ACE Models with the NLSY

William Howard Beasley (Howard Live Oak LLC, Norman)
Joseph Lee Rodgers (Vanderbilt University, Nashville)
David Bard (University of Oklahoma Health Sciences Center, OKC)
Kelly Meredith (Oklahoma City University, OKC)
Michael D. Hunter (University of Oklahoma, Norman)

November 20, 2013

Abstract

We describe how to use the NlsyLinks package to examine various biometric models, using the NLSY79.

Contents

1 Terminology

This package considers both Gen1 and Gen2 subjects. **Gen1** refers to subjects in the original NLSY79 sample (http://www.bls.gov/nls/nlsy79.htm). **Gen2** subjects are the biological offspring of the Gen1 females -*i.e.*, those in the NLSY79 Children and Young Adults sample (http://www.bls.gov/nls/nlsy79ch.htm). The NLSY97 is a third dataset that can be used for behavior genetic research (http://www.bls.gov/nls/nlsy97.htm), although this vignette focuses on the two generations in the NLSY79.

Standard terminology is to refer Gen2 subjects as 'children' when they are younger than age 15 (NSLY79-C), and as 'young adults' when they are 15 and older (NLSY79-YA); though they are the same respondents, different funding mechanisms and different survey items necessitate the distinction. This cohort is sometimes abbreviated as 'NLSY79-C', 'NLSY79C', 'NLSY-C' or 'NLSYC'.

The **SubjectTag** variable uniquely identify NLSY79 subjects when a dataset contains both generations. For Gen2 subjects, the `SubjectTag` is identical to their CID (*i.e.*, C00001.00 -the ID assigned in the NLSY79-Children files). However for Gen1 subjects, the `SubjectTag` is their CaseID (*i.e.*, R00001.00), with "00" appended. This manipulation is necessary to identify subjects uniquely in inter-generational datasets. A Gen1 subject with an ID of 43 becomes 4300. The `SubjectTag`s of her four children remain 4301, 4302, 4303, and 4304.

The **expected coefficient of relatedness** of a pair of subjects is typically represented by the variable `R`. Examples are: Monozygotic twins have `R`=1; dizygotic twins have `R`=0.5; full siblings (*i.e.*, those who share both biological parents) have `R`=0.5; half-siblings (*i.e.*, those who share exactly one biological parent) have `R`=0.25; adopted siblings have `R`=0.0. Other possibilities exist too. The font (and hopefully their context) should distinguish the variable `R` from the software R.

A subject's `ExtendedID` indicates their extended family. Two subjects will be in the same extended family if either: [1] they are Gen1 housemates, [2] they are Gen2 siblings, [3] they are Gen2 cousins (*i.e.*, they have mothers who are Gen1 sisters in the NLSY79), [4] they are mother and child (in Gen1 and Gen2, respectively), or [5] they are (aunt—uncle) and (niece—nephew) (in Gen1 and Gen2, respectively).

An **outcome variable** is directly relevant to the applied researcher; these might represent constructs like height, IQ, and income. A **plumbing variable** is necessary to manage BG datasets; examples are `R`, a subject's ID, and the date of a subject's last survey.

An ACE model is the basic biometrical model used by Behavior Genetic researchers, where the genetic and environmental effects are assumed to be additive. The three primary variance components are (1) the proportion of variability due to a shared genetic influence (typically represented as a^2, or sometimes h^2), (2) the proportion of variability due to shared common environmental influence (typically c^2), and (3) the proportion of variability due to unexplained/residual/error influence (typically e^2).

The variables are scaled so that they account for all observed variability in the outcome variable; specifically: $a^2 + c^2 + e^2 = 1$. Using appropriate designs that can logically distinguish these different components (under carefully specified assumptions), the basic biometrical modeling strategy is to estimate the magnitude of a^2, c^2, and e^2 within the context of a particular model. For gentle introductions to Behavior Genetic research, we recommend Plomin (1990) and Carey (2003). For more in-depth ACE model-fitting strategies, we recommend Neale & Cardon (1992). //This paragraph may get moved to the yet-to-be-written introduction that precedes the Terminology section.

