SVM Classification

Linus Fackler, Fernando Colman

Load the data & library

This data set was already split into test and train. The data represents the number of times a marketing person contacted a client about a product (bank term deposit) subscription. It can be found here: https://www.kaggle.com/datasets/dev523/ml-marathon-dataset-by-azure-developer-community

```
library(e1071)
train <- read.csv("data.csv", header = TRUE)
test <- read.csv("test_data.csv", header = TRUE)
str(train)
```

```
## 'data.frame':    8371 obs. of  17 variables:
##  $ age      : int  38 41 39 49 37 40 54 48 29 52 ...
##  $ job      : chr  "technician" "housemaid" "management" "blue-collar" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "tertiary" "primary" "tertiary" "primary" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  127 365 2454 6215 1694 -666 0 1730 560 431 ...
##  $ housing  : chr  "yes" "no" "yes" "yes" ...
##  $ loan     : chr  "no" "no" "no" "no" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "cellular" ...
##  $ day      : int  14 8 4 11 29 27 5 4 5 26 ...
##  $ month    : chr  "oct" "aug" "may" "may" ...
##  $ duration : int  113 203 716 549 404 107 744 361 459 73 ...
##  $ campaign : int  1 5 3 1 2 2 3 1 1 17 ...
##  $ pdays    : int  50 -1 263 -1 251 -1 -1 89 307 -1 ...
##  $ previous : int  2 0 2 0 6 0 0 3 1 0 ...
##  $ poutcome : chr  "success" "unknown" "failure" "unknown" ...
##  $ deposit  : chr  "no" "no" "yes" "no" ...
```

```
str(test)
```

```
## 'data.frame':    2791 obs. of  16 variables:
##  $ age      : int  31 49 51 33 34 49 22 42 27 49 ...
##  $ job      : chr  "blue-collar" "blue-collar" "self-employed"
"technician" ...
##  $ marital  : chr  "single" "married" "single" "married" ...
##  $ education: chr  "secondary" "primary" "tertiary" "secondary" ...
##  $ default  : chr  "yes" "no" "no" "no" ...
##  $ balance  : int  477 599 400 488 40 2999 33 419 446 598 ...
##  $ housing  : chr  "no" "no" "no" "yes" ...
##  $ loan     : chr  "no" "no" "yes" "no" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
```

```
##  $ day      : int  20 23 27 8 5 17 12 14 23 15 ...
##  $ month    : chr  "nov" "jul" "may" "may" ...
##  $ duration : int  426 464 200 703 125 717 369 96 205 219 ...
##  $ campaign : int  2 1 1 1 2 1 1 4 3 6 ...
##  $ pdays    : int  189 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  6 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : chr  "failure" "unknown" "unknown" "unknown" ...
```

Data Cleaning

```
sapply(df, function(x) sum(is.na(x)==TRUE))

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'language'

##   x df1 df2 ncp log
##   0   0   0   0   0
```

```
sapply(df, function(x) sum(is.na(x)==TRUE))

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'symbol'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
'language'

##   x df1 df2 ncp log
##   0   0   0   0   0
```

No NA's in either data set.

Factorizing all chr data types in train:

```
train$job <- factor(train$job)
train$marital <- factor(train$marital)
train$education <- factor(train$education)
train$default <- factor(train$default)
train$housing <- factor(train$housing)
train$loan <- factor(train$loan)
train$contact <- factor(train$contact)
train$poutcome <- factor(train$poutcome)
train$deposit <- factor(train$deposit)

str(train)
```

```
## 'data.frame':    8371 obs. of  17 variables:
##  $ age      : int  38 41 39 49 37 40 54 48 29 52 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 10 4 5 2 8 1
10 2 5 8 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 2 2 2
3 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 3 1 3 1 2 2 2 2
4 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  127 365 2454 6215 1694 -666 0 1730 560 431 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 1 2 1 1 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 1 1 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 1 1 3 2
1 1 1 ...
##  $ day      : int  14 8 4 11 29 27 5 4 5 26 ...
##  $ month    : chr  "oct" "aug" "may" "may" ...
##  $ duration : int  113 203 716 549 404 107 744 361 459 73 ...
##  $ campaign : int  1 5 3 1 2 2 3 1 1 17 ...
##  $ pdays    : int  50 -1 263 -1 251 -1 -1 89 307 -1 ...
##  $ previous : int  2 0 2 0 6 0 0 3 1 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 3 4 1 4 1 4 4 3 1 4
...
##  $ deposit  : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 2 2 1 ...
```

For month, we want it to be factorized in order, so we will change it to integer values representing each month.

```
unique(train$month)
```

```
##  [1] "oct" "aug" "may" "jan" "feb" "apr" "jul" "jun" "nov" "sep" "mar"
"dec"
```

```
train$month[train$month == "jan"] <- "1"
train$month[train$month == "feb"] <- "2"
train$month[train$month == "mar"] <- "3"
train$month[train$month == "apr"] <- "4"
train$month[train$month == "may"] <- "5"
train$month[train$month == "jun"] <- "6"
train$month[train$month == "jul"] <- "7"
```

```
train$month[train$month == "aug"] <- "8"
train$month[train$month == "sep"] <- "9"
train$month[train$month == "oct"] <- "10"
train$month[train$month == "nov"] <- "11"
train$month[train$month == "dec"] <- "12"
head(train$month, 40)
```

```
##  [1] "10" "8"  "5"  "5"  "1"  "5"  "2"  "5"  "4"  "8"  "2"  "8"  "7"  "6"
"6"
## [16] "1"  "5"  "8"  "5"  "7"  "6"  "6"  "8"  "2"  "5"  "7"  "8"  "8"  "6"
"4"
## [31] "8"  "8"  "11" "5"  "6"  "8"  "8"  "8"  "4"  "8"
```

Now, changing it to integers.

```
train <- transform(train, month = as.integer(month))
str(train)
```

```
## 'data.frame':    8371 obs. of  17 variables:
##  $ age      : int  38 41 39 49 37 40 54 48 29 52 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 10 4 5 2 8 1
10 2 5 8 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 2 2 2
3 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 3 1 3 1 2 2 2 2
4 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  127 365 2454 6215 1694 -666 0 1730 560 431 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 1 2 1 1 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 1 1 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 1 1 3 2
1 1 1 ...
##  $ day      : int  14 8 4 11 29 27 5 4 5 26 ...
##  $ month    : int  10 8 5 5 1 5 2 5 4 8 ...
##  $ duration : int  113 203 716 549 404 107 744 361 459 73 ...
##  $ campaign : int  1 5 3 1 2 2 3 1 1 17 ...
##  $ pdays    : int  50 -1 263 -1 251 -1 -1 89 307 -1 ...
##  $ previous : int  2 0 2 0 6 0 0 3 1 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 3 4 1 4 1 4 4 3 1 4
...
##  $ deposit  : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 2 2 1 ...
```

Factorizing all chr data types in test:

```
test$job <- factor(test$job)
test$marital <- factor(test$marital)
test$education <- factor(test$education)
test$default <- factor(test$default)
test$housing <- factor(test$housing)
test$loan <- factor(test$loan)
test$contact <- factor(test$contact)
```

```
test$poutcome <- factor(test$poutcome)

str(test)
```

```
## 'data.frame':    2791 obs. of  16 variables:
##  $ age      : int  31 49 51 33 34 49 22 42 27 49 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 2 2 7 10 1 1
1 2 5 2 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 3 2 3 2 2 1 3 1
3 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 2 1 3 2 2 2 2 1
3 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  477 599 400 488 40 2999 33 419 446 598 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 1 1 1 2 2 2 1 2 1 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 2 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 2 1 1
3 1 1 ...
##  $ day      : int  20 23 27 8 5 17 12 14 23 15 ...
##  $ month    : chr  "nov" "jul" "may" "may" ...
##  $ duration : int  426 464 200 703 125 717 369 96 205 219 ...
##  $ campaign : int  2 1 1 1 2 1 1 4 3 6 ...
##  $ pdays    : int  189 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  6 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 1 4 4 4 4 4 4 4 4 4
...
```

Same thing with months to integers as in train:

```
unique(test$month)
```

```
##  [1] "nov" "jul" "may" "feb" "jun" "oct" "aug" "mar" "apr" "sep" "jan"
"dec"
```

```
test$month[test$month == "jan"] <- "1"
test$month[test$month == "feb"] <- "2"
test$month[test$month == "mar"] <- "3"
test$month[test$month == "apr"] <- "4"
test$month[test$month == "may"] <- "5"
test$month[test$month == "jun"] <- "6"
test$month[test$month == "jul"] <- "7"
test$month[test$month == "aug"] <- "8"
test$month[test$month == "sep"] <- "9"
test$month[test$month == "oct"] <- "10"
test$month[test$month == "nov"] <- "11"
test$month[test$month == "dec"] <- "12"
head(train$month, 40)
```

```
##  [1] 10  8  5  5  1  5  2  5  4  8  2  8  7  6  6  1  5  8  5  7  6  6  8
2  5
## [26]  7  8  8  6  4  8  8 11  5  6  8  8  8  4  8
```

Now, changing it to integers.

```
test <- transform(test, month = as.integer(month))
str(test)
```

```
## 'data.frame':    2791 obs. of  16 variables:
##  $ age      : int  31 49 51 33 34 49 22 42 27 49 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 2 2 7 10 1 1
1 2 5 2 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 3 2 3 2 2 1 3 1
3 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 2 1 3 2 2 2 2 1
3 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  477 599 400 488 40 2999 33 419 446 598 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 1 1 1 2 2 2 1 2 1 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 2 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 2 1 1
3 1 1 ...
##  $ day      : int  20 23 27 8 5 17 12 14 23 15 ...
##  $ month    : int  11 7 5 5 5 5 2 5 6 5 ...
##  $ duration : int  426 464 200 703 125 717 369 96 205 219 ...
##  $ campaign : int  2 1 1 1 2 1 1 4 3 6 ...
##  $ pdays    : int  189 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  6 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 1 4 4 4 4 4 4 4 4 4
...
```

Other than that, the data looks pretty clean. The reason why the data was already split was, that there is no “deposit” values in the test data, since this is our target value.

