How to export and merge tables, graphs, and raw data from R to a single Excel file; which has multiple sheets.

As published in Benchmarks RSS Matters, September 2013

http://web3.unt.edu/benchmarks/issues/2013/09/rss-matters

Jon Starkweather, PhD

Jon Starkweather, PhD
jonathan.starkweather@unt.edu
Consultant
Research and **S**tatistical **S**upport

http://www.unt.edu

http://www.unt.edu/rss

RSS hosts a number of "Short Courses".
A list of them is available at:
http://www.unt.edu/rss/Instructional.htm

Those interested in learning more about R, or how to use it, can find information here:
http://www.unt.edu/rss/class/Jon/R_SC

How to export and merge tables, graphs, and raw data from R to a single Excel file; which has multiple sheets.

Continuing last month's theme, we again visit Excel – back by popular demand. And again, we are obligated to mention that Excel, as nice as it is, is not a statistical software package. RSS personnel do not recommend using Excel; for data storage, data display, or data analysis. An often quoted phrase[1] is the following; the only thing worse than using SPSS, is using Excel. For more information on the known problems with Excel and other spread sheet based software, see Burns (2013). RSS recommends storing data in plain text (.txt) files with comma delimiters; also known as a comma separated values (.csv) file type. The reason RSS recommends text (.txt) or comma separated values (.csv) file types is because those file types can be easily opened or imported into all the statistical software packages. However, if you feel you must use Excel, then this article may help you with the common task of getting tables, graphs, and data from R[2] into a single Excel file with one sheet for the table(s) and graph(s), and another sheet for the raw data.

Example

First, import the data. The data used in this example is available on the RSS server and can be accessed using the URL from the script below. Simply copy and paste the script below into an R console to import the data directly into R, naming it ts.df. Notice, the data has 6000 rows of 9 columns (1 time index & 8 time series).

```
ts.df <- read.table(
"http://www.unt.edu/rss/class/Jon/Benchmarks/ExcelFiles/time_series_001.txt
   header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
nrow(ts.df)
[1] 6000
ncol(ts.df)
[1] 9
```

Next, set the working directory (setwd) to the location where you want the finished Excel file stored. Here, for this example, we are using the desktop.

```
setwd("C:/Users/jds0282/Desktop/")
```

Next, create an empty table in which some basic descriptive statistics will go. Keep in mind, this table is just being created for the purpose of having a table to export into Excel. The table will contain the length of each time series, the mean of each time series, the standard deviation of each time series, and the mean fractal dimension of each time series. Fractal dimension can be thought of as a complexity measure of each time series.

[1] The phrase is believed to have originated with respected statistician and prominent R user Frank Harrell of Vanderbilt University at the 5th annual Bayesian Biostatistics Conference.

[2] `http://cran.r-project.org`

```
table.1 <- data.frame(matrix(rep(NA,36), nrow = 4))
names(table.1) <- c("stats",names(ts.df[,2:9]))
table.1[,1] <- factor(c("n","mean","sd","mean.fd"))
table.1
    stats ts.1 ts.2 ts.3 ts.4 ts.5 ts.6 ts.7 ts.8
1       n   NA   NA   NA   NA   NA   NA   NA   NA
2    mean   NA   NA   NA   NA   NA   NA   NA   NA
3      sd   NA   NA   NA   NA   NA   NA   NA   NA
4 mean.fd   NA   NA   NA   NA   NA   NA   NA   NA
```

Next, calculate the appropriate simple descriptive statistics and store them in the appropriate cells of the table.

```
table.1[1,2:9] <- rep(nrow(ts.df), 8)
table.1[2:3,2:9] <- data.frame(matrix(c(apply(ts.df[,2:9], 2, mean),
                    apply(ts.df[,2:9], 2, sd)), byrow = T, nrow = 2))
table.1
    stats        ts.1        ts.2        ts.3         ts.4        ts.5
1       n 6000.000000 6000.000000 6000.000000 6000.0000000 6000.000000
2    mean   -1.466314    1.647883   -2.080831    1.3697476   -1.226921
3      sd    2.054721    1.487989    1.170512    0.8129809    2.658949
4 mean.fd          NA          NA          NA           NA          NA
         ts.6        ts.7        ts.8
1 6000.000000 6000.000000 6000.000000
2   -2.065006    1.225880   -1.327785
3    1.016617    2.401352    2.265327
4          NA          NA          NA
```

