Package 'NlsyLinks'

June 6, 2017

Type Package

Title Utilities and Kinship Information for Research with the NLSY

Version 2.0.7.9000

Date 2017-05-26

Description Utilities and kinship information for behavior genetics and developmental research using the National Longitudinal Survey of Youth (NLSY; <http://www.bls.gov/nls/>).

URL http://liveoak.github.io/NlsyLinks, https://github.com/LiveOak/NlsyLinks, https://r-forge.r-project.org/projects/nlsylinks

BugReports https://github.com/LiveOak/NlsyLinks/issues

Depends R (>= 3.0.0), stats

Imports lavaan, methods

Suggests devtools, ggplot2, knitr, plyr, scales, testit, testthat, xtable

License GPL

LazyData TRUE

VignetteBuilder knitr

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

R topics documented:

`NlsyLinks-package` *Utilities and kinship information for Behavior Genetics and Developmental research using the NLSY.*

Description

Utilities and kinship information for Behavior Genetics and Developmental research using the NLSY.

Note

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 children of the Gen1 females -ie, those in the NLSY79 Children and Young Adults sample (http://www.bls.gov/nls/nlsy79ch.htm).

The release version is available through CRAN by running `install.packages('NlsyLinks')`. The most recent development version is available through GitHub by running `devtools::install_github (repo = 'LiveOak/NlsyLinks')` (make sure devtools is already installed). If you're having trouble with the package, please install the development version. If this doesn't solve your problem, please create a new issue, or email Will.

Author(s)

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)

Maintainer: Will Beasley wibeasley@hotmail.com

References

This package's development was largely supported by the NIH Grant 1R01HD65865, "NLSY Kinship Links: Reliable and Valid SiblingIdentification" (PI: Joe Rodgers). A more complete list of research articles using NLSY Kinship Links is maintained on our package'swebsite.

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

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

D'Onofrio, B.M., Van Hulle, C.A., Waldman, I.D., Rodgers, J.L., Rathouz, P.J., & Lahey, B.B. (2007). Causal inferences regarding prenatal alcoholexposure and childhood externalizing problems.. *Archives of General Psychiatry, 64*, 1296-1304.

Rodgers, J.L. & Doughty, D. (2000). Genetic and environmental influenceson fertility expectations and outcomes using NLSY kinship data. In J.L. Rodgers, D. Rowe, & W.B. Miller (Eds.) *Genetic influences on fertility and sexuality.* Boston: Kluwer Academic Press.

Cleveland, H.H., Wiebe, R.P., van den Oord, E.J.C.G., & Rowe, D.C. (2000). Behavior problems among children from different family structures: Theinfluence of genetic self-selection. *Child Development, 71*, 733-751.

Rodgers, J.L., Rowe, D.C., & Buster, M. (1999). Nature, nurture, and firstsexual intercourse in the USA: Fitting behavioural genetic models to NLSYkinship data. *Journal of Biosocial Sciences, 31*.

Rodgers, J.L., Rowe, D.C., & Li, C. (1994). Beyond nature versus nurture:DF analysis of nonshared influences on problem behaviors. *Developmental Psychology, 30*, 374-384.

Examples

```
library(NlsyLinks)    # Load the package into the current R session.
summary(Links79Pair)  # Summarize the five variables.
hist(Links79Pair$R)   # Display a histogram of the Relatedness values.
table(Links79Pair$R)  # Create a table of the Relatedness values for the whole sample.

## Not run:
# Install/update NlsyLinks with the release version from CRAN.
install.packages('NlsyLinks')

# Install/update NlsyLinks with the development version from GitHub
#install.packages('devtools') #Uncomment if 'devtools' isn't installed already.
devtools::install_github('LiveOak/NlsyLinks')
```

```
## End(Not run)
```

Ace *Estimates the heritability of additive traits using a single variable.*

Description

An ACE model is the foundation of most behavior genetic research. It estimates the additive heritability (with *a*), common environment (with *c*) and unshared heritability/environment (with *e*).

Usage

```
AceUnivariate(
  method         = c("DeFriesFulkerMethod1","DeFriesFulkerMethod3"),
  dataSet,
  oName_S1,
  oName_S2,
  rName          = "R",
  manifestScale  = "Continuous"
)

DeFriesFulkerMethod1(dataSet, oName_S1, oName_S2, rName="R")

DeFriesFulkerMethod3(dataSet, oName_S1, oName_S2, rName="R")
```

Arguments

`method`	The specific estimation technique.
`dataSet`	The base::data.frame that contains the two outcome variables and the relatedness coefficient (corresponding to `oName_S1`, `oName_S2`, and `rName`)
`oName_S1`	The name of the outcome variable corresponding to the first subject in the pair. This should be a `character` value.
`oName_S2`	The name of theoutcome variable corresponding to the second subject in the pair. This should be a `character` value.
`rName`	The name of the relatedness coefficient for the pair (this is typically abbreviated as `R`). This should be a `character` value.
`manifestScale`	Currently, only *continuous* manifest/outcome variables are supported.

Details

The `AceUnivariate()` function is a wrapper that calls `DeFriesFulkerMethod1()` or `DeFriesFulkerMethod3()`. Future versions will incorporate methods that use latent variable models.

Value

Currently, a list is returned with the arguments `ASquared`, `CSquared`, `ESquared`, and `RowCount`. In the future, this may be changed to an `S4` class.

