Assignment 3: Comparing discriminant rules. ROC curve and other methods

Víctor Villegas, Roger Llorenç, Luis Sierra

2024-03-05

Loading the necessary libraries and datasets.

```
while (dev.cur() != 1) {
  dev.off()
}
# Clears global environment
rm(list=ls())

library(wrapr)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:wrapr':
##
##     pack, unpack
```

```
## Loaded glmnet 4.1-8
```

```
library(class)
```

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

```
library(pROC)
```

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

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
path_1 <- ("C:/Users/Roger/Documents/3_UPC/1_Master_math/9_statistical_learning/5_spam_email_database/s
df <- read.table(path_1,sep=",")

path_2 <- ("C:/Users/Roger/Documents/3_UPC/1_Master_math/9_statistical_learning/5_spam_email_database/s
df.names <- c(read.table(path_2,sep=":",skip=33,nrows=53,as.is=TRUE)[,1],
              "char_freq_#",
              read.table(path_2,sep=":",skip=87,nrows=3,as.is=TRUE)[,1],
```

```
              "spam.01")
names(df) <- df.names # Adds headers to the df



spam_all <- df[df[, dim(df)[2]] == 1, ] # We have 1813 spam
no_spam_all <- df[df[, dim(df)[2]] == 0, ] # We have 2788 no-spam
```

Preparing the datasets.

We want to divide data into two parts: 2/3 for the training sample, 1/3 for the test sample, such that 2/3 are SPAM in training and also in test. This leads to 5/9 of the data will be SPAM and 4/9 NO-SPAM.

We have 2788 SPAM and 1813 NO-SPAM observations. We select a number of SPAM observations of such that it is multiple of 9,4,5 because of the ratios, that is smaller than 2788 (obviously), that 4/9 is smaller than 1813 and that is an integer so that partitions are exact. That is to solve for a: a$4$5*9 < 1813

```
a = floor(dim(spam_all)[1]/(4*5*9))
n_spam = a*9*4*5; n_no_spam = n_spam *(4/9)/(5/9)
print("spam observations"); print(n_spam); print("no-spam observations"); print(n_no_spam)
```

```
## [1] "spam observations"
```

```
## [1] 1800
```

```
## [1] "no-spam observations"
```

```
## [1] 1440
```

We will work with:

3240 initial data split into train (2/3) -> 2160 and test (1/3) -> 1080 spam_train (2/3) -> 1440 and nospam_train (1/3) -> 720 spam_test (1/3) -> 360 and nospam_test (2/3) -> 720

Total spam data (5/9) -> 1800 Total nospam data (4/9) -> 1440

We select these data randomly and make the splits. Randomness is only performed in the first pick.

```
set.seed(123)
no_spam <- no_spam_all[sample(nrow(no_spam_all), n_no_spam), ]
set.seed(321)
spam <- spam_all[sample(nrow(spam_all), n_spam), ]

midpoint_nospam <- floor(nrow(no_spam) / 2)
no_spam_train <- no_spam[1:midpoint_nospam, ]; no_spam_test <- no_spam[(midpoint_nospam + 1):nrow(no_spa

partition_spam <- floor(nrow(spam) *8 / 10)
spam_train <- spam[1:partition_spam, ]; spam_test <- spam[(partition_spam + 1):nrow(spam), ]



df_tr <- rbind.data.frame(no_spam_train, spam_train)

n<-dim(df_tr)[1]
p<-dim(df_tr)[2]-1 # Removes last column (response variable)
df_tr.01 <- df_tr[,p+1] # That is, the binary response variable
df_tr.vars <- as.matrix(df_tr[,1:p]) # That is X (n x p)
print("n_train = ");print(n);
```

```
## [1] "n_train = "
```

```
## [1] 2160
```

```
print("p_train = ");print(p);
```

```
## [1] "p_train = "
```

```
## [1] 57
```

```
print("Proportion of spam e-mails = "); print(round(mean(df_tr.01),4))
```

```
## [1] "Proportion of spam e-mails = "
```

```
## [1] 0.6667
```

```
df_test <- rbind.data.frame(no_spam_test, spam_test)

ntest<-dim(df_test)[1]
ptest<-dim(df_test)[2]-1 # Removes last column (response variable)
df_test.01 <- df_test[,p+1] # That is, the binary response variable
df_test.vars <- as.matrix(df_test[,1:p]) # That is X (n x p)
print("n_test = ");print(ntest);
```

```
## [1] "n_test = "
```

```
## [1] 1080
```

```
print("p_test = ");print(ptest);
```

```
## [1] "p_test = "
```

```
## [1] 57
```

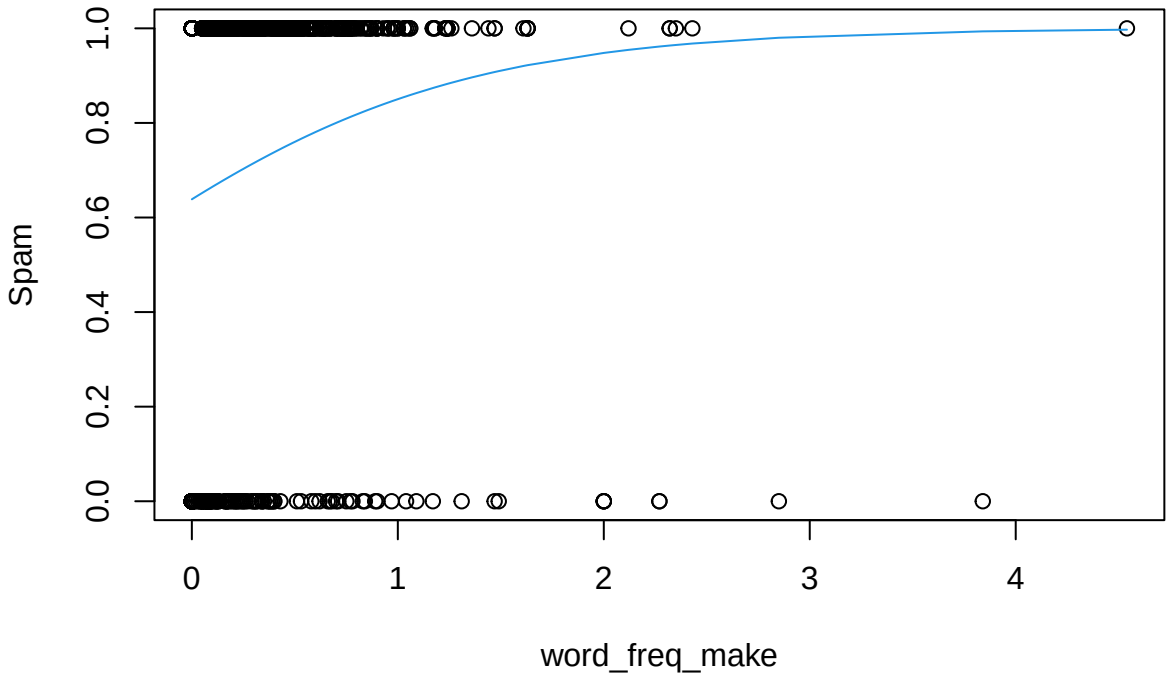
```
print("Proportion of spam e-mails = "); print(round(mean(df_test.01),4))
```

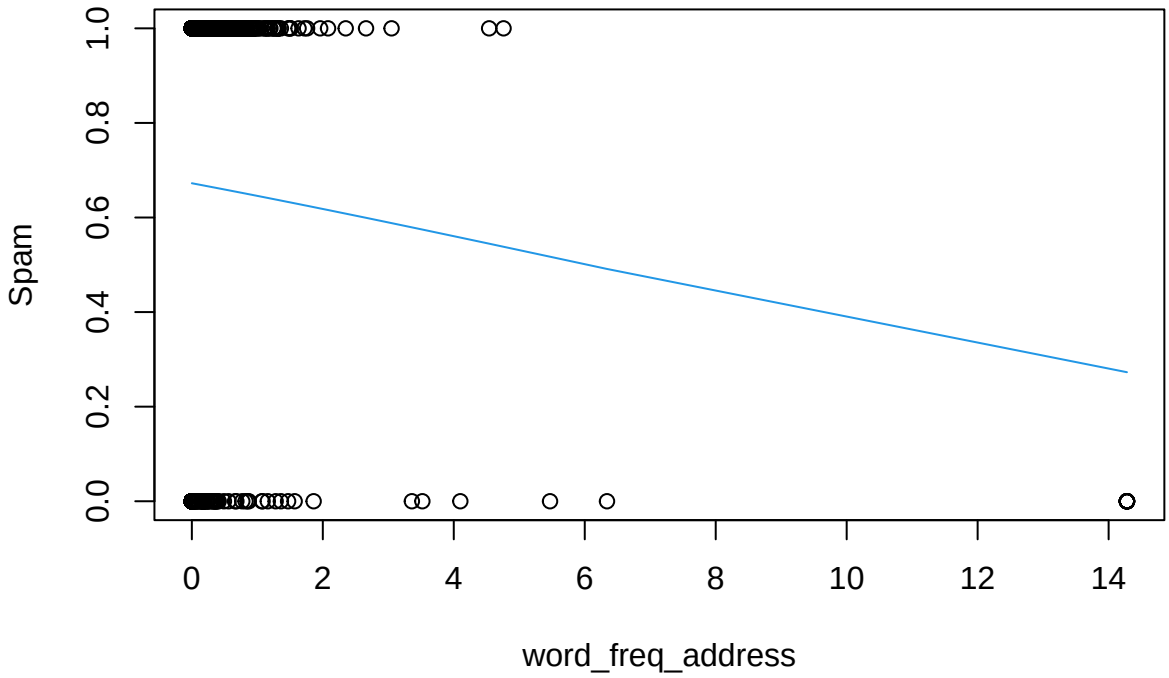
```
## [1] "Proportion of spam e-mails = "
```

```
## [1] 0.3333
```

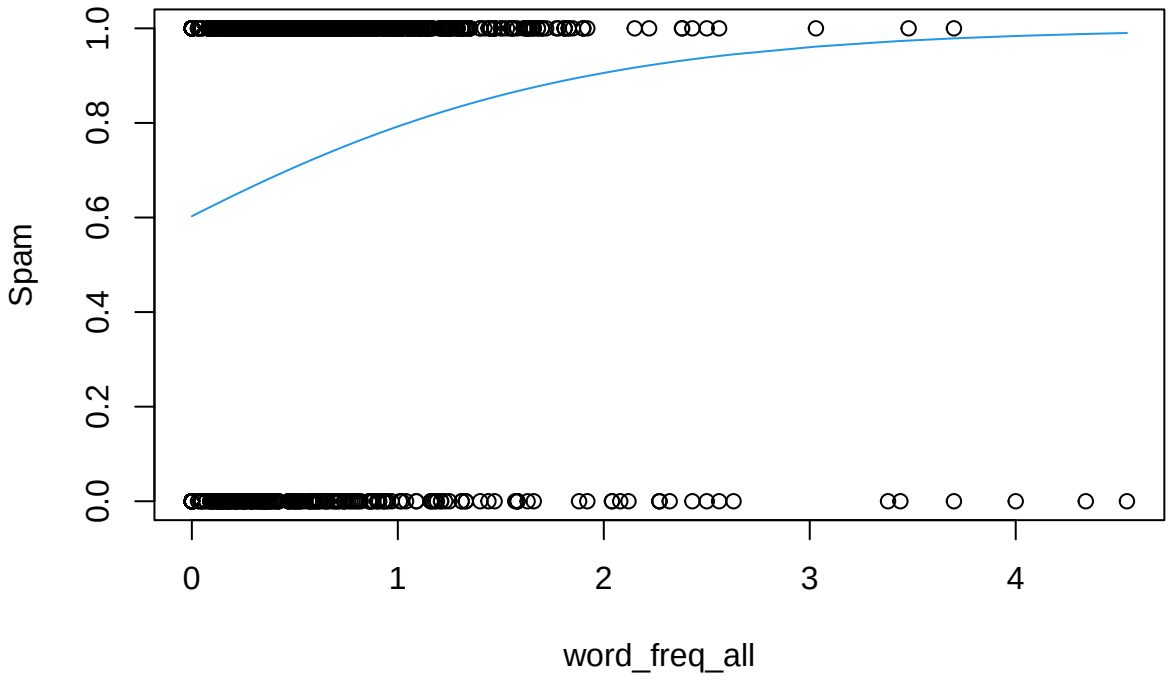
Logistic regression fitted by maximum likelihood (glm)

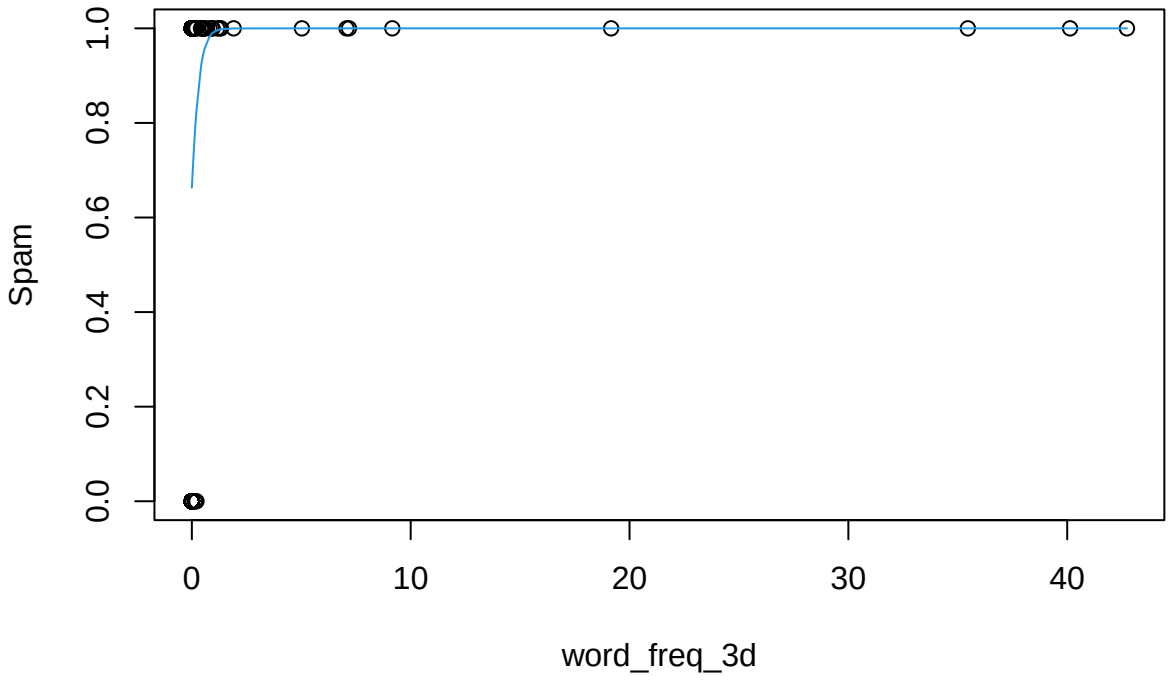
```
# One variable by one

for (j in 1:dim(df_tr.vars)[2]) {
  aux0 <- as.numeric(df_tr.vars[, j])
  perm <- orderv(list(aux0))
  aux1 <- aux0[perm]
  aux2 <- df_tr.01[perm]
  glm.df_tr <- glm(aux2 ~ aux1,family=binomial())
  # summary(glm.df_tr)
  plot(aux1,aux2,xlab = df.names[j], ylab="Spam")
  lines(aux1, glm.df_tr$fitted.values,col=4)

