Computer lab 6 in MMS075, Feb 27, 2020

1. Experiment with the usage of scientific and regular (fixed) number notation in R. The usage is regulated by the command **options(scipen=…)**, where **…** is replaced by a number. If this number is positive, that will display numbers more in the regular format, but if a negative number is given, then the scientific notation will be preferred. We now display a couple of numbers to experiment with this, changing the **scipen** argument to 999, then -999, then 0 and displaying the same numbers again to see the difference:

   options(scipen=999)
   1000
   1000^2
   1/1000
   1/1000^2
   options(scipen=-999)
   1000
   1000^2
   1/1000
   1/1000^2
   options(scipen= 0)
   1000
   1000^2
   1/1000
   1/1000^2

   You are encouraged to experiment with other values of **options(scipen=…))** and also with displaying other numbers!

2. This exercise explores some of the plotting possibilities and how such plots can aid variable selection in logistic regression. We will re-create the plots in exercise 1 in exercise class 6a and check whether we were right about X2 being a better predictor of Y. In the first step, we reproduce the data and put the variables into a data frame:

   x1=c(10,20,30,40,50,60,70,80,90,100,16,21,34,38,49,62,78,80,100,103)
   x2=c(2,1,3,4,4,7,6,10,7,9,-6,-9,-3,4,-1,5,-3,11,6,6)
   y=c(rep("Yes",10),rep("No",10))
   Dat=data.frame(x1,x2,y)

   Note: the **rep** function used in the definition of y stands for repeat (or replicate); if you want to learn more about how it is used, type **?rep** to get a description. It is a good idea to check the data frame that was created:

   View(Dat)

   If it looks fine, we proceed to creating the plots. In the scatter plot, cases (which we take as **"Yes"** values of the response **y**) are plotted separately from non-cases (i.e. **"No"** values of **y**), so we create separate variables for predictor values corresponding to cases and non-cases:

   x1Case=x1[y=="Yes"]
   x2Case=x2[y=="Yes"]
   x1NotCase=x1[y=="No"]

x2NotCase=x2[y=="No"]

What happened in these command lines? For example, in the first line, we defined a variable called **x1Case** as those elements of **x1** that correspond to **"Yes"** values of the variable **y** at the same location. It is important to use two equal signs in the formula; if you use only one equal sign, R will believe that you want to define the value of **y** instead of checking whether its value agrees with **"Yes"**.

We can now create the plot in two steps, starting, for example, with plotting the non-cases and adding the cases as extra points:

```
plot(x1NotCase,x2NotCase,col="grey50",pch=20,cex=1.6,cex.lab=1.8,xlab="X1",ylab="X2")
points(x1Case,x2Case, col="red",pch=3,cex=1.25)
```

In the first line, **col** is the color of points, **pch** is the type of points, **cex** corresponds to the size of points, **cex.lab** to the size of the axis labels, while **xlab** and **ylab** specify the names of the axis labels.

We have created the scatter plot, but we can work a bit more to make it look better. In this case, it may be relevant to change the margins around the figure to ensure that all labels are properly visible. We can do this with the **par(mar=c(…,…,…,…))** command, where the **…** values are replaced by the margins values we want, in the following order: bottom margin, left margin, top margin, right margin. After setting some new margins (see an example in the first line below), we can create the plot again to see the difference:

```
par(mar=c(5,5,2,1))
plot(x1NotCase,x2NotCase,col="grey50",pch=20,cex=1.6,cex.lab=1.8,xlab="X1",ylab="X2")
points(x1Case,x2Case, col="red",pch=3,cex=1.25)
```

A further potential improvement is to find better colors than the ones specified above; you can get an overview of different possibilities e.g. at this link. Feel free to try different color options as well as different point types and sizes by changing values of the **pch** and **cex** arguments until the figure looks nice enough.

Additional to the scatter plot, the box plots corresponding to the different values of the response variable can be informative for identifying potentially relevant predictors. In this case, creating these box plots can be done as follows:

```
plot(Dat$x1~Dat$y,col=c("grey50","red"),cex=1.2,cex.lab=1.75,xlab="Y",ylab="X1",cex.axis=1.2)
plot(Dat$x2~Dat$y,col=c("grey50","red"),cex=1.2,cex.lab=1.75,xlab="Y",ylab="X2",cex.axis=1.2)
```

Both the scatter plot and the box plot suggest that X2 may be a better predictor, because there seems to be a clearer separation of y values by X2 than by X1. Create three different logistic regression models to see if this is the case: one with only X1, another with only X2, and a model with both X1 and X2! What can you learn from the summaries of these models?

