Final Project Part 1

Andrew Marshall

12/15/2019

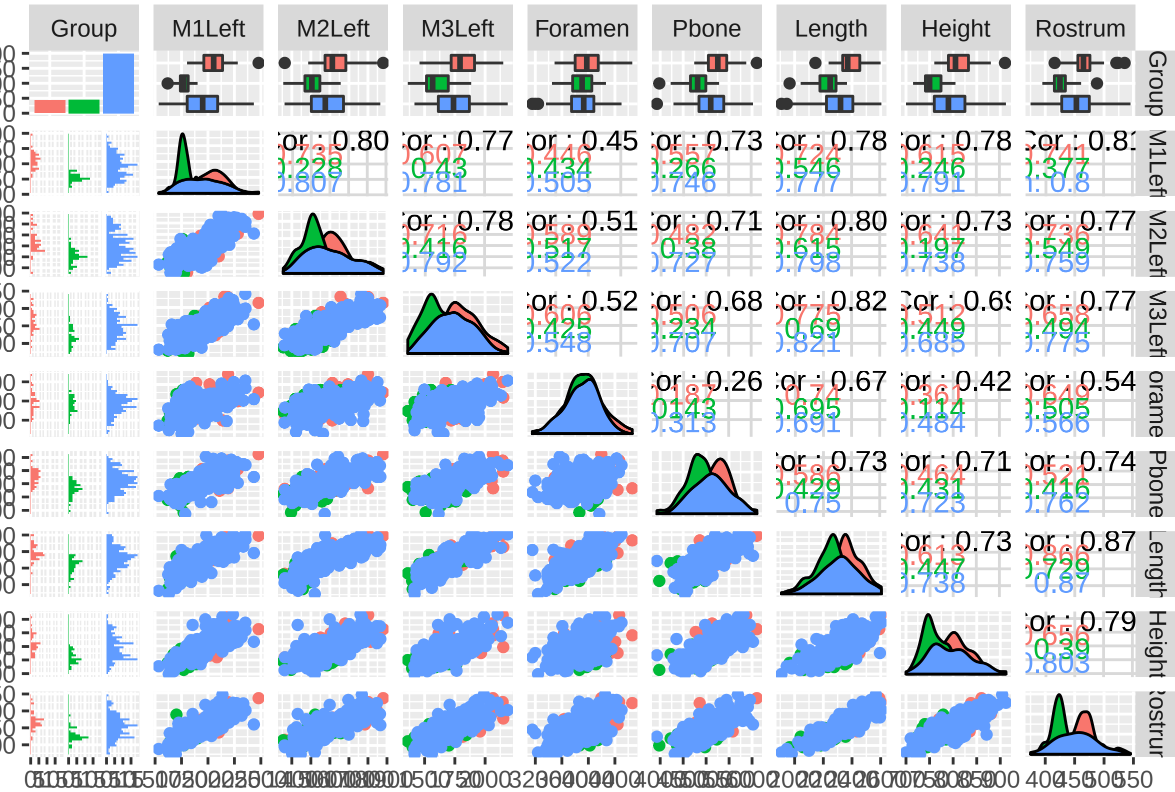
```r
#load Data
library(Flury)
data(microtus)
```

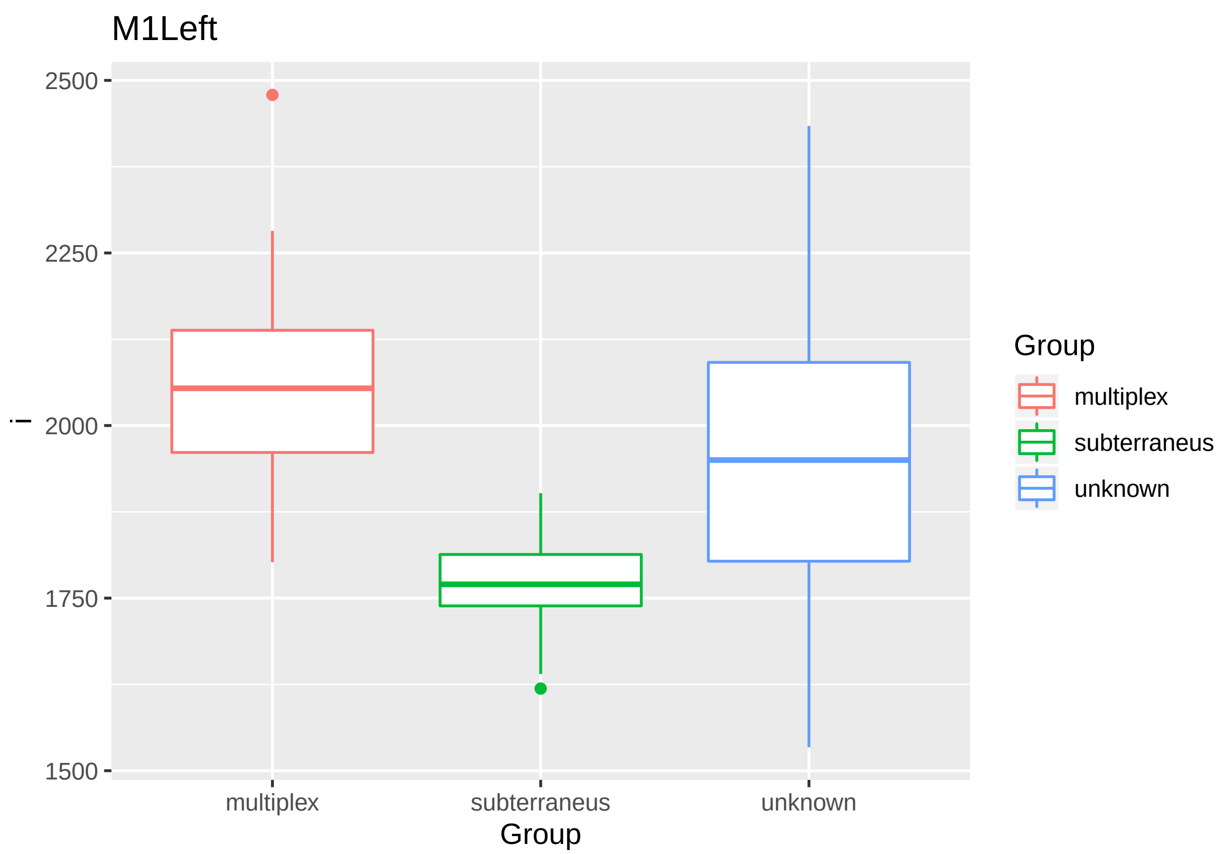
#Exploratory Analysis #Univaritate

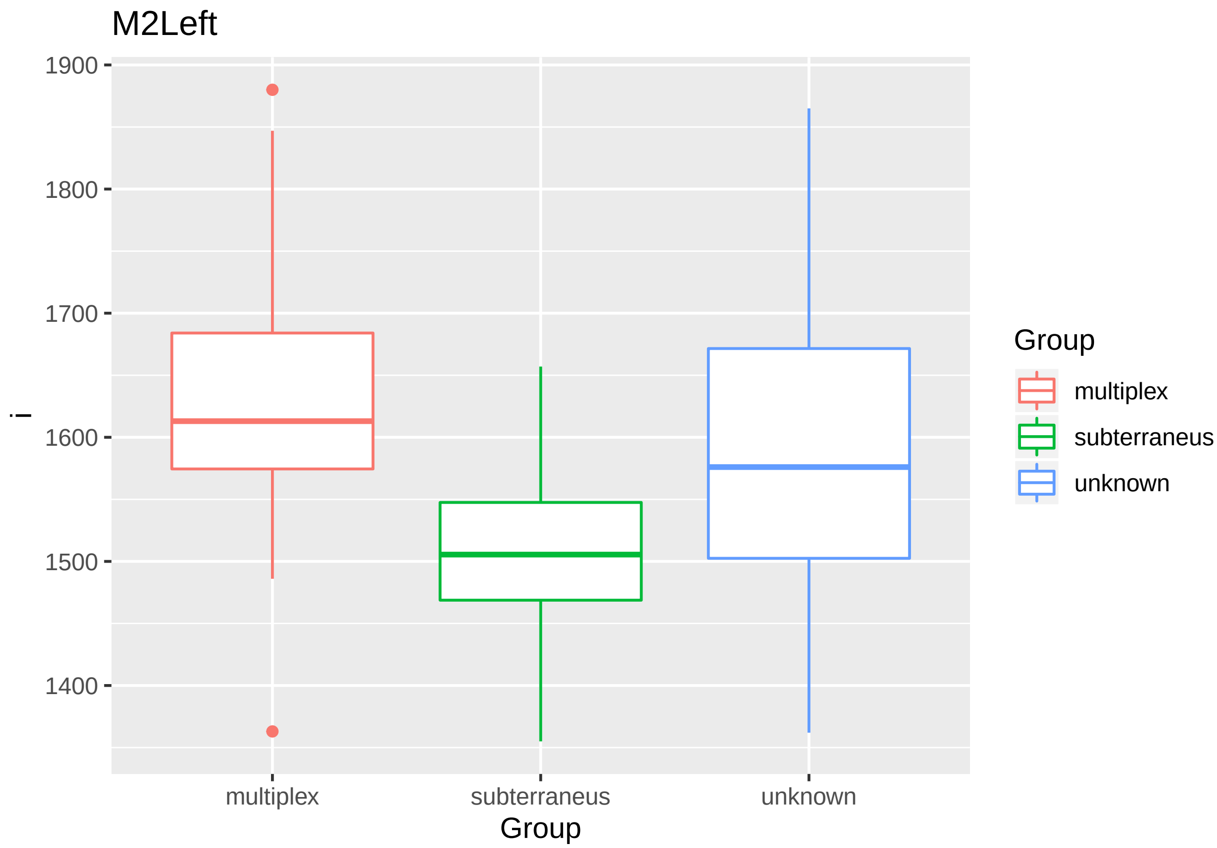
```
##           Group         M1Left         M2Left         M3Left
##  multiplex   : 43   Min.   :1534   Min.   :1355   Min.   :1361
##  subterraneus: 46   1st Qu.:1783   1st Qu.:1503   1st Qu.:1595
##  unknown     :199   Median :1923   Median :1570   Median :1724
##                     Mean   :1935   Mean   :1589   Mean   :1727
##                     3rd Qu.:2074   3rd Qu.:1660   3rd Qu.:1856
##                     Max.   :2479   Max.   :1880   Max.   :2187
##     Foramen         Pbone          Length         Height
##  Min.   :3155   Min.   :3928   Min.   :1908   Min.   :700.0
##  1st Qu.:3751   1st Qu.:4815   1st Qu.:2227   1st Qu.:759.2
##  Median :3932   Median :5079   Median :2312   Median :789.0
##  Mean   :3913   Mean   :5082   Mean   :2309   Mean   :790.8
##  3rd Qu.:4080   3rd Qu.:5328   3rd Qu.:2388   3rd Qu.:817.8
##  Max.   :4662   Max.   :6104   Max.   :2605   Max.   :912.0
##     Rostrum
##  Min.   :375.0
##  1st Qu.:425.0
##  Median :450.0
##  Mean   :451.2
##  3rd Qu.:475.0
##  Max.   :545.0
```

