Simulation as an important method for learning and a necessary step of good research practice.

As published in Benchmarks RSS Matters, April 2012

`http://web3.unt.edu/benchmarks/issues/2012/04/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

The programming scripts similar to those in this article can also be found at:
http://www.unt.edu/rss/class/Jon/R_SC

Simulation as an important method for learning and a necessary step of good research practice.

One of the things I've come to realize, working here at RSS[1], is the value of generating and playing with data. It sounds like a trivial or childish exercise but, it often leads to greater understanding of an analysis and the results one can expect from various oddities of real-life data. Creating fake data, or simulated data, forces one to have an understanding of the effects one is attempting to model as well as the underlying assumptions of a particular analysis. Regardless of whether one is doing a simple one-sample *t*-test or the most complex multivariate analyses; if one takes the time to create the model with simulated data, then one must understand the nature of the effects. All too often researchers are overconfident of their results without understanding the relationships underlying those results.

The rise of Samplonia

An excellent use of simulated data is in the realm of teaching. This idea first occurred to me after reading the first chapter of Bethlehem (2009). The first chapter of Bethleham's textbook describes the island nation of Samplonia and fortunately, Samplonia keeps a very detailed record of all its inhabitants (age, gender, etc.). Bethleham provides a table which shows the population of each province and district, as well as a few paragraphs description of each district's economy, agriculture, and citizen income. A map detailing Samplonia's provinces and districts, as well as geographical features and terrain appear at the end of the first chapter. Of course, Samplonia is fictitious and Bethleham created it (and its data) so that each example in his textbook would have relevance by using the Samplonia data set – thus giving the student/reader an enduring context for each chapter's examples and exercises. This is a wonderful idea and replicates the kind of familiarity researchers' gain when they collect their own data. The same principle could be easily applied to virtually any statistics course. Homework assignments and in class demonstrations could use one, albeit large, data set for an entire course. Furthermore, once the script was written for creating a population data set, an instructor would only need to run the script each semester to produce a new population, or sample, of data - thus making each semester's data unique but generated from the same model(s); with the same population parameters.

An example of simple model simulation

Suppose we are conducting some research and prior to collecting the data we would like to have some idea of how the data *should* look if our model is as good as we would like. For the purpose of this example, we are hypothesizing something as simple as an ordinary least squares (OLS) linear regression model with three interval/ratio predictor variables (x1, x2, & x3) and one interval/ratio outcome variable (y). The corresponding model looks like equation (1) below and assumes linear relationships among each of the predictors and the outcome, normally distributed error (ϵ), and no multicollinearity.

