Perform the Logistic Regression analysis of the credit data

Step 1: Collecting data

```
credit <-
read.csv("http://www.sci.csueastbay.edu/~esuess/classes/Statistics_6620/Prese
ntations/ml7/credit.csv")

str(credit)

## 'data.frame':    1000 obs. of  17 variables:
##  $ checking_balance    : Factor w/ 4 levels "< 0 DM","> 200 DM",..: 1 3 4
1 1 4 4 3 4 3 ...
##  $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
##  $ credit_history      : Factor w/ 5 levels "critical","good",..: 1 2 1 2
4 2 2 2 2 1 ...
##  $ purpose             : Factor w/ 6 levels "business","car",..: 5 5 4 5 2
4 5 2 5 2 ...
##  $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059
5234 ...
##  $ savings_balance     : Factor w/ 5 levels "< 100 DM","> 1000 DM",..: 5 1
1 1 1 5 4 1 2 1 ...
##  $ employment_duration : Factor w/ 5 levels "< 1 year","> 7 years",..: 2 3
4 4 3 3 2 3 4 5 ...
##  $ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ other_credit        : Factor w/ 3 levels "bank","none",..: 2 2 2 2 2 2
2 2 2 2 ...
##  $ housing             : Factor w/ 3 levels "other","own",..: 2 2 2 1 1 1
2 3 2 2 ...
##  $ existing_loans_count: int  2 1 1 1 2 1 1 1 1 2 ...
##  $ job                 : Factor w/ 4 levels "management","skilled",..: 2 2
4 2 2 4 2 1 4 1 ...
##  $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ phone               : Factor w/ 2 levels "no","yes": 2 1 1 1 1 2 1 2 1
1 ...
##  $ default             : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 1 1 1
2 ...
```

Step 2: Exploring and preparing the data

- Fix the default variable to be 0 or 1

```
credit$default = as.numeric(credit$default)
credit$default = credit$default - 1
```

