Simple Data Simulations in R (of course).

As published in Benchmarks Research Matters, April 2018

https://benchmarks.it.unt.edu/researchmatters

Jon Starkweather, PhD

Jon Starkweather, PhD
jonathan.starkweather@unt.edu
Consultant
Research and Statistical Support

https://www.unt.edu

https://it.unt.edu/research

R&SS hosts a number of “Short Courses”.
A list of them is available at:
https://it.unt.edu/researchshortcourses

Those interested in learning more about R, or how to use it, can find information here:
http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/

Simple Data Simulations in R (of course).

Simulating data can be a very useful learning tool. It can help each of us better understand the 'real world' data we collect by allowing us to mimic the structure of data we hope to collect or data we have collected. It can help us better understand the analyses and models we wish to use, as well as the algorithms upon which those rely. Simulation can also be extremely useful in a classroom setting; whether to help students understand the things mentioned above, or simply to provide quick data examples for exercises (e.g. homework).

Univariate Data Simulation

There are a large number of theoretical distributions which can be simulated using basic R functions (i.e. functions available without additional packages). For example, the function 'rnorm' creates random deviates from a Normal distribution, given a mean and standard deviation.

```
x <- rnorm(n = 10000, mean = 10, sd = 1.5)
hist(x)
mean(x)
sd(x)
```

There are associated functions for doing the same thing with other theoretical distributions; such as: 'rbeta' (Beta distribution), 'rexp' (Exponential distribution), 'rchisq' (non-central Chi-Squared distribution), 'rpois' (Poisson distribution), 'rf' (F distribution), 'rt' (Student's t distribution), etc. Each of the distribution functions create random realizations of a theoretical distribution, given the parameters for each distribution (e.g. mean & standard deviation for Normal distributions). Each distribution also has 3 other functions which provide density, distribution function, and quantile. So, for the Normal distribution(s) there are four standard functions: 'rnorm', 'dnorm' (density), 'pnorm' (distribution function), and 'qnorm' (for quantiles).

Common Regression-style simulation.

The section heading printed above was chosen because we wanted to demonstrate a couple of commonly used single-outcome prediction / explanation models. First, Ordinary Least Squares (OLS) linear regression. In the example below, we create *textbook perfect* realizations of such models. These examples are "perfect" because the assumptions of the models are met: only linear relationships, homoscedasticity of variances, normal distribution of residuals, no multicollinearity, no measurement error, and only the correct variables are included in the model (i.e. no model misspecifications [errors of omission or errors of inclusion]). Starting simple, with a bivariate example (i.e. simple correlation), we demonstrate how there are multiple ways to simulate this type of data. It is important to note that because the functions used below are drawing *random* deviates from theoretical distributions (given user supplied the parameters); the statistics returned will not be *exactly* the same as those specified during data generation. However, they will be very close to those specified.

```
x <- rnorm(100, 10, 1.5)
Zx <- scale(x)
Zy <- .8*Zx + rnorm(100, 0, sqrt(1 - (.8^2)))
```

Check to see if we can recover the parameter(s) used to create the data...

```
cor(Zx, Zy)
summary(lm(Zy ~ 0 + Zx))
```

To see the more familiar non-standardized version, simply use the original ‘x’ variable as the predictor and apply the same ‘scaling’ or parameters (i.e. mean & standard deviation) to the outcome, ‘y’.

```
y <- (1.5*Zy) + 10
cor(x, y)
summary(lm(y ~ x))
```

Of course, it is then simple to add complexity, such as a second predictor and an interaction term.

```
x1 <- rnorm(100, 10, 1.5)
Zx1 <- scale(x1)
x2 <- rnorm(100, 10, 1.5)
Zx2 <- scale(x2)
x3 <- x1*x2
Zx3 <- scale(x3)
Zy <- (.7*Zx1) + (.4*Zx2) + (.2*Zx3) + rnorm(100, 0, sqrt(1 -
      (.49 + .16 + .04)))
y <- (1.5*Zy) + 10

df.1 <- data.frame(x1, x2, x3, y)
rm(x, Zx, Zx1, x1, Zx2, x2, Zx3, x3, Zy, y)
```

Checking results...

```
summary(lm(y ~ x1 + x2 + x3, data = df.1))

summary(lm(scale(y) ~ 0 + scale(x1) + scale(x2) + scale(x1)*scale(x2),
           data = df.1))
```

The second common single outcome model is the binary logistic regression model, which will be demonstrated below using the ‘arm’ package (Gelman, et al., 2016); which contains the ‘invlogit’ function.

```
library(arm)
```