The **NLS Investigator** (http://www.nlsinfo.org/investigator/) is the best way to obtain the NLSY79 and NLSY97 datasets. See our vignette dedicated to the NLS Investigator by typing `vignette("NlsInvestigator")` or by visiting http://cran.r-project.org/web/packages/NlsyLinks/.

Before starting the real examples, first verify that the NlsyLinks package is installed correctly. If not, refer to Appendix C.

```
any(.packages(all.available = TRUE) == "NlsyLinks")  #Should evaluate to TRUE.
```

```
## [1] TRUE
```

```
require(NlsyLinks)  #Load the package into the current session.
```

```
## Loading required package:  NlsyLinks
```

The package's documentation manual can be opened by typing `?NlsyLinks` in R or RStudio.

2 Example: DF analysis with a Simple Outcome for Gen2 Subjects, Using a Package Variable

The vignette's first example uses a simple statistical model and all available Gen2 subjects. The `CreatePairLinksDoubleEntered` function will create a data frame where each represents one pair of siblings, respective of order (*i.e.*, there is a row for Subjects 201 and 202, and a second row for Subjects 202 and 201). This function examines the subjects' IDs and determines who is related to whom (and by how much). By default, each row it produces has at least six values/columns: (i) ID for the older member of the kinship pair: `Subject1Tag`, (ii) ID for the younger member: `Subject2Tag`, (iii) ID for their extended family: `ExtendedID`, (iv) their estimated coefficient of genetic relatedness: `R`, (v *and beyond*) outcome values for the older member; (vi *and beyond*) outcome values for the younger member.

A DeFries-Fulker (**DF**) Analysis uses linear regression to estimate the a^2, c^2, and e^2 of a univariate biometric system. The interpretations of the DF analysis can be found in Rodgers & Kohler (2005) and Rodgers, Rowe, & Li (1999). This vignette example uses the newest variation, which estimates two parameters; the corresponding function is called `DeFriesFulkerMethod3`. The steps are:

1. Use the NLS Investigator to select and download a Gen2 dataset.
2. Open R and create a new script (see Appendix B) and load the NlsyLinks package. If you haven't done so, first install the NlsyLinks package (see Appendix C).
3. Within the R script, load the linking dataset. Then select only Gen2 subjects. The 'Pair' version of the linking dataset is essentially an upper triangle of a symmetric sparse matrix.
4. Load and assign the `ExtraOutcomes79` dataset.
5. Specify the outcome variable name and filter out all subjects who have a negative value in this variable. The NLSY typically uses negative values to indicate different types of missingness (see 'Further Information' below).
6. Create a double-entered file by calling the `CreatePairLinksDoubleEntered` function. At minimum, pass the (i) outcome dataset, the (ii) linking dataset, and the (iii) name(s) of the outcome variable(s). *(There are occasions when a single-entered file is more appropriate for a DF analysis. See Rodgers & Kohler, 2005, for additional information.)*
7. Use `DeFriesFulkerMethod3` function (i.e., general linear model) to estimate the coefficients of the DF model.

```
### R Code for Example DF analysis with a simple outcome and Gen2 subjects
#Step 2: Load the package containing the linking routines.
require(NlsyLinks)

#Step 3: Load the LINKING dataset and filter for the Gen2 subjects
dsLinking <- subset(Links79Pair, RelationshipPath=="Gen2Siblings")
summary(dsLinking) #Notice there are 11,088 records (one for each unique pair).
```

```
##    ExtendedID    SubjectTag_S1     SubjectTag_S2          R
##  Min.   :    2   Min.   :    201   Min.   :    202   Min.   :0.250
##  1st Qu.: 3155   1st Qu.: 315501   1st Qu.: 315503   1st Qu.:0.250
##  Median : 6114   Median : 611402   Median : 611404   Median :0.500
##  Mean   : 5933   Mean   : 593658   Mean   : 593660   Mean   :0.417
##  3rd Qu.: 8511   3rd Qu.: 851101   3rd Qu.: 851103   3rd Qu.:0.500
##  Max.   :12673   Max.   :1267301   Max.   :1267302   Max.   :1.000
##        RelationshipPath
##  Gen1Housemates:    0
##  Gen2Siblings  :11088
##  Gen2Cousins   :    0
```

```
##  ParentChild   :    0
##  AuntNiece     :    0
##
```

```
#Step 4: Load the OUTCOMES dataset, and then examine the summary.
dsOutcomes <- ExtraOutcomes79 #'ds' stands for 'Data Set'
summary(dsOutcomes)
```

```
##    SubjectTag        SubjectID         Generation   HeightZGenderAge WeightZGenderAge
##  Min.   :    100   Min.   :      1   Min.   :1.00   Min.   :-3       Min.   :-3
##  1st Qu.: 314025   1st Qu.:   5998   1st Qu.:1.00   1st Qu.:-1       1st Qu.:-1
##  Median : 620050   Median :  12000   Median :1.00   Median : 0       Median : 0
##  Mean   : 618600   Mean   : 289254   Mean   :1.48   Mean   : 0       Mean   : 0
##  3rd Qu.: 914501   3rd Qu.: 577403   3rd Qu.:2.00   3rd Qu.: 1       3rd Qu.: 1
##  Max.   :1268600   Max.   :1267501   Max.   :2.00   Max.   : 3       Max.   : 5
##                                                     NA's   :4711     NA's   :4719
##  AfqtRescaled2006Gaussified      Afi             Afm        MathStandardized
##  Min.   :-3                 Min.   : 2      Min.   : 0      Min.   : 65
##  1st Qu.:-1                 1st Qu.:15      1st Qu.:12      1st Qu.: 92
##  Median : 0                 Median :17      Median :13      Median :100
##  Mean   : 0                 Mean   :17      Mean   :13      Mean   :100
##  3rd Qu.: 1                 3rd Qu.:18      3rd Qu.:14      3rd Qu.:108
##  Max.   : 3                 Max.   :27      Max.   :19      Max.   :135
##  NA's   :12510              NA's   :12740   NA's   :18165   NA's   :15085
```

```
#Step 5: This step isn't necessary for this example, because Kelly Meredith already
#   groomed the values.  If the negative values (which represent NLSY missing or
#   skip patterns) still exist, then:
dsOutcomes$MathStandardized[dsOutcomes$MathStandardized < 0] <- NA

#Step 6: Create the double entered dataset.
dsDouble <- CreatePairLinksDoubleEntered(
  outcomeDataset=dsOutcomes,
  linksPairDataset=dsLinking,
  outcomeNames=c('MathStandardized')
)
summary(dsDouble) #Notice there are 22176=(2*11088) records now (two for each unique pair).
```

```
##  SubjectTag_S1     SubjectTag_S2       ExtendedID          R
##  Min.   :    201   Min.   :    201   Min.   :    2   Min.   :0.250
##  1st Qu.: 315502   1st Qu.: 315502   1st Qu.: 3155   1st Qu.:0.250
##  Median : 611404   Median : 611404   Median : 6114   Median :0.500
##  Mean   : 593659   Mean   : 593659   Mean   : 5933   Mean   :0.417
##  3rd Qu.: 851102   3rd Qu.: 851102   3rd Qu.: 8511   3rd Qu.:0.500
##  Max.   :1267302   Max.   :1267302   Max.   :12673   Max.   :1.000
##
##        RelationshipPath MathStandardized_S1 MathStandardized_S2
##  Gen1Housemates:    0   Min.   : 65         Min.   : 65
##  Gen2Siblings  :22176   1st Qu.: 90         1st Qu.: 90
##  Gen2Cousins   :    0   Median : 98         Median : 98
##  ParentChild   :    0   Mean   : 98         Mean   : 98
##  AuntNiece     :    0   3rd Qu.:107         3rd Qu.:107
##                         Max.   :135         Max.   :135
##                         NA's   :3885        NA's   :3885
```

```
#Step 7: Estimate the ACE components with a DF Analysis
```

```
ace <- DeFriesFulkerMethod3(
    dataSet=dsDouble,
    oName_S1="MathStandardized_S1",
    oName_S2="MathStandardized_S2")
ace
```

```
## [1] "Results of ACE estimation: [show]"
##  ASquared  CSquared  ESquared CaseCount
## 7.735e-01 1.468e-01 7.968e-02 1.668e+04
```

Further Information: If the different reasons of missingness are important, further work is necessary. For instance, some analyses that use item `Y19940000` might need to distinguish a response of "Don't Know" (which is coded as -2) from "Missing" (which is coded as -7). For this vignette example, we'll assume it's safe to clump the responses together.

