Part 4: Understanding Data Sets

Summary

This Part presents an example of reading in a data set, and ensuring the data is fully understood and properly represented prior to any further analysis.

The ability to read in, clean, and understand data sets is a crucial skill. Data do not typically come in a convenient form. Careful consideration should be given to exploring the data to ensure full understanding of the nature of the data set.

Example Data Set

First, we will read in a data set that we will use in our following examples.

Visit the website

https://www.sec.gov/opa/data/market-structure/marketstructuredownloadshtml-by_security.html

and download the data from the first quarter of 2017. Create a directory for working on these examples, and place this file in that directory.

These data come from the U.S. Securities and Exchange Commission repository on market structure. This data set contains daily characteristics of a range of market variables over 5000 equities and ETFs, accumulated over several exchanges.

Be sure that the working directory is correctly set in R.

Now use the `read.table()` function to read in the file:

```
> fulldata = read.table("q1_2017_all.csv", sep=",",
+                        header=T,quote="")
```

This is a "csv" file, i.e., there are "comma-separated-values". This leads to the use of `sep=","`. The `header=T` argument specifies that the first line of the file gives the variable names.

Exercise: What happens if you use `read.table()` in the example above without `quote=""`?

Understanding the Variables

Whenever a new data set is encountered, one should fully understand its variables. Specific questions to address for each variable:

1. Describe the information that is stored in this column.
2. What type of variable is it? (Timestamp, ordinal factor, categorical factor, ratio scale, other?) Be sure that R represents the variable correctly.
3. Are there any missing values? What is the source of the missingness?
4. Are there any extreme/inappropriate values? What is the source of the extreme value?

Look at the variables:

```
> names(fulldata)
 [1] "Date"                     "Security"
 [3] "Ticker"                   "McapRank"
 [5] "TurnRank"                 "VolatilityRank"
 [7] "PriceRank"                "LitVol..000."
 [9] "OrderVol..000."           "Hidden"
[11] "TradesForHidden"          "HiddenVol..000."
[13] "TradeVolForHidden..000."  "Cancels"
[15] "LitTrades"                "OddLots"
[17] "TradesForOddLots"         "OddLotVol..000."
[19] "TradeVolForOddLots..000."
```

Background on the data can be found in the associated `README` file, available on Canvas in the “Data Sets” folder in the “Files” page. We will reference portions of it below.

Column 1: Date

The first column in `fulldata` is a date stamp for the observation. R has special functions for handling dates which will prove useful later when working with time series.

The `as.Date()` function transforms into the date format:

```
> print(fulldata$Date[1])   # This is Jan. 3, 2017
[1] 20170103
```

```
> fulldata$Date = as.Date(as.character(fulldata$Date),
+                               format="%Y%m%d")
```

Note that it is necessary to cast the dates as strings, and then specify the format of those strings.

Exercise: Explain the use of the `format` argument to `as.Date()`.

Columns 2 and 3: Security and Ticker

The variables `Security` and `Ticker` are already appropriately treated as factors by R:

```
> is.factor(fulldata$Sec) & is.factor(fulldata$Tick)
[1] TRUE
```

We also note that `Security` is either `Stock` or `ETF` (Exchange Traded Fund), and that there are over 5,000 equities under consideration:

```
> levels(fulldata$Security)
[1] "ETF"   "Stock"
```

```
> nlevels(fulldata$Ticker)
[1] 5338
```

Exercise: How many of the different ticker symbols are ETFs?

Exercise: Execute the commands below, and discuss what you find:

```
> length(unique(fulldata$Date))
> table(table(fulldata$Ticker))
> which.max(table(fulldata$Ticker))
> fulldata[duplicated(fulldata[,c(1,3)]),]
```

We will remove the duplicated lines using the following command:

```
> fulldata = fulldata[!duplicated(fulldata[,c(1,3)],
+                                 fromLast=TRUE),]
```

Note that by using `fromLast=TRUE`, we remove the **first** of each pair of duplicates.

Columns 4 through 7: Rank Variables

The next four columns provide ranks of the stocks/ETFs with respect to key attributes of Market Capitalization, Turnover, Volatility, and Price.

In general, we would prefer to start with data in its most “raw” format, and one could construct variables such as these from that data. Alas, we are often left to work with what is avaiable.

