Paper DV-06-2015

Intuitive Demonstration of Statistics through Data Visualization of Pseudo-Randomly Generated Numbers in R and SAS

Jack Sawilowsky, Ph.D., Union Pacific Railroad, Omaha, NE

ABSTRACT

Statistics training courses are offered to employees in a corporate environment. Monte Carlo simulations are created as teaching devices for live demonstrations of statistical concepts. These results are then coded into data visualizations to aid in intuitive understanding by the audience. Coding is done in both SAS® and R, and a comparison is made to show how both platforms are capable of performing the required tasks. Both code and output plots are presented.

KEYWORDS: SAS, R, Statistics, Data visualization, Monte Carlo simulation, Pseudo-random numbers

AUTOBIOGRAPHY

Jack Sawilowsky received his PhD in Evaluation and Research, an applied statistics program, in May 2014 from Wayne State University in Detroit, MI. He currently works for Union Pacific Railroad as a network analyst in the Measures and Evaluation division. Before moving to Omaha, NE, he taught statistics at the graduate level, created his own consulting company, and was part of evaluation teams on federally funded grants including the National Science Foundation. Jack is currently interested in learning about data science and big data tools such as Apache Hadoop and cluster computing.

INTRODUCTION

Statistics classes are periodically made available to employee cohorts at Union Pacific. The audiences range across departments such as Finance, Human Resources, Safety, and Environmental Engineering. The employees come from various backgrounds, with some brushing up on their skill sets from university years ago, while others are new to the concepts. Visualizing data can be a useful technique for teaching statistics to these students. Rather than simply telling them how statistics works, code can be written to intuitively demonstrate the concepts in live demonstrations.

TECHNOLOGY

All work was done on an Intel i7-4790 CPU at 3.60GHZ, with 16GB RAM and a 64-bit Operating System (Windows 7 Professional). The software used was SAS Enterprise Guide Version 5.1 and R Version 3.2.0, with RSTUDIO Version 0.98.1091 used as a front-end on top of R.

DEMONSTRATION 1: PEARSON CORRELATION

The Pearson product-moment correlation coefficient is one of the fundamental techniques used in statistics. SAS® and R are both fully functional statistics platforms that can be used to create demonstrations of the properties and expectations one can have when using this methodology. Both languages are used in this study to show students how the correlation works in a visually intuitive way.

This experiment was inspired by a work colleague who asked if an obtained Pearson's r of 0.30 meant x and y are 30% in common. Thinking about the best way to answer resulted in the creation of this exercise.

Pseudo-random data are generated from the uniform distribution ranging 1 through 100. Specifically, the values are whole numbers for maximum accessibility. Picking integers from 1 through 100 is something adult learners from essentially any background should understand. Specifically, 50 of these numbers are drawn and assigned to variable "x." This is then repeated for "y," leaving n = 50

bivariate pairs of (x,y) values. These data are then correlated, with the Pearson's r value saved in variable "a." Then the x and y values are returned and the process is repeated for i = 100 iterations. As a result, there are now 100 Pearson's r values. The result is depicted in Figure 1.

DEMONSTRATION 1: CORRELATION R CODE

```
a <- NULL
for (i in 1:100) {
       x <- round(runif(50, 1, 100))
       y <- round(runif(50, 1, 100))
       a[i] <- cor(x,y)
}
plot(a, col = "blue", main = "Figure 1: Correlations",
xlab = "Iteration", ylab = "Pearson's r")
abline(h=0)
print(sort(a))        #Step 1. Stop here.

b <- a[a<.01 & a>-.01]
plot(b, ylim=c(-.01, .01), col = "blue",
       main = "Figure 2: Correlations near Zero", xlab = "Count",
       ylab = "Pearson's r", xaxp = c(x1 = 1, x2 = length(b), n = length(b)-1))
abline(h=0)           #Step 2. Zoomed in.
```