Data Exploration of Training Data

First, we will explore the training data statistically and graphically

```
print(paste("Number of Rows: ", nrow(train)))
```

```
## [1] "Number of Rows:  8371"
```

```
print(paste("Average Age: ", mean(train$age)))
```

```
## [1] "Average Age:  41.1974674471389"
```

```
print(paste("Median Age: ", median(train$age)))
```

```
## [1] "Median Age:  39"
```

```
print(paste("Average Day: ", mean(train$day)))
```

```
## [1] "Average Day:  15.5884601600765"
```

```
print(paste("Median Day: ", median(train$day)))
```

```
## [1] "Median Day:  15"
```

```
print(paste("Average Balance: ", mean(train$balance)))
```

```
## [1] "Average Balance:  1517.81113367579"
```

```
print(paste("Median Balance: ", median(train$balance)))
```

```
## [1] "Median Balance:  532"
```

```
print(paste("Average Duration: ", mean(train$duration)))
```

```
## [1] "Average Duration:  372.898697885557"
```

```
print(paste("Median Duration: ", median(train$duration)))
```

```
## [1] "Median Duration:  255"
```

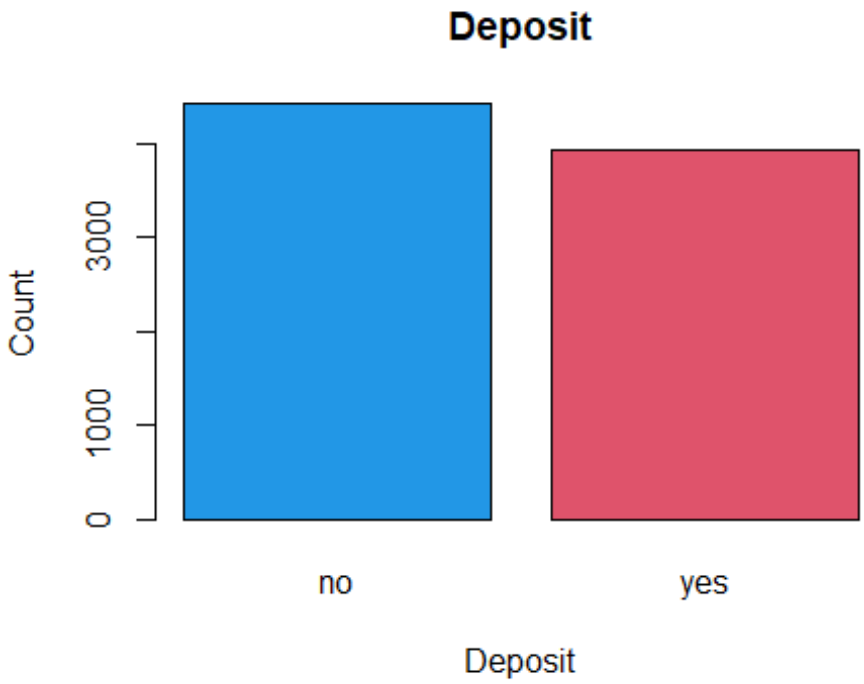
Plots

In the following section I will show the affects of different factors on the deposit value.

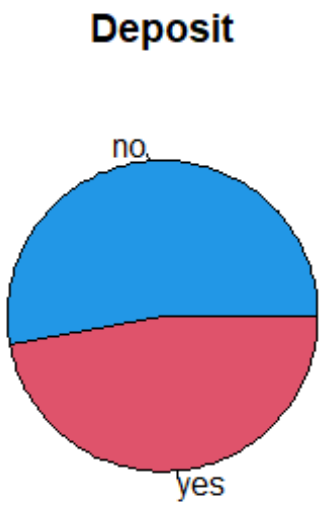
Deposit and no deposit

Now, we will use a barplot to see how many yes and no's Deposit has, which is our target values.

```
counts <- table(train$deposit)
barplot(counts, ylab = "Count", xlab = "Deposit", main="Deposit", col = c(4,
2))
```

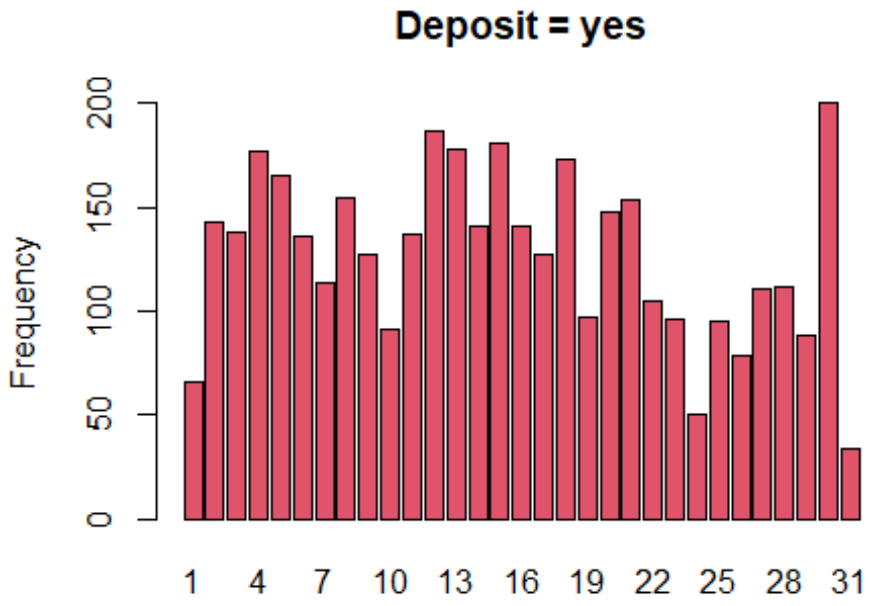



```
pie(counts, main="Deposit", col = c(4, 2))
```

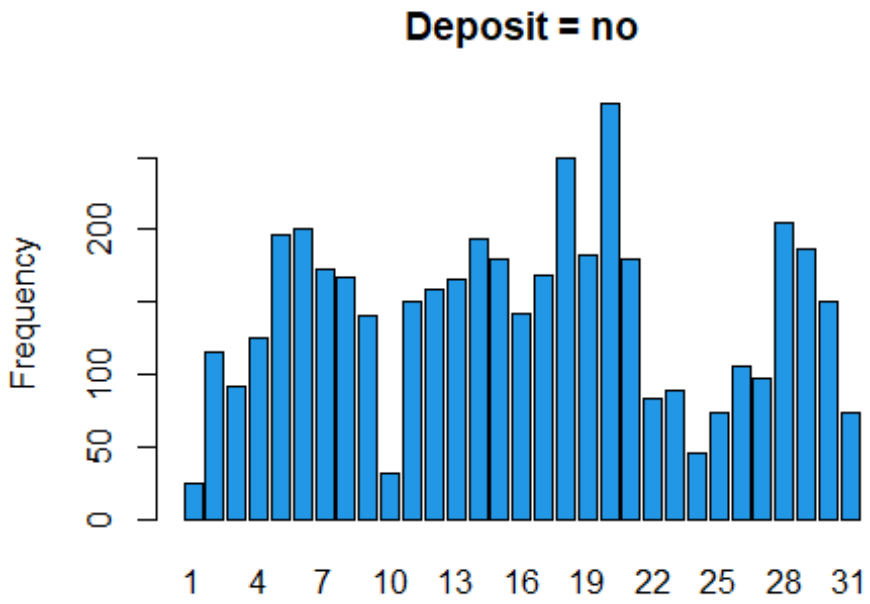


Day and Deposit

```
counts_yes <- table(train$day[train$deposit=="yes"])
counts_no <- table(train$day[train$deposit=="no"])
barplot(counts_yes, ylab = "Frequency", main="Deposit = yes", col = 2)
```