Exercise: Inspect the output of

```
> table(fulldata$VolatilityRank, fulldata$Security,
+       fulldata$Date)
```

What does this tell us about the structure of these variables? Why should we be very careful with these variables?

We will change these into ordered factors to reflect their ordinal nature:

```
> fulldata$McapRank =
+   factor(fulldata$McapRank, ordered=TRUE)
> fulldata$TurnRank =
+   factor(fulldata$TurnRank, ordered=TRUE)
> fulldata$VolatilityRank =
+   factor(fulldata$VolatilityRank, ordered=TRUE)
> fulldata$PriceRank =
+   factor(fulldata$PriceRank, ordered=TRUE)
```

Columns 8 thru 19: The Count Variables

The remaining column in the data set consist of trade and share counts of different types.

Note that some of the volume counts are reported in 1000's of shares, hence there are non-integer values among the counts.

The `summary()` function is a useful first step in understanding the basic properties of the variables. A primary concern is the presence of extreme or impossible values.

```
> summary(fulldata[,8:19])
  LitVol..000.       OrderVol..000.        Hidden        TradesForHidden
 Min.   :     0.00   Min.   :      0   Min.   : -143.0   Min.   :     0
 1st Qu.:     3.46   1st Qu.:    885   1st Qu.:   10.0   1st Qu.:    41
 Median :    44.57   Median :   4003   Median :   75.0   Median :   501
 Mean   :   431.76   Mean   :  35747   Mean   :  454.4   Mean   :  3476
 3rd Qu.:   254.41   3rd Qu.:  15581   3rd Qu.:  370.0   3rd Qu.:  2846
 Max.   :119014.06   Max.   :7293901   Max.   :89335.0   Max.   :533874
 HiddenVol..000.    TradeVolForHidden..000.    Cancels
 Min.   :-7903.64   Min.   :     0.00       Min.   :      0
 1st Qu.:    1.15   1st Qu.:     5.64       1st Qu.:   3126
 Median :    8.74   Median :    55.31       Median :  15583
 Mean   :   66.91   Mean   :   498.04       Mean   :  73579
 3rd Qu.:   44.02   3rd Qu.:   301.58       3rd Qu.:  60112
 Max.   :14821.94   Max.   :133714.59       Max.   :7607714
```

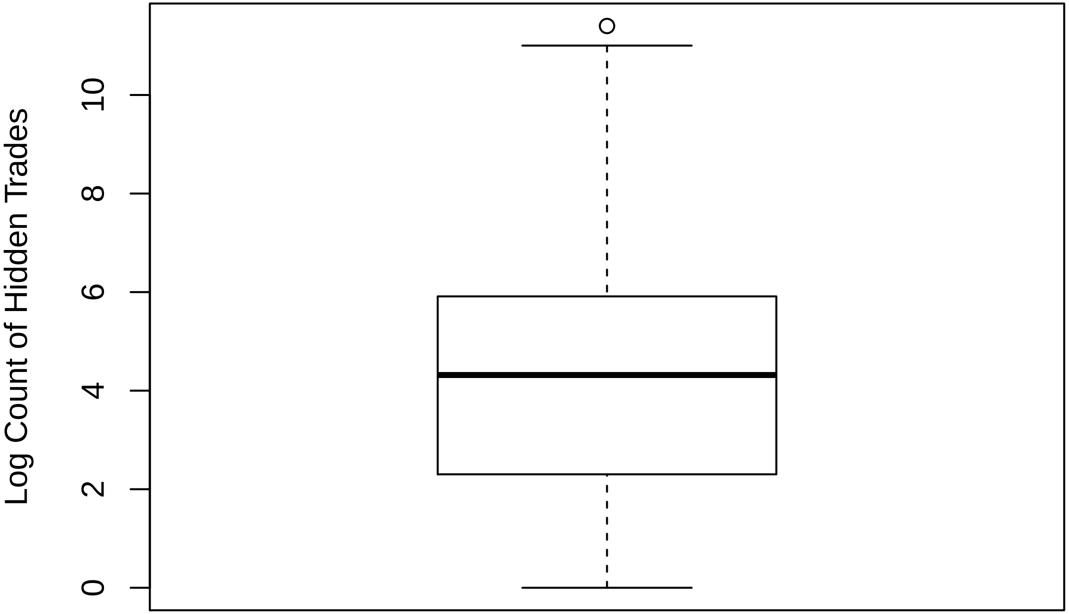
```
  LitTrades         OddLots       TradesForOddLots OddLotVol..000.
Min.   :     0   Min.   :     0   Min.   :     0   Min.   :   0.000
1st Qu.:    23   1st Qu.:     8   1st Qu.:    39   1st Qu.:   0.280
Median :   377   Median :   154   Median :   464   Median :   5.254
Mean   :  2708   Mean   :   916   Mean   :  3115   Mean   :  32.492
3rd Qu.:  2186   3rd Qu.:   887   3rd Qu.:  2558   3rd Qu.:  31.083
Max.   :383222   Max.   :130106   Max.   :462414   Max.   :3728.885
TradeVolForOddLots..000.
Min.   :     0.00
1st Qu.:     5.31
Median :    49.94
Mean   :   422.78
3rd Qu.:   260.18
Max.   :108034.79
```

