HW3

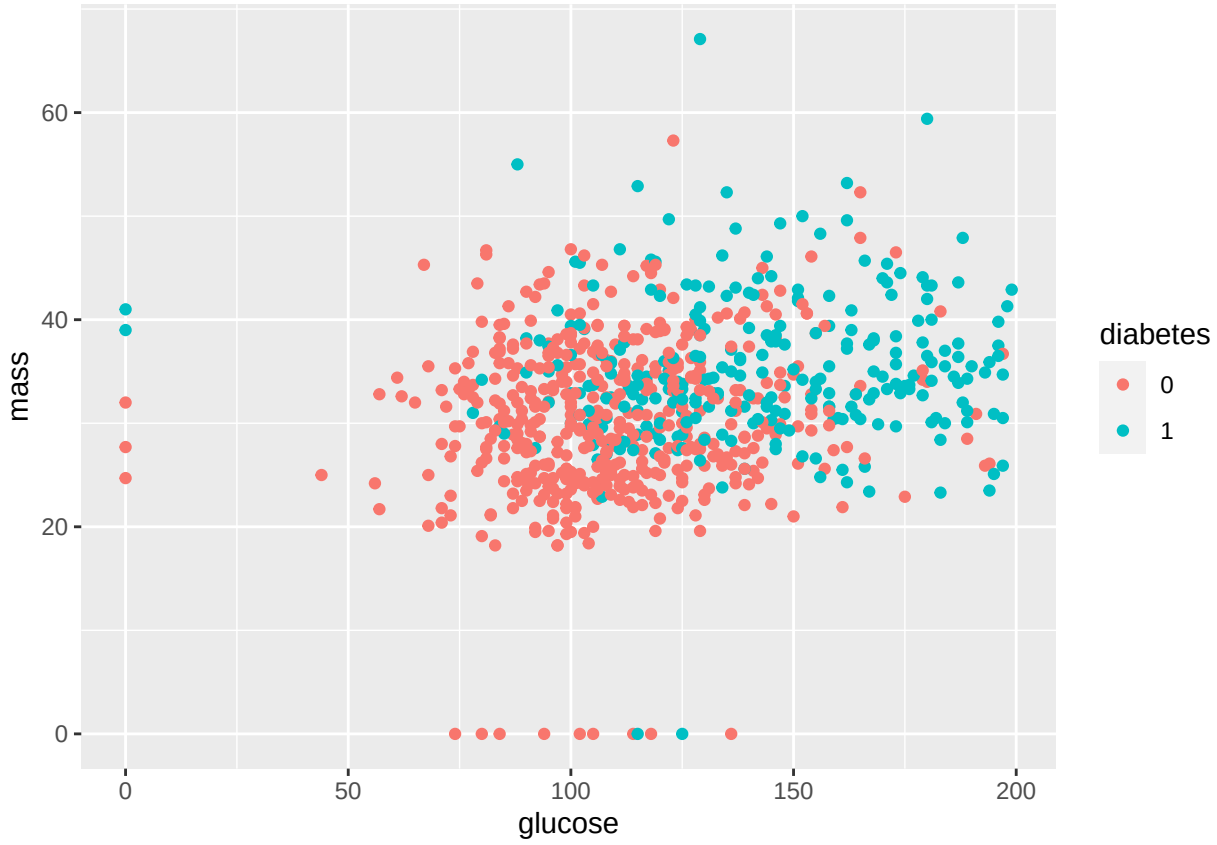
```
required_packages = c("mlbench", "ggplot2", "e1071", "ROCR")
packages_to_install = setdiff(required_packages, installed.packages()[,"Package"])

if (length(packages_to_install)!=0) {
  install.packages(packages_to_install)
}

library(mlbench)
library(ggplot2)
library(e1071)
library(tidyverse)
library(modelr)
library(broom)
library(ROCR)
library(ISLR)

set.seed(0)
```

Question 1. Support Vector Machine.In this question, we'll implement SVM and visualize its decision boundaries as we modify its parameters. We'll reuse "PimaIndiansDiabetes" dataset and the decision boundary plot from homework 1. (20 pts)

