Introduction to External Readings for DS1 – Data Concepts and Visualization

The Data Concepts and Visualization Curriculum committee decided to incorporate additional readings on Exploratory Data Analysis and Visualization into the syllabus. The key concepts and methods in these two disciplines have been well explained in well-known books not specific to insurance; a light introduction to these topics including examples specific to insurance is therefore included in the online course.

Why do we have external readings?

Rather than create original material when the topics are well covered in many well respected books, we opted to utilize already existing materials.

The additional readings are taken from:

Visualization

Stephen Few – *Show me the Numbers* - Analytics Press, Second Edition, 2012

Exploratory Data Analysis

Roger Peng *Exploratory Data Analysis with R* – Leanpub, 2016. Available in .pdf format at https://leanpub.com/exdata. Paperback copies of the book can be obtained from popular online book vendors.

William Cleveland, *Visualizing Data* - Hobart press, 1993 (Chapter 2 – Univariate Data)

Visualization and Exploratory Data Analysis

Andrew Gellman and Antony Unwin, "Infovis and Statistical Graphics: Different Goals, Different Views", 2012. Available at http://www.stat.columbia.edu/~gelman/research/published/vis14.pdf

The specific chapters and pages of the readings are provided in the syllabus for DS1 - Data Concepts and Visualization.

Notes on *Visualizing Data*, William S Cleveland, Chapter 2

Visualizing Data by William Cleveland is a classic. Although published in 1993, it has stood the test of time. Cleveland's philosophy and approach to graphically exploring data, introduced in this book, are still taught in statistics courses.

Given the age of the book, however, please keep the following in mind:

- Graphs, although done on a computer, reflect the technology of the time.
- The text contains references to obsolete technology such as floppy disks.
- The data sets used for illustrating Exploratory Data Analysis (EDA) techniques are older data sets. Examples include heights of singers (1979) and fusion times in viewing a stereogram (1975).
- The book predates the explosion in the use of open source tools such as R. Hence there are no data sets to download or R code to use for producing graphs.

Peng's *Exploratory Data Analysis in R* builds on the foundation laid by Cleveland and provides an introduction to using modern tools and publicly available databases in EDA. But Cleveland covers some important topics that Peng does not. For instance, Cleveland contains a more extensive introduction to quantile/quantile plots and box plots and how they are used. Cleveland also provides a thorough discussion of logarithmic and other transformations. The two books combined provide a more complete understanding of EDA and the tools we use today to pursue it.

Notes on R for Exploratory Data Analysis

The primary reference selected for exploratory data analysis is Exploratory Data Analysis with R by Roger Peng. This book was chosen because it provides a practical discussion of most of the fundamental approaches to exploring and understanding data. It does assume some knowledge of R, but actual use of R code is not required to understand the EDA concepts Peng discusses. This document provides a brief introduction to some of the R language features used in Peng.

Get book

A pdf copy of the book is available at: https://leanpub.com/exdata

In addition to purchasing the Peng book from an online retailer, it can also be purchased from Leanpub (http://leanpub.com/exdata).

If you do not use R today, but would like to replicate some of the R code described in the book, follow the steps below in order to install R on your computer and to be able to use an R editor.

