Lab 8: Logistic Regression

Stat 154, Spring 2018

The `Default` Data Set

Consider the `Default` data set that comes in the R package `"ISLR"`. We are interested in predicting whether an individual will default on his or her credit card payment, on the basis of annual income and monthly credit card balance.

```r
# remember to load package ISLR!
names(Default)
```

```
## [1] "default" "student" "balance" "income"
```

```r
dim(Default)
```

```
## [1] 10000     4
```

```r
summary(Default)
```

```
##  default    student       balance           income     
##  No :9667   No :7056   Min.   :   0.0   Min.   :  772  
##  Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340  
##                        Median : 823.6   Median :34553  
##                        Mean   : 835.4   Mean   :33517  
##                        3rd Qu.:1166.3   3rd Qu.:43808  
##                        Max.   :2654.3   Max.   :73554  
```

```r
summary(subset(Default, default == 'Yes'))
```

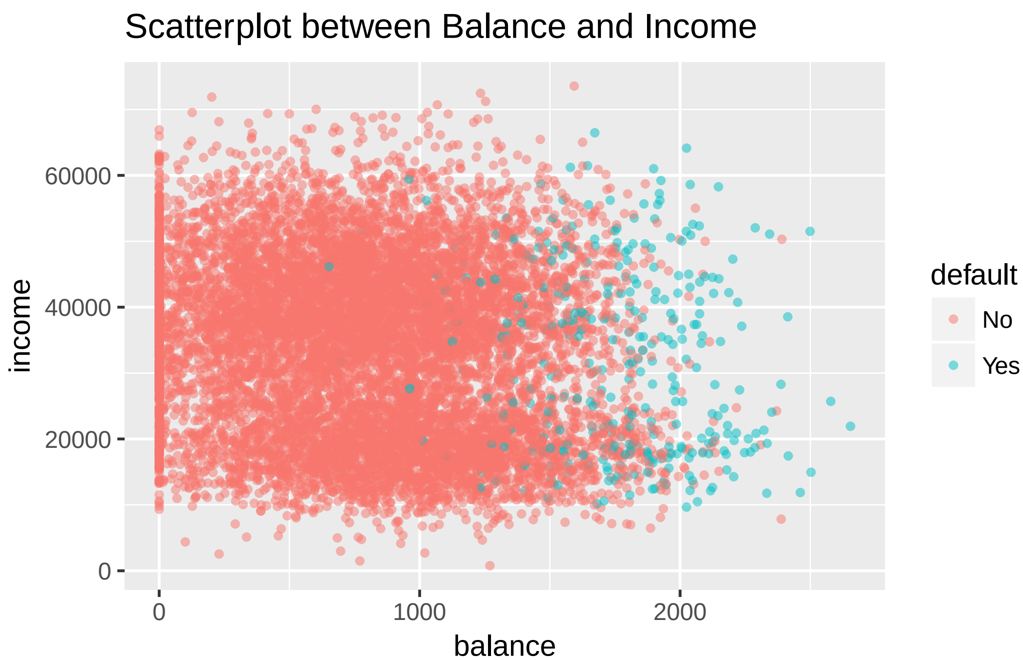
```
##  default   student      balance           income     
##  No :  0   No :206   Min.   : 652.4   Min.   : 9664  
##  Yes:333   Yes:127   1st Qu.:1511.6   1st Qu.:19028  
##                      Median :1789.1   Median :31515  
##                      Mean   :1747.8   Mean   :32089  
##                      3rd Qu.:1988.9   3rd Qu.:43067  
##                      Max.   :2654.3   Max.   :66466  
```

```r
summary(subset(Default, default == 'No'))
```