1.1 Data Loading & Processing (3 pts)

```
# load in PimaIndiansDiabetes

data(PimaIndiansDiabetes)


# binarize "pos" (1 = 'pos', 0 ='neg') in diabetes variable

PimaIndiansDiabetes$diabetes = as.character(PimaIndiansDiabetes$diabetes)
PimaIndiansDiabetes$diabetes[PimaIndiansDiabetes$diabetes == "pos"] = 1
PimaIndiansDiabetes$diabetes[PimaIndiansDiabetes$diabetes == "neg"] =0
PimaIndiansDiabetes$diabetes = factor(PimaIndiansDiabetes$diabetes)
PimaIndiansDiabetes$pregnant = NULL
PimaIndiansDiabetes$pressure = NULL
PimaIndiansDiabetes$triceps = NULL
PimaIndiansDiabetes$insulin = NULL
PimaIndiansDiabetes$pedigree = NULL
PimaIndiansDiabetes$age = NULL
```

```
# scatter plot glucose vs mass, coloring the points by diabetes

ggplot(PimaIndiansDiabetes, aes(glucose, mass, colour = diabetes)) +
  geom_point()
```




```
# split data into training & testing (70/30 split)

library(caret)
```

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

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

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

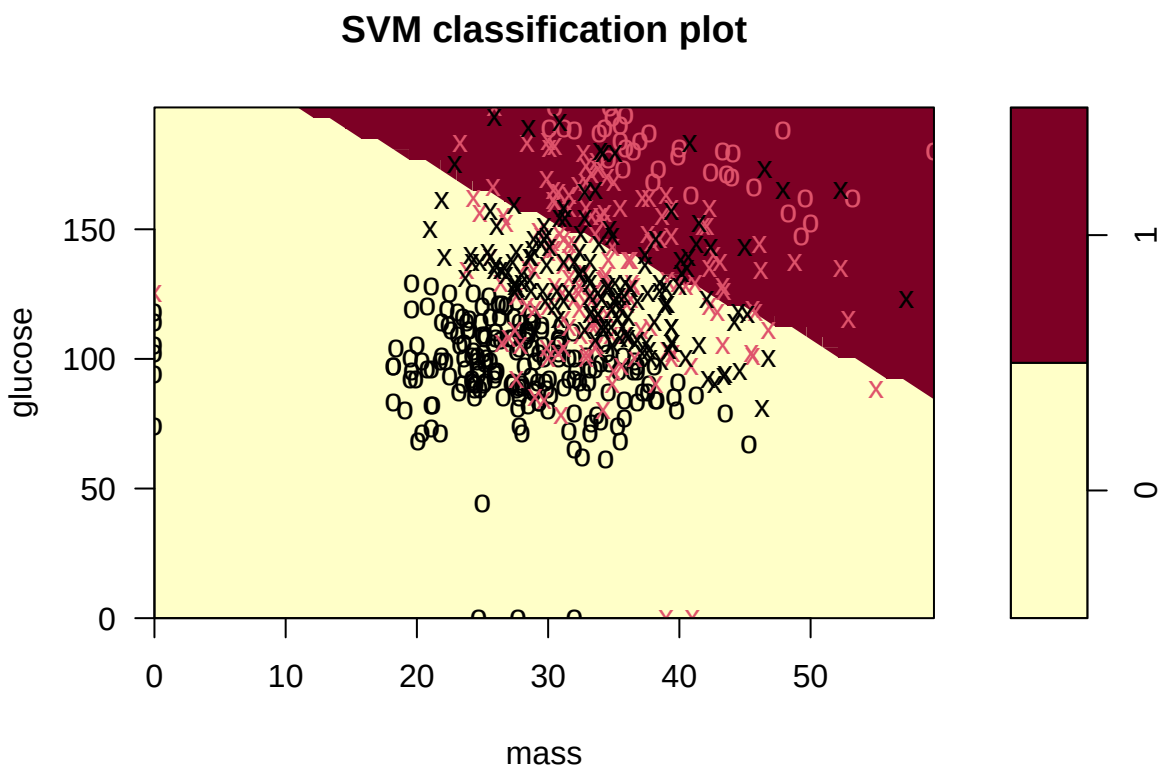
```
split = createDataPartition(y=PimaIndiansDiabetes$diabetes,p = 0.7,list = F,groups = 100)
train = PimaIndiansDiabetes[split,]
test = PimaIndiansDiabetes[-split,]
```