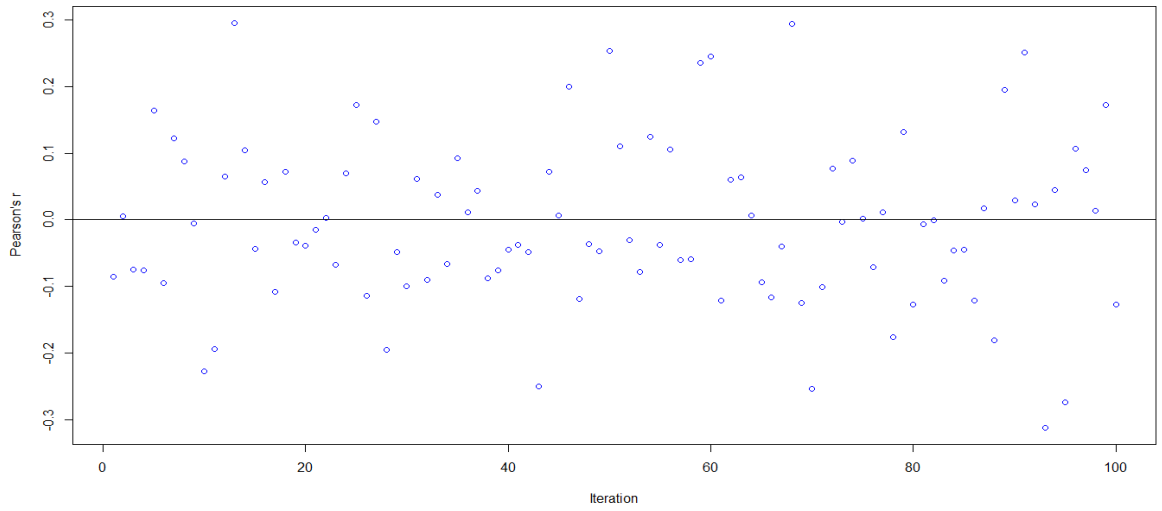

Figure 1. Pearson's r Values in R.

The X axis of the plot represents the repetition, ordered from 1 through 100. The Y axis shows the resulting Pearson's r value for that particular iteration. No seed is set in order to rerun the code multiple times and explore various random outcomes. Each time the code is executed, the dots move around. The plot has a horizontal line across at r = 0.

Ignoring direction (i.e., positive and negative), one thing that can be seen is how using random numbers, the magnitude of the relationship between x and y can be as high as roughly |0.3| in this case. In some cases when the experiment is run this gets as high as |0.4| or |0.5| in magnitude. The message to take away from this is to be suspicious with low correlation values, as even using totally random numbers with pure noise and no signal, low to moderate correlations might be found due to chance alone. At the very least, repetition of experiments is advised. Using this demonstration one can see how a correlation that low in magnitude could easily be, and is probably is, nothing meaningful.

Figure 2 is the R printout for all 100 Pearson's r values, sorted smallest to largest. Note how the code allows for both the data to be visualized and printed out so one can cross-reference. Once again, it can be seen that r = |0.3| is occurring at both the negative and positive sides. Specifically, by chance, the strongest negative correlation in this dataset was r = -0.344, while the strongest positive was 0.317.

```
> print(sort(a))        #Stop here.
  [1] -0.3127460860 -0.2747332436 -0.2535667798 -0.2503127903 -0.2274227435 -0.1950278318
  [7] -0.1947354055 -0.1809150617 -0.1767746675 -0.1275047639 -0.1268195839 -0.1249484976
 [13] -0.1219193990 -0.1219126747 -0.1184832274 -0.1171978146 -0.1142167175 -0.1087050887
 [19] -0.1012988295 -0.1001425033 -0.0949588083 -0.0944916705 -0.0909943321 -0.0905712013
 [25] -0.0883958443 -0.0853887943 -0.0781134425 -0.0764834494 -0.0760457242 -0.0747606928
 [31] -0.0717911935 -0.0677001089 -0.0661544880 -0.0609940813 -0.0595411893 -0.0487170267
 [37] -0.0480709463 -0.0474237367 -0.0459575077 -0.0448803406 -0.0444512739 -0.0438068096
 [43] -0.0406899537 -0.0392699294 -0.0380817920 -0.0380440808 -0.0369787475 -0.0344558496
 [49] -0.0309747691 -0.0146241423 -0.0070509269 -0.0054924034 -0.0034656504 -0.0005341699
 [55]  0.0014710544  0.0026463138  0.0054804505  0.0067869953  0.0068642516  0.0109048091
 [61]  0.0113026946  0.0140450406  0.0173574299  0.0231982049  0.0292957041  0.0369459292
 [67]  0.0430196029  0.0441237557  0.0565699868  0.0596822656  0.0614664437  0.0642647157
 [73]  0.0652648808  0.0699601208  0.0715619494  0.0724074530  0.0748541485  0.0766131932
 [79]  0.0871615422  0.0885796033  0.0919455391  0.1042895847  0.1057966632  0.1071174486
 [85]  0.1107276838  0.1222543755  0.1252186963  0.1316115673  0.1474927463  0.1636371589
 [91]  0.1719458158  0.1723692379  0.1956964954  0.2001366516  0.2353301014  0.2447699945
 [97]  0.2513890223  0.2538951049  0.2940873077  0.2959651812
>
```

Figure 2. Sorted R Console Printout of Pearson's r Values.