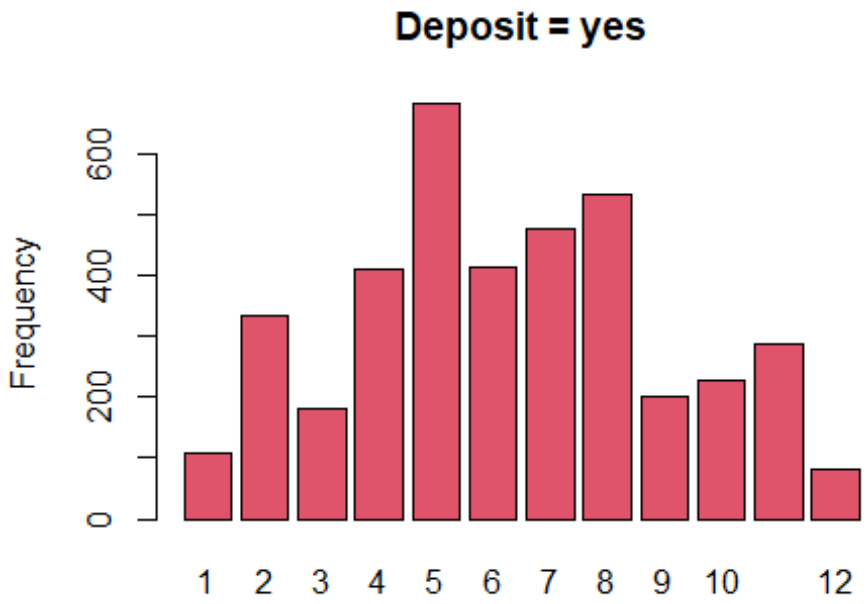

```
barplot(counts_no, ylab = "Frequency", main="Deposit = no", col = 4)
```



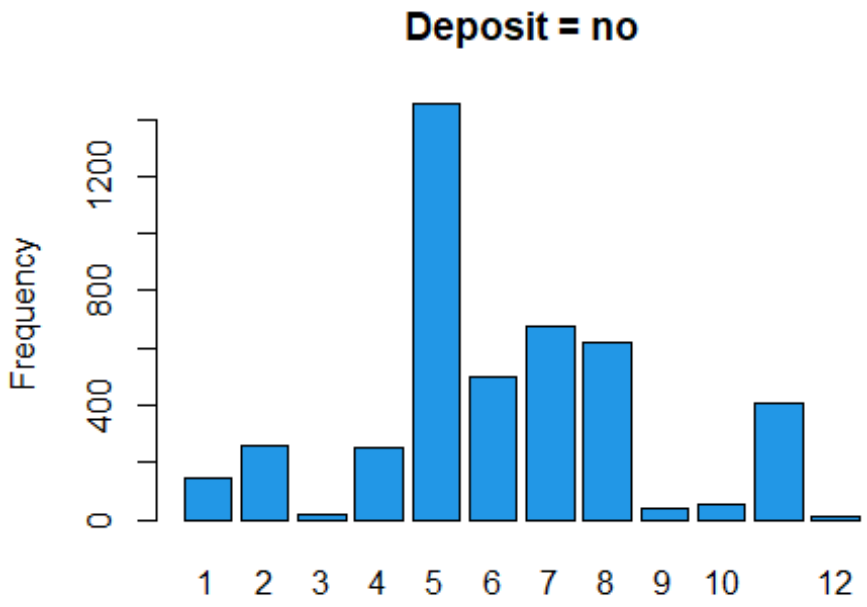
Month and Deposit

How many yes and no per month were made.

```
counts_yes <- table(train$month[train$deposit=="yes"])
counts_no <- table(train$month[train$deposit=="no"])
barplot(counts_yes, ylab = "Frequency", main="Deposit = yes", col = 2)
```



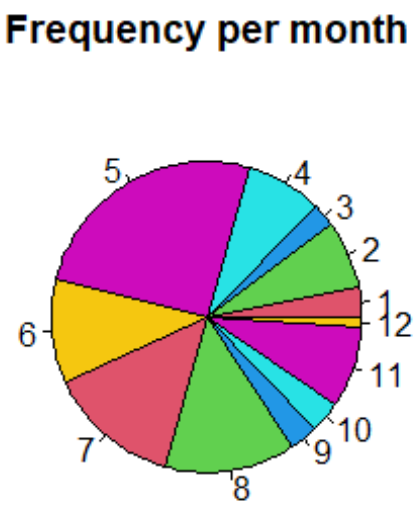

```
barplot(counts_no, ylab = "Frequency", main="Deposit = no", col = 4)
```



Deposits per month

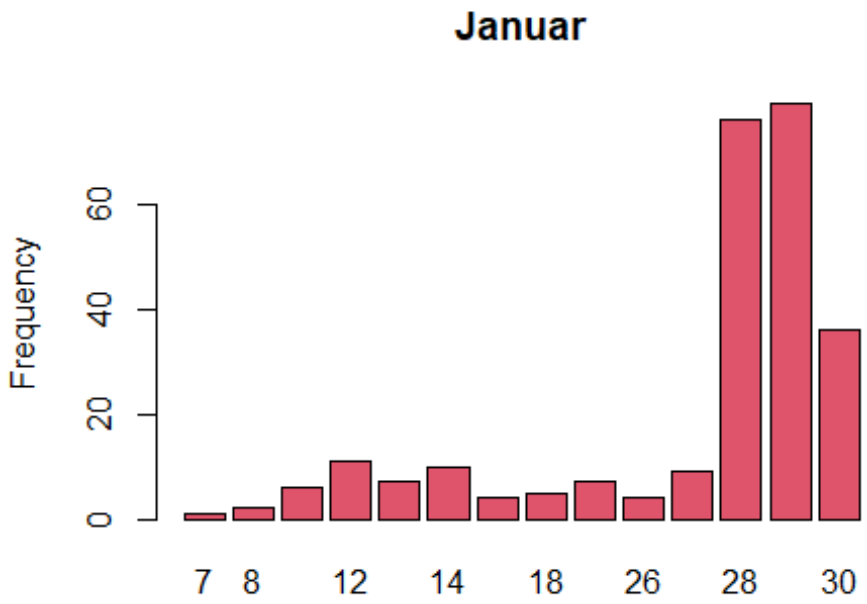
This pie chart shows us how many deposits per month are made percentage-wise.

```
counts_month <- table(train$month)
pie(counts_month, main="Frequency per month", col =
c(2,3,4,5,6,7,10,11,12,13,14,15))
```

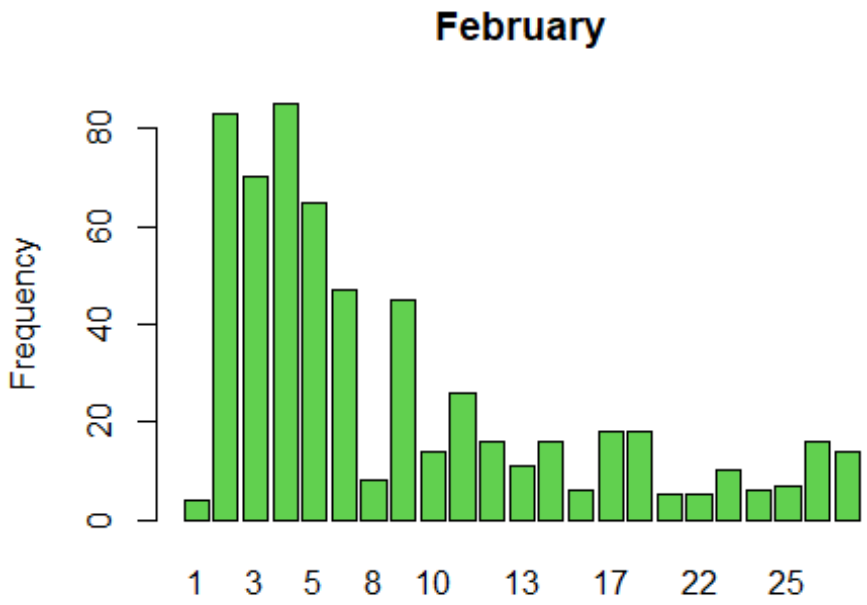


Day and Month

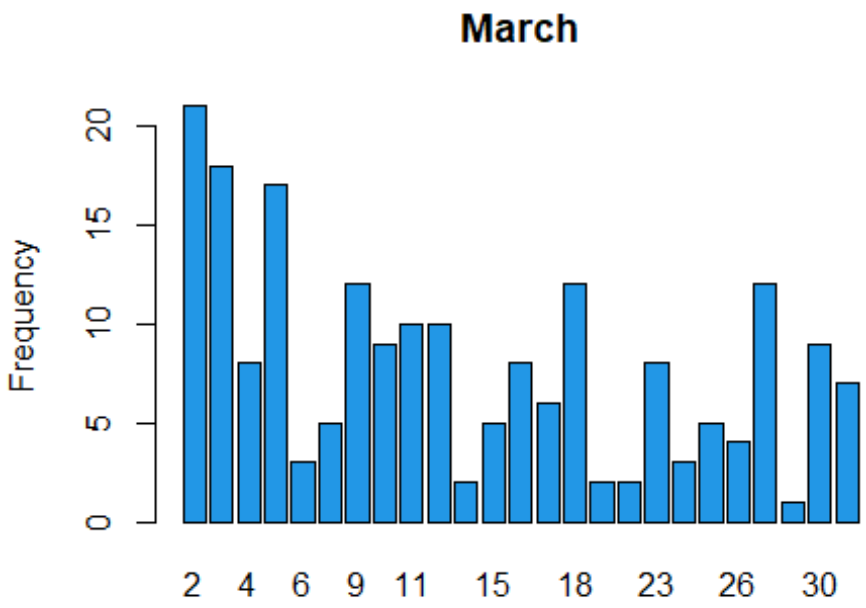
```
counts_1 <- table(train$day[train$month == 1])
barplot(counts_1, ylab = "Frequency", main = "Januar", col = 2)
```



