Simulation is the foundation of computational statistics and a fundamental organizing principle of the R language. For example, few complex tasks are more compactly expressed in any programming language than `rnorm(100)`. But while many simulation tasks are trivial in R, simulating adequate and convincing synthetic or "fake" data is a task whose cognitive demands increases quickly as complexity moves beyond independent random draws from named probability distributions. In this post, I would like to highlight a few of the many R packages that are useful for simulating data.

Before we begin, let's just list a few reasons why you may want to make fake data. Having some of these in mind should be useful for exploring the tools provided in the R packages listed below. Bear in mind that no one package is going to cover everything. But before simulating data on you own, it may be helpful to see where others thought it was worthwhile make the effort to write a package.

## Why simulate data?

- Simulation is part of exploratory data analysis. For example, when you may have data that comes from some kind of arrival process, it may be helpful to simulate Poisson arrivals to see if counts and arrival time distributions match with your data set. Having a generative model might really simplify your analysis, and knowing that you don't have one may be critical too.
- Carefully constructed fake data may be helpful in testing the limits of an algorithm or analysis. Everybody wants to publish an algorithm that really seems to explain some data set. Not too many people show you the edge conditions where their algorithm breaks down.
- Making fake data may be a good way to begin a project before you even have any data. There is no better way to expose you assumptions than to write them down in code.
- Fake data can be helpful when there are privacy concerns. Building a data set that reproduces some of the statistical properties of the real data while passing the eyeball test for being convincing may make all the difference in communicating your ideas.

## R Packages for Simulating Data

Here are a few of R package that ought to be helpful in nearly every project where you need to manufacture fake data.

bindata is an "oldie but goodie" from R Core members Friedrich Leisch. Andreas Weingessel, and Kurt Hornik that goes back to 1999 and still gets about twenty-five downloads a day. The package does one really nice trick: it provides multiple ways to create columns of binary random variables that have a pre-specified correlation structure.

Suppose you want to create a matrix whose columns are draws from binary random variables. You can begin by setting up a matrix, M, of "common probabilities". In the simple example below `M[1,1]` specifies the probability that the random variable `V1` will be a 1, while `M[2,2]` does the same for variable `V2`. The off diagonal elements specify the joint probabilities that `V1` and `V2` are both one.

```
M <-  matrix(c(.5, .2, .2, .5), nrow = 2, byrow = TRUE)
colnames(M) <- c("V1","V2")
M
```

```
##       V1  V2
## [1,] 0.5 0.2
## [2,] 0.2 0.5
```

```
res <- rmvbin(100,commonprob = M)
head(res)
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    1
## [3,]    0    1
## [4,]    0    0
```

```
## [5,]    1    1
## [6,]    1    1
```

For this example, it's easy to check the results by plotting a couple of histograms. The tricky part is deciding on the correlation structure. There are some limitations on what the common probabilities can be, but the authors provide a helper function: `check.commonprob(M)` so you can check ahead of time.

charlatan from Scott Chamberlain and the rOpenSci team allows you to make convincing fake addresses, person names, dates, times, coordinates, currencies, DOIs', jobs, phone numbers, DNA sequences and more.

Here is a small example of simulating gene sequences.

```
ch_gene_sequence(n = 10)
```

```
##  [1] "CTNNTTCTNCACNTCCNNATCGGNGACNCG" "CAGGACCAGNCCNNNTGAAAGTCANCNTCT"
##  [3] "CGNTTNAGTGATGCANGNCGCAAACCGTGC" "TGCNAACTATTCGGANNTCAGNCTTCTTNC"
##  [5] "NNCGGAGTTGNGTNAGNCNCGCGTTCGGNT" "NTCCCCTACACNNTTAANTTGNTATNTCGG"
##  [7] "GNGACNAAAGCANNNGGTAAGGTACTNNNA" "AGCGGNTNCNGGCGGCATNAGNCCNCNCTC"
##  [9] "AGAANNCNTGTGGACCGNNCNGNNAGCNTA" "TANCNCTCTTNNGNAANGNNTNNANACAGC"
```

