armassist      smoke   raterisk
fracscore
##          0          0          0          0          0          0
0
##   fracture    bonemed bonemed_fu  bonetreat
##          0          0          0          0
```

```
sum(is.na(glow_bonemed))
```

```
## [1] 0
```

```
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.3.3
##
## Attaching package: 'kableExtra'
## The following object is masked from 'package:dplyr':
```

```
##
##     group_rows
# different way to present no missing values
# kable Extra library to make document more presentable
colSums(is.na(glow_bonemed)) %>%
  kable("html", caption = "No missing values") %>%
  kable_styling()
```

No missing values

sub_id
site_id
phy_id
priorfrac
age
weight
height
bmi
premeno
momfrac

No missing values

armassist
smoke
raterisk
fracscore
fracture
bonemed
bonemed_fu
bonetreat

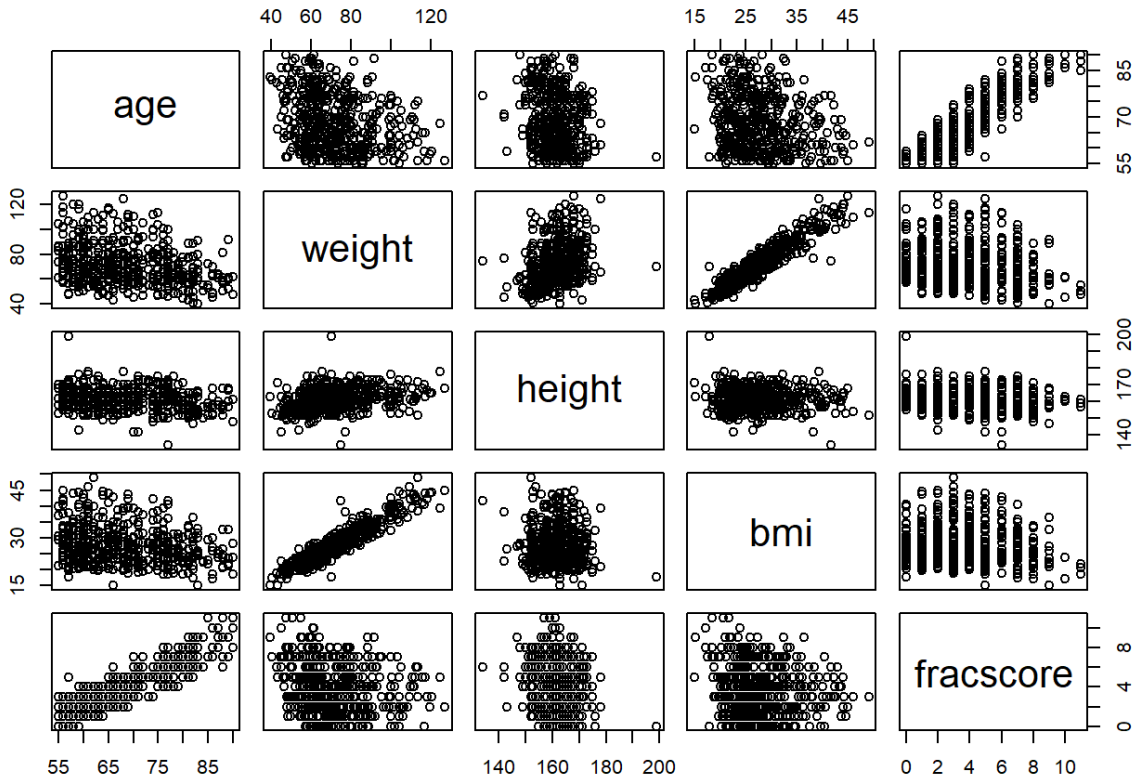
Age vs Weight: As weight increases the average age decreases
Age vs Height: Weak correlation of as height increases age decreases
Age vs BMI: As bmi increases the average age decreases
Age vs fracscore: As age increases the average fracscore increases

Weight vs Height: As height increases the average weight increases
Weight vs BMI: As bmi increases the average weight increases
Weight vs fracscore: As fracscore increases the average Weight decreases

Height vs BMI: As bmi increases the average height and variance stay the same
Height vs fracscore: As fracscore increases the average height stays the same though variance might decrease

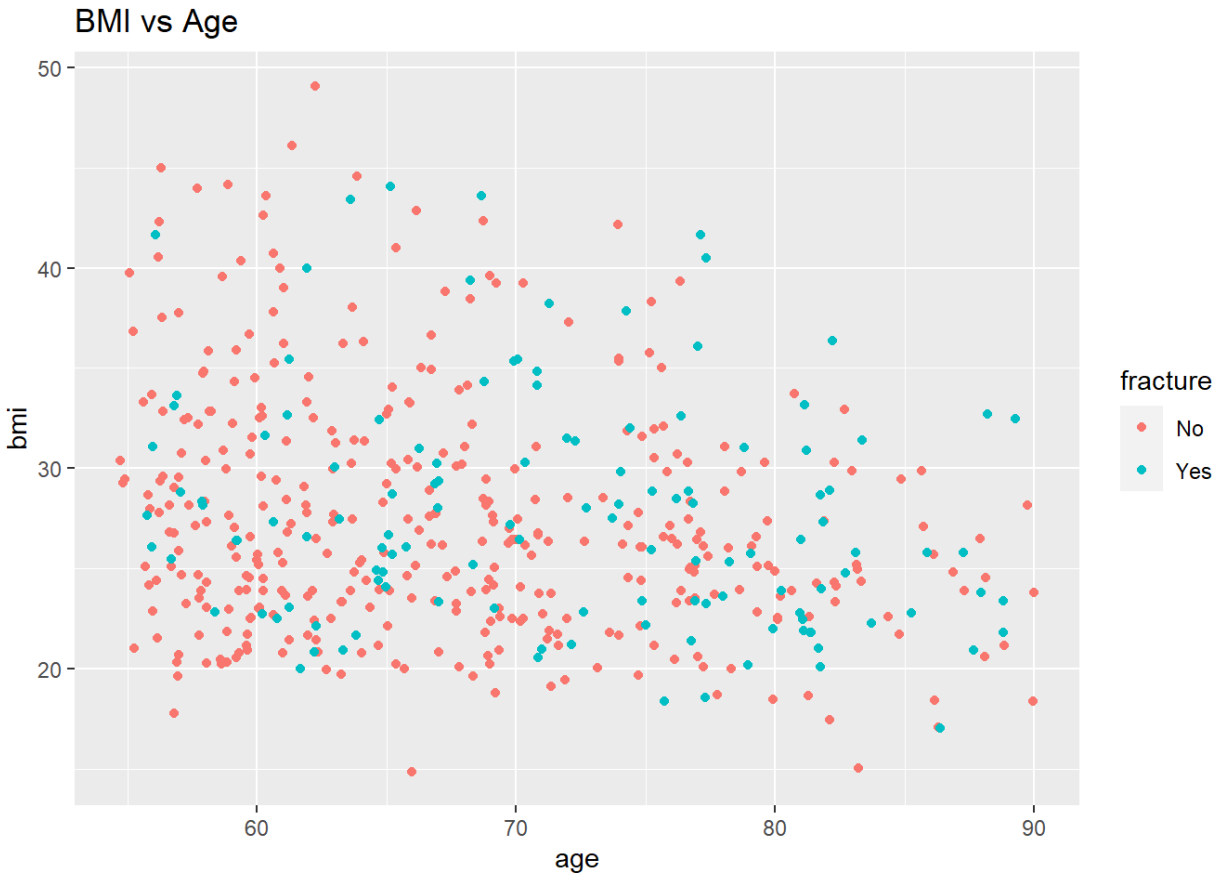
BMI vs fracscore: As fracscore increases the average bmi decreases

```
plot(glow_bonemed[, c(5:8, 14)])
```

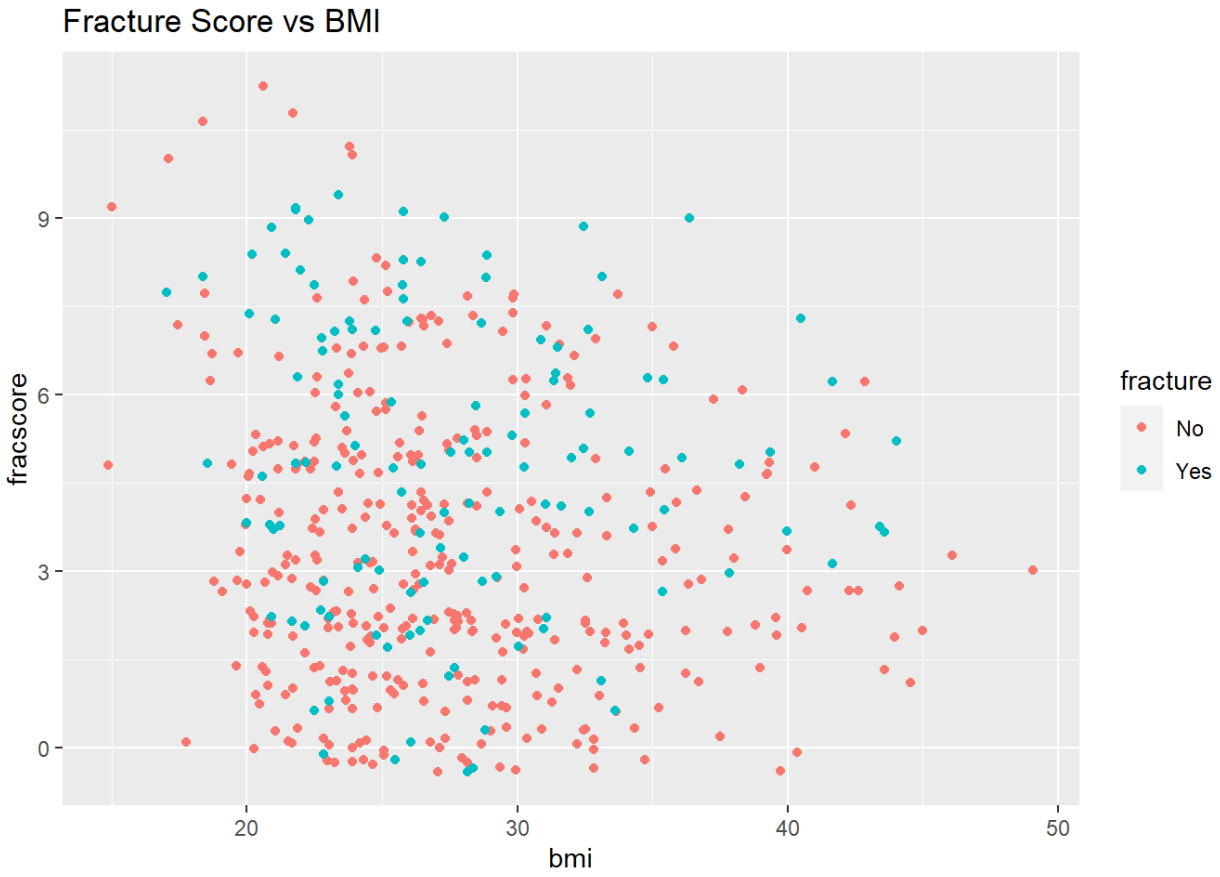


Non of the following scatter plots show strong groupings for the fracture/no fracture categorical variable

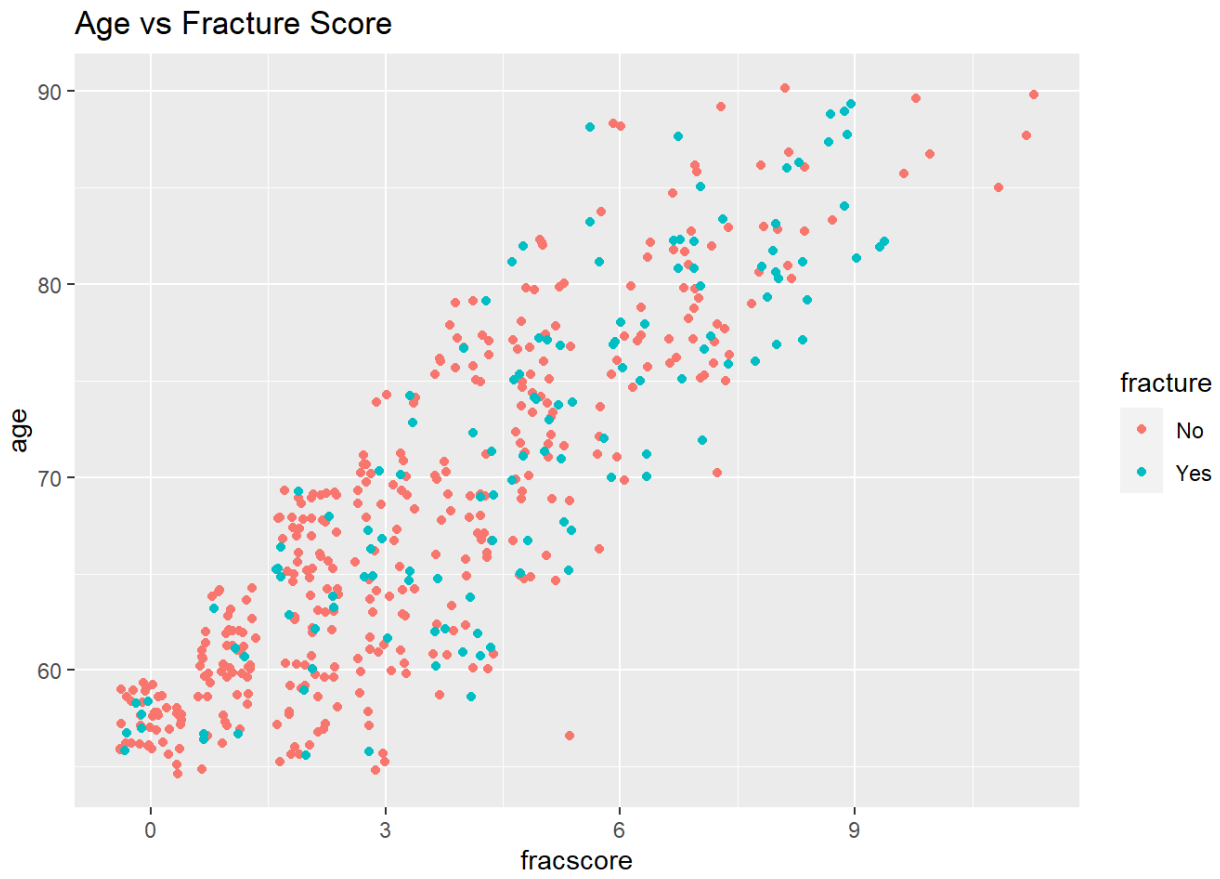
```
ggplot(glow_bonemed, aes(x = age, y = bmi, color = fracture)) +
  geom_jitter() +
  labs(title = "BMI vs Age")
```



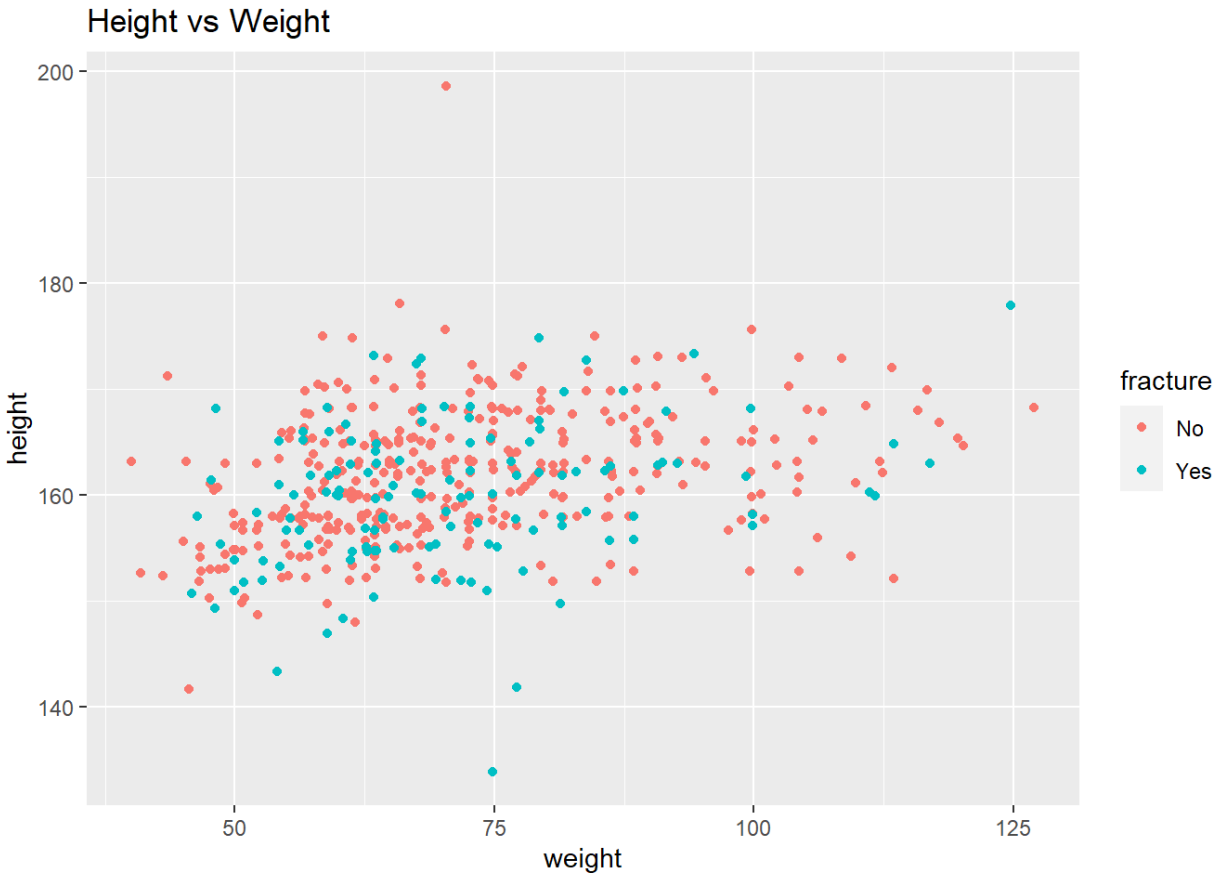