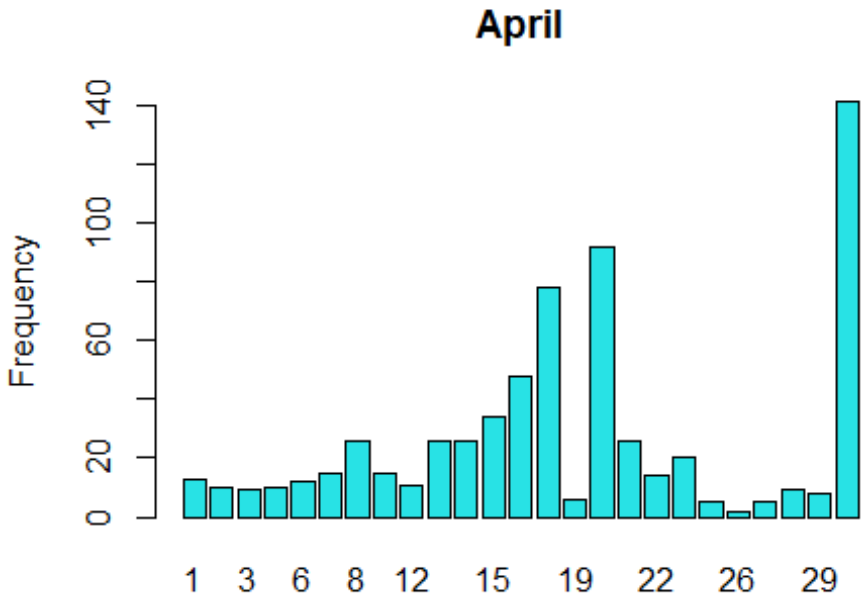

```r
counts_2 <- table(train$day[train$month == 2])
barplot(counts_2, ylab = "Frequency", main = "February", col = 3)
```



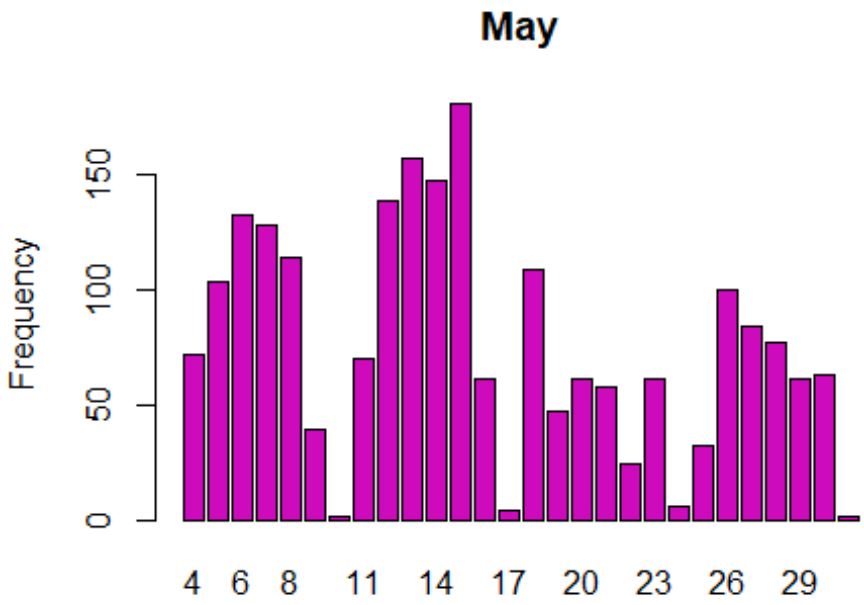
```r
counts_3 <- table(train$day[train$month == 3])
barplot(counts_3, ylab = "Frequency", main = "March", col = 4)
```



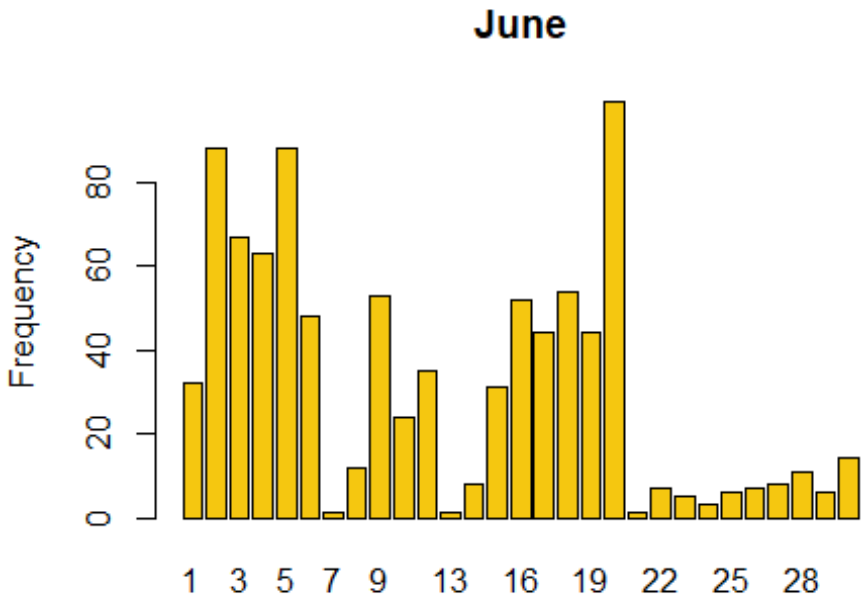

```r
counts_4 <- table(train$day[train$month == 4])
barplot(counts_4, ylab = "Frequency", main = "April", col = 5)
```



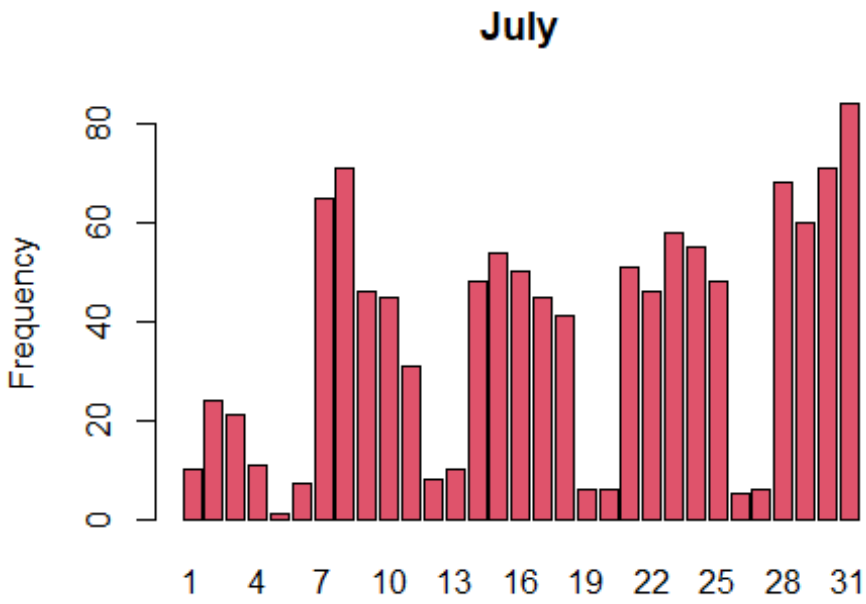

```
counts_5 <- table(train$day[train$month == 5])
barplot(counts_5, ylab = "Frequency", main = "May", col = 6)
```



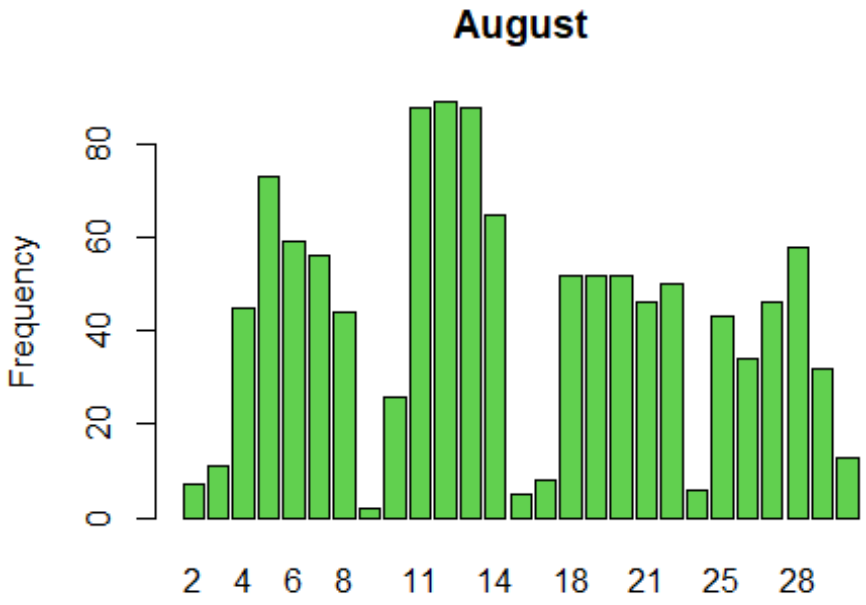
```
counts_6 <- table(train$day[train$month == 6])
barplot(counts_6, ylab = "Frequency", main = "June", col = 7)
```