Another relevant insight is that just because one looks at two completely random variables, does not mean it can be expected for the variables to be perfectly uncorrelated. Indeed, most dots fall nowhere near the line across at zero, despite there being 100 of them. So not only can random data be stronger than one might suspect on the high side, they also might be stronger on the low side. The last five lines of code allow the instructor to zoom in to show the class how few very low Pearson's r values there are. Specifically, the "b" value has been created to visualize all correlations smaller than |0.01|.

Figure 3 shows the results with this filter included. In this case, only 9 out of 100 correlations are even close to zero, with none of them exactly on the zero line. This helps students understand not to expect even random data to be perfectly uncorrelated at r = 0.000 like they might expect. Instead, they are encouraged to adopt the general heuristic that only at roughly r = 0.5 or 0.6 should a correlation start to be considered meaningful. Anything less can easily occur due to chance alone.

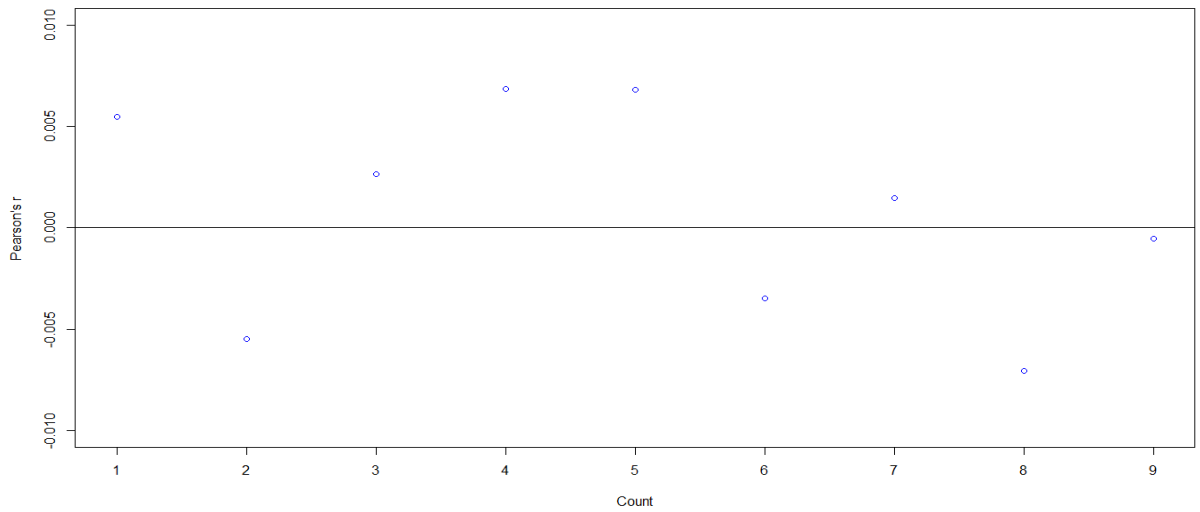

Figure 3. Pearson's r Values <|0.01| in R.

DEMONSTRATION 1: CORRELATION SAS® CODE

SAS® was used to replicate the existing R code. The software is similarly capable of creating correlation data from scratch. Once again, n = 50 bivariate (x,y) pairs of data were drawn and correlated for a total of i = 100 iterations. The code can be seen below, followed by the visual output in Figure 4.

```
DATA correlation;
DO i = 1 to 100;
       DO j = 1 to 50;
              x = ROUND((RAND('UNIFORM')*100), 1);
              y = ROUND((RAND('UNIFORM')*100), 1);
              output;
       END;
END;
PROC CORR DATA = correlation OUTP = r
       NOPRINT;
       VAR x y;
       BY i;

PROC SQL;
       CREATE TABLE r2 AS
       SELECT       t1.i,
              t1.x
               FROM WORK.r t1
               WHERE t1._NAME_ = 'y';
QUIT;
PROC SGPLOT data = r2;
       SCATTER x = i y = x;
       XAXIS  LABEL="Iteration";
       YAXIS  LABEL="Pearson's R";
REFLINE 0;
RUN;
```