}
```

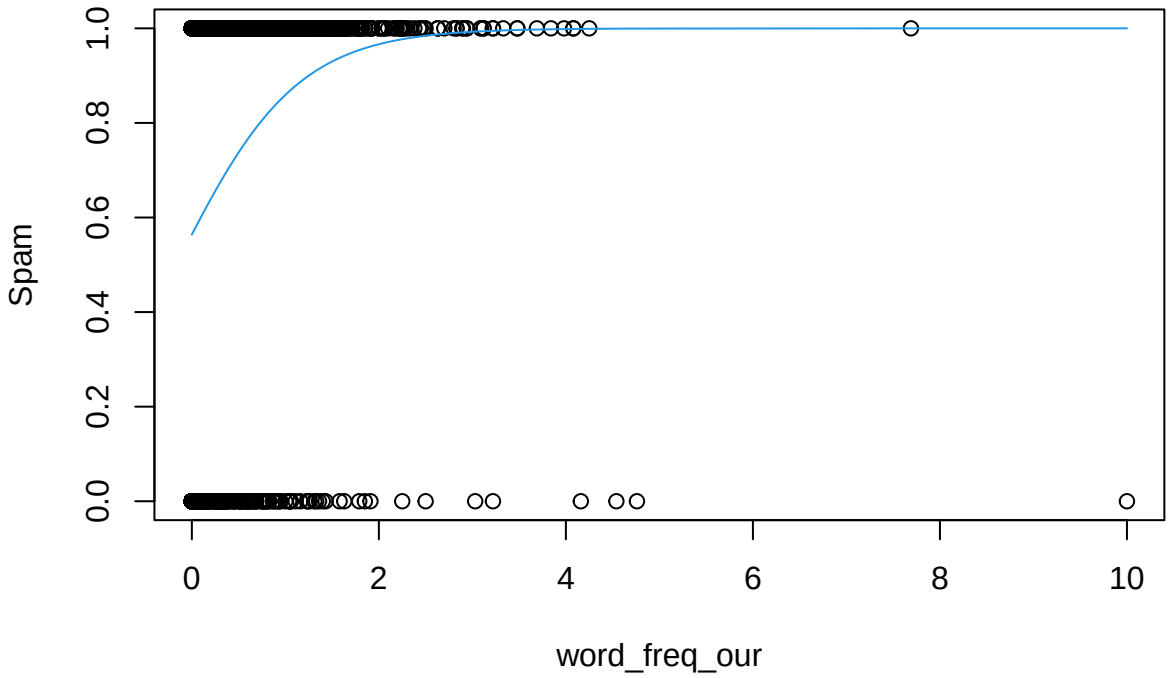

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
word_freq_make



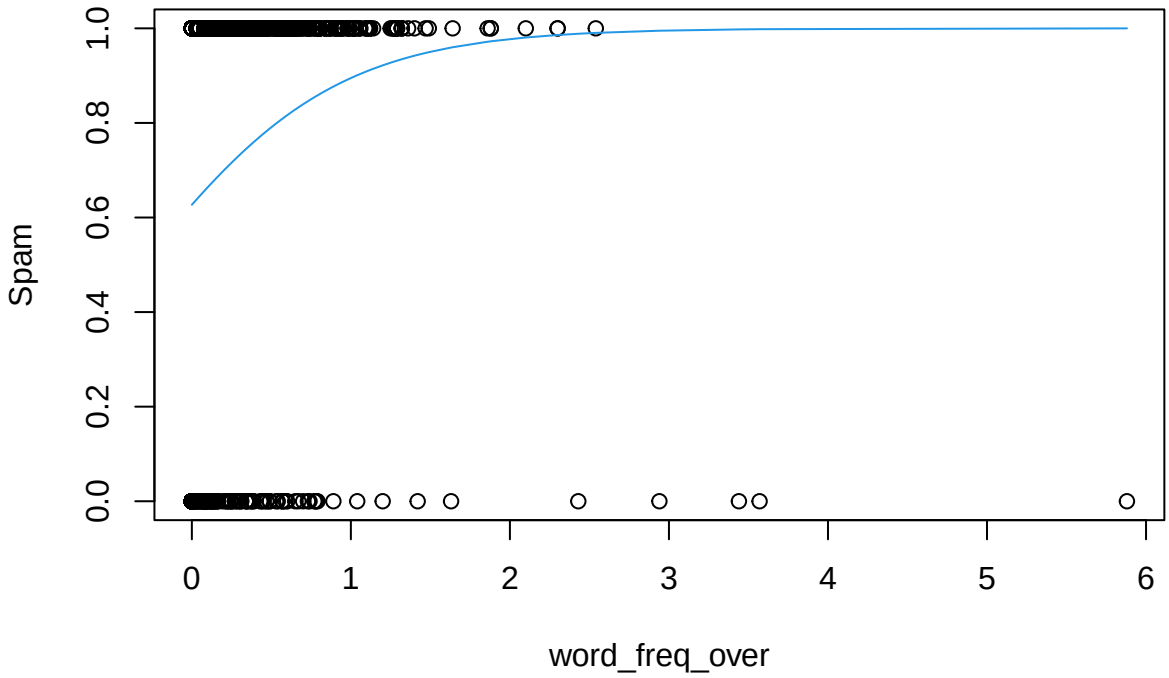

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

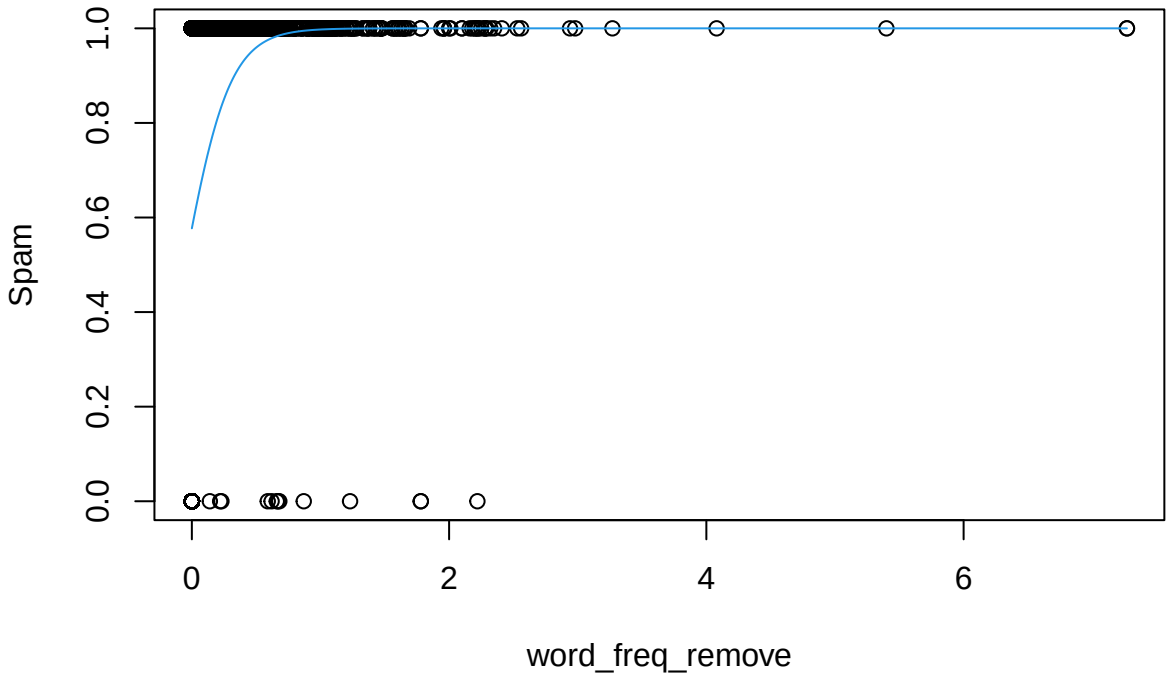

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
word_freq_all

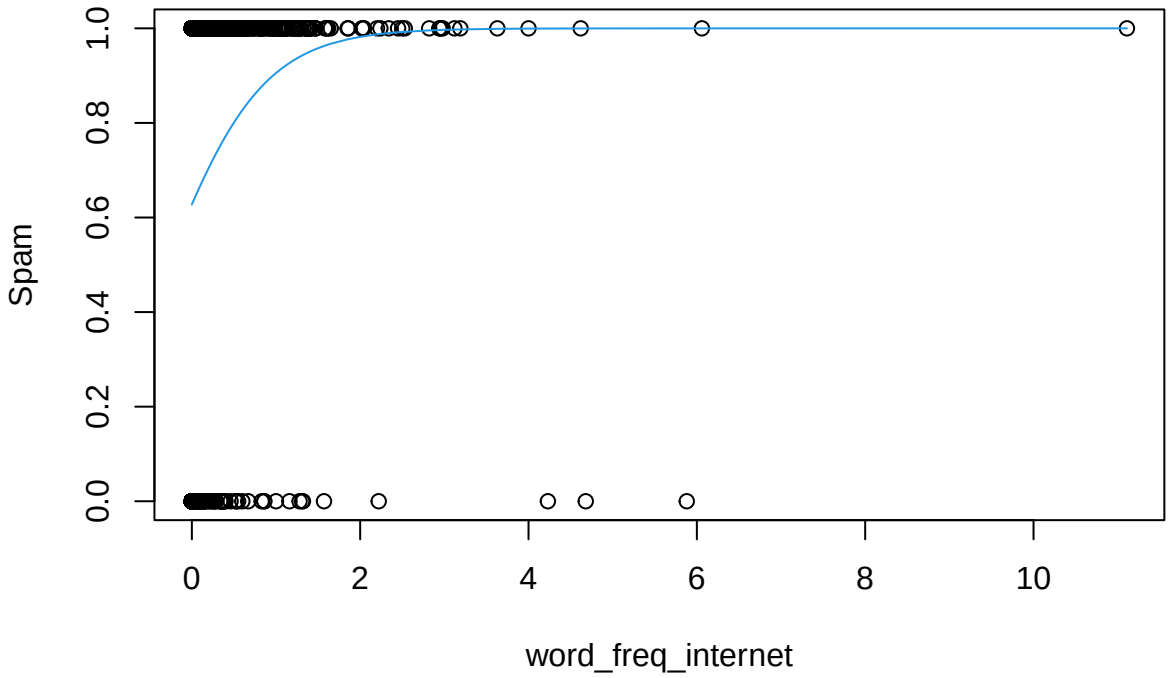

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
word_freq_3d

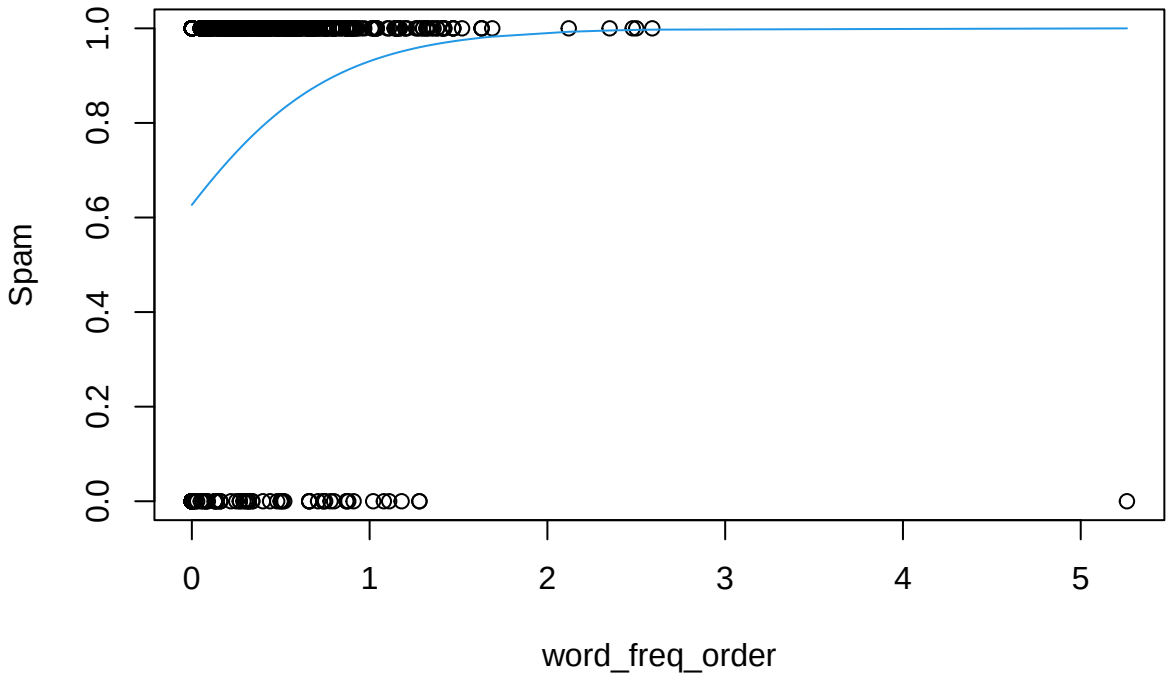


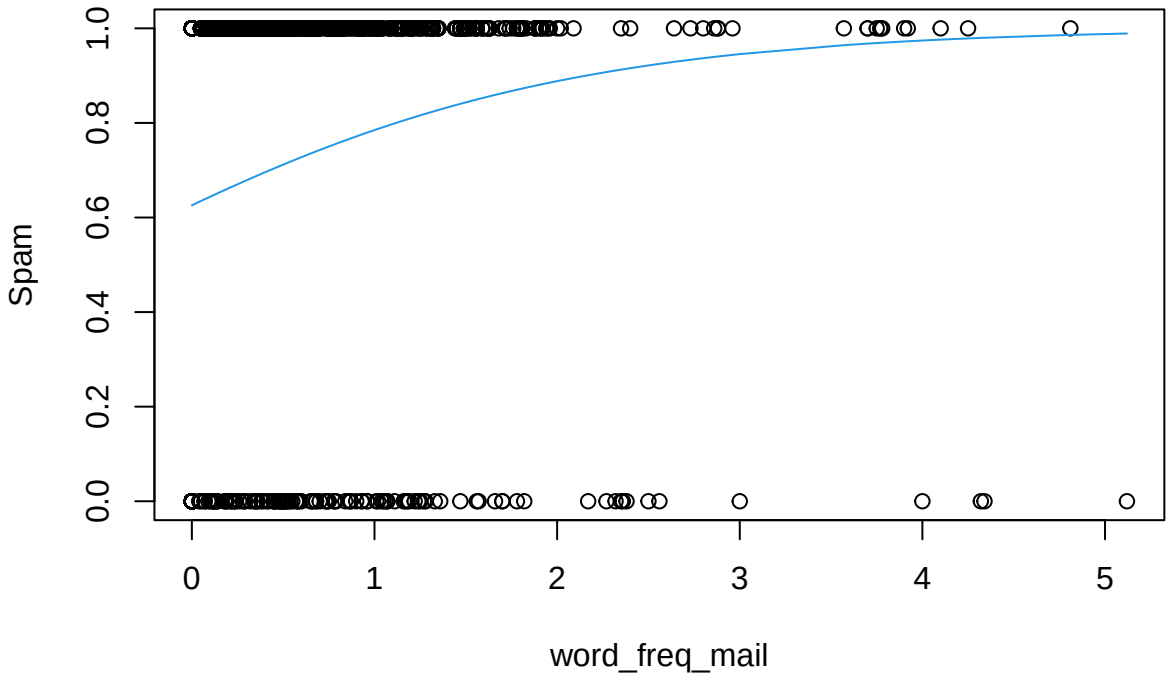

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

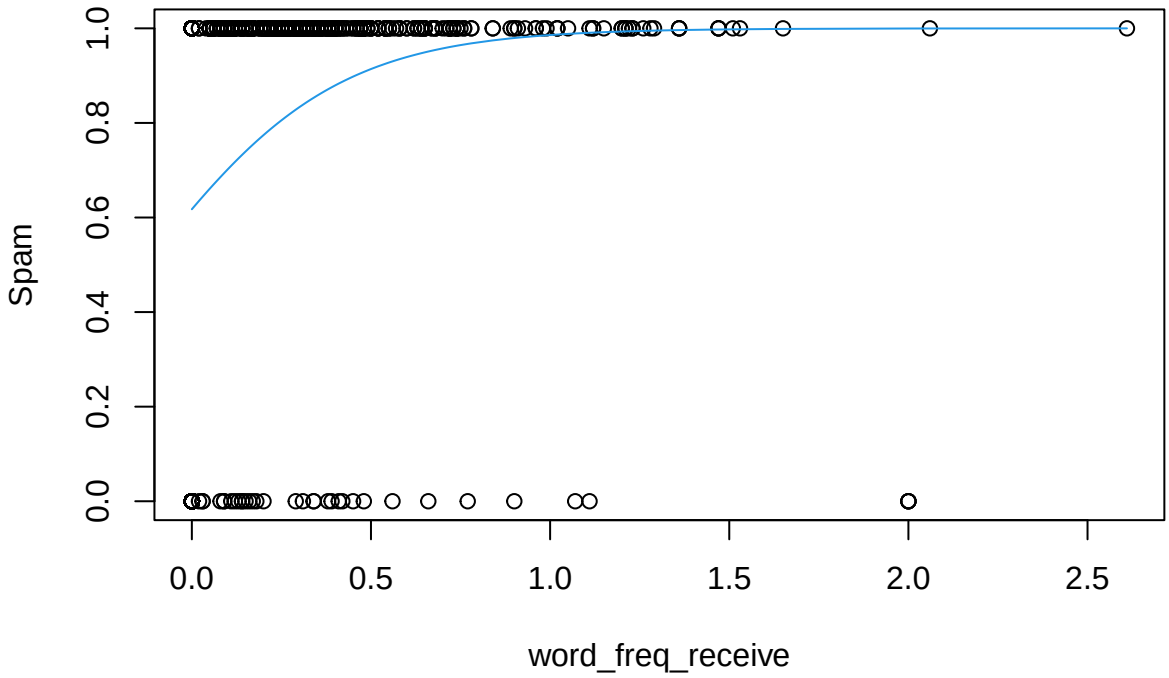

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
word_freq_over

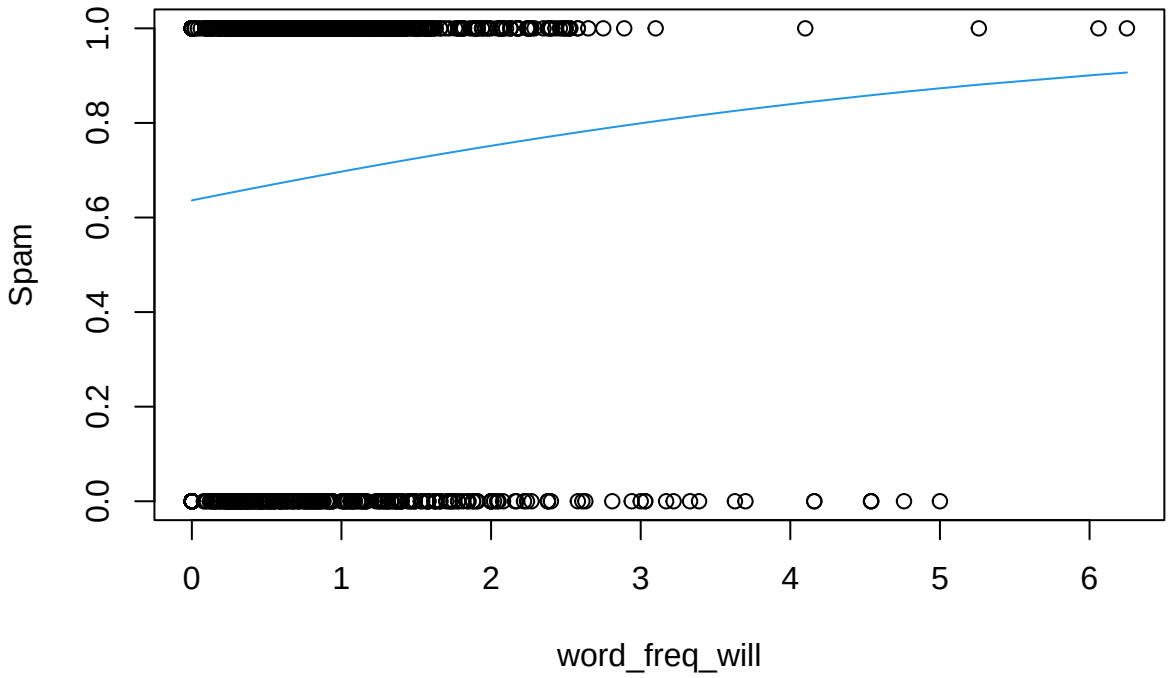

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
word_freq_remove

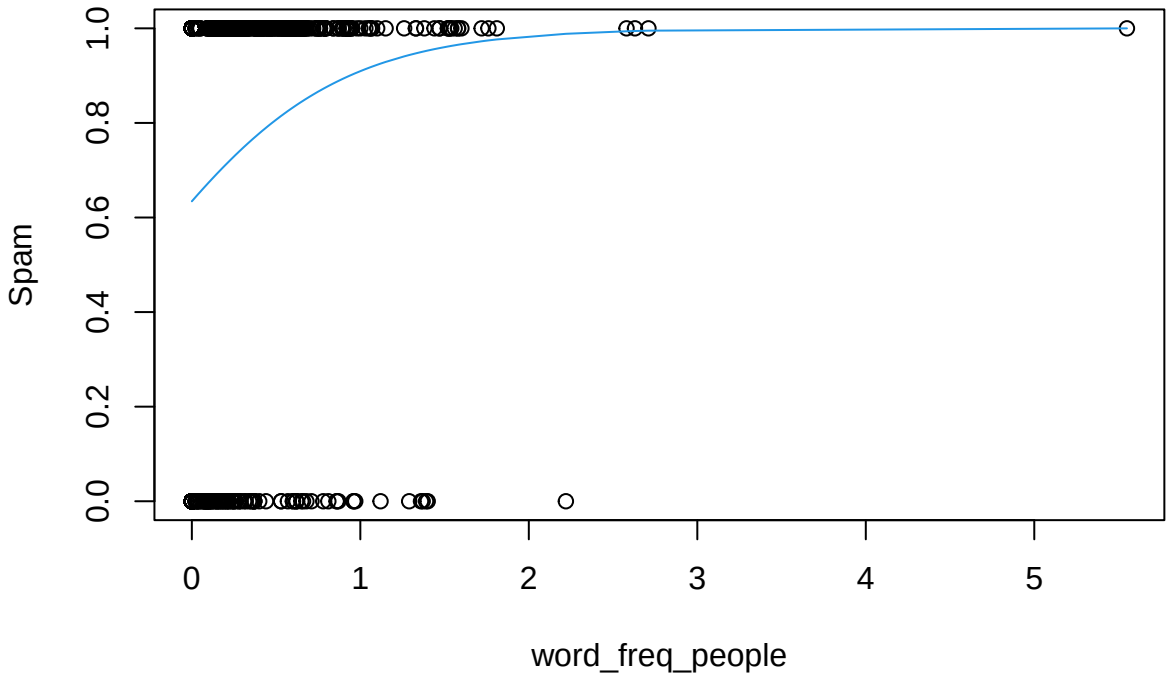

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
word_freq_internet

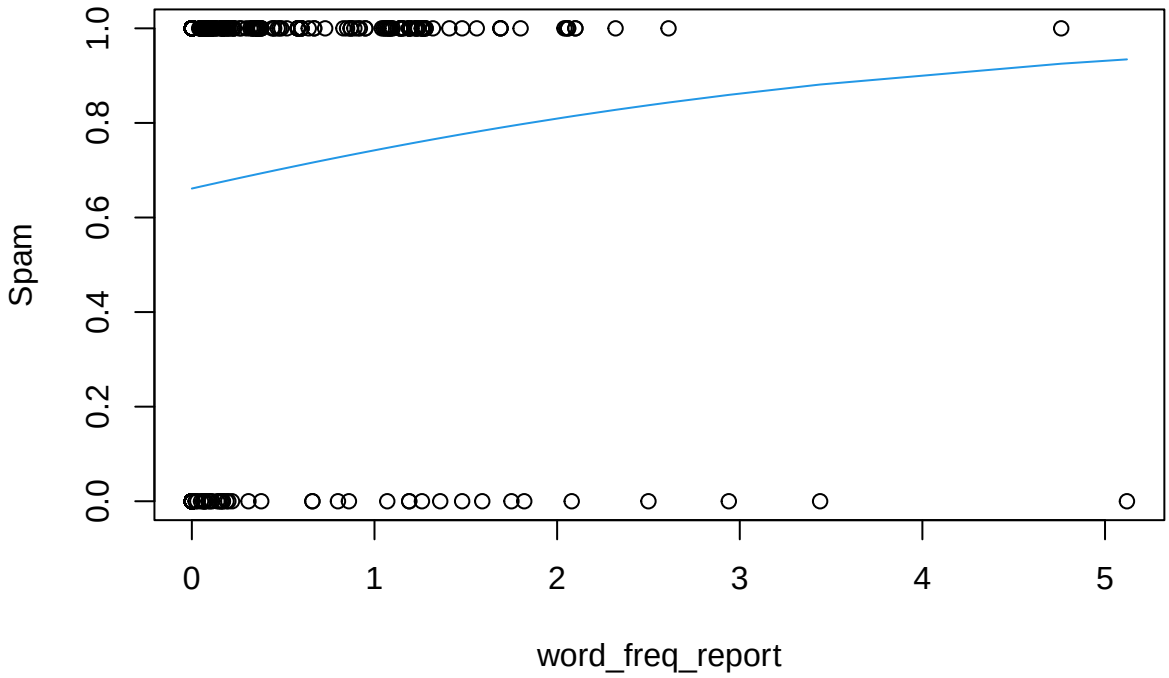

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
word_freq_order


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
word_freq_mail

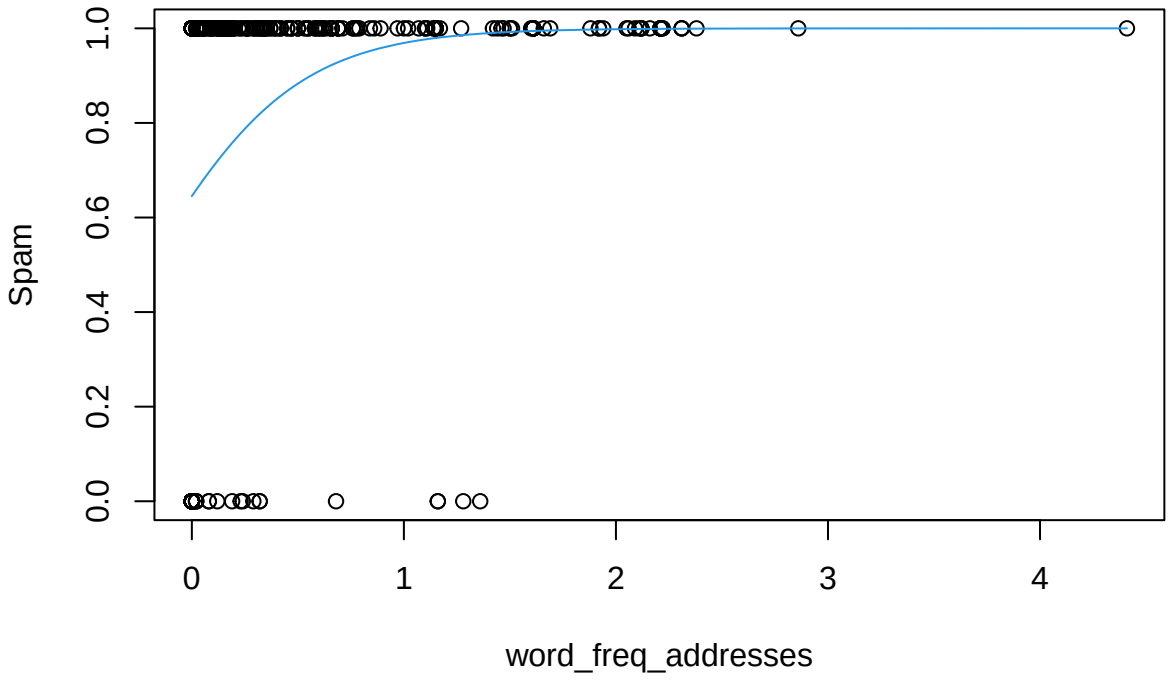

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
2.5
word_freq_receive

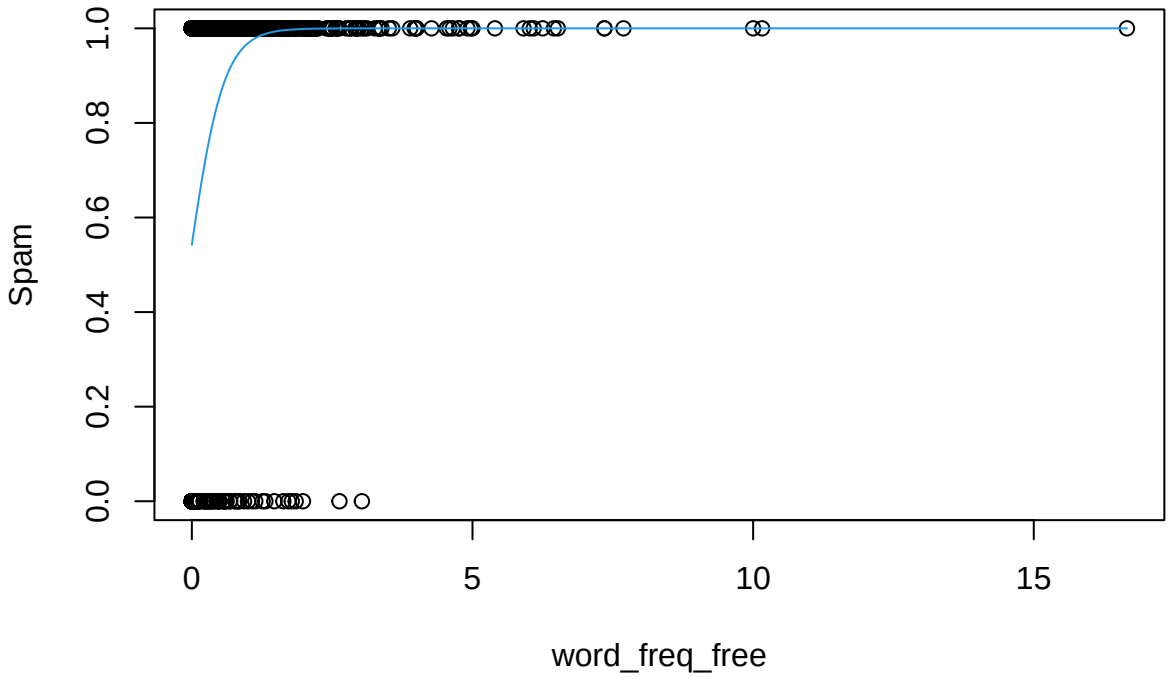

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
