ACE Models with the NLSY

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

November 13, 2012

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 SubjectTags 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 .

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 F:/Projects/RLibraries/NlsyLinks/help/NlsyLinks-package in 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 CreatePair-LinksDoubleEntered 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 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 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
data(Links79Pair)
dsLinking <- subset(Links79Pair, RelationshipPath == "Gen2Siblings")
summary(dsLinking)  #Notice there are 11,075 records.

##     ExtendedID     Subject1Tag       Subject2Tag            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.
data(ExtraOutcomes79)
dsOutcomes <- ExtraOutcomes79  #'ds' stands for 'DataSet'
summary(dsOutcomes)

##    SubjectTag        SubjectID        Generation   AfqtRescaled2006Gaussified
##  Min.   :    100   Min.   :       1  Min.   :1.00   Min.   :-3
##  1st Qu.: 314025   1st Qu.:    5998  1st Qu.:1.00   1st Qu.:-1
##  Median : 620050   Median :   12000  Median :1.00   Median : 0
##  Mean   : 618600   Mean   :  289254  Mean   :1.48   Mean   : 0
##  3rd Qu.: 914501   3rd Qu.:  577403  3rd Qu.:2.00   3rd Qu.: 0
##  Max.   :1268600   Max.   : 1267501  Max.   :2.00   Max.   : 3
##                                                     NA's   :12276
##  HeightZGender  HeightZGenderAge WeightZGender   WeightZGenderAge
##  Min.   :-3     Min.   :-3       Min.   :-3      Min.   :-4
##  1st Qu.:-1     1st Qu.:-1       1st Qu.:-1      1st Qu.:-1
##  Median : 0     Median : 0       Median : 0      Median : 0
##  Mean   : 0     Mean   : 0       Mean   : 0      Mean   : 0
##  3rd Qu.: 1     3rd Qu.: 1       3rd Qu.: 0      3rd Qu.: 0
##  Max.   : 3     Max.   : 3       Max.   : 8      Max.   : 7
##  NA's   :5029   NA's   :5030     NA's   :12104   NA's   :12105
##      Afi            Afm         MathStandardized
##  Min.   : 2     Min.   : 0     Min.   : 65
##  1st Qu.:15     1st Qu.:12     1st Qu.: 91
##  Median :17     Median :13     Median :100
##  Mean   :17     Mean   :13     Mean   :100
##  3rd Qu.:18     3rd Qu.:14     3rd Qu.:110
##  Max.   :27     Max.   :19     Max.   :135
##  NA's   :12740  NA's   :18165  NA's   :15048

# 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 22,150=(2*11,075) records.

##   Subject1Tag        Subject2Tag        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_1 MathStandardized_2
##  Gen1Housemates:    0  Min.   : 65        Min.   : 65
##  Gen2Siblings  :22176  1st Qu.: 89        1st Qu.: 89
##  Gen2Cousins   :    0  Median : 98        Median : 98
##  ParentChild   :    0  Mean   : 98        Mean   : 98
```

4

```
##  AuntNiece    :    0    3rd Qu.:108        3rd Qu.:108
##                         Max.   :135        Max.   :135
##                         NA's   :3815       NA's   :3815
# Step 7: Estimate the ACE components with a DF Analysis
ace <- DeFriesFulkerMethod3(dataSet = dsDouble, oName_1 = "MathStandardized_1", oName_2 = "MathStandardi
ace

## [1] "Results of ACE estimation: [show]"
##  ASquared  CSquared  ESquared CaseCount
## 8.480e-01 4.477e-02 1.073e-01 1.678e+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 and create a new script (see Appendix B) and load the NlsyLinks package.