And here is another small example showing how to simulate random longitude and latitude coordinates. (These very likely point to some places you've never been that might make good vacation destinations when you can actually go somewhere again.)

```
set.seed(1234)
locations <- as.data.frame(cbind(ch_lon(n=10),ch_lat(n=10)))
names(locations) <- c("lon", "lat")
leaflet(locations) %>% addProviderTiles("Stamen.Watercolor")  %>%
  addMarkers(~lon, ~lat)
```

fabricatr is part of the DeclareDesign suite of packages from Graeme Blair and his colleagues for "formally 'declaring' the analytically relevant features of a research design". As the package authors put it: "(the package) helps you imagine your data before you collect it", and provides functions for building hierarchical data structures and correlated data. To see some of what it can do, please have a look at this R Views post

by the package authors that works through examples of hierarchical data, longitudinal data and intra-class correlation. There is also a tutorial.

fakeR from Lily Zhang and Dustin Tingley helps to solve the problem of making fake data that matches your real data. It simulates data from a data set with various data types. It randomly samples character and factor data from contingency tables and numeric and ordered data from multivariate distributions that account for the between column correlations among the variables. There are also functions to simulate stationary time series.

In this example, we simulate data from the `USArrests` data set.

```
set.seed(1234)
state_names <- rownames(USArrests)
df <- tibble(state_names)
data <- USArrests
rownames(data) <- NULL
sim_data <- simulate_dataset(data)
```

```
## [1] "Numeric variables. No ordered factors..."
```

```
fake_arrests <- cbind(df, sim_data)
head(fake_arrests)
```

```
##   state_names Murder Assault UrbanPop  Rape
## 1     Alabama   3.96   179.8    77.89  7.28
## 2      Alaska  10.64   209.3    57.91 19.32
## 3     Arizona   3.01    87.8    54.51  7.86
## 4    Arkansas   5.48   175.9    79.31 22.17
## 5  California   4.81   104.5    55.52 31.75
## 6    Colorado   6.88   131.7    58.60 21.03
```

GenOrd by Bapiero and Ferrari implements gaussian copula based procedure for generating samples from discrete random variables with prescribed correlation matrix and marginal distributions.

The following is a slightly annotated version of Example 2 given on page nine of the package pdf. The problem is to draw samples from four different discrete random variables with different numbers of categories that have different specified uniform marginal distributions but conform to a specified correlation structure.

The example begins by specifying the marginal distributions and then running the function `corrcheck()` to get the upper and lower ranges for a feasible correlation matrix. Note that the random variables are set to have respectively 2, 3, 4, and 5 categories and that it is not necessary to specify the final 1 in each vector of cumulative marginal probabilities.

```
k <- 4 #number of random variables
marginal <- list(0.5, c(1/3,2/3), c(1/4,2/4,3/4), c(1/5,2/5,3/5,4/5))

corrcheck(marginal)
```

```
## [[1]]
## 4 x 4 Matrix of class "dsyMatrix"
##         [,1]    [,2]    [,3]    [,4]
## [1,]  1.0000 -0.8165 -0.8944 -0.8485
## [2,] -0.8165  1.0000 -0.9129 -0.9238
## [3,] -0.8944 -0.9129  1.0000 -0.9487
## [4,] -0.8485 -0.9238 -0.9487  1.0000
##
## [[2]]
## 4 x 4 Matrix of class "dsyMatrix"
##        [,1]   [,2]   [,3]   [,4]
## [1,] 1.0000 0.8165 0.8944 0.8485
## [2,] 0.8165 1.0000 0.9129 0.9238
```

```
## [3,] 0.8944 0.9129 1.0000 0.9487
## [4,] 0.8485 0.9238 0.9487 1.0000
```