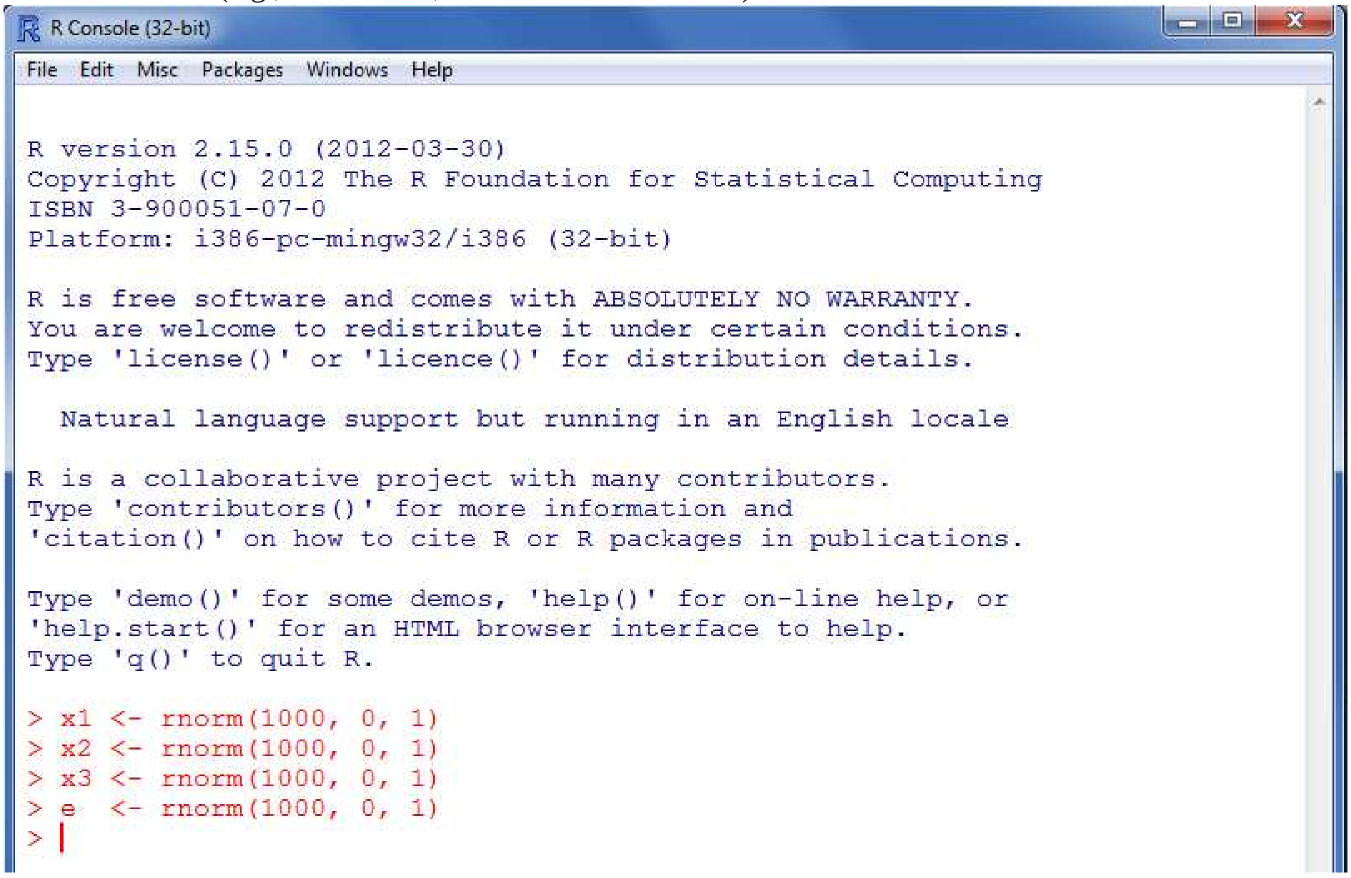
$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$$

Programming such a model into **R**[2] is rather easy with the 'rnorm' function - which creates a vector of random normal deviates for given mean and standard deviation. The defaults for the mean and standard deviation of those deviates are 0 and 1. In **R**, the following commands are used to create the data for the three predictors (x1, x2, & x3) and the error term (ϵ). Below, each variable (and the error term) has a

[1]http://www.unt.edu/rss/

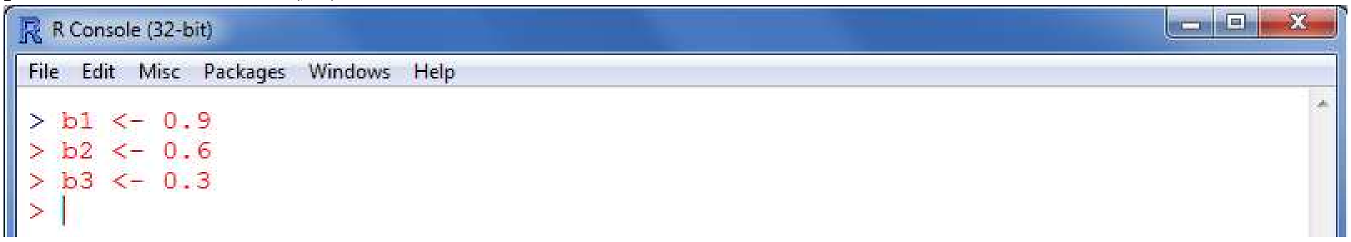
[2]http://cran.r-project.org/

mean of zero and a standard deviation of one; which, as stated, are the defaults of the 'rnorm' function. Also notice we are using a sample size of 1000. It is often useful to create variables (and models) using standardized (*Z*-score) data ($\mu = 0, \sigma = 1$) and then rescale the data into more meaningful measured scales afterward (e.g., mean of 20, standard deviation of 2).