word_freq_will

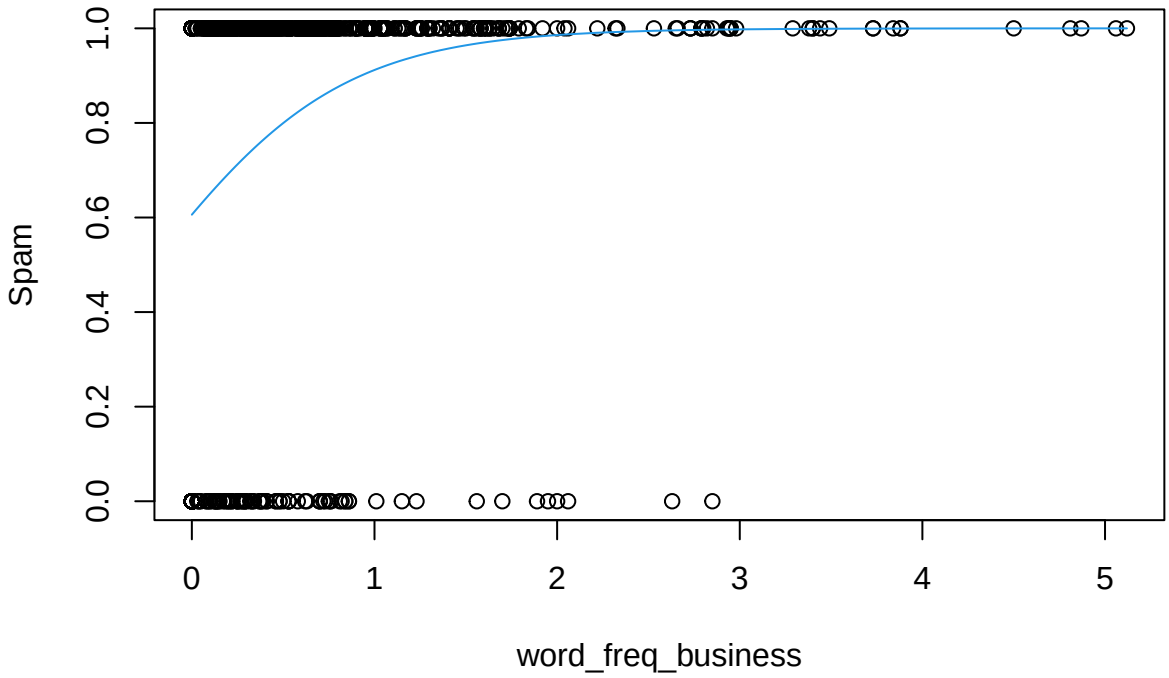

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
word_freq_people

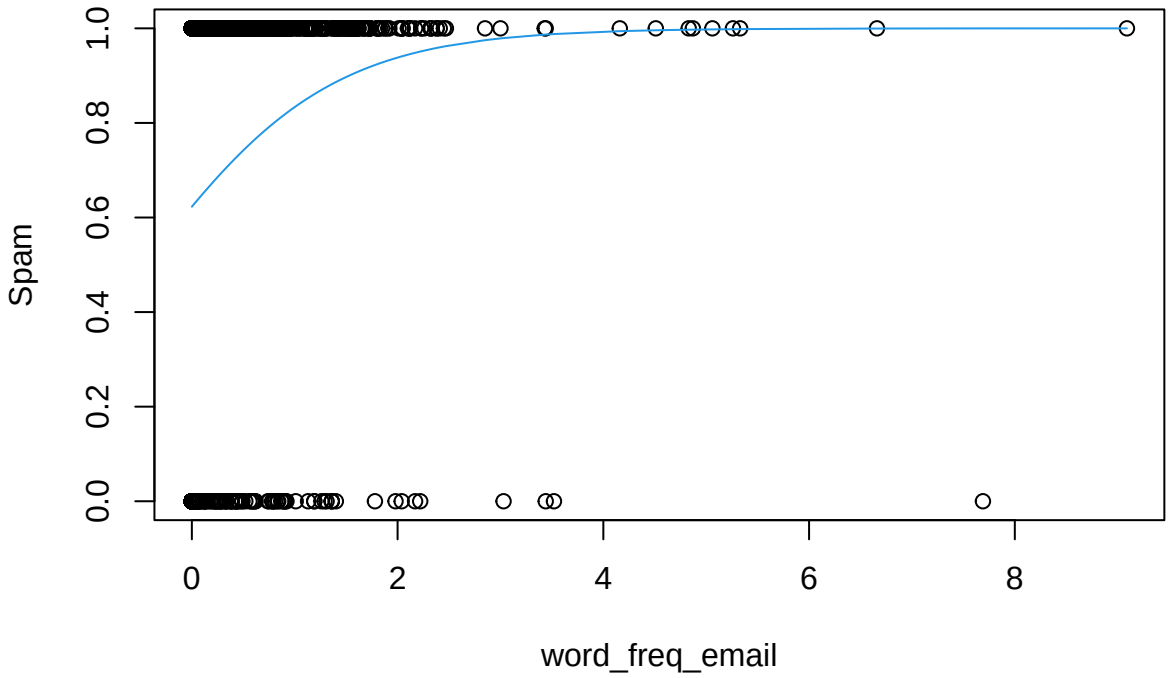



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

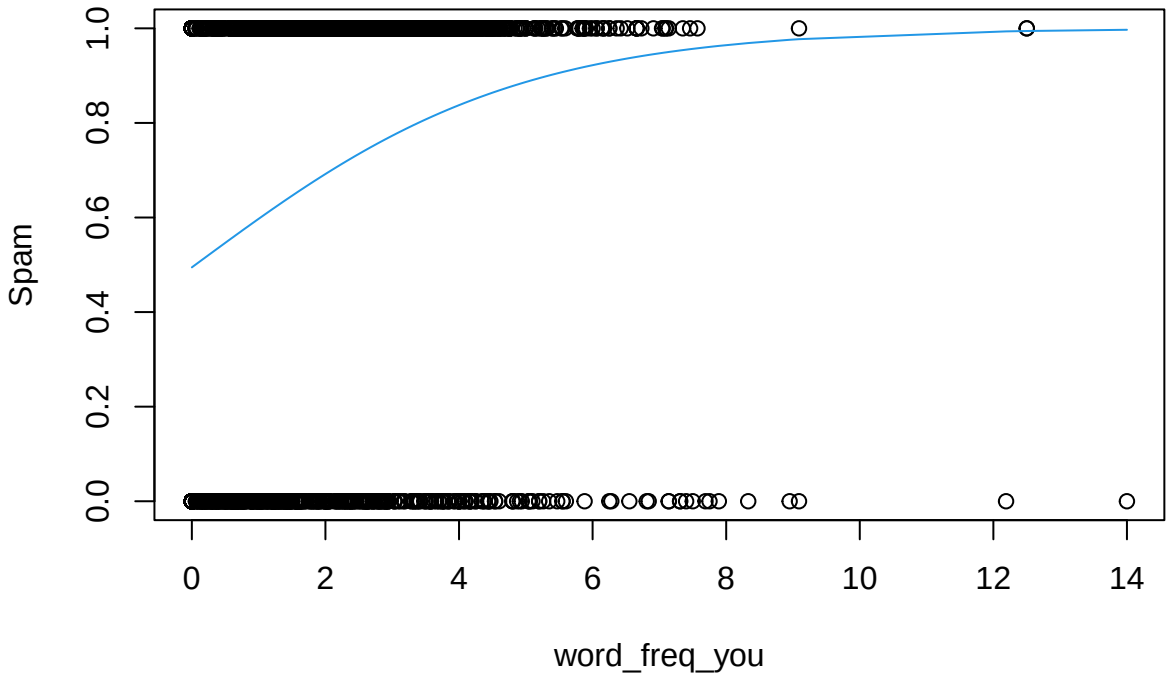

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
word_freq_addresses

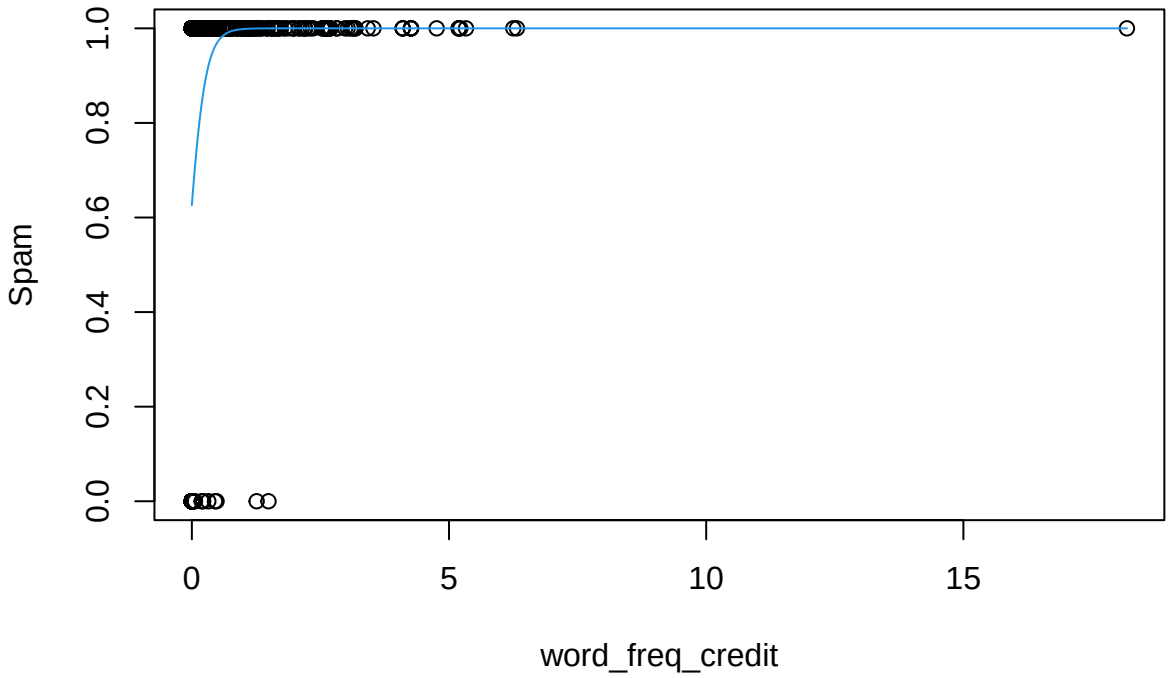

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
word_freq_free

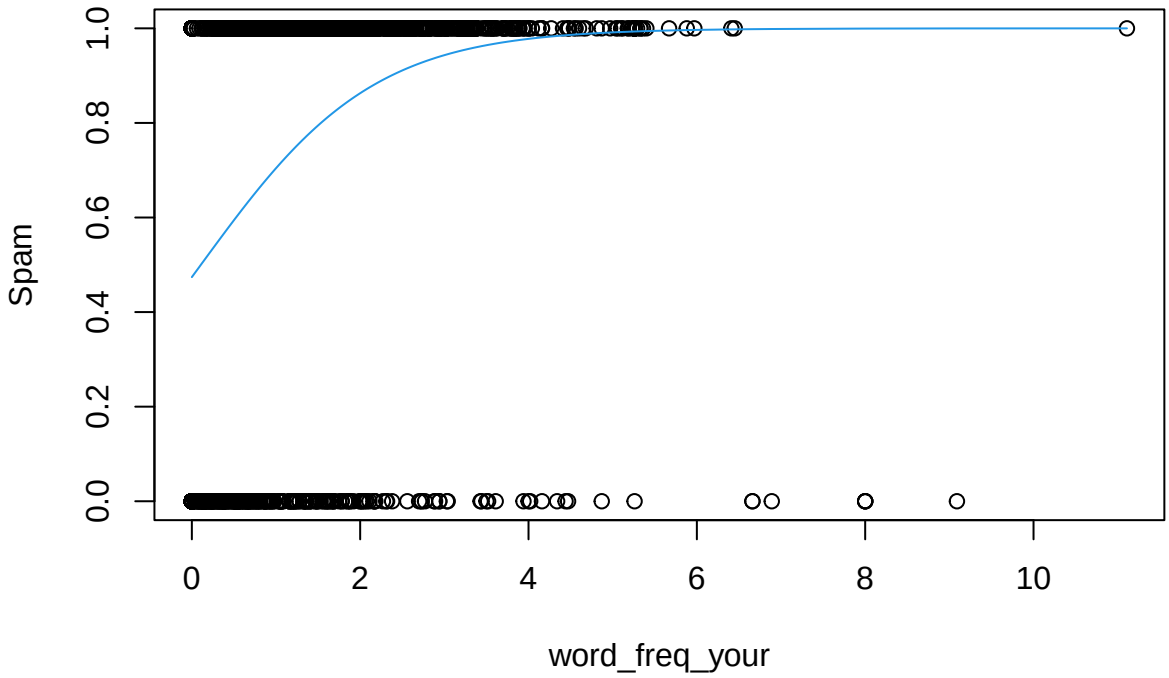

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
word_freq_business



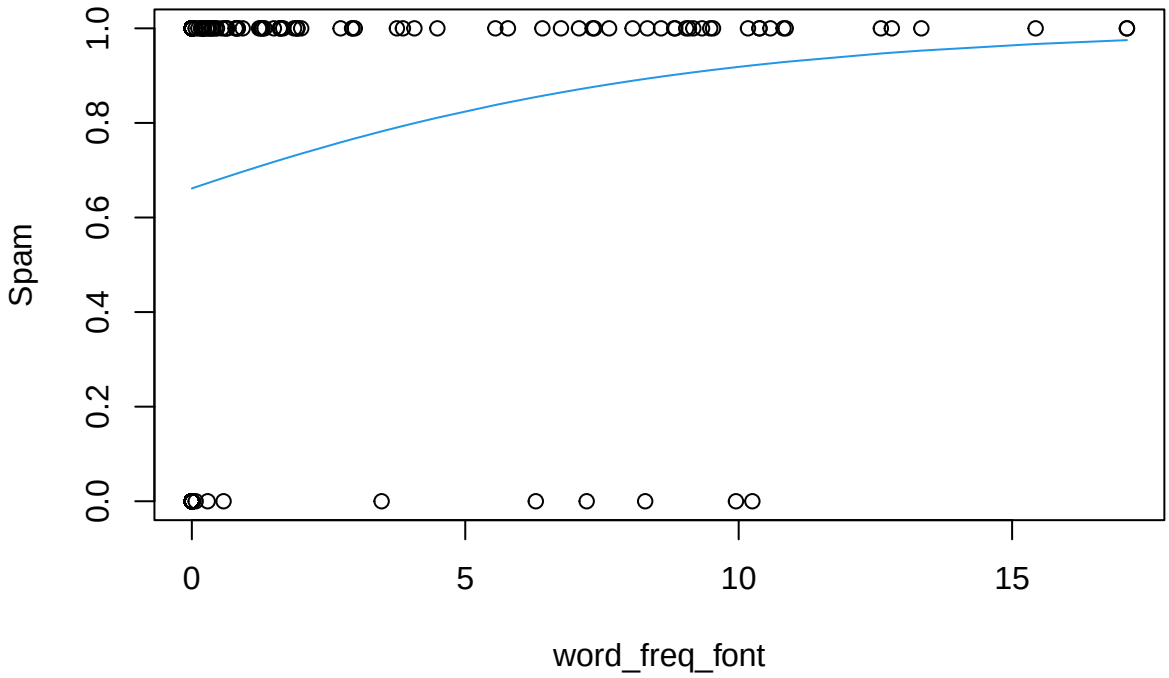

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
14
word_freq_you

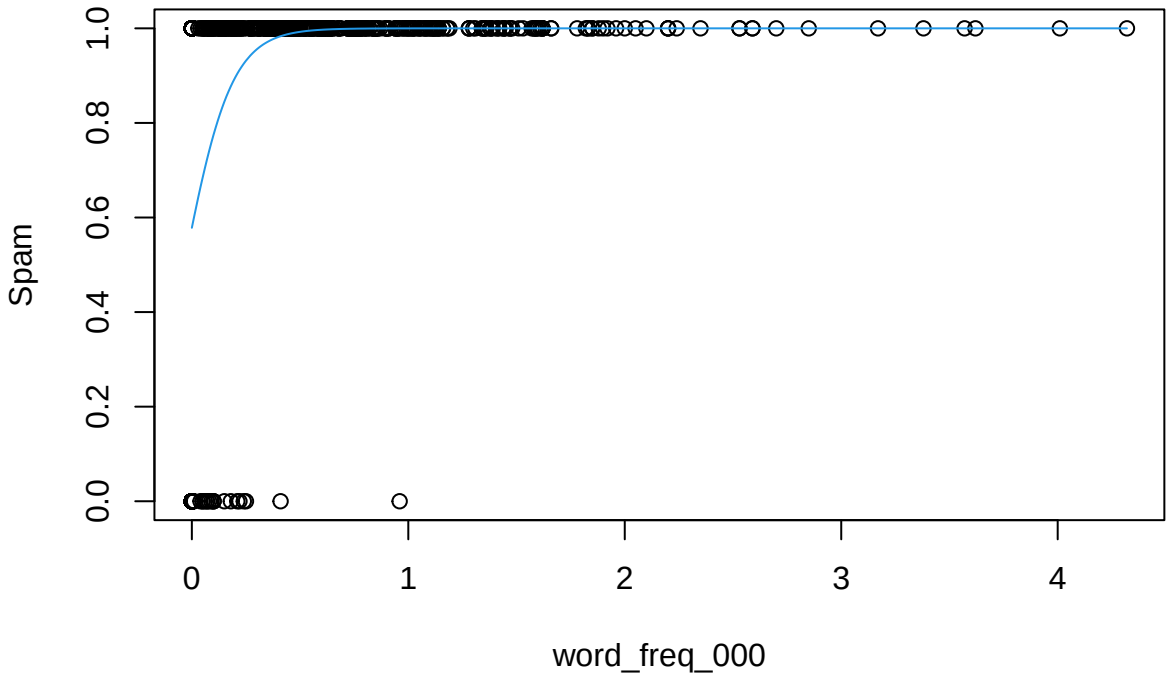

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
word_freq_credit



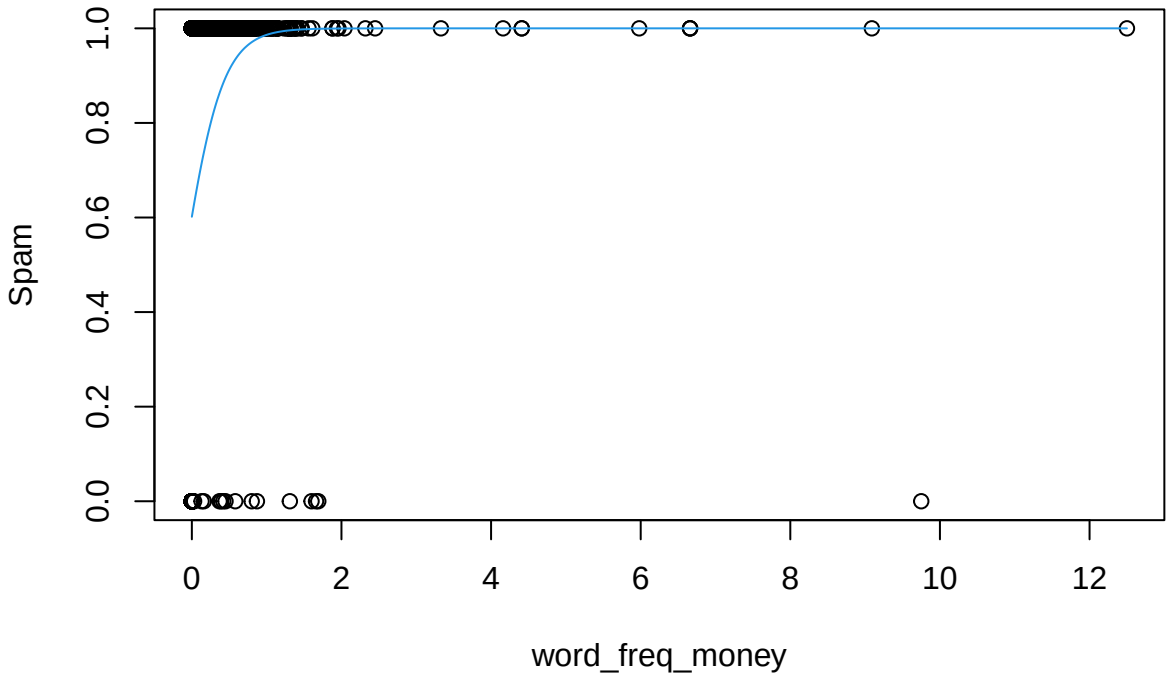

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



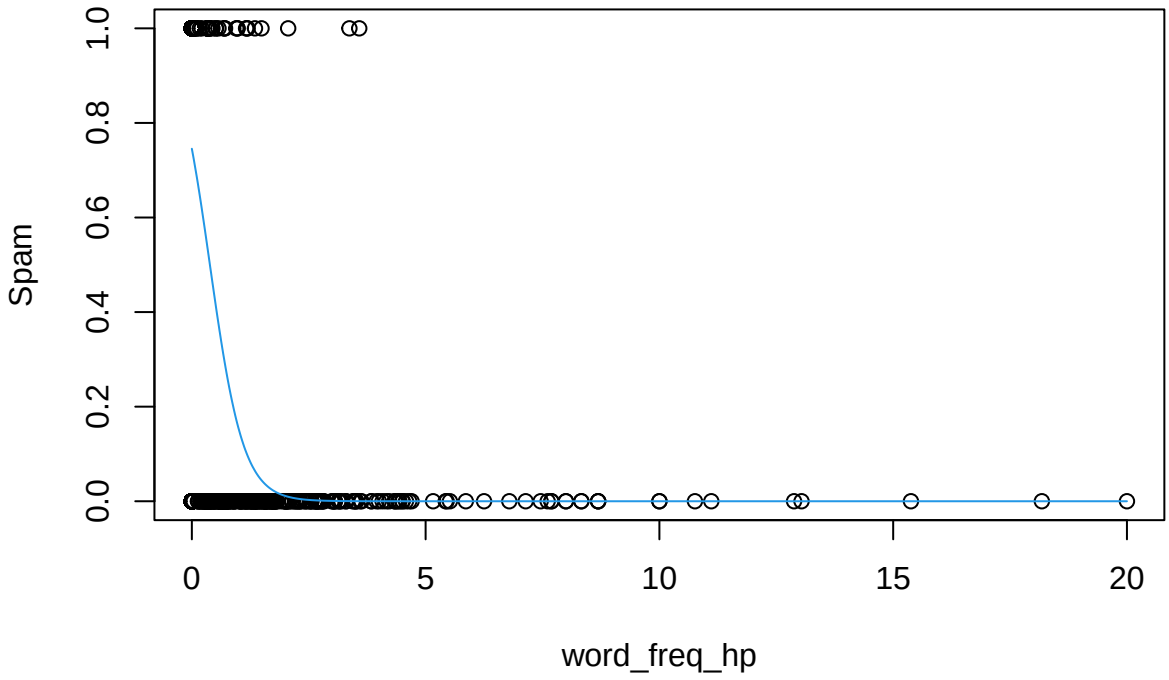

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



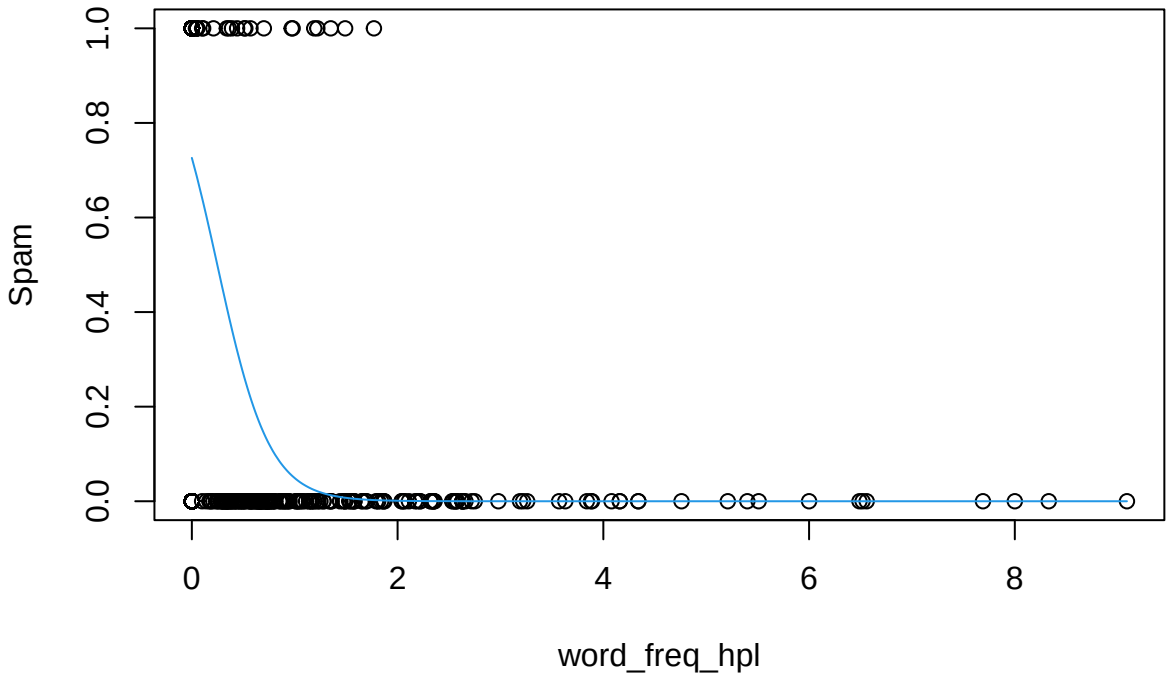

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

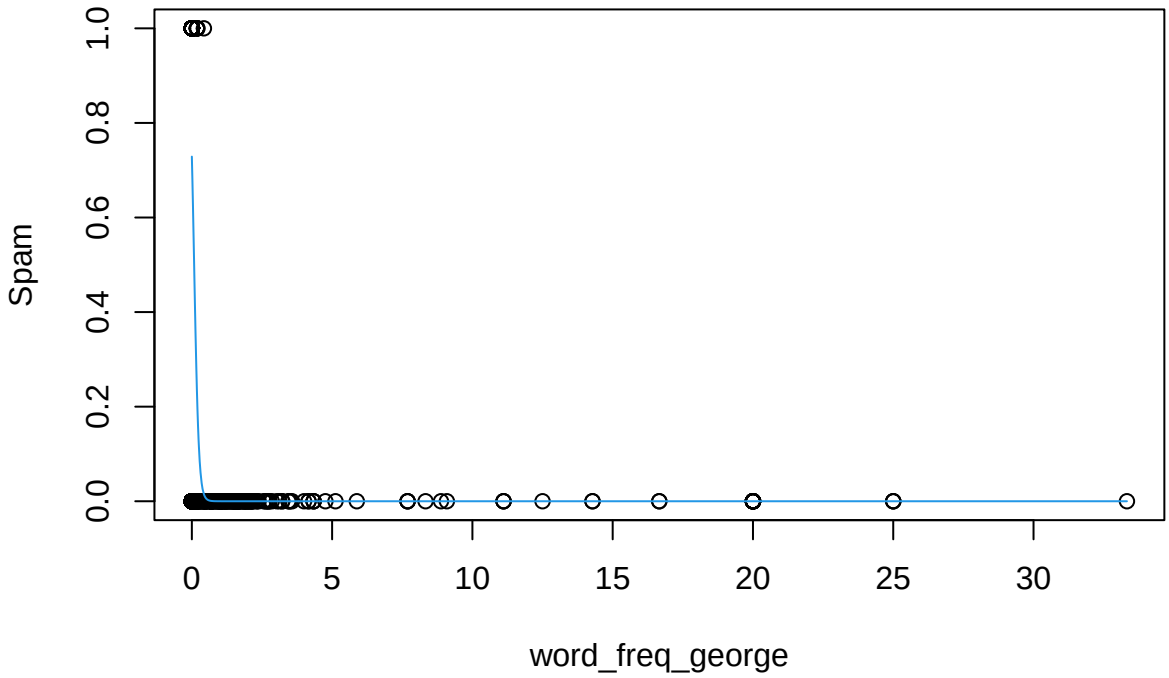



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

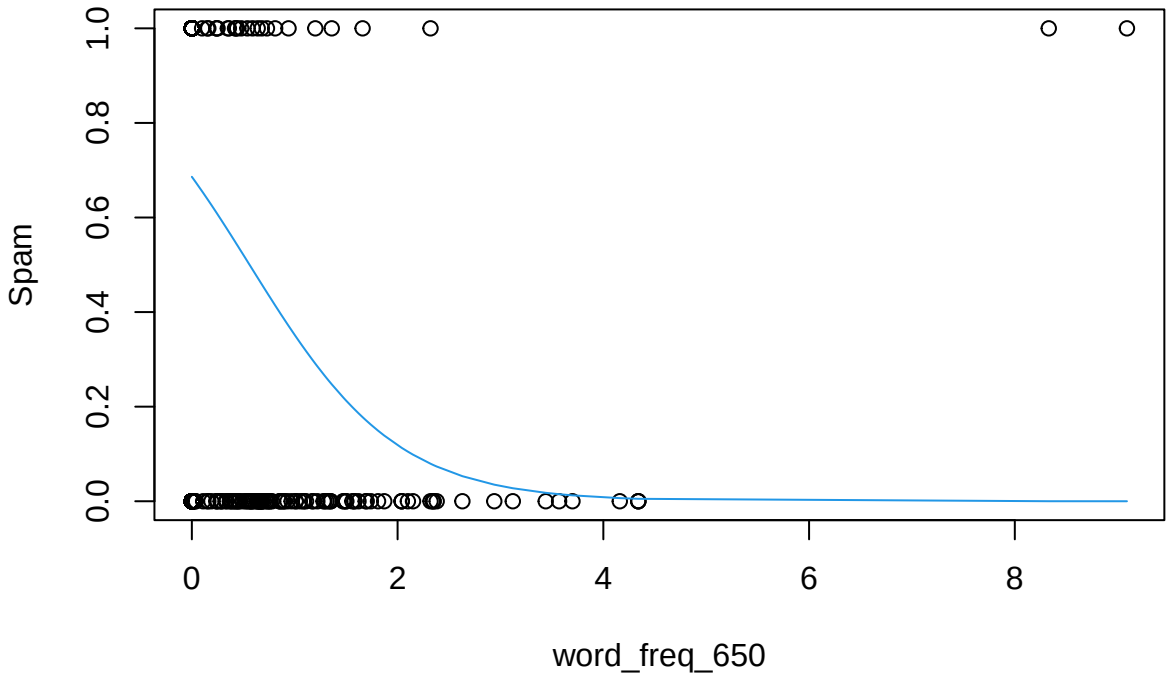


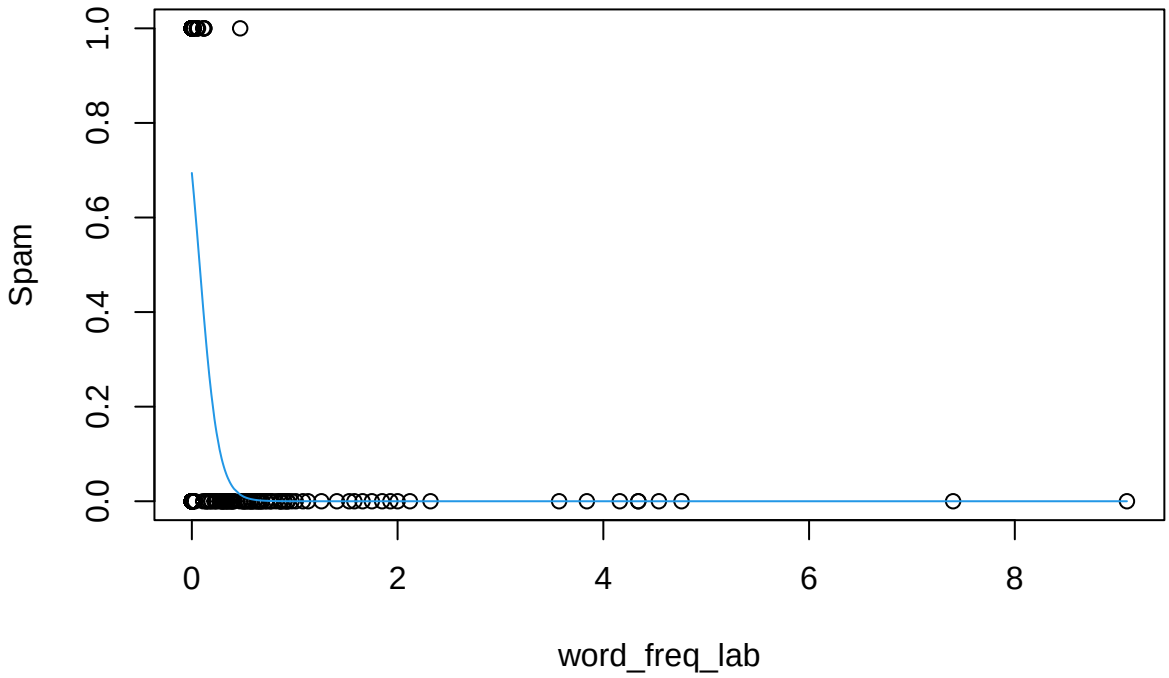

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