Exercise: Do any of these variables take values that would seem to be impossible? Can you speculate as to the source of these values?

Of course, graphical tools are useful for understanding the properties of variables. This topic will be covered more fully in the next Part, but here we consider a classic approach: the boxplot:

```
> boxplot(log(fulldata$Hidden),
+       ylab="Log Count of Hidden Trades")
```

The "box" of the plot extends from the 25th to the 75th percentile of the data, while the solid line in the middle of the box is at the median. The two "arms" extend to the minimum and maximum value, **except** that any value labelled an outlier is plotted separately.


Log Count of Hidden Trades
0
2
4
6
8
10

Exercise: Why was the logarithm of the variable utilized in this case? What issues did that create?

Exercise: Consider the outlier in the previous plot. Is this observation also extreme in the other variables? Can you find the source of this behavior?

Exercise: Comment on the output of

```
> apply(fulldata[,8:19],2,which.max)
```

Missing Data

Most data sets contain some amount of missing data, for many reasons.

R uses the special symbol `NA` to represent a missing value.

As a first step, it is important to recognize how your data set represents missing values, so that R handles them properly. The function `read.table()` has an argument `na.strings` to allow one to specify missing values. **Be careful:** It is not unusual for data sets to use numbers such as `-999` to indicate a missing value.

In a `.csv` file such as that we are using here, it is common to simply have blank values to indiciate missingness, i.e., there are consecutive commas without a value between.

Exercise: Does R appropriately handle cases where a missing value is indicated using blank values in a `.csv` file?

A useful tool is the function `md.pattern()` that is part of package `mice`. This function shows the different “patterns” of missingness in the data set, and how often they occur.

Consider the example on the following page, the result of calling

```
> md.pattern(data.frame(fulldata), plot=FALSE)
```

This shows that there are 325,082 complete cases in the data set. There is one case where `VolatilityRank` is missing, and there are 76 cases where `McapRank`, `TurnRank` and `PriceRank` are missing.

```
       Date Security Ticker LitVol..000. OrderVol..000. Hidden
325082    1        1      1            1              1      1
76        1        1      1            1              1      1
1         1        1      1            1              1      1
          0        0      0            0              0      0
       TradesForHidden HiddenVol..000. TradeVolForHidden..000. Cancels
325082               1               1                       1       1
76                   1               1                       1       1
1                    1               1                       1       1
                     0               0                       0       0
       LitTrades OddLots TradesForOddLots OddLotVol..000.
325082         1       1                1               1
76             1       1                1               1
1              1       1                1               1
               0       0                0               0
       TradeVolForOddLots..000. VolatilityRank McapRank TurnRank PriceRank
325082                        1              1        1        1         1
76                            1              1        0        0         0
1                             1              0        1        1         1
                              0              1       76       76        76

325082   0
76       3
1        1
       229
```

Exercise: Inspect the output from

```
> fulldata[!complete.cases(fulldata),]
```

What does this say about the cases with missing values?

Of course, more sophisticated visualization of the cases with missing data would likely provide more information as to the source of the missingness. In the next Part we will consider such tools.