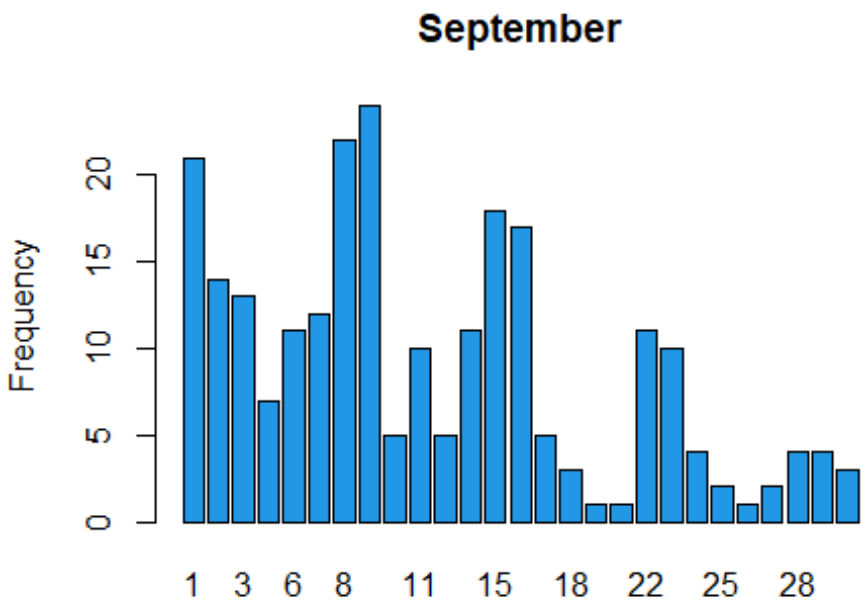

```
counts_7 <- table(train$day[train$month == 7])
barplot(counts_7, ylab = "Frequency", main = "July", col = 10)
```



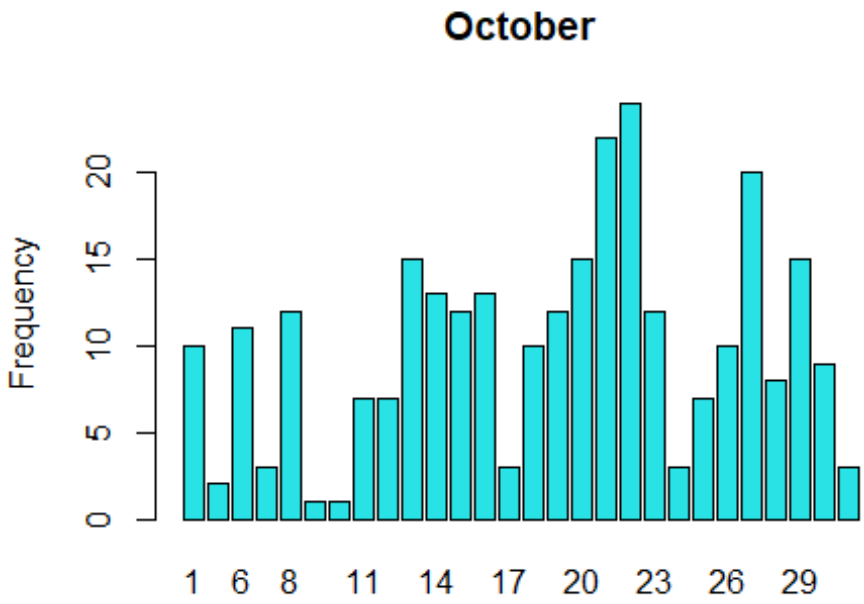

```
counts_8 <- table(train$day[train$month == 8])
barplot(counts_8, ylab = "Frequency", main = "August", col = 11)
```



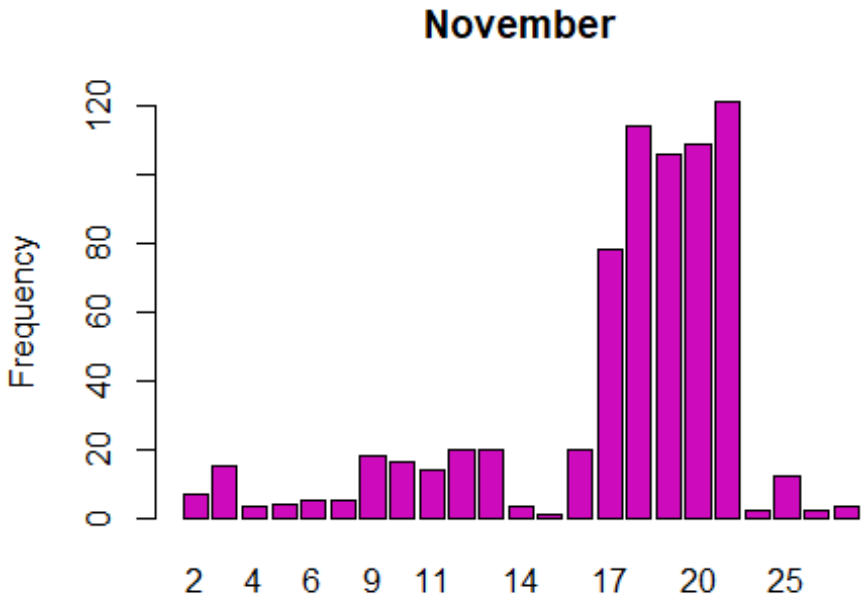
```
counts_9 <- table(train$day[train$month == 9])
barplot(counts_9, ylab = "Frequency", main = "September", col = 12)
```



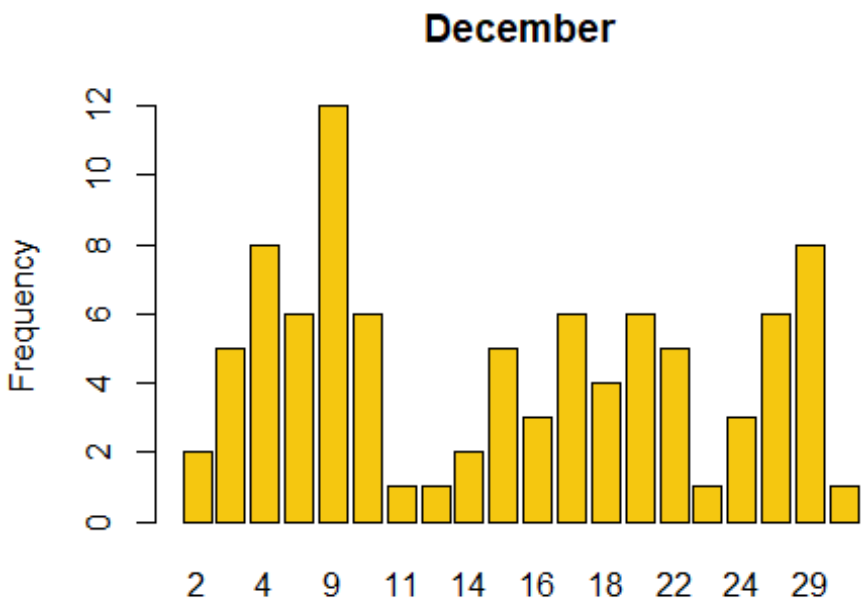

```
counts_10 <- table(train$day[train$month == 10])
barplot(counts_10, ylab = "Frequency", main = "October", col = 13)
```




```
counts_11 <- table(train$day[train$month == 11])
barplot(counts_11, ylab = "Frequency", main = "November", col = 14)
```



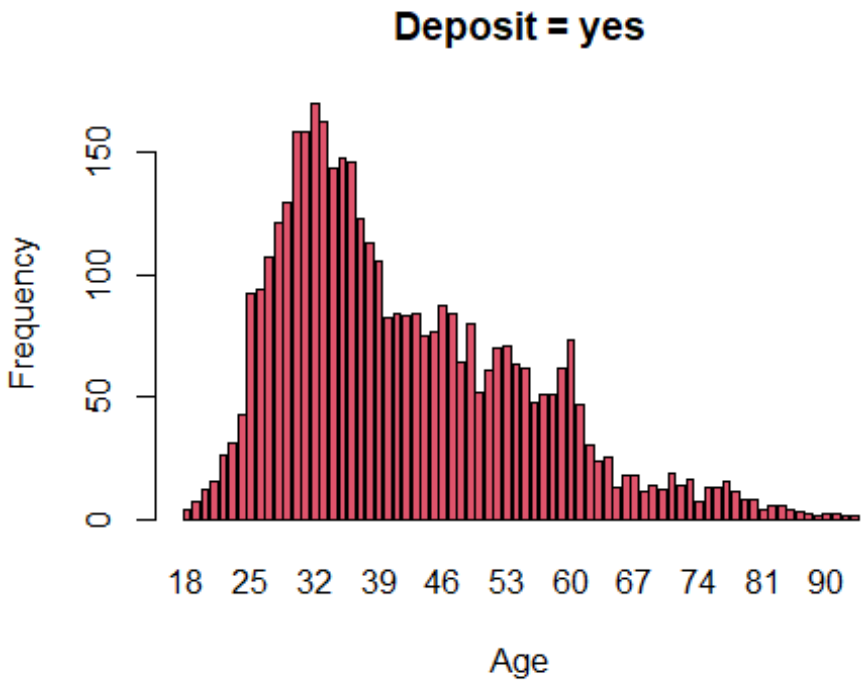
```
counts_12 <- table(train$day[train$month == 12])
barplot(counts_12, ylab = "Frequency", main = "December", col = 15)
```