word_freq_hpl

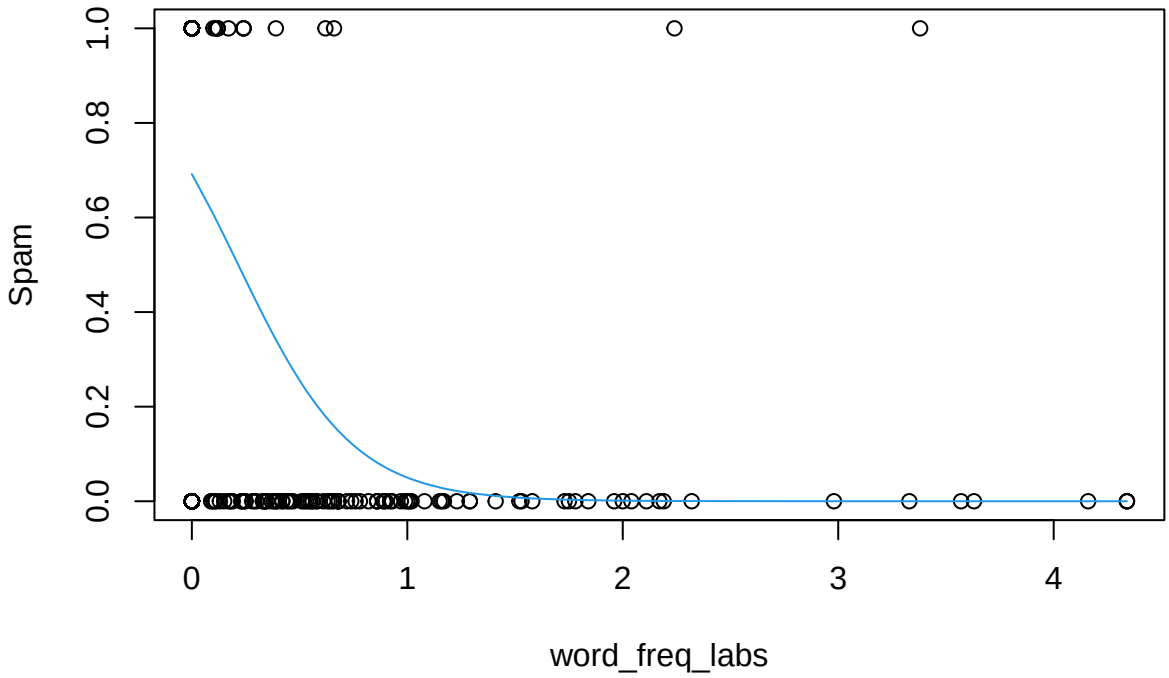



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

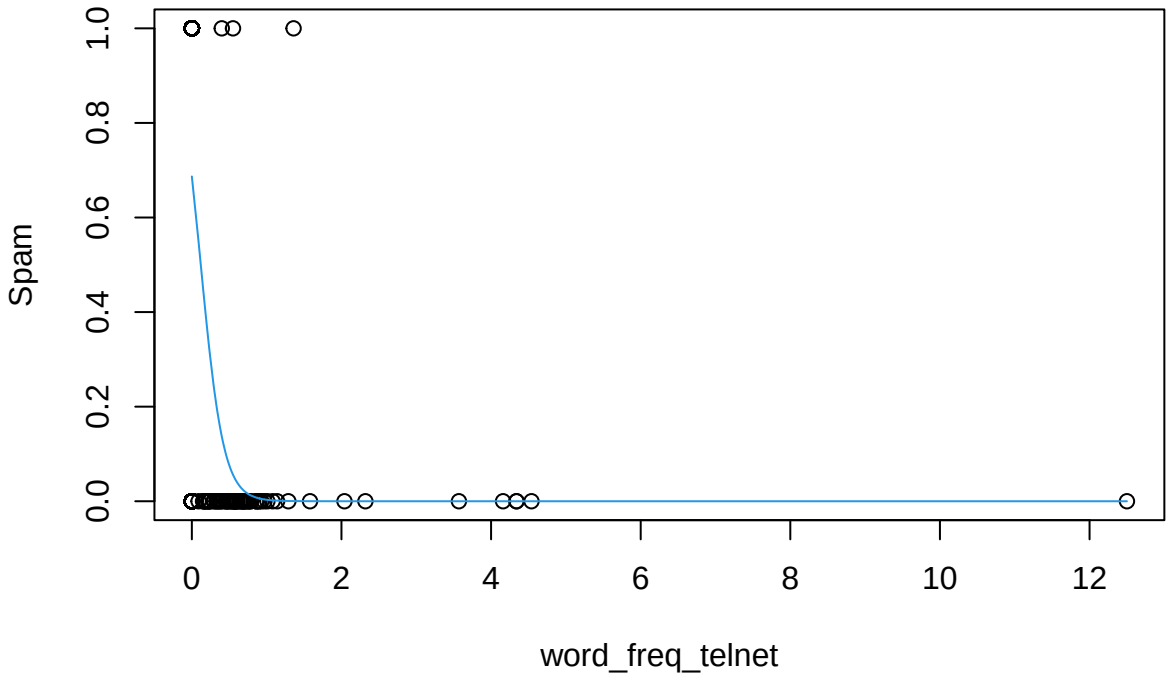

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
word_freq_650



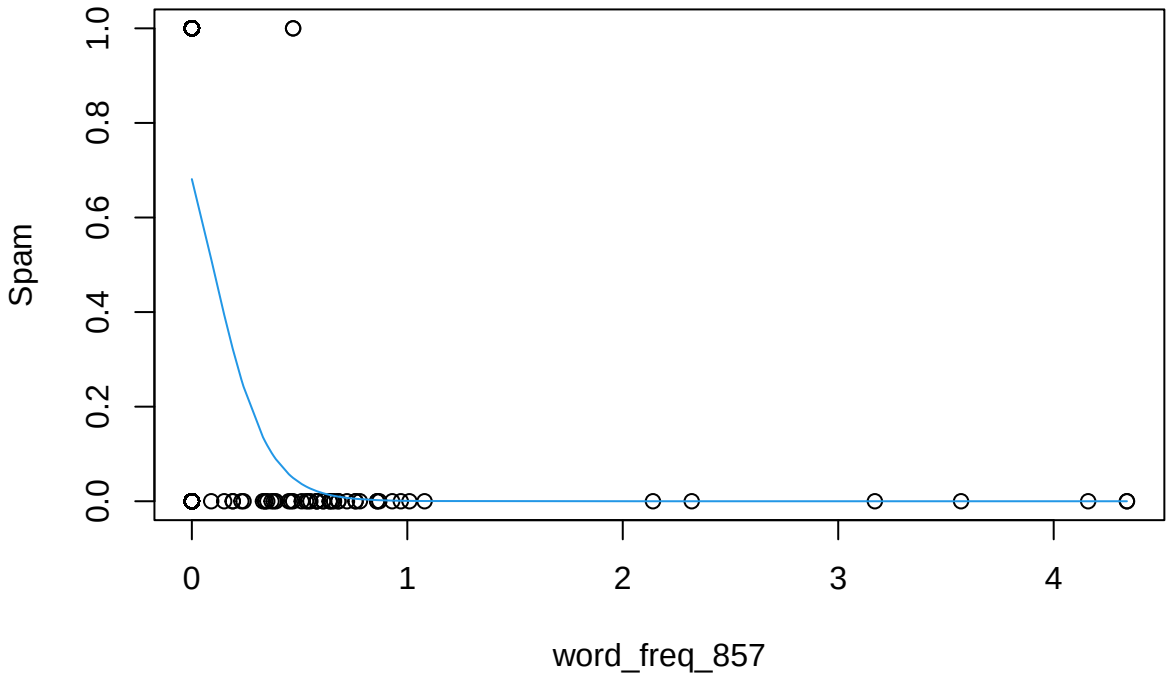

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



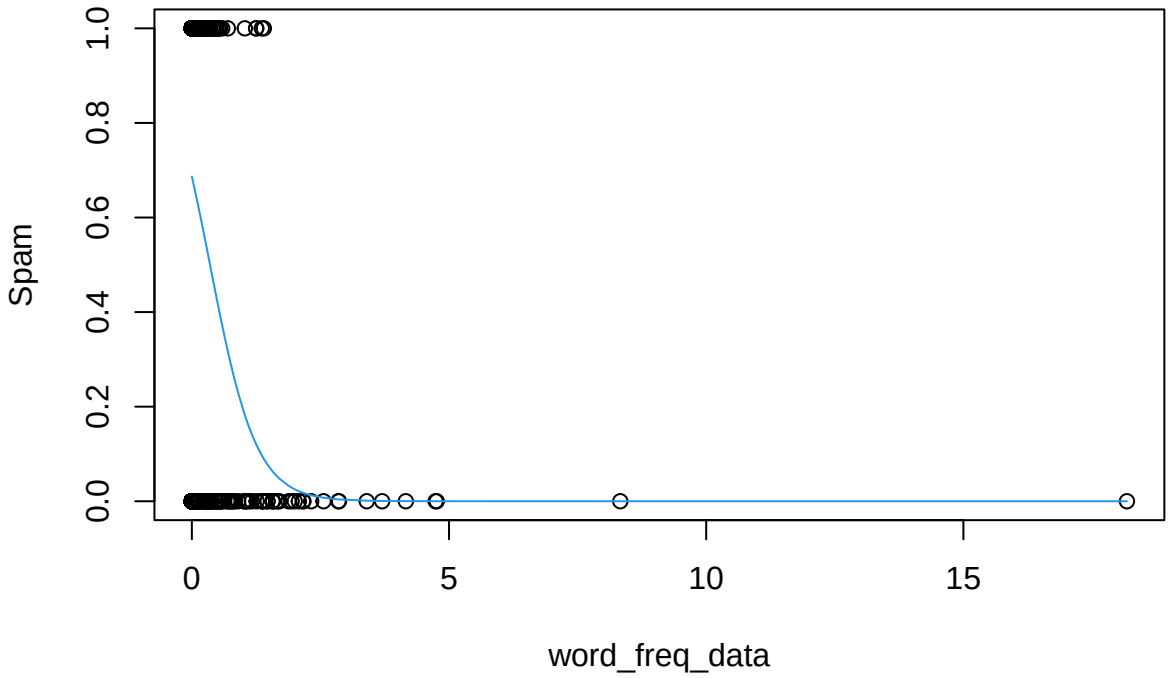

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

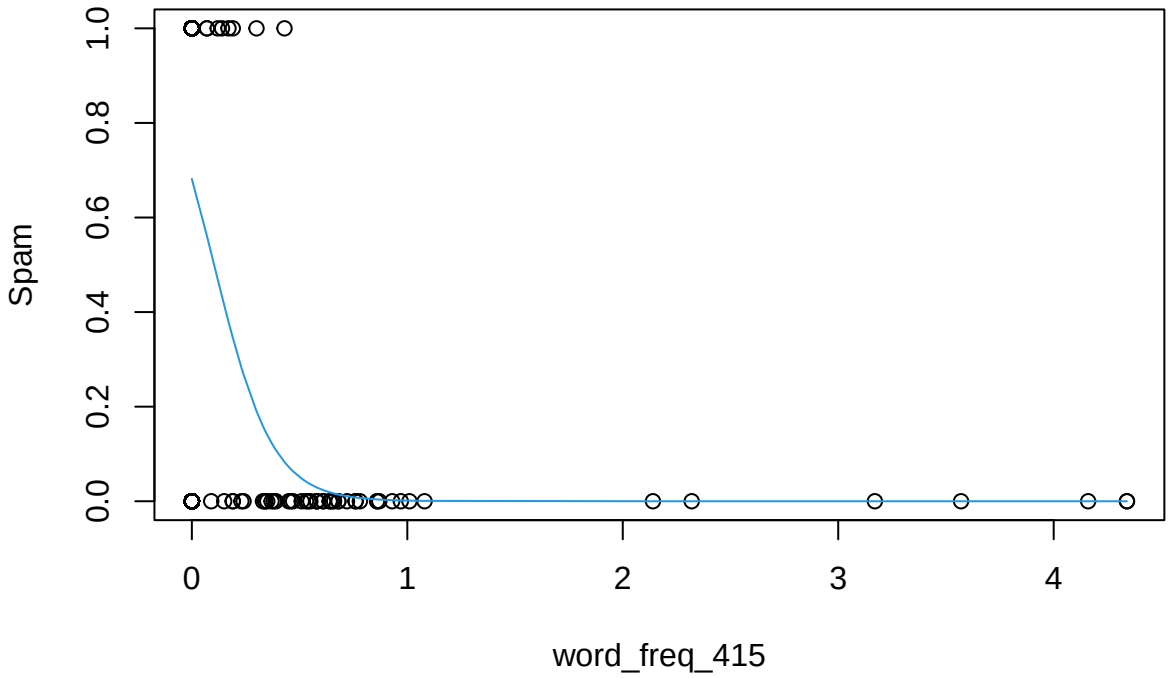


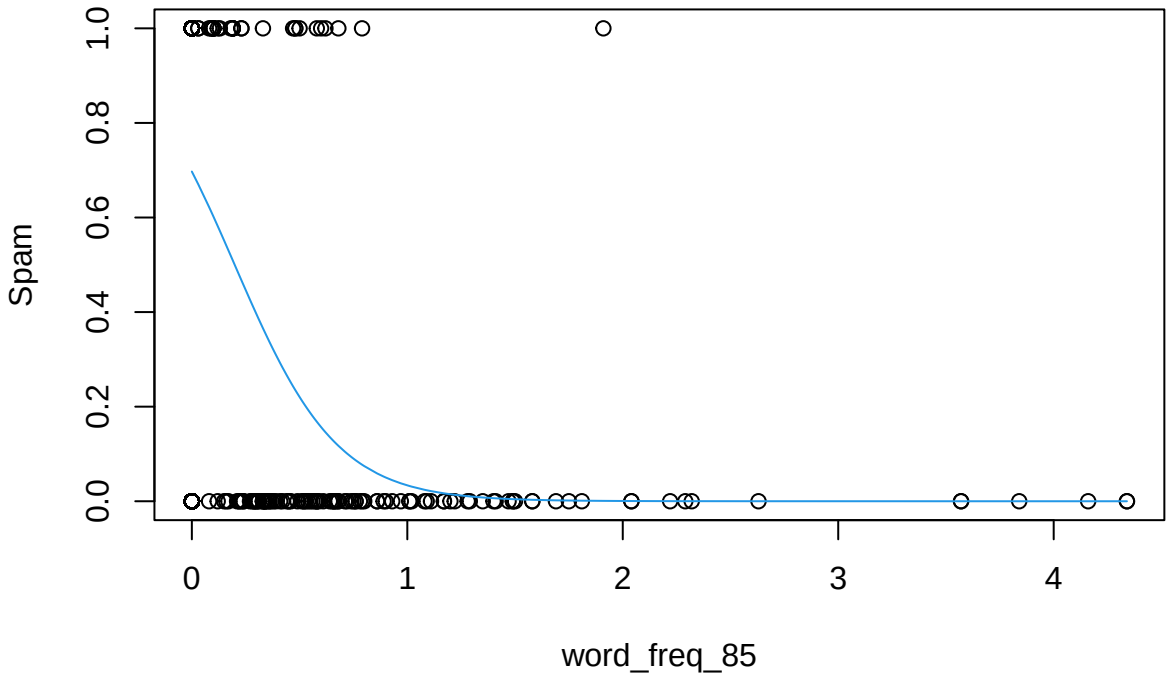

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

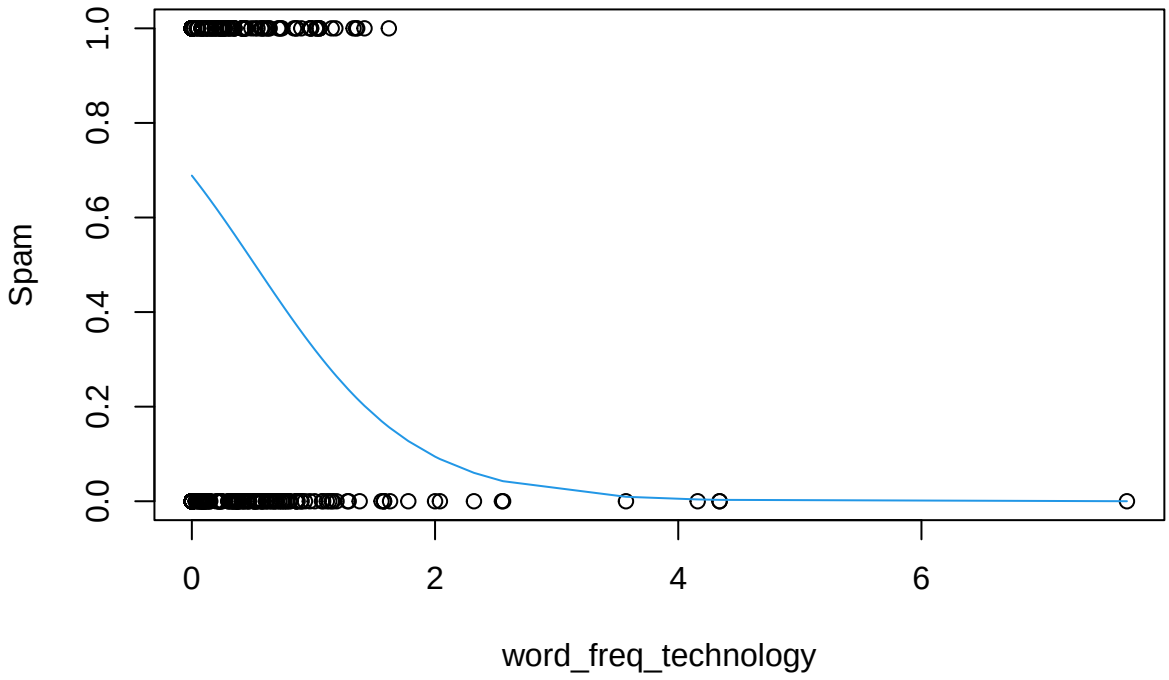


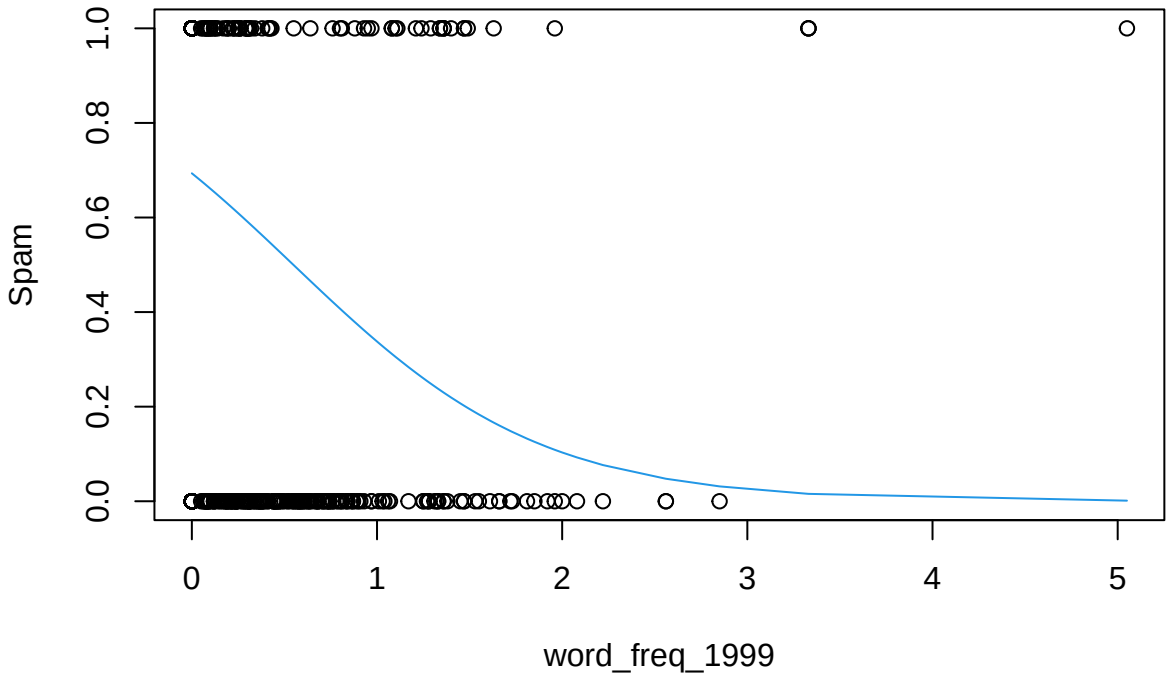

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

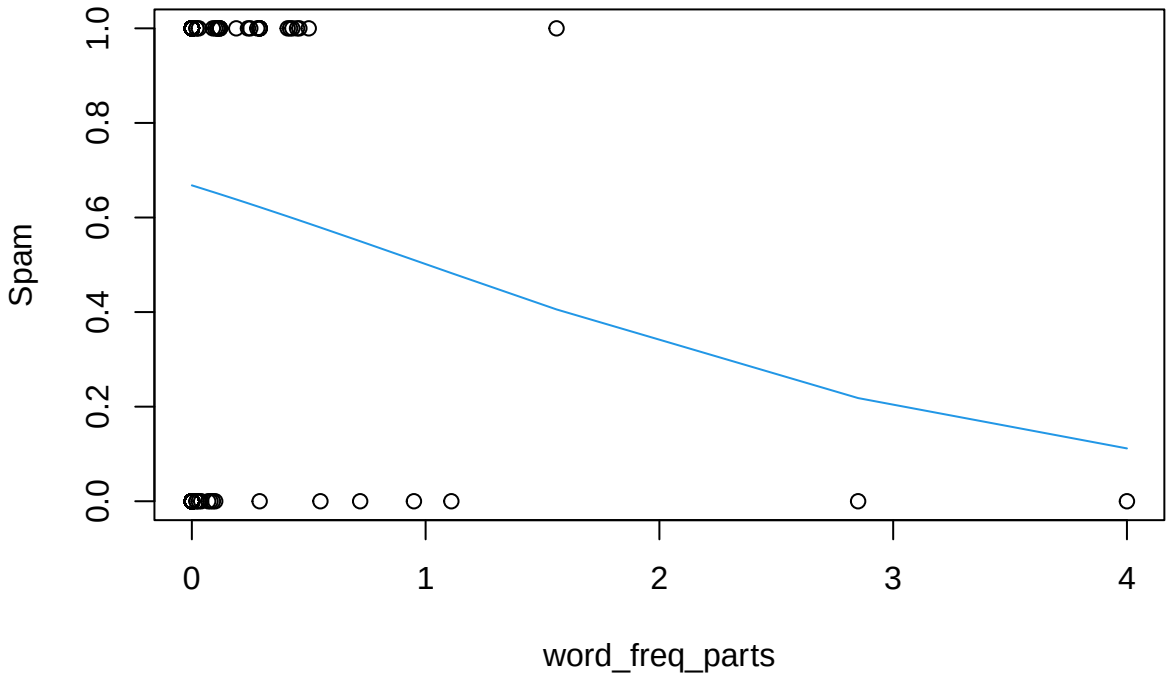

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
word_freq_data

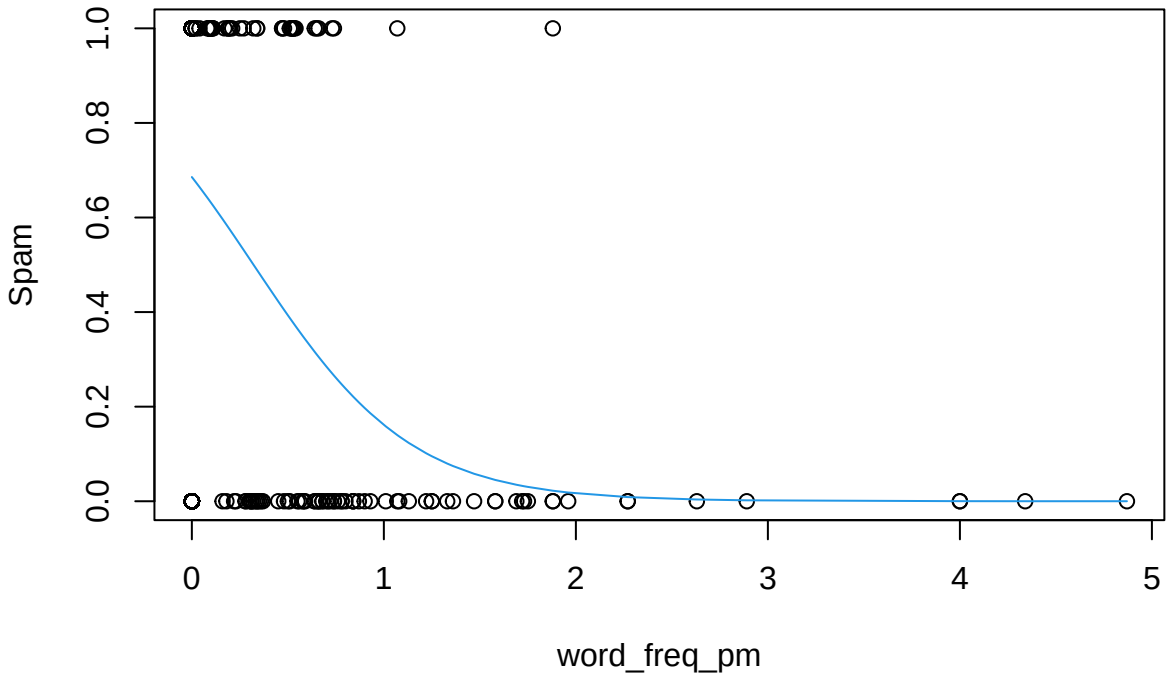

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
word_freq_415


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
word_freq_85

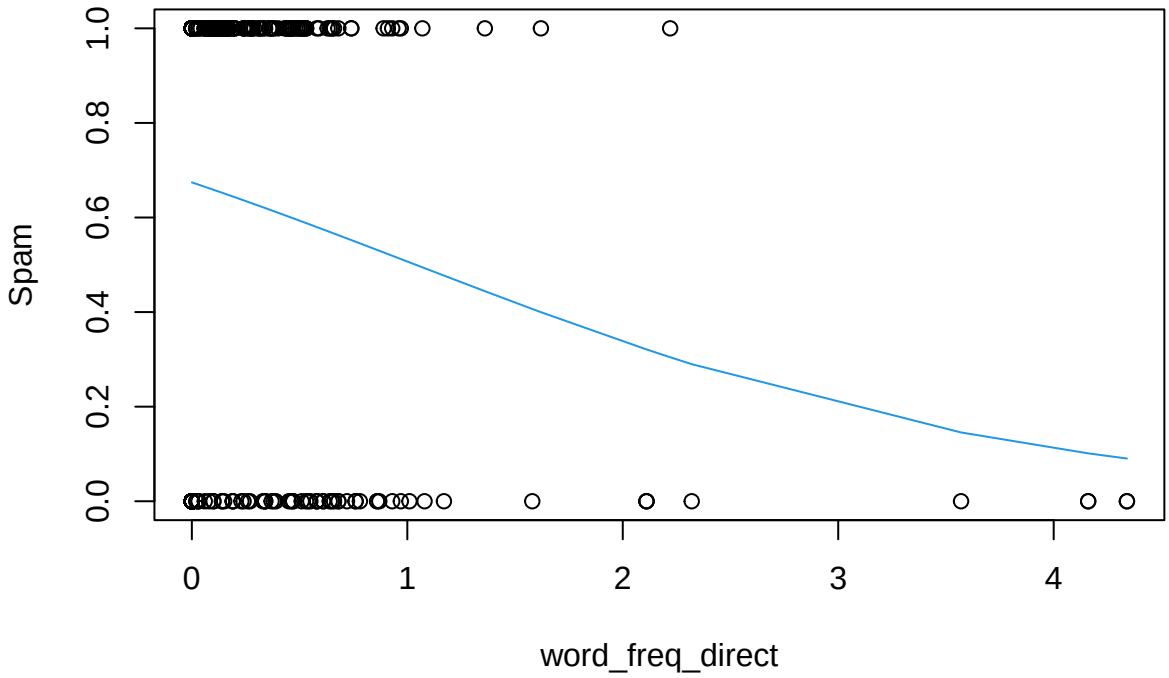

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
word_freq_technology


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
word_freq_1999

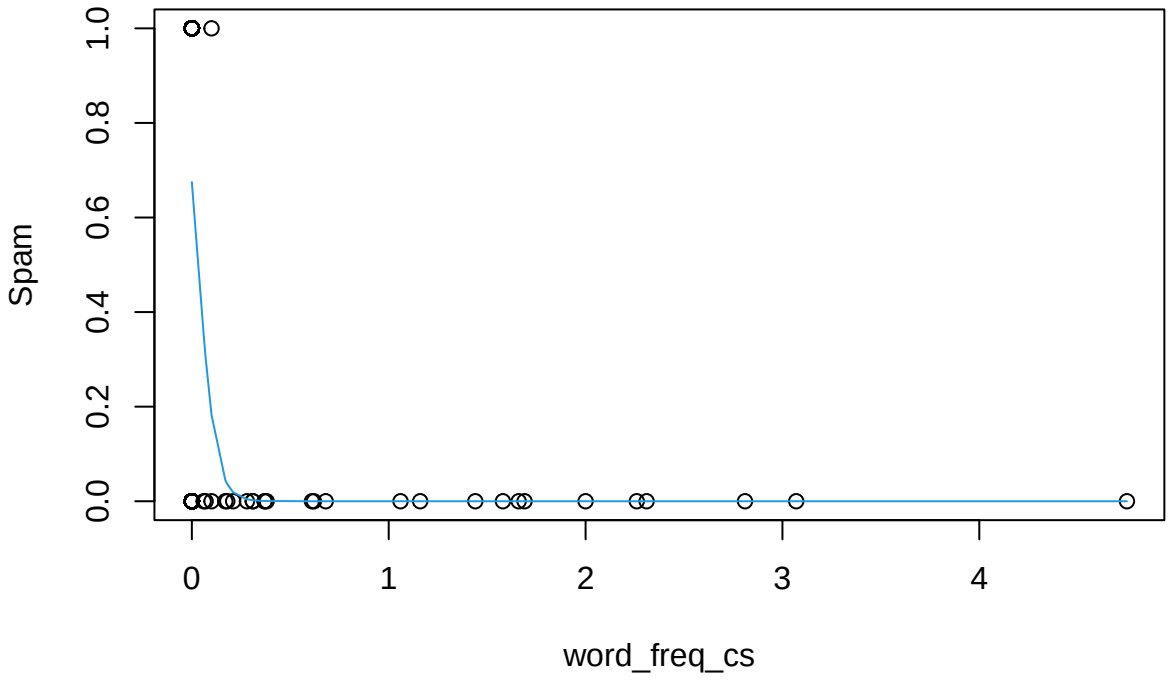

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
word_freq_parts