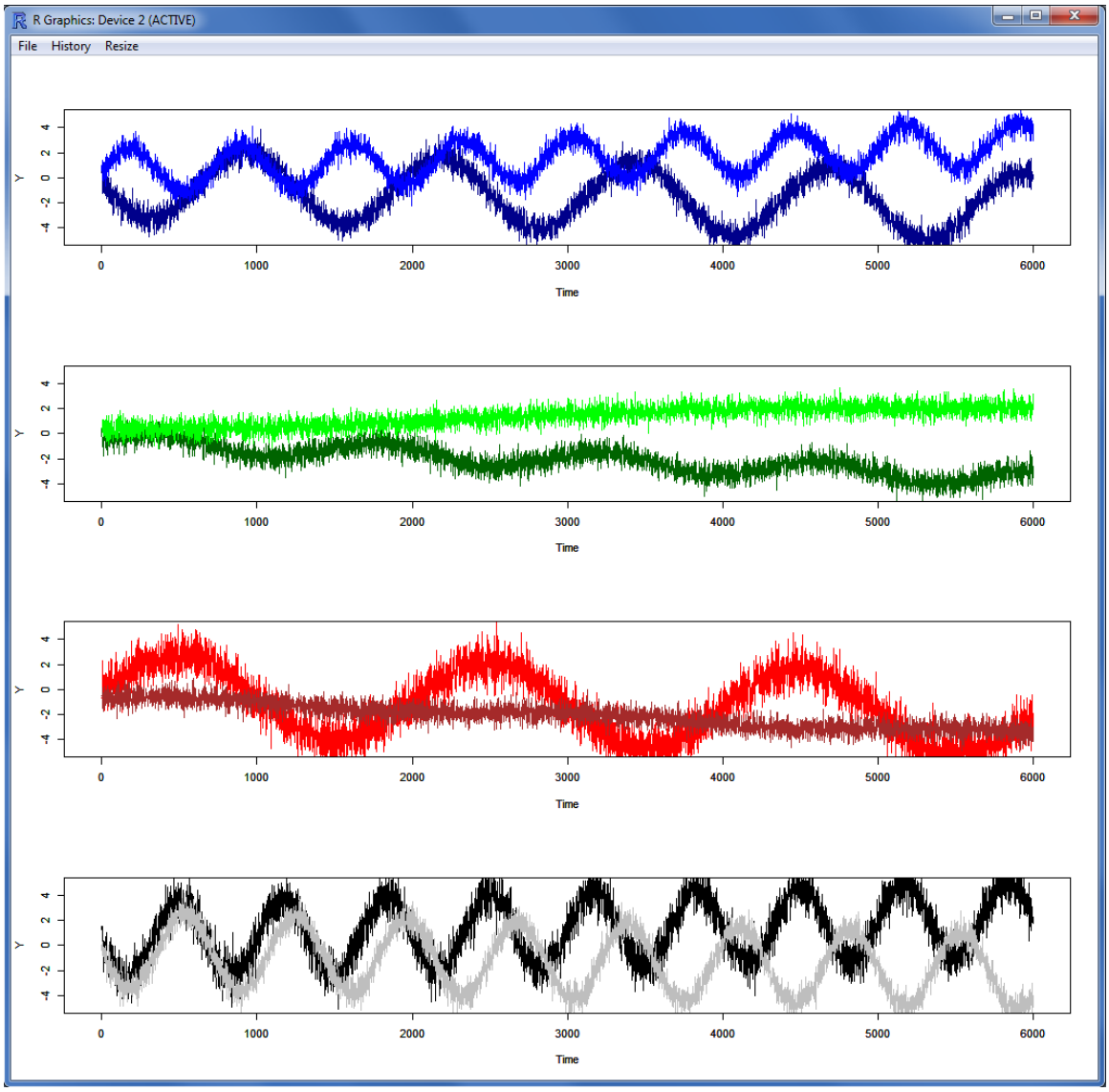
Next, calculate the mean fractal dimension of each time series using a simple for-loop and place the estimates in the appropriate cells of our table. Notice below, there are two necessary packages and it is necessary to specify the window size (w.s) in order to calculate multiple fractal dimension estimates (which will be used to calculate a mean fractal dimension for each time series). Also, notice below, each of the mean fractal dimension estimates are close to 2.0; which reflects the random nature of this simulated data. For more information on fractal dimension, see Gneiting, Sevcikova, and Percival (2010).

```
library(abind)
library(fractaldim)
w.s <- .1*nrow(ts.df); w.s
[1] 600
for(i in 1:8){
  q <- fd.estimate(data = ts.df[,i+1], methods = "madogram",
          window.size = w.s, step.size = w.s, trim = TRUE,
          keep.data = FALSE, keep.loglog = FALSE, parallel = FALSE,
          nr.nodes = NULL, plot.loglog = FALSE)
  table.1[4,i+1] <- mean(q$fd)
  }; rm(i,q,w.s)
```

```
detach("package:fractaldim")
detach("package:abind")
table.1
    stats        ts.1        ts.2        ts.3         ts.4        ts.5
1       n 6000.000000 6000.000000 6000.000000 6000.0000000 6000.000000
2    mean   -1.466314    1.647883   -2.080831    1.3697476   -1.226921
3      sd    2.054721    1.487989    1.170512    0.8129809    2.658949
4 mean.fd    1.979112    1.976728    1.999593    1.9830176    1.978095
         ts.6        ts.7        ts.8
1 6000.000000 6000.000000 6000.000000
2   -2.065006    1.225880   -1.327785
3    1.016617    2.401352    2.265327
4    1.970747    1.966416    1.988077
```