1.2 SVM & Varying Degree of Polynomial Kernel.Let's fit SVM to our data and visualize the decision boundaries as we change the parameters. Please use the following formula when fitting your model: diabetes ~ glucose + mass. (5 pts)

```
# function to plot decision boundary

# make grid
make.grid = function(x, n = 200) {
  grange = apply(x, 2, range)
  x1 = seq(from = grange[1,1], to = grange[2,1], length = n)
  x2 = seq(from = grange[1,2], to = grange[2,2], length = n)
  expand.grid(X1 = x1, X2 = x2)}

grid = make.grid(train)
```

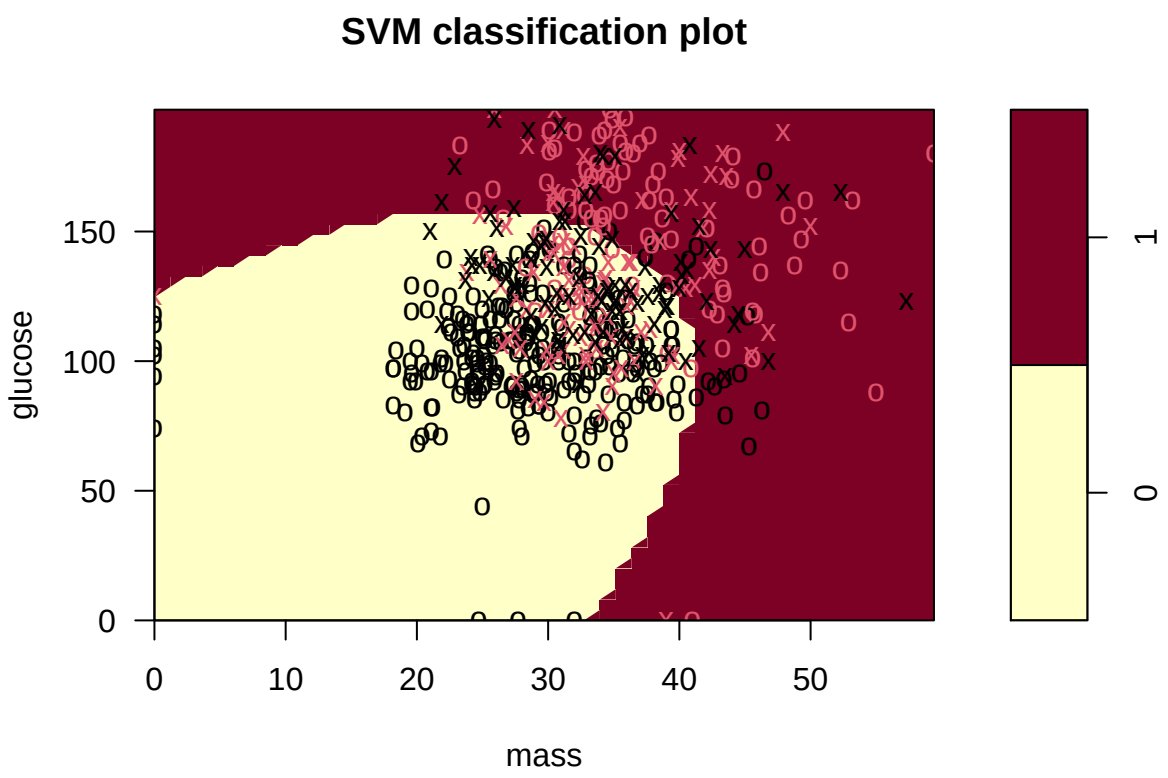
Let's use polynomial kernel and vary its degree from 1-3, the observe our decision boundaries.

```
# fit using "svm", with "polynomial" kernel, degree = 1, cost = 5, type = "C", and scale = F

svmPoly = svm(diabetes~glucose+mass,data = train, kernel='polynomial',scale=F,type='C-classification',d

# visualize the decision boundary of your SVM model

plot(svmPoly, train)
```