Author(s)

Will Beasley

References

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

Examples

```
library(NlsyLinks) #Load the package into the current R session.
dsOutcomes <- ExtraOutcomes79
dsOutcomes$SubjectTag <- CreateSubjectTag(
  subjectID    = dsOutcomes$SubjectID,
  generation   = dsOutcomes$Generation
)
dsLinks  <- Links79Pair
dsLinks  <- dsLinks[dsLinks$RelationshipPath=='Gen2Siblings', ] #Only Gen2 Sibs (ie, NLSY79C)
dsDF     <- CreatePairLinksDoubleEntered(
  outcomeDataset     = dsOutcomes,
  linksPairDataset   = dsLinks,
  outcomeNames       = c("MathStandardized", "HeightZGenderAge", "WeightZGenderAge")
)

estimatedAdultHeight <- DeFriesFulkerMethod3(
  dataSet    = dsDF,
  oName_S1   = "HeightZGenderAge_S1",
  oName_S2   = "HeightZGenderAge_S2"
)
estimatedAdultHeight #ASquared and CSquared should be 0.60 and 0.10 for this rough analysis.

estimatedMath <- DeFriesFulkerMethod3(
  dataSet    = dsDF,
  oName_S1   = "MathStandardized_S1",
  oName_S2   = "MathStandardized_S2"
)
estimatedMath #ASquared and CSquared should be 0.85 and 0.045.

class(GetDetails(estimatedMath))
summary(GetDetails(estimatedMath))
```

AceEstimate-class *Class AceEstimate*

Description

A class containing information about a single univariate ACE model.

Objects from the Class

Objects can be created by calls of the form: `new("AceEstimate", aSquared, cSquared, eSquared, caseCount, unity,`

Note

The contents of the `Details` list depends on the underlying estimation routine. For example, when the ACE model is estimated with a DF analysis, the output is an `stats::lm()` object, because the `stats::lm()` function was used (ie, the basical general linear model). Alternatively, if the user specified the `lavaan::lavaan()` package should estimate that ACE model, the output is a `lavaan::lavaan()` object.

Examples

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

showClass("AceEstimate")
est <- CreateAceEstimate(.5, .2, .3, 40)
est
print(est)
```

AceLavaanGroup *A simple multiple-group ACE model with the* **lavaan** *package.*

Description

This function uses the **lavaan** package to estimate a univariate ACE model, using multiple groups. Each group has a unique value of `R` (i.e., the *R*elatedness coefficient).

Usage

```
AceLavaanGroup(dsClean, estimateA = TRUE, estimateC = TRUE,
  printOutput = FALSE)
```

Arguments

dsClean	The base::data.frame containing complete cases for the R groups to be included in the estimation.
estimateA	Should the *A* variance component be estimated? A^2 represents the proportion of variability due to a shared genetic influence.
estimateC	Should the *C* variance component be estimated? C^2 represents the proportion of variability due to a shared environmental influence.
printOutput	Indicates if the estimated parameters and fit statistics are printed to the console.

Details

The variance component for *E* is always estimated, while the *A* and *C* estimates can be fixed to zero (when estimateA and/or *estimateC* are set to FALSE).

Value

An AceEstimate object.

Note

Currently, the variables in dsClean must be named O1, O2 and R; the letter 'O' stands for *O*utcome. This may not be as restrictive as it initially seems, because dsClean is intented to be produced by CleanSemAceDataset(). If this is too restrictive for your uses, we'd like to here about it (*please email wibeasley at hotmail period com*).

Author(s)

Will Beasley

References

The **lavaan** package is developed by Yves Rosseel at Ghent University. Three good starting points are the package website (http://lavaan.ugent.be/), the package documentation (https://cran.r-project.org/package=lavaan) and the JSS paper.

Rosseel, Yves (2012), lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software, 48*, (2), 1-36.

See Also

CleanSemAceDataset(). Further ACE model details are discussed in our package's vignettes.

Examples

```
library(NlsyLinks) # Load the package into the current R session.
dsLinks  <- Links79PairExpanded #Start with the built-in data.frame in NlsyLinks
dsLinks  <- dsLinks[dsLinks$RelationshipPath=='Gen2Siblings', ] # Use only Gen2 Siblings (NLSY79-C)

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

```
dsGroupSummary <- RGroupSummary(dsLinks, oName_S1, oName_S2)
dsClean <- CleanSemAceDataset(dsDirty=dsLinks, dsGroupSummary, oName_S1, oName_S2)

ace <- AceLavaanGroup(dsClean)
ace

#Should produce:
# [1] "Results of ACE estimation: [show]"
#     ASquared     CSquared     ESquared    CaseCount
#    0.6681874    0.1181227    0.2136900 8390.0000000

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

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

#Examine low-level details like each group's individual parameter estimates and standard errors.
summary(GetDetails(ace))

#Extract low-level details. This may be useful when programming simulations.
inspect(GetDetails(ace), what="converged") #The lavaan package defines 'inspect'.
inspect(GetDetails(ace), what="coef")
```

`CleanSemAceDataset` *Produces a cleaned dataset that works well with when using SEM to estimate a univariate ACE model.*

Description

This function takes a 'GroupSummary' base::data.frame (which is created by the `RGroupSummary()` function) and returns a base::data.frame that is used by the `Ace()` function.

Usage

```
CleanSemAceDataset(dsDirty, dsGroupSummary, oName_S1, oName_S2, rName = "R")
```

Arguments

`dsDirty`	This is the base::data.frame to be cleaned.
`dsGroupSummary`	The base::data.frame containing information about which groups should be included in the analyses. It should be created by the `RGroupSummary()` function.
`oName_S1`	The name of the manifest variable (in `dsDirty`) for the first subject in each pair.
`oName_S2`	The name of the manifest variable (in `dsDirty`) for the second subject in each pair.
`rName`	The name of the variable (in `dsDirty`) indicating the pair's relatedness coefficient.

Details

The function takes dsDirty and produces a new base::data.frame with the following features:

1. Only three existing columns are retained: O1, O2, and R. They are assigned these names.
2. A new column called GroupID is created to reflect their group membership (which is based on the R value). These valuesa re sequential integers, starting at 1. The group with the weakest R is 1. The group with the strongest R has the largest GroupID (this is typically the MZ tiwns).
3. Any row is excluded if it has a missing data point for O1, O2, or R.
4. The base::data.frame is sorted by the R value. This helps program against the multiple-group SEM API sometimes.

Value

A base::data.frame with one row per subject pair. The base::data.frame contains the following variables (which can NOT be changed by the user through optional parameters):

- **R** The pair's R value.
- **O1** The outcome variable for the first subject in each pair.
- **O2** The outcome variable for the second subject in each pair.
- **GroupID** Indicates the pair's group membership.