```
ggplot(glow_bonemed, aes(x = bmi, y = fracscore, color = fracture)) +
  geom_jitter() +
  labs(title = "Fracture Score vs BMI")
```




```
ggplot(glow_bonemed, aes(x = fracscore, y = age, color = fracture)) +
  geom_jitter() +
  labs(title = "Age vs Fracture Score")
```




```
ggplot(glow_bonemed, aes(x = weight, y = height, color = fracture)) +
  geom_jitter() +
  labs(title = "Height vs Weight")
```

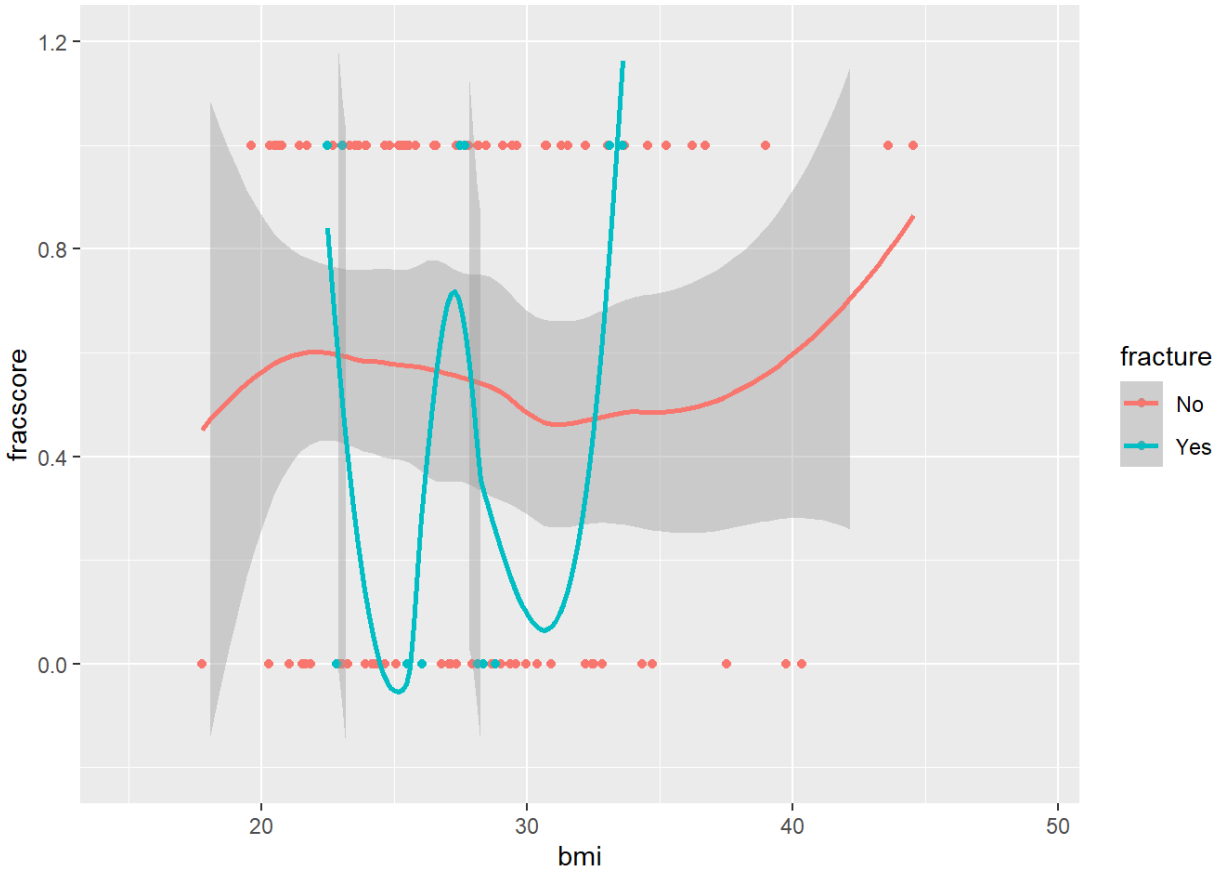


Once again there doesn't seem to be strong groupings of the fracture categorical variable

```
fracture3dplot = plot_ly(glow_bonemed,
  x = ~age,
  y = ~height,
  z = ~bmi,
  color = ~fracture,
  colors = c('#0C4B8E', '#BF382A')) %>% add_markers()
fracture3dplot
```

NoYes

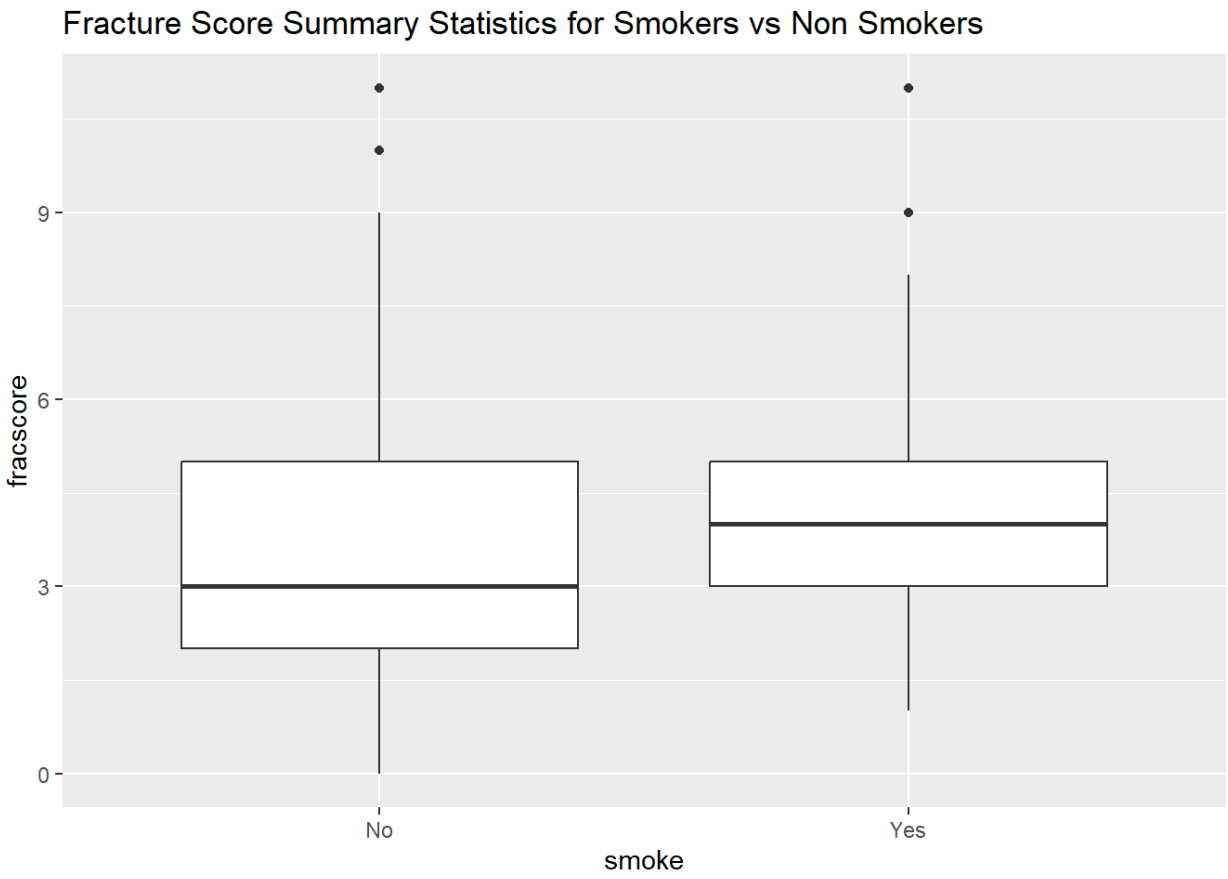
There are so little "yes" fracture results that the plot isn't very useful




```
## `geom_smooth()` using formula = 'y ~ x'
```

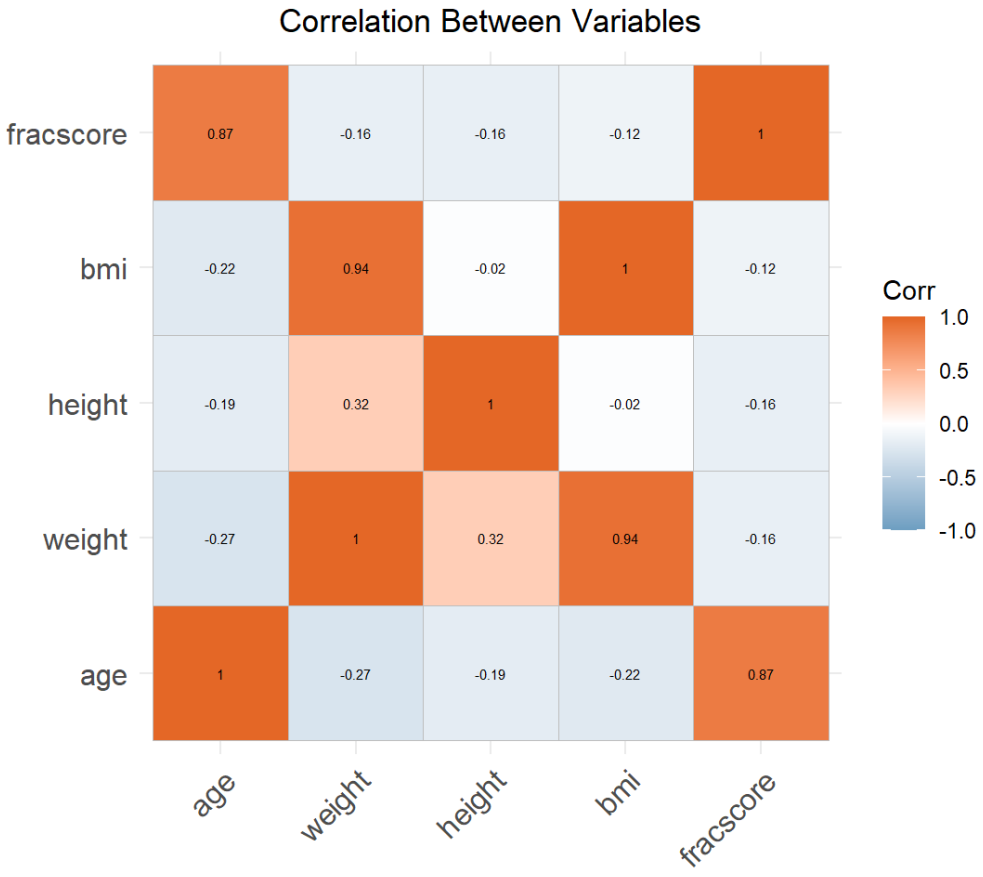
The boxplot shows that the mean fracscore seems to be slightly higher for smokers compared to non smokers