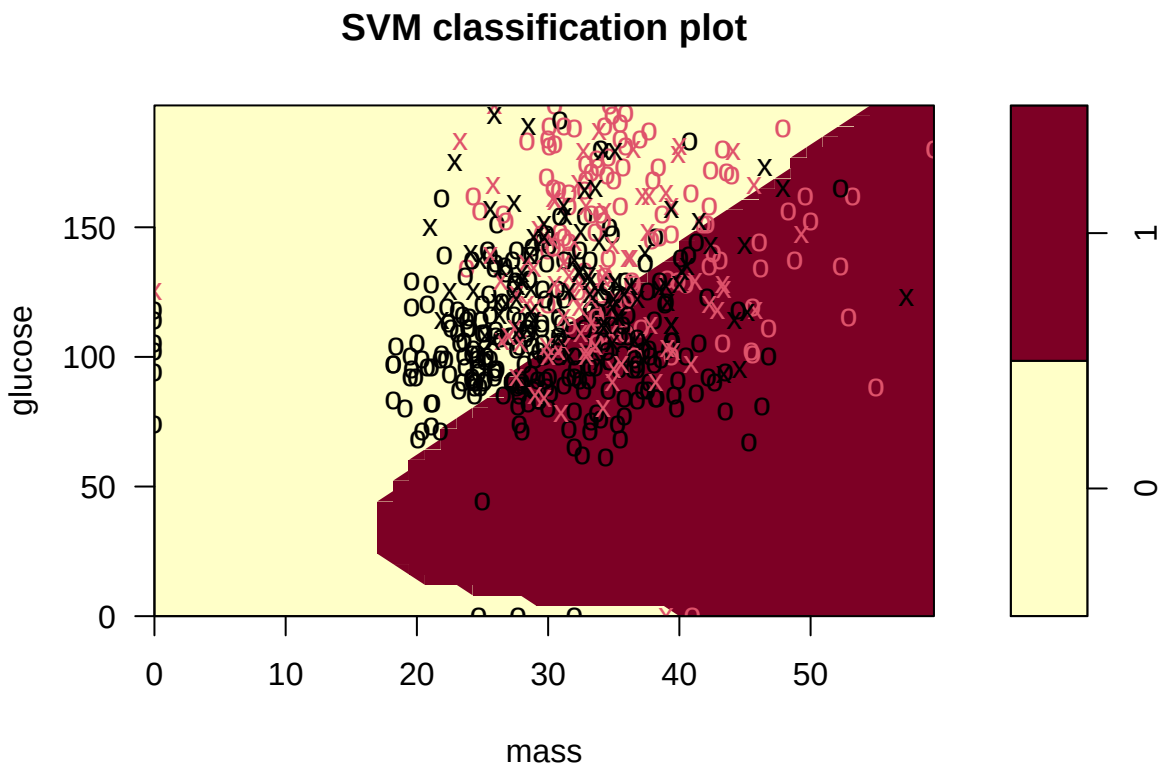
```r
# assess the accuracy of your prediction on the testing set

confmat = table(predict(svmPoly), train$diabetes, dnn=c("Prediction", "Actual"))
```

```r
# fit using "svm", with "polynomial" kernel, degree = 2, cost = 5, type = "C", and scale = F

svmPoly2 = svm(diabetes~glucose+mass,data = train, kernel='polynomial',scale=F,type='C-classification',c

# visualize the decision boundary of your SVM model
plot(svmPoly2, train)
```




```r
# assess the accuracy of your prediction on the testing set
confmat2 = table(predict(svmPoly2), train$diabetes, dnn=c("Prediction", "Actual"))
```

```r
# fit using "svm", with "polynomial" kernel, degree = 3, cost = 5, type = "C", and scale = F

svmPoly3 = svm(diabetes~glucose+mass,data = train, kernel='polynomial',scale=F,type='C-classification',c

# visualize the decision boundary of your SVM model

plot(svmPoly3, train)
```