3 Example: DF analysis with a univariate outcome from a Gen2 Extract

The vignette's second example differs from the previous example in two ways. First, the outcome variables are read from a CSV (comma separated values file) that was downloaded from the NLS Investigator. Second, the DF analysis is called through the function `AceUnivariate`; this function is a wrapper around some simple ACE methods, and will help us smoothly transition to more techniques later in the vignette.

The steps are:

1. Use the NLS Investigator to select and download a Gen2 dataset. Select the variables 'length of gestation of child in weeks' (`C03280.00`), 'weight of child at birth in ounces' (`C03286.00`), and 'length of child at birth' (`C03288.00`), and then download the *.zip file to your local computer.
2. Open R and create a new script (see Appendix B) and load the `NlsyLinks` package.
3. Within the R script, load the linking dataset. Then select only Gen2 subjects.
4. Read the CSV into R as a `data.frame` using `ReadCsvNlsy79Gen2`.
5. Verify the desired outcome column exists, and rename it something meaningful to your project. It is important that the `data.frame` is reassigned (*i.e.*, `ds <- RenameNlsyColumn(...)`). In this example, we rename column `C0328800` to `BirthWeightInOunces`.
6. Filter out all subjects who have a negative `BirthWeightInOunces` value. See the 'Further Information' note in the previous example.
7. Create a double-entered file by calling the `CreatePairLinksDoubleEntered` function. At minimum, pass the (i) outcome dataset, the (ii) linking dataset, and the (iii) name(s) of the outcome variable(s).
8. Call the `AceUnivariate` function to estimate the coefficients.

```
### R Code for Example of a DF analysis with a simple outcome and Gen2 subjects
#Step 2: Load the package containing the linking routines.
require(NlsyLinks)

#Step 3: Load the linking dataset and filter for the Gen2 subjects
dsLinking <- subset(Links79Pair, RelationshipPath=="Gen2Siblings")

#Step 4: Load the outcomes dataset from the hard drive and then examine the summary.
#   Your path might be: filePathOutcomes <- 'C:/BGResearch/NlsExtracts/Gen2Birth.csv'
filePathOutcomes <- file.path(path.package("NlsyLinks"), "extdata", "Gen2Birth.csv")
```

```r
dsOutcomes <- ReadCsvNlsy79Gen2(filePathOutcomes)
summary(dsOutcomes)
```

```
##    SubjectTag        SubjectID         ExtendedID      Generation SubjectTagOfMother
##  Min.   :    201   Min.   :    201   Min.   :    2   Min.   :2    Min.   :    200
##  1st Qu.: 310302   1st Qu.: 310302   1st Qu.: 3101   1st Qu.:2    1st Qu.: 310300
##  Median : 604607   Median : 604607   Median : 6045   Median :2    Median : 604600
##  Mean   : 601313   Mean   : 601313   Mean   : 6007   Mean   :2    Mean   : 601311
##  3rd Qu.: 876202   3rd Qu.: 876202   3rd Qu.: 8757   3rd Qu.:2    3rd Qu.: 876200
##  Max.   :1267501   Max.   :1267501   Max.   :12675   Max.   :2    Max.   :1267500
##                                      NA's   :2
##     C0005300       C0005400        C0005700       C0328000       C0328600
##  Min.   :1.00   Min.   :-3.00   Min.   :  -3   Min.   :-7.0   Min.   : -7
##  1st Qu.:2.00   1st Qu.: 1.00   1st Qu.:1981   1st Qu.:37.0   1st Qu.: 99
##  Median :3.00   Median : 1.00   Median :1985   Median :39.0   Median :115
##  Mean   :2.34   Mean   : 1.49   Mean   :1986   Mean   :33.5   Mean   :104
##  3rd Qu.:3.00   3rd Qu.: 2.00   3rd Qu.:1990   3rd Qu.:39.0   3rd Qu.:128
##  Max.   :3.00   Max.   : 2.00   Max.   :2008   Max.   :51.0   Max.   :768
##
##     C0328800
##  Min.   :-7.0
##  1st Qu.:18.0
##  Median :20.0
##  Mean   :16.5
##  3rd Qu.:21.0
##  Max.   :48.0
##
```

```r
#Step 5: Verify and rename an existing column.
VerifyColumnExists(dsOutcomes, "C0328600") #Should return '10' in this example.
```

```
## [1] 10
```

```r
dsOutcomes <- RenameNlsyColumn(dsOutcomes, "C0328600", "BirthWeightInOunces")

#Step 6: For this item, a negative value indicates the parent refused, didn't know,
#   invalidly skipped, or was missing for some other reason.
#   For our present purposes, we'll treat these responses equivalently.
#   Then clip/Winsorized/truncate the weight to something reasonable.
dsOutcomes$BirthWeightInOunces[dsOutcomes$BirthWeightInOunces < 0] <- NA
dsOutcomes$BirthWeightInOunces <- pmin(dsOutcomes$BirthWeightInOunces, 200)

#Step 7: Create the double entered dataset.
dsDouble <- CreatePairLinksDoubleEntered(
  outcomeDataset=dsOutcomes,
  linksPairDataset=dsLinking,
  outcomeNames=c('BirthWeightInOunces')
)

#Step 8: Estimate the ACE components with a DF Analysis
ace <- AceUnivariate(
  method="DeFriesFulkerMethod3",
  dataSet=dsDouble,
  oName_S1="BirthWeightInOunces_S1",
  oName_S2="BirthWeightInOunces_S2"
```

```
)
ace
```

```
## [1] "Results of ACE estimation: [show]"
##  ASquared  CSquared  ESquared CaseCount
## 5.042e-01 1.777e-01 3.182e-01 1.744e+04
```