Author(s)

Will Beasley

Examples

```
library(NlsyLinks) #Load the package into the current R session.
dsLinks        <- Links79PairExpanded #Start with the built-in data.frame in NlsyLinks
dsLinks      <- dsLinks[dsLinks$RelationshipPath=='Gen2Siblings', ] #Use only NLSY79-C siblings

oName_S1       <- "MathStandardized_S1" #Stands for Outcome1
oName_S2       <- "MathStandardized_S2" #Stands for Outcome2
dsGroupSummary <- RGroupSummary(dsLinks, oName_S1, oName_S2)

dsClean <- CleanSemAceDataset( dsDirty=dsLinks, dsGroupSummary, oName_S1, oName_S2, rName="R" )
summary(dsClean)

dsClean$AbsDifference <- abs(dsClean$O1 - dsClean$O2)
plot(jitter(dsClean$R), dsClean$AbsDifference, col="gray70")
```

`ColumnUtilities` *A collection of functions that helps data management data.frames, particularly those derived from NLSY Extracts.*

Description

A collection of functions that helps data management base::data.frames, particularly those derived from NLSY Extracts.

Usage

```
VerifyColumnExists( dataFrame, columnName )

RenameColumn( dataFrame, oldColumnName, newColumnName )

RenameNlsyColumn( dataFrame, nlsyRNumber, newColumnName )
```

Arguments

`dataFrame`	The base::data.frame whose columns are to be verified or renamed.
`columnName`	The name of the column to verify is present in the base::data.frame.
`nlsyRNumber`	The name of the column to change.
`oldColumnName`	The name of the column to change.
`newColumnName`	The desired name of the column.

Details

The RNumber assigned by the NLS has a pattern. In the Nlsy79 Gen1 dataset, the names start with a 'R' or 'T' and are followed by seven digits (eg, R0000100). In the Nlsy79 Gen2 dataset, the names start with 'C' or 'Y' and are followed by seven digits (eg, C0007030, Y1994600).

In the NLS Investigator, a decimal is present in the RNumber (eg, R00001.00). When the Investigator saves the dataset as a CSV, the decimal is removed (eg, R0000100).

Value

IMPORTANT The `RenameColumn()` and `RenameNlsyColumn()` functions do not use side-effects to rename the base::data.frame. Instead, it returns a new base::data.frame. In the example below, notice the assignment to `ds`: `ds <- RenameNlsyColumn(...)`.

The `VerifyColumnExists()` function check that exactly one column exists with the specified `columnName`. If so, the index of the column is returned. If not, an exception is thrown.

Author(s)

Will Beasley

`CreateAceEstimate` *Instantiate an AceEstimate object.*

Description

Creates an instance of the `S4` class, AceEstimate instantiating arguments set the parameter values estimated by the ACE model.

Usage

```
CreateAceEstimate(aSquared, cSquared, eSquared, caseCount, details = list(),
  unityTolerance = 1e-11)
```

Arguments

`aSquared`	The proportion of variability due to a shared genetic influence (typically represented as a^2, or sometimes h^2).
`cSquared`	The proportion of variability due to shared common environmental influence.
`eSquared`	The proportion of variability due to unexplained/residual/error influence.
`caseCount`	The number of cases used to estimate the model.
`details`	A `list` that contains the modeling output and details.
`unityTolerance`	Specifies how close the the sum of the ACE components should be to one, to be considered properly scaled to one.

Details

The contents of the `details` list depends on the underlying estimation routine. For example, when the ACE model is estimated with a DF analysis, the output is a stats::lm object, because the stats::lm function was used (ie, the basical general linear model). Alternatively, if the user specified the `lavaan` package should estimate that ACE model, the output is a lavaan::lavaan object.

Value

An S4 object of AceEstimate.

Author(s)

Will Beasley

CreatePairLinks *Creates a pairs linking file.*

Description

Creates a linking file for BG designs using this file structure (e.g., DF analysis, other ACE modeling). A DF analysis requires a double-entered file that contains the R value for the pair, and their two outcome variable values.

CreatePairLinksDoubleEnteredWithNoOutcomes() is intended to be a primarily a helper function for CreateSpatialNeighbours().

Usage

```
CreatePairLinksDoubleEntered(
  outcomeDataset,
  linksPairDataset,
  outcomeNames,
  linksNames             = c("ExtendedID", "R", "RelationshipPath"),
  validateOutcomeDataset = TRUE,
  subject1Qualifier      = "_S1",
  subject2Qualifier      = "_S2"
)

CreatePairLinksSingleEntered(
  outcomeDataset,
  linksPairDataset,
  outcomeNames,
  linksNames             = c("ExtendedID", "R", "RelationshipPath"),
  validateOutcomeDataset = TRUE,
  subject1Qualifier      = "_S1",
  subject2Qualifier      = "_S2"
)

CreatePairLinksDoubleEnteredWithNoOutcomes(
  linksPairDataset,
  linksNames = c("ExtendedID", "R", "RelationshipPath")
)
```

Arguments

outcomeDataset	A data frame containing the outcome variable(s)
linksPairDataset	A data frame containing the SubjectTags of each subject in the pair and their R coefficient.
outcomeNames	The column names of the outcome variable(s)

`linksNames` The column names desired to be prseent in the newly created data frame. `SubjectTag_S1` and `SubjectTag_S2` are included automatically.

`validateOutcomeDataset`
Indicates if characteristics of the outcomeDataset should be validated.

`subject1Qualifier`
Indicates how the outcome variable for the pair's first subject is distinguished from the other subject. The default is `_S1`.

`subject2Qualifier`
Indicates how the outcome variable for the pair's second subject is distinguished from the other subject. The default is `_S2`.