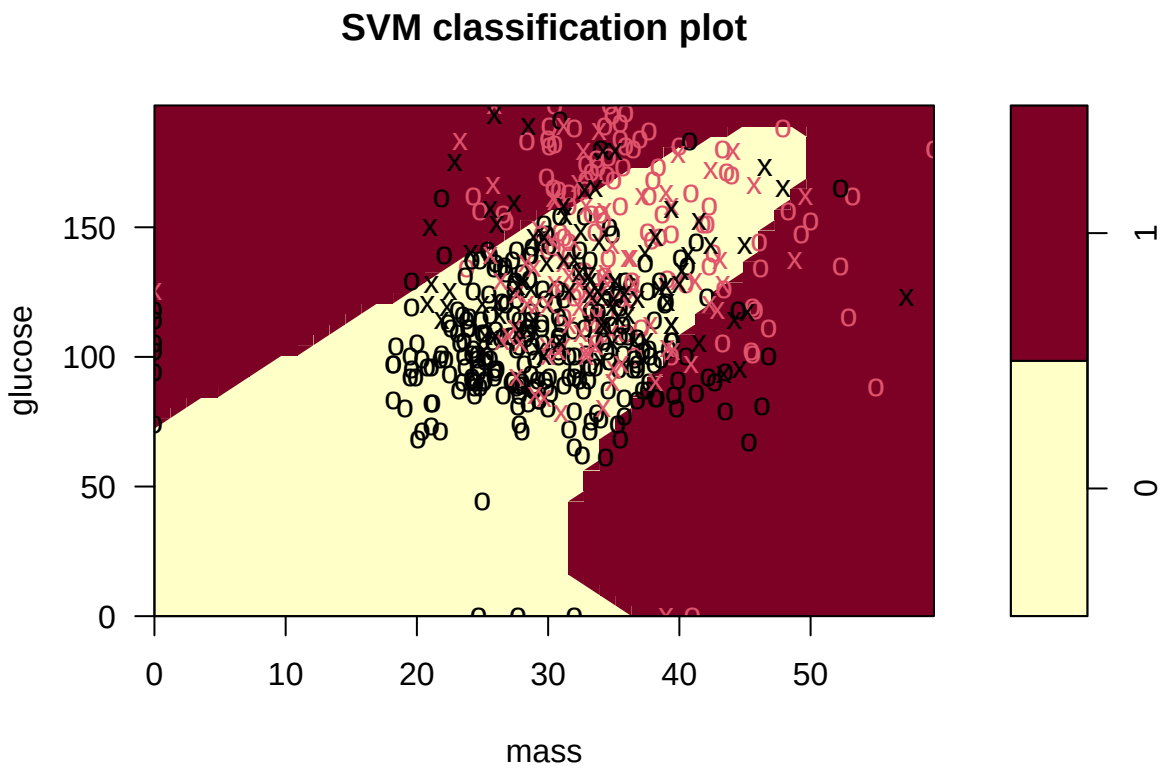

```
# assess the accuracy of your prediction on the testing set

confmat3 = table(predict(svmPoly3), train$diabetes, dnn=c("Prediction", "Actual"))
```

1.3 What happens to our decision boundary as the degree of the polynomial kernel increases? (2 pts)

The degree is a tuning of hyperplane. It takes on a radial shape, such that boundaries are more accurate. The number of iterations increases with each degree. The vectors are generally optimal at a degree of 3, where the clusters seemingly contain more correctly placed points.

1.4 SVM & Varying Cost. Let's vary our "cost" parameter and observe our decision boundaries.(2 pts)

```
# fit using "svm", with "polynomial" kernel, degree = 3, cost = 50, type = "C", and scale = F

svmPoly4 = svm(diabetes~glucose+mass,data = train, kernel='polynomial',scale=F,type='C-classification',c

# visualize the decision boundary of your SVM model

plot(svmPoly4, train)
```



