Project

Isaac Attuah

Saturday, February 19th 2022

Preparing Notebook

```
# Clear workspace
rm(list = ls())

# Set Working Directory
# setwd("./assignment_one")
# getwd()

# Load Libraries
library(ISLR2)
library(ggcorrplot)
```

```
## Loading required package: ggplot2
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
library(boot)
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(randomForest)
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Load Dataset
onset_data <- read.csv("onset.csv")
extra_data <- read.csv("armed.csv")

# Merge Datasets
war_data = merge(onset_data, extra_data, by = c("year", "gwno_a"))

# shuffle the dataframe by rows
# war_data <- war_data[sample(1:nrow(war_data)), ]

# Turn multiple columns to factor
cols <- c("gwno_a", "newconf", "onset1", "onset2", "onset3", "onset5", "onset10", "onset20", "intensity_
war_data[cols] <- lapply(war_data[cols], as.factor)

# Clean and Attach Data
# war_data <- war_data[!names(war_data) %in% c("year_prev")]

war_data <- na.omit(war_data)

#attach(war_data)
# wd_merge <- merge(x=war_data, y=extra, by="gwno_a")
# war_data <- wd_merge
# View Data
head(war_data, 10)
```

```
##    year gwno_a abc           name newconf onset1 onset2 onset3 onset5 onset10
## 1  1946    145 BOL        Bolivia       1      1      1      1      1       1
## 2  1946    200 UKG United Kingdom       1      1      1      1      1       1
## 3  1946    210 NTH    Netherlands       1      1      1      1      1       1
## 4  1946    220 FRN         France       1      1      1      1      1       1
## 5  1946    220 FRN         France       1      1      1      1      1       1
## 6  1946    220 FRN         France       1      1      1      1      1       1
## 7  1946    220 FRN         France       1      1      1      1      1       1
## 8  1946    220 FRN         France       1      1      1      1      1       1
## 9  1946    220 FRN         France       1      1      1      1      1       1
## 10 1946    220 FRN         France       1      1      1      1      1       1
##    onset20 year_prev duration incompatibility intensity_level
## 1        1      1815        7               1               1
## 2        1      1815        7               0               0
## 3        1      1815        7               0               0
## 4        1      1815        7               0               0
## 5        1      1815        7               0               0
## 6        1      1815        7               0               0
## 7        1      1815        7               0               1
## 8        1      1815        7               0               0
## 9        1      1815        7               0               0
## 10       1      1815        7               0               0
##    cumulative_intensity ep_end
## 1                     1      1
## 2                     0      1
## 3                     0      0
## 4                     0      1
## 5                     0      0
## 6                     0      0
## 7                     1      0
## 8                     0      1
## 9                     0      0
## 10                    0      0
```

Dataset Information

General Information

```
message('Dimensions of Dataset')
```

```
## Dimensions of Dataset
```

```
dim(war_data)
```

```
## [1] 1040   17
```

```
message("Number of Rows ", nrow(war_data))
```

```
## Number of Rows 1040
```

```r
message("Number of Columns ", ncol(war_data))
```

```
## Number of Columns 17
```

Data Summary

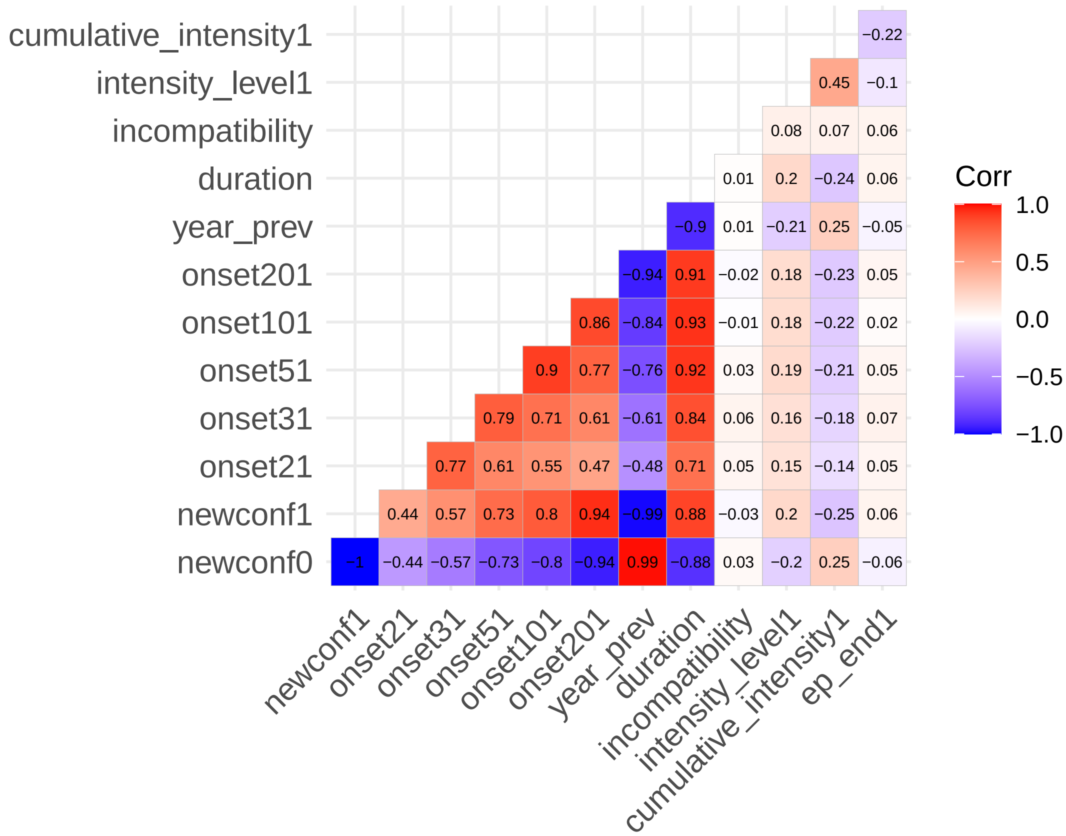
```r
summary(war_data)
```

```
##       year          gwno_a       abc                 name              newconf
##  Min.   :1946   750    :177   Length:1040        Length:1040        0:529
##  1st Qu.:1972   775    :165   Class :character   Class :character   1:511
##  Median :1991   530    : 77   Mode  :character   Mode  :character
##  Mean   :1987   220    : 46
##  3rd Qu.:2001   630    : 34
##  Max.   :2017   365    : 30
##                 (Other):511
##  onset1    onset2   onset3   onset5   onset10 onset20   year_prev       duration
##  0:   0    0:174    0:262    0:367    0:417    0:497    Min.   :1815   Min.   :1.00
##  1:1040    1:866    1:778    1:673    1:623    1:543    1st Qu.:1815   1st Qu.:2.00
##                                                          Median :1950   Median :6.00
##                                                          Mean   :1903   Mean   :4.84
##                                                          3rd Qu.:1992   3rd Qu.:7.00
##                                                          Max.   :2015   Max.   :7.00
##
##  incompatibility  intensity_level cumulative_intensity ep_end
##  Min.   :0.0000   0:838           0:475                0:669
##  1st Qu.:0.0000   1:202           1:565                1:371
##  Median :0.0000
##  Mean   :0.3404
##  3rd Qu.:1.0000
##  Max.   :3.0000
##
```

Correlation

Since the values are categorical, we will resort to using

```r
war_data_cor <- war_data[ , !names(war_data) %in% c("abc","name", "year", "gwno_a", "onset1")]

library(ggcorrplot)
model.matrix(~0+., data=war_data_cor) %>%
  cor(use="pairwise.complete.obs") %>%
  ggcorrplot(show.diag = F, type="lower", lab=TRUE, lab_size=2)
```



Dividing Into Training and Testing

```
# Divide data into training and testing (# 3)
# Examples for training = (0.80 * 683) = 546 entries
# Examples for testing test = (0.20 * 683) = 137 entries

set.seed(222)
sample_size = round(nrow(war_data)*.80) # setting sample size is 80%
index <- sample(seq_len(nrow(war_data)), size = sample_size)

train_better <- war_data[index, ]
test_better <- war_data[-index, ]

message("Number of Training Examples: ", nrow(train_better))
```

```
## Number of Training Examples: 832
```

```
message("Number of Testing Examples: ", nrow(test_better))
```

```
## Number of Testing Examples: 208
```

```r
# train_better
# test_better

train_valid <- train_better[ , !names(train_better) %in% c("abc","name", "year", "gwno_a", "onset1", "ye

test_valid <- test_better[ , !names(test_better) %in% c("abc","name", "year", "gwno_a", "onset1", "year_

war_data_valid <- war_data[ , !names(test_better) %in% c("abc","name", "year", "gwno_a", "onset1")]
```

Phase 1: Predicting War Outcome In 20 Years

Model Testing

Logistic Regression Model

```r
# Making model with all input variables

#There is not enough variation in onset1 so we will not include in the regression
glm.fits = glm(onset20 ~ newconf+onset2+onset3+onset5+onset10+duration+year_prev+duration+incompatibili
               data = train_better, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```r
# glm.fits = glm(duration ~ onset2+onset3+onset5+onset10+onset20, data = train_better)

summary(glm.fits)
```

```
##
## Call:
## glm(formula = onset20 ~ newconf + onset2 + onset3 + onset5 +
##     onset10 + duration + year_prev + duration + incompatibility +
##     intensity_level + cumulative_intensity + ep_end, family = binomial,
##     data = train_better)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.409e-06  -2.409e-06   2.409e-06   2.409e-06   2.409e-06
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -7.970e+01  2.427e+06   0.000    1.000
## newconf1             -5.313e+01  2.257e+05   0.000    1.000
## onset21              -5.313e+01  9.401e+04  -0.001    1.000
## onset31              -5.313e+01  9.742e+04  -0.001    1.000
## onset51              -5.313e+01  1.040e+05  -0.001    1.000
## onset101             -5.313e+01  1.221e+05   0.000    1.000
## duration              5.313e+01  7.757e+04   0.001    0.999
## year_prev             2.635e-12  1.213e+03   0.000    1.000
## incompatibility       1.213e-10  2.415e+04   0.000    1.000
```

```
## intensity_level1       3.972e-10  3.893e+04   0.000    1.000
## cumulative_intensity1  1.121e-10  3.100e+04   0.000    1.000
## ep_end1                3.333e-12  2.654e+04   0.000    1.000
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.1515e+03  on 831  degrees of freedom
## Residual deviance: 4.8269e-09  on 820  degrees of freedom
## AIC: 24
##
## Number of Fisher Scoring iterations: 25
```

```r
# Make predictions based on model
glm.probs = predict(glm.fits,test_better, type="response")

# Initialize vector with 109 elements
glm.pred = rep(0, nrow(test_better))
# Assign 1 to probabilities > 0.5
glm.pred[glm.probs >.5]=1

message('0 for no conflict, 1 for new conflict')
```

```
## 0 for no conflict, 1 for new conflict
```

```r
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```r
# Confusion Matrix
table(glm.pred,test_better$onset20)
```

```
##
## glm.pred   0   1
##        0 101   0
##        1   0 107
```

```r
# Test Error
message('Test Error Rate')
```

```
## Test Error Rate
```

```r
mean(glm.pred!=test_better$onset20)
```