```
ggplot(glow_bonemed, aes(x = smoke, y = fracscore)) +
  geom_boxplot() +
  labs(title = "Fracture Score Summary Statistics for Smokers vs Non Smokers")
```



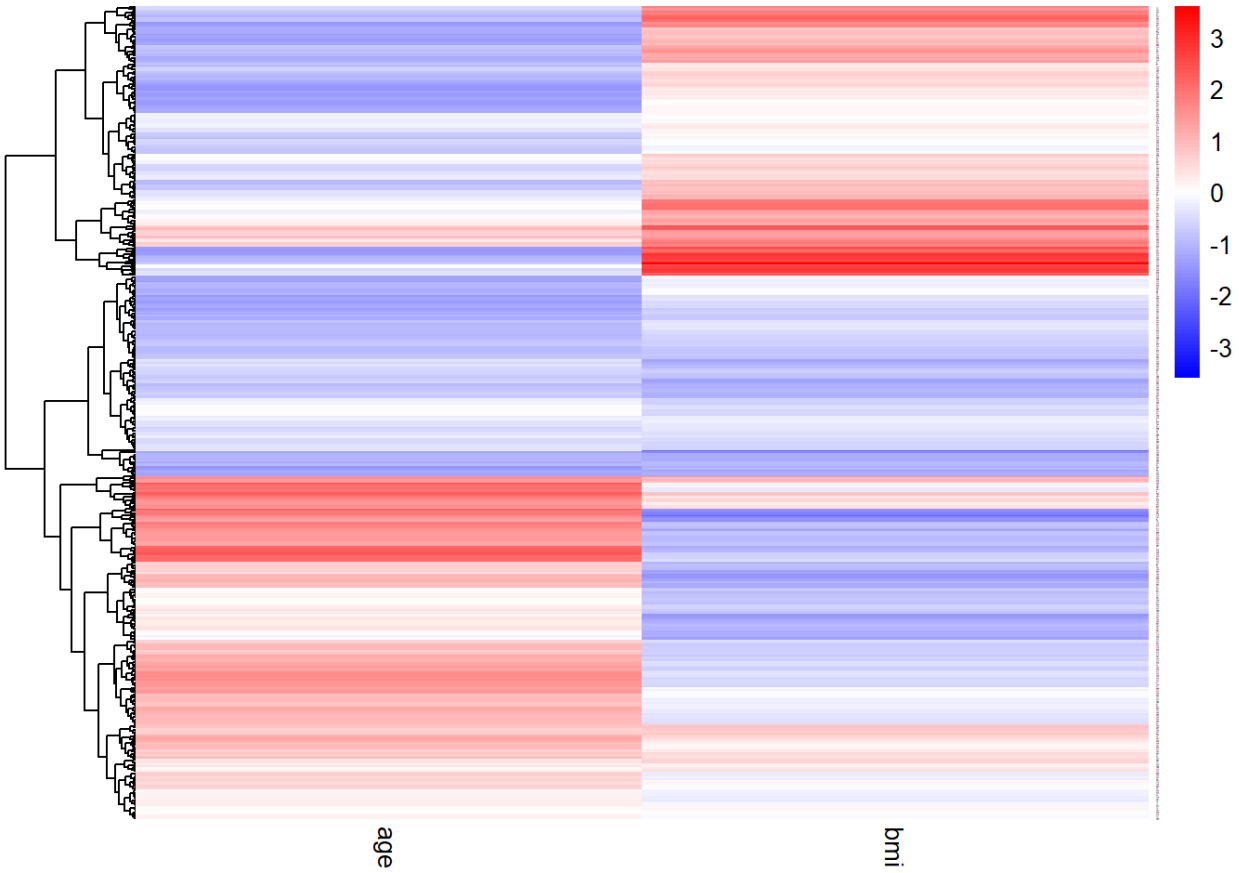
Plot confirms there is a strong correlation between age/fracscore, bmi/weight

```
corr_vars <- c("age", "weight", "height", "bmi", "fracscore")
corr_df <- glow_bonemed[, corr_vars]
corr_df <- cor(corr_df)
ggcorrplot(corr = corr_df, lab = TRUE, lab_size = 2,
  colors = c("#6D9EC1", "white", "#E46726")) +
  labs(title = "Correlation Between Variables") +
  theme(plot.title = element_text(hjust = .5),
  plot.subtitle = element_text(hjust = .5))
```

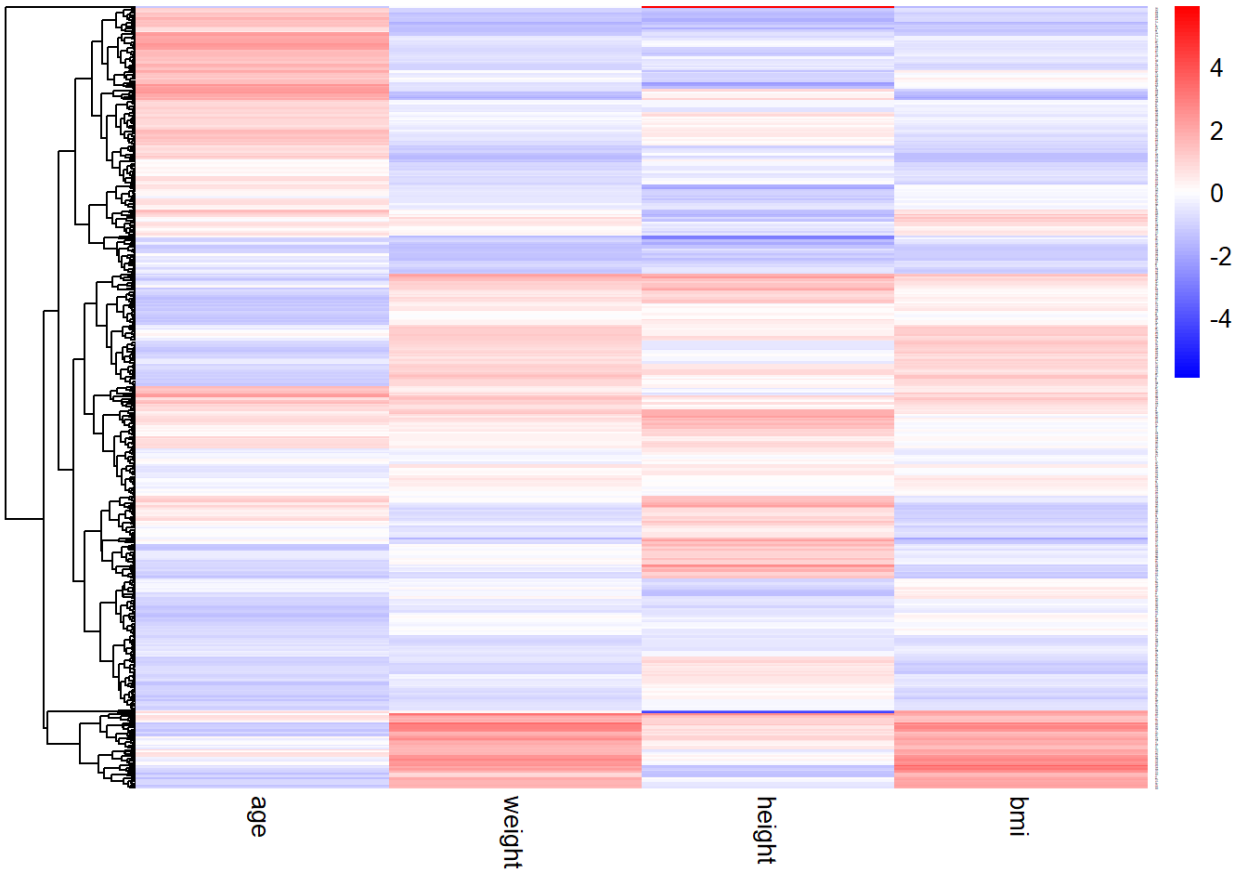


Clustering EDA

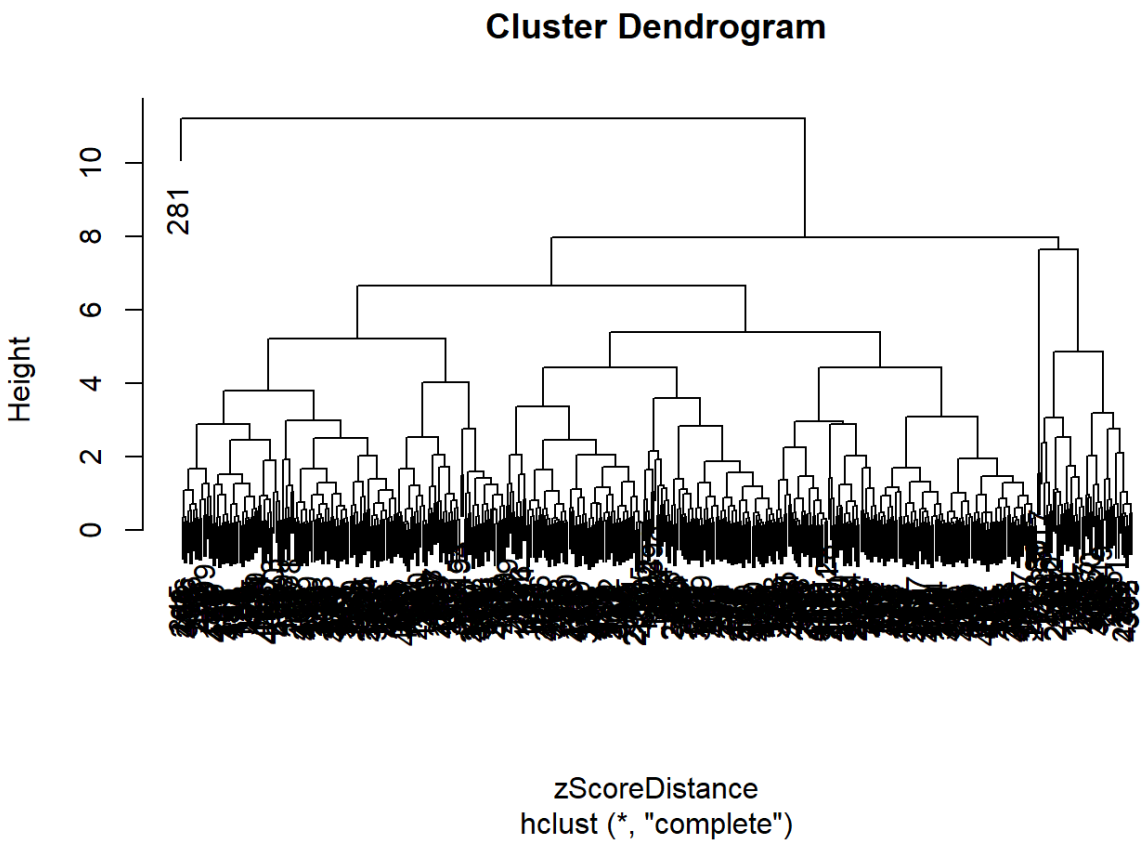
```
pheatmap(glow_bonemed[, c(5,8)], scale = "column", fontsize_row = 0.1,
cluster_cols = F, legend = T, color = colorRampPalette(c("blue", "white",
"red"), space = "rgb")(100))
```




```
pheatmap(glow_bonemed[, 5:8], scale = "column", fontsize_row = 0.1,
cluster_cols = F, legend = T, color = colorRampPalette(c("blue", "white",
"red"), space = "rgb")(100))
```




```
zScoreScale = scale(glow_bonemed[, 5:8])
zScoreDistance = dist(zScoreScale)
continuousVariableClustering = hclust(zScoreDistance, method = "complete")
plot(continuousVariableClustering)
```



Modeling

Split the data into a training/testing set

```
set.seed(4)
trainingIndices = sample(c(1:dim(glow_bonemed)[1]), dim(glow_bonemed)[1]*0.8)
trainingDataframe = glow_bonemed[trainingIndices,]
testingDataframe = glow_bonemed[-trainingIndices,]
```

Age is the only statistically significant continuous variable at the alpha = 0.2 level (p < 0.0001)

```
model = glm(fracture ~ age + weight + height + bmi, data = glow_bonemed,
family = "binomial")
summary(model)
```

```
##
## Call:
## glm(formula = fracture ~ age + weight + height + bmi, family = "binomial",
##     data = glow_bonemed)
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -13.29208   12.54412  -1.060    0.289
## age           0.05263    0.01237   4.255 2.09e-05 ***
## weight       -0.09720    0.08559  -1.136    0.256
## height        0.04914    0.07747   0.634    0.526
## bmi           0.27450    0.22072   1.244    0.214
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 562.34  on 499  degrees of freedom
## Residual deviance: 532.92  on 495  degrees of freedom
## AIC: 542.92
##
## Number of Fisher Scoring iterations: 4
```

```
AIC(model)
```

```
## [1] 542.9224
```

```
library(ResourceSelection)
```

```
## Warning: package 'ResourceSelection' was built under R version 4.3.3
```

```
## ResourceSelection 0.3-6   2023-06-27
```

```
hoslem.test(model$y,fitted(model)) # shows non-significant test result which
means this is a decent model fit
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  model$y, fitted(model)
## X-squared = 11.39, df = 8, p-value = 0.1806
```

```
# get odds ratio for model
exp((model$coefficients))
```

```
##  (Intercept)          age       weight       height          bmi
## 1.687806e-06 1.054042e+00 9.073722e-01 1.050367e+00 1.315867e+00
```

```
# get confidence intervals
```

```
exp(confint(model))
```

```
## Waiting for profiling to be done...
##                    2.5 %       97.5 %
## (Intercept) 3.277769e-17 76023.505396
## age         1.029041e+00     1.080263
## weight      7.644329e-01     1.070088
## height      9.024284e-01     1.222733
## bmi         8.597478e-01     2.047437
```