- Set up trainning and test data sets

```
indx = sample(1:nrow(credit), as.integer(0.9*nrow(credit)))
indx
```

```
##   [1]  622  983  500  784  922  290  923  988  964  480  356  822  153 
420
##  [15]  557  222  537  385  509  552  690  295   56  898  603  262   95 
372
##  [29]  419   68  593  365  524  883  914  177  545  457  300  166  776 
28
##  [43]  989  738  499  337  158  180  196  619  357  431  168  322  390 
623
##  [57]  944  864  597   90  345  406  717  814  644  426  659  245  126 
843
##  [71]  589   77  646  555   39  445  574  256  194  811  874  252  667 
81
##  [85]  917  312  792  835  511  446  531  193  281  473  570  976  487 
398
##  [99]  871  813  263  427  307   50  825  137  523   85   19  129  490 
207
## [113]  399  796  550  239  819  585  921  972  639  971  798  583  540 
859
## [127]   43  587  960  236  675  996  886  131  654  305  756  402  472 
538
## [141]  767    9  519  815  670   29  925  617   76  128  435  697  893 
466
## [155]   84  596  161  121  343  626  817  993  567  430  987  744  265 
515
## [169]  642  237  416   59  407  243   31  602  876  314  832  809  762 
561
## [183]   44  458  645  340  181  282  581  580   62  553  504  179  590 
973
## [197]  270  164  199  479  788  187  211  808  370  956  920  606  662 
769
## [211]  350  293  565  503  719   23   11   10  735  306  584  942  298 
786
## [225]  701  191  230  829  764  600  608  507  475  547  772  279  748 
70
## [239]   78  860  984  907  682  145  939  202  636  899  691  502   46 
369
## [253]  421  310   26  643  789  341  223    4  900  143  342  761  174 
65
## [267]  712  847  688  409  588  440  216  855  378  273  213  936  799 
393
## [281]  424  878  297  460  962   38  414  117  997  656  146  946  908 
889
## [295]  410  950  477  834  943  134  362  707  828  302  857  172  510 
981
## [309]  870  783  614  349  810  225   63  520  542   51  903   79   83 
615
## [323]  732  591  316   91  665    2  484  104  476  800  106  666  189 
178
## [337]  840  452    8  404  773  444  687  638  564  653  594  319  336 
785
```

```
## [351]  849  156  721  660  373   92  952  235  381  142  459  853  188
321
## [365]  632  579  562  330  703  367  974  348  872  861  854  354  715
198
## [379]  858  836  100  751  401  227  880  795  495  965  959  432  368
16
## [393]  136  266  866  411  217  968  787  765  148  558  897  621  259
846
## [407]  901  852  508  949  212  185  573  247  758   42  260  933  862
394
## [421]  778  830   25  423  678   55  549  556  827  482  940  113   30
955
## [435]  278  276  150  534  982  471  124  652  346  422  575  882  412
651
## [449]  434  906  605  686  648  916  548  729  496  467  339  233  453
122
## [463]  530  360  231   14  991  986   73  232  447  489  533  902  804
465
## [477]  692  820  160  288  867  680  601  926  627  672  837  253  165
5
## [491]  396  529  693  842  895  747  985  527  681  116  528  954  637
386
## [505]  200  668  904  718  563  284  905  344  112  775   97  780  909
823
## [519]  572  461  877  210  560  912  863  109  492    3  448  329  289
35
## [533]  182   52  283  677  115  418  449  244  366  184  274   21  286
141
## [547]  395  474  992  655  110  839  980  797  209  197  167  628  353
277
## [561]  709  741  380  328   24  454  812   12  892  228  685  481  195
759
## [575]  169  363   75  291  234  928  727  726  428  818   37  351  633
544
## [589]  824  754  347  135  152  856  190  723  127  326  292   96  757
518
## [603]  664  485  486  915  816  224  722  280  844  102  896   54   89
311
## [617]  255   64  455  436  551  140  375  994  728   58  248  661  387
635
## [631]   40   74  845  303  201   17  438  358  220  183  649  999  546
323
## [645]  716  413  376  881  391  451  522  970  910   36  689  384  250
215
## [659]  334  879  979  205  674  541  441    6  873  506  497   48  746
535
## [673]  186   99  214  838  491  313  111  958  159  885  442  841  332
238
## [687]  736  720  609  673  469  966  887   67  793  766  333  221  604
750
```

```
## [701]  157  694  269   80  571  155  516  612  249   47  932  770  724
285
## [715]  592  945  114  779  371  462  582  961  208  705  948  805   22
935
## [729]  130  930  977  468   27  494  725  577  714  304  888   69  257
299
## [743]  782  138  173  918  663  403  911  379  242  382   49   61  620
745
## [757]  478  268  464  998  803  634  702  327  147  598  831  696  101
919
## [771]  826  463  505  967  833  706  493  978  392   32  450  133  731
361
## [785]  640  192   98  389  532  359  641  768  708  203  990  320   45
443
## [799]  521  913  777  713 1000  934  684  794  294   41  275  241  301
296
## [813]   71  740    7  417  229  938  163  658  753  624  611  338  755
33
## [827]  607  175   72  219  576  821  850   86  433  149  425   87  760
618
## [841]  679  118  657   57  781  261  699   60  331  890  807  139  698
671
## [855]  995  730  272  613  963  969  743  868  891  364  374  352   15
630
## [869]  132  315   13  125  739  246  586  526   88  869  470  218  975
894
## [883]  251  154  629  951  107  650  162  151    1  806  105  176  318
771
## [897]  287  927  324  123

credit_train = credit[indx,]
credit_test = credit[-indx,]

credit_train_labels = credit[indx,17]
credit_test_labels = credit[-indx,17]
```

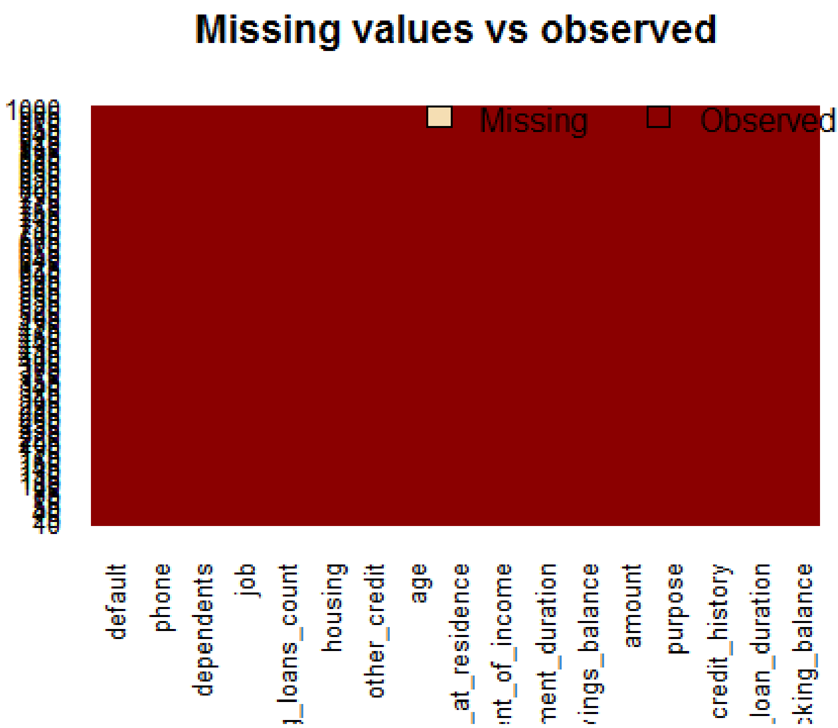
- Check if there are any missing values "Missing values vs observed"

```
library(Amelia)

## Loading required package: Rcpp

## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.4, built: 2015-12-05)
## ## Copyright (C) 2005-2017 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##

missmap(credit, main = "Missing values vs observed")
```