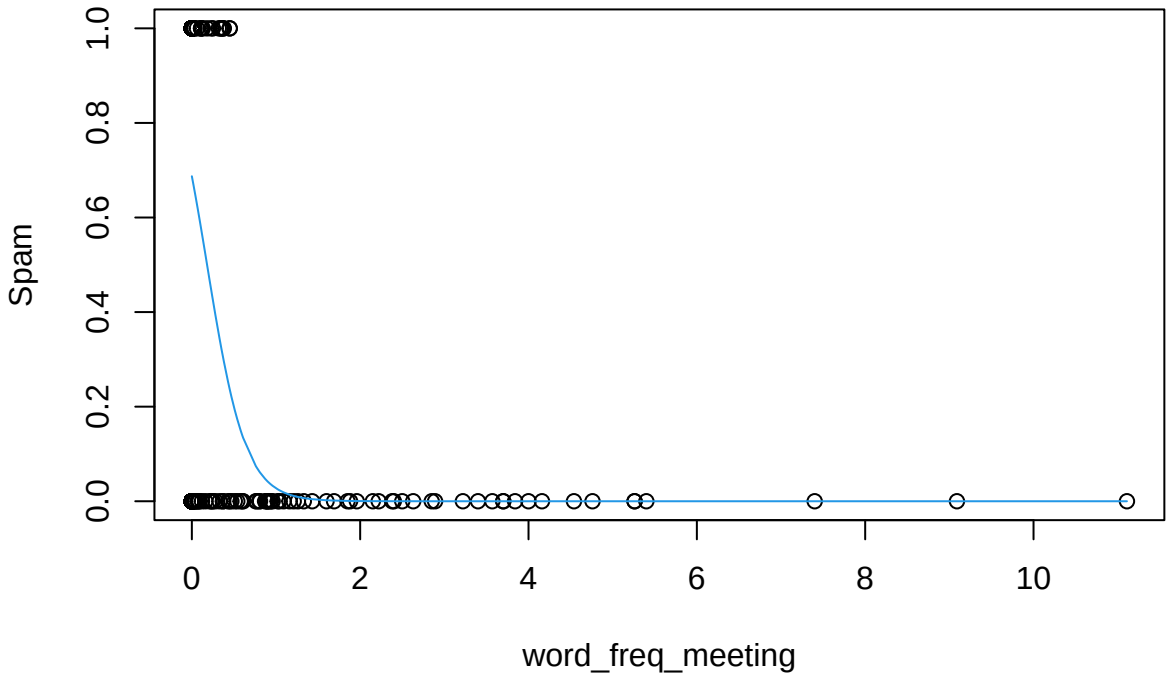



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

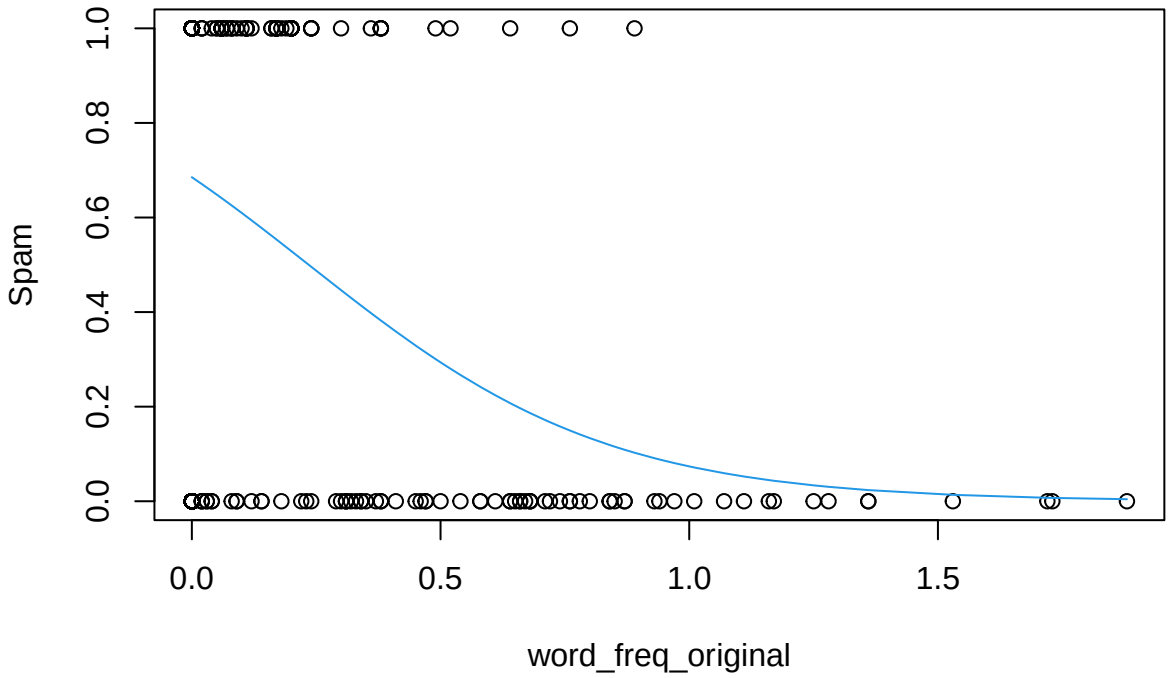


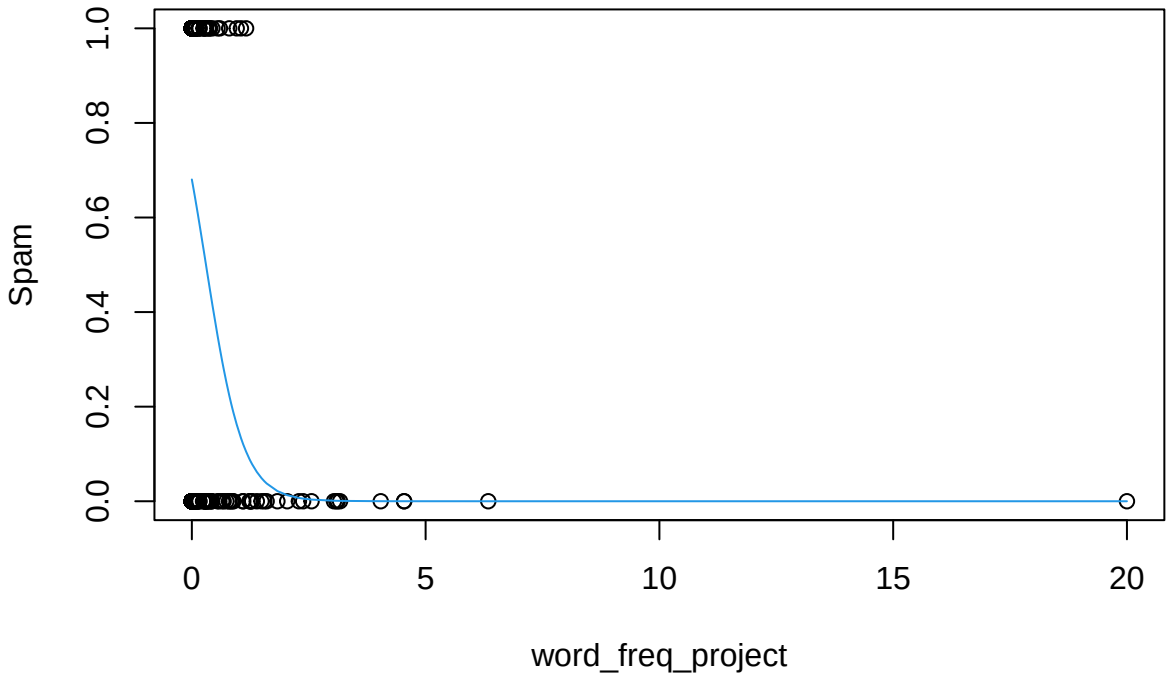

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

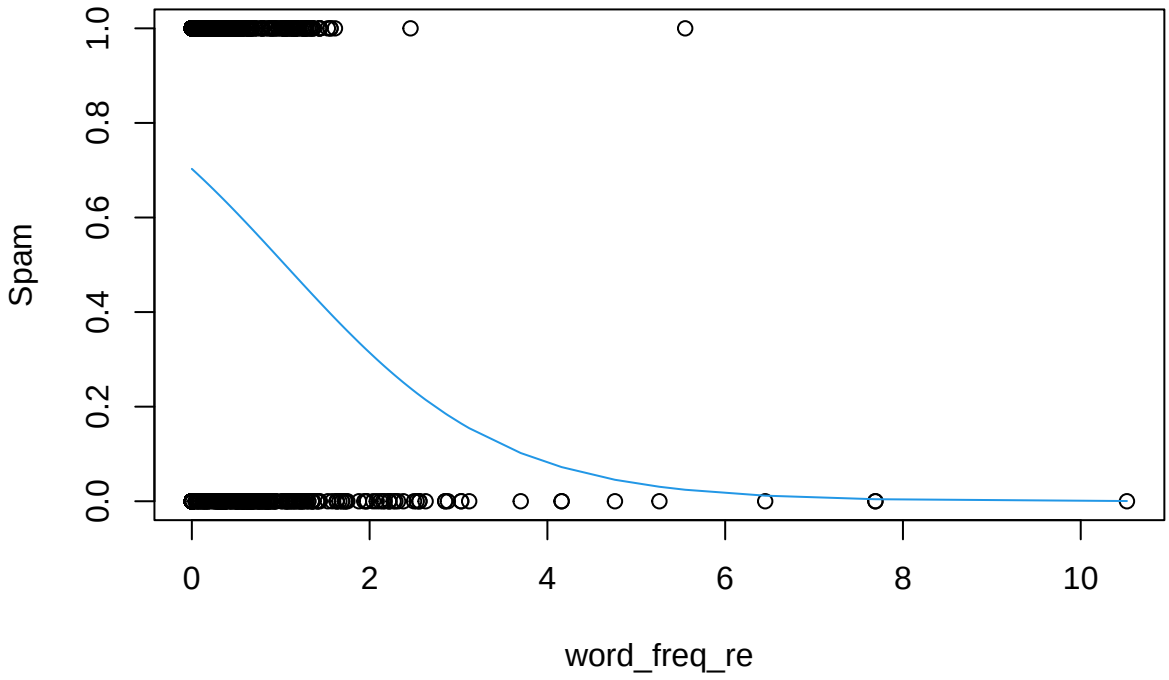

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
word_freq_cs

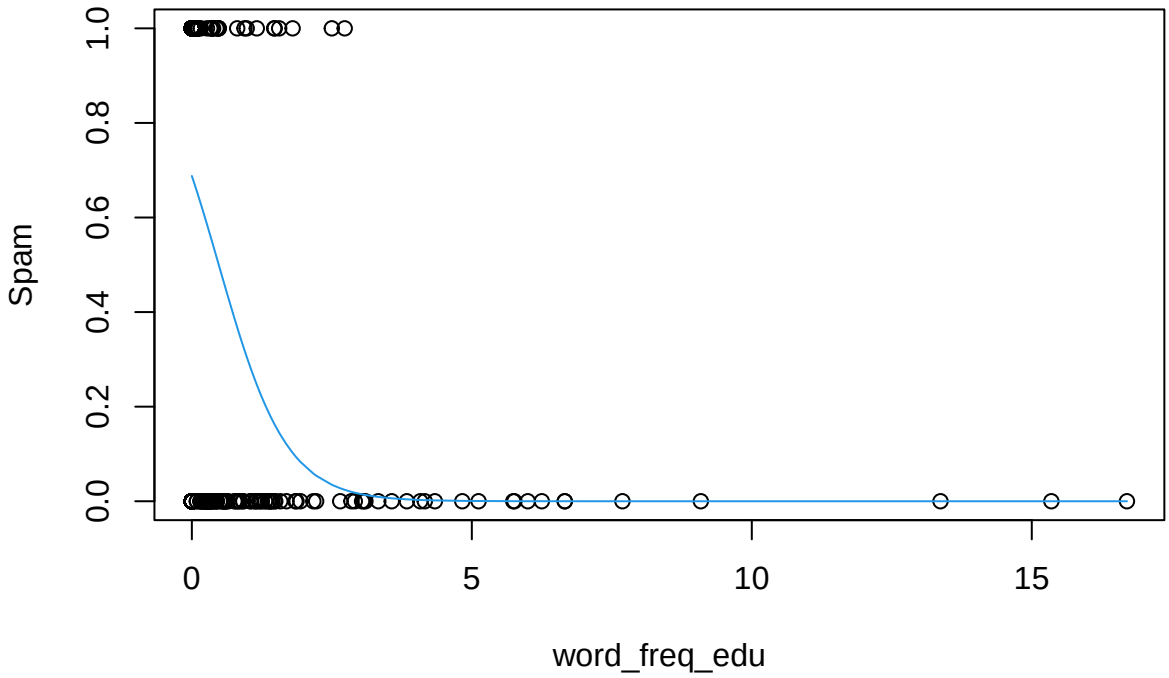



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

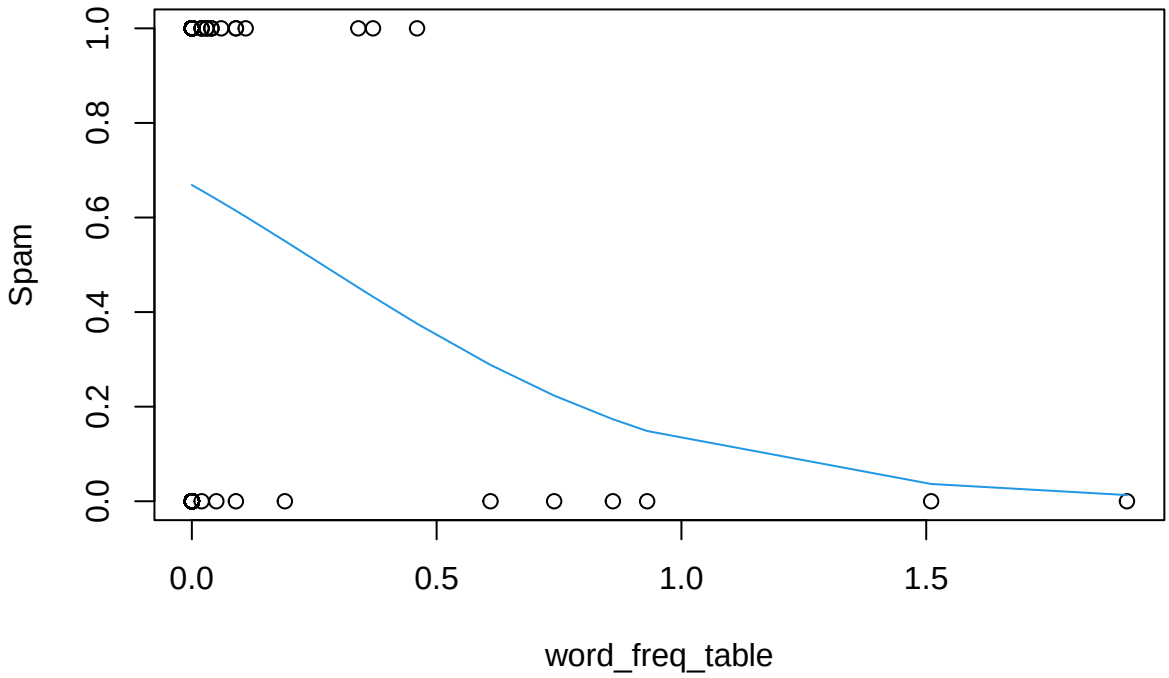

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
word_freq_original

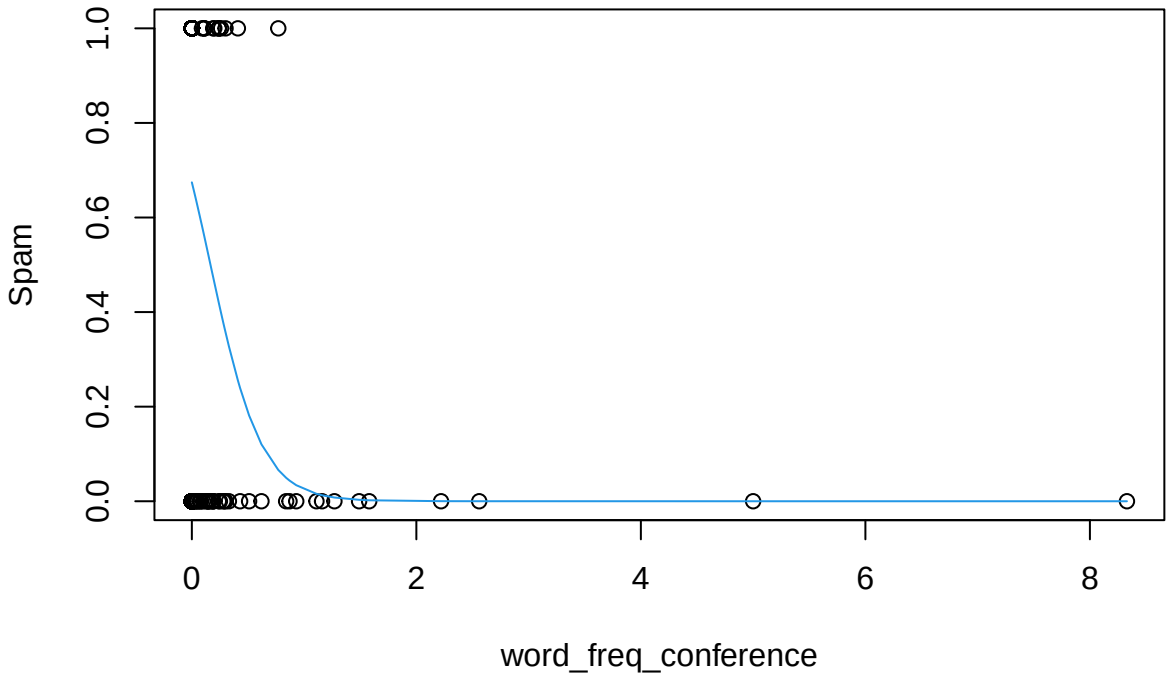

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
word_freq_project

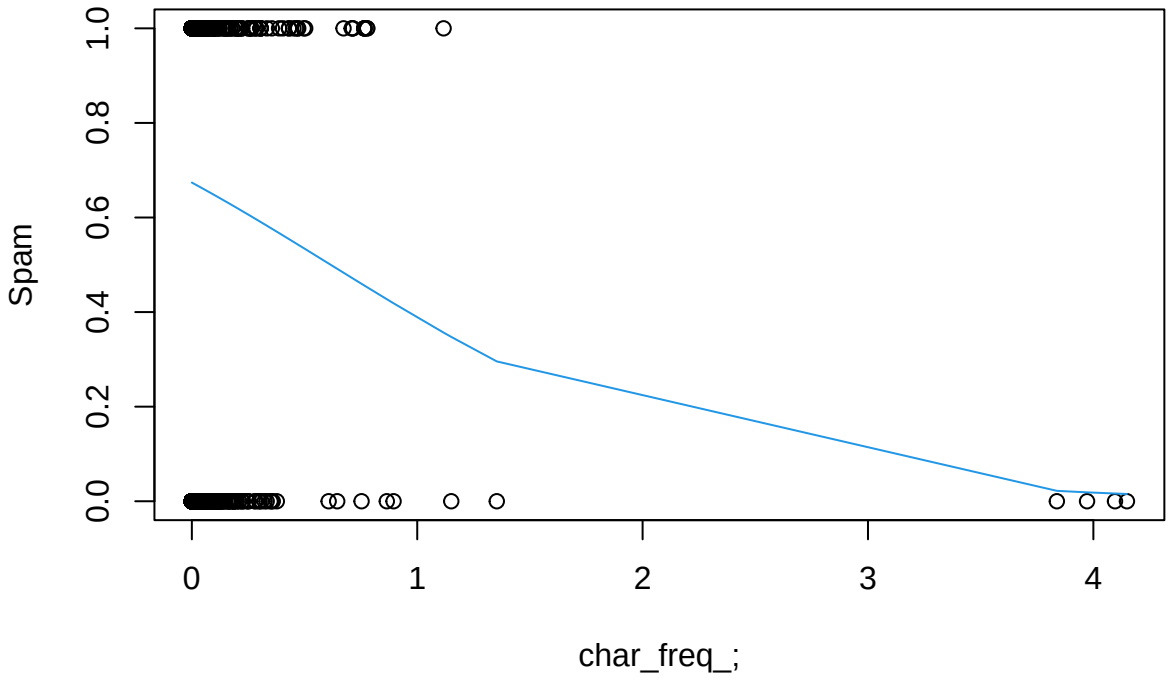

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
word_freq_re

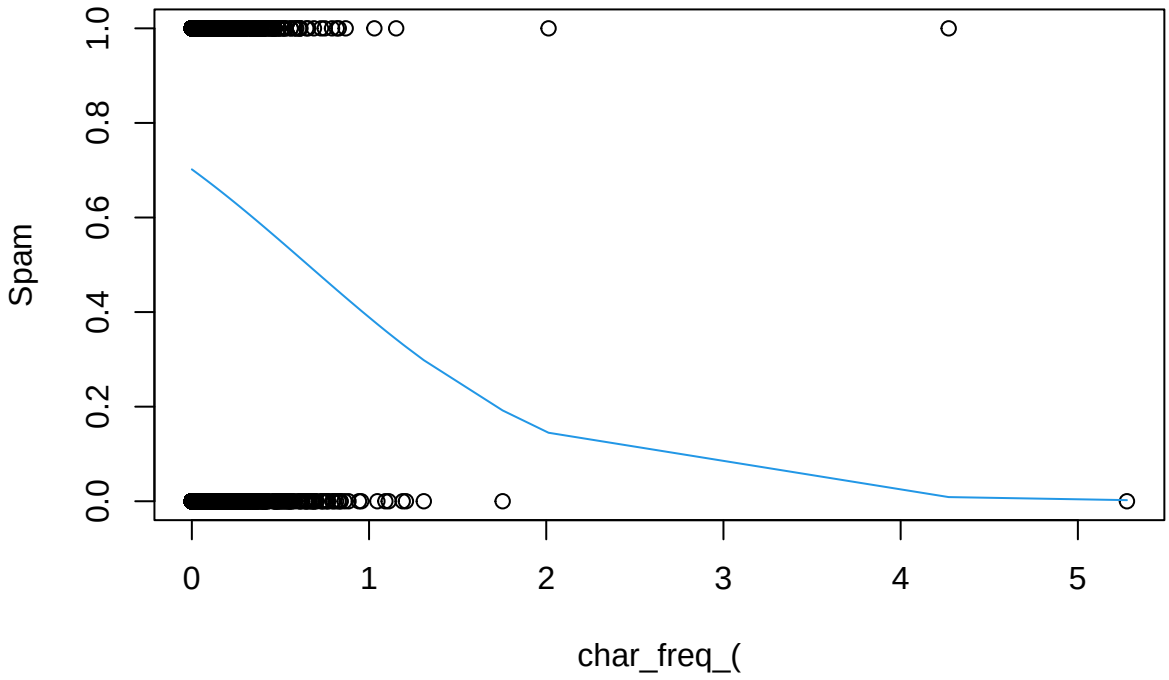



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

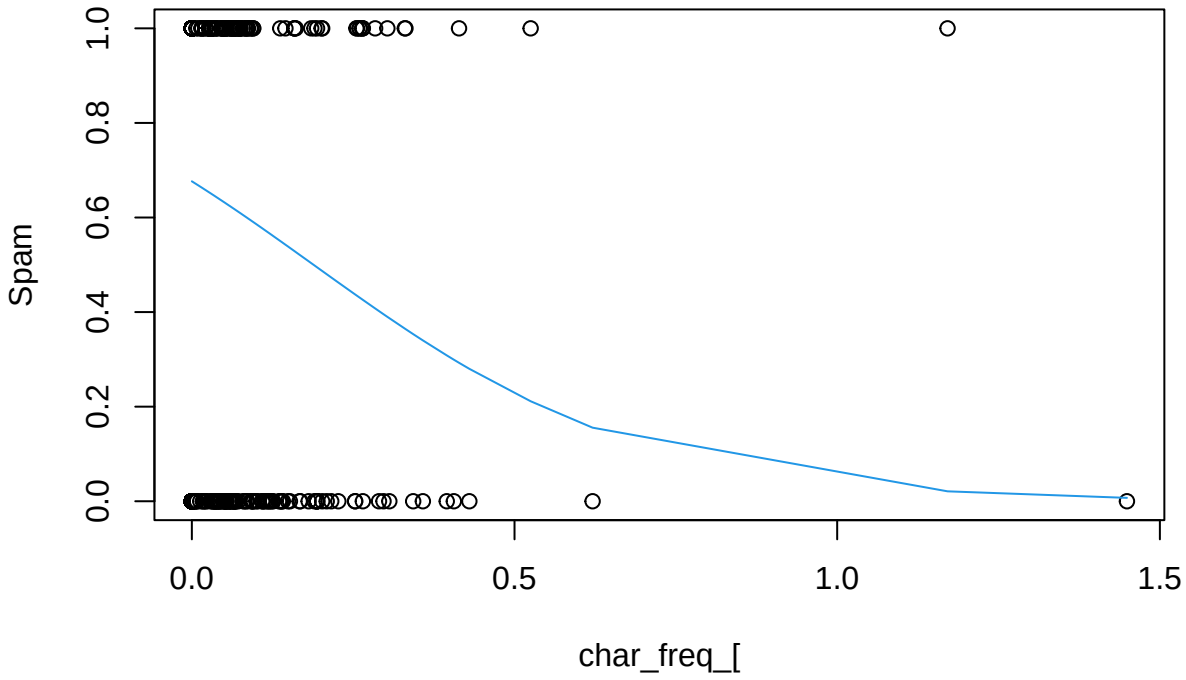

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
word_freq_table


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
word_freq_conference

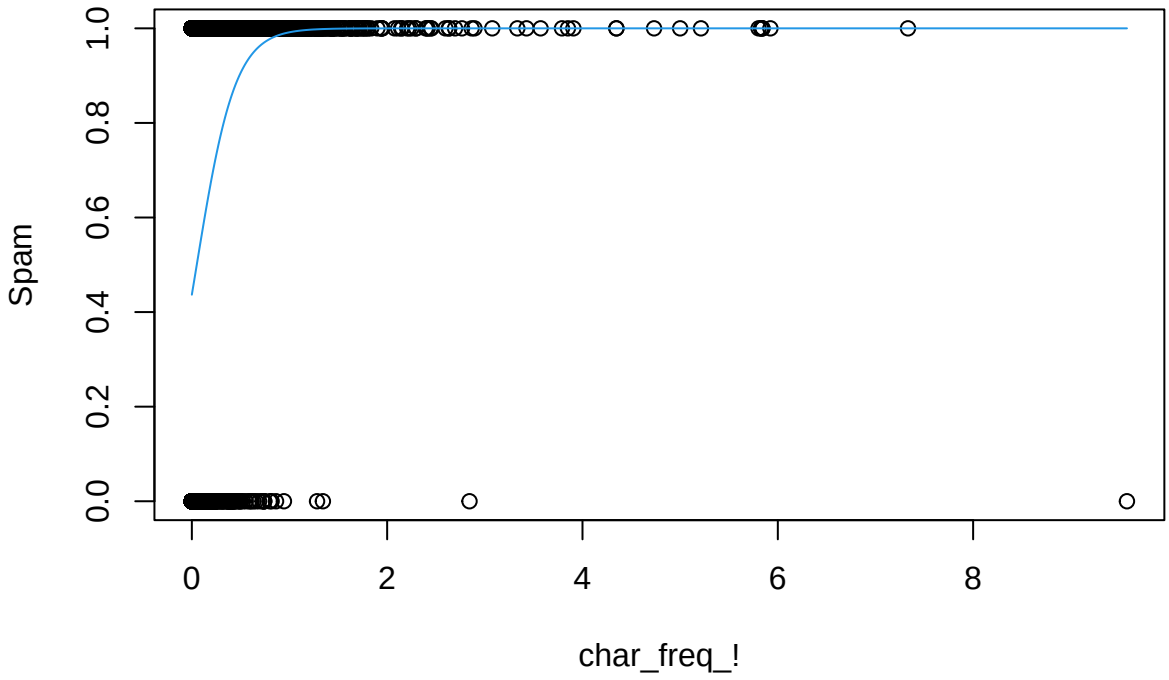

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
char_freq_;

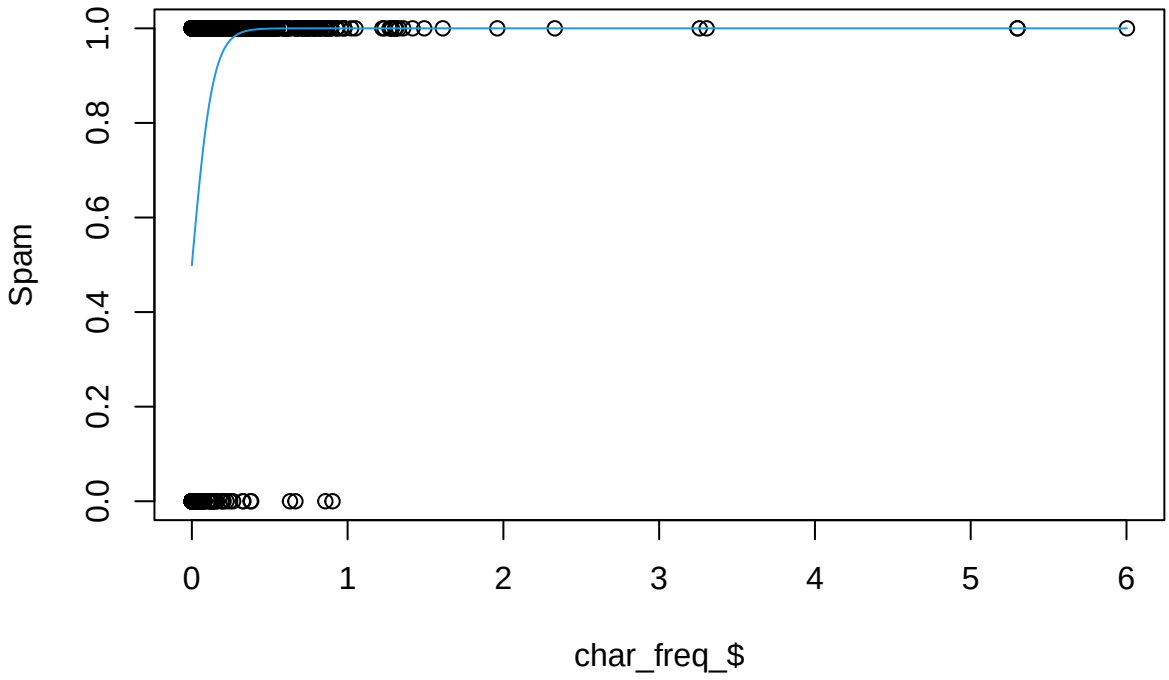



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

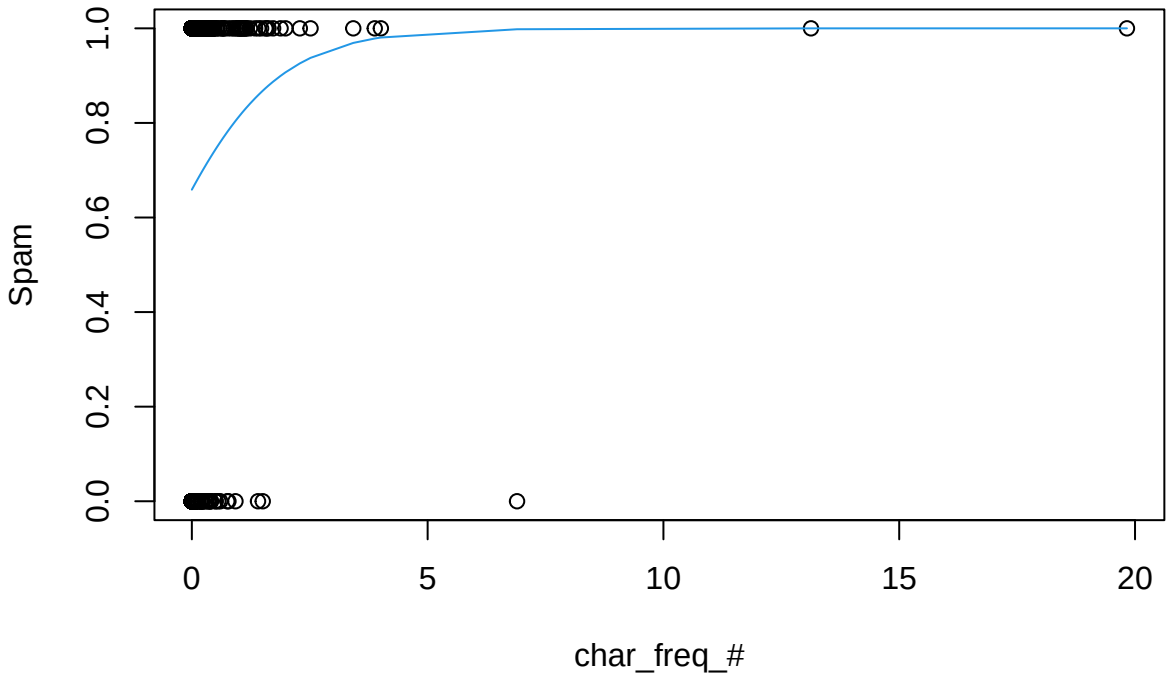