Given the bounds, we select a feasible correlation matrix, Sigma, and generate n samples of the four random variables.

```
Sigma <- matrix(c(1,0.5,0.4,0.3,0.5,1,0.5,0.4,0.4,0.5,1,0.5,0.3,0.4,0.5,1),
                k, k, byrow=TRUE)
Sigma
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  1.0  0.5  0.4  0.3
## [2,]  0.5  1.0  0.5  0.4
## [3,]  0.4  0.5  1.0  0.5
## [4,]  0.3  0.4  0.5  1.0
```

```
set.seed(1)
n <- 1000 # sample size
m <- ordsample(n, marginal, Sigma)
head(m,10)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    2    3    4    2
##  [2,]    1    2    3    1
##  [3,]    2    3    2    5
##  [4,]    1    1    1    1
##  [5,]    2    1    2    2
##  [6,]    1    3    3    5
##  [7,]    1    3    1    1
##  [8,]    1    1    3    3
##  [9,]    1    2    3    2
## [10,]    2    3    3    2
```

Finally, check how well the simulated correlation structure agrees with the specified correlation matrix, and compare the empirical cumulative marginal probabilities with the specified probabilities. Things look pretty good.

```
cor(m) # compare it with Sigma
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,] 1.0000 0.5199 0.4352 0.3097
## [2,] 0.5199 1.0000 0.4907 0.3812
## [3,] 0.4352 0.4907 1.0000 0.4690
## [4,] 0.3097 0.3812 0.4690 1.0000
```

```
cumsum(table(m[,4]))/n # compare it with the fourth marginal specified above
```

```
##     1     2     3     4     5
## 0.189 0.392 0.592 0.793 1.000
```

MultiOrd by Amatya, Demirtas and Gao generates multivariate ordinal data given marginal distributions and a correlation matrix using the method proposed in Demirtas (2006).

PoisBinOrdNonNor by Demirtas et al. uses the power polynomial method described in Demirtas (2012) to generate count, binary, ordinal, and continuous random variables, with specified correlations and marginal properties.

SimMultiCorrData by Allison Cynthia Fialkowski is a *tour de force* of a package that was built using ideas pioneered in some of the packages listed above, and was part of her award winning PhD thesis. A motivating goal of the package is to enable users to make synthetic data useful in clinical applications. This includes generating data to match theoretical probability distributions, and also to mimic empirical data sets. The

package is notable not only for the sophisticated capabilities it provides, but also for the technical documentation and references to source materials.

`SimMultiCorrData` provides tools to simulate random variables from continuous, ordinal, categorical, Poisson and negative binomial distributions with precision and error estimates. It exhibits attention to numerical detail that is rare in the machine learning literature, and impressive even by R's standards. For example, there are several functions to evaluate and visualize the quality of the synthetic data.

The package provides three main simulation functions. `nonnormvar1()` is preferred for simulating a single, continuous random variable, `rcorrvar()` and `rcorrvar2()` are two different methods for generating correlated ordinal, continuous, Poisson, and negative binomial random variables that match a specified correlation structure.

