# 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)

October 17, 2018

**Abstract**

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

## Contents

## 1 Terminology

Researchers and grad students interested using the NLSY for Behavior Genetics and family research, please start with our 2016 article, *The NLSY Kinship Links: Using the NLSY79 and NLSY-Children Data to Conduct Genetically-Informed and Family-Oriented Research*.

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'.

Within our own team, we've mostly stopped using terms like 'NLSY79', 'NLSY79-C' and 'NLSY79-YA', because we conceptualize it as one big sample containing two related generations. It many senses, the responses collected from the second generation can be viewed as outcomes of the first generation. Likewise, the parents in the first generation provide many responses that can be viewed as explanatory variables for the 2nd generation. Depending on your research, there can be big advantages of using one cohort to augment the other. There are also survey items that provide information about the 3rd generation and the 0th generation.

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 statistical 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 uncommon possibilities are mentioned the documentation for `Links79Pair`. The font (and hopefully their context) should distinguish the variable $R$ from the software R. To make things slightly more confusing the computer variable for $R$ in the `Links79Pair` dataset is written with a monospace font: `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 & Maes, (1992).

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 https://cran.r-project.org/package=NlsyLinks.

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

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

## [1] TRUE

library(NlsyLinks) #Load the package into the current session.

##
## Attaching package:  'NlsyLinks'

## The following objects are masked _by_ '.GlobalEnv':
##
##    Links79Pair, Links79PairExpanded
```

The package's documentation manual can be opened by typing ?NlsyLinks in R or clicking the appropriate entry in RStudio's 'Packages' tab (which is usually in the lower right panel).

## 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
### R Code for Example DF analysis with a simple outcome and Gen2 subjects
#Step 2: Load the package containing the linking routines.
library(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.: 3154   1st Qu.: 315401   1st Qu.: 315403   1st Qu.:0.250
##  Median : 6105   Median : 610703   Median : 610705   Median :0.500
##  Mean   : 5927   Mean   : 593061   Mean   : 593063   Mean   :0.417
##  3rd Qu.: 8507   3rd Qu.: 851001   3rd Qu.: 851003   3rd Qu.:0.500
##  Max.   :12673   Max.   :1267301   Max.   :1267302   Max.   :1.000
##          RelationshipPath
##  Gen1Housemates:    0
##  Gen2Siblings  :11114
##  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.   :1.000   Min.   :-2.985   Min.   :-2.985
##  1st Qu.: 314025   1st Qu.:   5998   1st Qu.:1.000   1st Qu.:-0.724   1st Qu.:-0.677
##  Median : 620050   Median :  12000   Median :1.000   Median :-0.045   Median :-0.149
##  Mean   : 618600   Mean   : 289254   Mean   :1.476   Mean   :-0.006   Mean   : 0.001
##  3rd Qu.: 914501   3rd Qu.: 577403   3rd Qu.:2.000   3rd Qu.: 0.648   3rd Qu.: 0.533
##  Max.   :1268600   Max.   :1267501   Max.   :2.000   Max.   : 2.996   Max.   : 4.945
##                                                      NA's   :4711     NA's   :4719
##  AfqtRescaled2006Gaussified       Afi             Afm         MathStandardized
##  Min.   :-2.895             Min.   : 2.00   Min.   : 0.00   Min.   : 65.0
##  1st Qu.:-0.692             1st Qu.:15.00   1st Qu.:12.00   1st Qu.: 92.5
##  Median :-0.024             Median :17.00   Median :13.00   Median :100.0
##  Mean   :-0.011             Mean   :16.66   Mean   :12.78   Mean   :100.1
##  3rd Qu.: 0.660             3rd Qu.:18.00   3rd Qu.:14.00   3rd Qu.:108.5
##  Max.   : 2.994             Max.   :27.00   Max.   :19.00   Max.   :135.0
##  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')
)
```

```r
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.: 315402  1st Qu.: 315402  1st Qu.: 3154   1st Qu.:0.250
##  Median : 610704  Median : 610704  Median : 6105   Median :0.500
##  Mean   : 593062  Mean   : 593062  Mean   : 5927   Mean   :0.417
##  3rd Qu.: 851002  3rd Qu.: 851002  3rd Qu.: 8508   3rd Qu.:0.500
##  Max.   :1267302  Max.   :1267302  Max.   :12673   Max.   :1.000
##
##        RelationshipPath MathStandardized_S1 MathStandardized_S2
##  Gen1Housemates:    0   Min.   : 65.0       Min.   : 65.0
##  Gen2Siblings  :22228   1st Qu.: 90.0       1st Qu.: 90.0
##  Gen2Cousins   :    0   Median : 98.5       Median : 98.5
##  ParentChild   :    0   Mean   : 98.4       Mean   : 98.4
##  AuntNiece     :    0   3rd Qu.:107.0       3rd Qu.:107.0
##                         Max.   :135.0       Max.   :135.0
##                         NA's   :3926        NA's   :3926
```

```r
#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
## 7.734447e-01 1.469204e-01 7.963486e-02 1.668000e+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
### R Code for Example of a DF analysis with a simple outcome and Gen2 subjects
#Step 2: Load the package containing the linking routines.
library(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/gen2-birth.csv'
filePathOutcomes <- file.path(path.package("NlsyLinks"), "extdata", "gen2-birth.csv")
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.: 876203   3rd Qu.: 876203   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.   :1.000   Min.   :-3.000   Min.   :  -3   Min.   :-7.00   Min.   : -7.0