Next, we create a graph; again, the graph is just for the purpose of having a graph to export to Excel. When doing this in R, the graph will not be displayed. Instead, the graph will be written as 'graph1.png' to the location specified as the working directory (from above). The graph file will only be written to that location when the line 'dev.off()' is processed. The graph is displayed after the code segment so that the reader will see what the graph looks like in R prior to seeing it in the finished Excel file.

```
jpeg('graph1.png')
  par(mfrow = c(4,1))
  plot(ts.df[,1],ts.df[,2], type = "l", col = "darkblue",  xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
  par(new = T)
  plot(ts.df[,1],ts.df[,3], type = "l", col = "blue",  xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
  plot(ts.df[,1],ts.df[,4], type = "l", col = "darkgreen",  xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
  par(new = T)
  plot(ts.df[,1],ts.df[,5], type = "l", col = "green",  xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
  plot(ts.df[,1],ts.df[,6], type = "l", col = "red", xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
  par(new = T)
  plot(ts.df[,1],ts.df[,7], type = "l", col = "brown",  xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
  plot(ts.df[,1],ts.df[,8], type = "l", col = "black",  xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
  par(new = T)
  plot(ts.df[,1],ts.df[,9], type = "l", col = "grey",  xlab = "Time",
    ylim = c(-5,5), ylab = "Y")
dev.off()
```



Next, we work on creating the Excel file by first creating the workbook, then the individual sheets - all within R. Three packages are required to accomplish these tasks; really only one package (xlsx) but, it has two dependent packages.

```
library(rJava)
library(xlsxjars)
library(xlsx)
```

First, we need to create the workbook, here simply named my.wb, by using the intuitively named create-Workbook function and supplying the Excel format we wish – here xls.

```
my.wb <- createWorkbook(type = "xls")
```

Next, we create two sheets; one for the table and graph, and one for the raw time series data. Again, the function is intuitively named: createSheet, and we supply the workbook in which to create each sheet and the sheet name we desire.

```
sheet.1 <- createSheet(my.wb, sheetName = "time.series.tables.and.graphs")
sheet.2 <- createSheet(my.wb, sheetName = "time.series.data")
```