```
# trying to figure out how to use sjPlot to mimic what we did in unit12 prelive
#plot_model(model, type = "pred", terms = c("age", "smoke"))
corr_vars <- c("age", "weight", "height", "bmi", "fracscore")
pc.result<-prcomp(glow_bonemed[, corr_vars],scale.=TRUE)
#Eigen Vectors
pc.result$rotation
```

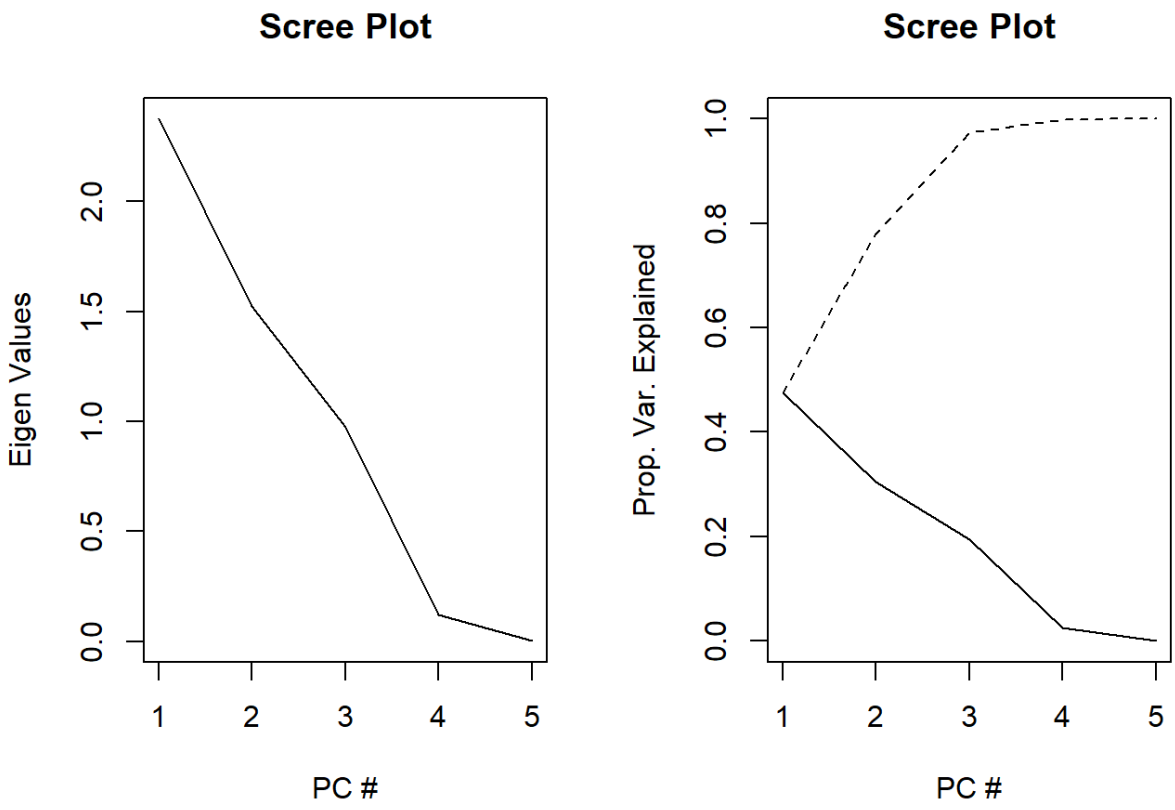
```
##                   PC1         PC2         PC3         PC4          PC5
## age         0.4947219  0.46742140 -0.15246583  0.71654567 -0.009160237
## weight     -0.5273035  0.46578775 -0.08840991  0.03240244 -0.704362523
## height     -0.2345770 -0.08196149 -0.93823245  0.01885633  0.240042129
## bmi        -0.4741030  0.51615173  0.24533677  0.05137380  0.667820563
## fracscore   0.4442985  0.53984137 -0.16872342 -0.69463484  0.013601399
```

```
#Eigen Values
eigenvals<-pc.result$sdev^2
eigenvals
```

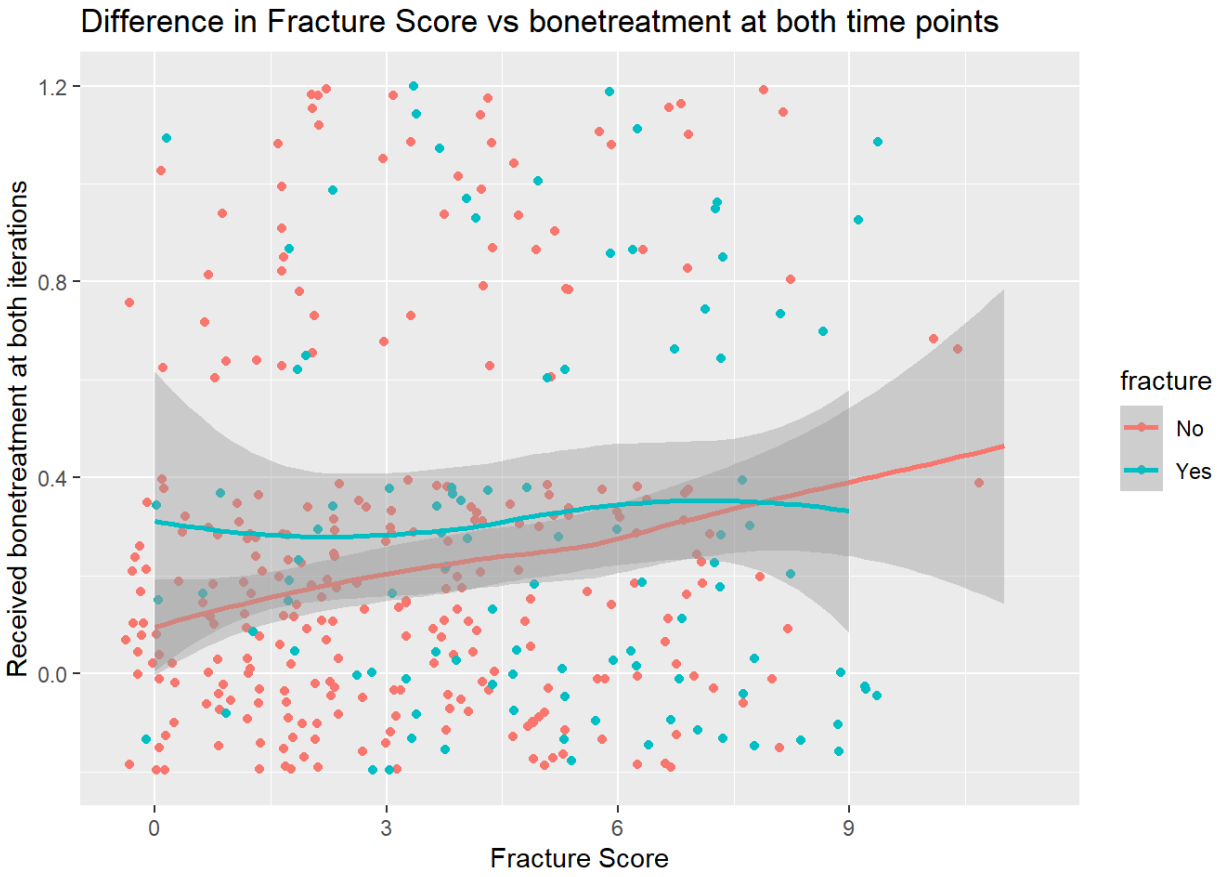
```
## [1] 2.374116591 1.523507236 0.975710317 0.123469514 0.003196342
```

```
#Scree plot
par(mfrow = c(1,2))
plot(eigenvals,type = "l",
     main = "Scree Plot",
     ylab = "Eigen Values",
     xlab = "PC #")
plot(eigenvals / sum(eigenvals),
     type = "l", main = "Scree Plot",
     ylab = "Prop. Var. Explained",
     xlab = "PC #",
```

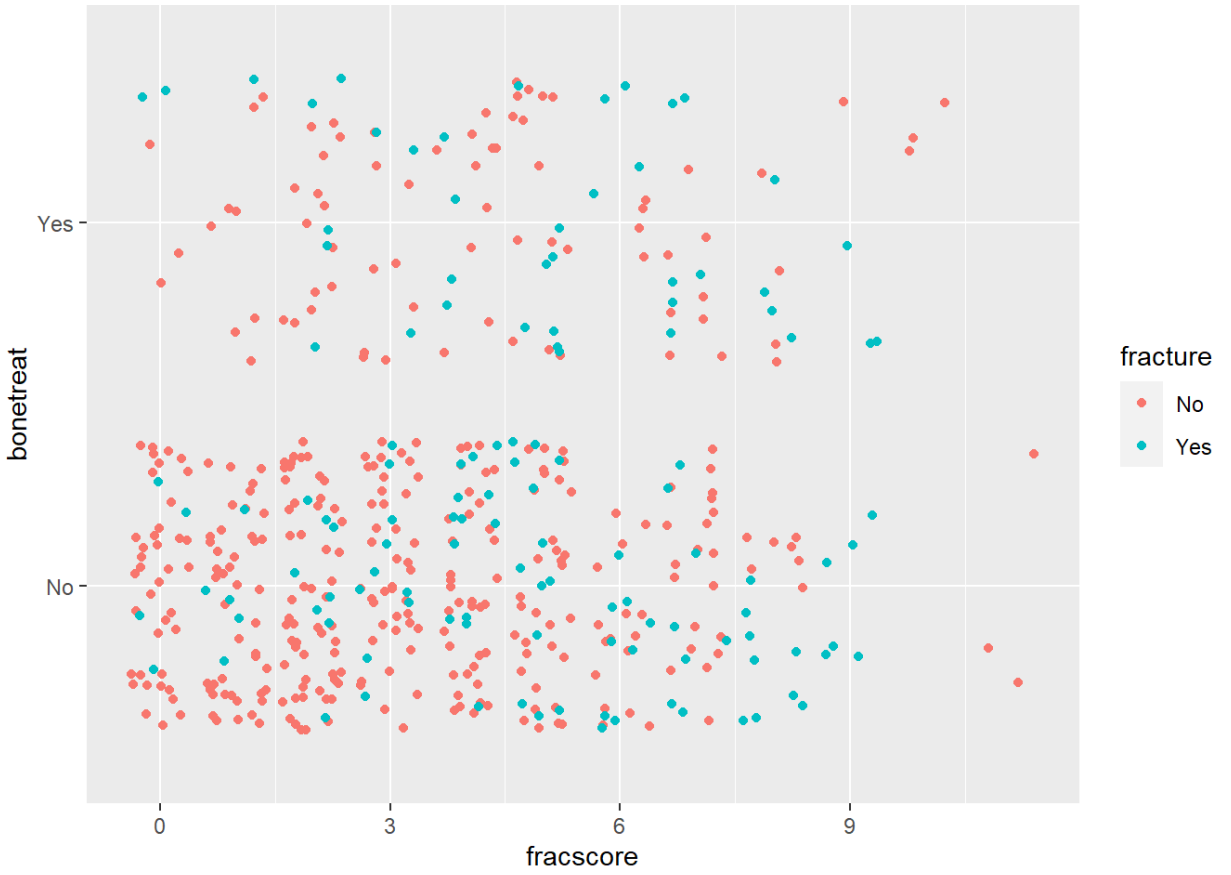
```
     ylim = c(0, 1))
cumulative.prop = cumsum(eigenvals / sum(eigenvals))
lines(cumulative.prop, lty = 2)
```




```
# Loess curve for fracscore by bonetreatment group showing fracture or not
glow_bonemed$bonetreat.num <- ifelse(glow_bonemed$bonetreat == "No", 0, 1)
ggplot(glow_bonemed, aes(x = fracscore, y = bonetreat.num, color = fracture))
+
  geom_jitter()+
  geom_smooth(method="loess",size=1,span=1)+
  ylim(-.2,1.2) +
  xlab("Fracture Score") +
  ylab("Received bonetreatment at both iterations") +
  ggtitle("Difference in Fracture Score vs bonetreatment at both time
points")
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 124 rows containing missing values (`geom_point()`).
```




```
# shows that there is not an increased risk, i.e. no changes, in likelihood
of fracture, whereas only receiving one or no treatments trends to increase
the likelihood of a fracture as the fracture score goes up


# plot breaking down to see if there is any separation
ggplot(glow_bonemed, aes(x = fracscore, y = bonetreat, color = fracture)) +
  geom_jitter()
```




```
# in bonetreat, i.e. bone meds at both time points, in the no group, there appear to be higher fracture rates with increased fracscore, which would be predicted, i.e. if you received treatment at both times there doesn't appear to be a correlation in fracscore and breaking a bone (fracture), vs the group that did not receive both treatments appears to be a correlation with a higher likelihood correlating to likelihood of fracture

# Loess curve for fracscore by physician group showing fracture or not

table(glow_bonemed$priorfrac) # show table of prior fractures
```

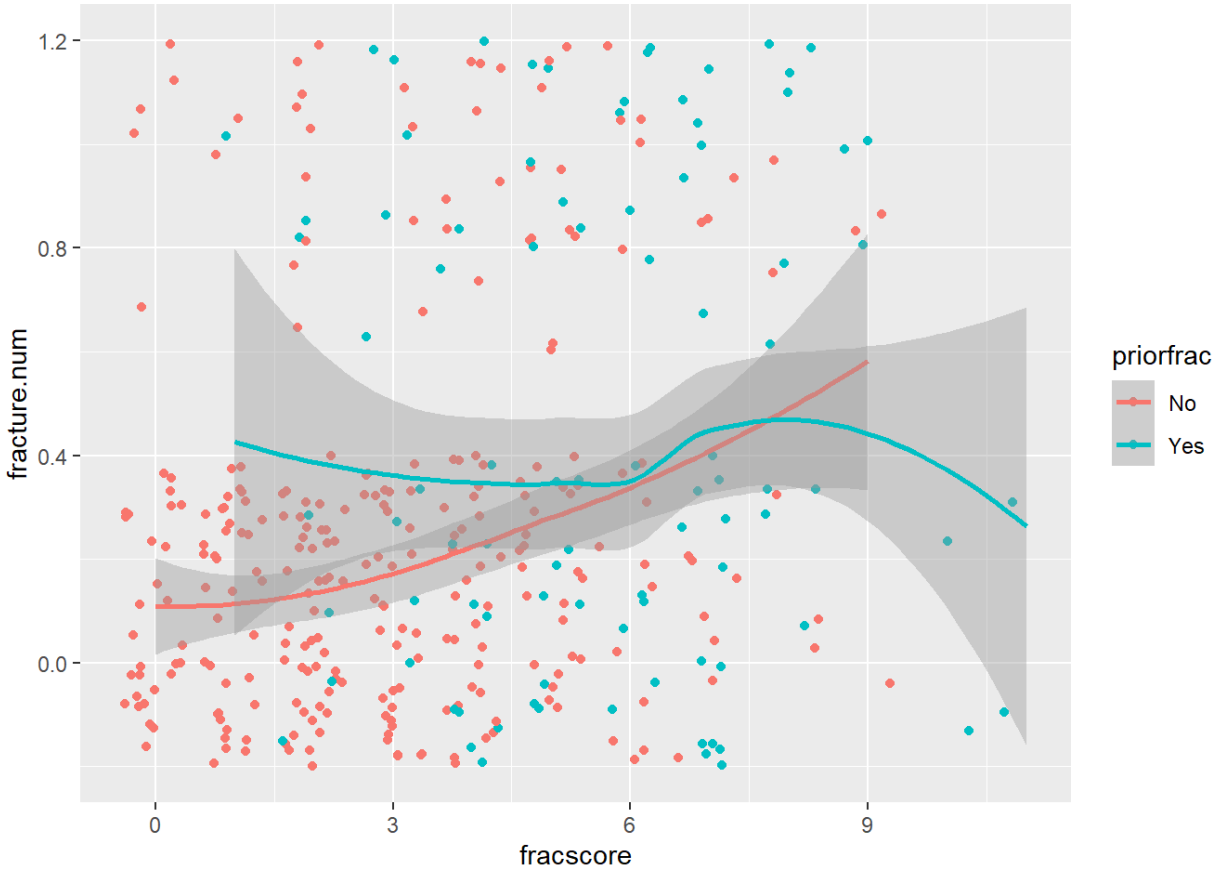
```
##
##  No Yes
## 374 126
```