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

LDA Model

```r
# Making model with all input variables
lda.fit=lda(onset20 ~ newconf+onset2+onset3+onset5+onset10+duration+year_prev+duration+incompatibility+i
               data = train_better)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
lda.fit
```

```
## Call:
## lda(onset20 ~ newconf + onset2 + onset3 + onset5 + onset10 +
##     duration + year_prev + duration + incompatibility + intensity_level +
##     cumulative_intensity + ep_end, data = train_better)
##
## Prior probabilities of groups:
##         0         1
## 0.4759615 0.5240385
##
## Group means:
##    newconf1   onset21  onset31  onset51  onset101 duration year_prev
## 0 0.0000000 0.6464646 0.479798 0.270202 0.1717172 2.568182  1988.338
## 1 0.9288991 1.0000000 1.000000 1.000000 1.0000000 6.928899  1825.901
##   incompatibility intensity_level1 cumulative_intensity1   ep_end1
## 0       0.3535354        0.1111111             0.6666667 0.3409091
## 1       0.3394495        0.2454128             0.4197248 0.3784404
##
## Coefficients of linear discriminants:
##                                LD1
## newconf1               4.3704336087
## onset21               -0.1416472147
## onset31               -0.1588930788
## onset51               -0.1567057302
## onset101               1.8061343812
## duration               0.2094459456
## year_prev              0.0001166536
## incompatibility        0.0377394397
## intensity_level1      -0.1753330702
## cumulative_intensity1  0.1759232544
## ep_end1               -0.0121873708
```

```r
summary(lda.fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   22     -none- numeric
## scaling 11     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  8     -none- list
```

```r
lda.pred <- predict(lda.fit , test_better)

message('2 for benign, 4 for malignant')
```

```
## 2 for benign, 4 for malignant
```

```r
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```r
# Confusion Matrix
lda.class <- lda.pred$class
table(lda.class, test_better$onset20)
```

```
##
## lda.class   0   1
##         0 101   1
##         1   0 106
```

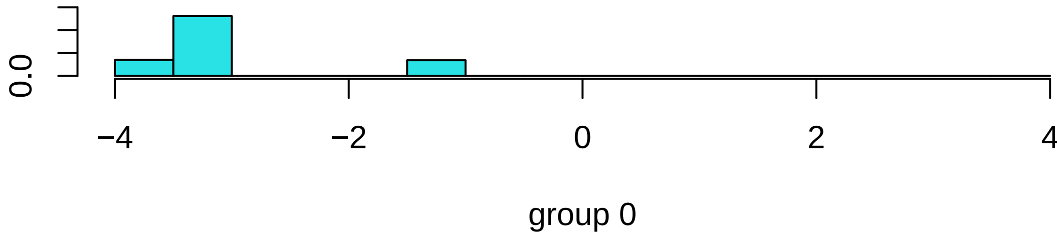
```r
message('Test Error Rate')
```

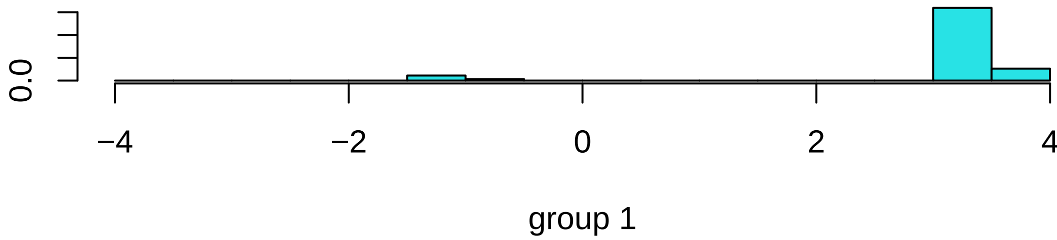
```
## Test Error Rate
```

```r
# Test Error
mean(lda.class != test_better$onset20)
```

```
## [1] 0.004807692
```

```r


plot(lda.fit)
```





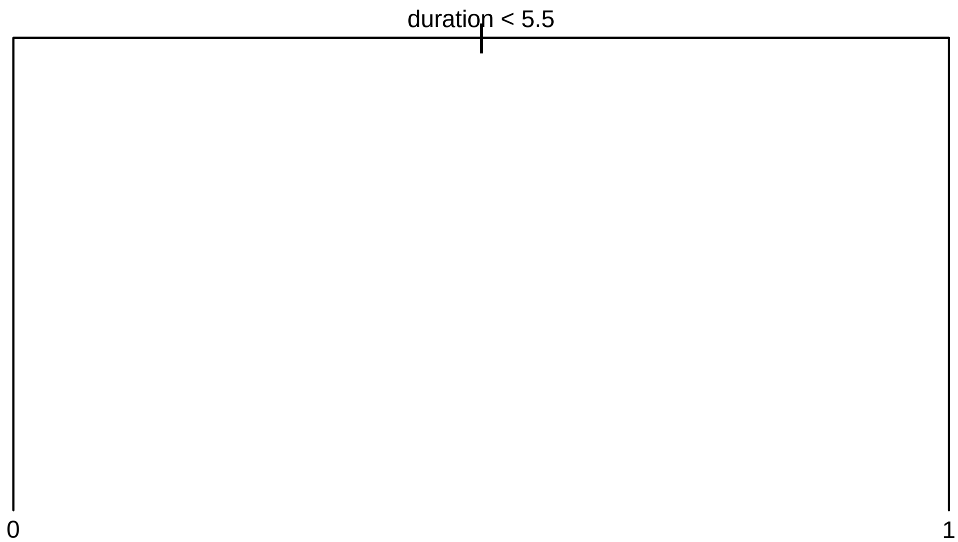
Linear Discriminants

Decision Trees (Generic)

```
tree.onset20=tree(onset20 ~ newconf+onset2+onset3+onset5+onset10+duration+year_prev+duration+incompatibi
summary(tree.onset20)
```

```
##
## Classification tree:
## tree(formula = onset20 ~ newconf + onset2 + onset3 + onset5 +
##     onset10 + duration + year_prev + duration + incompatibility +
##     intensity_level + cumulative_intensity + ep_end, data = war_data_valid)
## Variables actually used in tree construction:
## [1] "duration"
## Number of terminal nodes:  2
## Residual mean deviance:  0 = 0 / 1038
## Misclassification error rate: 0 = 0 / 1040
```

```
plot(tree.onset20)
text(tree.onset20, pretty = 0, cex=0.75)
```



Decision Trees (With Training & Testing)

```
# Train using training set
tree.onset20=tree(onset20 ~ newconf+onset2+onset3+onset5+onset10+duration+incompatibility+intensity_leve

# Test on test set using predict()
# type="class" to return the class prediction
tree.pred=predict(tree.onset20,test_valid,type="class")

# Confusion matrix
conf.matrix <- table(tree.pred,test_valid$onset20)
conf.matrix
```

```
##
## tree.pred   0   1
##         0 101   0
##         1   0 107
```

```
# Accuracy on test set
(conf.matrix[1,1] + conf.matrix[2,2])/(conf.matrix[1,1] + conf.matrix[2,2] + conf.matrix[1,2]+ conf.mat
```

