Lab09 - Classification (Logistic Regression)

0) Why not linear regression?
In general, there is **no natural way** to convert a ***qualitative*** response with more than two levels into a ***quantitative*** response ready for linear regression.

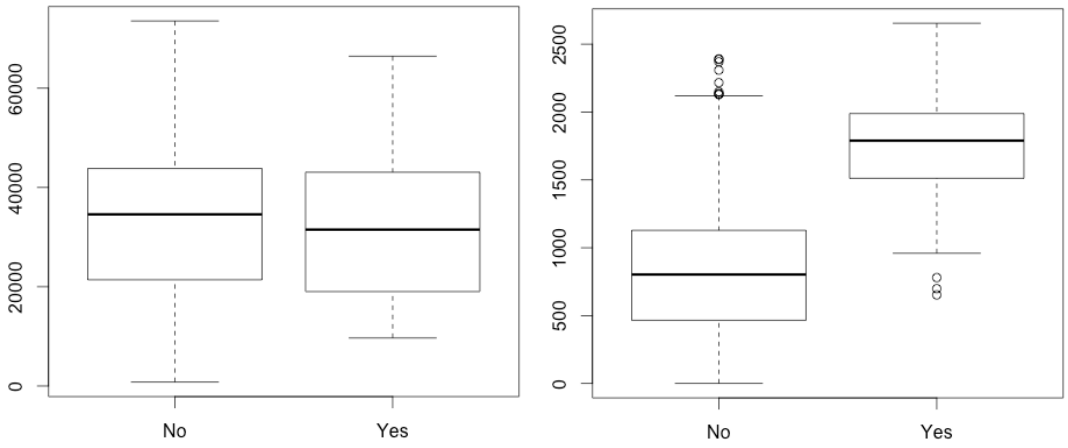
Remember the hospital example?

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases} \qquad Y = \begin{cases} 1 & \text{if epileptic seizure;} \\ 2 & \text{if stroke;} \\ 3 & \text{if drug overdose.} \end{cases}$$

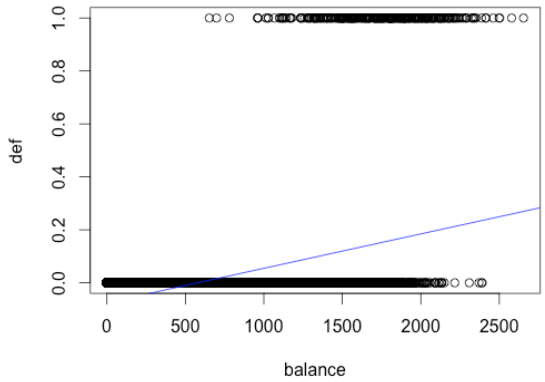
Let us try to do a linear regression on the "Default" dataset (see ISLR, page 129):

```
library(ISLR)
attach(Default)
str(Default)
summary(Default)

plot(balance[default=='No'],income[default=='No'],col='blue')
points(balance[default=='Yes'],income[default=='Yes'],col='red',pch
=4)

plot(default,income)
plot(default,balance)
```

(see income on the left, balance on the right, and note how people who default tend to have, on average, roughly the same income, but different credit card balances)




```
def = rep(1,10000)
def[default=='No'] = 0
plot(balance,def)
fit = lm(def~balance)
abline(fit)
```



look at how the line fits the data: what if balance < 500?