- Number of missing values in each column

```
sapply(credit,function(x) sum(is.na(x)))

##     checking_balance months_loan_duration       credit_history
##                    0                    0                    0
##              purpose               amount      savings_balance
##                    0                    0                    0
##  employment_duration    percent_of_income  years_at_residence
##                    0                    0                    0
##                  age         other_credit              housing
##                    0                    0                    0
## existing_loans_count                  job           dependents
##                    0                    0                    0
##                phone              default
##                    0                    0
```

- Number of unique values in each column

```
sapply(credit, function(x) length(unique(x)))

##     checking_balance months_loan_duration       credit_history
##                    4                   33                    5
##              purpose               amount      savings_balance
##                    6                  921                    5
##  employment_duration    percent_of_income  years_at_residence
##                    5                    4                    4
##                  age         other_credit              housing
##                   53                    3                    3
```

```
## existing_loans_count                    job          dependents
##                    4                    4                   2
##                phone              default
##                    2                    2
```

Step 3: Training a model on the data

- Fit the logistic regression model, with all predictor variables
- Look at the model that gives the smallest AIC

```
model <- glm(default ~.,family=binomial(link='logit'),data=credit_train)
model

## 
## Call:  glm(formula = default ~ ., family = binomial(link = "logit"), 
##     data = credit_train)
## 
## Coefficients:
##                  (Intercept)        checking_balance> 200 DM  
##                   -2.3686161                      -0.9228393  
##     checking_balance1 - 200 DM         checking_balanceunknown  
##                   -0.3475554                      -1.6691070  
##           months_loan_duration              credit_historygood  
##                    0.0298884                       1.0409840  
##          credit_historyperfect              credit_historypoor  
##                    1.8588390                       0.8850255  
##        credit_historyvery good                      purposecar  
##                    1.5285675                       0.2596966  
##                    purposecar0                purposeeducation  
##                   -1.1965242                       0.6394282  
##    purposefurniture/appliances               purposerenovations  
##                   -0.0691580                       0.7433135  
##                         amount          savings_balance> 1000 DM  
##                    0.0001111                      -1.0372568  
##    savings_balance100 - 500 DM     savings_balance500 - 1000 DM  
##                   -0.1618063                      -0.5772145  
##         savings_balanceunknown     employment_duration> 7 years  
##                   -0.9356791                      -0.4751296  
## employment_duration1 - 4 years   employment_duration4 - 7 years  
##                   -0.2854140                      -0.9923960  
##  employment_durationunemployed                percent_of_income  
##                    0.0392832                       0.3325119  
##             years_at_residence                              age  
##                    0.0187588                      -0.0096494  
##               other_creditnone               other_creditstore  
##                   -0.5032612                      -0.0014702  
##                   housingown                      housingrent  
##                   -0.0765991                       0.3401566  
##           existing_loans_count                      jobskilled  
##                    0.4145610                      -0.0246342  
##                  jobunemployed                     jobunskilled  
```

```
##                      -0.0816743                              -0.1222618
##                      dependents                                phoneyes
##                       0.0686739                              -0.2728584
##
## Degrees of Freedom: 899 Total (i.e. Null);  864 Residual
## Null Deviance:       1101
## Residual Deviance: 846.4     AIC: 918.4
```

```
summary(model)
```

```