Author(s)

Will Beasley

References

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

Examples

```
dsSingleLinks <- data.frame(
  ExtendedID=c(1, 1, 1, 2),
  SubjectTag_S1=c(101, 101, 102, 201),
  SubjectTag_S2=c(102, 103, 103, 202),
  R=c(.5, .25, .25, .5),
  RelationshipPath=rep("Gen2Siblings", 4)
)
dsSingleOutcomes <- data.frame(
  SubjectTag = c(101, 102, 103, 201, 202),
  DV1        = c( 11,  12,  13,  41,  42),
  DV2        = c( 21,  22,  23,  51,  52)
)
dsDouble <- CreatePairLinksDoubleEntered(
  outcomeDataset         = dsSingleOutcomes,
  linksPairDataset       = dsSingleLinks,
  outcomeNames           = c("DV1", "DV2"),
  validateOutcomeDataset = TRUE
)
dsDouble #Show the 8 rows in the double-entered pair links
summary(dsDouble) #Summarize the variables

ValidatePairLinksAreSymmetric(dsDouble) #Should return TRUE.
```

`CreateSpatialNeighbours` *Distances between related family members, formated for spatial analysis.*

Description

This helper function formats the LinksPair datasets so it can be used in some types of spatial analyses. The **spdep** (Spatial Dependence) uses a sparse matrix (actually a base::data.frame) to represent neigbours.

Usage

```
CreateSpatialNeighbours(linksPairsDoubleEntered)
```

Arguments

`linksPairsDoubleEntered`
: A base::data.frame containing the links, preferably created by a function like `CreatePairLinksDoubleEntered()`.

Details

There is one row per unique pair of subjects, *respecting order*. This has twice as many rows as Links79Pair and Links79PairExpanded (which have one row per unique pair of subjects, *irrespective of order*).

`CreateSpatialNeighbours()` accepts any paired relationships in a base::data.frame, as long as it contains the columns `SubjectTag_S1`, `SubjectTag_S2`, and `R`. See Links79Pair for more details about these columns.

Value

An S3 `spdep::spatial.neighbours` object to work with functions in the **spdep** package.

`SubjectTag_S1` is renamed '`from`'.

`SubjectTag_S2` is renamed '`to`'.

`R` is renamed '`weight`'.

The attribute `region.id` specifies each unique SubjectTag.

The attribue `n` specifies the number of unique subjects.

Note

Notice the British variant of 'neighbo*u*rs' is used, to be consistent with the `spdep::spatial.neighbour` class.

Author(s)

Will Beasley and David Bard

References

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.

Bivand, R., Pebesma, E., & Gomez-Rubio, V. (2013). Applied Spatial Data Analysis with R. New York: Springer. (Especially Chapter 9.)

Banerjee, S., Carlin, B.P., & Gelfand, A.E. (2004). Hierarchical Modeling and Analysis for Spatial Data. Boca Raton: CRC Press.

Lawson, A.B (2013). Bayesian Disease Mapping: Hierarchical Modeling in Spatial Epidemiology, Second Edition. Boca Raton: CRC Press.

The **spdep** package documentation: spdep: Spatial dependence: weighting schemes, statistics and models.

Examples

```
dsLinksAll             <- Links79Pair
dsLinksGen1Housemates  <- dsLinksAll[dsLinksAll$RelationshipPath=="Gen1Housemates", ]
dsLinksGen2Siblings    <- dsLinksAll[dsLinksAll$RelationshipPath=="Gen2Siblings", ]

spGen1 <- CreateSpatialNeighbours(dsLinksGen1Housemates)
spGen2 <- CreateSpatialNeighbours(dsLinksGen2Siblings)

head(spGen2)
#Returns:
#   from  to weight
# 3  201 202   0.50
# 6  301 302   0.50
# 7  301 303   0.50
# 9  302 303   0.50
#24  401 403   0.25
#28  801 802   0.50

table(spGen2$weight)
#Returns:
#0.25 0.375   0.5  0.75     1
#3442   610  6997    12    27
```

`CreateSubjectTag` *Creates a SubjectTag. This value uniquely identifies subjects, when both generations are included in the same dataset.*

Description

A SubjectTag uniquely identify subjects. For Gen2 subjects, the SubjectTag is identical to their CID (ie, C00001.00 -the SubjectID assigned in the NLSY79-Children files). However for Gen1 subjects, the SubjectTag is their CaseID (ie, 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.

Usage

```
CreateSubjectTag(subjectID, generation)
```

Arguments

subjectID	The ID assigned by the NLSY. For Gen1 subjects, this will be their CaseID (ie, R00001.00). For Gen2 subjects, this will be their CID (ie, C00001.00).
generation	The generation of the subject. Values are either 1 or 2, representing Gen1 and Gen2.

Details

For a fuller explanation of SubjectTag in context, see the Links79Pair dataset documentation.

Value

A integer value under normal circumstances. An error is thrown if the vectors subjectID and generation are different lengths. If either input vector has NA values, the respective output element(s) will be NA too.

Author(s)

Will Beasley

See Also

Links79Pair

Examples

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

#Typically these two vectors will come from a data frame.
subjectIDs <- c(71:82, 10001:10012)
generation <- c(rep(1, 12), rep(2, 12))

CreateSubjectTag(subjectIDs, generation)
#Returns 7100, ..., 8200, 10001, ..., 10012

# Use the ExtraOutcomes79 dataset, with numeric variables 'SubjectID' and 'Generation'.
ExtraOutcomes79$SubjectTag <- CreateSubjectTag(
   subjectID=ExtraOutcomes79$SubjectID,
```