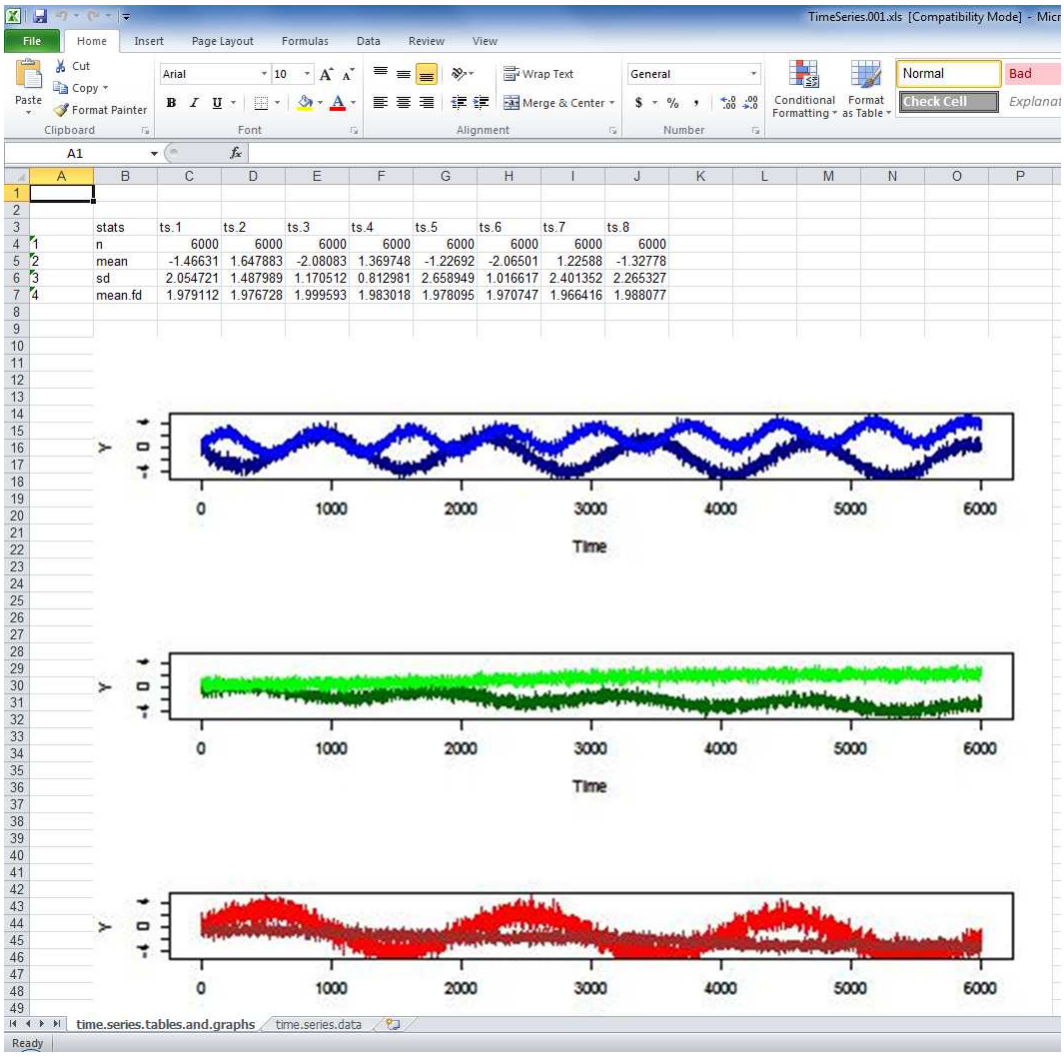
Next, we *add* the objects, such as the table, graph, and data, to the two sheets we have created in our workbook. When using the addDataFrame or addPicture functions, it may require some trial and error to place the data frame or picture in the sheet where it is desired. However, creating the file, checking placement, and if necessary, altering the start column and start row arguments to re-write the file is very easy to do. One other key point to keep in mind is the scale argument of the addPicture function. The scale argument can be used to adjust the size of the picture. Scale is set to 1.00 by default (if no scale is specified, no scaling factor is applied).

