Lab 0 - Exploring a New Cyber Dataset

Skills Needed

- Basic R language skills

Time Required

- Lab and reading should be accomplished in 20 to 30 minutes.

Background

Your analysis supervisor has given you a data file and ask you to review the file for anything interesting. This is a new data product customer and little is known about the structure. Your first task is to load the file, conduct data cleaning, and then display a sample of the data.

Acquiring the Dataset

In real life, your team should have acquired the dataset from a data product customer already or they may hand your a CD or USB storage device. For this lab we will be working with a dataset from SecRepo website [1]. This website hosts a set of links to security related datasets that are mostly under the Creative Commons [2] with some exceptions.

Cyber datasets can be very large and for that reason, many administrators compress the file to save disk space and transmission time when copying the file. We will discuss two methods for accessing a file and you should try both methods as these methods will cover accessing a weblink or a file stored on media.

Methods for accessing datasets in this lab include:

- Download the dataset from SecRepo and save it to your local drive. This provides the analyst with the flexibility to disconnect from the network or edit file to correct data collection errors. Cyber data is captured by devices and programs that sometimes emit garbage lines that is . This is the case with the lab dataset.
- Read the file from a weblink. We will read the file from the weblink after we understand what is in the file and the adjustments that we will need to make to the dataset to allow it to be read into R.

While this part of the lab may seem tedious, a 2016 data scientist Crowd Flower survey indicates that the bulk of the data scientist day is spent cleaning and organizing data with collecting data as a close second [3]. The exercises in this lab will walk you through the errors you will receive and how to mitigate the errors.

[1] http://www.secrepo.com/
[2] https://creativecommons.org/licenses/by/4.0/
[3] http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower__DataScienceReport__2016.pdf

Excercise 1

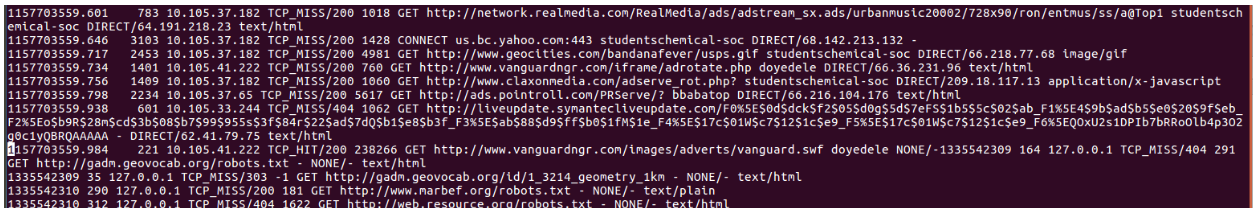
Before we run, download the file and store it locally on your machine Download file. A work horse function is read.table but the compressed file will not be able to be read.

Notice the errors that are output after the code block.

```
# Method 1a - Retrieve the data from a web link and store in a variable
#Clean up the environment
# rm(list=ls())
# ds1 <- read.table('http://www.secrepo.com/squid/access.log.gz', header = TRUE)
```

The commented out code block throws an error when run: line 1 appears to contain embedded nullsline 5 appears to contain embedded nulls Show Traceback Error in make.names(col.names, unique = TRUE) : invalid multibyte string 1

R throws an error stating that file contains embedded nulls. The problem is not the nulls as the file type is a gzip compressed file. You can determine that it is a compressed file as file ends in .gz. We will need use the R function gzfile to open a connection to the gz file. We can then read connection using read.table. This will present another problem in getting the data. Line 82,947 does not have ten elements in the row. You can open the file in your favorite text editor and read the line.

image 

```
1157703559.601    783 10.105.37.182 TCP_MISS/200 1018 GET http://network.realmedia.com/RealMedia/ads/adstream_sx.ads/urbanmusic20002/728x90/ron/entmus/ss/a@Top1 studentsch
emical-soc DIRECT/64.191.218.23 text/html
1157703559.646   3103 10.105.37.182 TCP_MISS/200 1428 CONNECT us.bc.yahoo.com:443 studentschemical-soc DIRECT/68.142.213.132 -
1157703559.717   2453 10.105.37.182 TCP_MISS/200 4981 GET http://www.geocities.com/bandanafever/usps.gif studentschemical-soc DIRECT/66.218.77.68 image/gif
1157703559.734   1401 10.105.41.222 TCP_MISS/200 760 GET http://www.vanguardngr.com/iframe/adrotate.php doyedele DIRECT/66.36.231.96 text/html
1157703559.756   1409 10.105.37.182 TCP_MISS/200 1060 GET http://www.claxonmedia.com/adserve_rot.php? studentschemical-soc DIRECT/209.18.117.13 application/x-javascript
1157703559.798   2234 10.105.37.65 TCP_MISS/200 5617 GET http://ads.pointroll.com/PRServe/? bbabatop DIRECT/66.216.104.176 text/html
1157703559.938    601 10.105.33.244 TCP_MISS/404 1062 GET http://liveupdate.symantecliveupdate.com/F0%5E$0d$dck$f2$05$d0g$5d$7eFS$1b5$5c$02$ab_F1%5E4$9b$ad$b5$e0$20$9f$eb_
F2%5Eo$b9R$28m$cd$3b$08$b7$99$955s$3f$84r$22$ad$7dQ$b1$e8$b3f_F3%5E$ab$88$d9$ff$b0$1fM$1e_F4%5E$17c$01W$c7$12$1c$e9_F5%5E$17c$01W$c7$12$1c$e9_F6%5EQOxU2s1DPIb7bRRoOlb4p3O2
g0c1yQBRQAAAAA - DIRECT/62.41.79.75 text/html
1157703559.984    221 10.105.41.222 TCP_HIT/200 238266 GET http://www.vanguardngr.com/images/adverts/vanguard.swf doyedele NONE/-1335542309 164 127.0.0.1 TCP_MISS/404 291
GET http://gadm.geovocab.org/robots.txt - NONE/- text/html
1335542309 35 127.0.0.1 TCP_MISS/303 -1 GET http://gadm.geovocab.org/id/1_3214_geometry_1km - NONE/- text/html
1335542310 290 127.0.0.1 TCP_MISS/200 181 GET http://www.marbef.org/robots.txt - NONE/- text/plain
1335542310 312 127.0.0.1 TCP_MISS/404 1622 GET http://web.resource.org/robots.txt - NONE/- text/html
```