3. We further investigate relevant plots for logistic regression, in particular how results for a categorical variable can be plotted and how this can aid the identification of confounding.

We work with the **Default** dataset in the **ISLR** library and our aim is to replicate the figure on slide 35 of Lecture 6a. The first step is to load the library and attach the dataset:

```
library(ISLR)
attach(Default)
```

In the next step, we define a single-variable model with **student** as the only predictor, and additionally a two-variable model including **student** and **balance** as predictors.

```
StudentModel=glm(default~student,family="binomial",data=Default)
summary(StudentModel)
SBModel=glm(default~student+balance,family="binomial",data=Default)
summary(SBModel)
```

In the next step, we first create a curve showing the predicted default probabilities of students at various balance values. We then add a new curve to the plot to show the predicted default probabilities for non-students as well:

```
curve(predict(SBModel,data.frame(balance=x,student="Yes"),type="response"), from = 0,
to = 3000,col="red",xlab="Balance ($)",ylab="Estimated probability of
default",cex.lab=1.5,cex.axis=1.2,lwd=2)
curve(predict(SBModel,data.frame(balance=x,student="No"),type="response"),add=TRUE,
from = 0, to = 3000,col="grey30",lwd=2)
```

How did we ensure that the curve command does not create a new plot for the non-student curve instead of adding a new curve to the same plot? Find the corresponding argument in the command line!

Now, add horizontal dashed lines to this graph representing the probability of default for students and non-students in the single-variable model **StudentModel**! This will allow us to compare the relationship of estimated probabilities for students and non-students in the different models and notice confounding. To add the lines, use the **abline** command with appropriate arguments – recall that **col** determines color and **lty="dashed"** can be used to draw dashed lines.

Finally, we create a legend for this figure:

```
legend("topleft", legend = c("Student, multivariate model","Non-student, multivariate
model","Student, one-variable model","Non-student, one-variable model"),
    col = c("red","grey30","red","grey30"),
    lty = c("solid","solid","dashed","dashed"), lwd = 2)
```

Note: the legend was created independently of the actual values. To demonstrate this, we can add another, meaningless legend in the bottom right corner:

```
legend("bottomright", legend = c("Ice cream","Sour
cream"),col=c("chocolate","orange"),lty=c("dotted","solid"), lwd = c(5,1))
```

4. This exercise addresses multinomial regression in R. First, we replicate the output given in exercise 5 yesterday about travel mode choice depending on the associated cost and time values of the alternatives. The function we use to fit multinomial regression is **multinom** which is included in the **nnet** library, and the travel mode data is in the dataset **Mode**,

included in the **Ecdat** library. Therefore, we start with loading these libraries (and installing them first if the commands below return an error message):
library(nnet)
library(Ecdat)

To get familiar with the analyzed dataset **Mode**, we first look at it:
View(Mode)

Now, we specify a reference level that the model should make comparisons against. This can be done using the relevel function:
Mode$choice=relevel(Mode$choice,ref="car")

We can finally fit a multinomial model with **choice** as response and all other variables in the **Mode** dataset as predictors. You can write out the names of the predictors if you wish, but the simplest way is to just write a dot after the tilde character, as follows:
multinom(choice~.,data=Mode)

For an easier overview of the results, we can reduce the number of digits displayed by R using the **options(digits = …)** command, where **…** is replaced by the desired number of digits. We set this to 2 below and re-run multinomial modelling to get the results again:
options(digits = 2)
multinom(choice~.,data=Mode)

We can also check what happens when we change the reference level:
Mode$choice=relevel(Mode$choice,ref="rail")
multinom(choice~.,data=Mode)

Compare the coefficient estimates for this and the previous output to understand the effect of changing the reference level!

5. If you are done with the previous exercises, do parts a) and b) of Exercise 11 in Section 4.7 of ISL (pages 171-172). To access the data, recall that the **Auto** dataset is in the **ISLR** library.

6. If you would like to ask something or give feedback, feel free to talk to me or enter your question/feedback at www.menti.com, using the code 91 38 21. Remember that you can also suggest topics to focus on for next week!