##   1st Qu.:2.000   1st Qu.: 1.000   1st Qu.:1981   1st Qu.:37.00   1st Qu.: 99.0
##   Median :3.000   Median : 1.000   Median :1985   Median :39.00   Median :115.0
##   Mean   :2.338   Mean   : 1.489   Mean   :1986   Mean   :33.51   Mean   :103.9
##   3rd Qu.:3.000   3rd Qu.: 2.000   3rd Qu.:1990   3rd Qu.:39.00   3rd Qu.:128.0
##   Max.   :3.000   Max.   : 2.000   Max.   :2008   Max.   :51.00   Max.   :768.0
##
##     C0328800
##   Min.   :-7.00
##   1st Qu.:18.00
##   Median :20.00
##   Mean   :16.51
##   3rd Qu.:21.00
##   Max.   :48.00
##

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

## [1] 10

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
## 5.103108e-01 1.752415e-01 3.144477e-01 1.744000e+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:
library(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.
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      2689  95.10450  95.97936   126.9489   150.1775       41.96914
## 2 0.375     TRUE       137  93.63139  93.36861   160.0120   136.6628       50.39790
## 3 0.500     TRUE      5491  99.89374 100.02868   168.7326   172.7293       90.04116
## 4 0.750    FALSE         2 108.50000 106.00000   220.5000    18.0000       63.00000
## 5 1.000     TRUE        21  98.21429  96.02381   289.4393   215.2369      229.10714
##    Correlation Determinant PosDefinite
## 1   0.3039577   17303.459        TRUE
## 2   0.3408090   19327.735        TRUE
## 3   0.5274225   21037.642        TRUE
## 4   1.0000000       0.000       FALSE
## 5   0.9179130    9807.933        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
##    0.6219253   0.2097338   0.1683408 8338.0000000

#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
library(lavaan) #Load the package to access methods of the lavaan class.
```

```
GetDetails(ace)

## lavaan 0.6-3 ended normally after 54 iterations
##
##   Optimization method                           NLMINB
##   Number of free parameters                         32
##   Number of equality constraints                    28
##
##   Number of observations per group
## 1                                                 2689
## 2                                                  137
## 3                                                 5491
## 4                                                   21
##
##   Estimator                                         ML
##   Model Fit Test Statistic                     447.241
##   Degrees of freedom                                16
##   P-value (Chi-square)                           0.000
##
## Chi-square for each group:
##
## 1                                            281.866
## 2                                             30.277
## 3                                            127.671
## 4                                              7.428

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

##              npar                  fmin                 chisq                    df
##             4.000                 0.027               447.241                16.000
##            pvalue        baseline.chisq           baseline.df       baseline.pvalue
##             0.000              2106.324                 4.000                 0.000
##               cfi                   tli                  nnfi                   rfi
##             0.795                 0.949                 0.949                    NA
##               nfi                  pnfi                   ifi                   rni
##                NA                 3.151                 0.794                 0.795
##              logl     unrestricted.logl                   aic                   bic
##        -65103.779            -64880.158            130215.557            130243.671
##            ntotal                  bic2                 rmsea        rmsea.ci.lower
##          8338.000            130230.960                 0.114                 0.105
##    rmsea.ci.upper           rmsea.pvalue                   rmr            rmr_nomean
##             0.123                 0.000                 9.992                12.765
##              srmr           srmr_bentler   srmr_bentler_nomean                  crmr
##             0.130                 0.130                 0.089                 0.178
##       crmr_nomean             srmr_mplus     srmr_mplus_nomean                 cn_05
##             0.026                 0.153                 0.083               491.246
##             cn_01                   gfi                  agfi                  pgfi
##           597.581                 0.999                 0.999                 0.799
##               mfi
##             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:
library(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)
dsGroupSummary

##      R Included PairCount      O1Mean       O2Mean O1Variance O2Variance O1O2Covariance
## 1 0.25     TRUE       280  0.04830041  0.056049671  1.0181579  1.1847214      0.2655632
## 2 0.50     TRUE      3894 -0.04986899 -0.027888465  0.9735617  1.0193789      0.4660286
## 3 1.00     TRUE        11 -0.08652006 -0.009039755  0.3171247  0.9518269      0.3583306
##   Correlation Determinant PosDefinite
## 1   0.2417977   1.1357096        TRUE
## 2   0.4678030   0.7752456        TRUE
## 3   0.6522137   0.1734469        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
##     0.7040149    0.1123658    0.1836193 4185.0000000

#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, Step 8 should change slightly. For instance, to MZ twins, replace the two lines in Step 8 with the following four. This is most for demonstration. It is unlikely to be useful idea in the current example, and is more likely to be useful when using the `RFull` variable, which includes all values of $R$ we were able to determine.