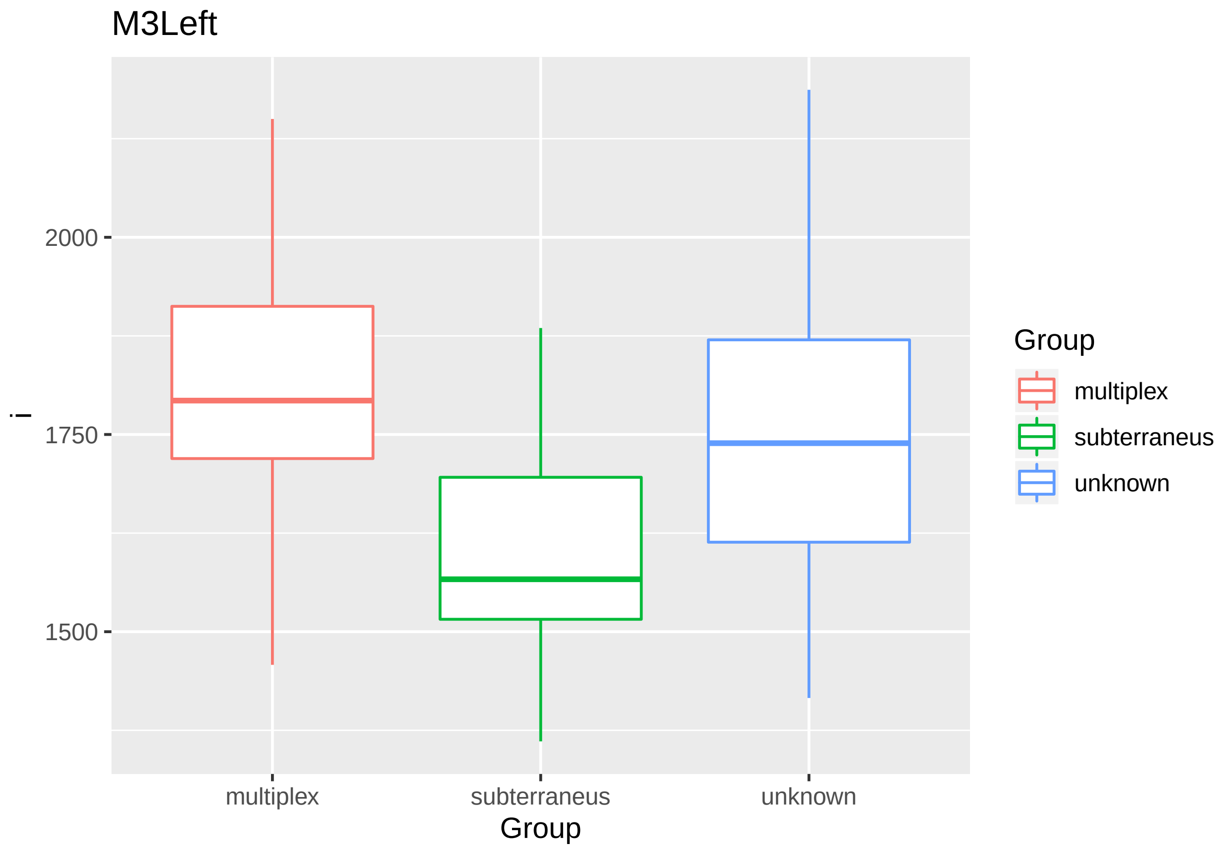
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