1) Logistic function

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

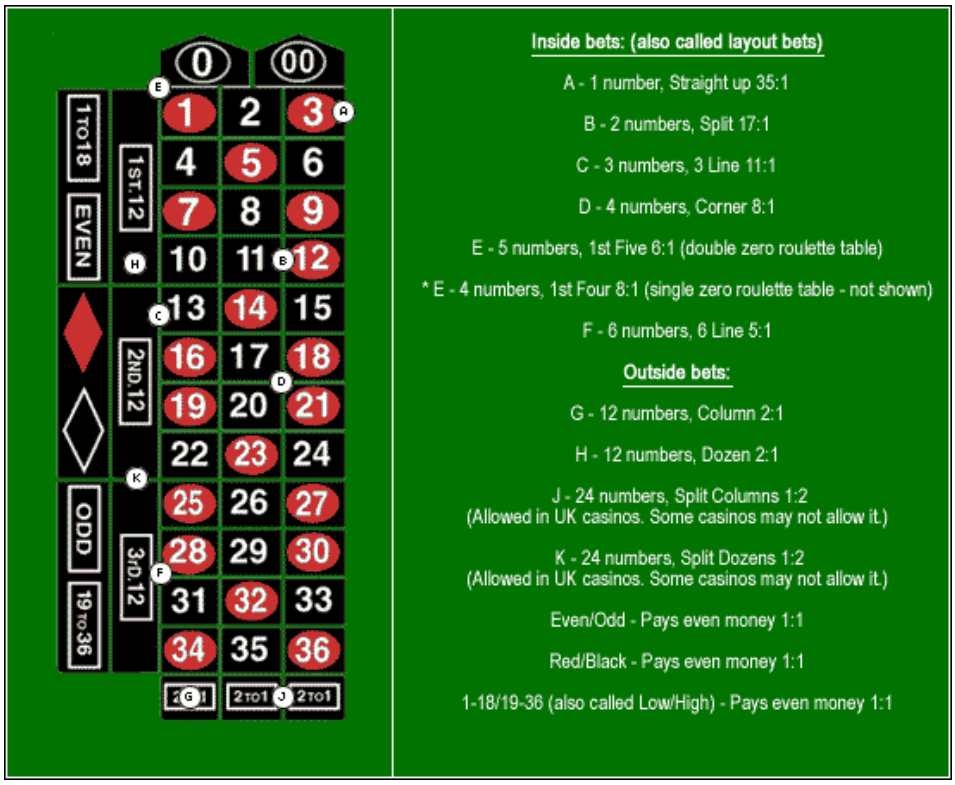
Check out what it looks like in R:

```
x = rnorm(100)
b0 = 0
b1 = 1
p = exp(b0+b1*x)/(1+exp(b0+b1*x))
plot(x,p)
```

The logistic function is equivalent to the definition of ***odds***:

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$$

... let us do some gambling now! Calculate the probabilities of the different bets in the game of roulette and verify whether the odds are fair (hint: no. Can you tell why?)



With log-odds or ***logit*** we get rid of the exp (note that the logit is linear in X):

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

We cannot easily derive a relationship between a change in X and a change in P(X) (as it depend both on beta1, on how much X changes, and on the value of X itself). However, we can at least say that if beta1 is positive, then increasing X will increase P(X), and if beta1 is negative, then an increase in X will decrease P(X).

2) Exercise on logits (from a previous exam)

(b) A hospital collected data for a group of patients to study the relationship between Heart-attack Risk Index (HRI=X_1), weekly hours of physical activity (PHY=X_2), and the probability Y of having a heart attack. Roughly, for a heart attack probability to be low the HRI should be below 5, and the more hours one spends exercising the better it is. After fitting a logistic regression, the following coefficients were estimated: $\hat{\beta}_0 = -9.7$, $\hat{\beta}_1 = 1.05$, and $\hat{\beta}_2 = -0.29$.

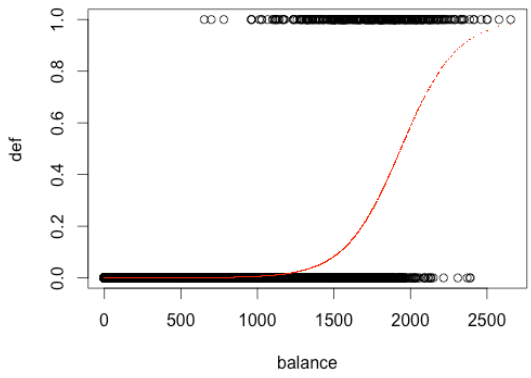
- estimate the probability for a patient with HRI=5 and PHY=2 to have a heart attack;
- estimate how many hours of PHY a patient with HRI=7.5 should do to have that same probability.

