Case 1: R Code Introduction

Dr. Avery Haviv
University of Rochester
MKT436 and MKT436R

Winter, 2019

This document introduces common R coding techniques (including those required on assignment 1 [1].) in the context of the Interview Case. Please see that case for details about the dataset, and the concepts. The first thing to do is load the dataset. You can load this dataset using 'import dataset' in the file menu, or you can read the file directly. Given where I saved the file, I use the `setwd` function to change the directory, and the `read.csv` function to read this dataset in:

```
setwd('C:/Users/Avery/Dropbox/Teaching Lectures/Interview Case')
resumeData = read.csv('resumeData.csv')
```

R Functions

In this section we will review all the functions needed to complete the first assignment. All R functions use regular brackets (i.e., '(' and ')'). Below is the `names` function. This function tells you the column names of dataset `resumeData`, which we loaded above. To call the `names` function, we write `names` down, open the brackets, and the put in the variable we want to apply the names function to. In this case, we want to apply it to `resumeData`, so we write:

```
names(resumeData)
```

```
##  [1] "name"         "gender"       "ethnicity"    "quality"
##  [5] "call"         "city"         "jobs"         "experience"
##  [9] "honors"       "volunteer"    "military"     "holes"
## [13] "school"       "email"        "computer"     "special"
## [17] "college"      "minimum"      "equal"        "wanted"
## [21] "requirements" "reqexp"       "reqcomm"      "reqeduc"
## [25] "reqcomp"      "reqorg"       "industry"
```

Similarly, if we want to know the number of observations in this dataset, we can use the `nrow` function.

```
nrow(resumeData)
```

```
## [1] 4870
```

We can include a comment by using the # symbol. The # tells R to ignore all other code on this line. This is helpful because it allows us to clarify our code using full english sentences. For example, we could change the above chunk as follows:

```
#Question 4, get the number of rows in the dataset
nrow(resumeData)
```

```
## [1] 4870
```

Note that the output is the same, even though we added all that text. Without the # symbol, that would lead to an error.

[1] The functions here will allow you to complete the assignment. If you're interested in supplementary reading or want additional source material, I recommend chapters 1, 2, 3, and 8 of this R tutorial , or get in touch with the TAs

The assignment requires the use of a few other functions. First, you need to run a linear regression using the `lm` function. This function requires two arguments, a formula and a data frame. The following code runs a regression where `call` is the dependent variable, and `quality` and city are independent variables:

```
lm(call~quality+city,data=resumeData)
```

```
##
## Call:
## lm(formula = call ~ quality + city, data = resumeData)
##
## Coefficients:
## (Intercept)   qualitylow  citychicago
##     0.10406     -0.01420     -0.02971
```

The first argument of the `lm` function is the formula. In an R formula, you

1. Write the name of the dependent variable, followed by a tilde ' '
2. Write the names of the independent variables, separated by a plus sign '+'

The `lm` function requires a second argument. To seperate different arguments of a function, you use a comma. This argument is called the `data` argument, and we want the dataset to be `resumeData`, so we write `data=resumeData`.

Functions also have optional arguments, which do not need to be present. For the assignment, you can use the `subset` argument to specify that you only want to use a subset of the data. In this example, suppose we only want to look at resumes that had one year of experience. We update the previous function call by setting `subset` to `experience==1`, which ensures that we only have records where a there is one year of experience [2]:

```
lm(call~quality+city,data=resumeData,subset=experience==1)
```

```
##
## Call:
## lm(formula = call ~ quality + city, data = resumeData, subset = experience ==
##     1)
##
## Coefficients:
## (Intercept)   qualitylow  citychicago
##     0.39785     -0.33333     -0.06452
```

Finally, while the regression code above delivers the coefficient estimates, it does not deliver standard errors or statistical significance. We can have R generate these statistics by calling the `summary` function on the previous regression:

```
summary(lm(call~quality+city,data=resumeData,subset=volunteer==1))
```

```
##
## Call:
## lm(formula = call ~ quality + city, data = resumeData, subset = volunteer ==
##     1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.10409 -0.10409 -0.06264 -0.06264  0.96970
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

[2] a single = represents a variable assignment. In this case, that would set the value of `experience` to `1`. Here, `==` tells R to compare two variables and see if they are the same