```
    generation=ExtraOutcomes79$Generation
)
```

`ExtraOutcomes79` *Extra outcome variables in the NLSY79*

Description

This dataset is provided primarily to facilitate documentation examples.

Format

A data frame with 11,495 observations on the following 6 variables. There is one row per subject.

- **SubjectTag** The ID value assigned by NLS to the first subject. For Gen1 Subjects, this is their "CaseID" (ie, R00001.00). For Gen2 subjects, this is their "CID" (ie, C00001.00).
- **SubjectID** The ID value assigned by NLS to the first subject. For Gen1 Subjects, this is their "CaseID" (ie, R00001.00). For Gen2 subjects, this is their "CID" (ie, C00001.00).
- **Generation** The generation of the subject. Values are either 1 or 2, representing Gen1 and Gen2. Note that this variable is not a `factor` (in constrast with data frames like Links79Pair). This dataset is supposed to mimick the dataset provided by the researcher, which typically will not have been converted to a `factor`.
- **HeightZGenderAge** The subject's height, standardized by gender and age (see Details).
- **WeightZGenderAge** The subject's weight, standardized by gender and age (see Details).
- **AfqtRescaled2006Gaussified** Armed Forces Qualification Test Score (Gen1 only; see Details).
- **Afi** Self-reported age of first intercourse (Gen1 only; see Details).
- **Afm** Self-reported age of first menstration (Gen1 only; see Details).
- **MathStandardized** Standardized PIAT Math scores (Gen2 only; see Details).

Details

The `SubjectTag` variable uniquely identify subjects. For Gen2 subjects, the SubjectTag is identical to their CID (ie, C00001.00 -the SubjectID assigned in the NLSY79-Children files). However for Gen1 subjects, the SubjectTag is their CaseID (ie, 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 has a `SubjectTag` of 4300. The SubjectTags of her four children remain 4301, 4302, 4303, and 4304.

For Gen2, an NLSY79 variable of `MathStandardized` is C05801.00.

`Afi` and `Afm`, values were simplified (to one value per subject) by Kelly Meredith in Sept 2010.

The variables for height and weight were manipulated in R files available in a repository available to the public. Find the appropriate subfolder, and view the HTML report for more details.

Author(s)

Will Beasley

Source

Gen1 information comes from the Summer 2013 release of the NLSY79 sample. Gen2 information comes from the Summer 2013 release of the NLSY79 Children and Young Adults sample. Data were extracted with the NLS Investigator (https://www.nlsinfo.org/investigator/).

Examples

```
library(NlsyLinks) #Load the package into the current R session.
gen2Outcomes <- subset(ExtraOutcomes79, Generation==2) #Create a dataset of only Gen2 subjects.

#plot(ExtraOutcomes79) #Uncomment to see a large scatterplot matrix.
summary(ExtraOutcomes79)

oldPar <- par(mfrow=c(3,2))
hist(ExtraOutcomes79$Generation)
hist(ExtraOutcomes79$MathStandardized)
hist(ExtraOutcomes79$HeightZGenderAge)
hist(ExtraOutcomes79$WeightZGenderAge)
hist(ExtraOutcomes79$Afi)
hist(ExtraOutcomes79$Afm)
par(oldPar)
```

GetDetails-methods *A generic function for extracting the* Details *slot of an object.*

Description

A generic function for extracting the `Details` slot of an AceEstimation object.

Author(s)

Will Beasley

Links79Pair *Kinship linking file for pairs of relatives in the NLSY79 and NLSY79 Children and Young Adults*

Description

This dataset specifies the relatedness coefficient (ie, 'R') between subjects in the same extended family. Each row represents a unique relationship pair.

NOTE: Two variable names changed in November 2013. `Subject1Tag` and `Subject2Tag` became `SubjectTag_S1` and `SubjectTag_S2`.

Format

A data frame with 42,773 observations on the following 5 variables. There is one row per unique pair of subjects, irrespective of order.

- **ExtendedID** Identity of the extended family of the pair; it corresponds to the HHID in the NLSY79. See References below.
- **SubjectTag_S1** Identity of the pair's first subject. See Details below.
- **SubjectTag_S2** Identity of the pair's second subject. See Details below.
- **R** The pair's Relatedness coefficient. See Details below.
- **RelationshipPath** Specifies the relationship category of the pair. This variable is a factor, with levels `Gen1Housemates=1`, `Gen2Siblings=2`, `Gen2Cousins=3`, `ParentChild=4`, `AuntNiece=5`.

Details

The dataset contains 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 children of the Gen1 females -ie, those in the NLSY79 Children and Young Adults sample (http://www.bls.gov/nls/nlsy79ch.htm).

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 (ie, 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).

The variables `SubjectTag_S1` and `SubjectTag_S2` uniquely identify subjects. For Gen2 subjects, the SubjectTag is identical to their CID (ie, C00001.00 -the SubjectID assigned in the NLSY79-Children files). However for Gen1 subjects, the SubjectTag is their CaseID (ie, 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 has a `SubjectTag` of 4300. The SubjectTags of her four children remain 4301, 4302, 4303, and 4304.

Level 5 of RelationshipPath (ie, AuntNiece) is gender neutral. The relationship could be either Aunt-Niece, Aunt-Nephew, Uncle-Niece, or Uncle-Nephew. If there's a widely-accepted gender-neutral term, please tell me.

An extended family with k subjects will have $k(k\text{-}1)/2$ rows. Typically, Subject1 is older while Subject2 is younger.

MZ twins have *R*=1. DZ twins and full-siblings have *R*=.5. Half-siblings have *R*=.25. Typical first cousins have *R*=.125. Unrelated subjects have *R*=0 (this occasionally happens for Gen1Housemates). Other *R* coefficients are possible.

There are several other uncommon possibilities, such as half-cousins (*R*=.0625) and ambiguous aunt-nieces (*R*=.125). The variable coding for genetic relatedness,R, in Links79Pair contains only the common values of *R* whose groups are likely to have stable estimates. However the variable RFull in Links79PairExpanded contains all *R* values. We strongly recommend using R in this base::data.frame. Move to RFull (or some combination) only if you have a good reason, and are willing to carefully monitor a variety of validity checks. Some of these excluded groups are too small to be estimated reliably.

Furthermore, some of these groups have members who are more strongly genetically related than their items would indicate. For instance, there are 41 Gen1 pairs who explicitly claim they are not biologically related (*ie*, RExplicit=0), yet their correlation for Adult Height is *r*=0.24. This is much higher than would be expected for two people sampled randomly; it is nearly identical to the *r*=0.26 we observed among the 268 Gen1 half-sibling pairs who claim they share exactly 1 biological parent.

Author(s)

Will Beasley

Source

Gen1 information comes from the Summer 2013 release of the NLSY79 sample. Gen2 information comes from the Summer 2013 release of the NLSY79 Children and Young Adults. Data were extracted with the NLS Investigator (https://www.nlsinfo.org/investigator/).

The internal version for the links is Links2011V84.

References

The NLSY79 variable HHID (ie, R00001.49) is the source for the ExtendedID variable. This is discussed at http://www.nlsinfo.org/nlsy79/docs/79html/79text/hhcomp.htm.

For more information on *R* (*ie*, the Relatedness coefficient), please see Rodgers, Joseph Lee, & Kohler, Hans-Peter (2005). Reformulating and simplifying the DF analysis model. *Behavior Genetics, 35* (2), 211-217.

See Also

The LinksPair79 dataset contains columns necessary for a basic BG analysis. The Links79PairExpanded dataset contains further information that might be useful in more complicated BG analyses.

A tutorial that produces a similar dataset is http://www.nlsinfo.org/childya/nlsdocs/tutorials/linking_mothers_and_children/li
It provides examples in SAS, SPSS, and STATA.

The current dataset (ie, Links79Pair) can be saved as a CSV file (comma-separated file) and imported into in other programs and languages. In the R console, type the following two lines of code:

```
library(NlsyLinks)

write.csv(Links79Pair, "C:/BGDirectory/Links79Pair.csv")
```

where "C:/BGDirectory/" is replaced by your preferred directory. Remember to use forward slashes instead of backslashes; for instance, the path "C:\BGDirectory\Links79Pair.csv" can be misinterpreted.

Examples

```
library(NlsyLinks)     # Load the package into the current R session.
summary(Links79Pair)   # Summarize the five variables.
hist(Links79Pair$R)    # Display a histogram of the Relatedness coefficients.
table(Links79Pair$R)   # Create a table of the Relatedness coefficients for the whole sample.

#Create a dataset of only Gen2 sibs, and display the distribution of R.
gen2Siblings <- subset(Links79Pair, RelationshipPath=='Gen2Siblings')
table(gen2Siblings$R)  # Create a table of the Relatedness coefficients for the Gen2 sibs.
```

Links79PairExpanded *Kinship linking file for pairs of relatives. It builds upon the Links79Pair dataset.*

Description

Please first read the documentation for Links79Pair. That dataset contains the same pairs/rows, but only a subset of the variables/columns.

NOTE: In Nov 2013, the variable naming scheme changed in order to be more consistent across variables. For variables that are measured separately for both subjects (eg, Gender), the subjects' variable name will have an _S1 or _S2 appended to it. For instance, the variables LastSurvey_S1 and LastSurvey_S2 correspond to the last surveys completed by the pair's first and second subject, respectively. Similarly, the functions CreatePairLinksDoubleEntered() and CreatePairLinksSingleEntered() now by default append _S1 and _S2, instead of _1 and _2. However this can be modified using the 'subject1Qualifier' and 'subject2Qualifier' parameters.

Format

A data frame with 11,075 observations on the following 22 variables. There is one row per unique pair of subjects, irrespective of order.

- **ExtendedID** see the variable of the same name in Links79Pair
- **SubjectTag_S1** see the variable of the same name in Links79Pair
- **SubjectTag_S2** see the variable of the same name in Links79Pair
- **R** see the variable of the same name in Links79Pair

- **RFull** This is a superset of R. This includes all the *R* values we estimated, while R (i.e., the variable above) excludes values like *R*=0 for Gen1Housemates, and the associated relationships based on this *R* value (i.e., Gen2Cousins and AuntNieces).
- **RelationshipPath** see the variable of the same name in Links79Pair
- **EverSharedHouse** Indicate if the pair likely live in the same house. This is TRUE for Gen1Housemates, Gen2Siblings, and ParentChild. This is FALSE for AuntNiece and Gen2Cousins
- **IsMz** Indicates if the pair is from the same zygote (ie, they are identical twins/triplets). This variable is a factor, with levels No=0, Yes=1, DoNotKnow=255.
- **LastSurvey_S1** The year of Subject1's most recently completed survey. This may be different that the survey's administration date.
- **LastSurvey_S2** The year of Subject2's most recently completed survey. This may be different that the survey's administration date.
- **RImplicitPass1** The pair's *R* coefficient, using only implicit information. Interpolation was NOT used.
- **RImplicit** The pair's *R* coefficient, using only implicit information. Interpolation was used.
- **RImplicit2004** The pair's *R* coefficient released in our previous projects (**need reference**). This variable is provided primarily for previous users wishing to replicate previous analyses.
- **RExplicitPass1** The pair's *R* coefficient, using only explicit information. Interpolation was NOT used.
- **RExplicit** The pair's *R* coefficient, using only explicit information. Interpolation was used.
- **RExplicitOlderSibVersion** The pair's *R* coefficient, according to the explicit item responses of the older sibling.
- **RExplicitYoungerSibVersion** The pair's *R* coefficient, according to the explicit item responses of the younger sibling.
- **RPass1** The pair's estimated *R* coefficient, using both implicit and explicit information. Interpolation was NOT used. The variable R is identically constructed, but it did use interpolation.
- **Generation_S1** The generation of the first subject. Values for Gen1 and Gen2 are 1 and 2, respectively.
- **Generation_S2** The generation of the second subject. Values for Gen1 and Gen2 are 1 and 2, respectively.
- **SubjectID_S1** The ID value assigned by NLS to the first subject. For Gen1 Subjects, this is their "CaseID" (ie, R00001.00). For Gen2 subjects, this is their "CID" (ie, C00001.00).
- **SubjectID_S2** The ID value assigned by NLS to the second subject.
- **MathStandardized_S1** The PIAT-Math score for Subject1. See ExtraOutcomes79 for more information about its source.
- **MathStandardized_S2** The PIAT-Math score for Subject2.
- **HeightZGenderAge_S1** The early adult height for Subject1. See ExtraOutcomes79 for more information about its source.
- **HeightZGenderAge_S2** The early adult height for Subject2.

Details

Specifies the relatedness coefficient (ie, '*R*') between subjects in the same extended family. Each row represents a unique relationship pair. An extended family with k subjects will have $k(k\text{-}1)/2$ rows. Typically, Subject1 is older while Subject2 is younger.

`RelationshipPath` variable. Code written using this dataset should NOT assume it contains only Gen2 sibiling pairs. See an example of filtering the relationship category in the in Links79Pair documentation.

Please first read the documentation for Links79Pair. That dataset contains the same pairs/rows, but only a subset of the variables/columns.

The specific steps to determine the *R* coefficient will be described in an upcoming publication. The following information may influence the decisions of an applied researcher.

A distinction is made between 'Explicit' and 'Implicit' information. Explicit information comes from survey items that directly address the subject's relationships. For instance in 2006, surveys asked if the sibling pair share the same biological father (eg, Y19940.00 and T00020.00). Implicit information comes from items where the subject typically isn't aware that their responses may be used to determine genetic relatedness. For instance, if two siblings have biological fathers with the same month of death (eg, R37722.00 and R37723.00), it may be reasonable to assume they share the same biological father.

'Interpolation' is our lingo when other siblings are used to leverage insight into the current pair. For example, assume Subject 101, 102, and 103 have the same mother. Further assume 101 and 102 report they share a biological father, and that 101 and 103 share one too. Finally, assume that we don't have information about the relationship between 102 and 103. If we are comfortable with our level of uncertainty of these determinations, then we can interpolate/infer that 102 and 103 are full-siblings as well.

The math and height scores are duplicated from ExtraOutcomes79, but are included here to make some examples more concise and accessible.

Author(s)

Will Beasley

Source

See Links79Pair.

Examples