```
glow_bonemed$priorfrac.num <- ifelse(glow_bonemed$priorfrac == "No", 0, 1) # create numeric variable


glow_bonemed$fracture.num <- ifelse(glow_bonemed$fracture == "No", 0, 1) # create numeric variable


levels(glow_bonemed$fracture)
```

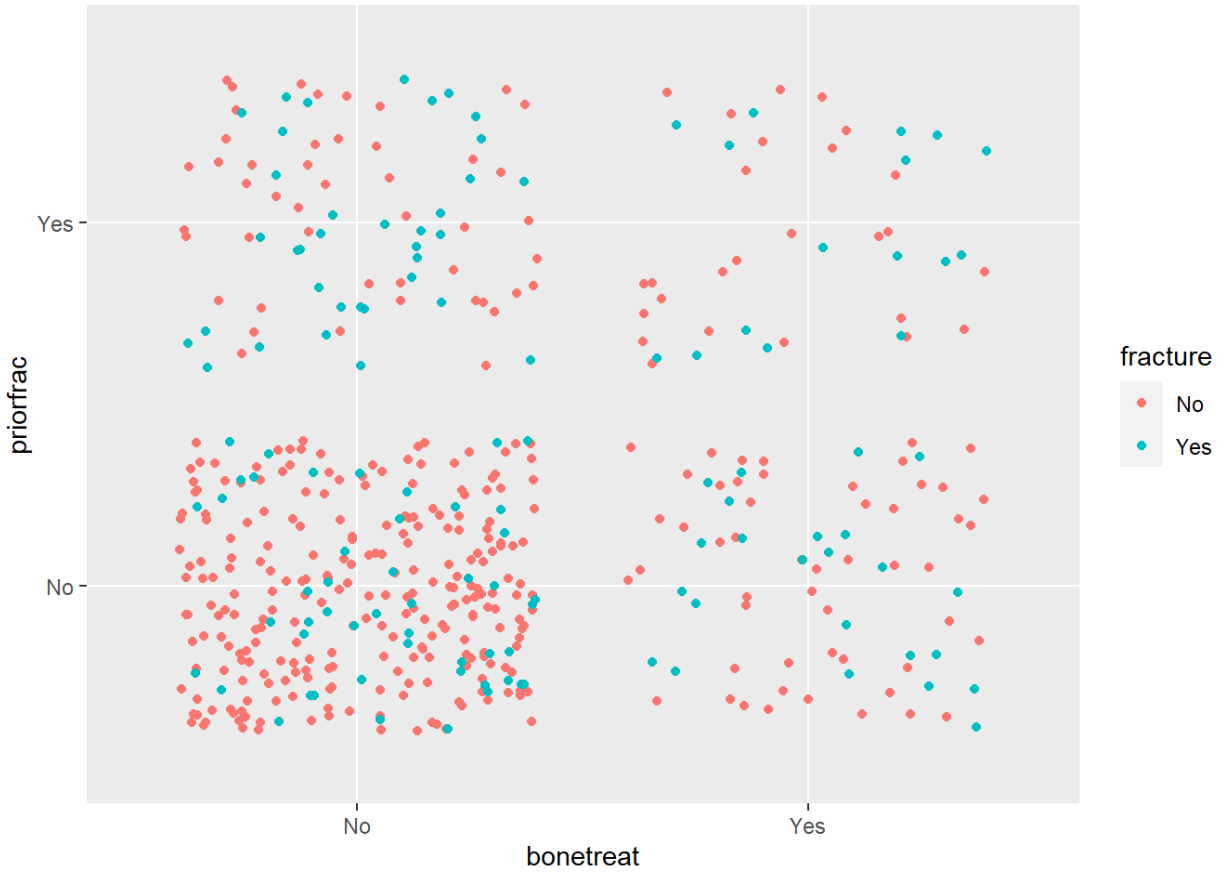
```
## [1] "No"  "Yes"
```

```
ggplot(glow_bonemed, aes(x = fracscore, y = fracture.num, color = priorfrac))
+
  geom_jitter()+
  geom_smooth(method="loess",size=1,span=1)+
  ylim(-.2,1.2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 129 rows containing missing values (`geom_point()`).
```




```
# shows that there is not an increased risk associated with higher fracscore,
i.e. no changes, in likelihood of an increased fracture if you previous had a
fracture  whereas the group that has never had a fracture tends to increase
the likelihood of a fracture as the fracture score goes up


# plot breaking down to see if there is any separation
ggplot(glow_bonemed, aes(x = bonetreat, y = priorfrac, color = fracture)) +
  geom_jitter()
```




```
library(caret)
```

```
## Loading required package: lattice
```

```
# plot

ggplot(glow_bonemed, aes(x = age, y = ifelse(glow_bonemed$smoke == "No", 0,
1), color = fracture)) +

  geom_jitter()+

  geom_smooth(method="loess",size=1,span=1)+

  ylim(-.2,1.2)
```

```
## Warning: Use of `glow_bonemed$smoke` is discouraged.

## ℹ Use `smoke` instead.

## Use of `glow_bonemed$smoke` is discouraged.

## ℹ Use `smoke` instead.

## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 130 rows containing missing values (`geom_point()`).
```


```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
set.seed(4)


#note CV and error metric are not really used here, but logLoss is reported
for the final model.

# set tuning parameters using logloss

fitControl<-
trainControl(method="repeatedcv",number=10,repeats=1,classProbs=TRUE,
summaryFunction=mnLogLoss)


# build glmnet model
```

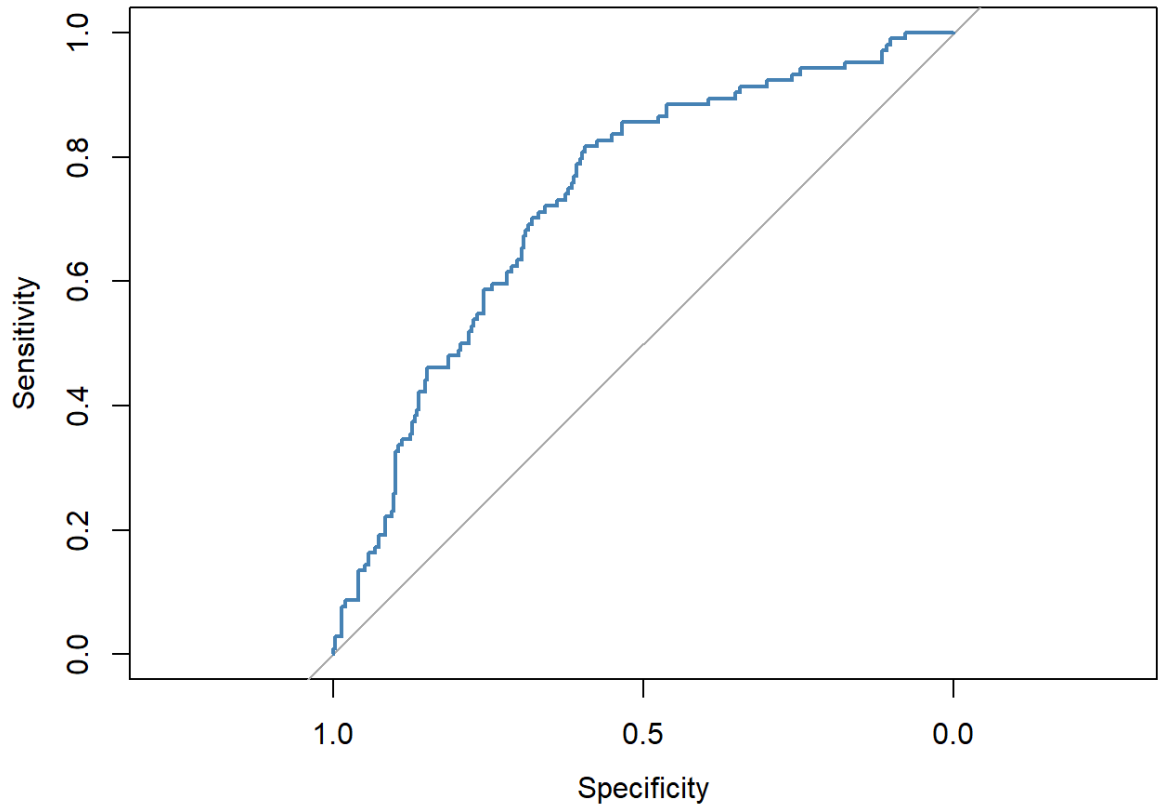
```
glmnet.fit<-train(fracture ~ . - sub_id,
                  data=trainingDataframe,
                  method="glmnet",
                  trControl=fitControl,
                  metric="logLoss")
coef(glmnet.fit$finalModel,glmnet.fit$finalModel$lambdaOpt)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)     -0.340276566
## site_id          0.039198564
## phy_id           .
## priorfracYes     0.350415528
## age              .
## weight           .
## height          -0.012466332
## bmi              0.004981546
## premenoYes       .
## momfracYes       0.370402577
## armassistYes     0.073010553
## smokeYes         .
## rateriskSame     0.156439065
## rateriskGreater  0.489383198
## fracscore        0.128442276
## bonemedYes       .
## bonemed_fuYes    0.394532573
## bonetreatYes     .
```

```
#Getting predictions for glmnet for Complex model
glmnetfit.predprobs<-predict(glmnet.fit, trainingDataframe ,type="prob")



# glmnet ROC
glmnet.roc<-roc(response= trainingDataframe$fracture,
predictor=glmnetfit.predprobs$No,levels=c("No","Yes"))
```

```
## Setting direction: controls > cases
```

```
plot(glmnet.roc,col="steelblue")
```




```
# Save for later
# plot(glmnet.roc,add=T,col="steelblue")
# legend("bottomright",
#       legend=c("Simple", "Complex","GLMNET"),
#       col=c("black", "red","steelblue"),
#       lwd=4, cex =1, xpd = TRUE, horiz = FALSE)
```

Left out the following variables: bonetreat, bonemed, smoke, premeno, weight, age, phy_id.

```
# Build complex model with interactions and/or polynomials

complex1 = glm(fracture ~  bmi + bonetreat + fracscore + priorfrac + bonemed
+ bonemed_fu + priorfrac:fracscore + bmi:fracscore + fracscore:bonetreat,
data = trainingDataframe, family = "binomial")

summary(complex1)
```

```
##
```

```
## Call:
## glm(formula = fracture ~ bmi + bonetreat + fracscore + priorfrac +
##     bonemed + bonemed_fu + priorfrac:fracscore + bmi:fracscore +
##     fracscore:bonetreat, family = "binomial", data = trainingDataframe)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.755675   1.360927  -1.290  0.19703
## bmi                     -0.028581   0.047829  -0.598  0.55013
## bonetreatYes            -1.686170   1.066664  -1.581  0.11393
## fracscore               -0.009717   0.273117  -0.036  0.97162
## priorfracYes             1.380602   0.704983   1.958  0.05019 .
## bonemedYes               1.266284   0.718496   1.762  0.07800 .
## bonemed_fuYes            1.534272   0.533329   2.877  0.00402 **
## fracscore:priorfracYes -0.175793   0.121858  -1.443  0.14913
## bmi:fracscore            0.010442   0.009815   1.064  0.28737
## bonetreatYes:fracscore -0.109089   0.110192  -0.990  0.32218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 458.45  on 399  degrees of freedom
## Residual deviance: 409.18  on 390  degrees of freedom
## AIC: 429.18
##
## Number of Fisher Scoring iterations: 4
```

```
AIC(complex1)
```

```
## [1] 429.1816
```

```
library(ResourceSelection)
hoslem.test(complex1$y,fitted(complex1)) # shows non-significant test result
which means this is a decent model fit
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
```

```
## data:  complex1$y, fitted(complex1)
## X-squared = 5.3987, df = 8, p-value = 0.7142
```

```
# get odds ratio for model
exp((complex1$coefficients))
```

```
##            (Intercept)                    bmi           bonetreatYes
##              0.1727906              0.9718236              0.1852275
##              fracscore          priorfracYes             bonemedYes
##              0.9903299              3.9772935              3.5476441
##          bonemed_fuYes fracscore:priorfracYes          bmi:fracscore
##              4.6379481              0.8387915              1.0104969
## bonetreatYes:fracscore
##              0.8966504
```

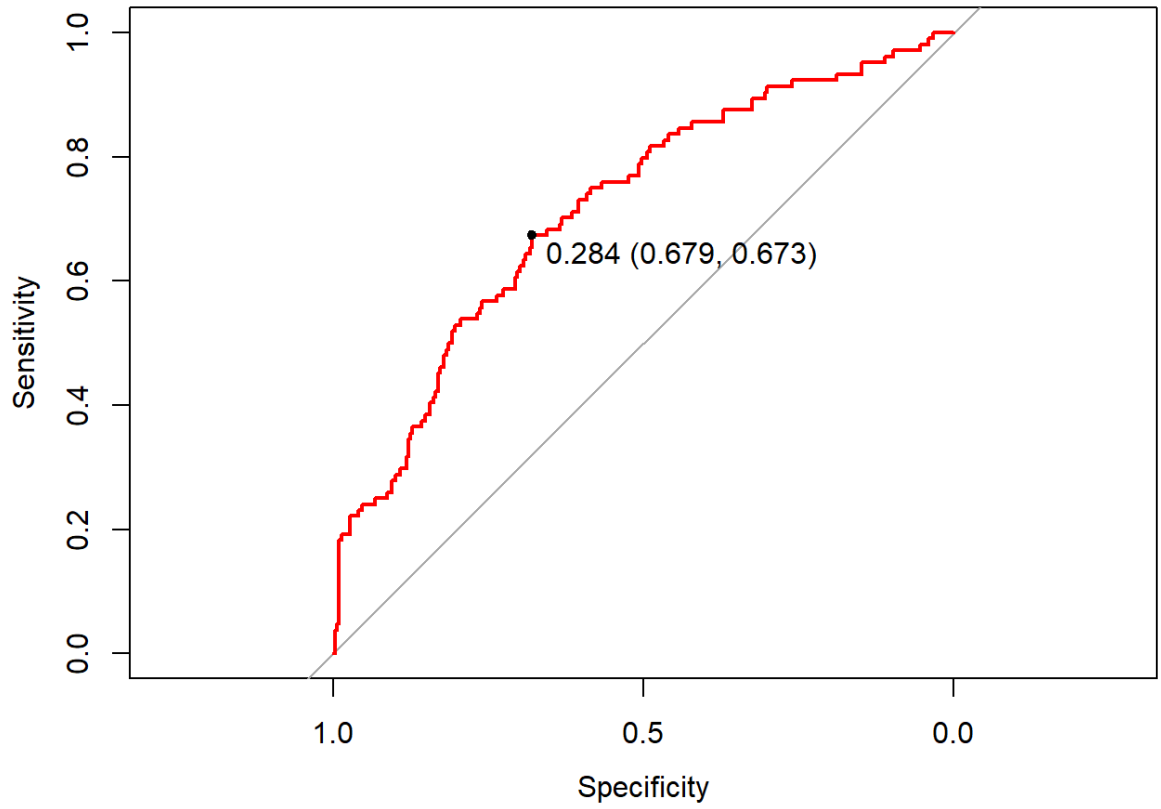
```
# get confidence intervals
exp(confint(complex1))
```