For another example of incorporating CSVs downloaded from the NLS Investigator, please see the "Race and Gender Variables" entry in the FAQ.

4 Example: Multiple Group SEM of a Simple Outcome for Gen2 Subjects

The example differs from the first one by the statistical mechanism used to estimate the components. The first example uses multiple regression to estimate the influence of the shared genetic and environmental factors, while this example uses structural equation modeling (SEM).

The `CreatePairLinksSingleEntered` function will create a `data.frame` where each row represents one unique pair of siblings, *irrespective of order*. Other than producing half the number of rows, this function is identical to `CreatePairLinksDoubleEntered`.

The steps are:

(Steps 1-5 proceed identically to the first example.)

6. Create a *single*-entered file by calling the `CreatePairLinksSingleEntered` function. At minimum, pass the (i) outcome dataset, the (ii) linking dataset, and the (iii) name(s) of the outcome variable(s).
7. Declare the names of the outcome variables corresponding to the two members in each pair. Assuming the variable is called 'ZZZ' and the preceeding steps have been followed, the variable 'ZZZ_S1' corresponds to the first members and ZZZ_S2' corresponds to the second members.
8. Create a GroupSummary `data.frame`, which identifies the `R` groups that should be considered by the model. Inspect the output to see if the groups show unexpected or fishy differences.
9. Create a `data.frame` with cleaned variables to pass to the SEM function. This `data.frame` contains only the three necessary rows and columns.
10. Estimate the SEM with the `lavaan` package. The function returns an `S4` object, which shows the basic ACE information.
11. Inspect details of the SEM, beyond the ACE components. In this example, we look at the fit stats and the parameter estimates. The `lavaan` package has additional methods that may be useful for your purposes.

```
### R Code for Example lavaan estimation analysis with a simple outcome and Gen2 subjects
#Steps 1-5 are explained in the vignette's first example:
require(NlsyLinks)
dsLinking <- subset(Links79Pair, RelationshipPath=="Gen2Siblings")
dsOutcomes <- ExtraOutcomes79
dsOutcomes$MathStandardized[dsOutcomes$MathStandardized < 0] <- NA

#Step 6: Create the single entered dataset.
dsSingle <- CreatePairLinksSingleEntered(outcomeDataset=dsOutcomes,
  linksPairDataset=dsLinking, outcomeNames=c('MathStandardized'))

#Step 7: Declare the names for the two outcome variables.
```

```r
oName_S1 <- "MathStandardized_S1" #Stands for Outcome1
oName_S2 <- "MathStandardized_S2" #Stands for Outcome2

#Step 8: Summarize the R groups and determine which groups can be estimated.
dsGroupSummary <- RGroupSummary(dsSingle, oName_S1, oName_S2)
dsGroupSummary
```

```
##       R Included PairCount O1Mean O2Mean O1Variance O2Variance O1O2Covariance
## 1 0.250     TRUE      2691  95.11  95.98      126.9      150.1          42.04
## 2 0.375     TRUE       133  93.54  93.41      163.1      131.6          46.56
## 3 0.500     TRUE      5493  99.89 100.02      168.7      172.9          90.12
## 4 0.750    FALSE         2 108.50 106.00      220.5       18.0          63.00
## 5 1.000     TRUE        21  98.21  96.02      289.4      215.2         229.11
##   Correlation Determinant PosDefinite
## 1      0.3046       17279        TRUE
## 2      0.3179       19286        TRUE
## 3      0.5276       21055        TRUE
## 4      1.0000           0       FALSE
## 5      0.9179        9808        TRUE
```

```r
#Step 9: Create a cleaned dataset
dsClean <- CleanSemAceDataset(dsDirty=dsSingle, dsGroupSummary, oName_S1, oName_S2)

#Step 10: Run the model
ace <- AceLavaanGroup(dsClean)
ace
```

```
## [1] "Results of ACE estimation: [show]"
##  ASquared  CSquared  ESquared CaseCount
##    0.6211    0.2100    0.1689 8338.0000
```

```r
#Notice the `CaseCount' is 8,390 instead of 17,440.
#  This is because (a) one pair with R=.75 was excluded, and
#  (b) the SEM uses a single-entered dataset instead of double-entered.
#
#Step 11: Inspect the output further
require(lavaan) #Load the package to access methods of the lavaan class.
GetDetails(ace)
```

```
## lavaan (0.5-15) converged normally after  55 iterations
##
##   Number of observations per group
##   1                                               2691
##   2                                                133
##   3                                               5493
##   4                                                 21
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              447.303
##   Degrees of freedom                                16
##   P-value (Chi-square)                           0.000
##
## Chi-square for each group:
##
##   1                                             281.432
##   2                                              31.252
```

```
##   3                                          127.180
##   4                                            7.439
```

```
#Examine fit stats like Chi-Squared, RMSEA, CFI, etc.
fitMeasures(GetDetails(ace)) #'fitMeasures' is defined in the lavaan package.
```

```
##              fmin             chisq                df            pvalue
##             0.027           447.303            16.000             0.000
##    baseline.chisq       baseline.df   baseline.pvalue               cfi
##          2107.252             4.000             0.000             0.795
##               tli              nnfi               rfi               nfi
##             0.949             0.949             0.947             0.788
##              pnfi               ifi               rni              logl
##             3.151             0.794             0.795        -65103.976
## unrestricted.logl              npar               aic               bic
##        -64880.324             4.000        130215.952        130244.066
##            ntotal              bic2             rmsea    rmsea.ci.lower
##          8338.000        130231.355             0.114             0.105
##    rmsea.ci.upper      rmsea.pvalue               rmr        rmr_nomean
##             0.123             0.000            10.095            12.902
##              srmr       srmr_nomean             cn_05             cn_01
##             0.130             0.090           491.178           597.499
##               gfi              agfi              pgfi               mfi
##             0.999             0.999             0.799             0.974
```

```
#Examine low-level details like each group's individual parameter estimates and standard
#  errors.  Uncomment the next line to view the entire output (which is roughly 4 pages).
#summary(GetDetails(ace))
```