We know that P(X)/(1-P(x)) = exp(b0 + b1 * HRI + b2 * PHY). Thus:

```
# first part: calculate heart attack probability for a patient with
HRI=5 and PHY=2
b0 = -9.7
b1 = 1.05
b2 = -0.29
X1 = 5 # HRI
X2 = 2 # PHY
e = exp(b0 + b1 * X1 + b2 * X2)
p = e/(1+e)
# p is ~0.0065

# second part: calculate how many hours of PHY a patient with
HRI=7.5 should have
# to have a heart-attack p as the one we just calculated
e = p/(1-p) # this is the same e as before!
phy = ( log(e) - b0 - b1 * 7.5 ) / b2
# phy is ~11.05
```

3) Logistic regression on the Default dataset

```
logit = glm(default ~ balance, family='binomial')
summary(logit)

b0 = logit$coefficients[1]
b1 = logit$coefficients[2]
probs = exp(b0+b1*balance)/(1+exp(b0+b1*balance))
# equivalent to:
# probs = predict(logit,type="response")
plot(balance,def)
points(balance,probs,pch=46,col='red')
```




```
# look at the predictions
pred = rep(0,10000)
pred[probs>.5]=1
table(def, pred)
mean(pred==def)
```

```
# comment the output of the "table" command comparing with the
picture below:
```

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

TABLE 4.6. *Possible results when applying a classifier or diagnostic test to a population.*

```
# also test this on income and student
logit = glm(default ~ income, family='binomial')
summary(logit)
probs = predict(logit,type="response")
# equivalent to:
# b0 = logit$coefficients[1]
# b1 = logit$coefficients[2]
# probs = exp(b0+b1*income)/(1+exp(b0+b1*income))
plot(income,def)
points(income,probs,pch=46,col='red')

logit = glm(default ~ student, family='binomial')
summary(logit)
probs = predict(logit,type="response")
# note that the values in probs are repeated (as they only depend
on the
# two student values Yes/No) and compare with the following (where
1 and
# 0 mean "student=Yes" and "student=No"
b0 = logit$coefficients[1]
b1 = logit$coefficients[2]
exp(b0+b1*1)/(1+exp(b0+b1*1))
exp(b0+b1*0)/(1+exp(b0+b1*0))


# MULTIPLE LOGISTIC REGRESSION
logit = glm(default ~ balance + income + student,
family='binomial')
summary(logit)
probs = predict(logit,type="response")
plot(balance,def)
points(balance[student=='No'],probs[student=='No'],pch=46,col='blue
')
points(balance[student=='Yes'],probs[student=='Yes'],pch=46,col='re
d')
# look at the predictions
```

```
pred = rep(0,10000)
pred[probs>.5]=1
table(def, pred)
mean(pred==def)
```

4) South African Heart Disease example

Go to http://statweb.stanford.edu/~tibs/ElemStatLearn/ for the dataset

```
#heart =
read.table("http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data", sep=",",head=T,row.names=1)
heart =
read.table("~/Downloads/SAheart.data", sep=",",head=T,row.names=1)

names(heart)
cor(heart[,c(1:4,6:10)])

attach(heart)
glm.fit = glm(chd ~ tobacco + age + adiposity + alcohol, family =
binomial)
summary(glm.fit)

glm.fit = glm(chd ~ ., data=heart, family = binomial)
summary(glm.fit)

glm.probs = predict(glm.fit,type="response")
max(glm.probs)
which.max(glm.probs)
glm.probs[407]
heart[407,]

# try to evaluate the accuracy of our model
glm.pred = rep(0,462)
glm.pred[glm.probs>.5]=1
table(chd, glm.pred)
mean(glm.pred==chd)
```

5) Now do a more realistic test, dividing the dataset into training and test

```
train = rep (FALSE,dim(heart)[1])
train[1:360]=TRUE
heart.test = heart[!train,]

glm.fit = glm(chd ~ ., data=heart, family = binomial, subset =
train)
glm.probs = predict(glm.fit, heart.test, type="response")

glm.pred = rep(0,dim(heart.test)[1])
glm.pred[glm.probs>.5]=1
```

```
chd.test = chd[!train]
table(chd.test, glm.pred)
mean(glm.pred==chd.test)
```

Example with spam

```
spam = read.table("~/Downloads/spam.data")
test = read.table("~/Downloads/spam.traintest")
train = (test==0)
spam.train = spam[train,]
spam.test = spam[!train,]

glm.fit = glm(V58 ~ ., data=spam.train, family=binomial)
glm.probs = predict(glm.fit, spam.test, type="response")

# we run the split by doing the default probs
glm.pred = rep(0,dim(spam.test)[1])
glm.pred[glm.probs>.5]=1
V58.test = spam$V58[!train]
V58.train = spam$V58[train]

table(glm.pred,V58.test, dnn=c("Spam (predicted)","Spam (ground
truth)"))
mean(glm.pred!=V58.test)
```