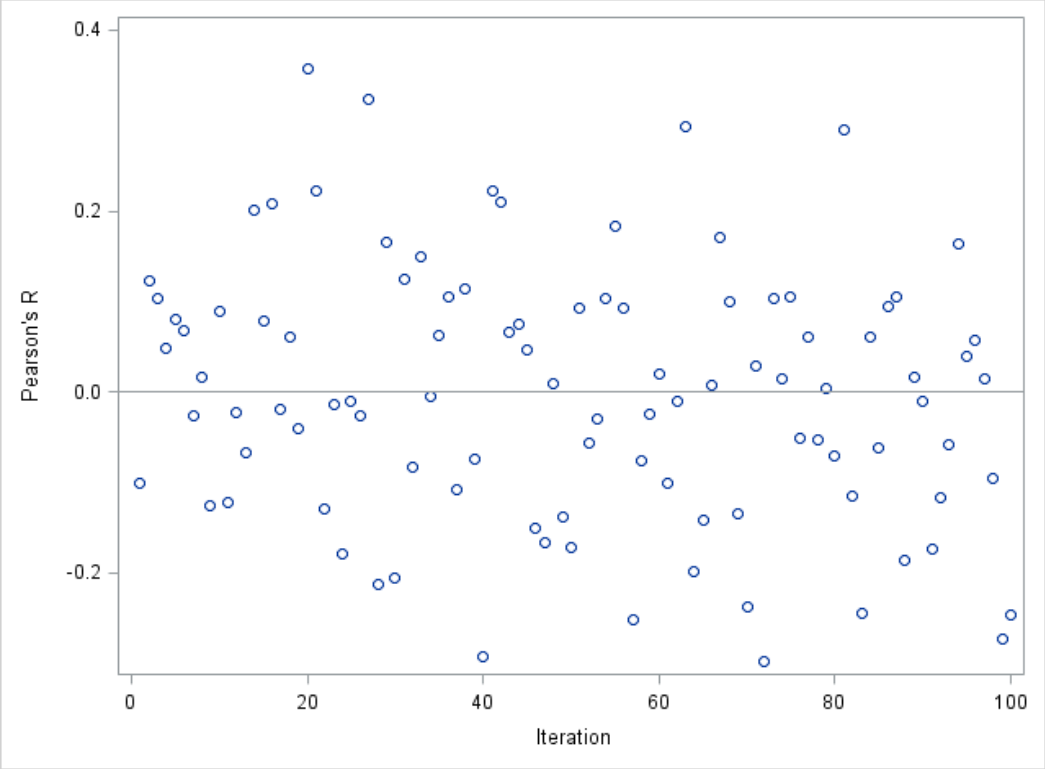

Figure 4. Pearson's r Values in SAS®.

The code was then run to filter only the values for which r < |0.01|. The results are visualized below in Figure 5.