5 Example: Multiple Group SEM of a Simple Outcome for Gen1 Subjects

The example differs from the previous one in three ways. First, Gen1 subjects are used. Second, standardized height is used instead of math. Third, pairs are dropped if their R is zero; we return to this last issue after the code is run.

```
### R Code for Example lavaan estimation analysis with a simple outcome and Gen1 subjects
#Steps 1-5 are explained in the vignette's first example:
require(NlsyLinks)
dsLinking <- subset(Links79Pair, RelationshipPath=="Gen1Housemates")
dsOutcomes <- ExtraOutcomes79
#The HeightZGenderAge variable is already groomed

#Step 6: Create the single entered dataset.
dsSingle <- CreatePairLinksSingleEntered(outcomeDataset=dsOutcomes,
  linksPairDataset=dsLinking, outcomeNames=c('HeightZGenderAge'))

#Step 7: Declare the names for the two outcome variables.
oName_S1 <- "HeightZGenderAge_S1"
oName_S2 <- "HeightZGenderAge_S2"

#Step 8: Summarize the R groups and determine which groups can be estimated.
dsGroupSummary <- RGroupSummary(dsSingle, oName_S1, oName_S2)
```

```
# rGroupsToDrop <- c( 0.0625 )
# dsGroupSummary[dsGroupSummary£R %in% rGroupsToDrop, "Included"] <- FALSE
dsGroupSummary
```

```
##      R Included PairCount   O1Mean   O2Mean O1Variance O2Variance O1O2Covariance
## 1 0.25     TRUE       280  0.04830  0.05605     1.0182     1.1847         0.2656
## 2 0.50     TRUE      3894 -0.04987 -0.02789     0.9736     1.0194         0.4660
## 3 1.00     TRUE        11 -0.08652 -0.00904     0.3171     0.9518         0.3583
##   Correlation Determinant PosDefinite
## 1      0.2418      1.1357        TRUE
## 2      0.4678      0.7752        TRUE
## 3      0.6522      0.1734        TRUE
```

```
#Step 9: Create a cleaned dataset
dsClean <- CleanSemAceDataset(dsDirty=dsSingle, dsGroupSummary, oName_S1, oName_S2)

#Step 10: Run the model
ace <- AceLavaanGroup(dsClean)
ace
```

```
## [1] "Results of ACE estimation: [show]"
##  ASquared  CSquared  ESquared CaseCount
##    0.7040    0.1124    0.1836 4185.0000
```

```
#Step 11: Inspect the output further (see the final step in the previous example).
```

Most of them responded they were `Non-relatives` to the explict items asked in 1979 (*i.e.*, NLSY79 variables `R00001.50` through `R00001.59`). Yet their height's observed correlations is far larger than would be expected for a sample of unrelated subjects. Since our team began BG research with the NLSY in the mid-1990s, the R=0 group has consistently presented higher than expected correlations, across many domains of outcome variables. For a long time, we have substantial doubts that subject pairs in this group share a low proportion of their selective genes. Consequently, we suggest applied researchers consider excluding this group from their biometric analyses.

If you wish to exclude additional groups from the analyses, one line of code in Step 8 should change. For instance, two exclude ambiguous sibs (in addition to $R = 0$ pairs), change
`rGroupsToDrop <- c( 0 )`
to
`rGroupsToDrop <- c( 0, .375 )`.

6 Example: Midstream data manipulation with SAS

The example differs from the previous one substantial way: After R is used to link the related pairs, and connect them to their outcome values, the dataset is exported so that the user can further manipulate the data in SAS.

After a presentation, several audience members at the 2012 BGA meeting informed us that this vignette example would help them be more efficient. This approach is also consistent with our feeling that analysts should use the workflow tools that are best suited to their needs and capabilities.

Should we export/import as CSVs (which would make the R code much simpler, because the foreign package wouldn't have to be loaded), but increase the code necessary on the SAS side? I really don't want to mess with sasbdats and xports.

A Appendix: Receiving Help for the NlsyLinks Package

A portion of our current grant covers a small, part-time support staff. If you have questions about BG research with our kinship links, or questions about our package, we'd like to hear from you.