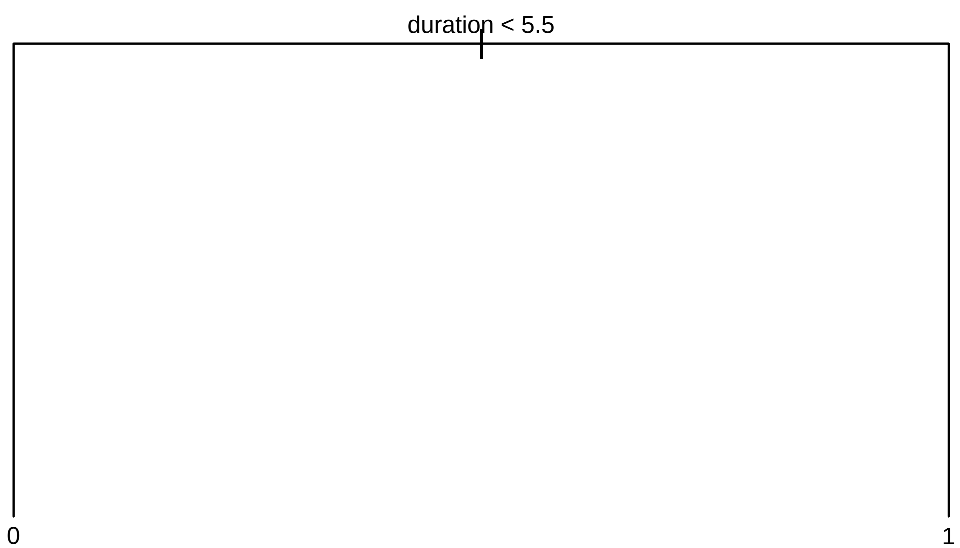
```
## [1] 1
```

Regression Trees

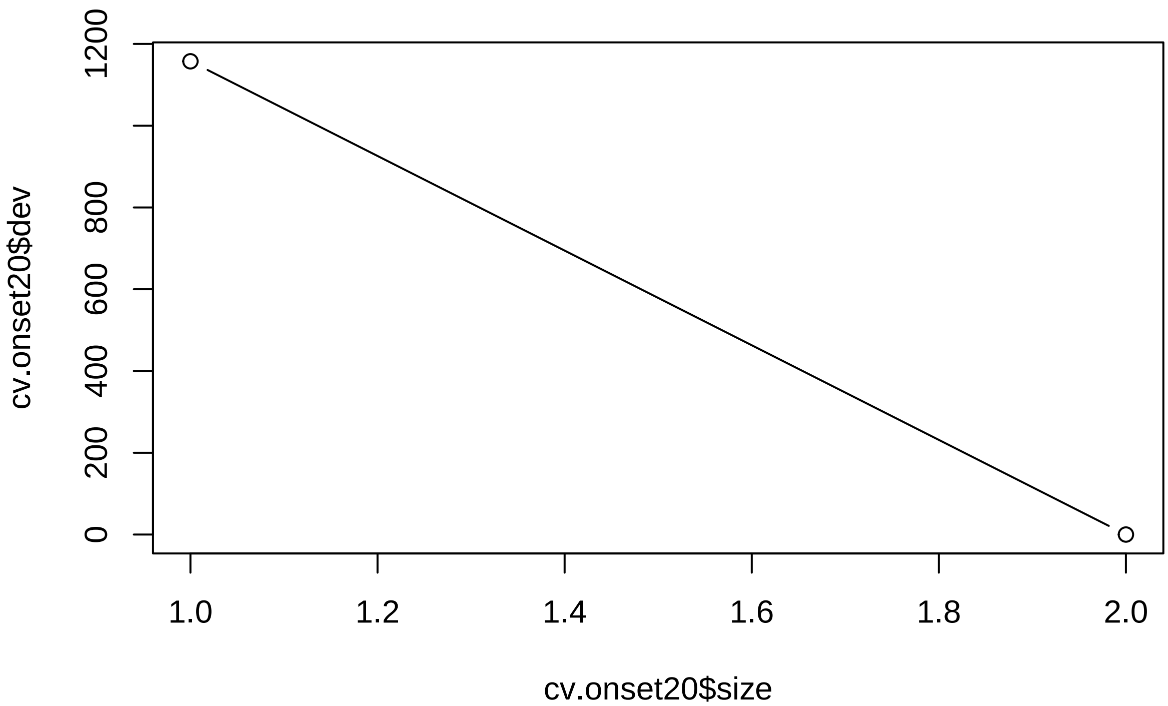
```r
set.seed(1)

tree.onset20=tree(onset20 ~ newconf+onset2+onset3+onset5+onset10+duration+incompatibility+intensity_leve
# Only a few of the variables were used in constructing the tree
# lstat: percentage of individuals with lower socioeconomic status
summary(tree.onset20)
```

```
##
## Classification tree:
## tree(formula = onset20 ~ newconf + onset2 + onset3 + onset5 +
##     onset10 + duration + incompatibility + intensity_level +
##     cumulative_intensity + ep_end, data = train_valid)
## Variables actually used in tree construction:
## [1] "duration"
## Number of terminal nodes:  2
## Residual mean deviance:  0 = 0 / 830
## Misclassification error rate: 0 = 0 / 832
```

```r
# Plot the tree
# Lower values of lstat correspond to more expensive houses
plot(tree.onset20)
text(tree.onset20,pretty=0,cex=0.75)
```



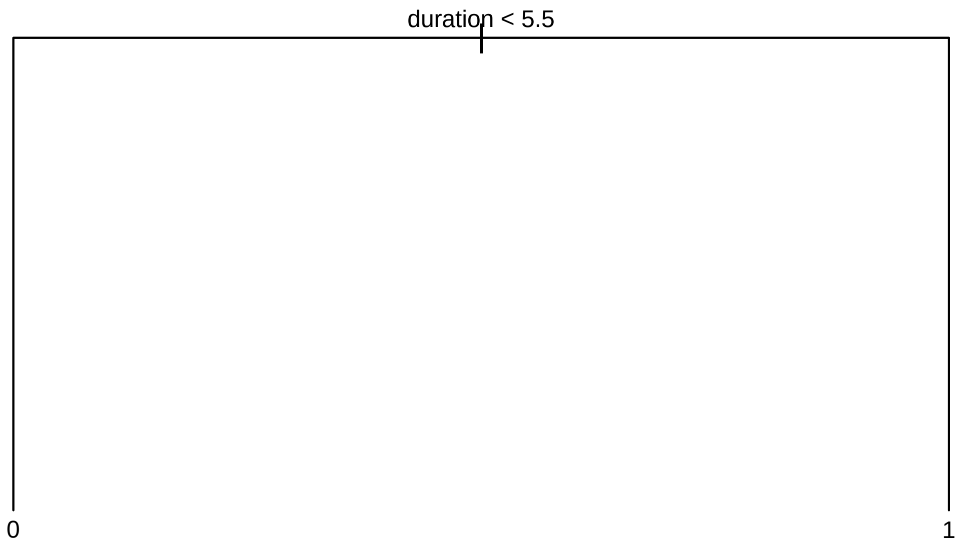
```r
# cv.tree() to determine whether pruning improves performance
cv.onset20=cv.tree(tree.onset20)
# It doesn't seem to be the case
plot(cv.onset20$size,cv.onset20$dev,type="b")
```




```r
# prune.tree(): function to prune to be used in case we wanted to prune the tree
prune.onset20=prune.tree(tree.onset20,best=5)
```

```
## Warning in prune.tree(tree.onset20, best = 5): best is bigger than tree size
```

```r
plot(prune.onset20)
text(prune.onset20,pretty=0,cex=0.75)
```




```
# Predicting based on CV results (i.e., use the unpruned tree)
yhat=predict(tree.onset20,newdata=test_valid)

# plot(yhat,test_valid$onset20)
# abline(0,1)
# Test error
mse=mean((yhat-test_valid$onset20)^2)
```

```
## Warning in Ops.factor(yhat, test_valid$onset20): '-' not meaningful for factors
```

```
mse
```

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

```
# This model leads to test predictions that are within around $5-6K of the true
# median home value for the suburb
sqrt(mse)
```

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

Random Forests

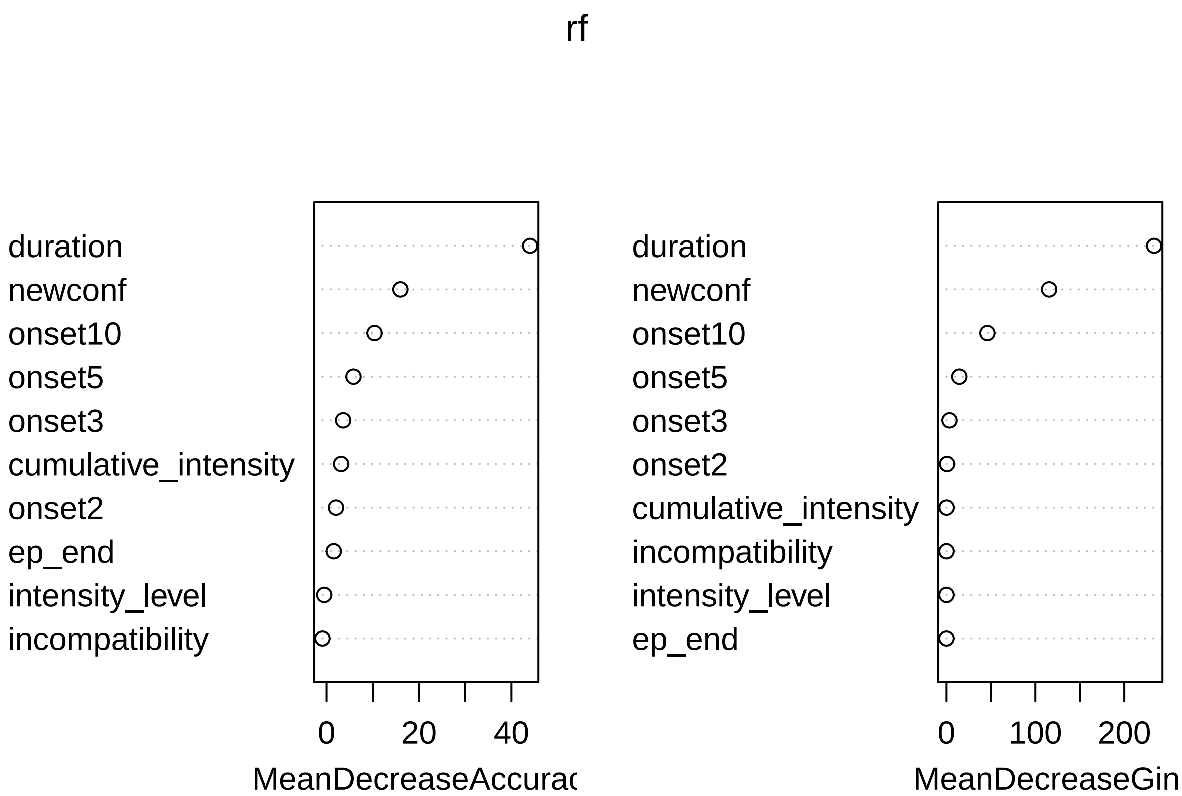
```r
# By default randomForest() uses m=p/3 for regression and m=sqrt(p) for classification
# Let's try m=6
set.seed(1)
rf=randomForest(onset20 ~ newconf+onset2+onset3+onset5+onset10+duration+incompatibility+intensity_level+
yhat.rf = predict(tree.onset20,newdata=test_valid)

mean((yhat.rf-as.integer(test_valid$onset20))^2)
```

```
## [1] 1.528846
```

```r
# importance(): view the importance of each variable
# %IncMSE: mean decrease of accuracy in predictions on the OOB samples when a
# given variable is excluded from the model
# IncNodeImpurity: total decrease in node impurity that results from splits over
# that variable, averaged over all trees (RSS in regr. vs. deviance in class.)
importance(rf)
```

```
##                              0          1 MeanDecreaseAccuracy
## newconf             15.6892516  4.7108616           15.9759772
## onset2               0.0000000  2.0489485            2.0521178
## onset3               0.0000000  3.5784432            3.5878985
## onset5              -1.4169759  5.8180605            5.8190757
## onset10              1.5221103 10.3675345           10.3795729
## duration            39.6159791 32.5857121           44.0281649
## incompatibility      0.8375265 -1.3084633           -0.8899624
## intensity_level      0.3354407 -0.7267548           -0.4953894
## cumulative_intensity 1.8979866  2.6486145            3.1606838
## ep_end               1.7834743  0.5808985            1.5507581
##                     MeanDecreaseGini
## newconf                 115.44478476
## onset2                    0.73073065
## onset3                    3.43096536
## onset5                   14.44642774
## onset10                  46.09759296
## duration                233.43477076
## incompatibility           0.07150790
## intensity_level           0.04697760
## cumulative_intensity      0.16936079
## ep_end                    0.04261412
```

```r
# varImpPlot(): Variance importance plot
varImpPlot(rf)
```