##
## Call:
## glm(formula = default ~ ., family = binomial(link = "logit"),
##     data = credit_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9448  -0.7506  -0.4021   0.7966   2.5451
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -2.369e+00  9.615e-01  -2.464 0.013757 *
## checking_balance> 200 DM      -9.228e-01  3.901e-01  -2.365 0.018010 *
## checking_balance1 - 200 DM    -3.476e-01  2.186e-01  -1.590 0.111823
## checking_balanceunknown       -1.669e+00  2.370e-01  -7.043 1.89e-12 ***
## months_loan_duration           2.989e-02  9.319e-03   3.207 0.001340 **
## credit_historygood             1.041e+00  2.736e-01   3.804 0.000142 ***
## credit_historyperfect          1.859e+00  4.714e-01   3.944 8.03e-05 ***
## credit_historypoor             8.850e-01  3.401e-01   2.602 0.009262 **
## credit_historyvery good        1.529e+00  4.380e-01   3.490 0.000483 ***
## purposecar                     2.597e-01  3.266e-01   0.795 0.426585
## purposecar0                   -1.197e+00  8.392e-01  -1.426 0.153910
## purposeeducation               6.394e-01  4.539e-01   1.409 0.158947
## purposefurniture/appliances   -6.916e-02  3.220e-01  -0.215 0.829956
## purposerenovations             7.433e-01  6.136e-01   1.211 0.225769
## amount                         1.111e-04  4.388e-05   2.533 0.011321 *
## savings_balance> 1000 DM      -1.037e+00  5.068e-01  -2.047 0.040674 *
## savings_balance100 - 500 DM   -1.618e-01  2.832e-01  -0.571 0.567800
## savings_balance500 - 1000 DM  -5.772e-01  4.518e-01  -1.278 0.201385
## savings_balanceunknown        -9.357e-01  2.657e-01  -3.521 0.000430 ***
## employment_duration> 7 years  -4.751e-01  2.971e-01  -1.599 0.109715
## employment_duration1 - 4 years -2.854e-01  2.408e-01  -1.185 0.235867
## employment_duration4 - 7 years -9.924e-01  3.055e-01  -3.248 0.001160 **
## employment_durationunemployed   3.928e-02  4.216e-01   0.093 0.925764
## percent_of_income              3.325e-01  8.873e-02   3.748 0.000179 ***
## years_at_residence             1.876e-02  8.790e-02   0.213 0.831012
## age                           -9.649e-03  9.186e-03  -1.050 0.293509
## other_creditnone              -5.033e-01  2.442e-01  -2.061 0.039329 *
## other_creditstore             -1.470e-03  4.312e-01  -0.003 0.997280
## housingown                    -7.660e-02  3.045e-01  -0.252 0.801372
```

```
## housingrent                      3.402e-01  3.541e-01   0.961 0.336726
## existing_loans_count             4.146e-01  1.990e-01   2.083 0.037265 *
## jobskilled                      -2.463e-02  2.872e-01  -0.086 0.931650
## jobunemployed                   -8.167e-02  6.396e-01  -0.128 0.898392
## jobunskilled                    -1.223e-01  3.486e-01  -0.351 0.725809
## dependents                       6.867e-02  2.502e-01   0.274 0.783740
## phoneyes                        -2.729e-01  2.056e-01  -1.327 0.184372
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1101.2  on 899  degrees of freedom
## Residual deviance:  846.4  on 864  degrees of freedom
## AIC: 918.4
##
## Number of Fisher Scoring iterations: 5
```

```
anova(model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: default
##
## Terms added sequentially (first to last)
##
##
##                      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                   899    1101.25
## checking_balance      3  112.514       896     988.73 < 2.2e-16 ***
## months_loan_duration  1   41.090       895     947.64 1.454e-10 ***
## credit_history        4   32.633       891     915.01 1.420e-06 ***
## purpose               5    6.819       886     908.19 0.2344505
## amount                1    0.451       885     907.74 0.5018424
## savings_balance       4   18.512       881     889.23 0.0009797 ***
## employment_duration   4   13.208       877     876.02 0.0103042 *
## percent_of_income     1   12.668       876     863.35 0.0003721 ***
## years_at_residence    1    0.188       875     863.16 0.6642070
## age                   1    1.819       874     861.34 0.1774551
## other_credit          2    5.406       872     855.94 0.0670024 .
## housing               2    3.238       870     852.70 0.1980781
## existing_loans_count  1    4.303       869     848.40 0.0380510 *
## job                   3    0.132       866     848.26 0.9876728
## dependents            1    0.084       865     848.18 0.7717120
## phone                 1    1.776       864     846.40 0.1826232
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Drop the insignificant predictors, alpha = 0.10

```
model <- glm(default ~ checking_balance + months_loan_duration + 
credit_history +  percent_of_income + 
age,family=binomial(link='logit'),data=credit_train)
model