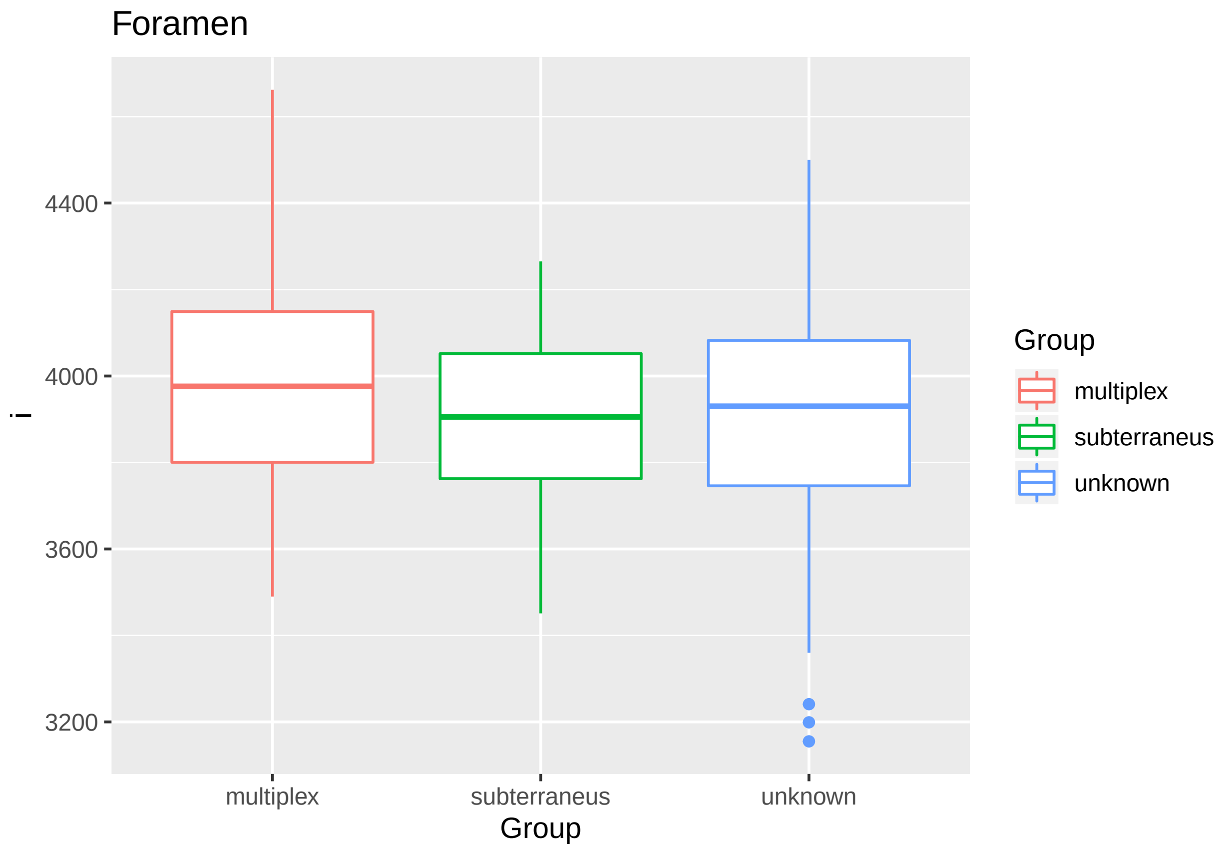
Microtus Summary

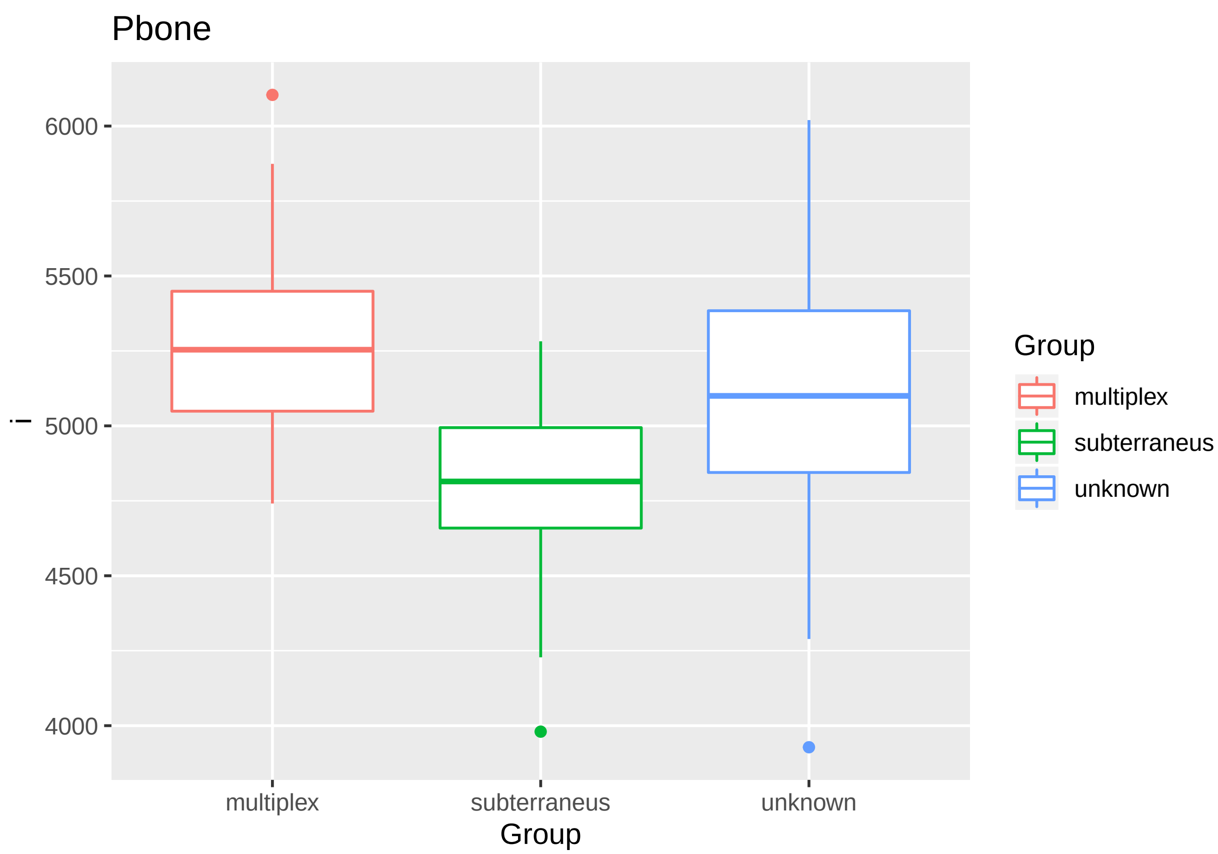


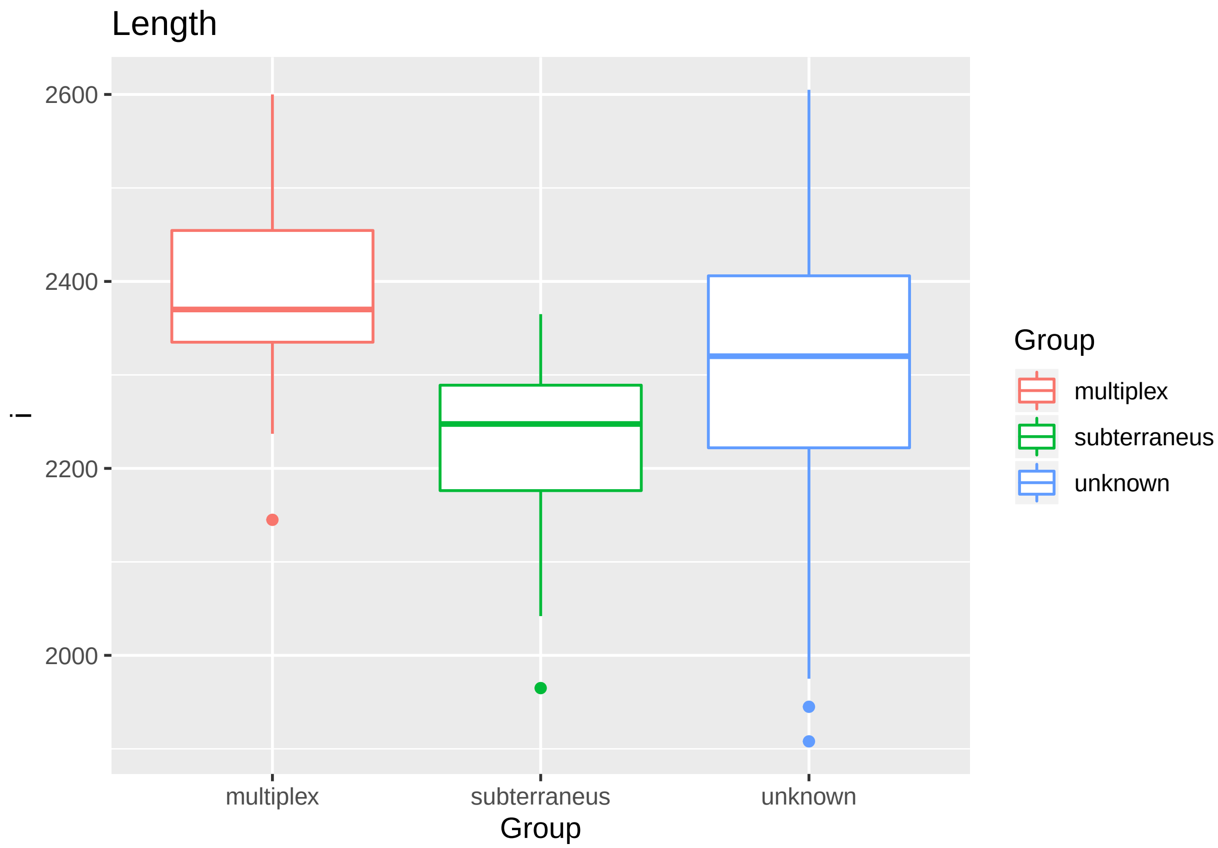

M1Left
2500
2250
2000
1750
1500
i
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown

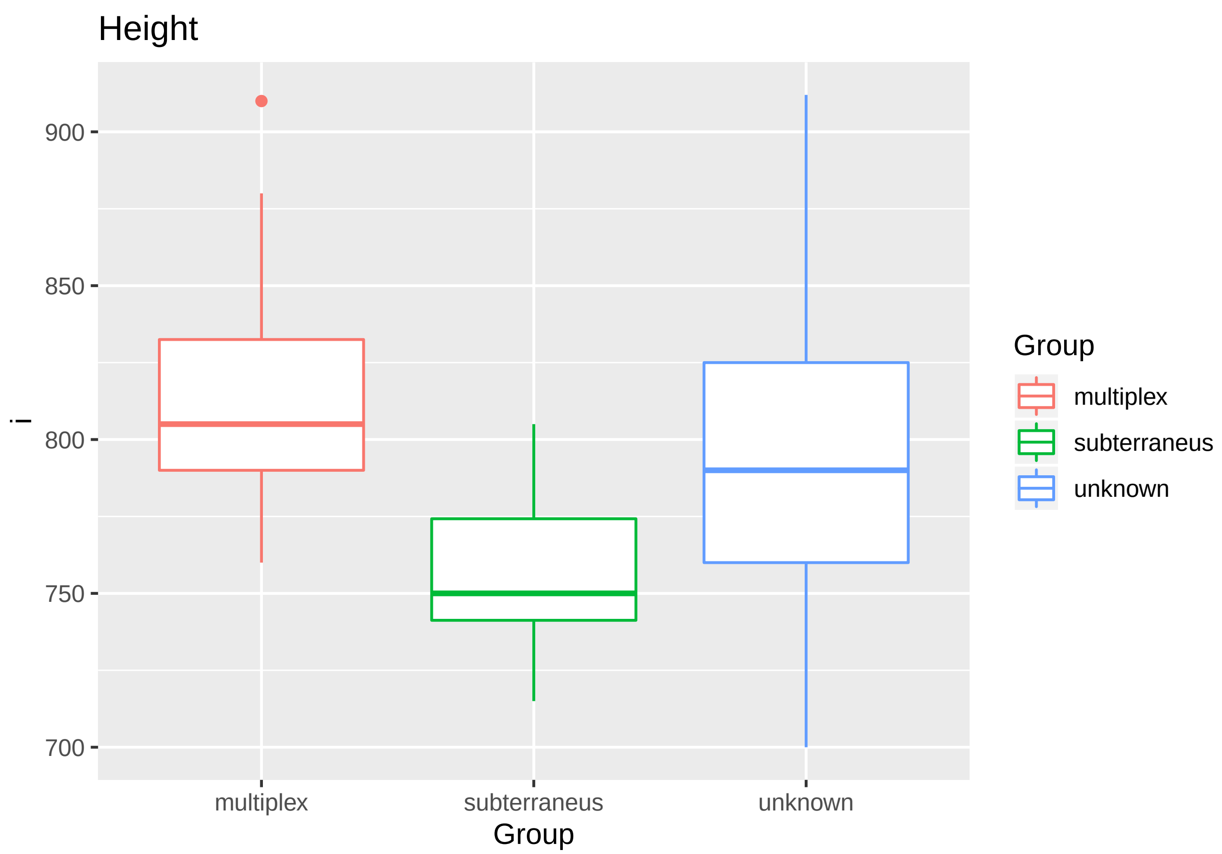

M2Left
1900
1800
1700
1600
1500
1400
i
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown

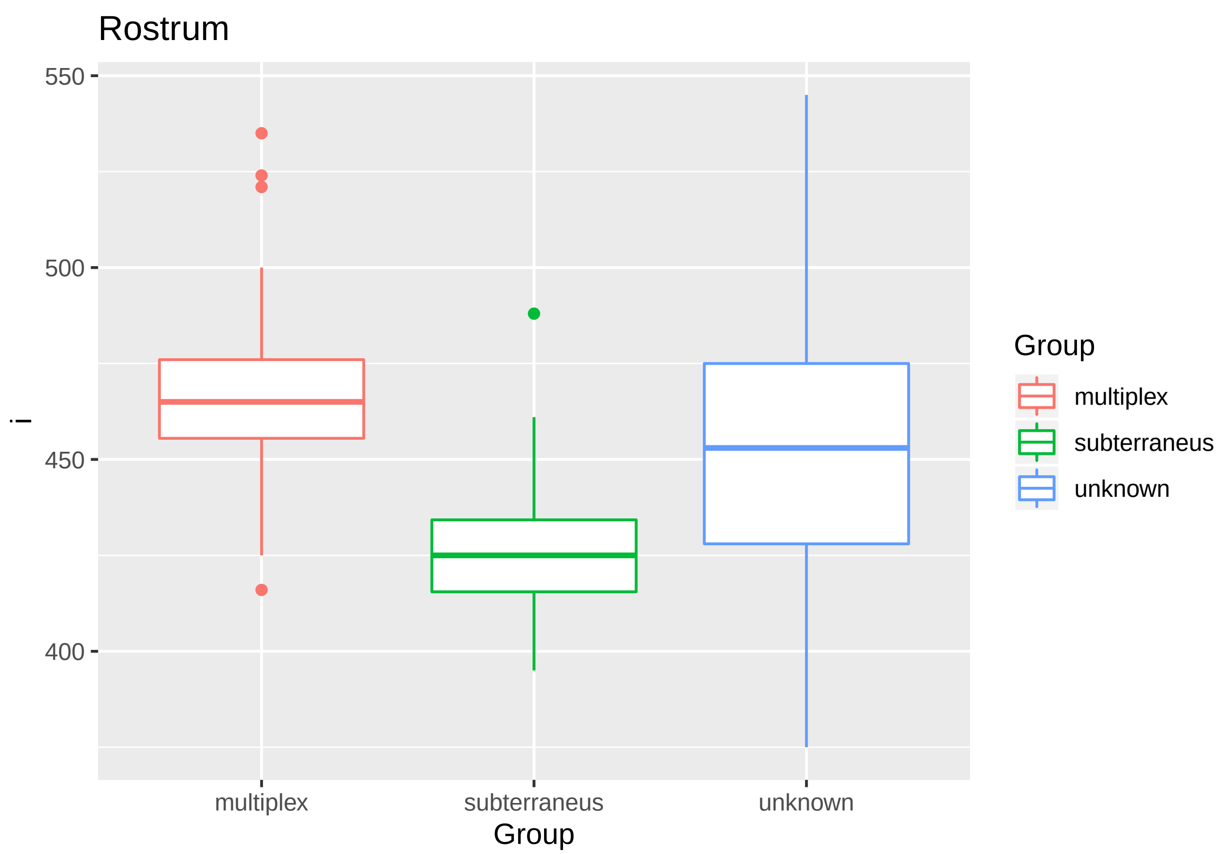

M3Left
i
2000
1750
1500
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown

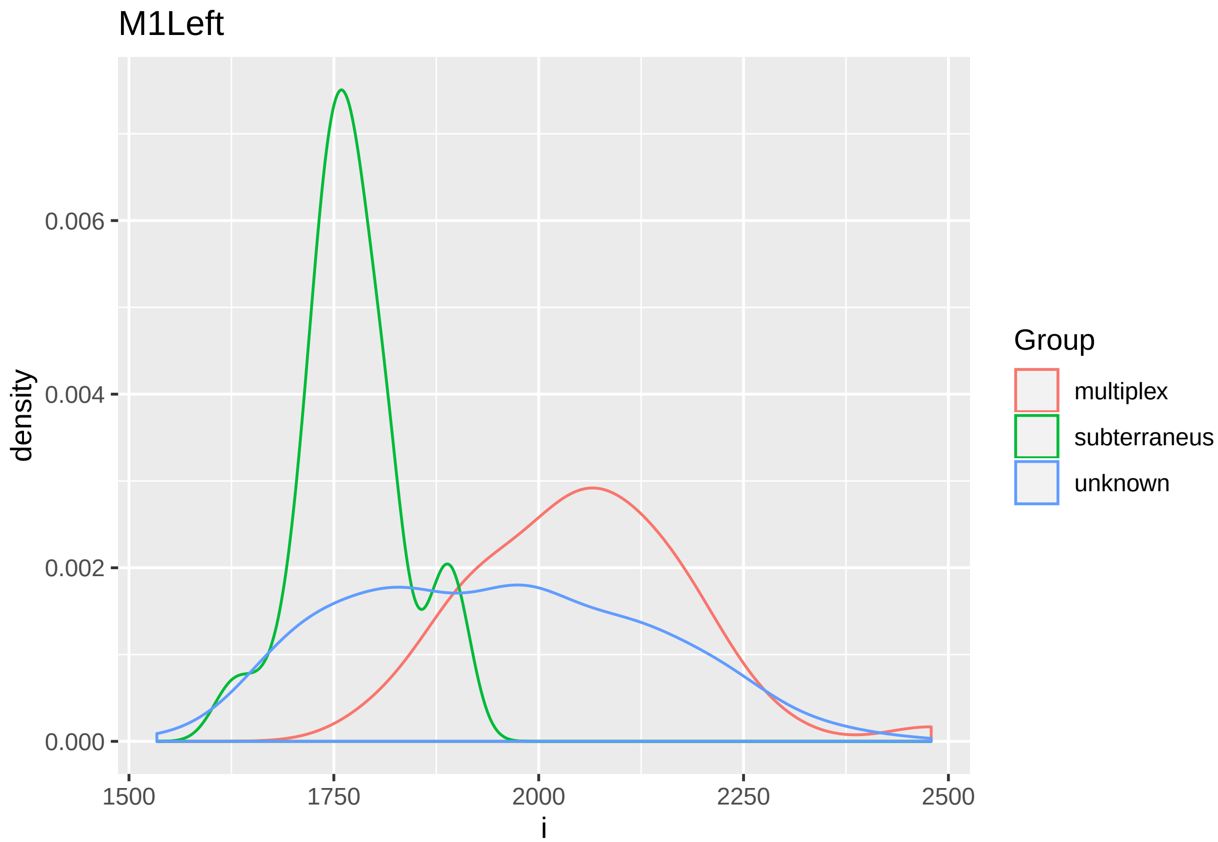

Foramen
4400
4000
3600
3200
i
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown

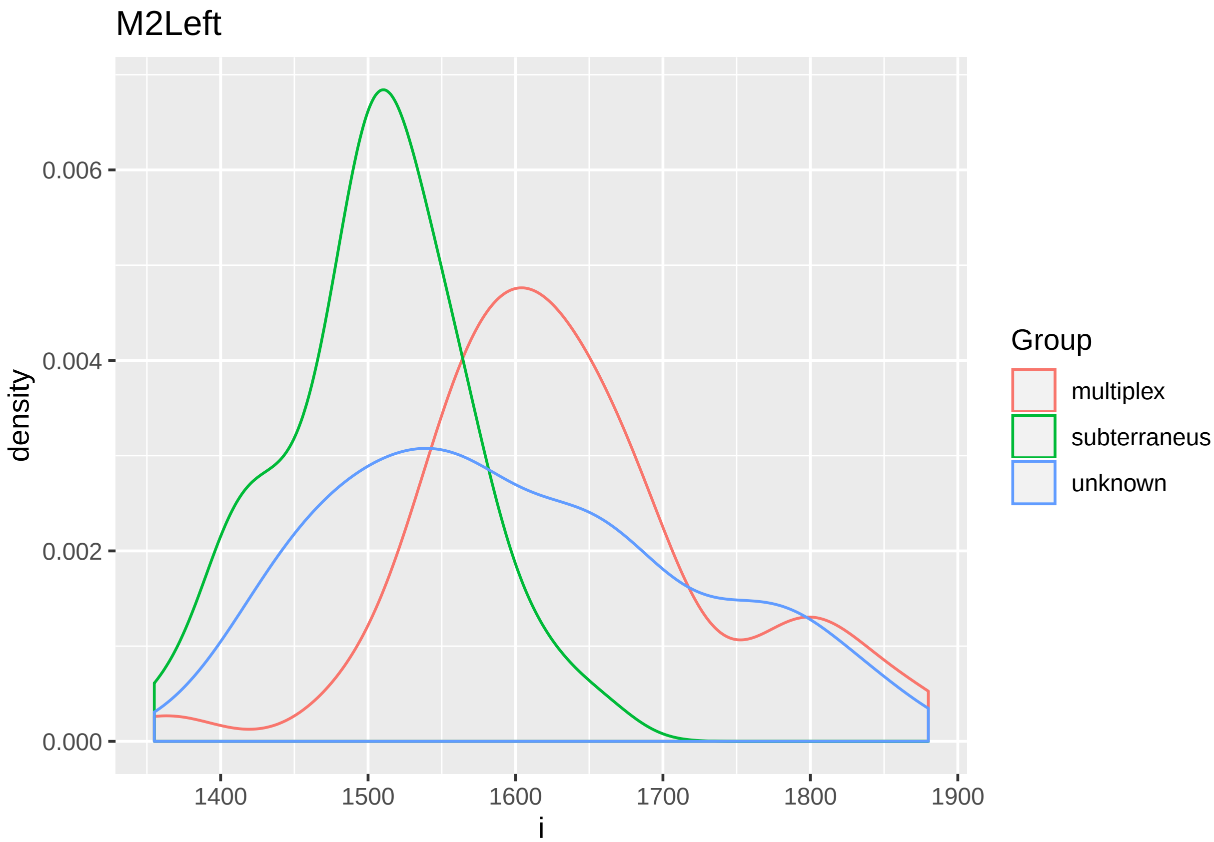

Pbone
6000
5500
5000
4500
4000
i
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown

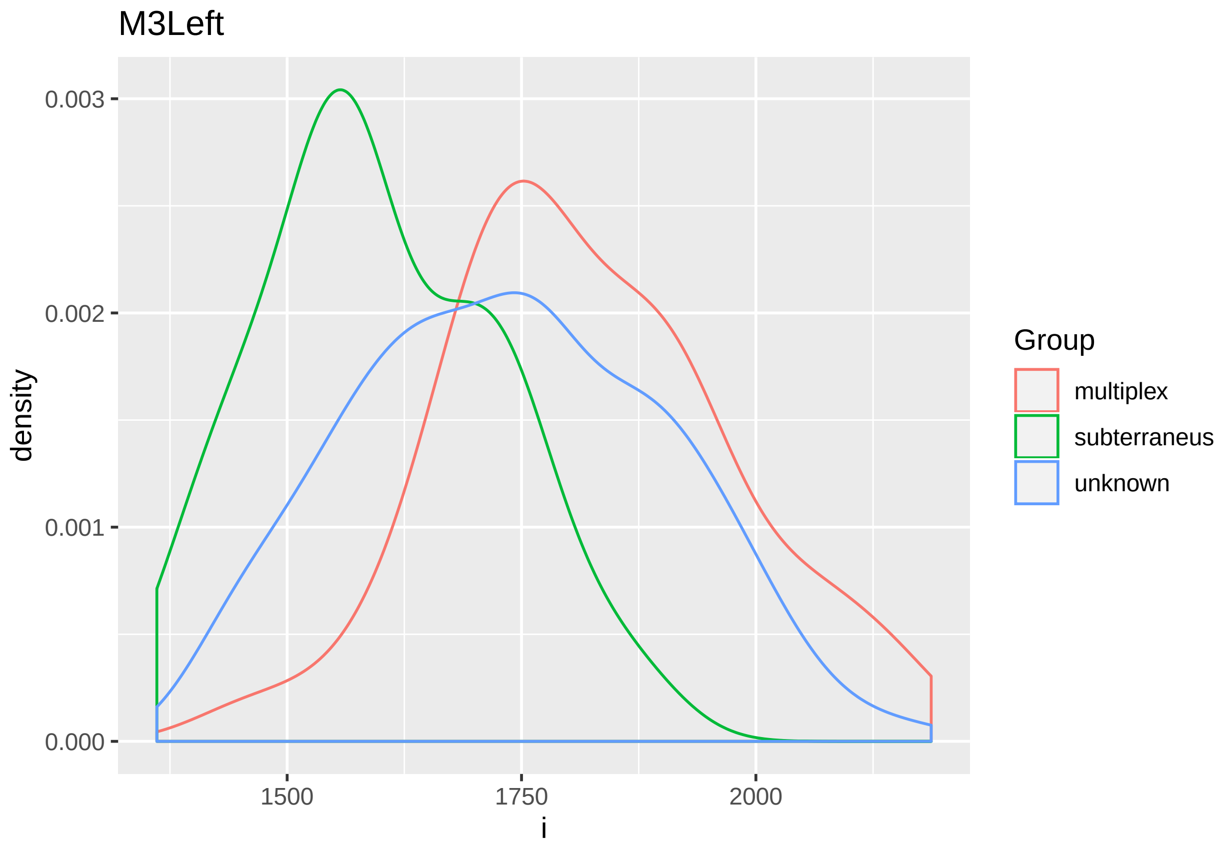

Length
2600
2400
2200
2000
l
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown


Height
900
850
800
750
700
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown

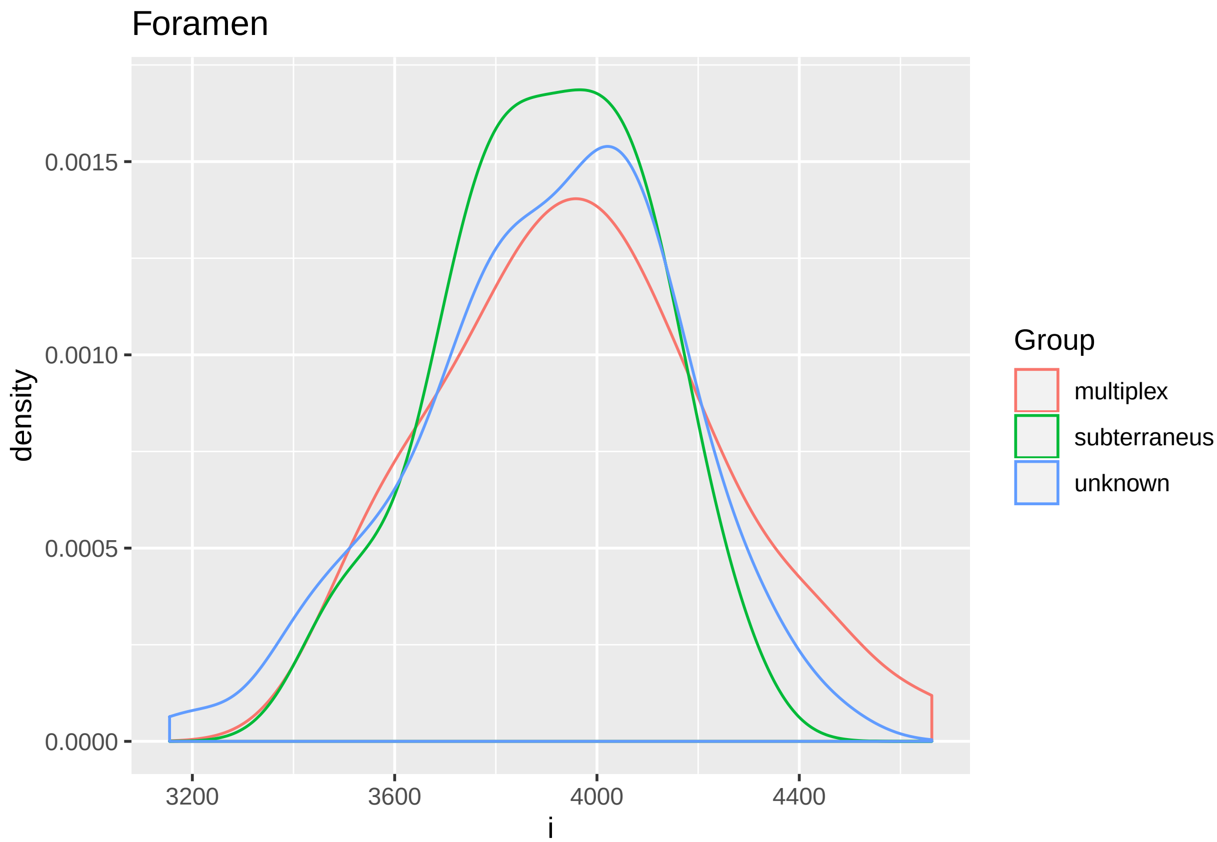

Rostrum
550
500
450
400
i
multiplex
subterraneus
unknown
Group
Group
multiplex
subterraneus
unknown


M1Left
density
0.006
0.004
0.002
0.000
1500
1750
2000
2250
2500
i
Group
multiplex
subterraneus
unknown

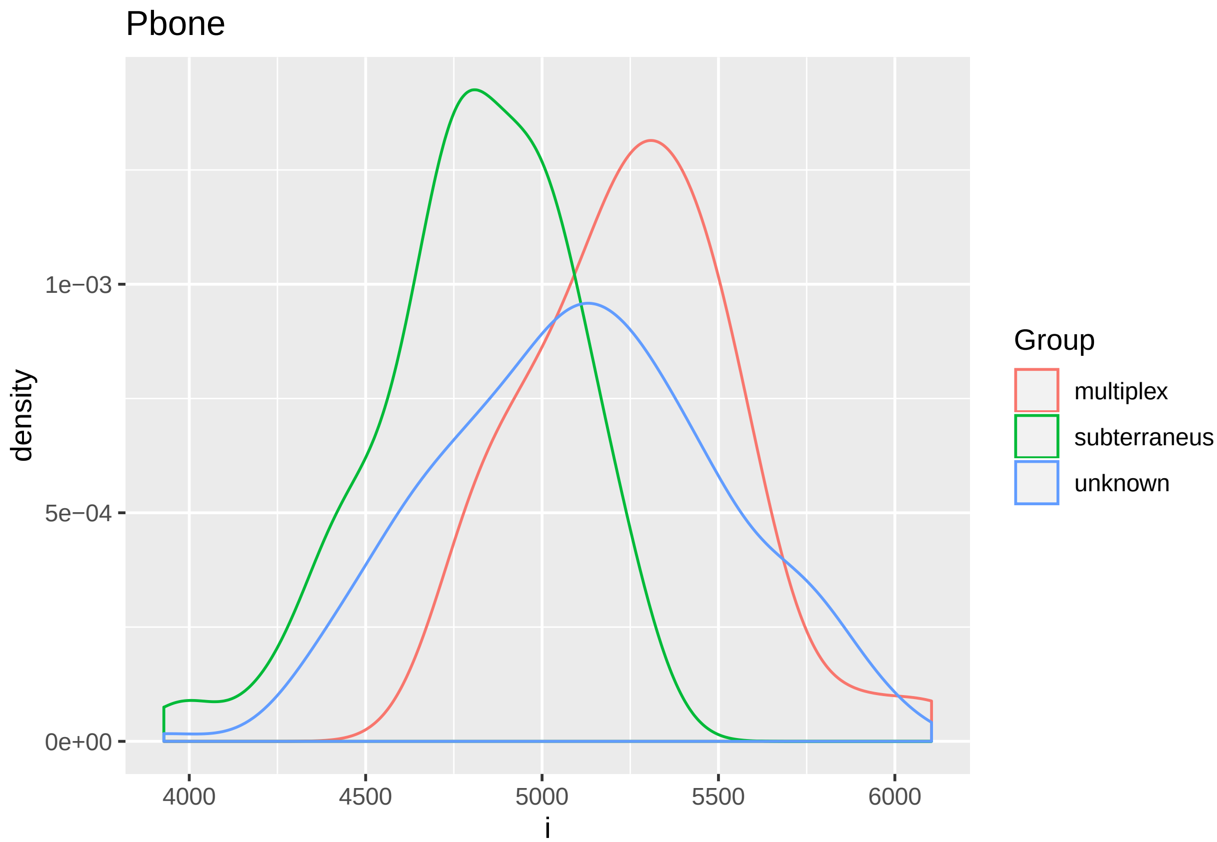

M2Left
density
0.006
0.004
0.002
0.000
1400
1500
1600
1700
1800
1900
i
Group
multiplex
subterraneus
unknown



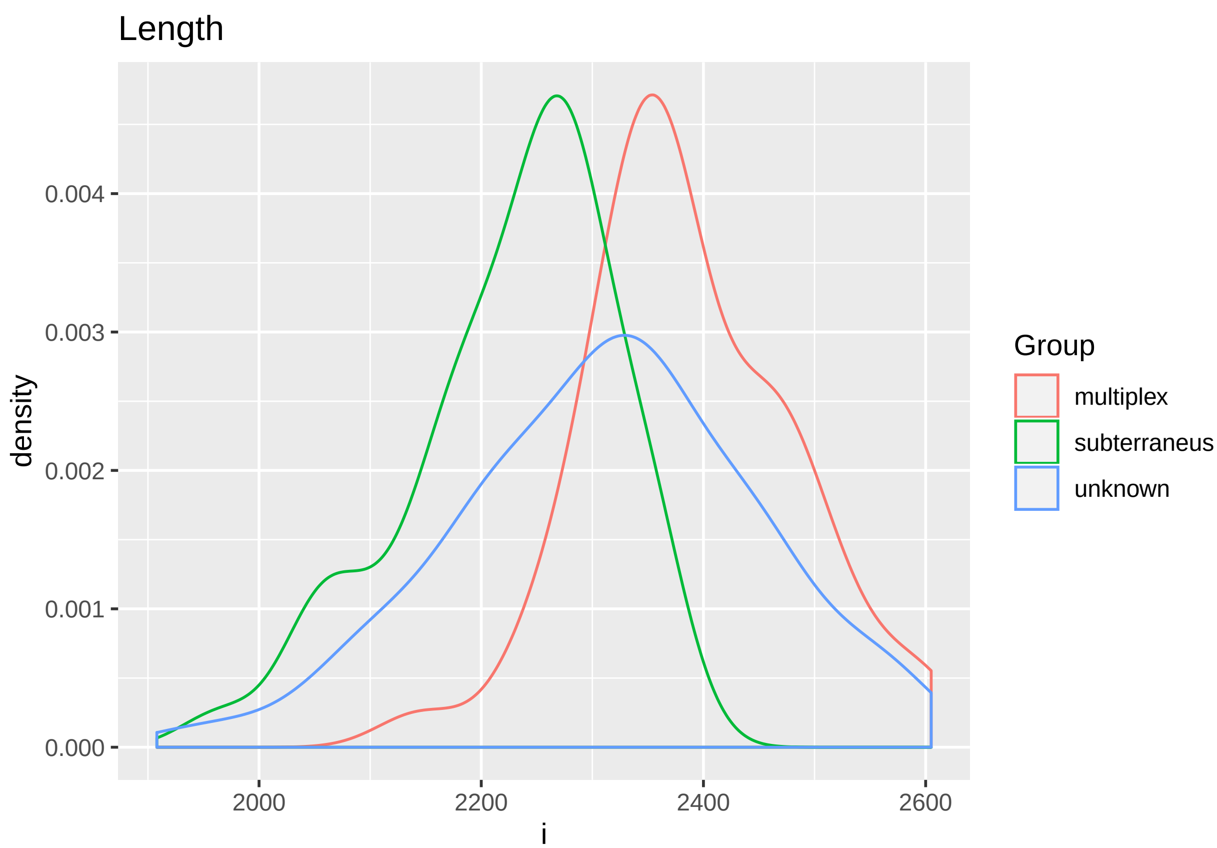
M3Left
0.003
0.002
0.001
0.000
density
1500
1750
2000
i
Group
multiplex
subterraneus
unknown