Other Models

- Penalized Logistic Regression -`plr`
- Conditional Inference Random Forest -`cforest`
- Random Forest - `rf`
- Bayesian Generalized Linear Model -`bayesglm`
- Boosted Generalized Additive Model **-** `gamboost`
- Support Vector Machines with Linear Kernel - `svmLinear`

```
library(caret)
```

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

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
#specify the cross-validation method
ctrl <- trainControl(method = "cv")

#fit a regression model and use LOOCV to evaluate performance
model <- train(onset20~newconf+onset2+onset3+onset5+onset10, data = train_better, method = "knn", trCont

#view summary of LOOCV
print(model)
```

```
## k-Nearest Neighbors
##
## 832 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 749, 748, 748, 749, 749, 748, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9627636  0.9256512
##   7  0.9627636  0.9256512
##   9  0.9627636  0.9256512
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
predictions <- predict(model, test_better, type="raw")

message('0 for no conflict, 1 for new conflict')
```

```
## 0 for no conflict, 1 for new conflict
```

```
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```
# Confusion Matrix
table(predictions,test_better$onset20)
```

```
##
## predictions   0   1
##           0 101   1
##           1   0 106
```

```r
# Test Error
message('Test Error Rate')
```

```
## Test Error Rate
```

```r
mean(predictions!=test_better$onset20)
```

```
## [1] 0.004807692
```

```r
confusionMatrix(data = predict(model, test_better), test_better$onset20)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 101   1
##          1   0 106
##
##                Accuracy : 0.9952
##                  95% CI : (0.9735, 0.9999)
##     No Information Rate : 0.5144
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9904
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9907
##          Pos Pred Value : 0.9902
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4856
##          Detection Rate : 0.4856
##    Detection Prevalence : 0.4904
##       Balanced Accuracy : 0.9953
##
##        'Positive' Class : 0
##
```

Phase 2: Predicting the End Of War

Logistic Regression Model

```r
# Making model with all input variables

#There is not enough variation in onset1 so we will not include in the regression
glm.fits = glm(ep_end ~ newconf+onset2+onset3+onset5+onset10+duration+year_prev+duration+incompatibility
               data = train_better, family = binomial)
```

```r
# glm.fits = glm(duration ~ onset2+onset3+onset5+onset10+onset20, data = train_better)

summary(glm.fits)
```

```
##
## Call:
## glm(formula = ep_end ~ newconf + onset2 + onset3 + onset5 + onset10 +
##     duration + year_prev + duration + incompatibility + intensity_level +
##     cumulative_intensity + onset20, family = binomial, data = train_better)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4624  -0.9715  -0.7358   1.2683   1.8536
##
## Coefficients: (1 not defined because of singularities)
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -4.276602  15.002399  -0.285   0.7756
## newconf1               0.712251   1.421590   0.501   0.6164
## onset21                0.298784   0.599713   0.498   0.6183
## onset31                0.315113   0.613570   0.514   0.6076
## onset51                0.247213   0.644233   0.384   0.7012
## onset101              -0.731827   0.766924  -0.954   0.3400
## duration              -0.055770   0.502538  -0.111   0.9116
## year_prev              0.001991   0.007498   0.266   0.7905
## incompatibility        0.310721   0.143585   2.164   0.0305 *
## intensity_level1       0.013462   0.246219   0.055   0.9564
## cumulative_intensity1 -0.957326   0.186151  -5.143 2.71e-07 ***
## onset201                     NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1087.8  on 831  degrees of freedom
## Residual deviance: 1036.0  on 821  degrees of freedom
## AIC: 1058
##
## Number of Fisher Scoring iterations: 4
```

```r
# Make predictions based on model
glm.probs = predict(glm.fits,test_better, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
# Initialize vector with 109 elements
glm.pred = rep(0, nrow(test_better))
# Assign 1 to probabilities > 0.5
glm.pred[glm.probs >.5]=1

message('0 for no conflict, 1 for new conflict')
```

```
## 0 for no conflict, 1 for new conflict
```

```r
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```r
# Confusion Matrix
table(glm.pred,test_better$ep_end)
```

```
##
## glm.pred   0   1
##        0 127  55
##        1  10  16
```

```r
# Test Error
message('Test Error Rate')
```

```
## Test Error Rate
```

```r
mean(glm.pred!=test_better$ep_end)
```

```
## [1] 0.3125
```

LDA Model

```r
# Making model with all input variables
lda.fit=lda(ep_end ~ newconf+onset2+onset3+onset5+onset10+duration+year_prev+duration+incompatibility+in
               data = train_better)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
lda.fit
```

```
## Call:
## lda(ep_end ~ newconf + onset2 + onset3 + onset5 + onset10 + duration +
##     year_prev + duration + incompatibility + intensity_level +
##     cumulative_intensity + onset20, data = train_better)
##
## Prior probabilities of groups:
##         0         1
## 0.6394231 0.3605769
##
## Group means:
##    newconf1   onset21   onset31   onset51  onset101 duration year_prev
## 0 0.4661654 0.8120301 0.7293233 0.6390977 0.5996241 4.755639  1906.714
## 1 0.5233333 0.8666667 0.7933333 0.6766667 0.6166667 5.026667  1897.010
##   incompatibility intensity_level1 cumulative_intensity1  onset201
## 0       0.3214286        0.2067669             0.6184211 0.5093985
## 1       0.3900000        0.1366667             0.3933333 0.5500000
##
```

```
## Coefficients of linear discriminants:
##                              LD1
## newconf1              1.143387203
## onset21               0.408209147
## onset31               0.492529631
## onset51               0.361322729
## onset101             -1.517663352
## duration              0.021684369
## year_prev             0.003584622
## incompatibility       0.595685678
## intensity_level1      0.010460217
## cumulative_intensity1 -1.854945518
## onset201             -0.107028235
```

```
summary(lda.fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   22     -none- numeric
## scaling 11     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  8     -none- list
```

```
lda.pred <- predict(lda.fit , test_better)

message('2 for benign, 4 for malignant')
```

```
## 2 for benign, 4 for malignant
```

```
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```
# Confusion Matrix
lda.class <- lda.pred$class
table(lda.class, test_better$onset20)
```

```
##
## lda.class  0  1
##         0 88 93
##         1 13 14
```

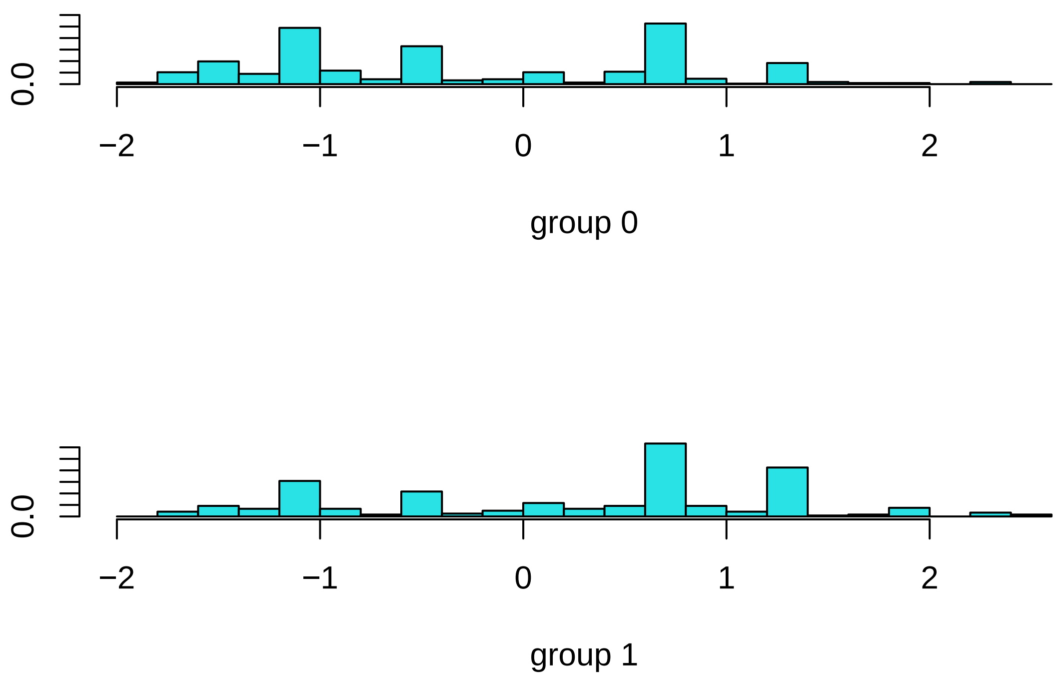
```
message('Test Error Rate')
```

```
## Test Error Rate
```

```r
# Test Error
mean(lda.class != test_better$onset20)
```

```
## [1] 0.5096154
```

```r
plot(lda.fit)
```



Linear Discriminants

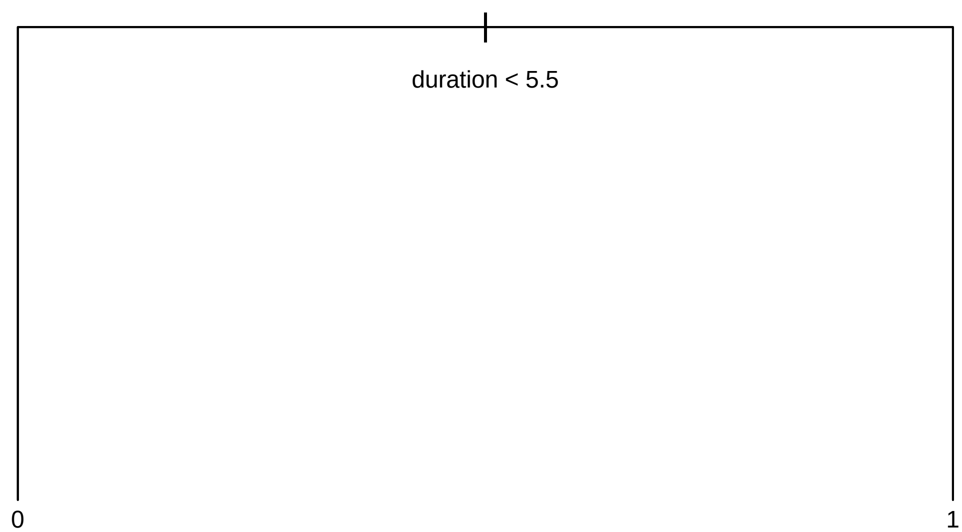
Decision Trees (Generic)