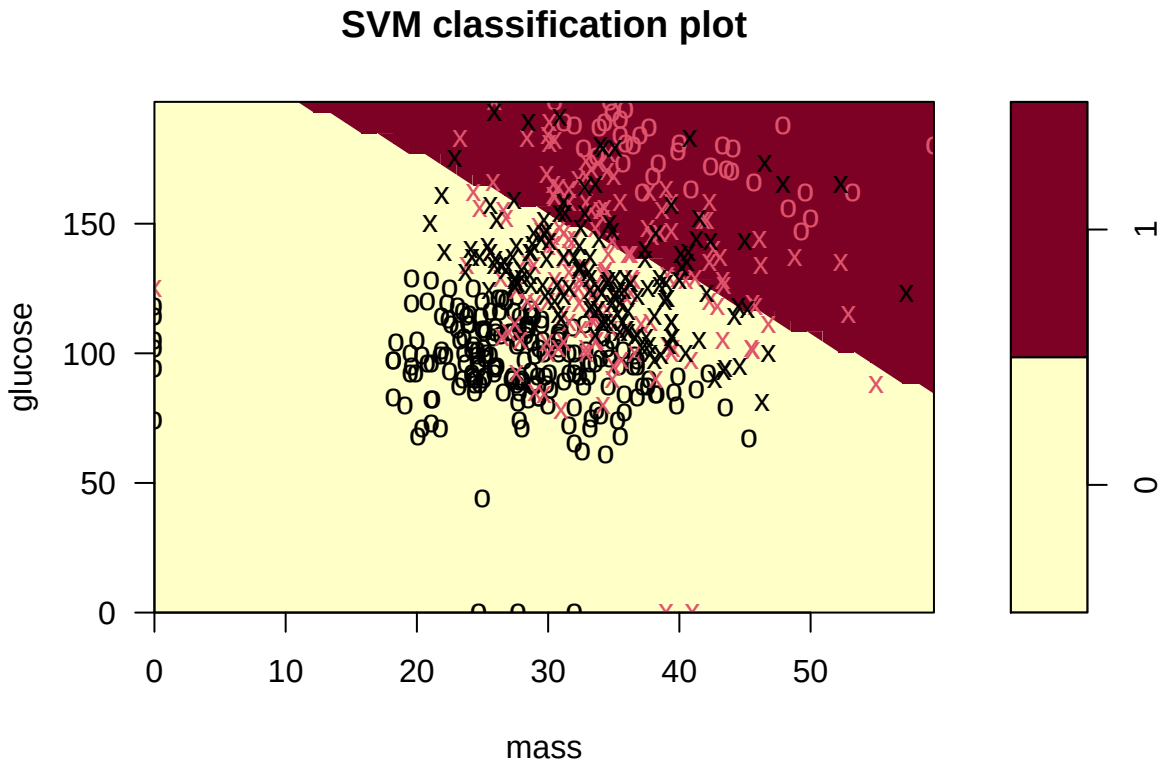

```
# assess the accuracy of your prediction on the testing set

confmat4 = table(predict(svmPoly4), train$diabetes, dnn=c("Prediction", "Actual"))
```

1.5 What happens to our decision boundary as we increase "cost"? (3 pts)

Cost = cost of constraints violation, cost of misclassification. The hyperplane of the model takes on a different shape that classifies more points correctly. However, predictions are less accurate.

1.6 SVM Parameter Tuning. Let's fit all of the data using SVM and search for the optimal parameter using 'tune.svm'. Vary your cost parameter as c(4, 8, 16, 32), set your kernel to "linear", type = 'C', and scale = F. Using formula: diabetes ~ glucose + mass.(5 pts)

```
# tune your SVM model

svmLinear = tune.svm(diabetes~glucose+mass,data = train, kernel='linear',scale=F,type='C-classification

# report the accuracy of your best performing model

confmat5 = table(predict(svmLinear$best.model), train$diabetes, dnn=c("Prediction", "Actual"))

# visualize the decision boundary of your SVM model

plot(svmLinear$best.model, train)
```



Question 2.Logistic Regression. In this exercise, you will work with logistic regression and we are using default data provided by ISLR package. This simulated dataset contains information on ten thousand customer such as whether the customer defaulted, is a student, the average balance carried by the customer and the income of the customer.(20 pts)

2.1 Load the dataset and preview it (2 pts)

```
# load dataset
data(Default)

Default$default = as.character(Default$default)
Default$default[Default$default== "Yes"] = 1
Default$default[Default$default== "No"] = 0
Default$default = as.numeric(Default$default)
```

2.2 split your data into traing and testing datasets (70% and 30%). Use seed(123) to be consistent (2 pts)

```r
# split data into training & testing (70/30 split)
set.seed(123)
split = createDataPartition(y=Default$default,p = 0.7,list = F,groups = 100)
train2 = Default[split,]
test2 = Default[-split,]
```