```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

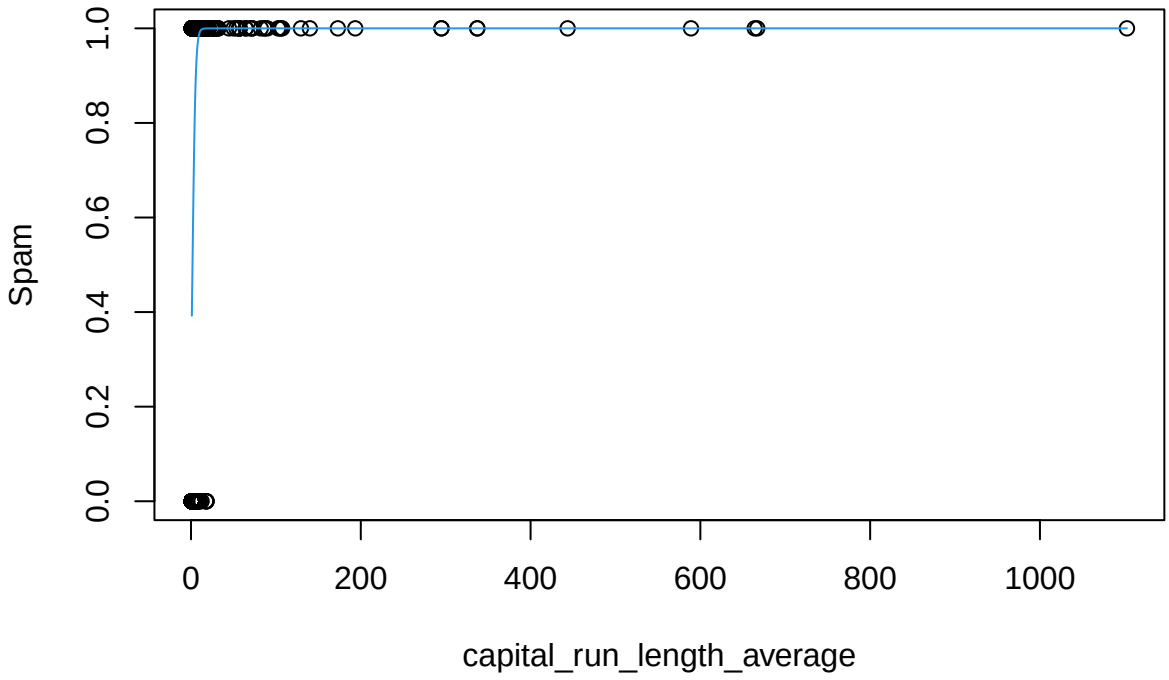

Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
char_freq_!