3. Within the script, load the linking dataset. Then select only Gen2 subjects.

4. Read the CSV into 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
# data(Links79Pair)
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'
```

```r
filePathOutcomes <- file.path(path.package("NlsyLinks"), "extdata", "Gen2Birth.csv")
dsOutcomes <- ReadCsvNlsy79Gen2(filePathOutcomes)
summary(dsOutcomes)

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

# Step 5: Verify and rename an existing column.
VerifyColumnExists(dsOutcomes, "C0328600")  #Should return '11' 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_1 = "BirthWeightInOunces
    oName_2 = "BirthWeightInOunces_2")
ace

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

6

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 manifest 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_1' corresponds to the first members and ZZZ_2' 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)
data(Links79Pair)
dsLinking <- subset(Links79Pair, RelationshipPath == "Gen2Siblings")
data(ExtraOutcomes79)
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 manifest variables.
oName_1 <- "MathStandardized_1"  #Stands for Manifest1
oName_2 <- "MathStandardized_2"  #Stands for Manifest2
# Step 8: Summarize the R groups and determine which groups can be estimated.
dsGroupSummary <- RGroupSummary(dsSingle, oName_1, oName_2)
dsGroupSummary
```

```
##        R Included PairCount O1Variance O2Variance O1O2Covariance Correlation
## 1 0.250    TRUE      2718      169.7      207.8         41.08      0.2188
## 2 0.375    TRUE       139      172.5      187.1         40.48      0.2253
## 3 0.500    TRUE      5511      230.5      233.0        107.37      0.4633
## 4 0.750   FALSE         2      220.5       18.0         63.00      1.0000
## 5 1.000    TRUE        22      319.2      343.1        277.59      0.8388
```

7

```
##    Determinant PosDefinite
## 1        33573        TRUE
## 2        30639        TRUE
## 3        42172        TRUE
## 4            0       FALSE
## 5        32466        TRUE

# Step 9: Create a cleaned dataset
dsClean <- CleanSemAceDataset(dsDirty = dsSingle, dsGroupSummary, oName_1, oName_2)
# Step 10: Run the model
ace <- AceLavaanGroup(dsClean)
ace

## [1] "Results of ACE estimation: [show]"
##   ASquared  CSquared  ESquared CaseCount
##     0.6673    0.1188    0.2139 8390.0000

# Notice the `CaseCount' is 8,292 instead of 16,588.  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.

## Loading required package:   lavaan

## Loading required package:   MASS

## Loading required package:   boot

## Loading required package:   mnormt

## Loading required package:   quadprog

## This is lavaan 0.5-10

## lavaan is BETA software!  Please report any bugs.

GetDetails(ace)

## lavaan (0.5-10) converged normally after   60 iterations
##
##    Number of observations per group
## 1                                                2718
## 2                                                 139
## 3                                                5511
## 4                                                  22
##
##    Estimator                                        ML
##    Minimum Function Chi-square                 447.272
##    Degrees of freedom                               16
##    P-value                                       0.000
##
## Chi-square for each group:
##
## 1                                            282.194
## 2                                             32.858
## 3                                            127.880
## 4                                              4.339
```

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

##             chisq                df            pvalue    baseline.chisq
##           447.272            16.000             0.000          1499.091
##       baseline.df    baseline.pvalue               cfi               tli
##             4.000             0.000             0.712             0.928
##              logl  unrestricted.logl              npar               aic
##        -68369.132        -68145.496             4.000        136746.264
##               bic            ntotal              bic2             rmsea
##        136774.404          8390.000        136761.692             0.113
##    rmsea.ci.lower    rmsea.ci.upper      rmsea.pvalue              srmr
##             0.104             0.123             0.000             0.128
##       srmr_nomean
##             0.093


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

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

RStudio allows you to create and save 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 wayto **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.

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, 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 (new versions of 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 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 the CRAN servers 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

F   Version Information

- R version 2.15.2 (2012-10-26), `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: boot 1.3-7, knitr 0.8, lavaan 0.5-10, MASS 7.3-22, mnormt 1.4-5, NlsyLinks 1.015, quadprog 1.5-4
- Loaded via a namespace (and not attached): digest 0.5.2, evaluate 0.4.2, formatR 0.6, plyr 1.7.1, stats4 2.15.2, stringr 0.6.1, tools 2.15.2