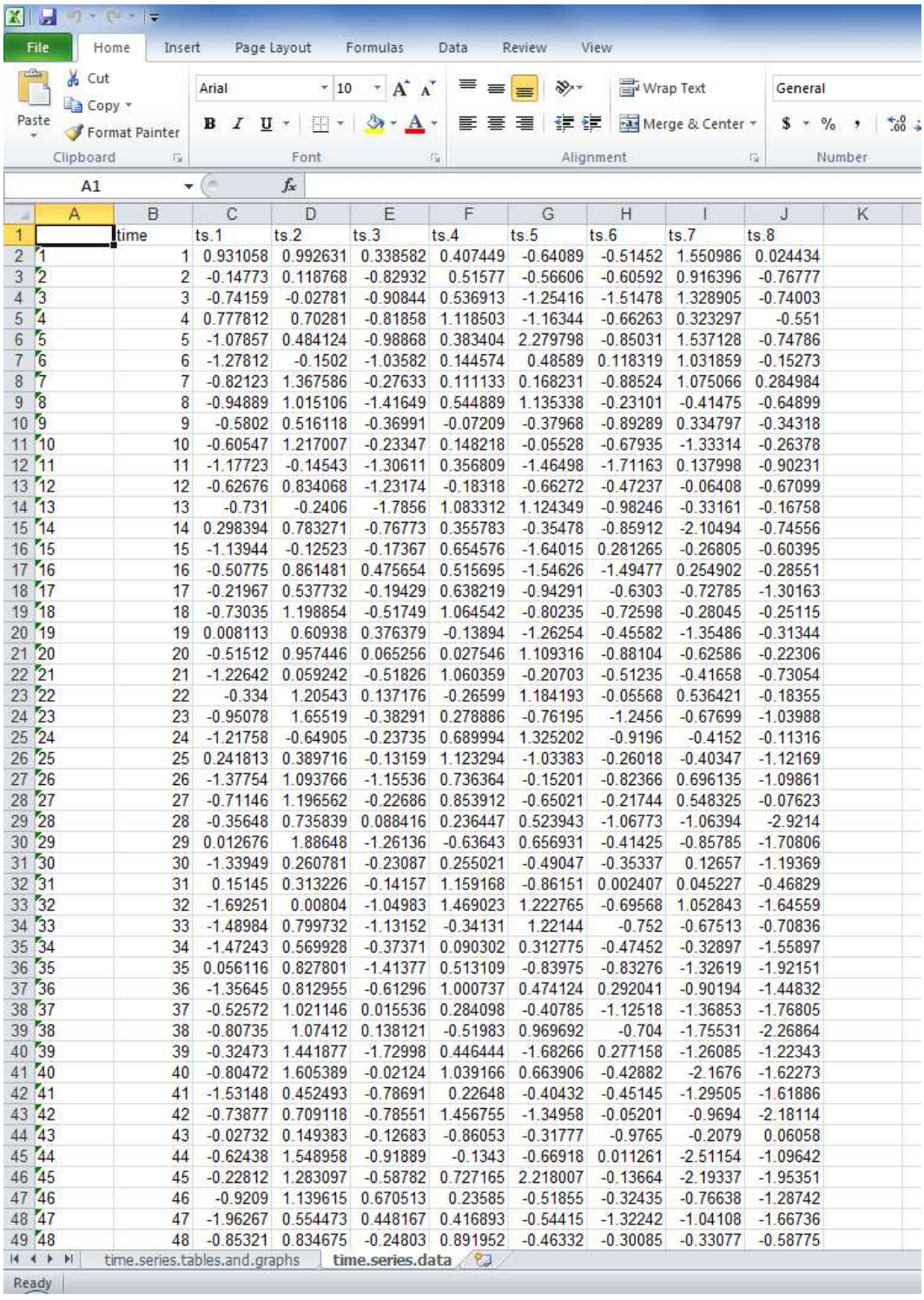
```
addDataFrame(table.1, sheet = sheet.1, startRow = 3, startColumn = 1)
my.file <- "C:/Users/jds0282/Desktop/graph1.png"
addPicture(file = my.file, sheet = sheet.1, scale = 2,
           startRow = 10, startColumn = 2)
addDataFrame(ts.df, sheet = sheet.2, startRow = 1, startColumn = 1)
```

Lastly, we must *save* the workbook. This is the step which actually creates the Excel file or workbook and sheets within it.

```
saveWorkbook(my.wb, "TimeSeries.001.xls")
```

Then, simply navigate to the working directory set at the beginning (here, the working directory is the desktop). Then open the Excel file and inspect the placement of the table(s) and graph(s). Screen captures of sheet 1 (table and graph) and then sheet 2 (raw data) are below.