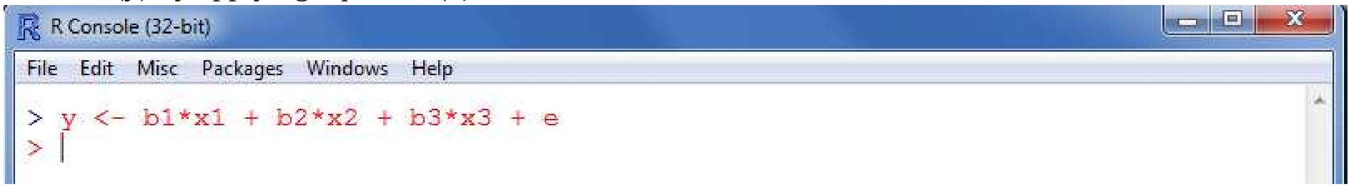
```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> x1 <- rnorm(1000, 0, 1)
> x2 <- rnorm(1000, 0, 1)
> x3 <- rnorm(1000, 0, 1)
> e  <- rnorm(1000, 0, 1)
> 
```

Next, we need to create the β coefficients – which should be approximated based on a thorough literature review and/or previous research with the variables under study. For the sake of the example, we use 0.9 for the first variable's coefficient (β_1), 0.6 for the second predictor's coefficient (β_2), and 0.3 for the third predictor's coefficient (β_3).



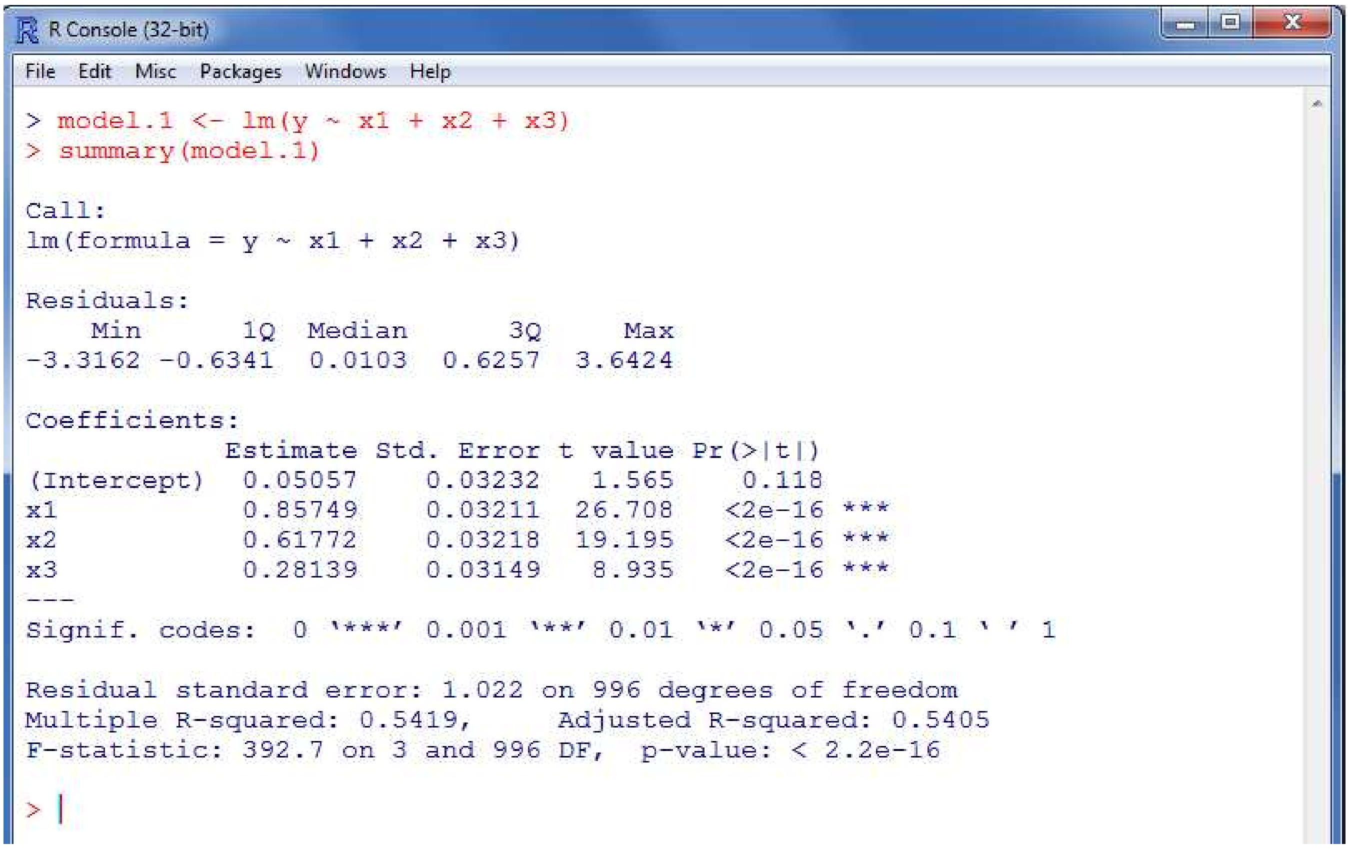
```
> b1 <- 0.9
> b2 <- 0.6
> b3 <- 0.3
> 
```