```
## (Intercept)  0.104086   0.008574  12.139  < 2e-16 ***
## qualitylow  -0.032334   0.035045  -0.923 0.356305
## citychicago -0.041448   0.012513  -3.312 0.000941 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2749 on 2001 degrees of freedom
## Multiple R-squared:  0.006684,   Adjusted R-squared:  0.005691
## F-statistic: 6.732 on 2 and 2001 DF,  p-value: 0.001219
```

Dataset Elements

It is also useful to know how to interact with parts of a dataset. We generally do this with a combination of square brackets (i.e., []) and the money sign $.

Datasets in R are typically stored as a data frame, which is like a spreadsheet in Excel. Each variable is stored as a column in this data frame, and each observation is a row. To see the element in a specific row and column, you would

1. Write the name of the data frame
2. Open the square brackets
3. Write the row number
4. Write a comma
5. Write the column number
6. Close the square brackets

While this looks like a lot of steps, it is much simpler in practice. If we want row 2, column 3 of `resumeData`, we would write:

```
resumeData[2,3]
```

```
## [1] cauc
## Levels: afam cauc
```

Referencing an individual element like this is rare. We are using a programming language because we want to work with many elements simultaneously. In the next example, instead of row 2, we get rows 1 through 5 by writing `1:5`. We also ask for all available columns by leaving the column area empty:

```
resumeData[1:5,]
```

```
##      name gender ethnicity quality  call    city jobs experience honors
## 1 Allison female      cauc     low FALSE chicago    2          6  FALSE
## 2 Kristen female      cauc    high FALSE chicago    3          6  FALSE
## 3 Lakisha female      afam     low FALSE chicago    1          6  FALSE
## 4 Latonya female      afam    high FALSE chicago    4          6  FALSE
## 5  Carrie female      cauc    high FALSE chicago    3         22  FALSE
##   volunteer military holes school email computer special college minimum
## 1     FALSE    FALSE  TRUE  FALSE FALSE     TRUE   FALSE    TRUE       5
## 2      TRUE     TRUE FALSE   TRUE  TRUE     TRUE   FALSE   FALSE       5
## 3     FALSE    FALSE FALSE   TRUE FALSE     TRUE   FALSE    TRUE       5
## 4      TRUE    FALSE  TRUE  FALSE  TRUE     TRUE    TRUE   FALSE       5
## 5     FALSE    FALSE FALSE   TRUE  TRUE     TRUE   FALSE   FALSE    some
##   equal     wanted requirements reqexp reqcomm reqeduc reqcomp reqorg
## 1  TRUE supervisor         TRUE   TRUE   FALSE   FALSE    TRUE  FALSE
```

```
## 2  TRUE supervisor         TRUE   TRUE   FALSE   FALSE    TRUE  FALSE
## 3  TRUE supervisor         TRUE   TRUE   FALSE   FALSE    TRUE  FALSE
## 4  TRUE supervisor         TRUE   TRUE   FALSE   FALSE    TRUE  FALSE
## 5  TRUE  secretary         TRUE   TRUE   FALSE   FALSE    TRUE   TRUE
##                           industry
## 1                    manufacturing
## 2                    manufacturing
## 3                    manufacturing
## 4                    manufacturing
## 5 health/education/social services
```

To improve the readability of our code, we can reference variables by their names, rather than their column numbers. There are two ways to do this. First, we can write the name of the variable in the column area. The following gets the first five names in the dataset:

```
resumeData[1:5,'name']
```

```
## [1] Allison Kristen Lakisha Latonya Carrie
## 36 Levels: Aisha Allison Anne Brad Brendan Brett Carrie Darnell ... Tyrone
```

Notice that we have to use quotation marks around `name`. Without quotation marks, R would look for the `name` variable, which we have not created (`name` does not exist by itself, it is only part of the `resumeData` data frame).

The preferred method to get a single variable is using $. We can get the same result as in the previous code chunk as follows:

```
resumeData$name[1:5]
```

```
## [1] Allison Kristen Lakisha Latonya Carrie
## 36 Levels: Aisha Allison Anne Brad Brendan Brett Carrie Darnell ... Tyrone
```

In this method, we first write the data frame, then $, then the name of the variable, and finally what rows of the dataset we want.

Dataset Subsetting

Instead of looking at specific rows or elements of a dataframe, we can look at a `subset` of the data frame. For example, we might want to investigate the data for female applicants.

The simplest way to do this is using the `subset` function. To use the subset function, we first write `subset`, the name of function, and put the arguments of function inside brackets. The first argument is the name of the data frame we want a subset of, in this case `resumeData`. The second uses the `==` operator to see when the `gender` is equal to `female`. Using the `nrow` function shows we have a smaller data frame.

```
femaleData = subset(resumeData,gender=='female')
nrow(femaleData)
```

```
## [1] 3746
```

To us