```
#Step 8: Summarize the R groups and determine which groups can be estimated.
dsGroupSummary <- RGroupSummary(dsSingle, oName_S1, oName_S2)
rGroupsToDrop <- c( 1 )
dsGroupSummary[dsGroupSummary$R %in% rGroupsToDrop, "Included"] <- FALSE
dsGroupSummary
```

# 6 Example: Multiple Group SEM of a Simple Outcome all pairs in Gen1 and Gen2

The example differs from the previous example in one way –all possible pairs are considered for the analysis. Pairs are only excluded (a) if they belong to one of the small $R$ groups that are difficult to estimate, or (b) if the value for adult height is missing. This includes all 42,836 relationships in the follow five types of NLSY79 relationships.

| | Relationship Frequency |
|---|---|
| Gen1Housemates | 5,302 |
| Gen2Siblings | 11,114 |
| Gen2Cousins | 5,000 |
| ParentChild | 11,521 |
| AuntNiece | 9,899 |

Table 1: Number of NLSY79 relationship, by `RelationshipPath`. (Recall that 'AuntNiece' also contains uncles and nephews.)

In our opinion, using the intergenerational links is one of the most exciting new opportunities for NLSY researchers to pursue. We will be happy to facilitate such research through consult or collaboration, or even by generating new data structures that may be of value. The complete kinship linking file facilitates many different kinds of cross-generational research, using both biometrical and other kinds of modeling methods.

```
### 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:
library(NlsyLinks)
dsLinking <- subset(Links79Pair, RelationshipPath %in%
                    c("Gen1Housemates", "Gen2Siblings", "Gen2Cousins",
                      "ParentChild", "AuntNiece"))
#Because all five paths are specified, the line above is equivalent to:
#dsLinking <- Links79Pair

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)
dsGroupSummary

##          R Included PairCount       O1Mean       O2Mean O1Variance O2Variance
## 1 0.0625     TRUE       202  0.22968753 -0.07575395  1.0509023  0.8271487
## 2 0.1250     TRUE      2422 -0.02213186  0.01502572  1.0334694  0.9809845
## 3 0.2500     TRUE      7136 -0.05441460 -0.03885252  1.0189426  1.0294526
## 4 0.3750     TRUE        48  0.13536238 -0.10786196  1.1497643  0.9269109
## 5 0.5000     TRUE     14862 -0.05738494 -0.01738686  0.9602559  0.9851045
## 6 0.7500    FALSE         0          NA          NA         NA         NA
## 7 1.0000     TRUE        27 -0.11092113 -0.12601206  0.6934418  0.9891146
##   O1O2Covariance Correlation Determinant PosDefinite
## 1      0.1112158   0.1192871   0.8568836        TRUE
## 2      0.1487539   0.1477368   0.9916897        TRUE
## 3      0.2721042   0.2656790   0.9749124        TRUE
## 4      0.4074779   0.3947123   0.8996908        TRUE
## 5      0.4146816   0.4263636   0.7739915        TRUE
## 6             NA          NA          NA       FALSE
## 7      0.6863894   0.8287857   0.2147630        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
## 7.366543e-01 6.580003e-02 1.975456e-01 2.469700e+04