Using the coefficients and the random normal deviates (created above), we can then create the outcome variable (y) by applying equation (1).



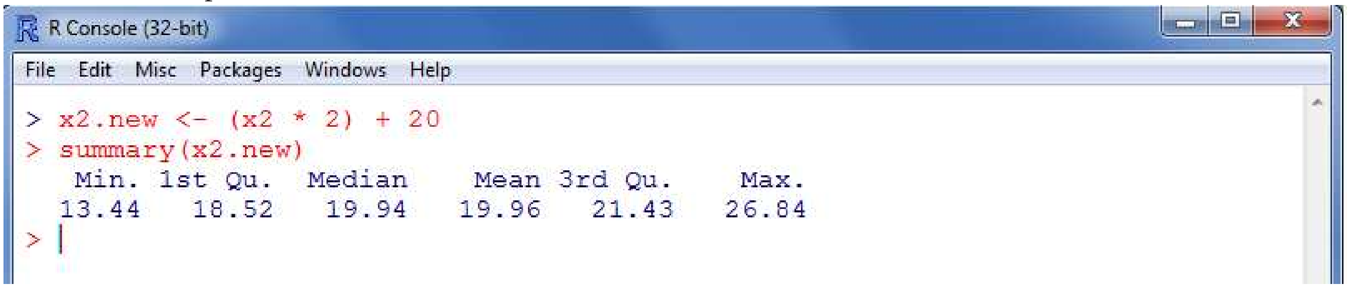
```
> y <- b1*x1 + b2*x2 + b3*x3 + e
> 
```

We can check to make sure the model accurately reflects what we put into it by running a simple linear model function (lm) and requesting a summary of the fitted model (model.1).