TimeSeries.001.xls [Compatibility Mode] - Micr
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Cut
Copy
Format Painter
Paste
Clipboard
Arial
10
Font
Wrap Text
Merge & Center
Alignment
General
Number
Conditional Formatting
Format as Table
Normal
Bad
Check Cell
Explanat
A1
stats ts.1 ts.2 ts.3 ts.4 ts.5 ts.6 ts.7 ts.8
1 n 6000 6000 6000 6000 6000 6000 6000 6000
2 mean -1.46631 1.647883 -2.08083 1.369748 -1.22692 -2.06501 1.22588 -1.32778
3 sd 2.054721 1.487989 1.170512 0.812981 2.658949 1.016617 2.401352 2.265327
4 mean.fd 1.979112 1.976728 1.999593 1.983018 1.978095 1.970747 1.966416 1.988077
Y
Time
0
1000
2000
3000
4000
5000
6000
time.series.tables.and.graphs
time.series.data
Ready



	A	B	C	D	E	F	G	H	I	J	K
1		time	ts.1	ts.2	ts.3	ts.4	ts.5	ts.6	ts.7	ts.8	
2	1	1	0.931058	0.992631	0.338582	0.407449	-0.64089	-0.51452	1.550986	0.024434	
3	2	2	-0.14773	0.118768	-0.82932	0.51577	-0.56606	-0.60592	0.916396	-0.76777	
4	3	3	-0.74159	-0.02781	-0.90844	0.536913	-1.25416	-1.51478	1.328905	-0.74003	
5	4	4	0.777812	0.70281	-0.81858	1.118503	-1.16344	-0.66263	0.323297	-0.551	
6	5	5	-1.07857	0.484124	-0.98868	0.383404	2.279798	-0.85031	1.537128	-0.74786	
7	6	6	-1.27812	-0.1502	-1.03582	0.144574	0.48589	0.118319	1.031859	-0.15273	
8	7	7	-0.82123	1.367586	-0.27633	0.111133	0.168231	-0.88524	1.075066	0.284984	
9	8	8	-0.94889	1.015106	-1.41649	0.544889	1.135338	-0.23101	-0.41475	-0.64899	
10	9	9	-0.5802	0.516118	-0.36991	-0.07209	-0.37968	-0.89289	0.334797	-0.34318	
11	10	10	-0.60547	1.217007	-0.23347	0.148218	-0.05528	-0.67935	-1.33314	-0.26378	
12	11	11	-1.17723	-0.14543	-1.30611	0.356809	-1.46498	-1.71163	0.137998	-0.90231	
13	12	12	-0.62676	0.834068	-1.23174	-0.18318	-0.66272	-0.47237	-0.06408	-0.67099	
14	13	13	-0.731	-0.2406	-1.7856	1.083312	1.124349	-0.98246	-0.33161	-0.16758	
15	14	14	0.298394	0.783271	-0.76773	0.355783	-0.35478	-0.85912	-2.10494	-0.74556	
16	15	15	-1.13944	-0.12523	-0.17367	0.654576	-1.64015	0.281265	-0.26805	-0.60395	
17	16	16	-0.50775	0.861481	0.475654	0.515695	-1.54626	-1.49477	0.254902	-0.28551	
18	17	17	-0.21967	0.537732	-0.19429	0.638219	-0.94291	-0.6303	-0.72785	-1.30163	
19	18	18	-0.73035	1.198854	-0.51749	1.064542	-0.80235	-0.72598	-0.28045	-0.25115	
20	19	19	0.008113	0.60938	0.376379	-0.13894	-1.26254	-0.45582	-1.35486	-0.31344	
21	20	20	-0.51512	0.957446	0.065256	0.027546	1.109316	-0.88104	-0.62586	-0.22306	
22	21	21	-1.22642	0.059242	-0.51826	1.060359	-0.20703	-0.51235	-0.41658	-0.73054	
23	22	22	-0.334	1.20543	0.137176	-0.26599	1.184193	-0.05568	0.536421	-0.18355	
24	23	23	-0.95078	1.65519	-0.38291	0.278886	-0.76195	-1.2456	-0.67699	-1.03988	
25	24	24	-1.21758	-0.64905	-0.23735	0.689994	1.325202	-0.9196	-0.4152	-0.11316	
26	25	25	0.241813	0.389716	-0.13159	1.123294	-1.03383	-0.26018	-0.40347	-1.12169	
27	26	26	-1.37754	1.093766	-1.15536	0.736364	-0.15201	-0.82366	0.696135	-1.09861	
28	27	27	-0.71146	1.196562	-0.22686	0.853912	-0.65021	-0.21744	0.548325	-0.07623	
29	28	28	-0.35648	0.735839	0.088416	0.236447	0.523943	-1.06773	-1.06394	-2.9214	
30	29	29	0.012676	1.88648	-1.26136	-0.63643	0.656931	-0.41425	-0.85785	-1.70806	
31	30	30	-1.33949	0.260781	-0.23087	0.255021	-0.49047	-0.35337	0.12657	-1.19369	
32	31	31	0.15145	0.313226	-0.14157	1.159168	-0.86151	0.002407	0.045227	-0.46829	
33	32	32	-1.69251	0.00804	-1.04983	1.469023	1.222765	-0.69568	1.052843	-1.64559	
34	33	33	-1.48984	0.799732	-1.13152	-0.34131	1.22144	-0.752	-0.67513	-0.70836	
35	34	34	-1.47243	0.569928	-0.37371	0.090302	0.312775	-0.47452	-0.32897	-1.55897	
36	35	35	0.056116	0.827801	-1.41377	0.513109	-0.83975	-0.83276	-1.32619	-1.92151	
37	36	36	-1.35645	0.812955	-0.61296	1.000737	0.474124	0.292041	-0.90194	-1.44832	
38	37	37	-0.52572	1.021146	0.015536	0.284098	-0.40785	-1.12518	-1.36853	-1.76805	
39	38	38	-0.80735	1.07412	0.138121	-0.51983	0.969692	-0.704	-1.75531	-2.26864	
40	39	39	-0.32473	1.441877	-1.72998	0.446444	-1.68266	0.277158	-1.26085	-1.22343	
41	40	40	-0.80472	1.605389	-0.02124	1.039166	0.663906	-0.42882	-2.1676	-1.62273	
42	41	41	-1.53148	0.452493	-0.78691	0.22648	-0.40432	-0.45145	-1.29505	-1.61886	
43	42	42	-0.73877	0.709118	-0.78551	1.456755	-1.34958	-0.05201	-0.9694	-2.18114	
44	43	43	-0.02732	0.149383	-0.12683	-0.86053	-0.31777	-0.9765	-0.2079	0.06058	
45	44	44	-0.62438	1.548958	-0.91889	-0.1343	-0.66918	0.011261	-2.51154	-1.09642	
46	45	45	-0.22812	1.283097	-0.58782	0.727165	2.218007	-0.13664	-2.19337	-1.95351	
47	46	46	-0.9209	1.139615	0.670513	0.23585	-0.51855	-0.32435	-0.76638	-1.28742	
48	47	47	-1.96267	0.554473	0.448167	0.416893	-0.54415	-1.32242	-1.04108	-1.66736	
49	48	48	-0.85321	0.834675	-0.24803	0.891952	-0.46332	-0.30085	-0.33077	-0.58775	