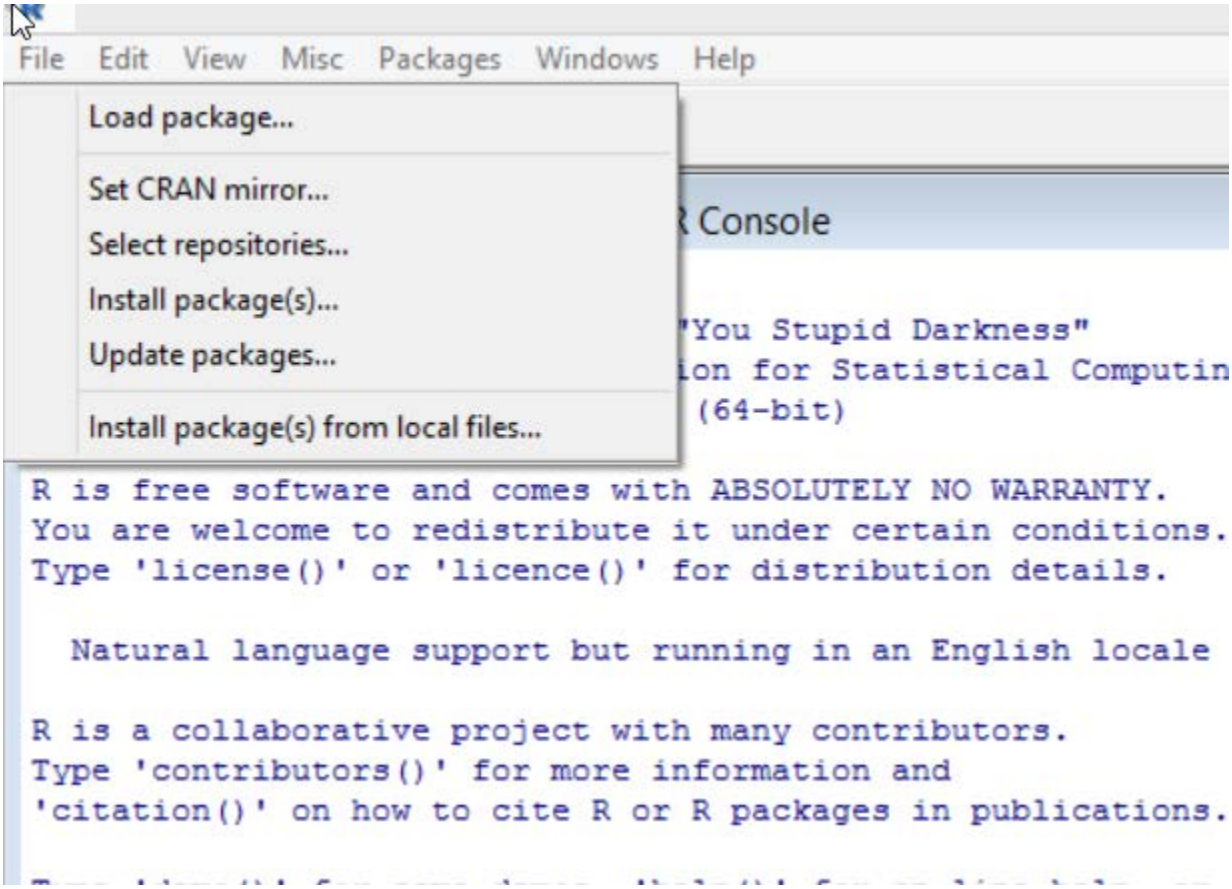
Get data

Sources for many of the data sets used in the book are documented in footnotes, so for many examples you will be able to download the data sets and work through the illustrations that are in the book. However, you can also download the data sets along with an electronic copy of the book from the Leanpub website. There will be a small charge for the data.

Install R

- Go to www.r-project.org
- Click on the CRAN link (on the left side of the screen). Select a CRAN mirror (i.e. one example in the US is the Berkley site but there are many others in many countries)
- Download latest version of R
- This will be an execute (.exe) file
- Click it to install R
- We recommend the RStudio editor.
    - Go to https://www.rstudio.com/
    - Download and install RStudio
    - Other editors such as word and Notepad can be used but you get a lot of added functionality with RStudio

Install packages (such as *dplyr*)



R packages provide many of the tools that can be used in exploratory data analysis. The Peng book makes a lot of use of the *dplyr* package (package and function names are italicized). This package must be installed in R before it can be used. The R software contains a tab for packages at the top. You need to click on this and select any packages you want installed. Alternatively, the tools menu in RStudio provides an option for installment packages.

Read data

To read data using the *read_csv* function as done by Peng you will need to install the *readr* package: https://cran.r-project.org/web/packages/readr/README.html .

The data sets used by R are commonly stored in text files (i.e., space, tab, or comma delimited). Although Peng uses the *read_csv* function to read in data, the *read.table* (and related *read.csv*) function is also commonly used. The following code would read comma delimited data from the file SampleData in the directory Data, where the data file contains variable names in the first row.

```
traindata<-read.csv("C:/WPData/SampleData.csv" ,header=TRUE)
```

To get help on the use of the *read.table* (*read.csv*) functions, type ?*read.table* into your editor and run the code or type it into the command line of R. This will bring up a help file that explains how to use the function.