#Step 11: Inspect the output further (see the final step two examples above).
```

Notice the ACE estimates are very similar to the previous version, but the number of pairs has increased by 6x –from 4,185 to 24,700. The number of *subjects* doubles when Gen2 is added, and the number of *relationship pairs* really takes off. When an extended family's entire pedigree is considered by the model, many more types of links are possible than if just nuclear families are considered. This increased statistical power is even more important when the population's $a^2$ is small or moderate, instead of something large like 0.7.

You may notice that the analysis has 24,697 relationships instead of the entire 42,836. This is primarily because not all subjects have a value for 'adult height' (and that's mostly because a lot of Gen2 subjects are too young). There are 42,088 pairs with a nonmissing value in `RFull`, meaning that 98.3% are classified. We feel comfortable claiming that if a researcher has a phenotype for both members of a pair, there's a 99+% chance we have an `RFull` for it. For a description of the `R` and `RFull` variables, please see the `Links79Pair` entry in the package reference manual.

**References:**

The standard errors (but not the coefficients) are biased downward in these analyses, because individuals are included in multiple pairs. Our MDAN article presents a GEE method for handling this (p. 572). The CARB model (or any model that treats the full pedigree as a single unit of analysis in the multivariate or

multilevel sense) also would produce more accurate standard error estimates.

One of our 2013 BGA presentations discusses these benefits in the context of the current NlsyLinks package, and our 2008 MDAN article accomplishes something similar using a GEE with females in both generations.

Bard, D.E., Beasley, W.H., Meredith, K., & Rodgers, J.L. (2012). *Biometric Analysis of Complex NLSY Pedigrees: Introducing a Conditional Autoregressive Biometric (CARB) Mixed Model*. Behavior Genetics Association 42nd Annual Meeting. [Slides]

Beasley, W.H., Bard, D.E., Meredith, K., Hunter, M., &Rodgers, J.L. (2013). *NLSY Kinship Links: Creating Biometrical Design Structures from Cross-Generational Data*. Behavior Genetics Association 43rd Annual Meeting. [Slides]

Rodgers, J. L., Bard, D., Johnson, A., D'Onofrio, B., & Miller, W. B. (2008). The Cross-Generational Mother-Daughter-Aunt-Niece Design: Establishing Validity of the MDAN Design with NLSY Fertility Variables. *Behavior Genetics, 38*, 567-578.

# 7  Example: Midstream data manipulation with SAS

*Mike Hunter is writing a separate vignette for this example. The SAS code is complete, and the surrounding text is the only thing remaining. Contact us if you're interested in a pre-release version. We'll post the polished version in the 'other-software' location in our online forums within the next 2-4 weeks.*

The example differs from the previous example inone 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.

# 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 most 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

A list of some articles that have used the NLSY for behavior genetics is available at: http://liveoak.github.io/NlsyLinks/research_publications.html.

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, J. L., Bard, D., Johnson, A., D'Onofrio, B., & Miller, W. B. (2008). The Cross-Generational Mother-Daughter-Aunt-Niece Design: Establishing Validity of the MDAN Design with NLSY Fertility Variables. *Behavior Genetics, 38*, 567-578.

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

# F  Version Information

- R version 3.5.1 Patched (2018-09-10 r75281), `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`

- Running under: `Windows >= 8 x64 (build 9200)`

- Matrix products: default

- Base packages: base, datasets, graphics, grDevices, methods, stats, utils

- Other packages: bindrcpp 0.2.2, knitr 1.20, lavaan 0.6-3, magrittr 1.5, NlsyLinks 2.0.8.9000, xtable 1.8-3

- Loaded via a namespace (and not attached): assertthat 0.2.0, backports 1.1.2, bindr 0.1.1, checkmate 1.8.5, cli 1.0.1, colorspace 1.3-2, compiler 3.5.1, crayon 1.3.4, devtools 1.13.6, digest 0.6.18, dplyr 0.7.7, evaluate 0.12, fansi 0.4.0, glue 1.3.0, highr 0.7, hms 0.4.2.9001, MASS 7.3-51, memoise 1.1.0, mnormt 1.5-5, munsell 0.5.0, packrat 0.4.9-3, pbivnorm 0.6.0, pillar 1.3.0, pkgconfig 2.0.2, purrr 0.2.5, R6 2.3.0, Rcpp 0.12.19, readr 1.2.0, rlang 0.2.2, rstudioapi 0.8, scales 1.0.0, stats4 3.5.1, stringi 1.2.4, stringr 1.3.1, tibble 1.4.2, tidyr 0.8.1, tidyselect 0.2.5, tools 3.5.1, utf8 1.1.4, withr 2.1.2