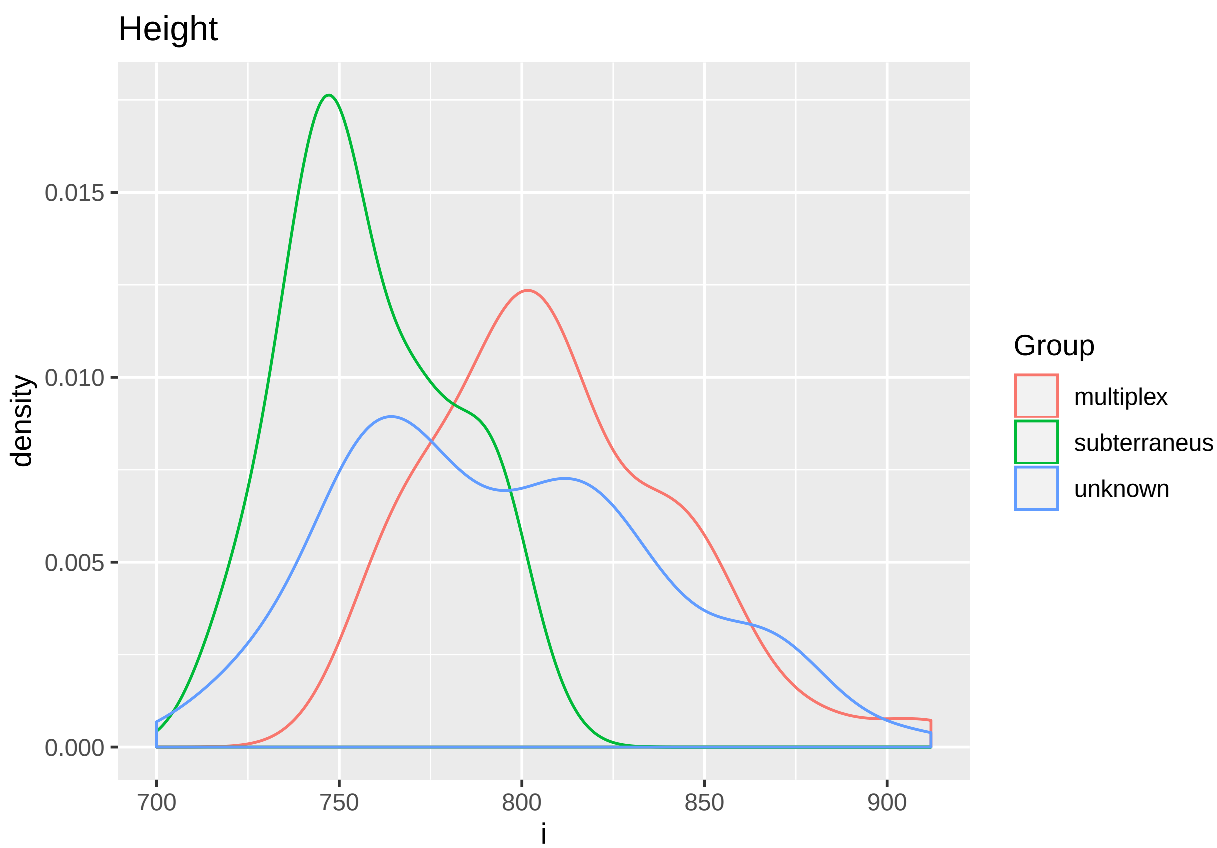


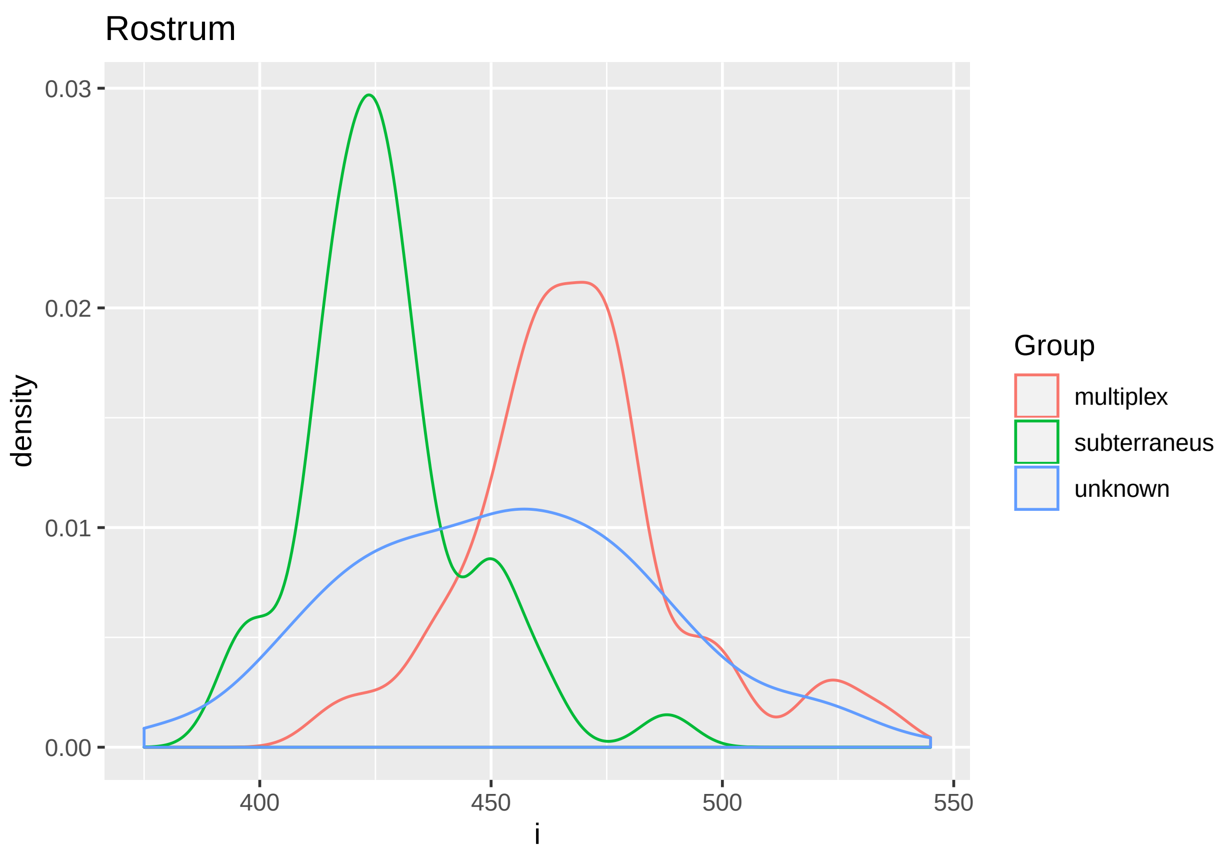
Foramen
0.0015
0.0010
0.0005
0.0000
density
3200
3600
4000
4400
i
Group
multiplex
subterraneus
unknown



Pbone
density
1e−03
5e−04
0e+00
4000
4500
5000
5500
6000
i
Group
multiplex
subterraneus
unknown


Length
density
0.004
0.003
0.002
0.001
0.000
2000
2200
2400
2600
i
Group
multiplex
subterraneus
unknown


Height
density
0.015
0.010
0.005
0.000
700
750
800
850
900
i
Group
multiplex
subterraneus
unknown



#correlation analysis #variables selection

#build base models #logistic regression

```
##           Group          M3Left          Foramen           Pbone
##  multiplex   :43   Min.   :1361   Min.   :3451   Min.   :3980
##  subterraneus:46   1st Qu.:1561   1st Qu.:3764   1st Qu.:4773
##                    Median :1712   Median :3941   Median :5004
##                    Mean   :1705   Mean   :3932   Mean   :5025
##                    3rd Qu.:1815   3rd Qu.:4078   3rd Qu.:5254
##                    Max.   :2150   Max.   :4662   Max.   :6104
##      Height
##  Min.   :715.0
##  1st Qu.:750.0
##  Median :776.0
##  Mean   :782.9
##  3rd Qu.:805.0
##  Max.   :910.0
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.52142  -0.36236   0.00866   0.13316   2.13858
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 94.627953  35.549137   2.662  0.00777 **
## M3Left      -0.015172   0.007357  -2.062  0.03918 *
## Foramen     -0.001864   0.002426  -0.768  0.44224
## Pbone       -0.003345   0.002604  -1.285  0.19889
## Height      -0.056895   0.024388  -2.333  0.01965 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 74.786  on 53  degrees of freedom
## Residual deviance: 27.051  on 49  degrees of freedom
## AIC: 37.051
##
## Number of Fisher Scoring iterations: 7
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.703704e-01   7.407407e-01   7.509878e-01   9.462570e-01   5.185185e-01
## AccuracyPValue  McnemarPValue
##   4.922101e-08   1.000000e+00
```

#Manual Feature Elimination

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8443  -0.5576   0.1478   0.4563   1.9320
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 55.49945   14.57006   3.809 0.000139 ***
## Height      -0.07105    0.01867  -3.806 0.000141 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 74.786  on 53  degrees of freedom
## Residual deviance: 40.717  on 52  degrees of freedom
## AIC: 44.717
##
## Number of Fisher Scoring iterations: 6
```

#Built in Feature selection with glmnet

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                         1
```

```
## (Intercept)  1.376597e+01
## M3Left      -1.905660e-03
## Foramen     -3.648563e-05
## Pbone       -7.924270e-04
## Height      -8.121889e-03
```

#GLM Summary

```r
#binary lables to pass to ROCR for ROC curve
microtus_Train <- microtus_Train %>% mutate(
  group_flag = if_else(Group == "multiplex", 1, 0)
)

#create predictions with probabilities
glm_1_pred = predict(glm_1, type = "prob")
glm_2_pred = predict(glm_2, type = "prob")
glm_3_pred = predict(glm_3, type = "prob")

#create prediction objects with multiplex column (note need to select just one column)
pred_1 <- prediction(glm_1_pred$multiplex, as.numeric(microtus_Train$group_flag))
pred_2 <- prediction(glm_2_pred$multiplex, as.numeric(microtus_Train$group_flag))
pred_3 <- prediction(glm_3_pred$multiplex, as.numeric(microtus_Train$group_flag))

#?performance
roc.perf_1 <- performance(pred_1, measure = "tpr", x.measure = "fpr")
roc.perf_1_AUC <- performance(pred_1, measure = "auc")
glm_1_pred_AUC<- roc.perf_1_AUC@y.values

roc.perf_2 <- performance(pred_2, measure = "tpr", x.measure = "fpr")
roc.perf_2_AUC <- performance(pred_2, measure = "auc")
glm_2_pred_AUC<- roc.perf_2_AUC@y.values

roc.perf_3 <- performance(pred_3, measure = "tpr", x.measure = "fpr")
roc.perf_3_AUC <- performance(pred_3, measure = "auc")
glm_3_pred_AUC<- roc.perf_3_AUC@y.values


#calc AUC
"glm_1_pred_AUC"
```

```
## [1] "glm_1_pred_AUC"
```

```r
glm_1_pred_AUC
```

```
## [[1]]
## [1] 0.9491758
```

```r
"glm_2_pred_AUC"
```

```
## [1] "glm_2_pred_AUC"
```

```
glm_2_pred_AUC
```

```
## [[1]]
## [1] 0.9086538
```