R operations: Arithmetic

R performs basic arithmetic operations of addition, subtraction, multiplication, and division, along with other common mathematical procedures as illustrated below. The syntax for performing the arithmetic operations is the same or similar to that of many other programming languages. However, the basic unit for any operation in R is a vector, which in the illustrations below is a vector of numbers. Thus, addition is performed by adding two vectors and multiplication by multiplying two vectors. This should be kept in mind when performing operations in R. Ideally, the vectors will be the same size; if not, R has rules for replicating the elements of one vector to match the number in the other vector and R will issue a warning that the vectors are not the same size.

>x<-c(1,2,3,4,5)

>y = c(6,7,8,9,10)

Note that in the example above both "<-" and the = can be used as the equal sign in an expression. Also note the "c" or combine operator which is used to combine elements, in this case numbers, in a vector. The variables in the example are named x and y. In R, variable names are case sensitive so X and Y would refer to different variables. In the examples below, R code is shown on lines beginning with a greater than symbol (">"), as this is how code appears in the command window of R. The results of R code after it has been run are also shown below. Note in the examples R output is shown in slightly larger font.

add 2 variables

>z<-x+y

The pound sign is used for commenting:

print z by typing variable name and pressing enter. The result is indexed to [1] as it is the first line printed

>z

```
[1]  7  9 11 13 15
```

this is the same as print(z)

>print(z)

```
[1]  7  9 11 13 15
```

>z=x*y

>z

```
[1]  6 14 24 36 50
```

square z

>z=x^2

>z

```
[1]  1  4  9 16 25
```

take natural log of z

>z=log(x)

>z

```
[1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379
```

Statistical operations.

R has a number of statistical operations that are often used, including mean, standard deviation, variance, minimum, and maximum.

Below are illustrations of applying the common statistical functions:

>mean (x)

```
[1] 3
```

>sd(x)

```
[1] 1.581139
```

```
>cor(x,y)
```

```
[1] 1
```

Type

>?`stats-package

for more information on the statistics available in base R.

Tabular summaries

R has several important functions used to summarize data. In order to produce a frequency table or a crosstabulation, use the *table* function:

>class<-c("A","B","B","A","A","C")

produce a frequency table

>table(class)

```
A B C
3 2 1
```

produce a cross tabulation from two variables

>x2<-c(1,2,3,1,2,3)

>table(x2,class)

```
   class
x2  A B C
  1 2 0 0
  2 1 1 0
  3 0 1 1
```

In order to obtain a mean or other summary statistic by a grouping variable, use the *tapply* (table apply) function. The function requires a vector or list of numbers, a grouping variable, and the statistic desired for each group:

x2.mean<-tapply(x2,class,mean)

x2.mean

```
       A        B        C
1.333333 2.500000 3.000000
```

Use the help function to find out more about the *tapply* function and how to use it:

?tapply

<u>data frames</u>

The most commonly used form of data when using R for data analysis is the data frame. A data frame can be created by using the "data.frame" function. The x2 and class vectors created, which are the same length, can be combined into one data frame with the following code:

>test.data<-data.frame(x2,class)

>test.data

```
x2 class
1  1     A
2  2     B
3  3     B
4  1     A
5  2     A
6  3     C
```

Now let's see how many rows are in the data with the nrow function

>nrow(test.data)

```
[1] 6
```

We can use the names functions to get the variable names of the variables in a dataframe.

>names(test.data)

```
[1] "x2"    "class"
```

Specific variables in a data frame can be referenced in several ways. The first is to reference the variable using the "$" symbol to reference the variable after the data frame name.