Here we are dealing with a binary outcome variable and so, we use the ‘rbinom’ function.

```
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- rnorm(100)
b0 <- 1
b1 <- 1.5
b2 <- 2
```

```
b3 <- 0.5
y <- rbinom(100, 1, invlogit(b0 + b1*x1 + b2*x2 + b3*x3))
df.2 <- data.frame(x1, x2, x3, y)
rm(x1, x2, x3, b0, b1, b2, b3, y)
summary(df.2)
df.3 <- df.2
df.3[,4] <- as.factor(df.2[,4])
summary(df.3)
```

Checking results with both ‘numeric’ and ‘factor’ (i.e. categorical) versions of ‘x2’ & ‘y’.

```
summary(glm(y ~ x1 + x2 + x3, data = df.2, family = "binomial"))
summary(glm(y ~ x1 + x2 + x3, data = df.3, family = "binomial"))
```

Detaching the package ‘arm’ and its dependent packages.

```
detach("package:arm")
detach("package:lme4")
detach("package:Matrix")
detach("package:MASS")
```

Simulating multivariate data structures.

We can use the ‘mvrnorm’ function from the ‘MASS’ package (Ripley, et al., 2017; Venables & Ripley, 2002) to create multivariate normal deviates; given means and correlations among the variables. This function can be useful for a variety of data structures, such as simulating multicollinear predictor variables in a regression-style model, simulating components, simulating factors (there are better ways of doing this, covered further below), simulating canonical correlations, etc. To demonstrate the function, we create four multivariate normal variables with specified correlations (`Sigma = sig`), specified means (`mu = 0` for all four), and use ‘`empirical = TRUE`’ to replicate the exact correlations among the simulated variables.

```
sig <- matrix(c(1.0, 0.8, 0.5, 0.2,
                0.8, 1.0, 0.5, 0.5,
                0.5, 0.5, 1.0, 0.5,
                0.2, 0.5, 0.5, 1.0), nrow = 4)
library(MASS)
df.4 <- data.frame(mvrnorm(n = 1000, mu = rep(0, 4), Sigma = sig, empirical
detach("package:MASS")
summary(df.4)
ncol(df.4)
nrow(df.4)
```

Check the correlations of the matrix supplied by ‘sig’ versus the correlation matrix of the 1000 cases of the 4 variables.

```
round(sig, 2)
round(cor(df.4), 2)
```

We could then use the first two columns of the data frame as factor scores to recreate a data structure for factor analysis with two related factors (*r* = 0.8). Again, there are other ways to do this; mentioned further below.

```
v1 <- .8*df.4[,1] + rnorm(1000, 0, sqrt(1 - (.8^2)))
v2 <- .7*df.4[,1] + rnorm(1000, 0, sqrt(1 - (.7^2)))
v3 <- .6*df.4[,1] + rnorm(1000, 0, sqrt(1 - (.6^2)))
v4 <- .5*df.4[,1] + rnorm(1000, 0, sqrt(1 - (.5^2)))

v5 <- .5*df.4[,2] + rnorm(1000, 0, sqrt(1 - (.5^2)))
v6 <- .6*df.4[,2] + rnorm(1000, 0, sqrt(1 - (.6^2)))
v7 <- .7*df.4[,2] + rnorm(1000, 0, sqrt(1 - (.7^2)))
v8 <- .8*df.4[,2] + rnorm(1000, 0, sqrt(1 - (.8^2)))

df.5 <- data.frame(v1,v2,v3,v4,v5,v6,v7,v8)
rm(v1,v2,v3,v4,v5,v6,v7,v8)
```

Check the factor structure (i.e. loadings) with an oblique rotation strategy using the ‘GPArotation’ package (Bernaards & Jennrich, 2005; 2014).

```
library(GPArotation)
factanal(x = df.5, factors = 2, rotation = "oblimin")
```

Although the above example is nice for gaining some insight into multivariate normal data; it is recommended that if one is interested in simulating common specific types of multivariate data structures there are two packages which we (R&SS) find invaluable. The ‘psych’ package (Revelle, 2017a; 2017b) contains functions for simulating ANOVA / linear models, multilevel models, factor structures (hierarchical models, bi-factor models), simplex and circumplex structures; as well as others. The second package we (R&SS) find invaluable is the ‘lavaan’ package (Rosseel, et al., 2012; 2017) which has functions for simulating data for structural models (e.g., structural equation models [SEM]) and the model syntax (for ‘lavaan’ functions) is very intuitive and easy to learn. To see the full (and frequently updated) functionality of these two packages, please visit their respective web pages on CRAN and on the package authors’ web pages. As one example, below we create a structural model using the ‘lavaan’ package.

```
library(lavaan)
```