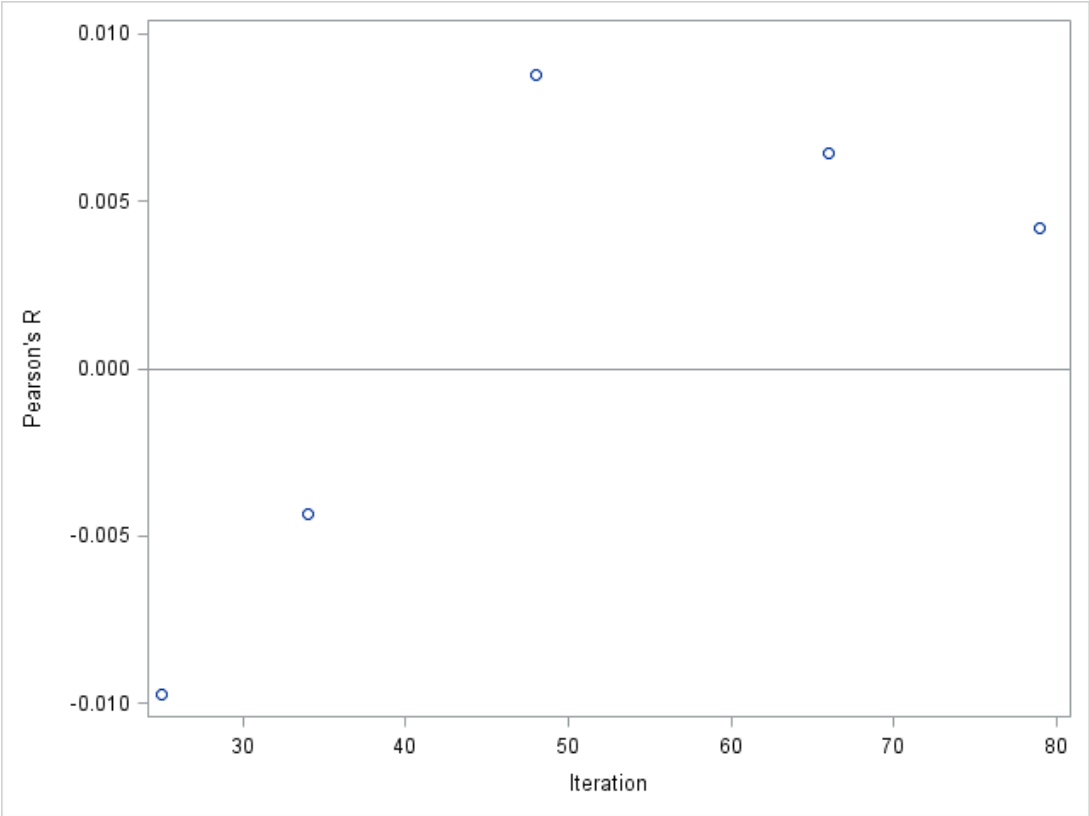

Figure 5. Pearson's r Values <|0.01| in SAS®.

DEMONSTRATION 2:

One of the important concepts to understand in statistics is Type I error, or false positives. Nominal alpha is set in order to have an objective goalpost. However, if set to alpha = 0.05, then in 5% or 1 in 20 of hypothesis tests the null will be rejected (on average). This is true due to probability and occurs regardless of using real, sensible data or purely pseudo-random data. Therefore, a Monte Carlo simulation was coded to demonstrate how type I errors can appear. The lessons learned are to replicate scientific studies, as well as use ANOVA combined with post hoc analyses and Hotelling's t^2 whenever appropriate.

The X axis in Figure 6 shows the p-values sorted lowest to highest. The Y axis shows what the p-values are. In this particular case, exactly 5 out of 100 p-values are lower than the cut-off point, which can vary. No seed is set so the experiment can be repeated to show a variety of chance outcomes. Note the false positive rate can vary each experiment, but will be 5% on average.

DEMONSTRATION 2: P-VALUES R CODE

```
a <- NULL
b <- 0
c <- NULL
for (i in 1:1000) {
   x <- rnorm(50)
   y <- rnorm(50)
   a[i] <- t.test(x, y, var.equal = TRUE)$p.value
   if(a[i] <= 0.05) b <- b+1
}
c <- b/i*100
plot(sort(a), col = "blue")
abline(h=0.05)
print(sort(a))
paste(b,"out of",i,"which is",c,"percent",sep = " ")
```