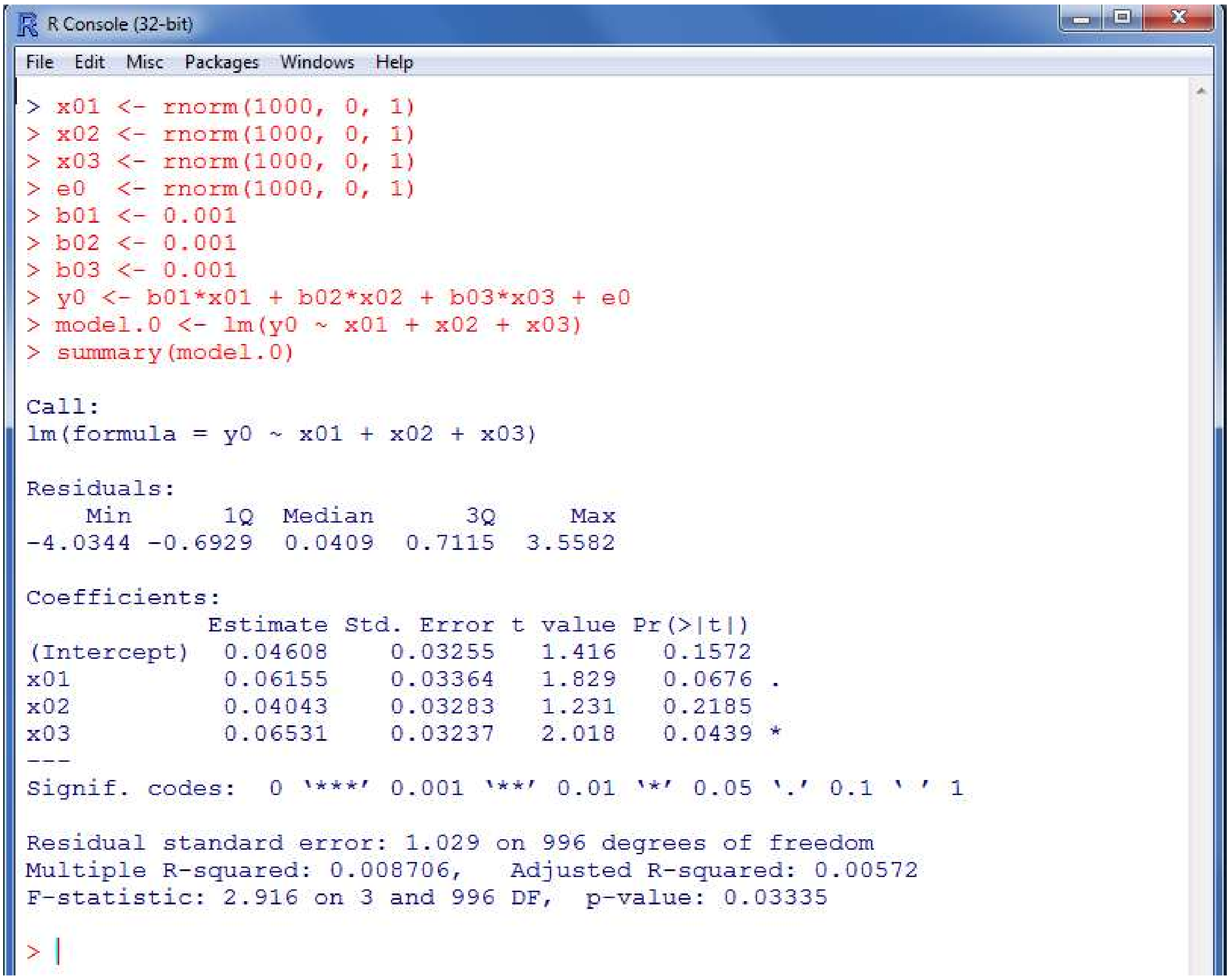
```
> model.1 <- lm(y ~ x1 + x2 + x3)
> summary(model.1)

Call:
lm(formula = y ~ x1 + x2 + x3)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3162 -0.6341  0.0103  0.6257  3.6424 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.05057    0.03232   1.565    0.118    
x1           0.85749    0.03211  26.708   <2e-16 ***
x2           0.61772    0.03218  19.195   <2e-16 ***
x3           0.28139    0.03149   8.935   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.022 on 996 degrees of freedom
Multiple R-squared: 0.5419,     Adjusted R-squared: 0.5405 
F-statistic: 392.7 on 3 and 996 DF,  p-value: < 2.2e-16

> 
```

We could rescale the variables (x1, x2, x3, & y) into any numeric scale we desired by simply multiplying the variable by the desired standard deviation and then adding the mean. As an example, we can transform the standardized scores of x2 to have a mean of 20 and a standard deviation of 2; creating the new variable 'x2.new' in the process.



```
> x2.new <- (x2 * 2) + 20
> summary(x2.new)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  13.44   18.52   19.94   19.96   21.43   26.84 
> 
```