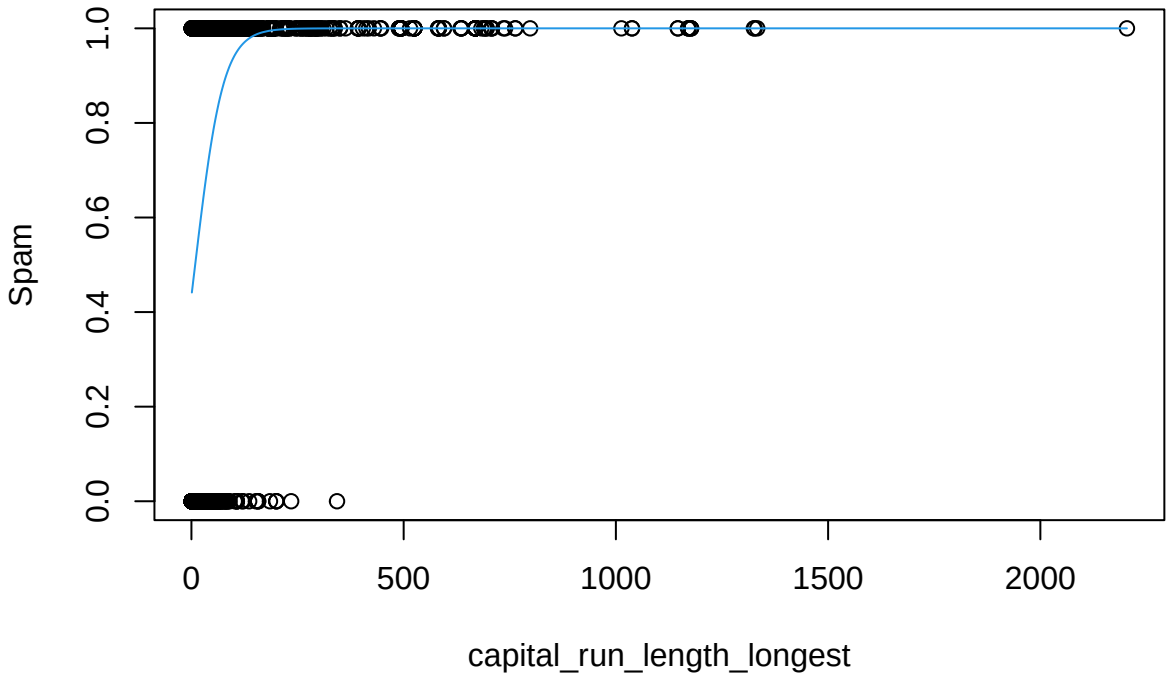


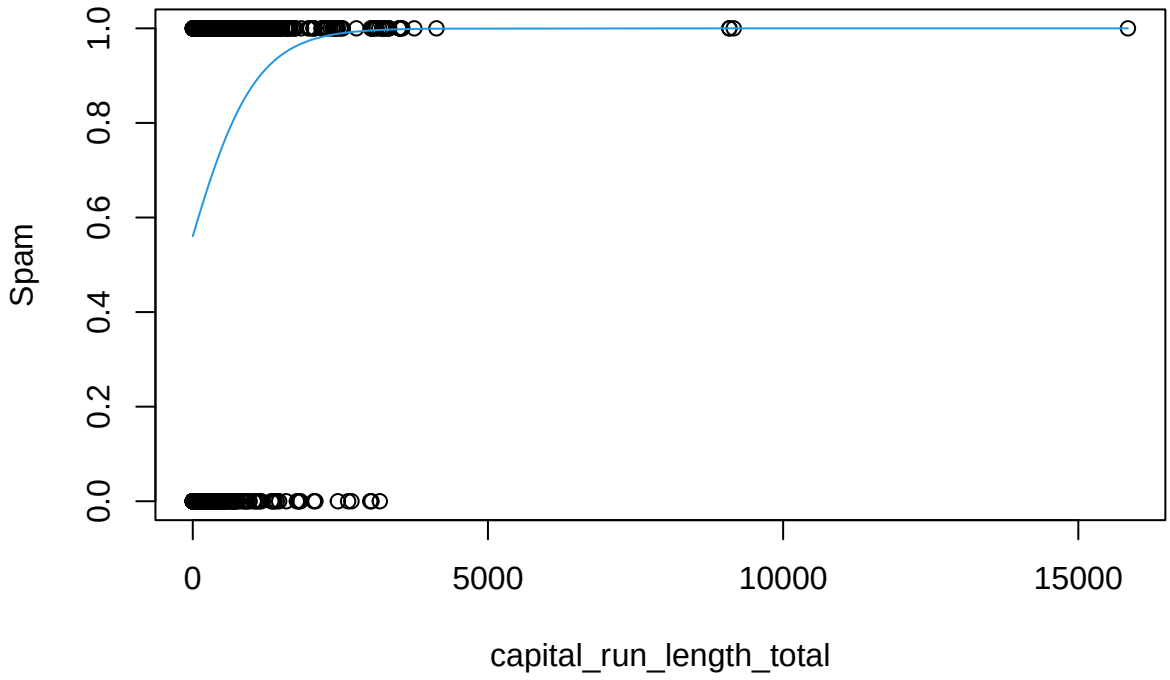

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```




```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
capital_run_length_average


Spam
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
capital_run_length_longest



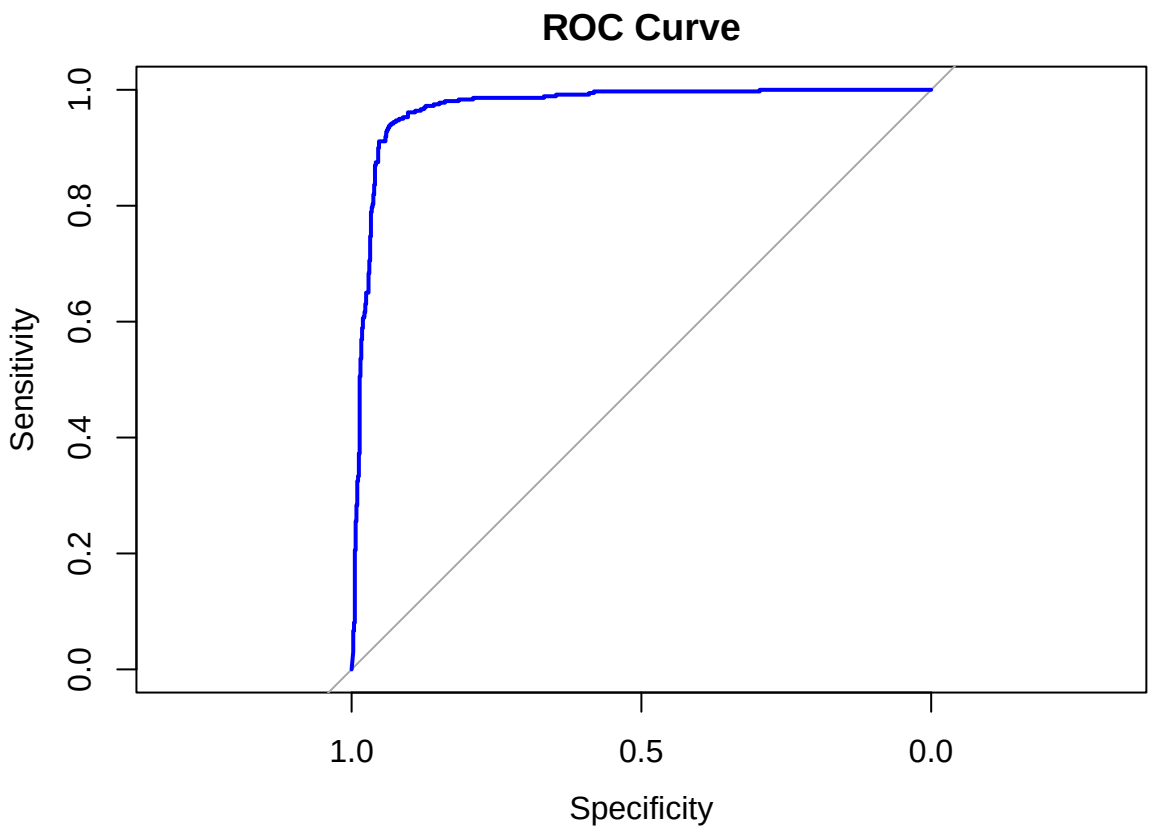

```
# General (ns com funciona la veritat...)

glm.df_tr <- glm(df_tr.01 ~ ., data = as.data.frame(df_tr.vars), family = binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
predictions_glm <- predict(glm.df_tr, newdata = as.data.frame(df_test.vars), type = "response")

roc_curve_glm <- roc(df_test.01, predictions_glm)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(roc_curve_glm, main = "ROC Curve", col = "blue")
```



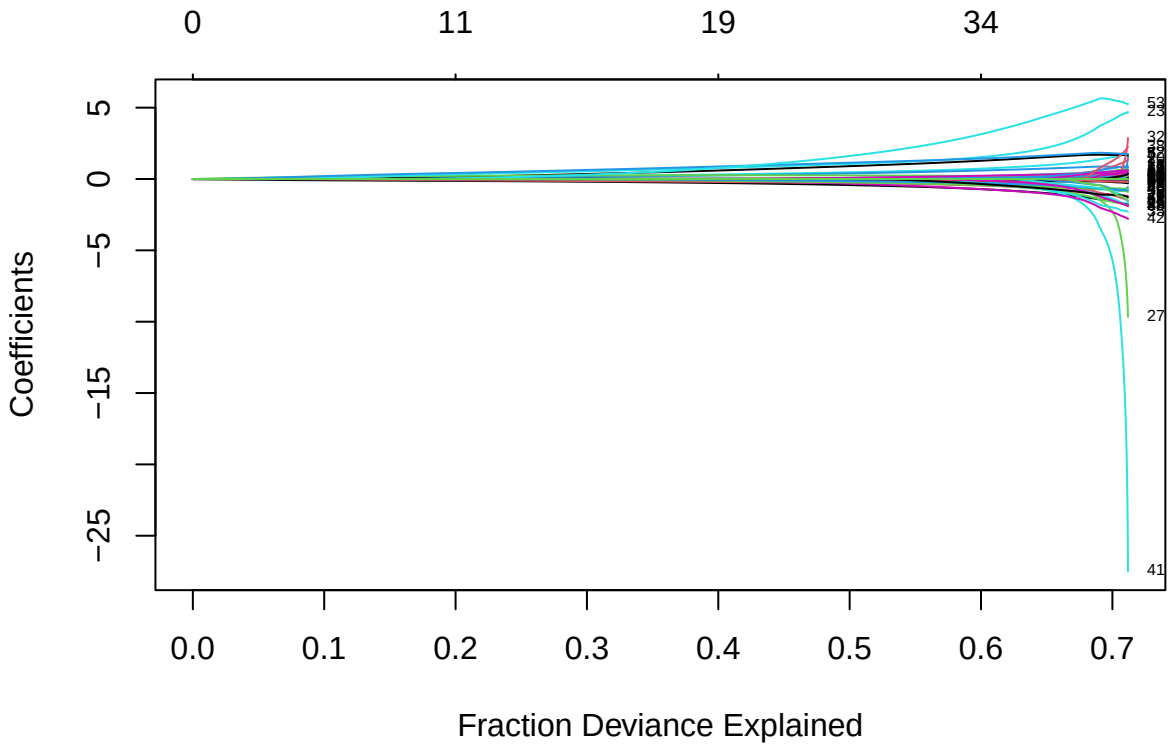

```
predicted_classes_glm <- ifelse(predictions_glm >= 0.5, 1, 0)
misclassification_rate_glm <- mean(predicted_classes_glm != df_test.01)
print("missclassification rate for GLM ussing c = 1/2: "); print(misclassification_rate_glm)
```

```
## [1] "missclassification rate for GLM ussing c = 1/2: "
```

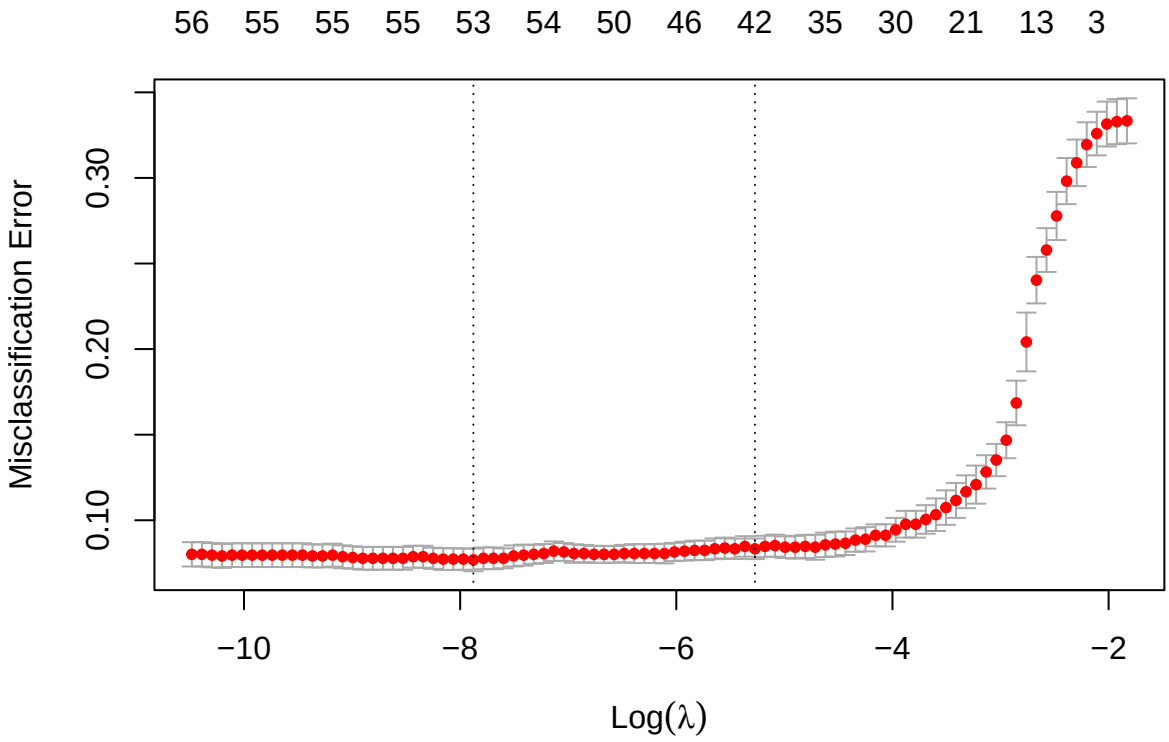
```
## [1] 0.08055556
```

Logistic regression fitted by Lasso (glmnet)

```
set.seed(234) # To ensure replicability

fit_lasso <- glmnet(df_tr.vars, df_tr.01, family = "binomial")
cvfit_lasso = cv.glmnet(df_tr.vars, df_tr.01, family = "binomial", type.measure = "class")

plot(fit_lasso, xvar = "dev", label = TRUE)
```



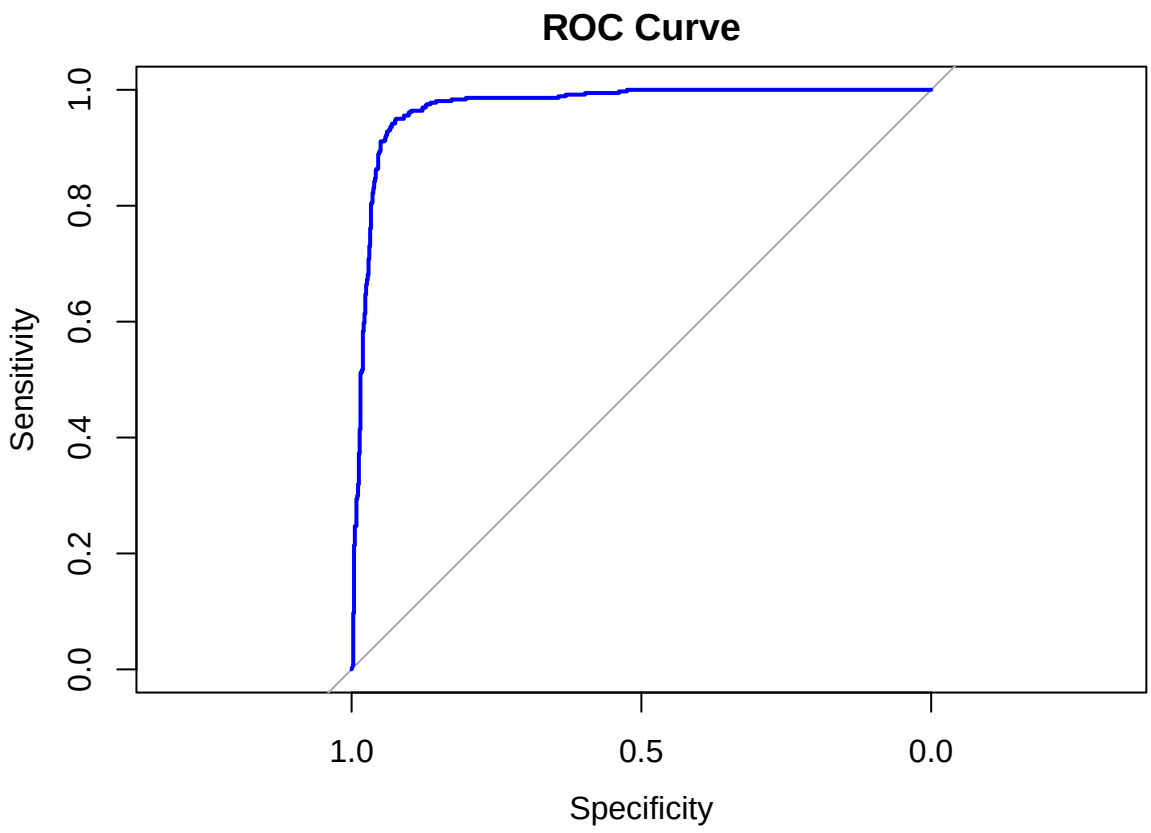

```
plot(cvfit_lasso)
```




```
final_fit_lasso <- glmnet(df_tr.vars, df_tr.01, family = "binomial", lambda = cvfit_lasso$lambda.min)

predictions_lasso <- predict(final_fit_lasso, newx = df_test.vars, type = "response")

roc_curve_lasso <- roc(df_test.01, predictions_lasso)
```

```
## Setting levels: control = 0, case = 1
## Warning in roc.default(df_test.01, predictions_lasso): Deprecated use a matrix
## as predictor. Unexpected results may be produced, please pass a numeric vector.
## Setting direction: controls < cases
```

```
plot(roc_curve_lasso, main = "ROC Curve", col = "blue")
```




```
predicted_classes_lasso <- ifelse(predictions_lasso >= 0.5, 1, 0)
misclassification_rate_lasso <- mean(predicted_classes_lasso != df_test.01)
print("missclassification rate for LASSO ussing c = 1/2: "); print(misclassification_rate_lasso)
```

```
## [1] "missclassification rate for LASSO ussing c = 1/2: "
## [1] 0.08055556
```

AFEGIT –> ES POT TREURE...

```
cvfit_lasso$lambda.min
```

```
## [1] 0.0003791285
```

```
coefs_min_lasso <- coef(cvfit_lasso, s = "lambda.min")

print("The coefficients sorted by impact for lambda_{min} are: ")
```

```
## [1] "The coefficients sorted by impact for lambda_{min} are: "
```

```
coefs_min_lasso_aux = coefs_min_lasso[-1,]
coefs_min_lasso_aux[order(abs(coefs_min_lasso_aux))]
```

```
##              word_freq_857              word_freq_415
##                 0.00000000                 0.00000000
##         word_freq_original capital_run_length_longest
##                 0.00000000                 0.00000000
##   capital_run_length_total             word_freq_labs
##                 0.00113036                -0.00648816
##             word_freq_will            word_freq_order
```

```
##                0.01291253                0.02337009
##             word_freq_you           word_freq_people
##                0.02742598                0.09256507
##             word_freq_font          word_freq_receive
##                0.09596313               -0.11603344
##            word_freq_email             word_freq_all
##                0.14099258               -0.14518890
##             word_freq_your            word_freq_mail
##                0.14901711                0.15377830
## capital_run_length_average           word_freq_telnet
##                0.16711107               -0.16924691
##          word_freq_address                char_freq_(
##               -0.16960930               -0.18175099
##             word_freq_make            word_freq_over
##               -0.18616517                0.23204592
##             word_freq_1999         word_freq_internet
##               -0.23639753                0.31441393
##            word_freq_money              word_freq_3d
##                0.34483262                0.35530859
##               char_freq_#           word_freq_report
##                0.40422734                0.45432387
##       word_freq_technology         word_freq_business
##                0.56057955                0.56409074
##        word_freq_addresses              word_freq_re
##                0.62084721               -0.65560514
##              word_freq_edu           word_freq_direct
##               -0.74798159               -0.79160794
##               word_freq_pm             word_freq_650
##               -0.80513414                0.81835186
##             word_freq_free            word_freq_table
##                0.84873961               -0.99631305
##           word_freq_credit                char_freq_;
##                1.00677152               -1.12735392
##              word_freq_hpl                char_freq_[
##               -1.12821671               -1.14407922
##          word_freq_project            word_freq_parts
##               -1.24661482                1.51421155
##             word_freq_lab             word_freq_our
##               -1.53369019                1.60077862
##              word_freq_hp       word_freq_conference
##               -1.62900737               -1.63809021
##             word_freq_data           word_freq_remove
##               -1.67733234                1.68495878
##               char_freq_!              word_freq_85
##                1.78492812               -2.08919754
##          word_freq_meeting           word_freq_george
##               -2.47120618               -3.11629849
##             word_freq_000                char_freq_$
##                4.36265744                5.46424872
##              word_freq_cs
##               -8.01556313
```

From these observations, we can observe which predictors are related to NO spam (negative ones) and the ones related to spam (positive ones) and ordered. The most relevant predictors to detect NO spam are: “george”,

"conference", "cs", "meeting" The most relevant predictors to detect spam are: "$", "remove", "000", "money" Some discarded predictors are: "857", "415", "capital_run_length_total" and "capital_run_length_longest"

```
cvfit_lasso$lambda.1se
```

```
## [1] 0.00512979
```

```
coefs_min_lasso <- coef(cvfit_lasso, s = "lambda.1se")

print("The coefficients sorted by impact for lambda_{min} are: ")
```

```
## [1] "The coefficients sorted by impact for lambda_{min} are: "
```

```
coefs_min_lasso_aux = coefs_min_lasso[-1,]
coefs_min_lasso_aux[order(abs(coefs_min_lasso_aux))]
```

```
##             word_freq_make              word_freq_all
##               0.0000000000               0.0000000000
##          word_freq_receive             word_freq_will
##               0.0000000000               0.0000000000
##           word_freq_people        word_freq_addresses
##               0.0000000000               0.0000000000
##             word_freq_labs              word_freq_857
##               0.0000000000               0.0000000000
##              word_freq_415            word_freq_parts
##               0.0000000000               0.0000000000
##         word_freq_original               char_freq_(
##               0.0000000000               0.0000000000
##               char_freq_[               char_freq_#
##               0.0000000000               0.0000000000
## capital_run_length_average   capital_run_length_total
##               0.0000000000               0.0005427456
## capital_run_length_longest       word_freq_technology
##               0.0023748514               0.0028510536
##             word_freq_over               word_freq_3d
##               0.0111513856               0.0160914960
##              word_freq_you            word_freq_order
##               0.0194034079               0.0429990866
##           word_freq_direct           word_freq_telnet
##              -0.0546924212              -0.0547054424
##          word_freq_address             word_freq_mail
##              -0.0985430658               0.1335086491
##           word_freq_report            word_freq_email
##               0.1335148198               0.1382706973
##             word_freq_font             word_freq_1999
##               0.1474484068              -0.1730064176
##             word_freq_your              word_freq_650
##               0.1817628292               0.1824316992
##            word_freq_table           word_freq_credit
##              -0.2461965881               0.2700390688
##         word_freq_internet            word_freq_money
##               0.2789950784               0.3169182039
##          word_freq_project           word_freq_george
##              -0.3186845517              -0.3217933939
##              word_freq_lab         word_freq_business
##              -0.3349871063               0.3350163586
```

```
##       word_freq_conference              word_freq_edu
##               -0.3822693190              -0.4268364022
##                word_freq_pm               word_freq_re
##               -0.5193462867              -0.5660340933
##                 char_freq_;              word_freq_hpl
##               -0.6765299983              -0.6776994202
##              word_freq_free               word_freq_85
##                0.7332672172              -0.7375996775
##              word_freq_data               word_freq_cs
##               -0.7502023666              -0.7701167515
##               word_freq_our               word_freq_hp
##                0.9719774908              -0.9873579715
##           word_freq_meeting           word_freq_remove
##               -1.0279058137               1.5366034282
##                 char_freq_!              word_freq_000
##                1.6396429361               2.1800238435
##                 char_freq_$
##                4.3980445592
```

From these observations, we can observe which predictors are related to NO spam (negative ones) and the ones related to spam (positive ones) and ordered. The most relevant predictors to detect NO spam are: “cs”, “conference”, “meeting”, “hp” The most relevant predictors to detect spam are: “$”, “remove”, “000”, “money” Some discarded predictors are: “make”, “all”, “mail” and “addresses”

k-nn binary regression (class)

```
k_vec = c(1, 2, 5, 10, 20, 50, 100) # Rule of thumb (sqrt(n)) ~=50 so we pick 50...

for(j in 1:length(k_vec)){
  set.seed(555)
  knn_fit <- knn(train = df_tr.vars, test = df_tr.vars, cl = df_tr.01, k = k_vec[j])
  knn_cv <- knn.cv(train = df_tr.vars, cl = df_tr.01, k = k_vec[j])

  predictions_knn <- knn(train = df_tr.vars, test = df_test.vars, cl = df_tr.01, k = k_vec[j])
  predictions_knn <- as.integer(as.character(predictions_knn))

  roc_curve_knn <- roc(df_test.01, predictions_knn)
  plot(roc_curve_knn, main = "ROC Curve", col = "blue")

  predicted_classes_knn <- ifelse(predictions_knn >= 0.5, 1, 0)
  misclassification_rate_knn <- mean(predicted_classes_knn != df_test.01)
  print("missclassification rate for LASSO ussing c = 1/2 and k = to : ");print(k_vec[j]);
  print("is: "); print(misclassification_rate_knn )