Specify the population’s structural model with coefficients.

```
sem.model <- '
  f1 =~ x1 + .8*x2 + .6*x3 + .4*x4
  f2 =~ x5 + .8*x6 + .6*x7 + .4*x8
  f3 =~ x9 + .8*x10 + .6*x11 + .4*x12
  f4 =~ x13 + .4*x14 + .6*x15 + .8*x16
  f5 =~ x17 + .6*x18 + .8*x19
  f4 ~ .6*f1
  f3 ~ .8*f2
  f5 ~ .3*f2 + .5*f3
  f1 ~~ 0*f2
  '
```

Simulate the data from the specified model.

```
df.6 <- simulateData(model = sem.model, sample.nobs = 1000)
summary(df.6)
```

To check the model, specify the SEM structural model...

```
str.model <- '
  f1 =~ x1 + x2 + x3 + x4
  f2 =~ x5 + x6 + x7 + x8
  f3 =~ x9 + x10 + x11 + x12
  f4 =~ x13 + x14 + x15 + x16
  f5 =~ x17 + x18 + x19
  f4 ~ f1
  f3 ~ f2
  f5 ~ f2 + f3
  f1 ~~ 0*f2
  f1 ~~ f3
  f1 ~~ f5
  f2 ~~ f4
  f3 ~~ f4
  f4 ~~ f5
  '
```

...then fit that model to the data and get a summary of the fit. The summary will show the loadings and path coefficients which can be compared to those we provided when generating the data.

```
sem.1 <- sem(model = str.model, data = df.6,
             std.lv = FALSE,
             parameterization = "default",
             std.ov = FALSE,
             ridge = 1e-05,
             estimator = "MLR", likelihood = "default", link = "default",
             information = "default", se = "robust.huber.white")
summary(sem.1, fit.measures = TRUE, standardize = TRUE)
```

A version of the R script used in this article can be found on the Research and Statistical Support (R&SS) *Do-It-Yourself Introduction to R* website[1] in the Module 9 section. Research and Statistical Support

[1] `http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/`

(R&SS)[2] and High-Performance Computing (HPC)[3] services are available to help facilitate quality research.

Until next time; *It is pitch black. You are likely to be eaten by a grue.*

If you found these materials useful, please participate in the R&SS Client Feedback Survey.
`https://unt.az1.qualtrics.com/SE/?SID=SV_diLLVU9iuJZN8C9`

References & Resources

Bernaards, C. A. & Jennrich, R. I. (2005). Gradient Projection Algorithms and Software for Arbitrary Rotation Criteria in Factor Analysis. *Educational and Psychological Measurement, 65*, 676 - 696.
`http://www.stat.ucla.edu/research/gpa`

Bernaards, C. A. & Jennrich, R. I. (2014). Package 'GPArotation' (many rotations available) documentation and user manual are available at CRAN:
`https://cran.r-project.org/web/packages/GPArotation/index.html`

Gelman, A., Su, Y., Yajima, M., Hill, J., Pittau, M. G., Kerman, J., Zheng, T., & Dorie, V. (2016). Package 'arm' documentation and manual are available at CRAN:
`https://cran.r-project.org/web/packages/arm/index.html`

Revelle, W. (2017a). *psych: Procedures for Personality and Psychological Research.* Northwestern University press: Evanston, IL.

Revelle, W. (2017b). Package 'psych' documentation and manual are available at CRAN:
`https://cran.r-project.org/web/packages/psych/index.html`

Ripley, B. D., et al. (2017). Package 'MASS' ('mvrnorm' function) documentation and manual are available at CRAN:
`https://cran.r-project.org/web/packages/MASS/index.html`

Rosseel, Y., et al. (2012). lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software, 48*(2), 1 - 36.
`http://www.jstatsoft.org/v48/i02/`

Rosseel, Y., et al. (2017). Package 'lavaan' documentation and user manual are available at CRAN:
`https://cran.r-project.org/web/packages/lavaan/index.html`

Venables, W. N. & Ripley, B. D. (2002). *Modern Applied Statistics with S* (4th ed.). Springer: New York. ISBN 0-387-95457-0.

[2]`https://it.unt.edu/research`
[3]`https://hpc.unt.edu/`

This article was last updated on February 21, 2018.

This document was created using LaTeX