```r
tree.ep_end=tree(ep_end ~ newconf+onset2+onset3+onset5+onset10+duration+year_prev+duration+incompatibili
summary(tree.ep_end)
```

```
##
## Classification tree:
## tree(formula = ep_end ~ newconf + onset2 + onset3 + onset5 +
##     onset10 + duration + year_prev + duration + incompatibility +
##     intensity_level + cumulative_intensity + ep_end + onset20,
##     data = war_data_valid)
## Variables actually used in tree construction:
```

```
## [1] "ep_end"
## Number of terminal nodes:  2
## Residual mean deviance:  0 = 0 / 1038
## Misclassification error rate: 0 = 0 / 1040
```

```
plot(tree.ep_end)
text(tree.onset20, pretty = 0, cex=0.75)
```



Decision Trees (With Training & Testing)

```
# Train using training set
tree.ep_end=tree(ep_end ~ newconf+onset2+onset3+onset5+onset10+duration+incompatibility+intensity_level

# Test on test set using predict()
# type="class" to return the class prediction
tree.ep_end=predict(tree.onset20,test_valid,type="class")

# Confusion matrix
conf.matrix <- table(tree.ep_end,test_valid$ep_end)
conf.matrix
```

```
##
## tree.ep_end  0  1
```

```
##           0 70 31
##           1 67 40
```

```r
# Accuracy on test set
(conf.matrix[1,1] + conf.matrix[2,2])/(conf.matrix[1,1] + conf.matrix[2,2] + conf.matrix[1,2]+ conf.mat
```

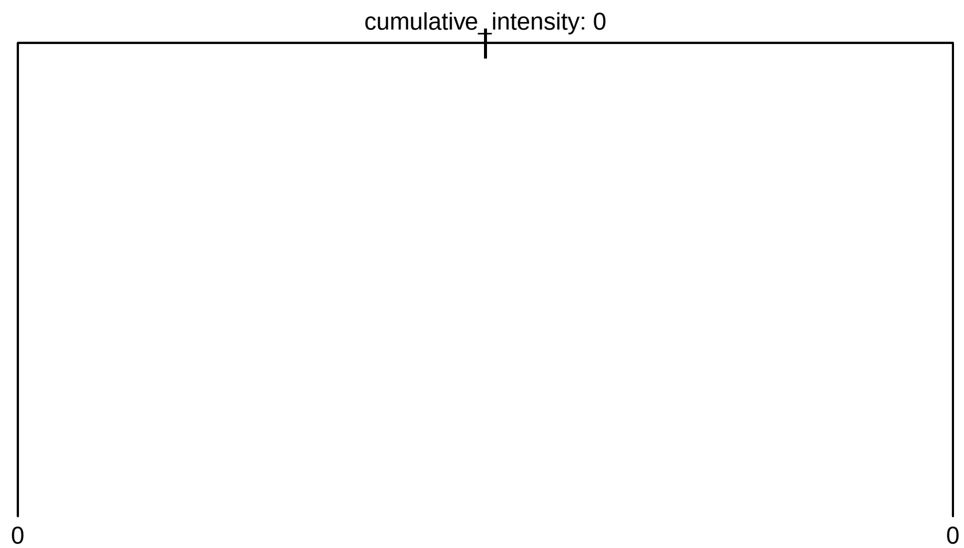
```
## [1] 0.5288462
```

Regression Trees

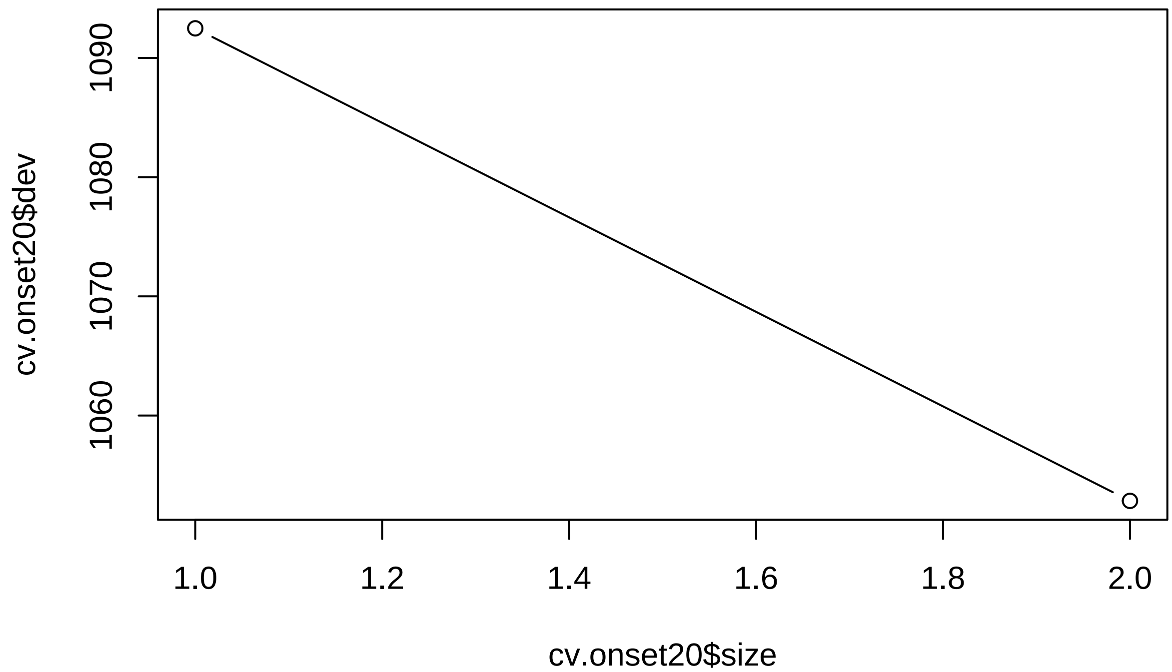
```r
set.seed(1)

tree.onset20=tree(ep_end  ~ newconf+onset2+onset3+onset5+onset10+duration+incompatibility+intensity_leve
# Only a few of the variables were used in constructing the tree
# lstat: percentage of individuals with lower socioeconomic status
summary(tree.onset20)
```

```
##
## Classification tree:
## tree(formula = ep_end ~ newconf + onset2 + onset3 + onset5 +
##     onset10 + duration + incompatibility + intensity_level +
##     cumulative_intensity + onset20, data = train_valid)
## Variables actually used in tree construction:
## [1] "cumulative_intensity"
## Number of terminal nodes:  2
## Residual mean deviance:  1.263 = 1049 / 830
## Misclassification error rate: 0.3606 = 300 / 832
```

```r
# Plot the tree
# Lower values of lstat correspond to more expensive houses
plot(tree.onset20)
text(tree.onset20,pretty=0,cex=0.75)
```



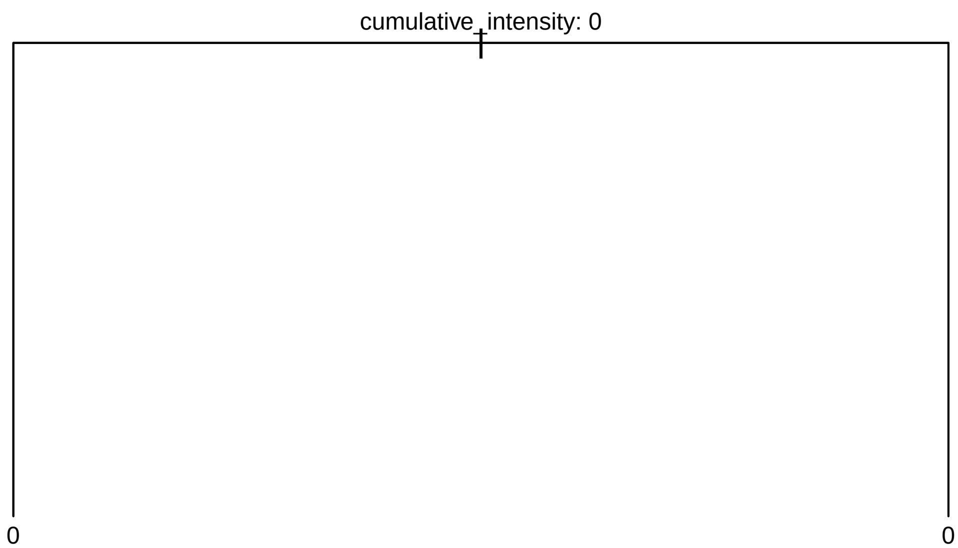

```r
# cv.tree() to determine whether pruning improves performance
cv.onset20=cv.tree(tree.onset20)
# It doesn't seem to be the case
plot(cv.onset20$size,cv.onset20$dev,type="b")
```




```r
# prune.tree(): function to prune to be used in case we wanted to prune the tree
prune.onset20=prune.tree(tree.onset20,best=5)
```

```
## Warning in prune.tree(tree.onset20, best = 5): best is bigger than tree size
```

```r
plot(prune.onset20)
text(prune.onset20,pretty=0,cex=0.75)
```




```
# Predicting based on CV results (i.e., use the unpruned tree)
yhat=predict(tree.onset20,newdata=test_valid)

# plot(yhat,test_valid$onset20)
# abline(0,1)
# Test error
mse=mean((yhat-test_valid$onset20)^2)
```

```
## Warning in Ops.factor(yhat, test_valid$onset20): '-' not meaningful for factors
```