```
## Waiting for profiling to be done...
##                               2.5 %    97.5 %
## (Intercept)            0.01199624  2.539720
## bmi                    0.88195989  1.064655
## bonetreatYes           0.02205353  1.491962
## fracscore              0.57419435  1.685052
## priorfracYes           0.98048590 15.791502
## bonemedYes             0.85958345 15.579380
## bonemed_fuYes          1.65454068 13.772257
## fracscore:priorfracYes 0.65990118  1.066216
## bmi:fracscore          0.99162476  1.030752
## bonetreatYes:fracscore 0.72373545  1.117224
```

```
# Get Predictions
#Complex model from previous
complex1.predprobs<-predict(complex1,trainingDataframe ,type="response")


# complex model ROC
complex1.roc<-
roc(response=trainingDataframe$fracture,predictor=complex1.predprobs,levels=c
("No","Yes"))
```

```
## Setting direction: controls < cases
```

```
# plot ROC

plot(complex1.roc,print.thres="best",col="red")
```




```
# Now check validation in test set

set.seed(4)

validateComplexPred <- predict(complex1, newdata = testingDataframe,
type="response")



# check confusion matrix positive class is no fracture

threshold = .284

validateComplexPredictions<-
factor(ifelse(validateComplexPred>threshold,"No","Yes"))


#Confusion matrix for objective 2 complex model 1 with interactions

confusionMatrix(data = validateComplexPredictions, reference =
testingDataframe$fracture, positive="Yes")
```

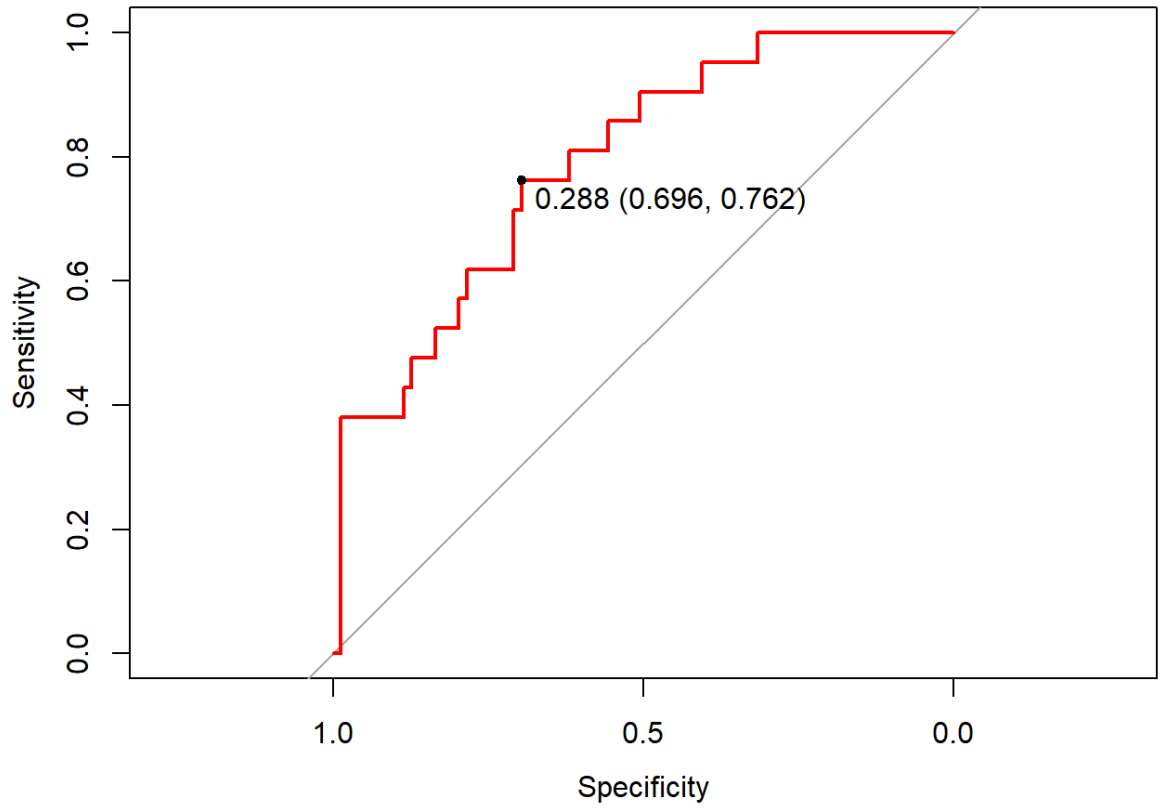
```
## Confusion Matrix and Statistics
```

```
##
##            Reference
## Prediction No Yes
##        No  25  16
##        Yes 54   5
##
##                Accuracy : 0.3
##                  95% CI : (0.2124, 0.3998)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.2677
##
##  Mcnemar's Test P-Value : 9.764e-06
##
##             Sensitivity : 0.23810
##             Specificity : 0.31646
##          Pos Pred Value : 0.08475
##          Neg Pred Value : 0.60976
##              Prevalence : 0.21000
##          Detection Rate : 0.05000
##    Detection Prevalence : 0.59000
##       Balanced Accuracy : 0.27728
##
##        'Positive' Class : Yes
##
```

```
# complex model ROC
complex1.roc.Valid<-
roc(response=testingDataframe$fracture,predictor=validateComplexPred,levels=c
("No","Yes"))
```

```
## Setting direction: controls < cases
```

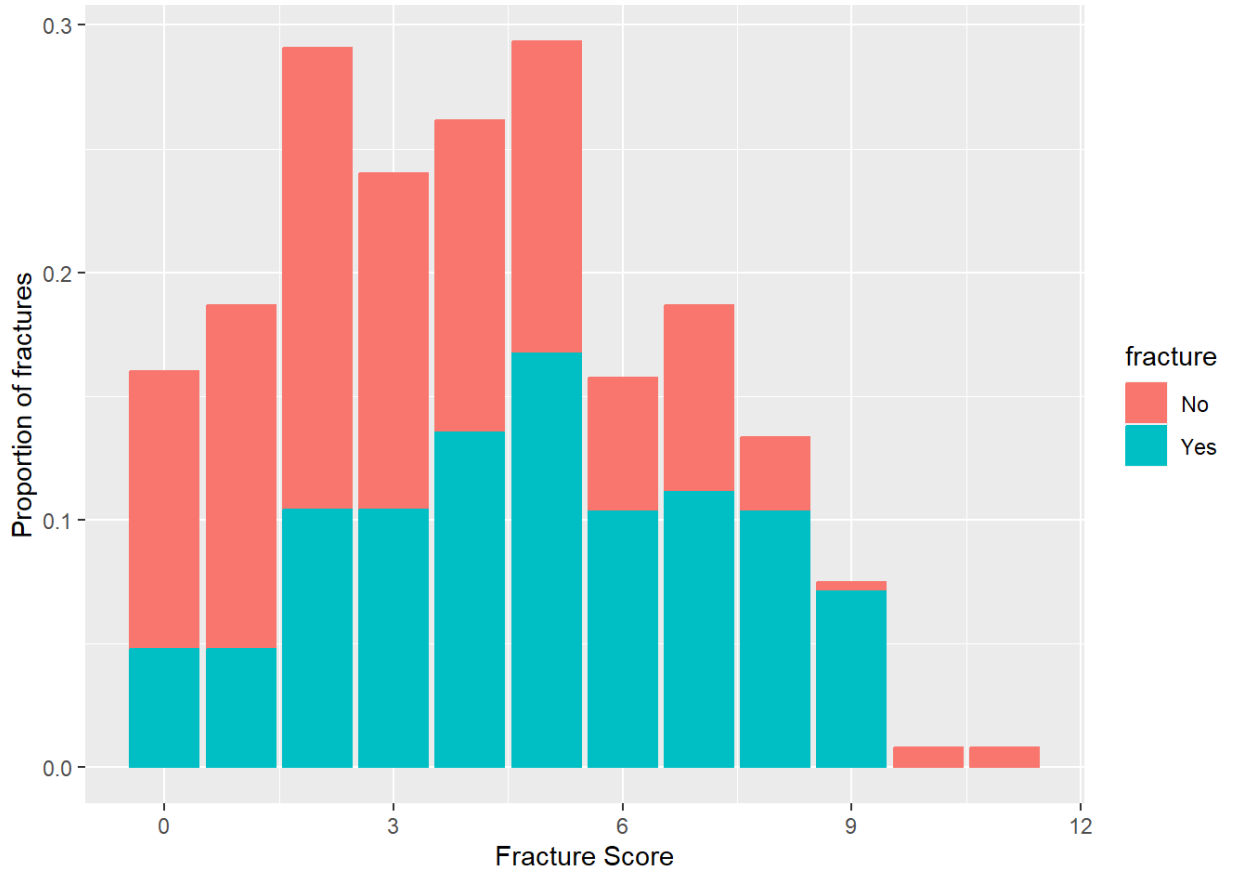
```
# plot ROC
plot(complex1.roc.Valid,print.thres="best",col="red")
```




```
## effects plots for complex model 1
library(sjPlot)
library(sjmisc)
#plot_model(complex1,type="pred",terms=c("fracscore","bonetreat",
"fracture")) # shows predictive probability of fracturing based on bonetreat
and fracscore
library(tidyr)
library(dplyr)


g1<-glow_bonemed %>%
  group_by(fracture,fracscore) %>%
  summarise(cnt=n()) %>%
  mutate(perc=round(cnt/sum(cnt),4))%>%
  arrange(desc(perc))
## `summarise()` has grouped output by 'fracture'. You can override using the
## `.groups` argument.
g1
```

```
## # A tibble: 22 × 4
## # Groups:   fracture [2]
##    fracture fracscore   cnt  perc
##    <fct>        <int> <int> <dbl>
##  1 No               2    70 0.187
##  2 Yes              5    21 0.168
##  3 No               1    52 0.139
##  4 No               3    51 0.136
##  5 Yes              4    17 0.136
##  6 No               4    47 0.125
##  7 No               5    47 0.125
##  8 No               0    42 0.112
##  9 Yes              7    14 0.112
## 10 Yes              2    13 0.104
## # ℹ 12 more rows
```

```
ggplot(g1,aes(x=fracscore,y=perc,colour=fracture))+
  geom_bar(aes(fill=fracture),show.legend=T,stat="identity")+
  ylab("Proportion of fractures")+
  xlab("Fracture Score")
```




```
library(caret)

fitControl<-
trainControl(method="repeatedcv",number=10,repeats=1,classProbs=TRUE,
summaryFunction=mnLogLoss)


set.seed(4)


#Version 1

lda.fit<-train(fracture ~ . - sub_id + priorfrac:fracscore + bmi:fracscore +
fracscore:bonetreat,

               data=trainingDataframe,

               method="lda",

               trControl=fitControl,

               preProc = c("center", "scale"),

               metric="logLoss")


lda.fit # logLoss = 0.5663
```

```
## Linear Discriminant Analysis
##
## 400 samples
##  17 predictor
##   2 classes: 'No', 'Yes'
##
## Pre-processing: centered (20), scaled (20)
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 359, 361, 361, 359, 360, 359, ...
## Resampling results:
##
##   logLoss
##   0.5662895
```

```
#Computing predicted probabilities on the training data
ldafit.predprobs<-predict(lda.fit, trainingDataframe, type = "prob")[,"Yes"]


summary(ldafit.predprobs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02687 0.11020 0.20730 0.25753 0.35907 0.90566
```

```
ldafit.roc<-roc(response=trainingDataframe$fracture,predictor=
ldafit.predprobs,levels=c("No","Yes"))
```

```
## Setting direction: controls < cases
```

```
# Now check validation in test set
set.seed(4)
validatePredictions <- predict(lda.fit, newdata = testingDataframe)


table(validatePredictions) # sanity check
```

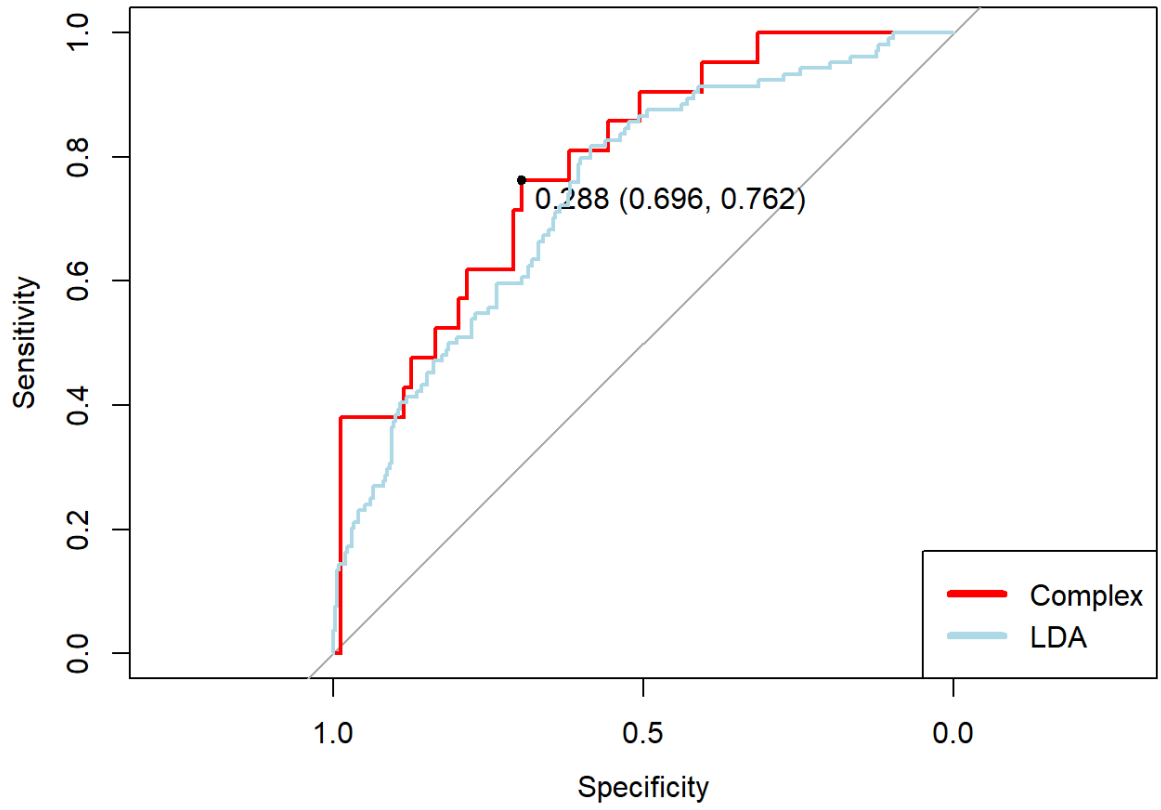
```
## validatePredictions
##  No Yes
##  87  13
```