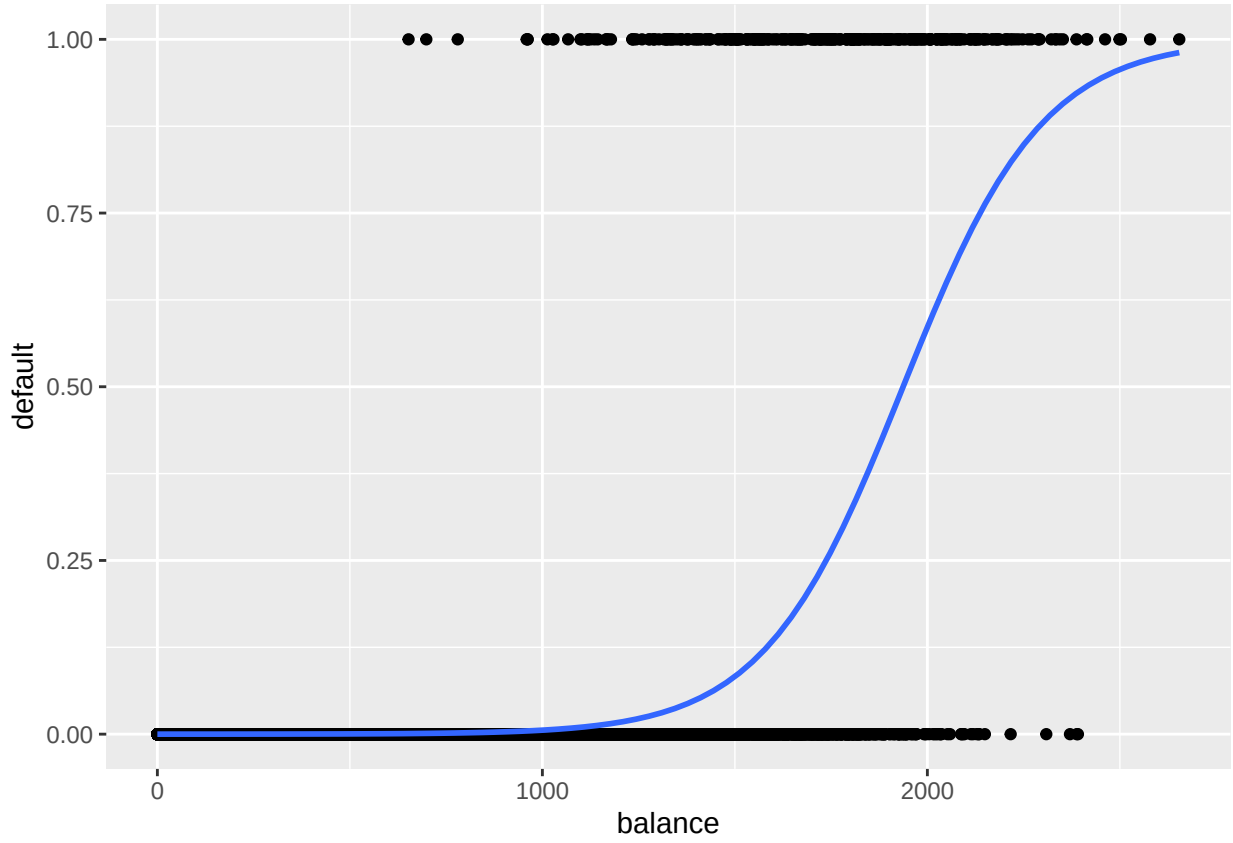
2.3 Fit a logisitic regression model to predict the probability of a customer defaultig based on the average balance carried by the customer (4 pts)

```r
#use glm function

fit = glm(default~ balance, train2, family = "binomial")
```

2.4 Plot the logistic function (4 pts)

```r
#plot logistic function

ggplot(Default, aes(balance, default))+
  geom_point()+
  stat_smooth(method = 'glm', method.args=list(family="binomial"), se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



2.5 predict the probability of defualting based on balance of $1000 and $2000 (2 pts)

```
#predict the probability of defualting

pred = predict(fit, newdata=data.frame(balance = c(1000,2000)), type="response")

pred
```

```
##           1           2
## 0.005648303 0.593966756
```

2.6 Test the predicted target varaiable vs. the observed values of the model(2 pts)

```
#Test the predicted target varaiable vs. the observed values

pred2 = predict(fit, newdata=test2, type="response")
predictions=ifelse(pred2>0.5,"Yes","No")
test2$default = ifelse(test2$default>=0.5,"Yes","No")
```

2.7 Form the confusion matrix to see the classification performance (2 pts)

```
#Form the confusion matrix

table(test2$default,predictions,dnn=c("Actual","Predicted"))
```

```
##       Predicted
## Actual   No  Yes
##    No  2884   16
##    Yes   71   29
```

2.8 Determine the misclassification rate(2 pts)

```
#misclassification rate

round(mean(predictions!=test2$default),4)
```

```
## [1] 0.029
```