}
```

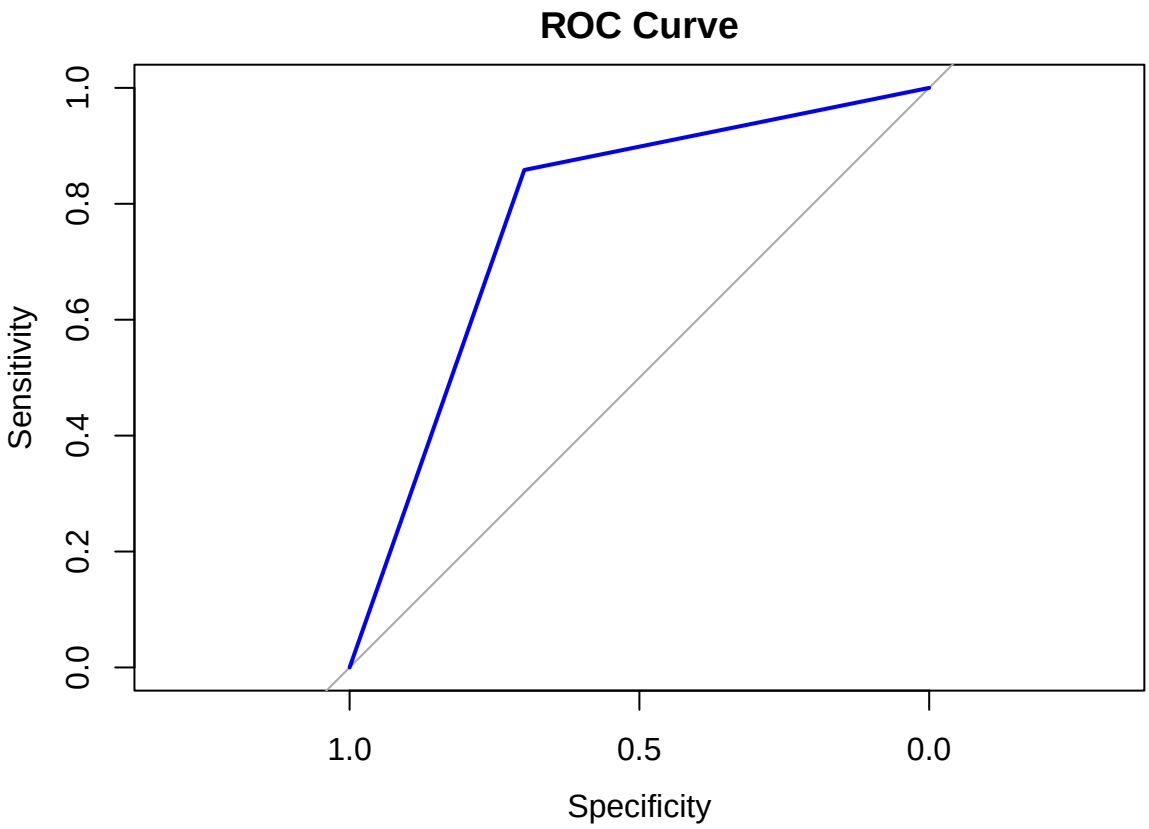
```
## Setting levels: control = 0, case = 1
```

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

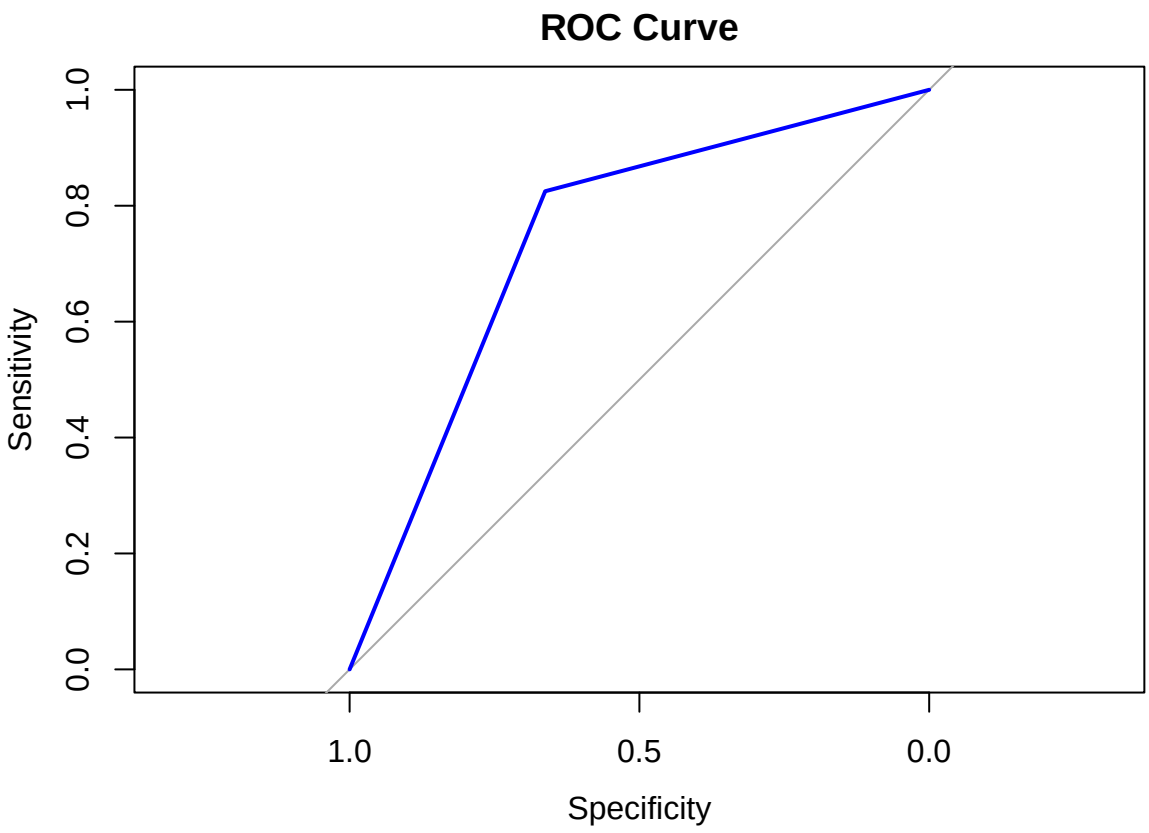
```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 1
## [1] "is: "
## [1] 0.2481481
```

```
## Setting levels: control = 0, case = 1
```

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

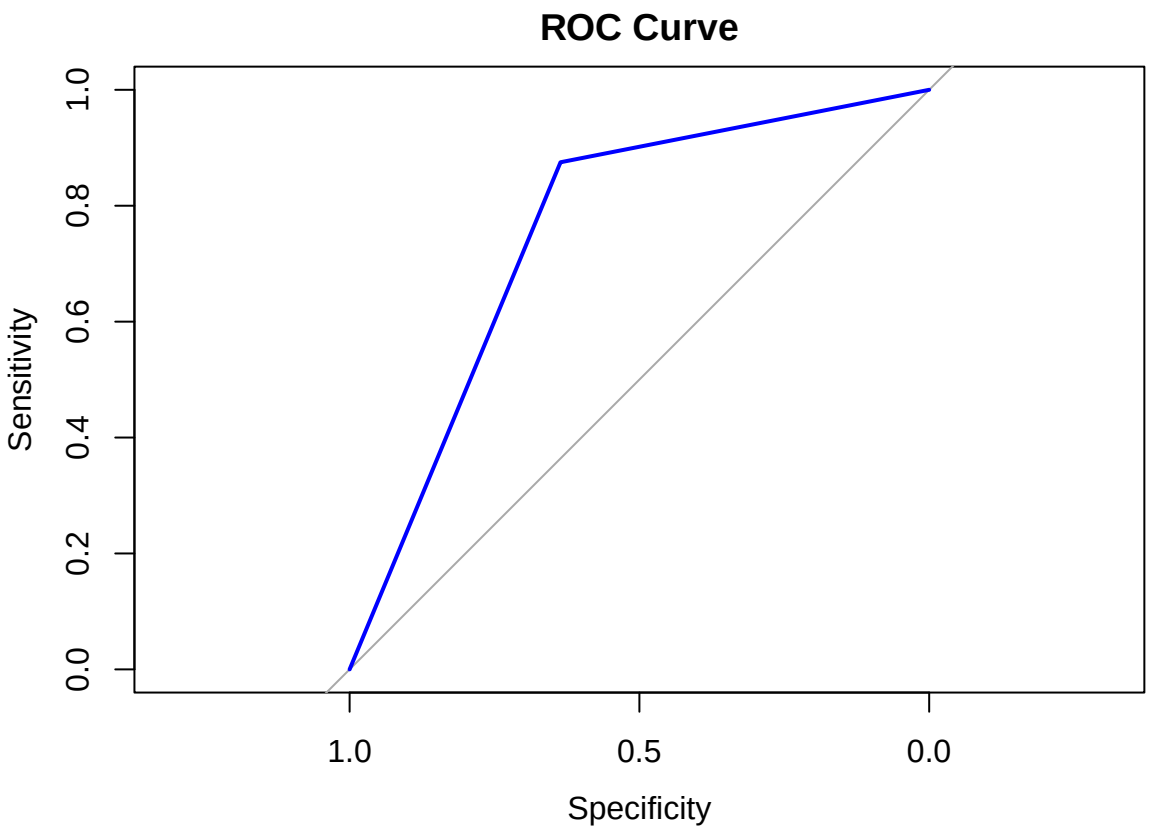



```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 2
## [1] "is: "
## [1] 0.2833333
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



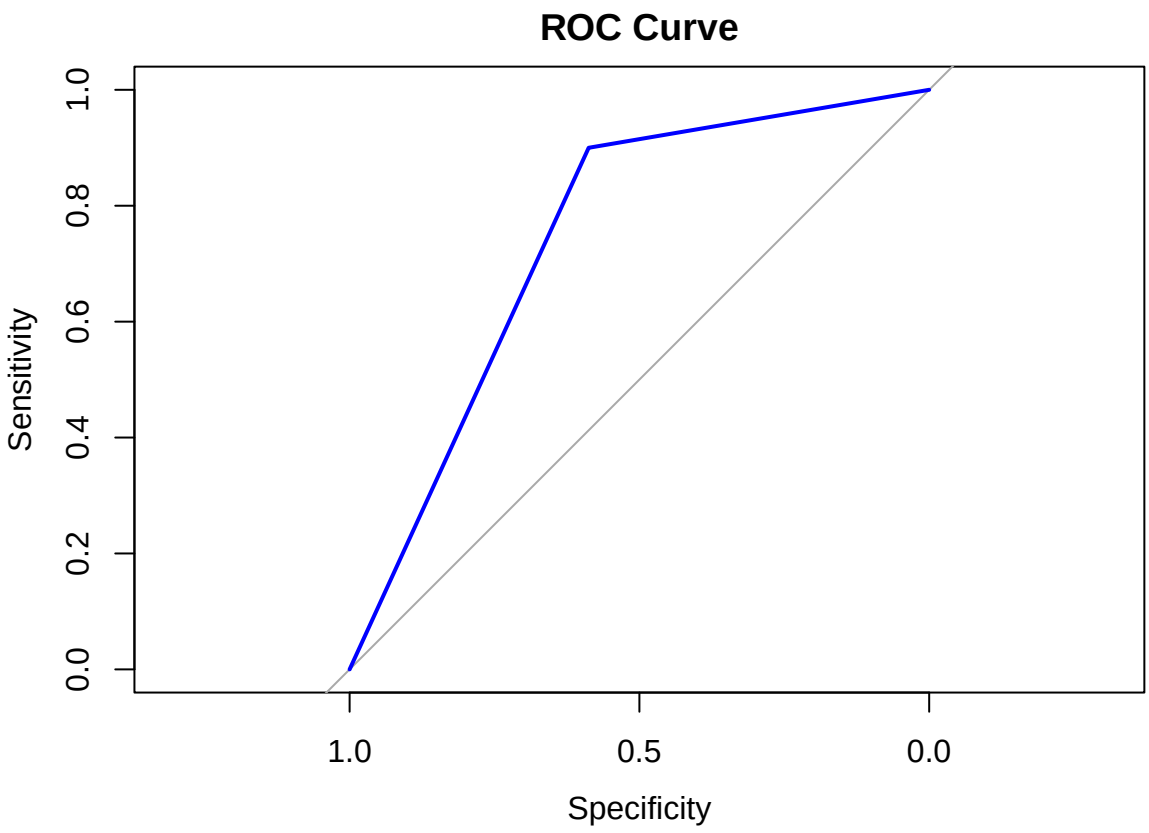

```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 5
## [1] "is: "
## [1] 0.2842593
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



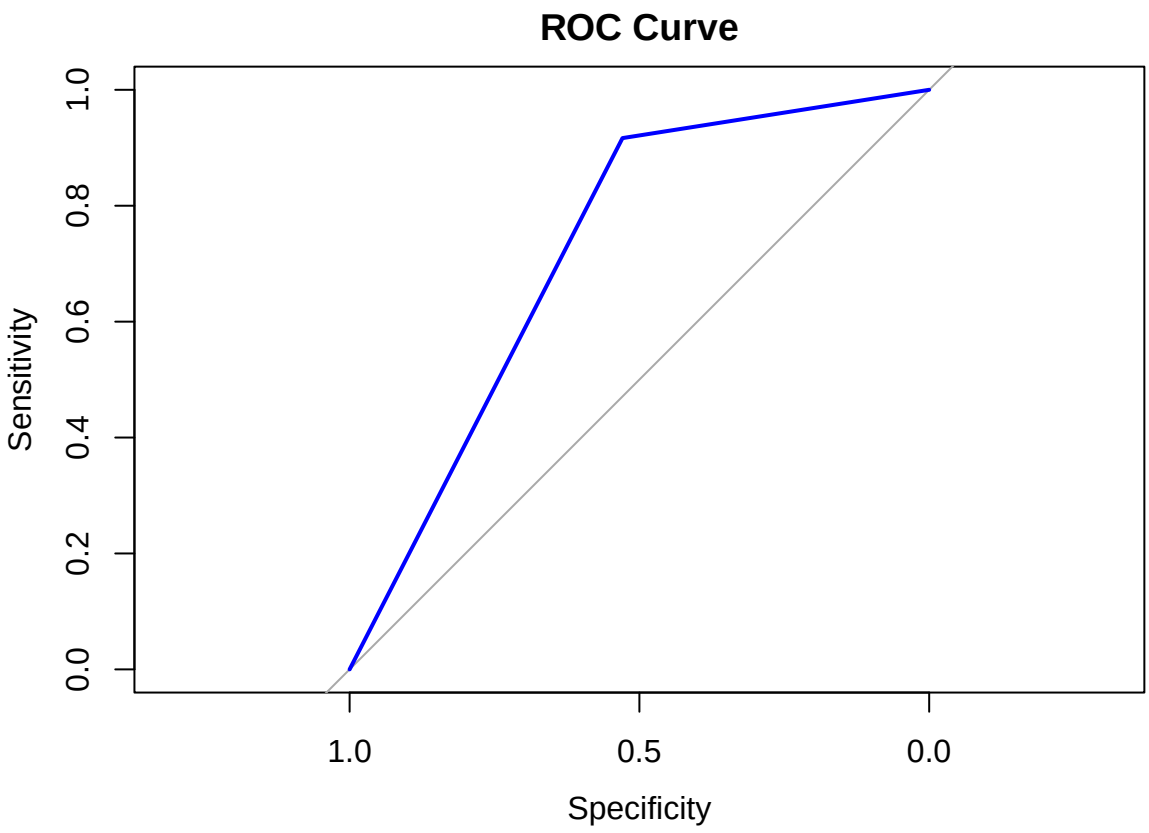

```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 10
## [1] "is: "
## [1] 0.3083333
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



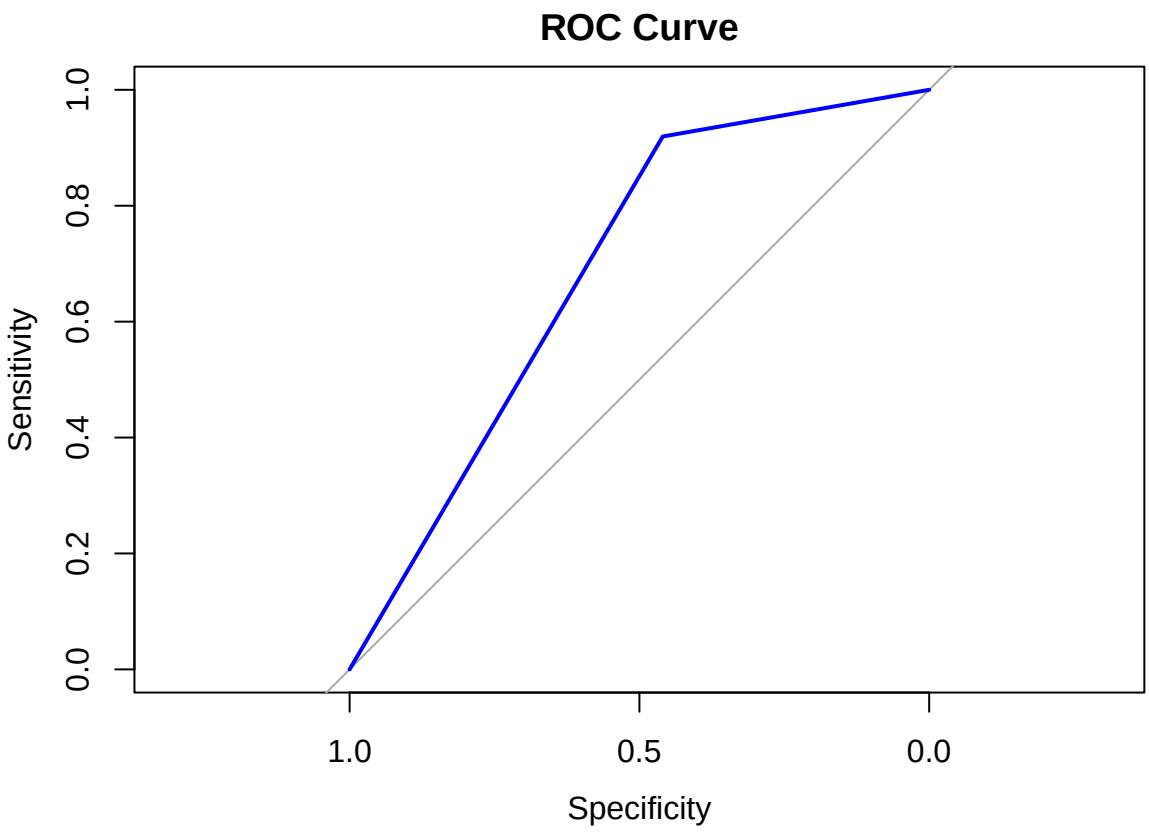

```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 20
## [1] "is: "
## [1] 0.3416667
```

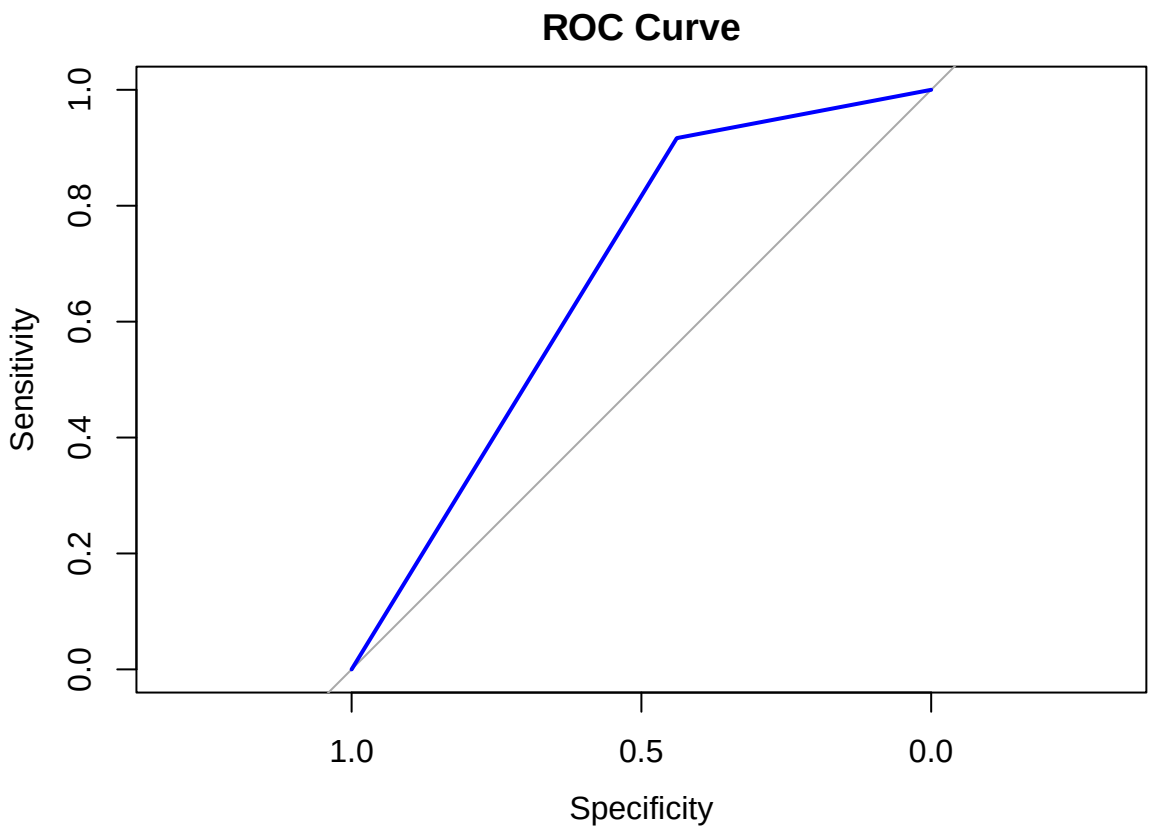
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```




```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 50
## [1] "is: "
## [1] 0.387037
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```


ROC Curve
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity

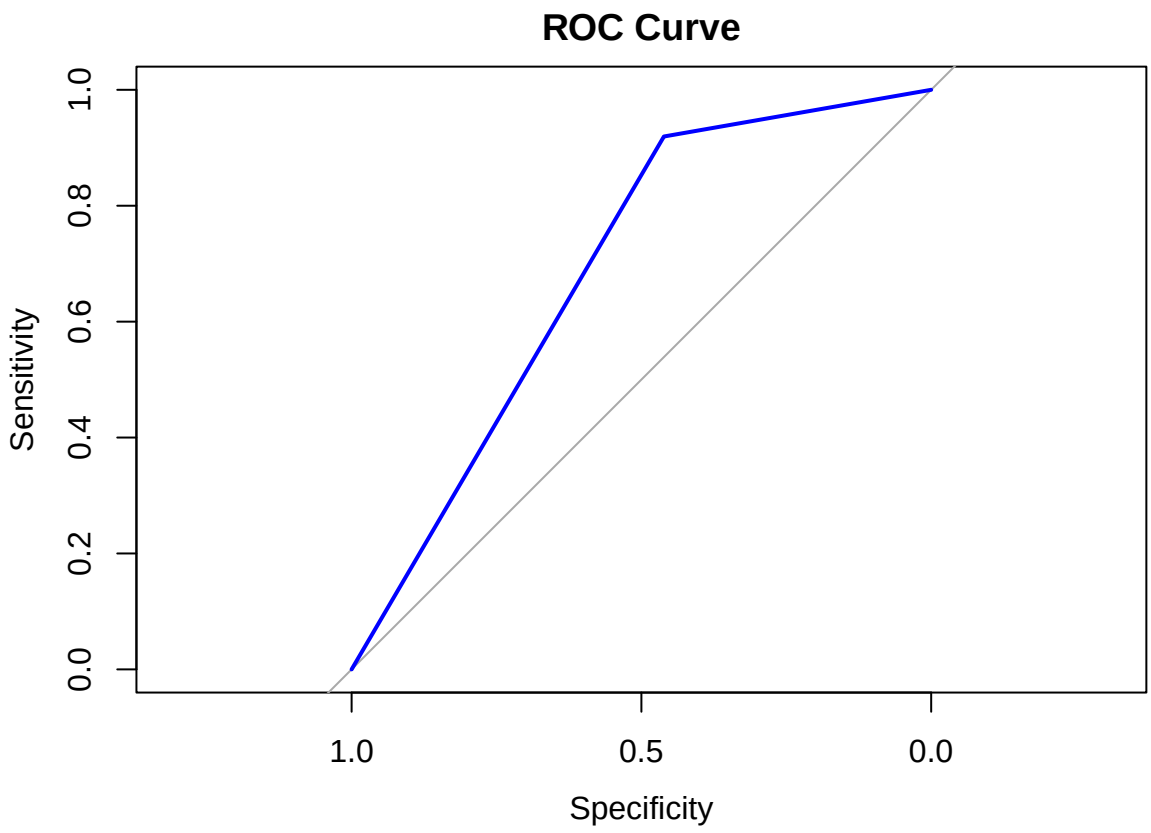



```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 100
## [1] "is: "
## [1] 0.4018519
```

```
k = 50
set.seed(555)
knn_fit <- knn(train = df_tr.vars, test = df_tr.vars, cl = df_tr.01, k = k)
knn_cv <- knn.cv(train = df_tr.vars, cl = df_tr.01, k = k_vec[j])

predictions_knn <- knn(train = df_tr.vars, test = df_test.vars, cl = df_tr.01, k = k)
predictions_knn <- as.integer(as.character(predictions_knn))

roc_curve_knn <- roc(df_test.01, predictions_knn)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(roc_curve_knn, main = "ROC Curve", col = "blue")
```




```
predicted_classes_knn <- ifelse(predictions_knn >= 0.5, 1, 0)
misclassification_rate_knn <- mean(predicted_classes_knn != df_test.01)
print("missclassification rate for LASSO ussing c = 1/2 and k = to : ");print(k);
```

```
## [1] "missclassification rate for LASSO ussing c = 1/2 and k = to : "
## [1] 50
```

```
  print("is: "); print(misclassification_rate_knn )
```

```
## [1] "is: "
## [1] 0.3861111
```