```
# check confusion matrix positive class is no fracture
confusionMatrix(data = validatePredictions, reference =
testingDataframe$fracture, positive="No")
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction No Yes
##        No  75  12
##        Yes  4   9
##
##                Accuracy : 0.84
##                  95% CI : (0.7532, 0.9057)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 0.13316
##
##                   Kappa : 0.4394
##
##  Mcnemar's Test P-Value : 0.08012
##
##             Sensitivity : 0.9494
##             Specificity : 0.4286
##          Pos Pred Value : 0.8621
##          Neg Pred Value : 0.6923
##              Prevalence : 0.7900
##          Detection Rate : 0.7500
##    Detection Prevalence : 0.8700
##       Balanced Accuracy : 0.6890
##
##        'Positive' Class : No
##
```

```
# Plot both complex and lda models
plot(complex1.roc.Valid,print.thres="best",col="red")
plot(ldafit.roc, col="lightblue", add = T, legend = T)
 legend("bottomright",
       legend=c("Complex", "LDA"),
       col=c("red", "lightblue"),
       lwd=4, cex =1, xpd = TRUE, horiz = FALSE)
```




```
library(ranger)

# set tuning parameters using logloss

fitControl<-
trainControl(method="repeatedcv",number=5,repeats=1,classProbs=TRUE,
savePredictions = T)



randomForestModel<-train(fracture ~ . - sub_id,
                     data=trainingDataframe,
                     method="ranger",
                     trControl=fitControl,
                     preProc = c("center", "scale"))



summary(randomForestModel)
```

```
##                             Length Class          Mode
## predictions                 800    -none-         numeric
```

```
## num.trees                    1    -none-        numeric
## num.independent.variables    1    -none-        numeric
## mtry                         1    -none-        numeric
## min.node.size                1    -none-        numeric
## prediction.error             1    -none-        numeric
## forest                      10    ranger.forest list
## splitrule                    1    -none-        character
## treetype                     1    -none-        character
## call                         9    -none-        call
## importance.mode              1    -none-        character
## num.samples                  1    -none-        numeric
## replace                      1    -none-        logical
## dependent.variable.name      1    -none-        character
## xNames                      17    -none-        character
## problemType                  1    -none-        character
## tuneValue                    3    data.frame    list
## obsLevels                    2    -none-        character
## param                        0    -none-        list
```

```
randomForestModel$results
```

```
##   mtry min.node.size  splitrule Accuracy       Kappa AccuracySD    KappaSD
## 1    2             1       gini   0.7050 -0.02153427 0.02091650 0.04783884
## 2    2             1 extratrees   0.7250  0.01502437 0.01976424 0.05344202
## 3    9             1       gini   0.7350  0.13932665 0.02709935 0.08980599
## 4    9             1 extratrees   0.7125  0.09962772 0.03061862 0.07573492
## 5   17             1       gini   0.7350  0.13929980 0.02709935 0.08994861
## 6   17             1 extratrees   0.7150  0.12993548 0.03579455 0.10189421
```

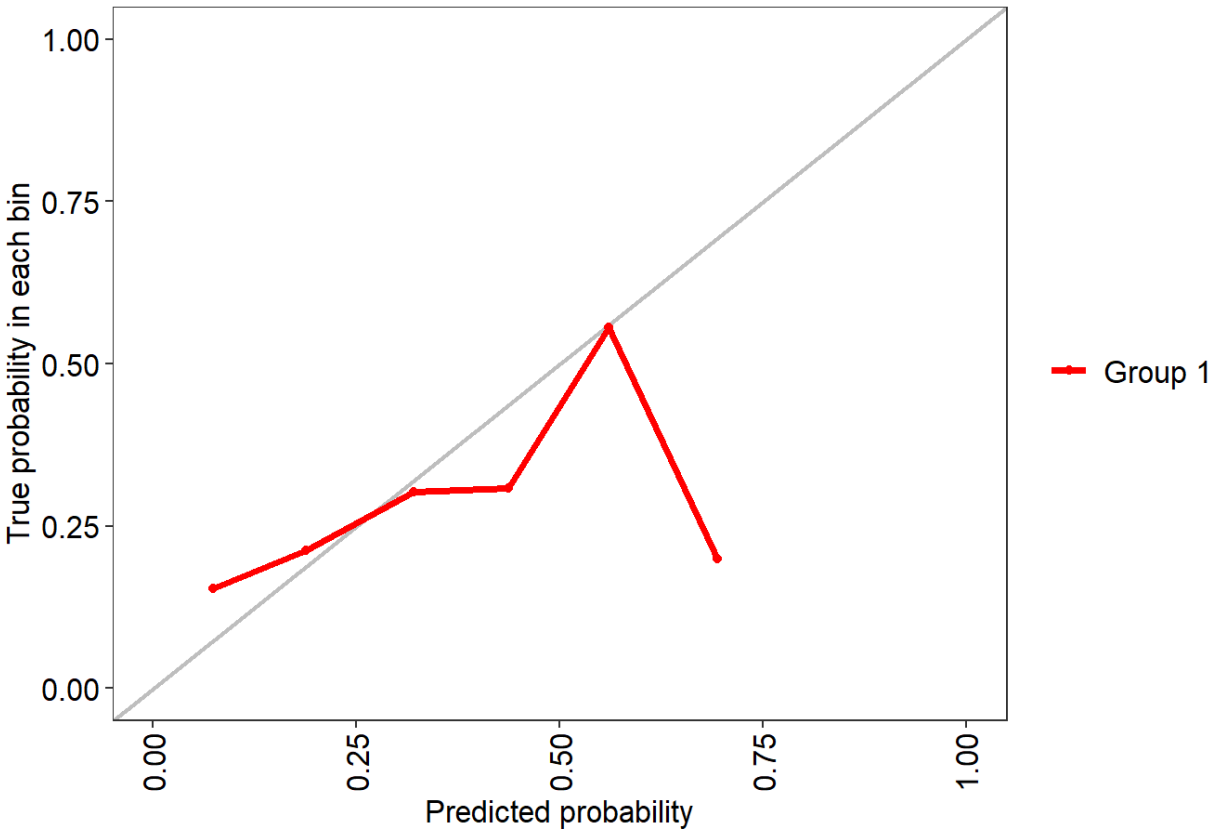
```
library(MLeval)
```

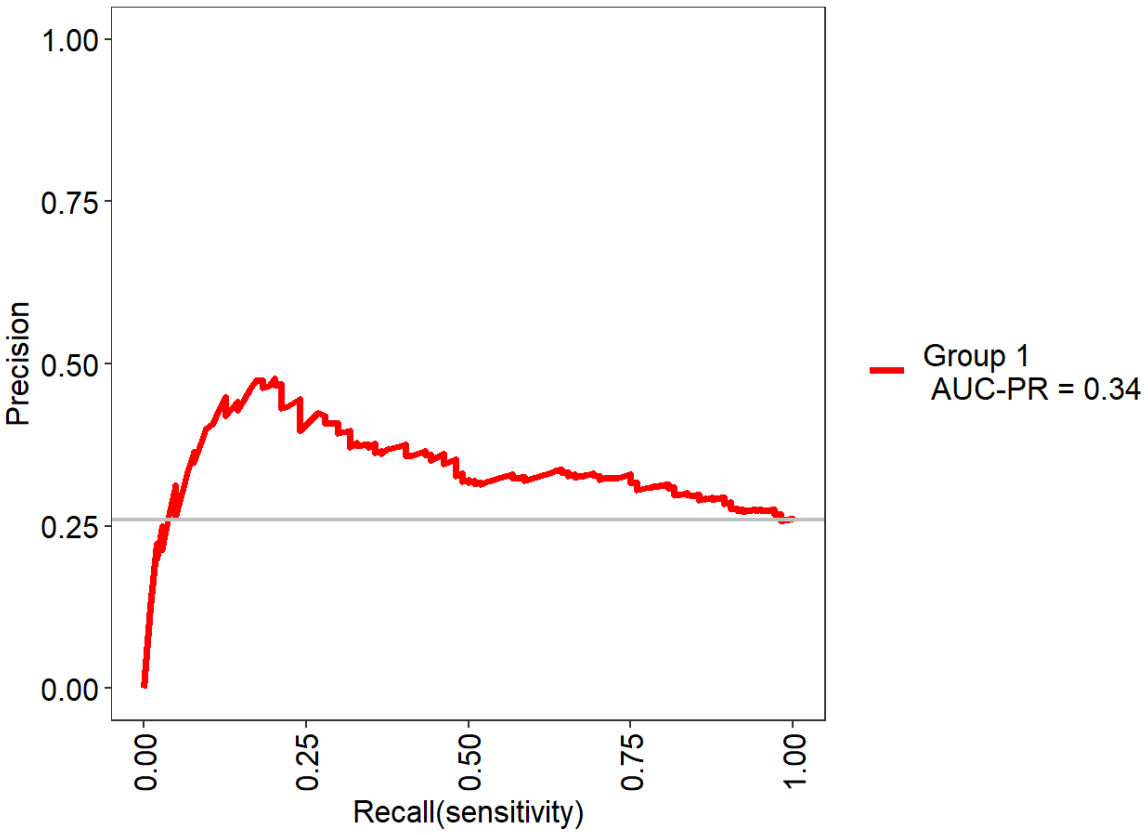
```
## Warning: package 'MLeval' was built under R version 4.3.3
```

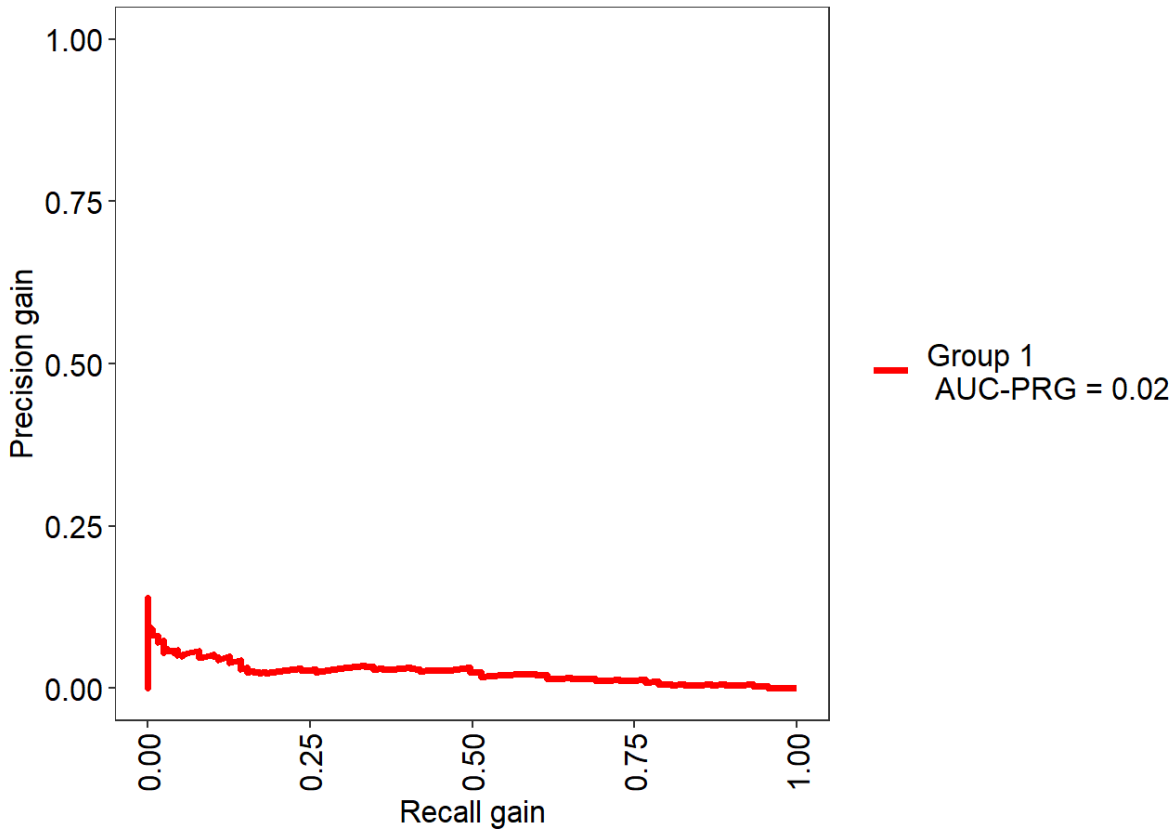
```
result <- evalm(randomForestModel)
```