```
mse
```

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

```
# This model leads to test predictions that are within around $5-6K of the true
# median home value for the suburb
sqrt(mse)
```

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

Random Forests

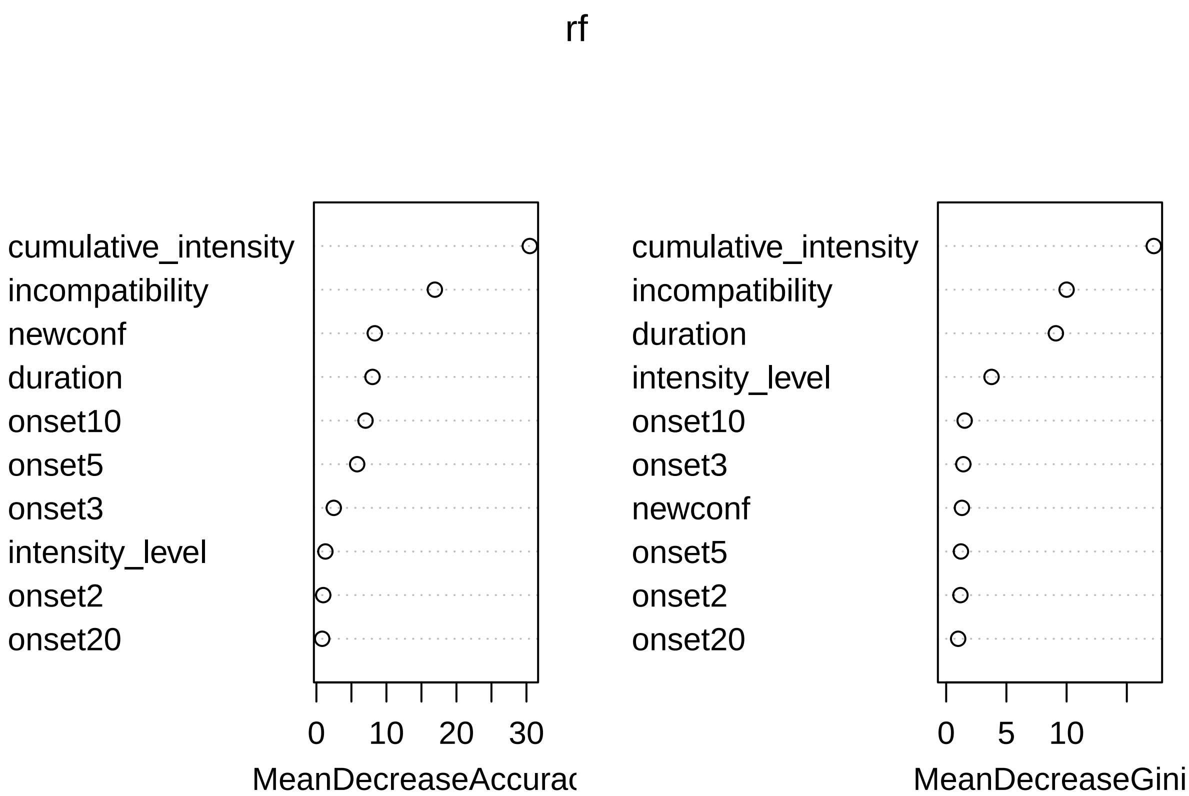
```r
# By default randomForest() uses m=p/3 for regression and m=sqrt(p) for classification
# Let's try m=6
set.seed(1)
rf=randomForest(ep_end ~ newconf+onset2+onset3+onset5+onset10+duration+incompatibility+intensity_level+c
yhat.rf = predict(tree.onset20,newdata=test_valid)

mean((yhat.rf-as.integer(test_valid$onset20))^2)
```

```
## [1] 1.31077
```

```r
# importance(): view the importance of each variable
# %IncMSE: mean decrease of accuracy in predictions on the OOB samples when a
# given variable is excluded from the model
# IncNodeImpurity: total decrease in node impurity that results from splits over
# that variable, averaged over all trees (RSS in regr. vs. deviance in class.)
importance(rf)
```

```
##                             0          1 MeanDecreaseAccuracy MeanDecreaseGini
## newconf              9.201656 -6.007442            8.3412216        1.3093954
## onset2               1.955090 -1.906697            0.9790256        1.1856673
## onset3               3.515001 -2.485181            2.4813395        1.4251111
## onset5               9.347825 -9.814094            5.8210619        1.2251029
## onset10              9.410704 -7.425381            7.0121900        1.5338187
## duration            12.534343 -9.563917            8.0113211        9.1080641
## incompatibility     19.804919  4.511146           16.9176142        9.9989336
## intensity_level     -3.112003  5.660214            1.2834327        3.7662430
## cumulative_intensity 18.746861 22.425895           30.4773404       17.2397718
## onset20              1.888401 -1.411583            0.8251728        0.9932799
```

```r
# varImpPlot(): Variance importance plot
varImpPlot(rf)
```



Other Models

- Penalized Logistic Regression -`plr`
- Conditional Inference Random Forest -`cforest`
- Random Forest - `rf`
- Bayesian Generalized Linear Model -`bayesglm`
- Boosted Generalized Additive Model - `gamboost`
- Support Vector Machines with Linear Kernel - `svmLinear`

```
library(caret)

#specify the cross-validation method
ctrl <- trainControl(method = "cv")

#fit a regression model and use LOOCV to evaluate performance
model <- train(ep_end ~ newconf+onset2+onset3+onset5+onset10+duration+incompatibility+intensity_level+cu

#view summary of LOOCV
print(model)
```

```
## k-Nearest Neighbors
```

```
##
## 832 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 749, 749, 748, 749, 748, 749, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6453815  0.1190491
##   7  0.6478772  0.1246593
##   9  0.6538439  0.1273210
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
predictions <- predict(model, test_better, type="raw")

message('0 for no conflict, 1 for new conflict')
```

```
## 0 for no conflict, 1 for new conflict
```

```
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```
# Confusion Matrix
table(predictions,test_better$ep_end)
```

```
##
## predictions   0   1
##           0 130  55
##           1   7  16
```

```
# Test Error
message('Test Error Rate')
```

```
## Test Error Rate
```

```
mean(predictions!=test_better$ep_end)
```

```
## [1] 0.2980769
```

```
confusionMatrix(data = predict(model, test_better), test_better$onset20)
```

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

```
##           Reference
## Prediction  0  1
##          0 92 93
##          1  9 14
##
##                Accuracy : 0.5096
##                  95% CI : (0.4396, 0.5794)
##     No Information Rate : 0.5144
##     P-Value [Acc > NIR] : 0.5826
##
##                   Kappa : 0.0408
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9109
##             Specificity : 0.1308
##          Pos Pred Value : 0.4973
##          Neg Pred Value : 0.6087
##              Prevalence : 0.4856
##          Detection Rate : 0.4423
##    Detection Prevalence : 0.8894
##       Balanced Accuracy : 0.5209
##
##        'Positive' Class : 0
##
```

Phase 3: Predicting War Susceptibility

Logistic Regression Model

```
# Making model with all input variables

#There is not enough variation in onset1 so we will not include in the regression
glm.fits = glm(intensity_level ~ newconf+onset2+onset3+onset5+onset10+onset20+duration+year_prev+duratio
               data = train_better, family = binomial)

# glm.fits = glm(duration ~ onset2+onset3+onset5+onset10+onset20, data = train_better)

summary(glm.fits)
```

```
##
## Call:
## glm(formula = intensity_level ~ newconf + onset2 + onset3 + onset5 +
##     onset10 + onset20 + duration + year_prev + duration + cumulative_intensity +
##     ep_end + incompatibility, family = binomial, data = train_better)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.69725  -0.42526  -0.00006  -0.00002   2.61536
##
## Coefficients: (1 not defined because of singularities)
```

```
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             60.03491  836.96455   0.072 0.942817
## newconf1                -3.82102    1.84333  -2.073 0.038182 *
## onset21                  0.77111    0.53679   1.437 0.150851
## onset31                 -0.18438    0.54513  -0.338 0.735192
## onset51                  0.69125    0.59928   1.153 0.248721
## onset101                -1.07498    0.61835  -1.738 0.082131 .
## onset201                -0.77004    0.74490  -1.034 0.301253
## duration                      NA         NA      NA       NA
## year_prev               -0.04172    0.01152  -3.621 0.000293 ***
## cumulative_intensity1   20.60617  836.65014   0.025 0.980351
## ep_end1                 -0.09638    0.27060  -0.356 0.721723
## incompatibility          0.24231    0.21655   1.119 0.263148
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 788.15  on 831  degrees of freedom
## Residual deviance: 434.79  on 821  degrees of freedom
## AIC: 456.79
##
## Number of Fisher Scoring iterations: 19
```

```r
# Make predictions based on model
glm.probs = predict(glm.fits,test_better, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
# Initialize vector with 109 elements
glm.pred = rep(0, nrow(test_better))
# Assign 1 to probabilities > 0.5
glm.pred[glm.probs >.5]=1

message('0 for no conflict, 1 for new conflict')
```

```
## 0 for no conflict, 1 for new conflict
```

```r
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```r
# Confusion Matrix
table(glm.pred,test_better$intensity_level)
```

```
##
## glm.pred   0   1
##        0 142  14
##        1  15  37
```

```r
# Test Error
message('Test Error Rate')
```

```
## Test Error Rate
```

```r
mean(glm.pred!=test_better$intensity_level)
```

```
## [1] 0.1394231
```

LDA Model

```r
# Making model with all input variables
lda.fit=lda(intensity_level ~ newconf+onset2+onset3+onset5+onset10+onset20+duration+year_prev+duration+c
               data = train_better)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
lda.fit
```

```
## Call:
## lda(intensity_level ~ newconf + onset2 + onset3 + onset5 + onset10 +
##     onset20 + duration + year_prev + duration + cumulative_intensity +
##     ep_end + incompatibility, data = train_better)
##
## Prior probabilities of groups:
##         0         1
## 0.8185096 0.1814904
##
## Group means:
##    newconf1   onset21   onset31   onset51  onset101  onset201 duration
## 0 0.4419971 0.8061674 0.7224670 0.6138032 0.5697504 0.4831131 4.637298
## 1 0.6887417 0.9470199 0.8874172 0.8278146 0.7682119 0.7086093 5.827815
##   year_prev cumulative_intensity1   ep_end1 incompatibility
## 0  1911.604             0.4346549 0.3803231       0.3274596
## 1  1865.384             1.0000000 0.2715232       0.4304636
##
## Coefficients of linear discriminants:
##                                LD1
## newconf1              -1.910244432
## onset21                0.382364337
## onset31                0.130624856
## onset51                0.546471805
## onset101              -0.509113280
## onset201              -0.530473192
## duration              -0.091087508
## year_prev             -0.023597594
## cumulative_intensity1  2.289064670
## ep_end1                0.003104976
## incompatibility        0.130908183
```