```
library(NlsyLinks) # Load the package into the current R session.
olderR  <- Links79PairExpanded$RExplicitOlderSibVersion  # Declare a concise variable name.
youngerR <- Links79PairExpanded$RExplicitYoungerSibVersion # Declare a concise variable name.

plot(jitter(olderR), jitter(youngerR))  # Scatterplot the siblings' responses.
table( youngerR, olderR)  # Table of the relationship between the siblings' responses.
ftable(youngerR, olderR, dnn=c("Younger's Version", "Older's Version")) # A formatted table.

# write.csv(
#   Links79PairExpanded,
#   file      ='~/NlsyLinksStaging/Links79PairExpanded.csv',
```

```
#   row.names = FALSE
# )
```

`ReadCsvNlsy79` *Read a CSV file downloaded from the NLS Investigator*

Description

The function accepts a (file path to) CSV file and creates a base::data.frame. The base::data.frame is modified and augmented with columns to assist later routines.

Usage

```
ReadCsvNlsy79Gen1(filePath, dsExtract = utils::read.csv(filePath))
```

Arguments

`filePath`	A path to the CSV file. Remember to use double back-slashes in Windows, or forward-slashes in Windows or Linux.
`dsExtract`	A 'data.frame' (containing the extract) can be passed instead of the file path if the data has already been read into R's memory.

Details

The function does seven things.

1. Reads the CSV into a base::data.frame.
2. Checks that the NLSY variables `C00001.00` and `C00002.00` exist in the base::data.frame.
3. The NLSY variable `C00001.00` is renamed `SubjectID`.
4. A variable named `Generation` is given a value of 2 for all subjects.
5. The `SubjectTag` variable is created.
6. The NLSY variable `C00002.00` is multiplied by 100 and renamed `SubjectTagOfMother`.
7. The NLSY variable `R00001.49` (ie, their Mother's `HHID` is attached to each Gen2 record).

Value

A base::data.frame to facililate biometric analysis.

Author(s)

Will Beasley

Examples

```
## Not run:
filePathGen2 <- "~/Nlsy/Datasets/gen2-birth.csv"
ds <- ReadCsvNlsy79Gen2(filePath=filePathGen2)

## End(Not run)
```

RGroupSummary *Calculates summary statistics for each* R*elatedness Group in the sample.*

Description

Before and after running ACE Models, it is important to examine the characteristics of the different groups. When the ACE is estimated with an SEM using multiple groups, it is even even more important. Groups may contain too few subjects to have a well-behaved covariance matrix.

If a group's covariance matrix is not Positive Definite (or it's misbehaving in some other way), it's typically recommended to exclude that group from the SEM.

Usage

```
RGroupSummary(ds, oName_S1, oName_S2, rName = "R",
  determinantThreshold = 1e-05)
```

Arguments

ds	The base::data.frame containing the following variables:
oName_S1	The name of the outcome variable corresponding to the first subject in the pair.
oName_S2	The name of the outcome variable corresponding to the first subject in the pair.
rName	The name of the variable specifying the pair's Relatedness coefficient.
determinantThreshold	The minimum value the covariance matrix's determinant (for the group) should exceed to be considered Positive Definite.

Details

This function isn't specific to an ACE model and groups defined by R. It could be applied to any multiple-group SEM with two manifest/outcome variables. In the future, we may generalize it beyond two manifest variables.

To get summary stats for the entire sample, create a dummy indicator variable that assigns everyone to the same group. See the second example below.

The default determinantThreshold value is nonzero, in order to forgive slight numerical inaccuracies caused by fixed-precision arithmetic.

Value

A base::data.frame with one row per group. The base::data.frame contains the following variables:

- **R** The group's R value. Note the name of this variable can be changed by the user, by specifying a non-default value to the rName argument.
- **Included** Indicates if the group should be included in a multiple-group SEM.
- **PairCount** The number of pairs in the group with *complete* data for R and the two outcome/manifest variables.
- **O1Mean** The mean (of the outcome variable) among the group's first members, excluding the missing values.
- **O2Mean** The mean (of the outcome variable) among the group's second members, excluding the missing values.
- **O1Variance** The variance (of the outcome variable) among the group's first members.
- **O2Variance** The variance (of the outcome variable) among the group's second members.
- **O1O2Covariance** The covariance (of the outcome variable) across the group's first and second members.
- **Correlation** The correlation (of the outcome variable) across the group's first and second members.
- **Determinant** The determinant of the group's covariance matrix.
- **PosDefinite** Indicates if the group's covariance matrix is positive definite.

Author(s)

Will Beasley and David Bard

References

Please see Neale & Maes for more information about SEM with multiple groups.

Examples