It would be very simple to simulate a corresponding null model (no effect) for the alternative (or hypothesized) model specified above. To create a null version of this model, one would simply replace one or all coefficient values (0.9, 0.6, & 0.3) with values at or near zero (0.001, 0.001, & 0.001).



```
> x01 <- rnorm(1000, 0, 1)
> x02 <- rnorm(1000, 0, 1)
> x03 <- rnorm(1000, 0, 1)
> e0  <- rnorm(1000, 0, 1)
> b01 <- 0.001
> b02 <- 0.001
> b03 <- 0.001
> y0 <- b01*x01 + b02*x02 + b03*x03 + e0
> model.0 <- lm(y0 ~ x01 + x02 + x03)
> summary(model.0)

Call:
lm(formula = y0 ~ x01 + x02 + x03)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.0344 -0.6929  0.0409  0.7115  3.5582 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  0.04608    0.03255   1.416   0.1572  
x01          0.06155    0.03364   1.829   0.0676 .
x02          0.04043    0.03283   1.231   0.2185  
x03          0.06531    0.03237   2.018   0.0439 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.029 on 996 degrees of freedom
Multiple R-squared: 0.008706,   Adjusted R-squared: 0.00572 
F-statistic: 2.916 on 3 and 996 DF,  p-value: 0.03335 

> |
```

Notice in the image above, our null model (i.e. no effect) still indicates significance for one of the predictors and the *F* value. This is due to the sample size of 1000; which reinforces the idea that *p* values can be extremely misleading.

Conclusions

Virtually any model can be simulated in **R** using simple functions like the ones shown here. The 'rnorm' function is the one most frequently used, however **R** contains several other analogous functions for other types of distributions. The 'stats' package, which is included in the base install of **R**, contains the 'rexp' function produces random deviates for an exponential distribution, the 'rpois' produces for a Poisson distribution, the 'rbinom' function for a binomial distribution, the 'rbeta' function for a beta distribution, the 'runif' function for a uniform distribution, the 'rchisq' for a chi-square distribution, the 'rcauch' for a Cauchy distribution, the 'rgamma' for a gamma distribution, the 'rt' function for a *t* distribution, the 'rweibull' function for a Weibull distribution, and the 'rlogis' for a logistic distribution. Other packages contain even more functions related to simulating data, for example; the 'MASS' packages contains the 'mvrnorm' function for creating variables from a user specified multivariate normal distribution. Other packages related to simulation include; asd, betategarch, BoolNet, csampling, ecolMod, egarch, ezsim, Geneclust, GenOrd, glmdm, hypred, memisc, MigClim, nanop, NPsimex, pensim, phylosim, polyapost, portfolioSim, qrfactor, RandomFields, sde, simecol, simFrame, simPopulation, and TreeSim. Another very popular package is ‘Zelig’ developed by Imai, King, and Lau[3], which also has easy to use functions for simulation. Keep in mind, entire populations can be simulated in **R** and then sampled from using the ‘sample’ function (either randomly or nonrandomly). See the Introduction to **R** Short Course[4] for more complex examples of simulating data and models.

Until next time; *me and my Arrow are straighter than narrow...*

[3] `http://gking.harvard.edu/zelig`

[4] `http://www.unt.edu/rss/class/Jon/R_SC/`

References & Resources

Brown, P. J., & Vannucci, T. F. (2002). Bayes model averaging with selection of regressors. *Journal of the Royal Statistical Society (Series B: Statistical Methodology)*, *64*(3), 519 – 536.

Bethlehem, J. (2009). *Applied Survey Methods: A Statistical Perspective.* Hoboken, NJ: John Wiley & Sons.

Burton, A., Altman, D., Royston, P., & Holder, R. (2006). The design of simulation studies in medical statistics. *Statistics in Medicine*, *25*, 4279 - 4292.

Maldonado, G., & Greenland, S. (1997). The importance of critically interpreting simulation studies. *Epidemiology, 8*, 453 - 456.

Metcalfe, C., & Thompson, S. G. (2006). The importance of varying the event generation process in simulation studies of statistical methods for recurrent events. *Statistics in Medicine*, *25*, 165 - 179.

'R' Programming. Wikibook: `http://en.wikibooks.org/wiki/R_Programming`
Linear models: `http://en.wikibooks.org/wiki/R_Programming/Linear_Models`
Binomial models:
`http://en.wikibooks.org/wiki/R_Programming/Binomial_Models#Zelig`
Multinomial models:
`http://en.wikibooks.org/wiki/R_Programming/Multinomial_Models`

Weldon, K. L. (2005). Modern introductory statistics using simulation and data analysis. *International Statistical Institute*, 55th Session.
`http://www.stat.auckland.ac.nz/~iase/publications/13/Weldon.pdf`

This article was last updated on May 14, 2012.

This document was created using LaTeX