```r
summary(lda.fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   22     -none- numeric
## scaling 11     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  8     -none- list
```

```r
lda.pred <- predict(lda.fit , test_better)

message('2 for benign, 4 for malignant')
```

```
## 2 for benign, 4 for malignant
```

```r
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```r
# Confusion Matrix
lda.class <- lda.pred$class
table(lda.class, test_better$ep_end)
```

```
##
## lda.class  0  1
##         0 97 59
##         1 40 12
```

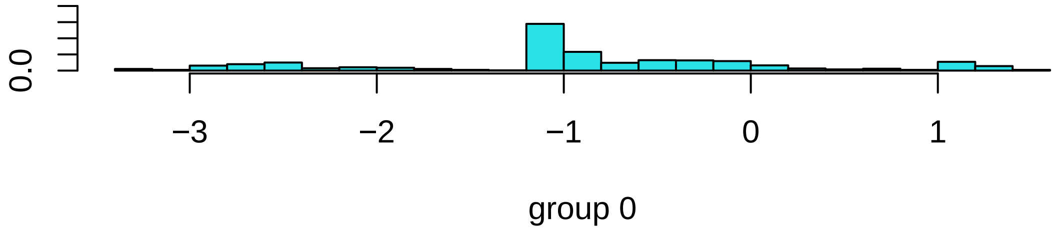
```r
message('Test Error Rate')
```

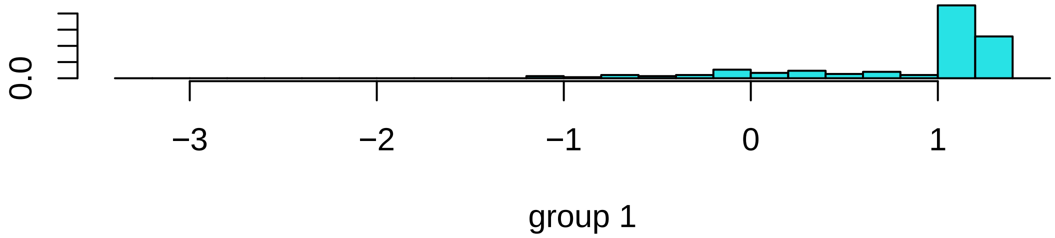
```
## Test Error Rate
```

```r
# Test Error
mean(lda.class != test_better$ep_end)
```

```
## [1] 0.4759615
```

```r
plot(lda.fit)
```





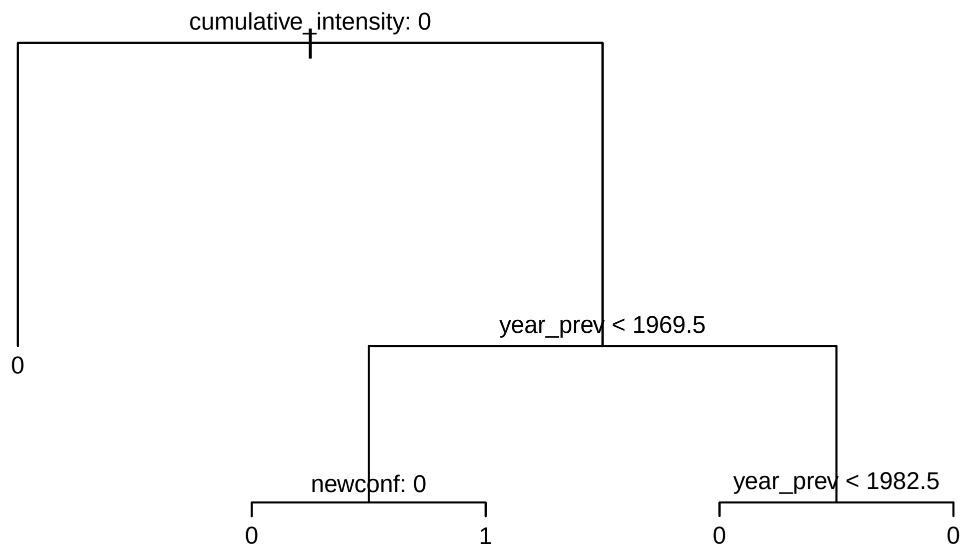
Linear Discriminants

Decision Trees

```
tree.onset20=tree(intensity_level ~ newconf+onset2+onset3+onset5+onset10+onset20+duration+year_prev+dura
summary(tree.onset20)
```

```
##
## Classification tree:
## tree(formula = intensity_level ~ newconf + onset2 + onset3 +
##     onset5 + onset10 + onset20 + duration + year_prev + duration +
##     cumulative_intensity + ep_end + incompatibility, data = war_data_valid)
## Variables actually used in tree construction:
## [1] "cumulative_intensity" "year_prev"            "newconf"
## Number of terminal nodes:  5
## Residual mean deviance:  0.5435 = 562.6 / 1035
## Misclassification error rate: 0.1288 = 134 / 1040
```

```
plot(tree.onset20)
text(tree.onset20, pretty = 0, cex=0.75)
```



Decision Trees (With Training & Testing)

```
# Train using training set
tree.onset20=tree(intensity_level ~ newconf+onset2+onset3+onset5+onset10+onset20+duration+year_prev+dura

# Test on test set using predict()
# type="class" to return the class prediction
tree.pred=predict(tree.onset20,test_better,type="class")

# Confusion matrix
conf.matrix <- table(tree.pred,test_better$intensity_level)
conf.matrix
```

```
##
## tree.pred   0   1
##         0 142  15
##         1  15  36
```

```
# Accuracy on test set
(conf.matrix[1,1] + conf.matrix[2,2])/(conf.matrix[1,1] + conf.matrix[2,2] + conf.matrix[1,2]+ conf.mat
```

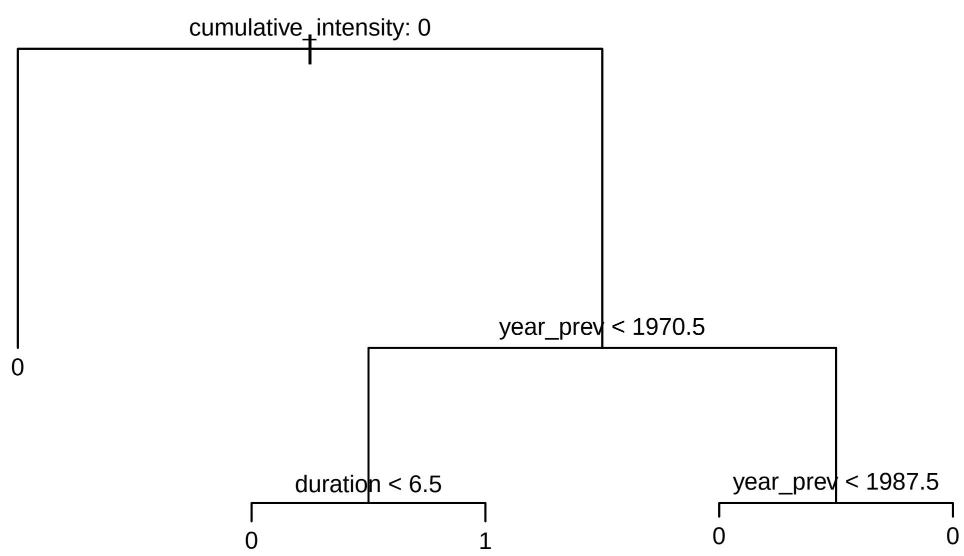
```
## [1] 0.8557692
```

Regression Trees

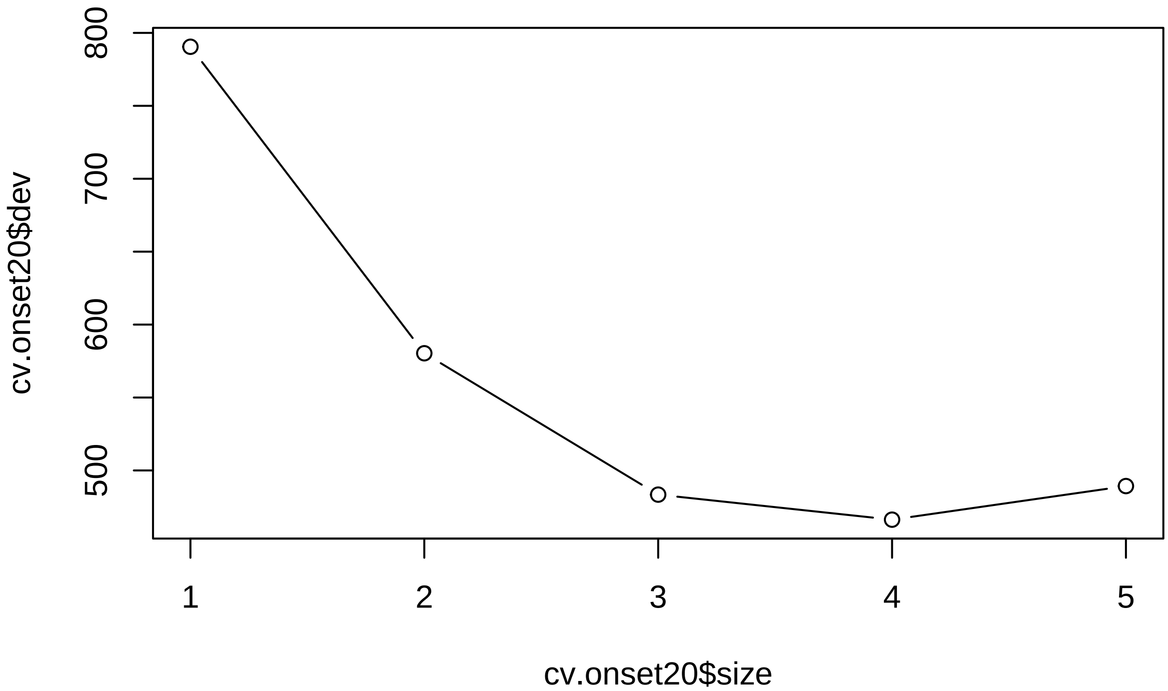
```
set.seed(1)

tree.onset20=tree(intensity_level ~ newconf+onset2+onset3+onset5+onset10+onset20+duration+year_prev+dura
# Only a few of the variables were used in constructing the tree
# lstat: percentage of individuals with lower socioeconomic status
summary(tree.onset20)
```