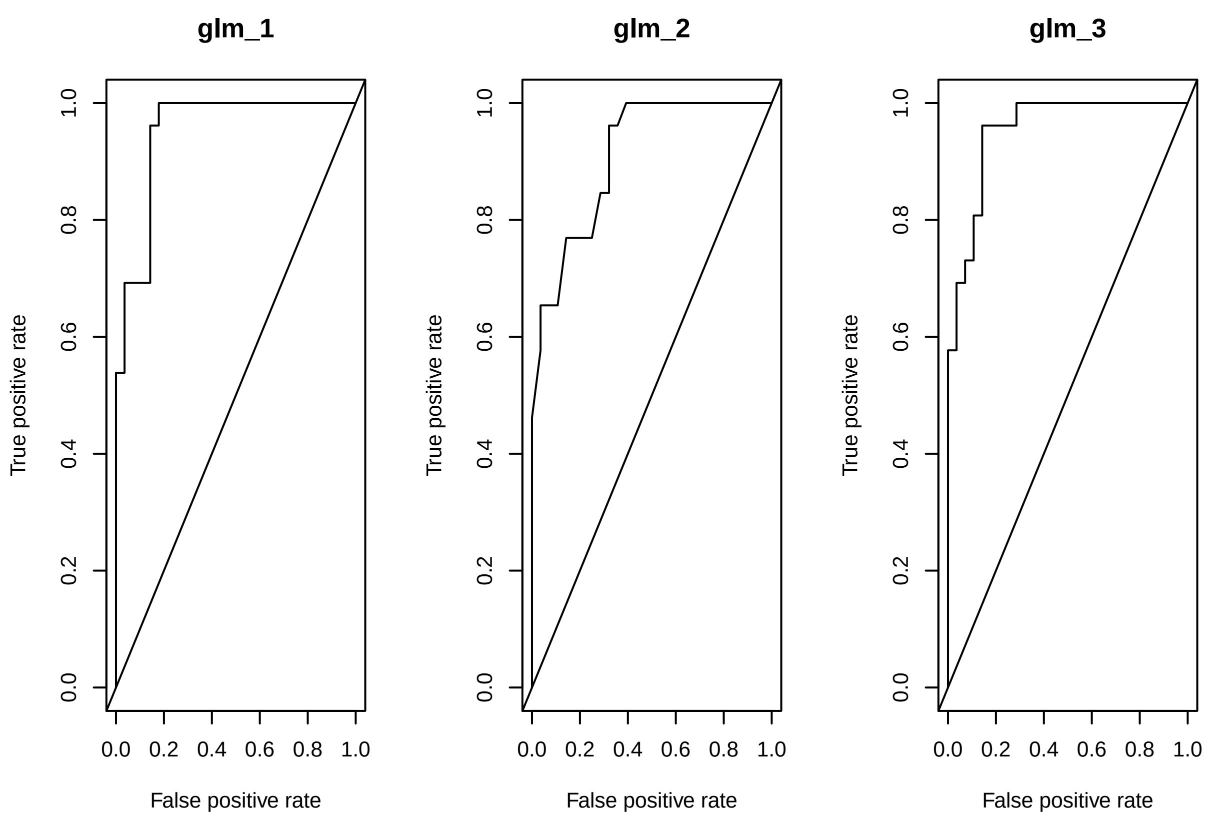
```
"glm_3_pred_AUC"
```

```
## [1] "glm_3_pred_AUC"
```

```
glm_3_pred_AUC
```

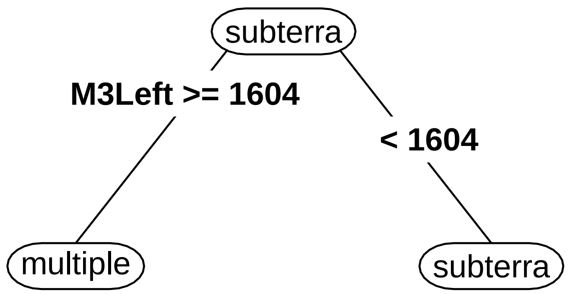
```
## [[1]]
## [1] 0.9519231
```

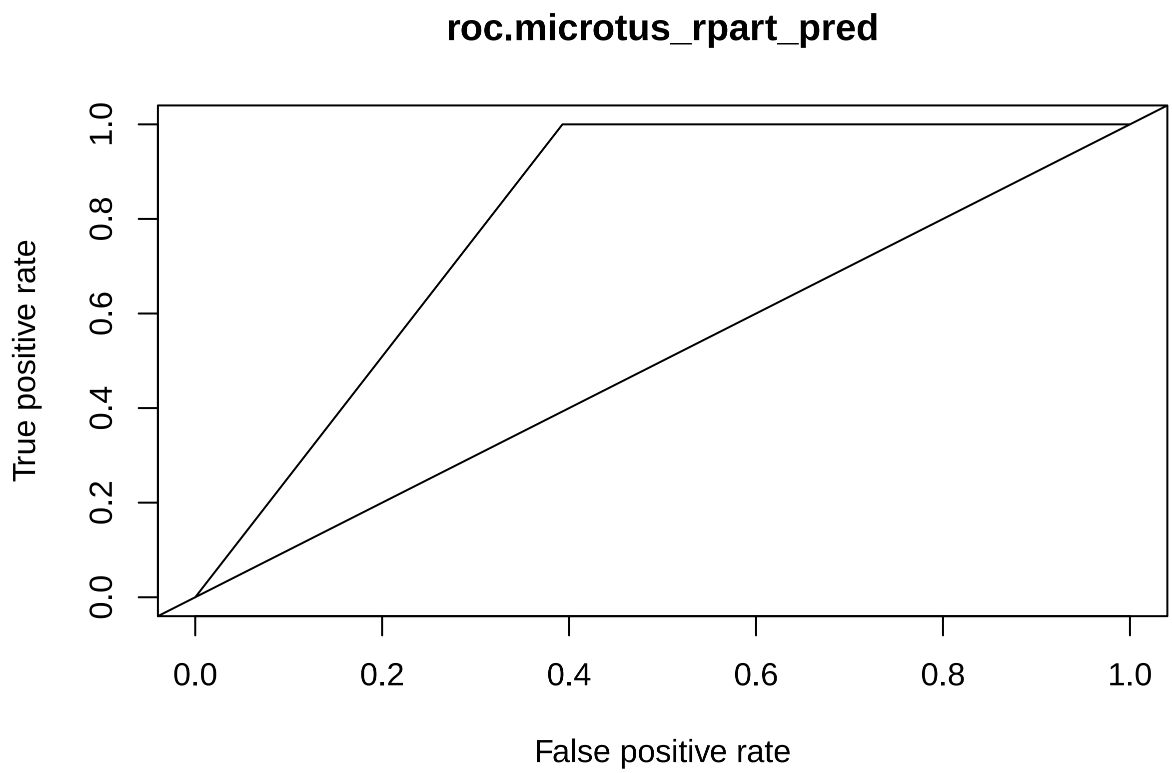
```
#plot data
par(mfrow=c(1,3))
plot(roc.perf_1, main = "glm_1")
abline(0,1)
plot(roc.perf_2, main = "glm_2")
abline(0,1)
plot(roc.perf_3, main = "glm_3")
abline(0,1)
```



#Tree Based Methods (recursive partitioning)

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was
## not in the result set. ROC will be used instead.
```



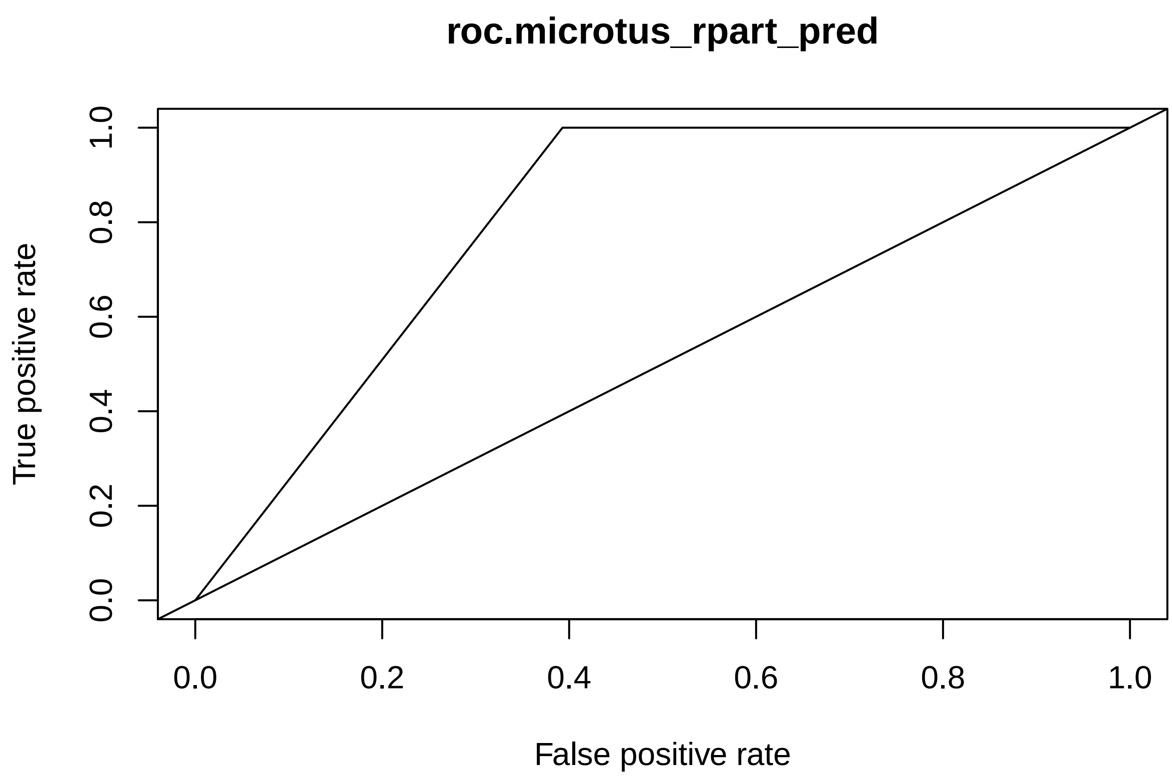



```
## [[1]]
## [1] 0.8035714
```

```
tree_ctrl <- trainControl(method = "cv", number = 10,
                          returnResamp = "all",
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary,
                          seeds = seed)
#treebag method used
microtus_tree_bag <- train(Group ~ .,
                           data = microtus_Train_no_flag,
                           method = "treebag",
                           trControl = tree_ctrl,
                           metric = "ROC",
                           nbagg = 10)

#microtus_tree_bag
#summary(microtus_tree_bag)

microtus_tree_bag_pred <- predict(microtus_tree_bag, newdata = microtus_Train_no_flag)
microtus_tree_bag_pred_cf <- confusionMatrix(microtus_tree_bag_pred, microtus_Train_no_flag$Group)
```

```
#Calculate Area Under the Curve for model

microtus_tree_bag_pred <- predict(microtus_tree_bag, newdata = microtus_Train_no_flag)
```

```r
#confusionMatrix(microtus_tree_bag_pred, microtus_Train_no_flag$Group)

#create predictions with probabilities
microtus_tree_bag_pred = predict(microtus_tree_bag, type = "prob")

#create prediction objects with multiplex column (note need to select just one column)
pred_tree_bag <- prediction(microtus_tree_bag_pred$multiplex, as.numeric(microtus_Train$group_flag))

roc.microtus_tree_bag_pred <- performance(pred_rpart, measure = "tpr", x.measure = "fpr")