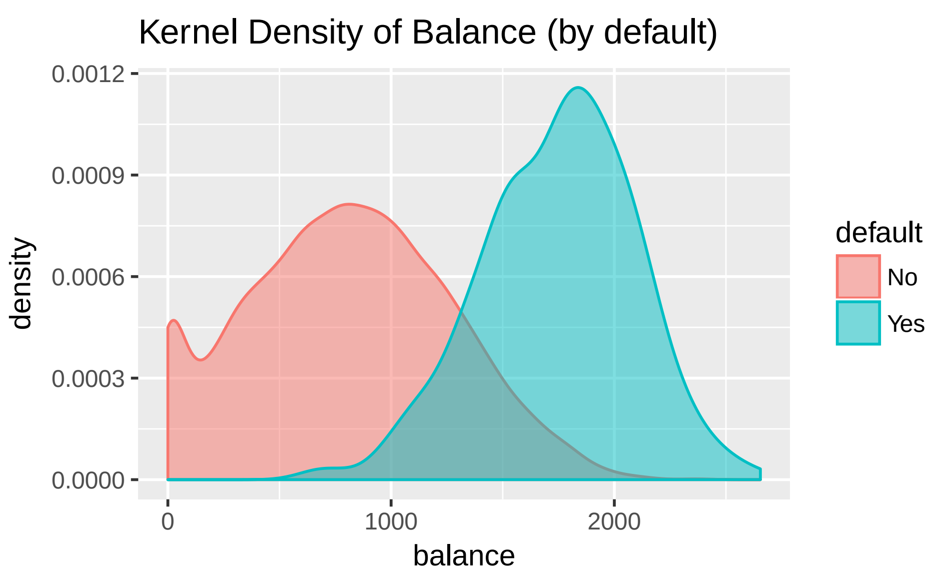
```
##  default    student       balance           income     
##  No :9667   No :6850   Min.   :   0.0   Min.   :  772  
##  Yes:   0   Yes:2817   1st Qu.: 465.7   1st Qu.:21405  
##                        Median : 802.9   Median :34589  
##                        Mean   : 803.9   Mean   :33566  
```

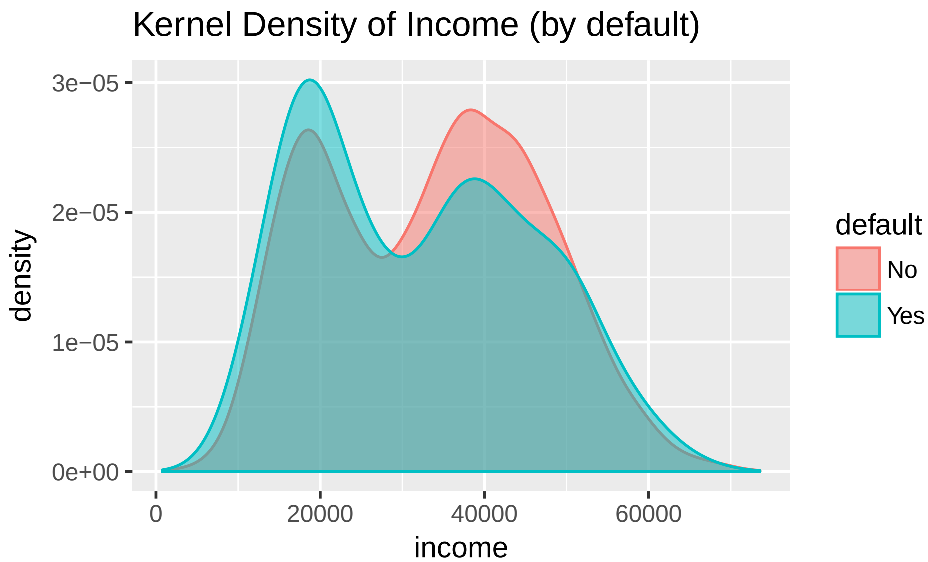
```
##                          3rd Qu.:1128.2   3rd Qu.:43824
##                          Max.   :2391.0   Max.   :73554
```

Begin with some exploratory displays of the data. You can start with a scatterplot of `balance` and `income`, distinguishing observations based on `default`, like in the image below



Make density plots of `balance` and `income`, for instance:





OLS Regression

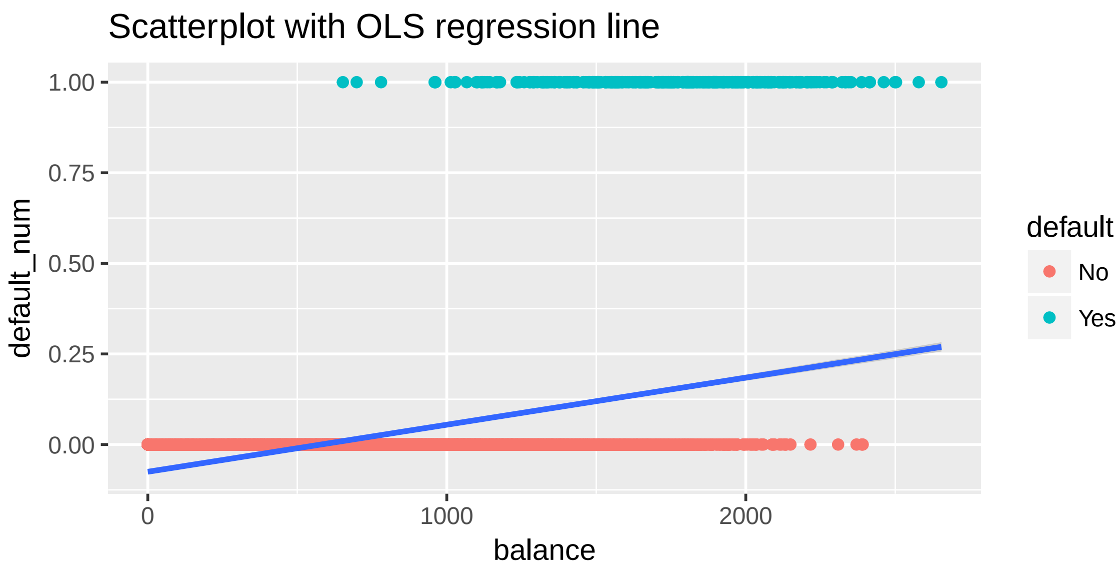
Out of curiosity, let's fit an OLS model by regressing `default` on `balance`. Because `default` is a factor, you should create a numeric *default* vector:

```
# code default as numeric
default_numeric <- rep(0, nrow(Default))
default_numeric[Default$default == 'Yes'] <- 1
Default$default_num <- default_numeric

ols_reg <- lm(default_num ~ balance, data = Default)
summary(ols_reg)
```

```
##
## Call:
## lm(formula = default_num ~ balance, data = Default)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23533 -0.06939 -0.02628  0.02004  0.99046
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.519e-02  3.354e-03  -22.42   <2e-16 ***
## balance      1.299e-04  3.475e-06   37.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1681 on 9998 degrees of freedom
```

```
## Multiple R-squared:  0.1226, Adjusted R-squared:  0.1225
## F-statistic:  1397 on 1 and 9998 DF,  p-value: < 2.2e-16
```



Logistic Regression

The response `default` falls into one of two categories: `"Yes"` or `"No"`. Rather than modeling `default` directly, logistic regression models the probability that the response Y belongs to a particular category.

The probability of default given `balance` can be written as:

$$Pr(\texttt{default} = \texttt{Yes|balance})$$

To fit a logistic regression model you use the function `glm()`. The syntax of `glm()` is similar to that of `lm()`, except you must specify the argument `family = binomial` in order to tell R to run a logistic regression rather than some other type of generalized linear model.

Notice that `glm()` knows how to handle the response `default` which is a factor:

```
logreg_default <- glm(default ~ balance, family = binomial, data = Default)

summary(logreg_default)$coefficients
```

```
##                  Estimate   Std. Error   z value      Pr(>|z|)
## (Intercept) -10.651330614 0.3611573721 -29.49221 3.623124e-191
## balance       0.005498917 0.0002203702  24.95309 1.976602e-137
```

How do we interpret the coefficients? A one-unit increase in `balance` is associated with an increase in the log odds of `default` by 0.005 units.

Many aspects of the logistic regression output shown in the `summary()` are similar to the `lm()` output. For example, you can measure the accuracy of the coefficient estimates by computing their standard errors.

To make predictions we can use the coefficient estimates. For instance, the predicted default probability for an individual with a `balance` of $1,000 is:

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 1000}}{1 + e^{-10.6513 + 0.0055 \times 1000}} = 0.00576$$

which is below 1%. In contrast, the predicted probability of default for an individual with a balance of $2,000 is 0.586 or 58.6%.

Your turn

- Find out how to use `predict()` to obtain the probability of default for individuals with `balance` values of $100, $200, $300, …, $2,000
- Fit another logistic regerssion model by regressing `default` on `student`. How would you interpret the coefficient estimate?
- Fit a third logistic regression by regressing `default` on `balance`, `student`, and `income`.
- Are all coefficient estimates significant?
- How would you explain the apparent contradiction between the opposite signs of the `student` coefficients (this regression versus the previous one)?
- Answer to these questions are in pages 134-137 of ISL.