>class=test.data$class

The second is to use array index references. An element of a data frame can be referenced by its row and column number within square brackets. To reference one cell of a data frame, reference the row and column in square brackets:

>test.data[1,2]

```
[1] A
```

To reference a variable, i.e., all the rows of a given column, use:

>class=test.data[,2] or class=test.data[2]

>class

```
[1] A B B A A C
Levels: A B C
```

When using the read.table or read.csv function to read in data, the data read in is by default a data frame. Peng, beginning in Chapter 5 of *Exploratory Data Analysis with R*, makes heavy use of the readr package, as it is more efficient in reading in large data sets. You can use read_csv, instead of read.csv. The result of using read_csv is a tibble (table dataframe) which has some properties and features that R data frames do not have. To illustrate, we will read in some data (the first 10,000 rows of ozone data) used in chapter 5:

use read.csv to read data

>data<-read.csv("C:/PengData/hourly_44201_2014.csv",nrows=25000)

>names(data)

```
 [1] "State.Code"          "County.Code"
 [3] "Site.Num"            "Parameter.Code"
 [5] "POC"                 "Latitude"
 [7] "Longitude"           "Datum"
```

```
 [9] "Parameter.Name"      "Date.Local"
[11] "Time.Local"          "Date.GMT"
[13] "Time.GMT"            "Sample.Measurement"
[15] "Units.of.Measure"    "MDL"
[17] "Uncertainty"         "Qualifier"
[19] "Method.Type"         "Method.Name"
[21] "State.Name"          "County.Name"
[23] "Date.of.Last.Change"
```

Now the read_csv version:

use read_csv to read data

install library readr

>install.packages("readr")

load library

>library(readr)

read in the ozone hourly data

>data2<-read_csv("C:/PengData/hourly_44201_2014.csv",n_max=25000)

>names(data2)

```
 [1] "State Code"          "County Code"
 [3] "Site Num"            "Parameter Code"
 [5] "POC"                 "Latitude"
 [7] "Longitude"           "Datum"
 [9] "Parameter Name"      "Date Local"
[11] "Time Local"          "Date GMT"
[13] "Time GMT"            "Sample Measurement"
[15] "Units of Measure"    "MDL"
[17] "Uncertainty"         "Qualifier"
[19] "Method Type"         "Method Name"
[21] "State Name"          "County Name"
[23] "Date of Last Change"
```

The variable names have spaces in them in the data read in by read_csv reflecting the variable names in the original file. The read.csv converts the variable names to have a period rather than a space separator separating two words in a name. Under the tibble formatting convention the spaces in the variable names are allowed, however you will need to utilize help functions about referencing variables with spaces in them.[1] In chapter 5, variables are referred to using variable names with periods rather than spaces in them. There are two ways to change the variable names to remove the spaces.

1. Use the names function per the example in Peng:
   names(data2)<-make.names(data2))
2. Use the data.frame function which will convert the data to a data frame with periods in the variable name

[1] In general, it is necessary to put quotes around the variable name as in data2$'State Name'

```
data2<-data.frame(data2)
```

Packages as sources of many functions

Certain R functions such as the *mean* function are available in base R. However, many of the R functions data scientists use are in R packages. For instance, the *read_csv* function is contained in the R readr package. It is necessary to install the packages you will be needing before you can use them.

One of the packages that Peng makes a lot of use of is the dplyr package. The dplyr package is a data management package that has many of the same data management functions that SQL possesses and that other R functions could be used for, though they are often less efficient. A good introduction to the dplyr package is provided in Chapter 4[2]. Some of the dplyr functions include select() for selecting variables from a database, filter() for selecting records from a database, arrange() for sorting records, and mutate() to produce transformations of variables. In addition, dplyr uses the pipe operator "%>%"[3]. The pipe operator is used to organize steps of data processing into one statement, as opposed to a sequence of functions.

The user will need to make sure all needed packages are installed before they start programming in R. In addition, to use a library (i.e., package), the library() function will need to be run before the code that calls functions from the package.

R Quick Introduction References

The following R references are recommended for those who seek additional introductory information about R.

Gareth J, Witten D, Hastie T., Tibshirani R, *An Introduction to Statistical Learning with Applications in R*, 2013 Springer. Available for free download as a pdf file. See especially chapter 2.3, "Lab Introduction to R"

Peng, Roger R Programming for Data Science, 2014, Leanpub, available free at ttps://leanpub.com/rprogramming. See especially Chapter 5 "R Nuts and Bolts". The various chapters of the book have accompanying videos that serve as tutorial lessons on R.

[2] See also the section on dplyr in Peng, Roger R Programming for Data Science, 2014, Leanpub, available free at https://leanpub.com/rprogramming

[3] See https://cran.r-project.org/web/packages/magrittr/vignettes/magrittr.html for an introduction to the magrittr package and the pipe operator.