DSP Data Generation Example

06/21/2015

Preface: This document was created as a twofold exercise. Firstly, to experiment with and obtain a basic understanding of the behavior and characteristics of the data generation sofware. And secondly to gain a little experience working in markdown - in this case `R Markdown`.

Generating data

First of all we have to load all of the necessary functions into the global environment. It is worth nothing that `knitr` very conveniently sets the `R` working directory to the location of the `R Markdown` file from which it is called when evaluating code chunks. So for example, if we want to *source* a file, then we specify its location relative to the `R Markdown` caller.

```
source("SampBaseline.R")
source("SampCycle.R")
source("SampDaily.R")
source("SampDataComb.R")
source("SampPreg.R")
set.seed(0)
```

The first thing to do when generating the data is to specify the subject id's. In this case we simply specify

```
subjId <- 1:300
```

although the labels could have been any unique identifiers of our choosing. Note that this intrinsically specifies the study size in terms of subjects (in this case 300).

Next we will sample the baseline data. Note that this could have been done at any time before sampling pregnancy outcome, but we will arbitrarily do it now. We include the appendage *pre* to the `R` object `baselinePre` and some of the following objects to emphasize that the data is "pre-completion". Of course the baseline will not change, but it is still named as such for consistency with the naming scheme.

```
baseNames <- c("age","race","bmi","gravid","edu","depr","smoke","drinkAlc","hormContrac")

baseDist <- expression( sample( c("< 30","30-35","35-38","38+"), size=1,
                                prob=c(0.41, 0.39, 0.11, 0.09) ),              #  <-- age
                        sample( c("cauc","black","hisp","other"),
                                size=1, prob=c(0.64, 0.12, 0.16, 0.08) ),      #  <-- race
                        sample( c("< 25","25-30","30+"), size=1,
                                prob=c(0.63, 0.21, 0.16) ),                    #  <-- bmi
                        sample( c("no","yes"), size=1, prob=c(0.42,0.58) ),    #  <-- gravid
                        sample( c("no college","college","grad/prof"),
                                size=1, prob=c(0.58, 0.30, 0.12) ),            #  <-- edu
                        sample( c("no","yes"), size=1, prob=c(0.90, 0.10) ),   #  <-- depr
                        sample( c("no","yes"), size=1, prob=c(0.82, 0.18) ),   #  <-- smoke
                        sample( c("no","yes"), size=1, prob=c(0.44, 0.56) ),   #  <-- drinkAlc
                        sample( c("no","yes"), size=1, prob=c(0.83, 0.17) ) )  #  <-- hormContrac

baselinePre <- sampBaseline(subjId=subjId, varDist=baseDist, varNames=baseNames)
```

```r
dim(baselinePre)
```

```
## [1] 300   9
```

```r
names(baselinePre)
```

```
## [1] "age"         "race"        "bmi"         "gravid"      "edu"
## [6] "depr"        "smoke"       "drinkAlc"    "hormContrac"
```