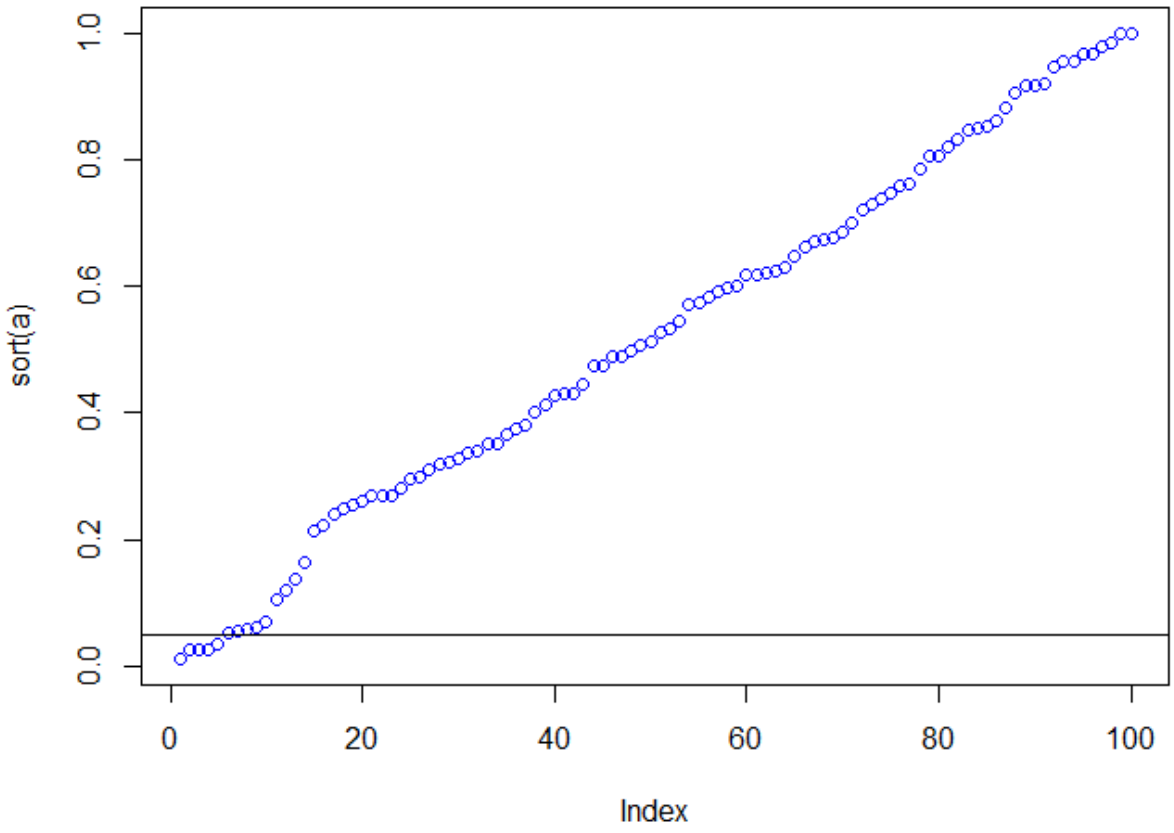

Figure 6. Sorted p-values, i = 100 reps. Lower is Stronger; Below Line is Statistical Significance.

The last line of code instructs the console to output how many p-values were statistically significant and at what percent. In this case, in Figure 7 shows an exact 5%.

```
> paste(b,"out of",i,"which is",c,"percent", sep = " ")
[1] "5 out of 100 which is 5 percent"
```

Figure 7. R Console Output showing Type I Error Rate. i = 100 reps

One can also change the number of iterations for great effect. Figure 7 is with i = 20 repetitions, as 5% is 1 in 20, making it the lowest common denominator.

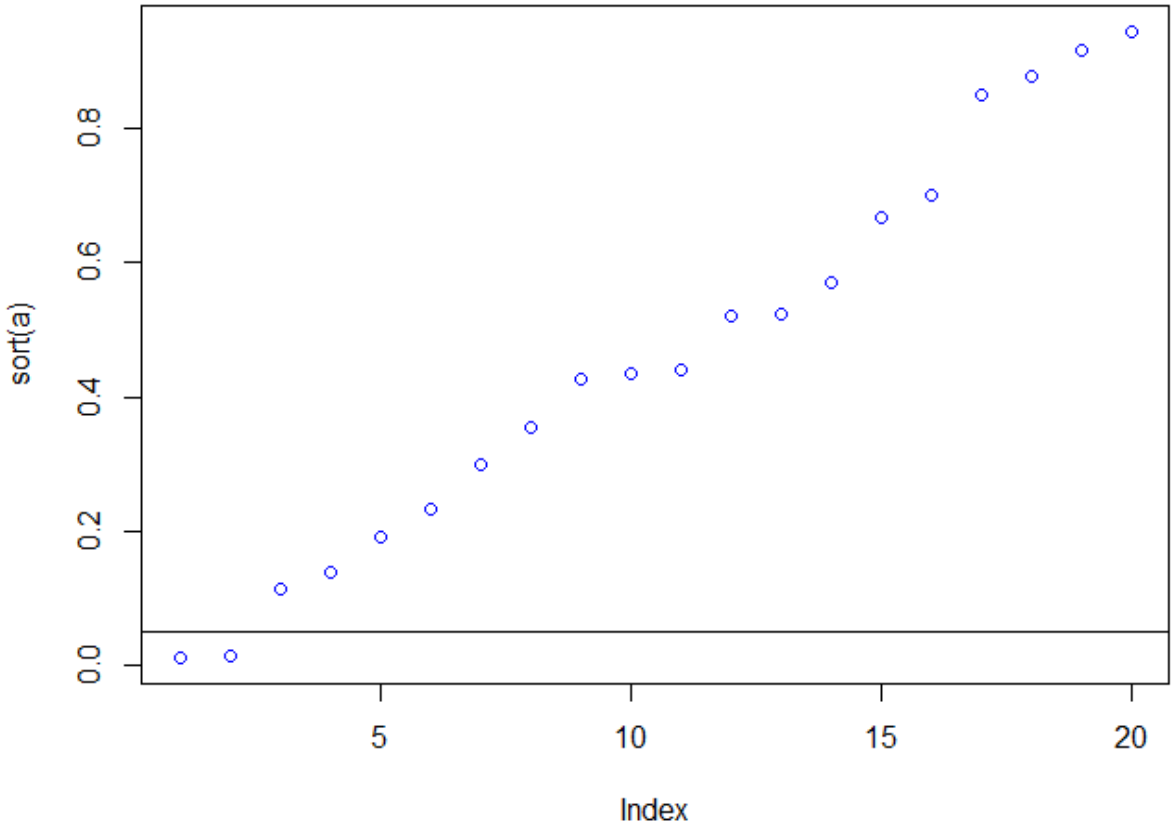

Figure 8. Sorted p-values, i = 20 reps. Lower is Stronger; Below Line is Statistical Significance.