```
##
## Classification tree:
## tree(formula = intensity_level ~ newconf + onset2 + onset3 +
##     onset5 + onset10 + onset20 + duration + year_prev + duration +
##     cumulative_intensity + ep_end + incompatibility, data = train_better)
## Variables actually used in tree construction:
## [1] "cumulative_intensity" "year_prev"            "duration"
## Number of terminal nodes:  5
## Residual mean deviance:  0.5278 = 436.5 / 827
## Misclassification error rate: 0.125 = 104 / 832
```

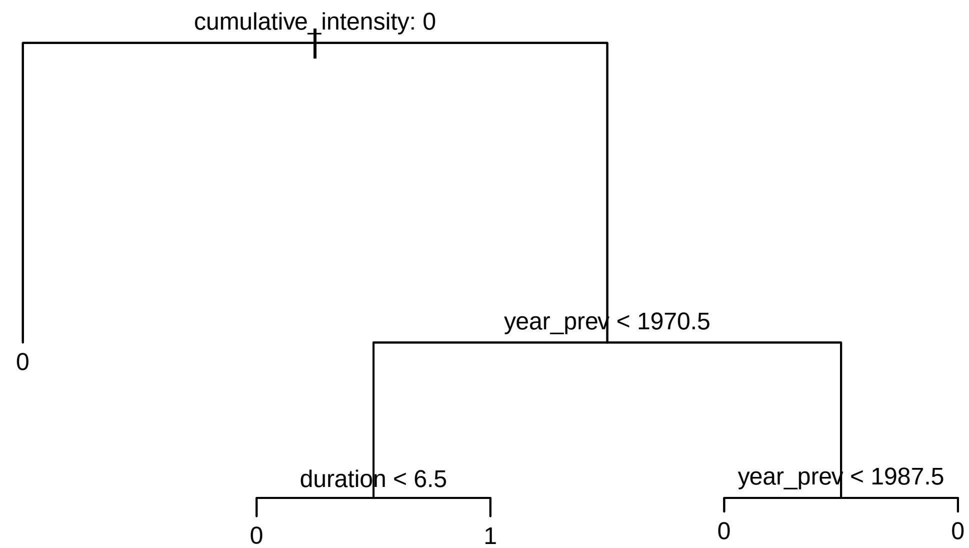
```
# Plot the tree
# Lower values of lstat correspond to more expensive houses
plot(tree.onset20)
text(tree.onset20,pretty=0,cex=0.75)
```



```r
# cv.tree() to determine whether pruning improves performance
cv.onset20=cv.tree(tree.onset20)
# It doesn't seem to be the case
plot(cv.onset20$size,cv.onset20$dev,type="b")
```




```r
# prune.tree(): function to prune to be used in case we wanted to prune the tree
prune.onset20=prune.tree(tree.onset20,best=5)
plot(prune.onset20)
text(prune.onset20,pretty=0,cex=0.75)
```




```
# Predicting based on CV results (i.e., use the unpruned tree)
yhat=predict(tree.onset20,newdata=test_better)

# plot(yhat,test_valid$onset20)
# abline(0,1)
# Test error
mse=mean((yhat-test_valid$onset20)^2)
```

```
## Warning in Ops.factor(yhat, test_valid$onset20): '-' not meaningful for factors
```

```
mse
```

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

```
# This model leads to test predictions that are within around $5-6K of the true
# median home value for the suburb
sqrt(mse)
```

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

Random Forests

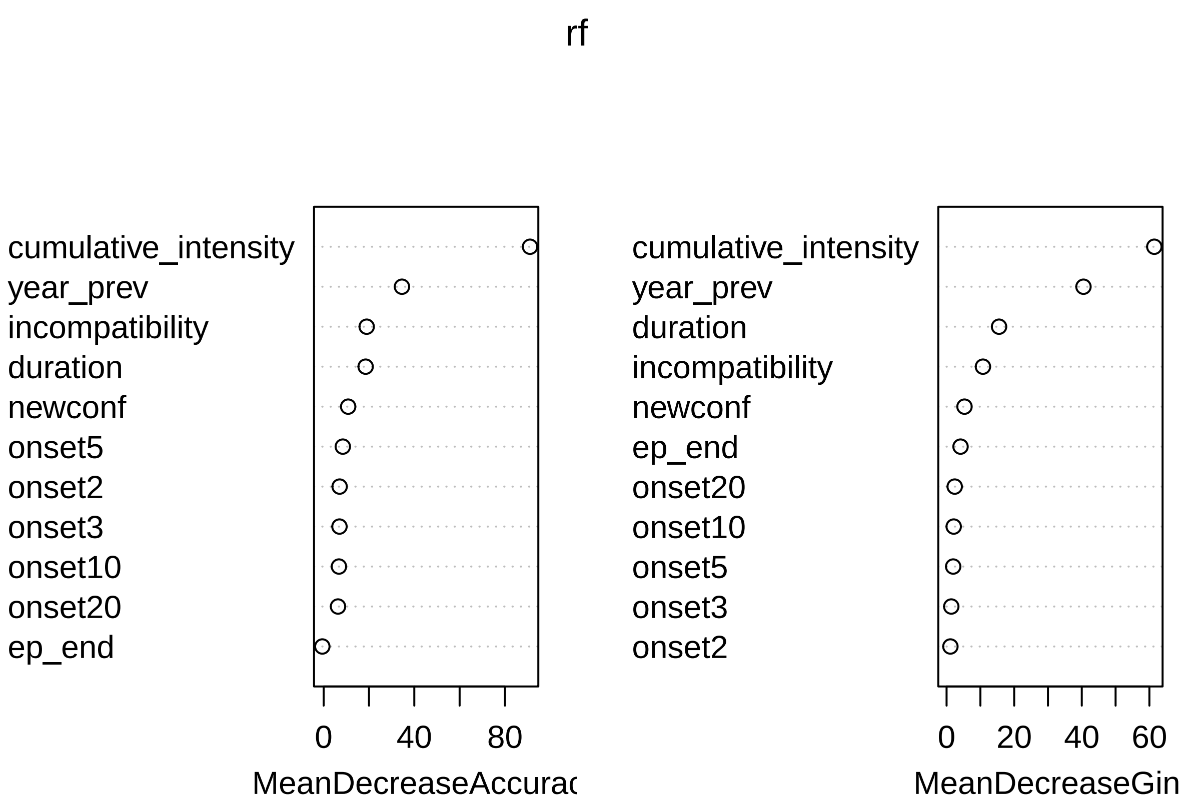
```r
# By default randomForest() uses m=p/3 for regression and m=sqrt(p) for classification
# Let's try m=6
set.seed(1)
rf=randomForest(intensity_level ~ newconf+onset2+onset3+onset5+onset10+onset20+duration+year_prev+durati
yhat.rf = predict(tree.onset20,newdata=test_better)

mean((yhat.rf-as.integer(test_better$intensity_level))^2)
```

```
## [1] 0.8921158
```

```r
# importance(): view the importance of each variable
# %IncMSE: mean decrease of accuracy in predictions on the OOB samples when a
# given variable is excluded from the model
# IncNodeImpurity: total decrease in node impurity that results from splits over
# that variable, averaged over all trees (RSS in regr. vs. deviance in class.)
importance(rf)
```

```
##                            0          1 MeanDecreaseAccuracy MeanDecreaseGini
## newconf              5.473129 13.0509594            10.823956         5.296934
## onset2               5.752058  3.3409108             7.056920         1.126125
## onset3               6.575704  2.5822461             6.976675         1.390664
## onset5               8.881560 -2.9449943             8.453287         1.952786
## onset10              3.268522  5.7462269             6.746310         2.109857
## onset20             -6.250389  8.1246338             6.334236         2.410396
## duration            13.288941 15.1205533            18.528238        15.521490
## year_prev           24.056893 21.2224538            34.542379        40.501639
## cumulative_intensity 85.197794 82.4450055            91.054931        61.450479
## ep_end              -1.223064  0.7097914            -0.587375         4.125935
## incompatibility     16.503136  9.9642445            18.991830        10.754785
```

```r
# varImpPlot(): Variance importance plot
varImpPlot(rf)
```



Other Models

- Penalized Logistic Regression -`plr`
- Conditional Inference Random Forest -`cforest`
- Random Forest - `rf`
- Bayesian Generalized Linear Model -`bayesglm`
- Boosted Generalized Additive Model - `gamboost`
- Support Vector Machines with Linear Kernel - `svmLinear`

```
library(caret)

#specify the cross-validation method
ctrl <- trainControl(method = "cv")

#fit a regression model and use LOOCV to evaluate performance
model <- train(intensity_level ~ newconf+onset2+onset3+onset5+onset10+onset20+duration+year_prev+durati

#view summary of LOOCV
print(model)
```

```
## k-Nearest Neighbors
##
## 832 samples
##  11 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 749, 749, 749, 749, 749, 749, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.8714859  0.5869409
##   7  0.8738669  0.5899490
##   9  0.8738669  0.5904556
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```r
predictions <- predict(model, test_better, type="raw")

message('0 for no conflict, 1 for new conflict')
```

```
## 0 for no conflict, 1 for new conflict
```

```r
message('Confusion Matrix')
```

```
## Confusion Matrix
```

```r
# Confusion Matrix
table(predictions,test_better$intensity_level)
```

```
##
## predictions   0   1
##           0 142  15
##           1  15  36
```

```r
# Test Error
message('Test Error Rate')
```

```
## Test Error Rate
```

```r
mean(predictions!=test_better$intensity_level)
```

```
## [1] 0.1442308
```

```r
confusionMatrix(data = predict(model, test_better), test_better$ep_end)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 98 59
##          1 39 12
##
##                Accuracy : 0.5288
##                  95% CI : (0.4586, 0.5982)
##     No Information Rate : 0.6587
##     P-Value [Acc > NIR] : 0.99996
##
##                   Kappa : -0.1241
##
##  Mcnemar's Test P-Value : 0.05495
##
##             Sensitivity : 0.7153
##             Specificity : 0.1690
##          Pos Pred Value : 0.6242
##          Neg Pred Value : 0.2353
##              Prevalence : 0.6587
##          Detection Rate : 0.4712
##    Detection Prevalence : 0.7548
##       Balanced Accuracy : 0.4422
##
##        'Positive' Class : 0
##
```