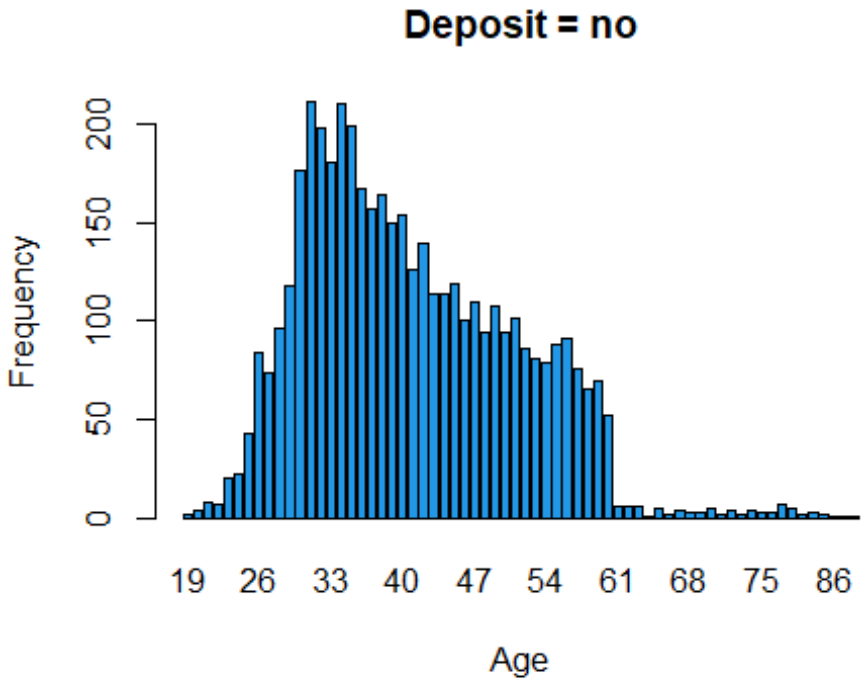
Age and Deposits

This shows us the number of deposits made according to customers age.

```
counts_yes <- table(train$age[train$deposit=="yes"])
counts_no <- table(train$age[train$deposit=="no"])
barplot(counts_yes, ylab = "Frequency", xlab = "Age", main="Deposit = yes",
col = 2)
```



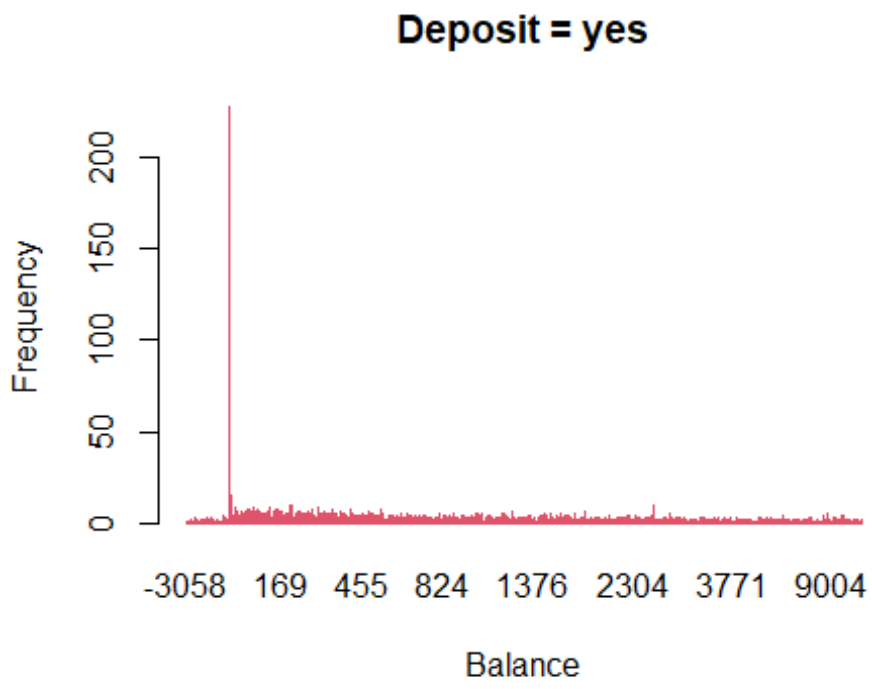

```
barplot(counts_no, ylab = "Frequency", xlab = "Age", main="Deposit = no", col
= 4)
```



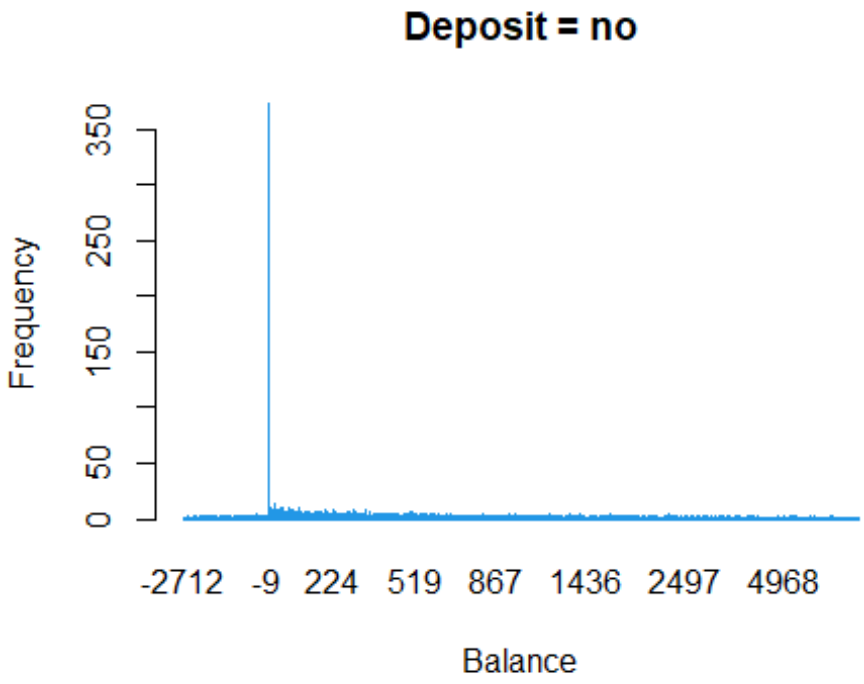
Balance and Deposit

Number of deposits compared to the balance of the customer.

```
counts_yes <- table(train$balance[train$deposit=="yes"])
counts_no <- table(train$balance[train$deposit=="no"])
barplot(counts_yes, ylab = "Frequency", xlab = "Balance", main="Deposit =
yes", border = 2)
```

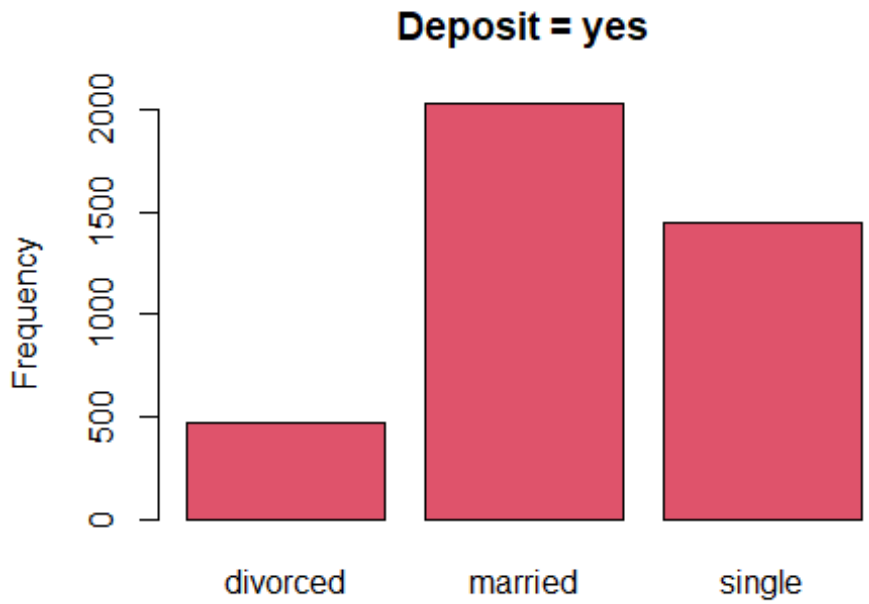



```
barplot(counts_no, ylab = "Frequency", xlab = "Balance", main="Deposit = no",
border = 4)
```