The printout automatically changes to Figure 9.

```
> paste(b,"out of",i,"which is",c,"percent", sep = " ")
[1] "2 out of 20 which is 10 percent"
```

Figure 9. R Console Output showing Type I Error Rate. i = 20 reps

In this case the result is slightly off of 5% at 2 out of 20. This example has been cherry picked to demonstrate its possibility. Finally, one can look at the Type I error rate when conducting i = 1000 tests, as can be seen in Figure 10.

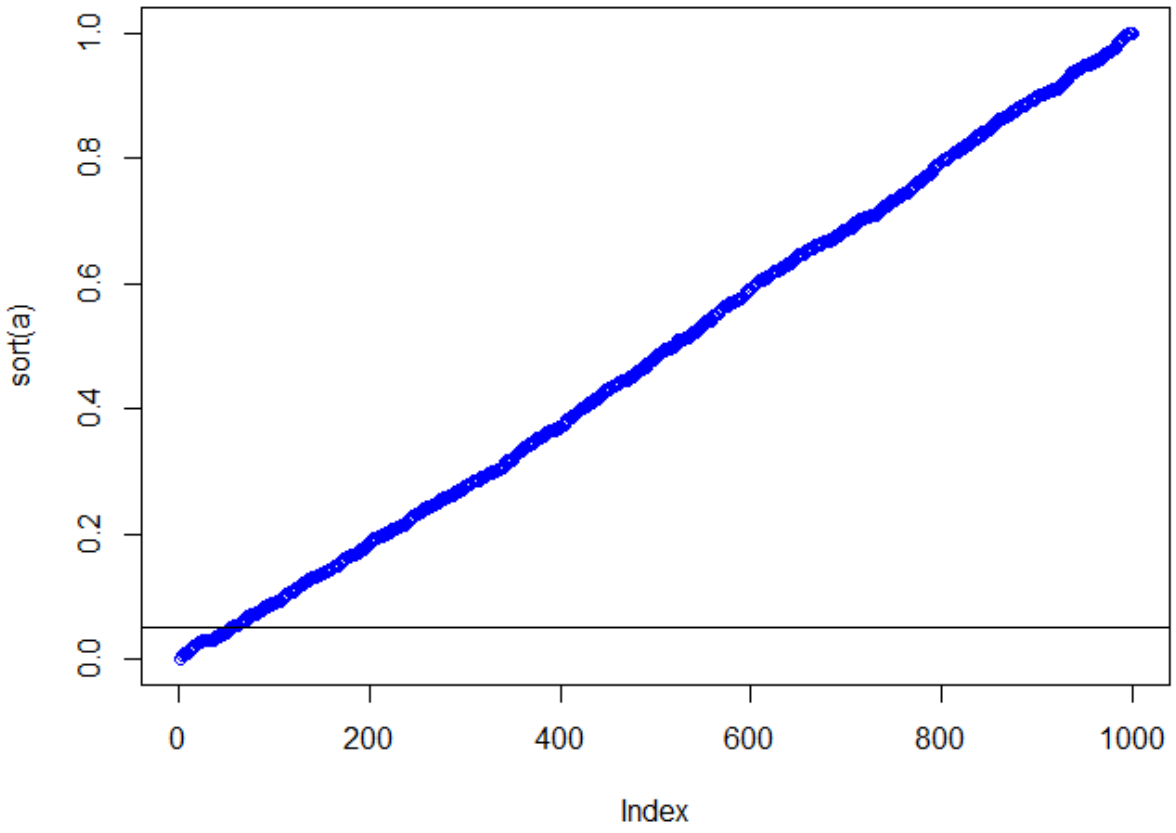

Figure 10. Sorted p-values, i = 1000 reps. Lower is Stronger; Below Line is Statistical Significance.

The console output corresponding to Figure 10 can be seen in Figure 11.

```
> paste(b,"out of",i,"which is",c,"percent", sep = " ")
[1] "54 out of 1000 which is 5.4 percent"
```

Figure 11. R Console Output showing Type I Error Rate. i = 1000 reps

Once again, it is not a perfect 5%, but if one runs the code repeatedly it will generally be slightly higher or lower than 5%, sometimes 5% on the dot, distributed as a bell curve due to central limit theorem.