The following example is a stripped-down, annotated version of the example provided in the package pdf for the `rcorrvar()` function. The example simulates one ordinal random variable (`k_cat`), two continuous random variables (`k_cont` one is logistic and the other Weibull), one Poisson random variable (`k_pois`), and one negative binomial random variable (`k_nb`). The portion replicated below concludes with generating the data. The version in the package pdf also works through the process of evaluating the fake data.

```
# Binary, Ordinal, Continuous, Poisson, and Negative Binomial Variables
options(scipen = 999) # decides when to switch between fixed or exponential
notation
seed <- 1234
n <- 10000 # number of random draws to simulate
Dist <- c("Logistic", "Weibull") # the 2 continuous rvs to be simulated
#Params list: first element location and scale for logistic rv
#             second element shape and scale for Weibull rv
Params <- list(c(0, 1), c(3, 5))
# Calculate theoretical parameters for Logistic rv including 4th, 5th and 6th
moments
Stcum1 <- calc_theory(Dist[1], Params[[1]])
# Calculate theoretical parameters for a Weibull rv
Stcum2 <- calc_theory(Dist[2], Params[[2]])
Stcum <- rbind(Stcum1, Stcum2)
rownames(Stcum) <- Dist
colnames(Stcum) <- c("mean", "sd", "skew", "skurtosis", "fifth", "sixth")
Stcum # matrix of parameters for continuous random variables
```

```
##           mean    sd   skew skurtosis  fifth sixth
## Logistic 0.000 1.814 0.0000    1.2000  0.000 6.857
## Weibull  4.465 1.623 0.1681   -0.2705 -0.105 0.595
```

```
# Six is a list of vectors of correction values to add to the sixth cumulants
# if no valid pdf constants are found
Six <- list(seq(1.7, 1.8, 0.01), seq(0.10, 0.25, 0.01))
marginal <- list(0.3) # cum prob for Weibull rv, Logistic assumed = 1
lam <- 0.5 # Constant for Poisson rv
size <- 2 # Size parameter for Negative Binomial RV
prob <- 0.75 # prob of success
Rey <- matrix(0.4, 5, 5)  # target correlation matrix: the order is important
diag(Rey) <- 1
# Make sure Rey is within upper and lower correlation limits
# and that a valid power method exists
valid <- valid_corr(k_cat = 1, k_cont = 2, k_pois = 1, k_nb = 1,
                    method = "Polynomial", means = Stcum[, 1],
                    vars = Stcum[, 2]^2, skews = Stcum[, 3],
                    skurts = Stcum[, 4], fifths = Stcum[, 5],
                    sixths = Stcum[, 6], Six = Six, marginal = marginal,
                    lam = lam, size = size, prob = prob, rho = Rey,
```

```
                    seed = seed)
```

```
##
##  Constants: Distribution  1
##
##  Constants: Distribution  2
## All correlations are in feasible range!
```

Simulate the data

```
Sim1 <- rcorrvar(n = n, k_cat = 1, k_cont = 2, k_pois = 1, k_nb = 1,
                 method = "Polynomial", means = Stcum[, 1],
                 vars = Stcum[, 2]^2, skews = Stcum[, 3],
                 skurts = Stcum[, 4], fifths = Stcum[, 5],
                 sixths = Stcum[, 6], Six = Six, marginal = marginal,
                 lam = lam, size = size, prob = prob, rho = Rey,
                 seed = seed)
```

```
##
##  Constants: Distribution  1
##
##  Constants: Distribution  2
##
## Constants calculation time: 0.141 minutes
## Intercorrelation calculation time: 0.008 minutes
## Error loop calculation time: 0 minutes
## Total Simulation time: 0.149 minutes
```

Unpack the simulated data

```
ord_rv <- Sim1$ordinal_variables; names(ord_rv) <- "cat_rv"
cont_rv <- Sim1$continuous_variables; names(cont_rv) <- c("logistic_rv",
"Weibull_rv")
pois_rv <- Sim1$Poisson_variables; names(pois_rv) <- "Poisson_rv"
neg_bin_rv <- Sim1$Neg_Bin_variables; names(neg_bin_rv) <- "Neg_Bin_rv"
fake_data <- tibble(ord_rv, cont_rv, pois_rv, neg_bin_rv)
fake_data
```

```
## # A tibble: 10,000 x 5
##    cat_rv logistic_rv Weibull_rv Poisson_rv Neg_Bin_rv
##
##  1      1      -0.496       2.29          0          3
##  2      2       1.67        4.39          0          0
##  3      2       2.48        5.43          0          1
##  4      2      -1.02        3.86          2          2
##  5      2       4.26        5.98          1          2
##  6      2       0.945       2.33          0          1
##  7      1      -4.02        3.45          0          0
##  8      2      -1.31        7.78          1          1
##  9      1      -3.87        1.85          0          0
## 10      1       1.01        6.28          0          1
## # … with 9,990 more rows
```

Well, that's it for now. My short list of R packages for simulating data is far from being exhaustive and does not include some really good stuff, but it covers the basics. Hopefully, it will motivate you to acquire the good habit of using simulation to learn more about your data, or make your life easier if you have already acquired this habit. I hope to return to this topic again in the near future, and explore simulating survival data.

If you have any favorite R packages for simulating data please let me know.