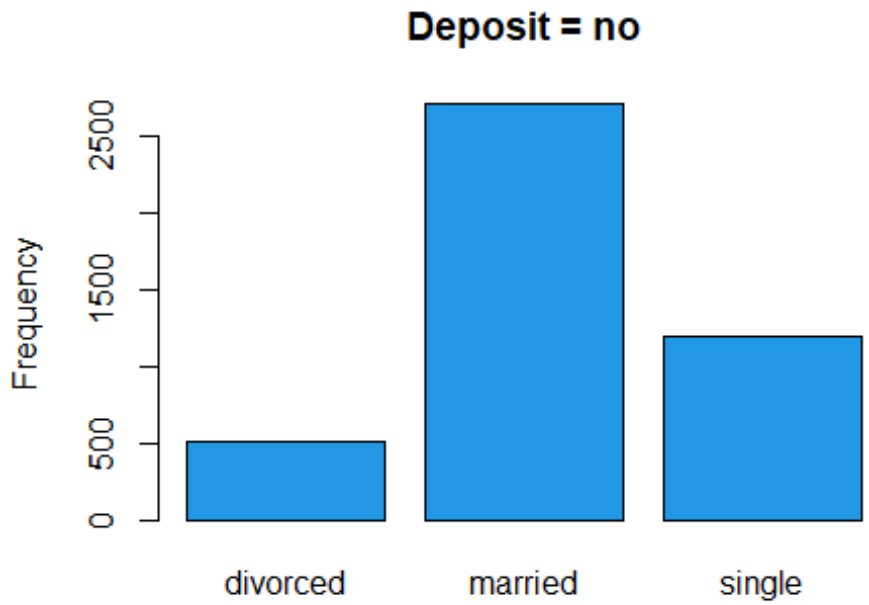


Marital Status and Deposits

```
counts_yes <- table(train$marital[train$deposit == "yes"])
counts_no <- table(train$marital[train$deposit == "no"])
barplot(counts_yes, ylab = "Frequency", main="Deposit = yes", col = 2)
```

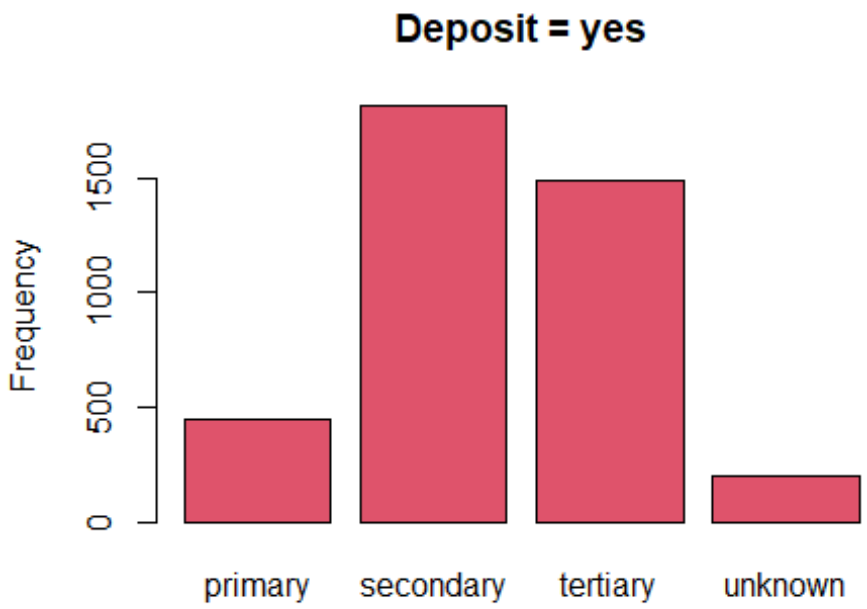



```
barplot(counts_no, ylab = "Frequency", main="Deposit = no", col = 4)
```

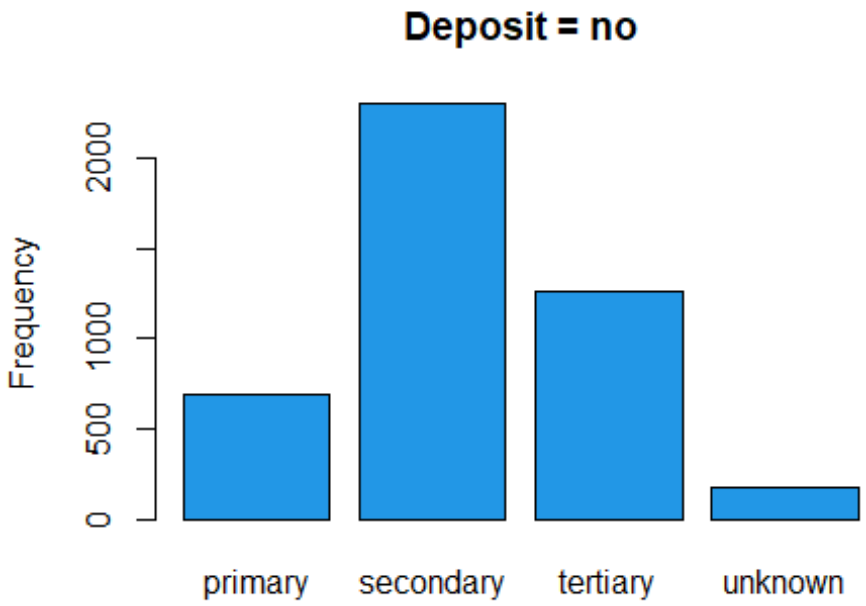


Education Status and Deposits

```
counts_yes <- table(train$education[train$deposit == "yes"])
counts_no <- table(train$education[train$deposit == "no"])
barplot(counts_yes, ylab = "Frequency", main="Deposit = yes", col = 2)
```

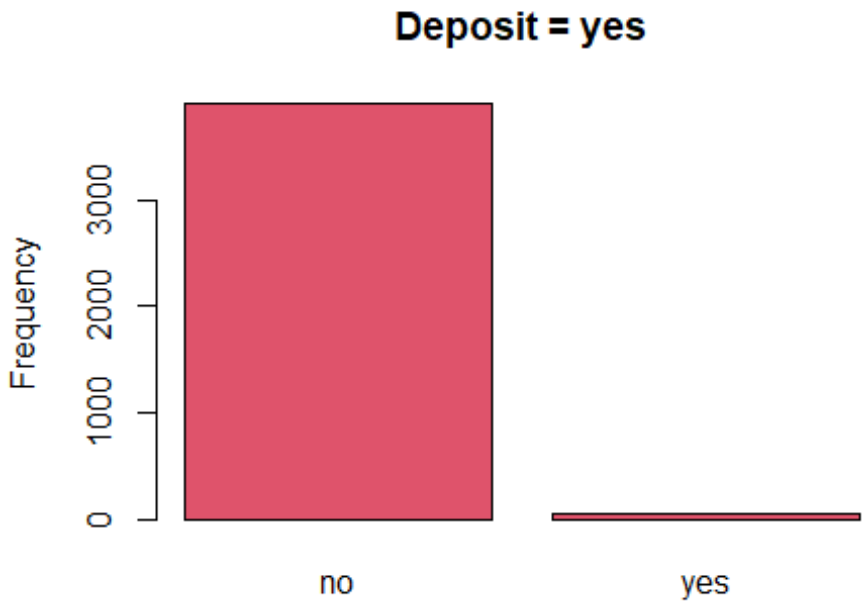



```
barplot(counts_no, ylab = "Frequency", main="Deposit = no", col = 4)
```

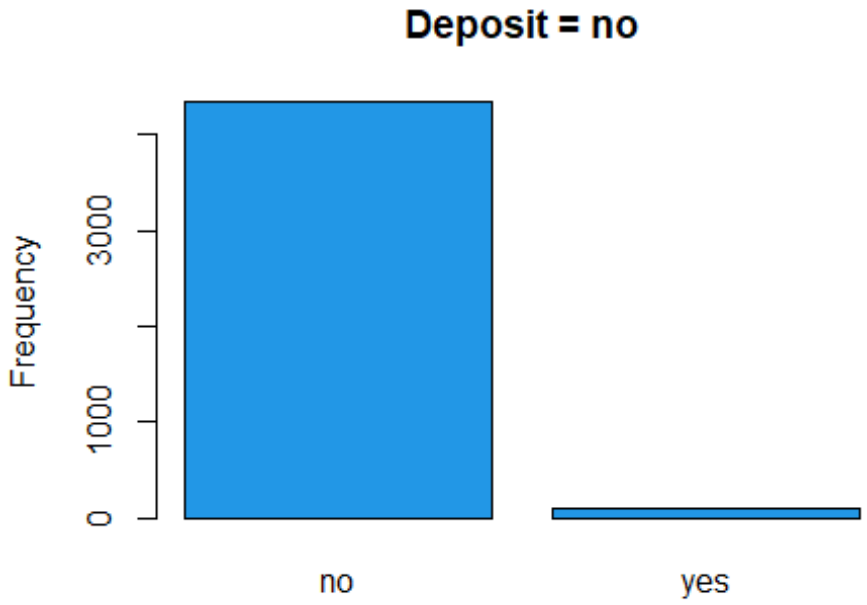


Default and Deposits

```
counts_yes <- table(train$default[train$deposit == "yes"])
counts_no <- table(train$default[train$deposit == "no"])
barplot(counts_yes, ylab = "Frequency", main="Deposit = yes", col = 2)
```