Conclusions

Keep in mind, there are a variety of different ways of accomplishing what was accomplished in this article. All of the functions used in this article have optional arguments for more precise control over the objects and their placement in an Excel file being created. The example here was admittedly simple in order to illustrate the general use of functions which can be used to export objects (e.g., tables, graphs,

& data) to Excel from R. As stated last month, that is another benefit of using R, the flexibility it affords the analyst in deciding what to do and how to do it. For more information on what R can do, please visit the Research and Statistical Support Do-It-Yourself Introduction to R[3] course website. Lastly, for those interested in seeing how the example data was created in R; please take a look at the script[4] which was used. An Adobe.pdf version of this article can be found here[5].

Until next time; *Ground control to Major Tom...*

References & Resources

Burns, P. (2013). Spreadsheet Addiction. Available at:
http://www.burns-stat.com/documents/tutorials/spreadsheet-addiction/

Dragulescu, A. A. (2013). Package xlsx. Documentation available at:
http://cran.r-project.org/web/packages/xlsx/index.html

Gneiting, T., Sevcikova, H., & Percival D.B. (2010). *Estimators of Fractal Dimension: Assessing the Roughness of Time Series and Spatial Data*. Technical Report No. 577, Department of Statistics, University of Washington. Available at:
http://www.stat.washington.edu/research/reports/2010/tr577.pdf

Sevcikova, H., Gneiting, T., & Percival, D. (2013). Package fractaldim. Documentation available at:
http://cran.r-project.org/web/packages/fractaldim/index.html

This article was last updated on August 28, 2013.

This document was created using LaTeX

[3] http://www.unt.edu/rss/class/Jon/R_SC/
[4] http://www.unt.edu/rss/class/Jon/R_SC/Module3/MultiExcelDataCreation.R
[5] http://www.unt.edu/rss/rssmattersindex.htm