Notice how alphanumerics flow down the left side of the page with whitespace delineating columns. Line 82947 has a burst of characters. In the field in may be essential to correct every line of a file. In this case you would contact your subject matter expert and ask them to interpete the line for you.

```
# Method 1b - Retrieve the data from a web link and store in a variable
#Clean up the environment
rm(list=ls())
ds1 <- gzfile('~/github/Lab-0---Exploring-Cyber-Datasets/Data/access.log.gz')
ds1.table <- read.table(ds1, fill = TRUE, stringsAsFactors = FALSE)
colnames(ds1.table)<- c("time", "elapsed", "remotehost", "code/status", "bytes", "method", "URL", "rfc9
```

When we read the table, V# is assigned to each row where # is a number. The Squid documentation provides the 10 column names that are the default which we set using colnames. We could have used read.table with the Header = TRUE but the data file does not contain headers.

The last line of code in this R chunk counts the number of NA's in the dataset. As a matter of practice you should count NAs in your data. Knowing the number of missing values will help the reliability of your statistics.

Using TidyR to Restructure Data

Now that we have data loaded, we need to structure the data for analysis. Most of the R visualization packages will require a column and row order for input to the visualization function. A good place to start with structuring data is the Tidyr package. The Tidyr (tidyr) is package has data manipulation functions to make it easier. Data science guidelines for data structure for structure data are:

- Columns - each column represents a variable
- Rows - each row is an observation

- Cell - each cell in the row by column intersection is a value.

Data structure manipulation is a valuable technique to learn as other analysis functions will demand that your data be structured coorrectly to fit into the key-value pairs. Horizontal rows should be observations with the column as the value you are interested in analysing.

```
ds1.table$time <- as.POSIXct(as.numeric(ds1.table$time), origin = "1960-01-01")
```

```
## Warning in as.POSIXct(as.numeric(ds1.table$time), origin = "1960-01-01"):
## NAs introduced by coercion
```

```
ds1.table$elapsed <- as.numeric(ds1.table$elapsed)
```

```
## Warning: NAs introduced by coercion
```

```
ds1.table$bytes <- as.numeric(ds1.table$bytes)
```

```
## Warning: NAs introduced by coercion
```

```
sum(is.na(ds1.table))
```

```
## [1] 3
```

```
ds1.table <- ds1.table[!is.na(ds1.table)]
```

This R code chunk converts from character vectors to time and numeric. During the conversion NAs will be introduced. We use the is.na function to identify and remove the NAs that were introduced. At this point we are ready to save the data for use later.

```
saveRDS(ds1.table, '~/github/Lab-0---Exploring-Cyber-Datasets/Data/lab1.rds')
```

The code above saves your environment data and values in a compressed format. When exciting R Studion you will be asked if you wish to save your desktop you which should answer ‘no’. Answering ‘yes’ will save anything you have in the evironment to .RData. Saving your desktop to .RData file that will load the next time you run R Studio. This will prevent saving a large data structure to the .RData file.

Summary

In this lab we:

- Discovered that R errors should be evaluated in the context of what you are doing. R errors are not always evident as to what is throwing the error. You may need to comment out lines and rerun one line at a time until you find the offending line. Using a Google search with ‘[R]’ in your query focuses the results on R language information which helps save time.
- Loaded a compressed file after creating a connection that allows you to access the compressed file. Cyber logs are text files made up of alphanueric characters. The file in is lab is approximately 23.1 megs and uncompresses to 220.1megs. Compressing a file save disk space, but the R programmer needs to consider memory implications. You can quickly determine a files uncompressed size by using gunzip -c <filename.gz> | wc –bytes. This will print out the decompressed value without uncompressing the file.

- Converted values into numeric values that can be analyzed. Part of being able to report on the variables in a dataset is being able to sum, count, average, etc. You need numeric values to perform basic statistic analysis.

- Saved the data to a compressed file for later use. Instead of saving the environment variables to .RData we saved to our own RDS file.

Endnotes

Contact Information

https://www.linkedin.com/in/brian-thomas-ph-d-pmp-b113381/