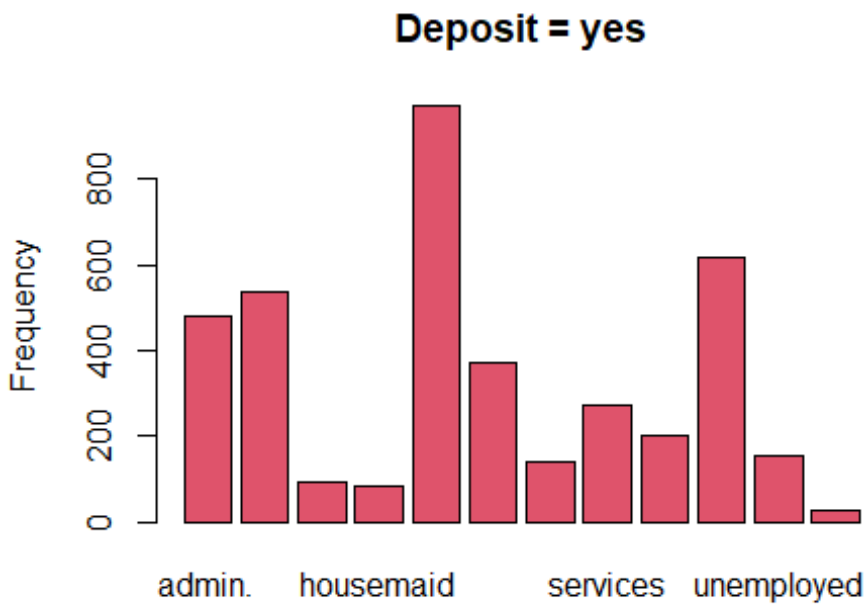



```
barplot(counts_no, ylab = "Frequency", main="Deposit = no", col = 4)
```

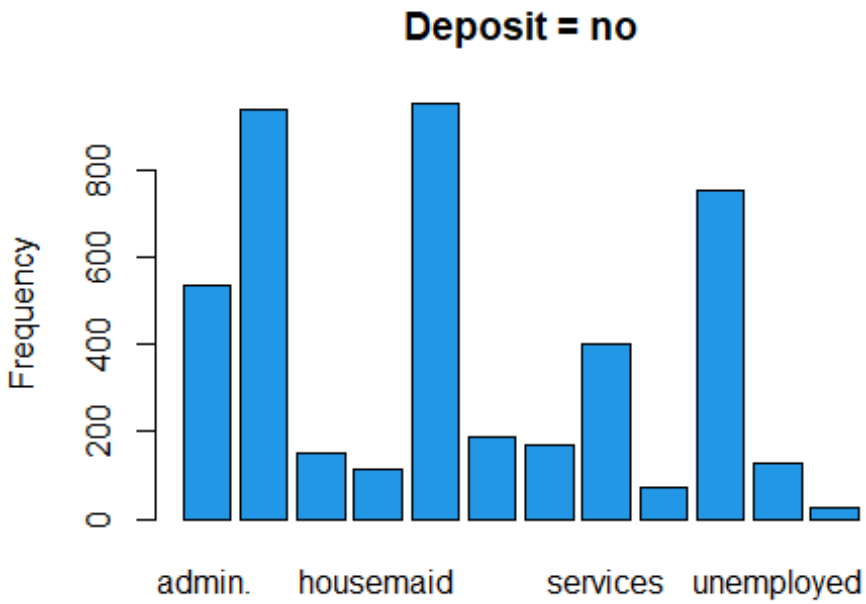


Job and Deposits

```
counts_yes <- table(train$job[train$deposit == "yes"])
counts_no <- table(train$job[train$deposit == "no"])
barplot(counts_yes, ylab = "Frequency", main="Deposit = yes", col = 2)
```




```
barplot(counts_no, ylab = "Frequency", main="Deposit = no", col = 4)
```



Linear SVM

I will first split our data sets into smaller ones, since my computer cannot run/takes too long for the current number of observations.

```
trainsmall <- head(train, 3000)
testsmall <- tail(train, 600)
valdsmall <- train[3001:3601,]

svm1 <- svm(deposit~., data=trainsmall, kernel="linear", cost=10, scale=TRUE)
summary(svm1)

##
## Call:
## svm(formula = deposit ~ ., data = trainsmall, kernel = "linear",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  1382
##
##  ( 696 686 )
##
```

```
##
## Number of Classes:  2
##
## Levels:
##  no yes
```

Evaluate

```
pred <- predict(svm1, newdata=testsmall)
table(pred, testsmall$deposit)

##
## pred   no yes
##   no  258  69
##   yes  48 225

mean(pred==testsmall$deposit)

## [1] 0.805
```

This algorithm shows a 80.5% accuracy.

Tune

```
tune_svm1 <- tune(svm, deposit~. , data=valdsmall, kernel="linear",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm1)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.2445082
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4710109 0.06025228
## 2 1e-02 0.2611475 0.06807122
## 3 1e-01 0.2445082 0.06944018
## 4 1e+00 0.2511749 0.05912607
## 5 5e+00 0.2478962 0.05621412
## 6 1e+01 0.2545355 0.05698775
## 7 1e+02 0.2545355 0.05698775
```

Evaluate on best linear SVM

```
pred <- predict(tune_svm1$best.model, newdata=testsmall)
table(pred, testsmall$deposit)
```

```
##
## pred   no yes
##   no  236  42
##   yes  70 252

mean(pred==testsmall$deposit)

## [1] 0.8133333
```

Already 2% better.

Polynomial SVM

```
svm2 <- svm(deposit~., data=trainsmall, kernel="polynomial", cost=10,
scale=TRUE)
summary(svm2)

##
## Call:
## svm(formula = deposit ~ ., data = trainsmall, kernel = "polynomial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  1743
##
##  ( 865 878 )
##
##
## Number of Classes:  2
##
## Levels:
##  no yes
```

Evaluate

```
pred <- predict(svm2, newdata=testsmall)
table(pred, testsmall$deposit)

##
## pred   no yes
##   no  252  57
##   yes  54 237

mean(pred==testsmall$deposit)

## [1] 0.815
```

Tune

```
tune_svm2 <- tune(svm, deposit~. , data=valdsmall, kernel="polynomial",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.2561749
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4792350 0.07038783
## 2 1e-02 0.4792350 0.07038783
## 3 1e-01 0.4759016 0.06784615
## 4 1e+00 0.4359563 0.09328272
## 5 5e+00 0.2943169 0.08207712
## 6 1e+01 0.2677869 0.04528353
## 7 1e+02 0.2561749 0.05470006
```

We were already using the best model for polynomial SVM. Cost: 10

Radial SVM

```
svm3 <- svm(deposit~., data=trainsmall, kernel="radial", cost=10, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = deposit ~ ., data = trainsmall, kernel = "radial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  1340
##
##  ( 660 680 )
##
##
## Number of Classes:  2
##
```

```
## Levels:
##  no yes

pred <- predict(svm3, newdata=testsmall)
table(pred, testsmall$deposit)

##
## pred   no yes
##   no  244  54
##   yes  62 240

mean(pred==testsmall$deposit)

## [1] 0.8066667
```

Slightly worse than the best polynomial. Now, let's see if it improves after tuning hyperparameters.

Tune hyperparameters

```
set.seed(1234)
tune.out <- tune(svm, deposit~., data=valdsmall, kernel="radial",
ranges=list(cost=c(0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)))
summary(tune.out)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.2329508
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.4790437 0.05740842
## 2  1e+00   0.5 0.2445902 0.06085981
## 3  1e+01   0.5 0.2329508 0.07028325
## 4  1e+02   0.5 0.2329508 0.07028325
## 5  1e+03   0.5 0.2329508 0.07028325
## 6  1e-01   1.0 0.4790437 0.05740842
## 7  1e+00   1.0 0.3278142 0.06300680
## 8  1e+01   1.0 0.3110656 0.06474945
## 9  1e+02   1.0 0.3110656 0.06474945
## 10 1e+03   1.0 0.3110656 0.06474945
## 11 1e-01   2.0 0.4790437 0.05740842
## 12 1e+00   2.0 0.4307923 0.06910979
## 13 1e+01   2.0 0.4242077 0.05967389
## 14 1e+02   2.0 0.4242077 0.05967389
```

```
## 15 1e+03   2.0 0.4242077 0.05967389
## 16 1e-01   3.0 0.4790437 0.05740842
## 17 1e+00   3.0 0.4574317 0.05687667
## 18 1e+01   3.0 0.4424590 0.06260352
## 19 1e+02   3.0 0.4424590 0.06260352
## 20 1e+03   3.0 0.4424590 0.06260352
## 21 1e-01   4.0 0.4790437 0.05740842
## 22 1e+00   4.0 0.4640710 0.05889403
## 23 1e+01   4.0 0.4607377 0.06170336
## 24 1e+02   4.0 0.4607377 0.06170336
## 25 1e+03   4.0 0.4607377 0.06170336
```

As we can see, we already used the best cost for our radial SVM model.

Analysis

The linear SVM with the best hyperparameters showed an accuracy of 82.5%. Polynomial is second with 81.5%, and radial third with 80.7%.

The linear kernel works best in this dataset, because as we can see in the many many plots I have generated in the evaluation section, the data is linearly separable. There is not many outliers, making the use of radial or polynomial kernels useless, as no improvement will happen, if the line becomes a polynomial graph. The data is very linear already, which is why we get the best results using the linear kernel.

Therefore, the linear kernel works best in this dataset.