## 
## Call:  glm(formula = default ~ checking_balance + months_loan_duration + 
##     credit_history + percent_of_income + age, family = binomial(link = 
"logit"), 
##     data = credit_train)
## 
## Coefficients:
##                (Intercept)    checking_balance> 200 DM  
##                  -1.765778                   -1.126488  
## checking_balance1 - 200 DM     checking_balanceunknown  
##                  -0.446300                   -1.844604  
##       months_loan_duration          credit_historygood  
##                   0.035747                    0.698223  
##      credit_historyperfect          credit_historypoor  
##                   2.030841                    0.820808  
##    credit_historyvery good           percent_of_income  
##                   1.448799                    0.210030  
##                        age  
##                  -0.009849  
## 
## Degrees of Freedom: 899 Total (i.e. Null);  889 Residual
## Null Deviance:       1101 
## Residual Deviance: 905.6     AIC: 927.6

summary(model)

## 
## Call:
## glm(formula = default ~ checking_balance + months_loan_duration + 
##     credit_history + percent_of_income + age, family = binomial(link = 
"logit"), 
##     data = credit_train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.8435  -0.7941  -0.4706   0.8862   2.3924  
## 
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)                -1.765778   0.442123  -3.994 6.50e-05 ***
## checking_balance> 200 DM   -1.126488   0.367265  -3.067 0.002160 ** 
## checking_balance1 - 200 DM -0.446300   0.197583  -2.259 0.023896 *  
## checking_balanceunknown    -1.844604   0.218692  -8.435  < 2e-16 ***
## months_loan_duration        0.035747   0.006644   5.380 7.44e-08 ***
```

```
## credit_historygood          0.698223   0.211349   3.304 0.000954 ***
## credit_historyperfect       2.030841   0.443369   4.580 4.64e-06 ***
## credit_historypoor          0.820808   0.316537   2.593 0.009512 **
## credit_historyvery good     1.448799   0.379122   3.821 0.000133 ***
## percent_of_income           0.210030   0.074854   2.806 0.005018 **
## age                        -0.009849   0.007528  -1.308 0.190752
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1101.25  on 899  degrees of freedom
## Residual deviance:  905.58  on 889  degrees of freedom
## AIC: 927.58
##
## Number of Fisher Scoring iterations: 4
```

```
anova(model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: default
##
## Terms added sequentially (first to last)
##
##
##                      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                   899    1101.25
## checking_balance      3  112.514       896     988.73 < 2.2e-16 ***
## months_loan_duration  1   41.090       895     947.64 1.454e-10 ***
## credit_history        4   32.633       891     915.01 1.420e-06 ***
## percent_of_income     1    7.687       890     907.32  0.005561 **
## age                   1    1.739       889     905.58  0.187324
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Step 4: Evaluating model performance

- Check Accuracy which is 75% accurate

```
fitted.results <- predict(model,newdata=credit_test,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)

misClasificError <- mean(fitted.results != credit_test$default)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.72"
```

Step 5: Improving model performance

- Using ROC curve to improve the model
- This classifier has an AUC of .77 which shows that the classifier has done a pretty good job at classifying bad credit and good credit.

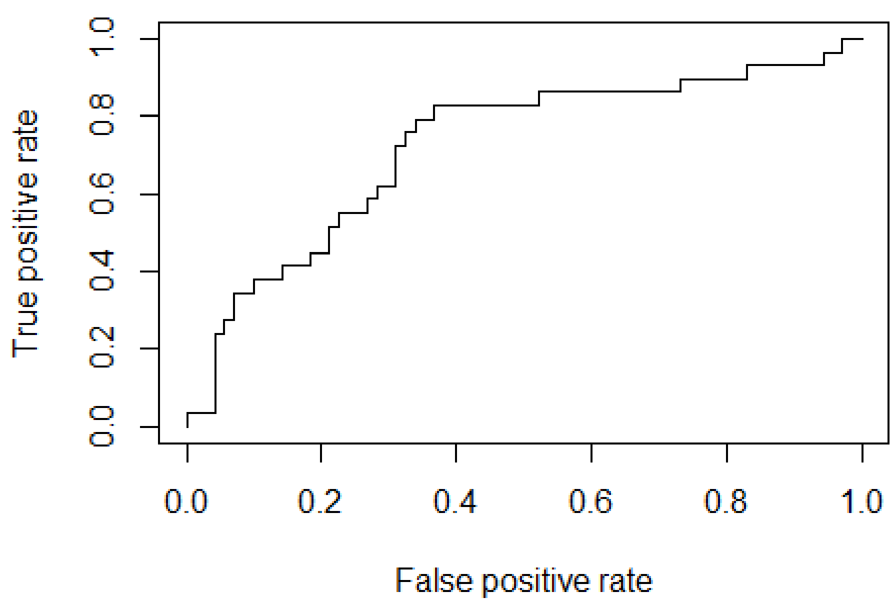
```
library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

p <- predict(model, newdata=credit_test, type="response")
pr <- prediction(p, credit_test$default)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc

## [1] 0.7231666
```