We provide personal support for researchers in several ways. Perhaps the best place to start are the forums on R-Forge (`http://r-forge.r-project.org/forum/?group_id=1330`); there are forums for people using R, as well as other software such as SAS. This post is a good overview of the current project is, which originally was an email Joe sent to previous users of our kinship links (many of them are/were SAS users).

B Appendix: Creating and Saving R Scripts

There are several options and environments for executing R code. Our current recommendation is RStudio, because it is easy to install, and has features targeting beginnner and experienced R users. We've had good experiences with it on Windows, OS X, and Ubuntu Linux.

RStudio allows you to create and save R files; these are simply text files that have an file extension of '.R'. RStudio will execute the commands written in the file. Help documentation for RStudio can be found at `http://www.rstudio.com/ide/docs/`.

C Appendix: Installing and Loading the NlsyLinks Package

There are three operations you'll typically do with a package: (a) install, (b) load, and (c) update.

The simplest way to **install** NlsyLinks is to type `install.packages("NlsyLinks")`. You may be asked to select a CRAN mirror to download the package from; if so, choose a close location.

R then will download NlsyLinks on your local computer. It may try to save and install the package to a location that you don't have permission to write files in. If so, R will ask if you would like to install it to a better location (*i.e.*, somewhere you do have permission to write files). Approve this decision (which is acceptable for everyone except for some network administrators).

For a given computer, you'll need to *install* a package only once for each version of R (new versions of R are released every few months). However, you'll need to *load* a package in every session that you call its functions. To **load** NlsyLinks, type either `library(NlsyLinks)` or `require(NlsyLinks)`; (the difference between the two commands is likely irrelevant for your uses.) Loading reads NlsyLinks information from the hard drive and places it in temporary memory. Once it's loaded, you won't need to load it again until R is closed and reopened later.

Developers are continually improving their packages by adding functions and documentation. These newer versions are then uploaded to the CRAN servers. You may **update** all your installed packages at once by typing `update.packages()`. The command checks a CRAN server for newer versions of the packages installed on your local machine. Then they are automatically downloaded and installed.

The grant supporting NlsyLinks extends until Summer 2014. Until then, we'll be including new features and documentation, as we address additional user needs (if you have suggestions, we'd like to hear from you). When the NLSY periodically updates its data, we'll update our kinship links (embedded in NlsyLinks) with the newest information.

D Appendix: References

Carey, Gregory (2002). *Human Genetics for the Social Sciences.* Sage.

Plomin, Robert (1990). *Nature and nurture: an introduction to human behavioral genetics.* Brooks/Cole Publishing Company.

Rodgers, Joseph Lee, & Kohler, Hans-Peter (2005). Reformulating and simplifying the DF analysis model. *Behavior Genetics, 35 (2)*, 211-217.

Rodgers, Joseph Lee, Rowe, David C., & Li, Chengchang (1994). Beyond nature versus nurture: DF analysis of nonshared influences on problem behaviors. *Developmental Psychology, 30 (3)*, 374-384.

Neale, Michael C., & Cardon, Lou R. (1992). *Methodology for genetic studies of twins and families.* Norwell, MA: Kluwer Academic Publishers. (Also see Neale & Maes: `http://www.vipbg.vcu.edu/OpenMxFall09/NMbook05.pdf`).

E Notes

This package's development was largely supported by the NIH Grant 1R01HD65865, "NLSY Kinship Links: Reliable and Valid Sibling Identification" (PI: Joe Rodgers; Vignette Construction by Will Beasley)

F Version Information

- R version 3.0.2 (2013-09-25), `x86_64-w64-mingw32`
- Locale: `LC_COLLATE=English_United States.1252`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: knitr 1.5, lavaan 0.5-15, NlsyLinks 1.229
- Loaded via a namespace (and not attached): evaluate 0.5.1, formatR 0.10, highr 0.3, mnormt 1.4-5, pbivnorm 0.5-1, quadprog 1.5-5, stats4 3.0.2, stringr 0.6.2, tools 3.0.2