DEMONSTRATION 2: P-VALUES SAS® CODE

```
DATA t;
DO i = 1 to 100;
   DO j = 1 to 100;
      x = RAND('NORMAL');
      if j <= 50 then group = "a";
      if j > 50 then group = "b";
      output;
   END;
END;
PROC TTEST DATA = t;
   CLASS group;
   VAR x;
   BY i;
ODS output TTESTS = p;
PROC SQL;
   ALTER table p
      ADD counter num;
   CREATE TABLE p2 AS
   SELECT t1.i,
         t1.Probt,
         t1.counter
      FROM WORK.P t1
      WHERE t1.Variances = 'Equal'
      ORDER BY t1.Probt;
QUIT;
DATA p2;
   SET p2;
      id = _N_;
   IF Probt <= .05 then counter = 1;
      ELSE counter = 0;
RUN;
PROC SGPLOT data = p2;
SCATTER x = id y = Probt;
XAXIS  LABEL="Sorted Iteration";
YAXIS  LABEL="P-Value";
REFLINE 0.05;
RUN;
PROC PRINT DATA = p2;
   SUM counter;
RUN;
```

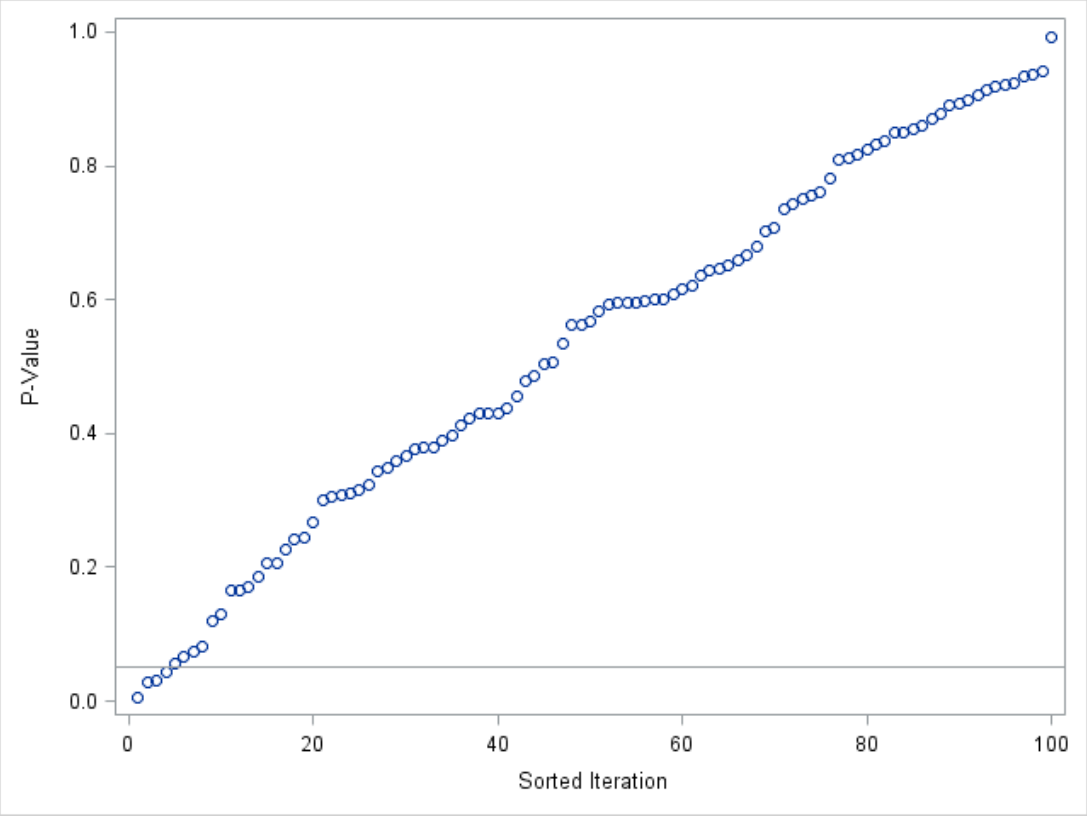

Figure 12. Sorted p-values, i = 100 reps. Lower is Stronger; Below Line is Statistical Significance.

The graphical printout in Figure 12 shows 5 false positives out of 100 iterations. Therefore, the R demonstration has again been successfully replicated in SAS®.

CONCLUSION

Ultimately, the purpose of these exercises was to visually demonstrate how statistics works under the hood. Data were pseudo-randomly drawn from theoretical distributions in order to gain real world insights into how a researcher should use the techniques. Specifically, the correlation and t-test were used to intuitively demonstrate the results one can expect when examining data trends and conducting hypothesis tests.

CONTACT INFORMATION

Name: Jack Sawilowsky, Ph.D.
Enterprise: Union Pacific Railroad
Address: 1400 Douglas STOP 1120
City, State ZIP: Omaha, NE 68102
Work Phone: 402-544-5295
E-mail: jsitm585@gmail.com
Web: www.linkedin.com/in/jacksawilowsky

Trademark Citation