The Stock Market `Smarket` Data

You will be working with the `Smarket` data, which is part of the `"ISLR"` package. This data consists of percentage returns fro the S&P 500 stock index over 1,250 days, from the beginning of 2001 until the end of 2005. For each date, the percentage returns for each of the five previous tradings has been records, `Lag1` through `Lag5`. Other variables are:

- `Volume` = the number of shares traded on the prevous day, in billions
- `Today` = the percentage return on the data in question
- `Direction` = whether the market was `Up` or `Down` on this date

```
# remember to load package ISLR
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

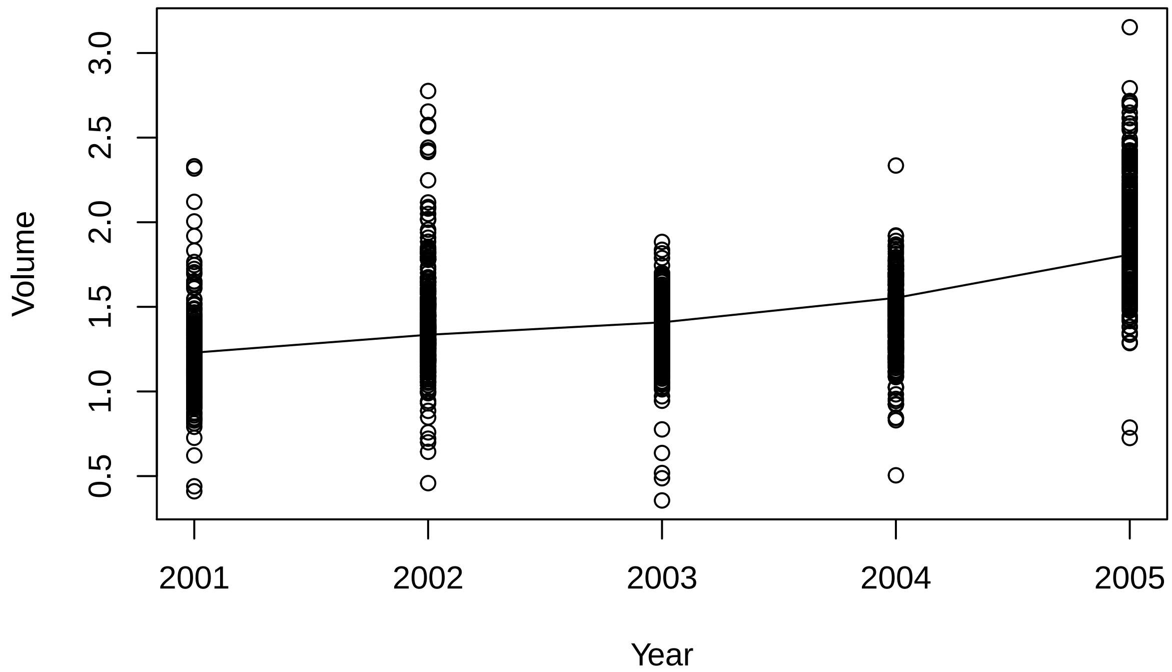
```r
dim(Smarket)
```

```
## [1] 1250    9
```

```r
summary(Smarket)
```

```
##       Year           Lag1                Lag2
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500
##  Median :2003   Median : 0.039000   Median : 0.039000
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000
##       Lag3                Lag4                Lag5
##  Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200
##  1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000
##  Median : 0.038500   Median : 0.038500   Median : 0.03850
##  Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561
##  3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700
##  Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300
##      Volume           Today           Direction
##  Min.   :0.3561   Min.   :-4.922000   Down:602
##  1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648
##  Median :1.4229   Median : 0.038500
##  Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   :3.1525   Max.   : 5.733000
```

- Compute the matrix of correlations of the variables in `Smarket`, excluding the variable `Direction`
- Perform a PCA on `Smarket[ ,-9]` to get a visual display of the variables. You can accomplish this with the function `PCA()` from the `"FactoMineR"` package. By default, it plots a *circle of correlations*
- How correlated are the lag variables with today's returns? Are previous day's returns highly correlated with today's returns?
- Make a scatterplot of `Year` and `Volume`



Logistic Regression

Use the `glm()` function to fit a logistic regression model in order to predict `Direction` using `Lag1` through `Lag5` and `Volume`.

- Inspect the `summary()` of the `"glm"` object containing the output of the logistic regression.
- Looking at the p-values of the regression coefficients, which coefficient seems to be significant?
- What is the coefficient value of `Lag1`? How would you interpret the sign of this coefficient?
- Use the `predict()` function to predict the probability that the market will go up, given values of the predictors. Use the argument `type = "response"` which tells R to output probabilities of the form $P(Y = 1|X)$, as oppose to other information such as the logit. The first 10 probabilities look like this:

```
##         1         2         3         4         5         6         7
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509
##         8         9        10
## 0.5092292 0.5176135 0.4888378
```

Estimation of Parameters

In this part of the lab, your mission is to implement code in R that allows you to estimate the logistic regression coefficients.

The estimation of the coefficients is carried out by Maximum Likelihood. Consider, for instance, a logistic model with one predictor and one response. We look for $\hat{\beta}_0$ and $\hat{\beta}_1$ that maximize the log-likelihood $l(\hat{\beta}_0, \hat{\beta}_1)$. To do so, we set the first order partial derivatives of $l(\boldsymbol{\beta})$ to zero.

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \beta_0} = \sum_{i=1}^{n} (y_i - p(x_i)) = 0$$
$$\frac{\partial l(\boldsymbol{\beta})}{\partial \beta_1} = \sum_{i=1}^{n} x_i (y_i - p(x_i)) = 0$$

Unfortunately, there is no analytical solution to this problem. So how do you actually compute the estimates? Using the Newton-Raphson algorithm.

- Let $\mathbf{y}$ be the column vector of response Y
- Let $\mathbf{X}$ be the $n \times (p+1)$ input (design) matrix
- Let $\mathbf{p}$ be the n-vector of fitted probabilities with the i-th element $p(x_i; \beta^{old})$
- Let $\mathbf{W}$ be an $n \times n$ diagonal matrix of weights with i-th element $p(x_i; \beta^{old})(1 - p(x_i; \beta^{old}))$

Then:

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{X}^{\mathsf{T}}(\mathbf{y} - \mathbf{p})$$
$$\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^{\mathsf{T}}} = -\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X}$$

Newton-Raphson algorithm

Here's the pseudo-code to obtain the estimate coefficients.

1. $\mathbf{b^{old}} \longleftarrow \mathbf{0}$
2. Compute $\mathbf{p}$ by setting its elements to:

$$p(x_i) = \frac{e^{\mathbf{x_i}^{\mathsf{T}} \mathbf{b^{old}}}}{1 + e^{\mathbf{x_i}^{\mathsf{T}} \mathbf{b^{old}}}}$$

3. Compute the diagonal matrix $\mathbf{W}$ with the i-th diagonal element: $p(x_i)(1 - p(x_i)), \quad i = 1, \dots, n$

4. $\mathbf{z} \longleftarrow \mathbf{X}\mathbf{b}^{\mathbf{old}} + \mathbf{W}^{-\mathbf{1}}(\mathbf{y} - \mathbf{p})$
5. $\mathbf{b}^{\mathbf{new}} \longleftarrow (\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X})^{-\mathbf{1}}\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{z}$
6. Check whether $\mathbf{b}^{\mathbf{old}}$ and $\mathbf{b}^{\mathbf{new}}$ are close "enough", otherwise update $\mathbf{b}^{\mathbf{old}} \leftarrow \mathbf{b}^{\mathbf{new}}$, and go back to step 2.

Your turn: Write code in R for the algorithm described above. Use the `Lag` variables and `Volume` as predictors, and `Direction` as response. You will have to convert `Direction` into a numeric vector by coverting `'Up'` to `1` and `'Down'` to `0`.

Simplified Algorithm

Since $\mathbf{W}$ is an $n \times n$ diagonal matrix, direct matrix operations with it may be very inefficient. A modified pseudo code is provided next.

1. $\mathbf{b}^{\mathbf{old}} \longleftarrow \mathbf{0}$
2. Compute $\mathbf{p}$ by setting its elements to:

$$p(x_i) = \frac{e^{\mathbf{x_i}^\mathsf{T}\mathbf{b}^{\mathbf{old}}}}{1 + e^{\mathbf{x_i}^\mathsf{T}\mathbf{b}^{\mathbf{old}}}}$$

3. Compute the $n \times (p+1)$ matrix $\tilde{\mathbf{X}}$ by multiplying the i-th row of matrix $\mathbf{X}$ by $p(x_i)(1 - p(x_i)), \quad i = 1, \ldots, n$
4. $\mathbf{b}^{\mathbf{new}} \longleftarrow \mathbf{b}^{\mathbf{old}} + (\mathbf{X}^\mathsf{T}\tilde{\mathbf{X}})^{-\mathbf{1}}\mathbf{X}^\mathsf{T}(\mathbf{y} - \mathbf{p})$
5. Check whether $\mathbf{b}^{\mathbf{old}}$ and $\mathbf{b}^{\mathbf{new}}$ are close "enough", otherwise update $\mathbf{b}^{\mathbf{old}} \leftarrow \mathbf{b}^{\mathbf{new}}$, and go back to step 2.

Your turn: Write code in R for the simplified algorithm described above. Use the `Lag` variables and `Volume` as predictors, and `Direction` as response. You will have to convert `Direction` into a numeric vector by coverting `'Up'` to `1` and `'Down'` to `0`.