Next we sample the cycle data. The data-generation mechanism here requires a little more delicacy to simulate. First, we wish to specify the (possible) left- and right-truncation observed in the study subjects. This behavior is specified as the `entryDist` and `lengthDist` variables. `entryDist` is interpreted as the *number of cycles during which the subject has attempted to become pregnant*, and should specify a distribution with support contained in the natural numbers. `lengthDist` is interpreted as the *number of cycles the subject is willing to stay in the study after the first*, assuming no pregnancy occurs or that the study ends, and should have support contained in the nonnegative integers. `maxStudyCyc` is an optional third parameter, which specifies the maximum attempted cycle allowed in the study. The distributions for the cycle-specific variables are specified similarly to the baseline variable specifications.

```r
entryDist <- expression( sample.int(n=3, size=1, prob=c(0.34,0.33,0.33)) )
lengthDist <- expression( rgeom(n=1, prob=0.25) )

cycNames <- c("cycleLen","opk_use","bleed_intermen","bleed_luteal")

cycDist <- expression( sample( c("< 28 days","28-31 days","32+ days"),
                               size=1, prob=c(0.11, 0.69, 0.20) ),             #  <-- cycleLen
                       sample( c("no","yes"), size=1, prob=c(0.70, 0.30) ),    #  <-- opk_use
                       sample( c("no","yes"), size=1, prob=c(0.57, 0.43) ),    #  <-- bleed_intermen
                       sample( c("no","yes"), size=1, prob=c(0.64, 0.36) ) )   #  <-- bleed_luteal

cyclePre <- sampCycle(subjId=subjId, entryDist=entryDist, lengthDist=lengthDist,
                      maxStudyCyc=12, varDist=cycDist, varNames=cycNames)
```

```r
length( unique(cyclePre$subjId) )
```

```
## [1] 300
```

```r
dim(cyclePre)
```

```
## [1] 1208    6
```

```r
names(cyclePre)
```

```
## [1] "subjId"         "cycle"          "cycleLen"       "opk_use"
## [5] "bleed_intermen" "bleed_luteal"
```

Now we want to sample the daily data. Besides the now familiar variable specification, we also have to specify the length of what we consider to be the fertile window. `nTot` specifies the number of *potential* cycles which may occur.

```
dailyNames <- c("intercourse","cm_monit","lube")

dailyDist <- expression( sample( c("no","yes"), size=1, prob=c(0.64, 0.36) ),    #  <-- intercourse
                         sample( c("didn't check","type 1",
                                   "type 2","type 3","type 4"),
                                 size=1, prob=c(0.04, 0.05, 0.08, 0.04, 0.79) ), #  <-- cm_monit
                         sample( c("no","yes"), size=1, prob=c(0.88, 0.12) ) )   #  <-- lube

dailyPre <- sampDaily(nTot=nrow(cyclePre), fwLen=5, varDist=dailyDist, varNames=dailyNames)

dim(dailyPre)
```

```
## [1] 6040    4
```

```
names(dailyPre)
```

```
## [1] "cycleDay"    "intercourse" "cm_monit"    "lube"
```

Next we will combine the potential cycle datasets into a single dataset by expanding baseline- and cycle-specific variables into daily variables (i.e. a baseline variable is the same for each day observation within a subject and similarly for cycle variables). Then we specify the effect size for the variables with a nonzero effect on probability. `betaDays` specifies the effect size for fertile window day and uses cell-means coding, and so should have length equal to the number of days in the fertile window. `betaCovs` specifies the effect size for other covariates and uses reference-cell coding, and should have length one less then the number of factors in the variable (or length 1 for a continuous variable). Then we sample pregnancies and remove the unneeded potential cycles from the datasets.

```
analyData <- sampDataComb(baseline=baselinePre, cycle=cyclePre, daily=dailyPre, fwLen=5)

betaDays <- log( c(0.14, 0.08, 0.34, 0.31, 0.08) )
betaCovs <- list( age = c(-0.08, -0.43, -1.03),
                  bmi = c(-0.22, -0.47),
                  gravid = 1.21 )

pregVecPre <- sampPreg(dspDat=analyData, betaDays=betaDays, betaCovs=betaCovs, phi=1, fwLen=5)

finalData <- rmSuperfluous(baselinePre, cyclePre, dailyPre, pregVecPre)
list2env(finalData, envir=.GlobalEnv)
rm(list=setdiff(ls(), c("baseline","cycle","daily")))
```

Dataset characteristics

We can check some characteristics of the simulated dataset. In particular the proportion of cycles that result in a pregnancy can be computed with the following command.

```
mean(cycle$pregInd)
```

```
## [1] 0.2357043
```

Furthermore, the average number of cycles per individual can be computed.

```r
mean(table(cycle$subjId))
```

```
## [1] 2.39
```