#plot data
#par(mfrow=c(1,3))
plot(roc.microtus_tree_bag_pred, main = "roc.microtus_rpart_pred")
abline(0,1)
```



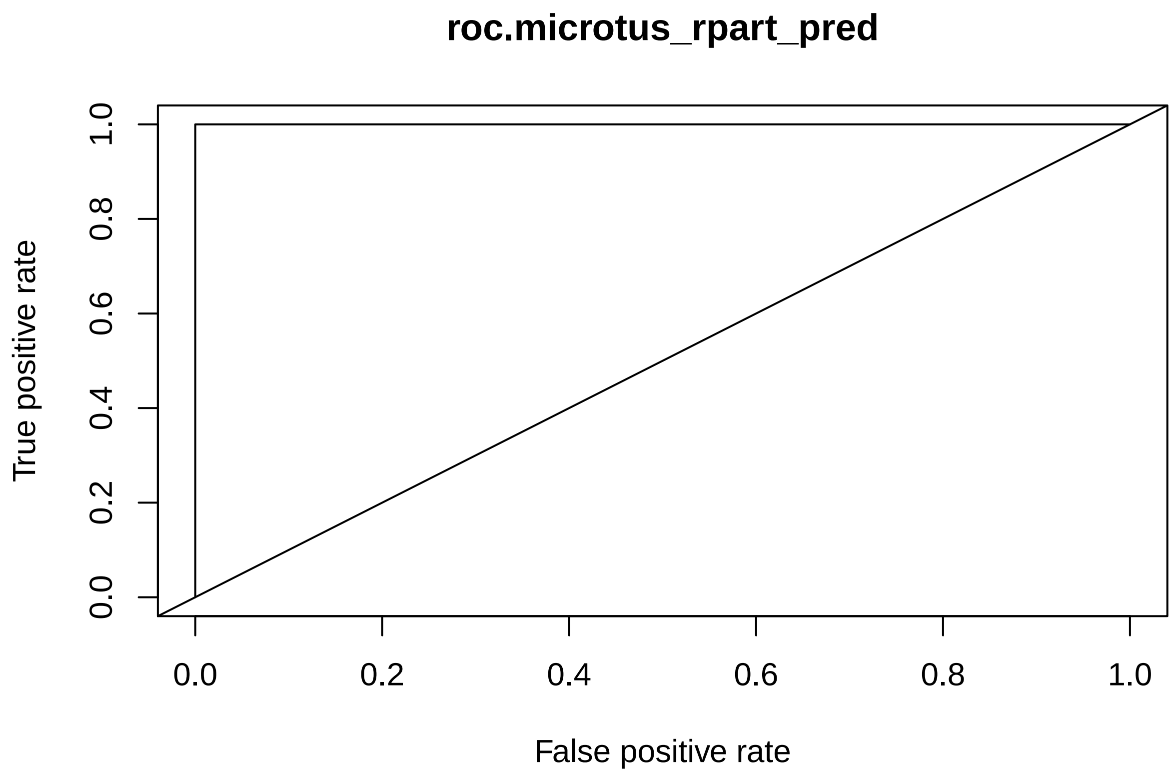

```r
roc.microtus_tree_bag_pred_AUC <- performance(pred_tree_bag, measure = "auc")
microtus_tree_bag_pred_AUC<- roc.microtus_tree_bag_pred_AUC@y.values
microtus_tree_bag_pred_AUC
```

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

#recursive feature elimination wrapper method to fit random forest

```r
# # define the control using a recursive feature elimination (backwards) selection function
#
# # define the control using a random forest selection function
# rfe_controller <- rfeControl(functions=rfFuncs, method="cv", number=10)
# # run the RFE algorithm
# x=microtus_Train_no_flag[,2:5]
# y=microtus_Train_no_flag[,1]
# rfe_results <- rfe(x=x, y, sizes=c(1:4), rfeControl=rfe_controller)
# # summarize the results
# print(rfe_results)
# # list the chosen features
# #predictors(rfe_results)
# # plot the results
# plot(rfe_results, type=c("g", "o"))
# rfe_results
#
# microtus_tree_ranFor_rfe <- rfe_results$fit
# microtus_tree_ranFor_rfe
#
# microtus_tree_ranForRFE_pred
```

```r
#Calculate Area Under the Curve for model

microtus_tree_ranFor_pred <- predict(microtus_tree_ranFor, newdata = microtus_Train_no_flag)

#confusionMatrix(microtus_tree_ranFor_pred, microtus_Train_no_flag$Group)
#create predictions with probabilities
microtus_tree_ranFor_pred = predict(microtus_tree_ranFor, type = "prob")

#create prediction objects with multiplex column (note need to select just one column)
pred_tree_ranFor <- prediction(microtus_tree_ranFor_pred$multiplex, as.numeric(microtus_Train$group_fla

roc.microtus_tree_ranFor_pred <- performance(pred_tree_ranFor, measure = "tpr", x.measure = "fpr")

#plot data
#par(mfrow=c(1,3))
plot(roc.microtus_tree_ranFor_pred, main = "roc.microtus_rpart_pred")
abline(0,1)
```




```
roc.microtus_tree_ranFor_pred_AUC <- performance(pred_tree_ranFor, measure = "auc")
microtus_tree_ranFor_pred_AUC<- roc.microtus_tree_ranFor_pred_AUC@y.values
microtus_tree_ranFor_pred_AUC
```

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

```
#Classification Metrics
"glm_1_pred_cf"
```

```
## [1] "glm_1_pred_cf"
```

```
glm_1_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.703704e-01   7.407407e-01   7.509878e-01   9.462570e-01   5.185185e-01
## AccuracyPValue  McnemarPValue
##   4.922101e-08   1.000000e+00
```

```
"glm_2_pred_cf"
```

```
## [1] "glm_2_pred_cf"
```

```r
glm_2_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.777778e-01   5.549451e-01   6.440009e-01   8.795642e-01   5.185185e-01
## AccuracyPValue  McnemarPValue
##   7.839236e-05   1.000000e+00
```

```r
"glm_3_pred_cf"
```

```
## [1] "glm_3_pred_cf"
```

```r
glm_3_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.888889e-01   7.780822e-01   7.736868e-01   9.581162e-01   5.185185e-01
## AccuracyPValue  McnemarPValue
##   7.515084e-09   6.830914e-01
```

```r
"microtus_rpart_pred_cf"
```

```
## [1] "microtus_rpart_pred_cf"
```

```r
microtus_rpart_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.962963e-01   5.981055e-01   6.646951e-01   8.936807e-01   5.185185e-01
## AccuracyPValue  McnemarPValue
##   2.262793e-05   2.568832e-03
```

```r
"microtus_tree_bag_pred_cf"
```

```
## [1] "microtus_tree_bag_pred_cf"
```

```r
microtus_tree_bag_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.814815e-01   9.628611e-01   9.010848e-01   9.995313e-01   5.185185e-01
## AccuracyPValue  McnemarPValue
##   2.023278e-14   1.000000e+00
```

```r
"microtus_tree_ranFor_pred_cf"
```

```
## [1] "microtus_tree_ranFor_pred_cf"
```

```r
microtus_tree_ranFor_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   1.000000e+00   1.000000e+00   9.339685e-01   1.000000e+00   5.185185e-01
## AccuracyPValue  McnemarPValue
##   3.956131e-16            NaN
```

fit best training models to test datasets

```
#glmnet
glm_3_test_pred <- predict(glm_3, newdata = microtus_Test, type = "raw")
glm_3_test_pred_cf <- confusionMatrix(data = glm_3_test_pred, reference = microtus_Test$Group)

glm_3_test_pred <- predict(glm_3, newdata = microtus_Test, type = "prob")

#create prediction objects with multiplex column (note need to select just one column)
pred_glm_3_test <- prediction(glm_3_test_pred$multiplex, as.numeric(microtus_Test$group_flag))

glm_3_test_pred_AUC <- performance(pred_glm_3_test, measure = "auc")
glm_3_test_pred_AUC@y.values
```

```
## [[1]]
## [1] 0.9084967
```

```
#bagged Tree
microtus_tree_bag_test_pred <- predict(microtus_tree_bag, newdata = microtus_Test)
microtus_tree_bag_test_pred_cf <- confusionMatrix(microtus_tree_bag_test_pred, microtus_Test$Group)

microtus_tree_bag_test_pred <- predict(microtus_tree_bag, newdata = microtus_Test, type = "prob")

#create prediction objects with multiplex column (note need to select just one column)
pred_tree_bag_test <- prediction(microtus_tree_bag_test_pred$multiplex, as.numeric(microtus_Test$group_

tree_bag_test_AUC <- performance(pred_glm_3_test, measure = "auc")
tree_bag_test_AUC@y.values
```

```
## [[1]]
## [1] 0.9084967
```

```
#Random Forest
microtus_tree_ranFor_test_pred <- predict(microtus_tree_ranFor, newdata = microtus_Test)
microtus_tree_ranFor_test_pred_cf <- confusionMatrix(microtus_tree_ranFor_test_pred, microtus_Test$Grou

# microtus_tree_ranFor_test_pred <- predict(microtus_tree_ranFor, newdata = microtus_Test, type = "prob
# #create prediction objects with multiplex column (note need to select just one column)
# pred_tree_ranFor_test <- prediction(microtus_tree_ranFor_test_pred$multiplex, as.numeric(microtus_Tes
#
# microtus_tree_ranFor_test_pred_cf <- confusionMatrix(pred_tree_ranFor_test, microtus_Test$Group)
```

```
glm_3_test_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   0.8285714286   0.6568627451   0.6635017000   0.9343781988   0.5142857143
## AccuracyPValue  McnemarPValue
##   0.0001126156   1.0000000000
```

```r
microtus_tree_bag_test_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   0.8285714286   0.6557377049   0.6635017000   0.9343781988   0.5142857143
## AccuracyPValue  McnemarPValue
##   0.0001126156   0.6830913983
```

```r
microtus_tree_ranFor_test_pred_cf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.571429e-01   7.135843e-01   6.974286e-01   9.519392e-01   5.142857e-01
## AccuracyPValue  McnemarPValue
##   2.275361e-05   1.000000e+00
```

##use GLMNET to predict all classes based on Kappa Score

```r
# microtus <- microtus %>% mutate(
#   final_pred_flag = if_else(microtus$Group == "unknown",1,0))

microtus <- microtus %>% mutate(
  final_pred = if_else(microtus$Group == "unknown",
                       predict(glm_3, newdata = microtus),
                       microtus$Group))
summary(microtus)
```

```
##           Group         M1Left         M2Left         M3Left
##  multiplex   : 43   Min.   :1534   Min.   :1355   Min.   :1361
##  subterraneus: 46   1st Qu.:1783   1st Qu.:1503   1st Qu.:1595
##  unknown     :199   Median :1923   Median :1570   Median :1724
##                     Mean   :1935   Mean   :1589   Mean   :1727
##                     3rd Qu.:2074   3rd Qu.:1660   3rd Qu.:1856
##                     Max.   :2479   Max.   :1880   Max.   :2187
##     Foramen         Pbone          Length         Height
##  Min.   :3155   Min.   :3928   Min.   :1908   Min.   :700.0
##  1st Qu.:3751   1st Qu.:4815   1st Qu.:2227   1st Qu.:759.2
##  Median :3932   Median :5079   Median :2312   Median :789.0
##  Mean   :3913   Mean   :5082   Mean   :2309   Mean   :790.8
##  3rd Qu.:4080   3rd Qu.:5328   3rd Qu.:2388   3rd Qu.:817.8
##  Max.   :4662   Max.   :6104   Max.   :2605   Max.   :912.0
##     Rostrum             final_pred
##  Min.   :375.0   multiplex   :151
##  1st Qu.:425.0   subterraneus:137
##  Median :450.0
##  Mean   :451.2
##  3rd Qu.:475.0
##  Max.   :545.0
```