```
## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Averaging probs.
## Group 1 type: repeatedcv
## Observations: 400
```

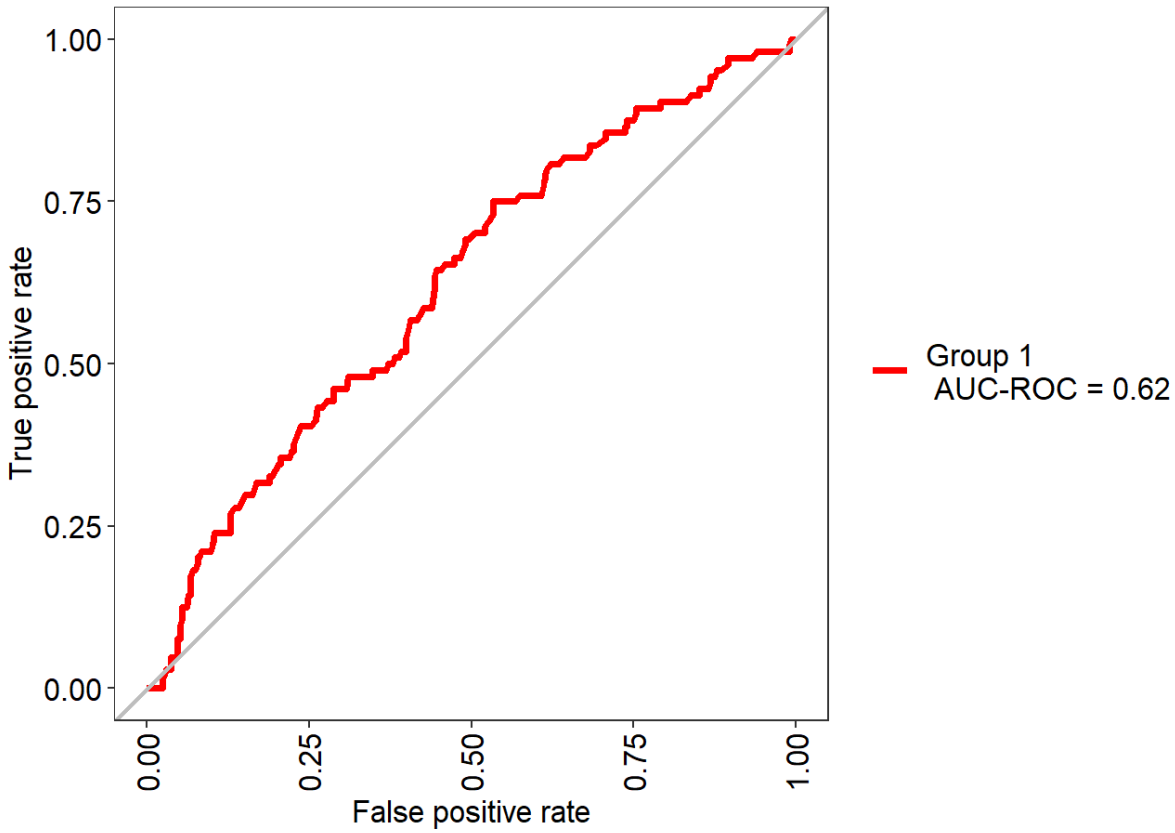
```
## Number of groups: 1
## Observations per group: 400
## Positive: Yes
## Negative: No
## Group: Group 1
## Positive: 104
## Negative: 296
## ***Performance Metrics***
```


1.00
0.75
0.50
0.25
0.00
True probability in each bin
0.00
0.25
0.50
0.75
1.00
Predicted probability
Group 1

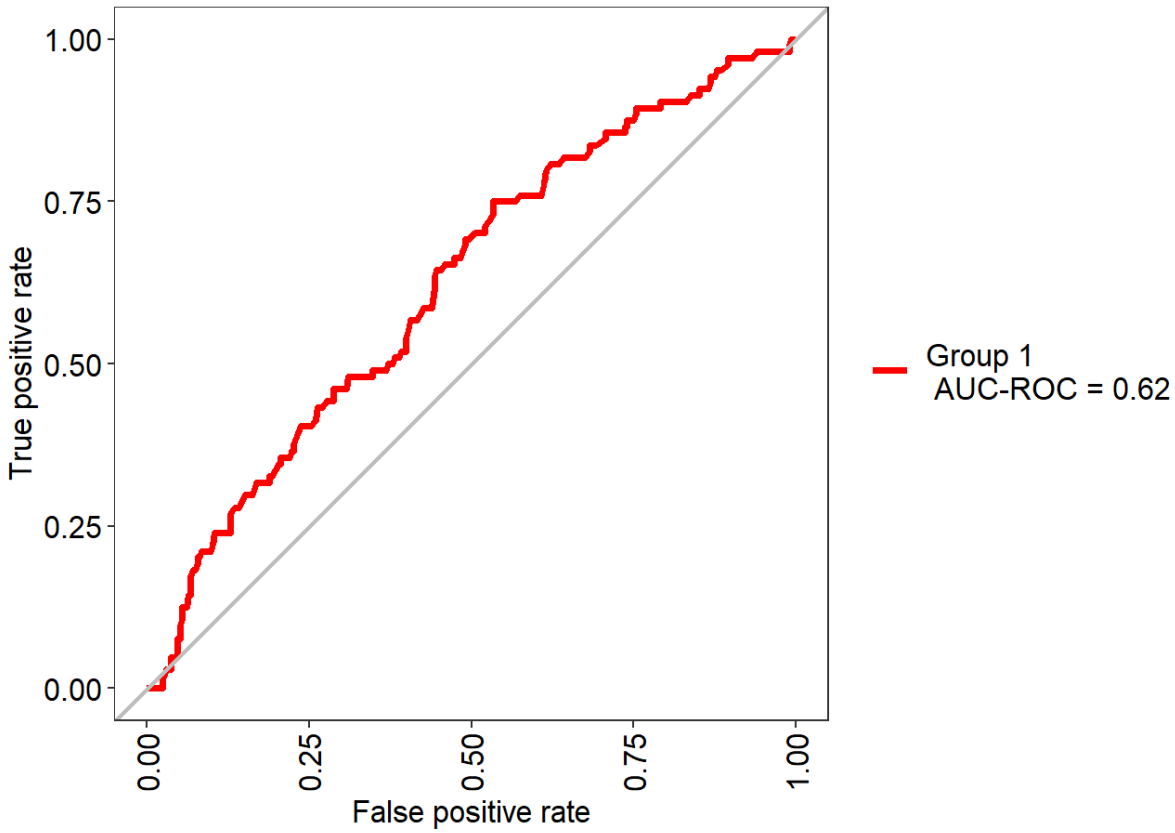

1.00
0.75
0.50
0.25
0.00
Precision
0.00
0.25
0.50
0.75
1.00
Recall(sensitivity)
Group 1
AUC-PR = 0.34




```
## Group 1 Optimal Informedness = 0.216216216216216
## Group 1 AUC-ROC = 0.62
```




```
#get AUROC
result$roc
```




```
#Computing predicted probabilities on the training data

rf.predprobs<-predict(randomForestModel, trainingDataframe, type =
"prob")[,"Yes"]


summary(rf.predprobs) # just looking
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0560  0.1160  0.2660  0.6565  0.9080
```

```
rffit.roc<-roc(response=trainingDataframe$fracture,predictor=
rf.predprobs,levels=c("No","Yes"))
```

```
## Setting direction: controls < cases
```

```
# Now check validation in test set

set.seed(4)

validatePredictions <- predict(randomForestModel, newdata = testingDataframe)


table(validatePredictions) # sanity check
```

```
## validatePredictions

##  No Yes
```

```
##  89  11
```

```
# check confusion matrix positive class is no fracture
confusionMatrix(data = validatePredictions, reference =
testingDataframe$fracture, positive="Yes")
```

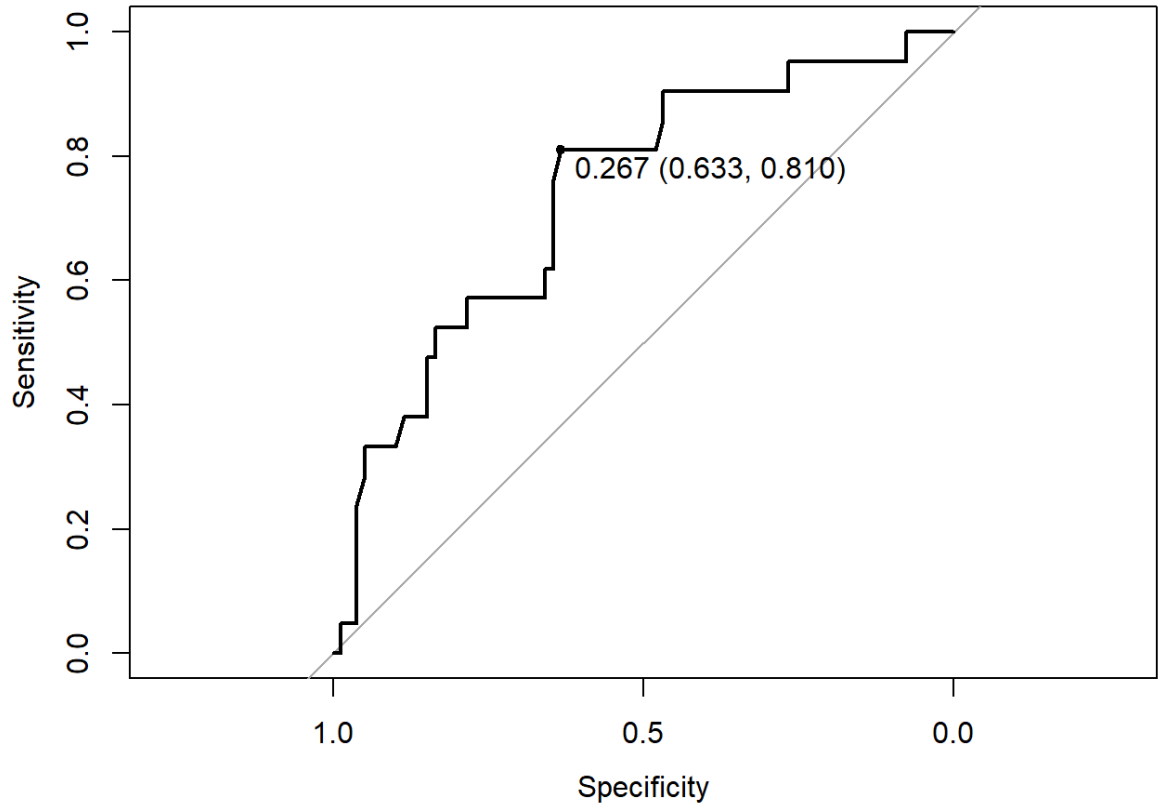
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  75  14
##        Yes  4   7
##
##                Accuracy : 0.82
##                  95% CI : (0.7305, 0.8897)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 0.27477
##
##                   Kappa : 0.3426
##
##  Mcnemar's Test P-Value : 0.03389
##
##             Sensitivity : 0.3333
##             Specificity : 0.9494
##          Pos Pred Value : 0.6364
##          Neg Pred Value : 0.8427
##              Prevalence : 0.2100
##          Detection Rate : 0.0700
##    Detection Prevalence : 0.1100
##       Balanced Accuracy : 0.6414
##
##        'Positive' Class : Yes
##
```

```
#### trying new way to get AUC-ROC for random forest model
#Computing predicted probabilities on the training data


# Prediction
```

```
rf.predicted.prob <- predict(randomForestModel, testingDataframe,
type="prob")[,"Yes"]


# draw ROC curve

rf.result.roc <- roc(response=testingDataframe$fracture,predictor=
rf.predicted.prob,levels=c("No","Yes"))
```

```
## Setting direction: controls < cases
```

```
# plot

plot(rf.result.roc, print.thres="best",
print.thres.best.method="closest.topleft")
```

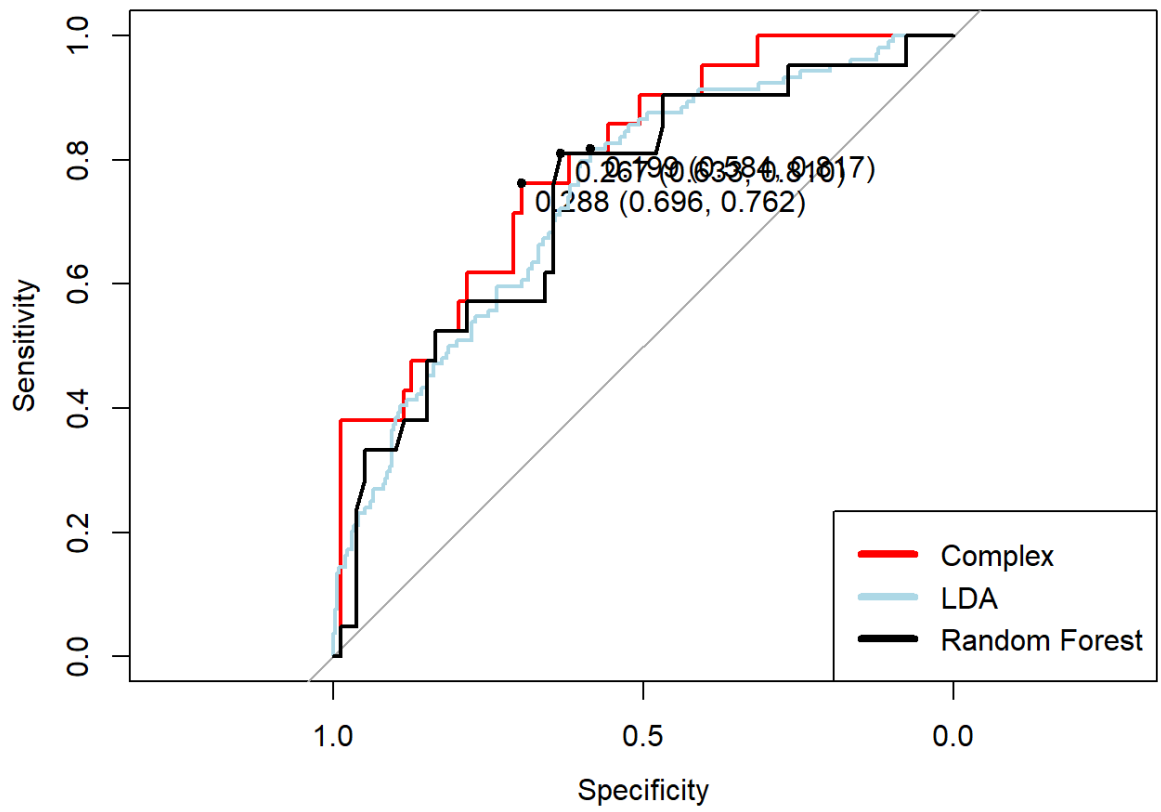



```
result.coords <- coords(rf.result.roc, "best", best.method="closest.topleft",
ret=c("threshold", "sensitivity", "specificity"))


print(result.coords)#to get threshold and sensitivity and specificity
```

```
##   threshold sensitivity specificity
## 1     0.267   0.8095238   0.6329114
```

```
# Plot complex, lda models, and random forest
```

```
plot(complex1.roc.Valid,print.thres="best",col="red")

plot(ldafit.roc, col="lightblue", add = T, legend = T, print.thres="best")

plot(rf.result.roc, print.thres="best",
print.thres.best.method="closest.topleft", add = T, col = "black")

 legend("bottomright",

       legend=c("Complex", "LDA", "Random Forest"),

       col=c("red", "lightblue", "black"),

       lwd=4, cex =1, xpd = TRUE, horiz = FALSE)
```




```
# get AUC-ROC for RF

auc(rf.result.roc) # AUC = 0.7306

## Area under the curve: 0.7366

# compare all object 2 models with AUC-ROC metrics (higher is better), so in
this case the LDA model was best

auc(complex1.roc) # AUC = 0.7163

## Area under the curve: 0.7163

auc(ldafit.roc) # AUC = 0.7436
```

```
## Area under the curve: 0.7436
```

```
auc(rf.result.roc) # AUC = 0.7306
```

```
## Area under the curve: 0.7366
```