```
library(NlsyLinks) #Load the package into the current R session.
dsLinks        <- Links79PairExpanded   # Load the dataset from the NlsyLinks package.
dsLinks        <- dsLinks[dsLinks$RelationshipPath=='Gen2Siblings', ]
oName_S1       <- "MathStandardized_S1" # Stands for Outcome1
oName_S2       <- "MathStandardized_S2" # Stands for Outcome2
dsGroupSummary <- RGroupSummary(dsLinks, oName_S1, oName_S2)
dsGroupSummary

#Should return:
#     R Included PairCount   O1Mean   O2Mean O1Variance O2Variance O1O2Covariance Correlation
#1 0.250     TRUE      2718  94.6439  95.5990    169.650    207.842       41.0783    0.218761
#2 0.375     TRUE       139  92.6043  93.1655    172.531    187.081       40.4790    0.225311
#3 0.500     TRUE      5511  99.8940 100.1789    230.504    232.971      107.3707    0.463336
#4 0.750    FALSE         2 108.5000 106.0000    220.500     18.000       63.0000    1.000000
#5 1.000     TRUE        22  98.6364  95.5455    319.195    343.117      277.5887    0.838789
#  Determinant PosDefinite
```

```
#1     33573.0        TRUE
#2     30638.7        TRUE
#3     42172.2        TRUE
#4         0.0       FALSE
#5     32465.6        TRUE

#To get summary stats for the whole sample, create one large inclusive group.
dsLinks$Dummy <- 1
(dsSampleSummary <- RGroupSummary(dsLinks, oName_S1, oName_S2, rName="Dummy"))

#Should return:
#  Dummy Included PairCount   O1Mean   O2Mean O1Variance O2Variance O1O2Covariance
#1     1     TRUE      8392 98.07162 98.56864    216.466   229.2988       90.90266
#  Correlation Determinant PosDefinite
#1   0.4080195     41372.1        TRUE
###
### ReadCsvNlsy79
###
## Not run:
filePathGen2 <- "~/Nlsy/Datasets/gen2-birth.csv"
ds <- ReadCsvNlsy79Gen2(filePath=filePathGen2)

## End(Not run)
```

`SubjectDetails79` *Dataset containing further details of the Gen1 and Gen2 subjects.*

Description

These variables are useful to many types of analyses (not just behavior genetics), and are provided to save users time.

Format

A data frame with 24,181 observations on the following 12 variables.

- **SubjectTag** see the variable of the same name in Links79Pair
- **ExtendedID** see the variable of the same name in Links79Pair
- **Generation** Indicates if the subject is in generation `1` or `2`.
- **Gender** Indicates if the subject is `Male` or `Female`.
- **RaceCohort** Indicates if the race cohort is `Hispanic`, `Black` or `Nbnh` (*ie*, Non-black, non-hispanic). This comes from the Gen1 variable `R02147.00` and Gen2 variable `C00053.00`.
- **SiblingCountInNls** The number of the subject's siblings, including himself/herself (a singleton has a value of one). This considers only the siblings in the NLSY. For Gen1, this can exclude anyone outside the age range. For Gen2, this excludes anyone who doesn't share the same mother.

- **BirthOrderInNls** Indicates the subject's birth order among the NLSY siblings.
- **SimilarAgeCount** The number of children who were born within roughly 30 days of the subject's birthday, including the subject (for instance, even an only child will have a value of 1). For Gen2 subjects, this should reflect how many children the Gen1 mother gave birth to at the same time (1: singleton; 2: twins, 3: triplets). For Gen1 subjects, this is less certain, because the individual might have been living with a similarly-aged housemate, born to a different mother.
- **HasMzPossibly** Indicates if the subject *might* be a member of an MZ twin/triplet. This will be true if there is a sibling with a DOB within a month, and they are the same gender.
- **IsMz** Indicates if the subject has been identified as a member of an MZ twin/triplet.
- **KidCountBio** The number of biological children known to the NLSY (but not necessarily interviewed by the NLSY.
- **KidCountInNls** The number of children who belong to the NLSY. This is nonnull for only Gen1 subjects.
- **Mob** The subject's month of birth. The exact day is not available to the public. By default, we set their birthday to the 15th day of the month.
- **LastSurveyYearCompleted** The year of the most recently completed survey.
- **AgeAtLastSurvey** The subject's age at the most recently completed survey.
- **IsDead** ##This variable is not available yet## Indicates if the subject was alive for the last attempted survey.
- **DeathDate** ##This variable is not available yet## The subject's month of death. The exact day is not available to the public. By default, we set their birthday to the 15th day of the month.

Author(s)

Will Beasley

Source

Gen1 information comes from the Summer 2013 release of the NLSY79 sample. Gen2 information comes from the Summer 2013 release of the NLSY79 Children and Young Adults sample. Data were extracted with the NLS Investigator (https://www.nlsinfo.org/investigator/).

Examples

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

summary(SubjectDetails79)

oldPar <- par(mfrow=c(3,2), mar=c(2,2,1,.5), tcl=0, mgp=c(1,0,0))
hist(
  SubjectDetails79$SiblingCountInNls,
  main   = "",
  breaks =seq(from=0, to=max(SubjectDetails79$SiblingCountInNls, na.rm=TRUE), by=1)
)
hist(
  SubjectDetails79$BirthOrderInNls,
```

```
  main   = "",
  breaks = seq(from=0, to=max(SubjectDetails79$BirthOrderInNls, na.rm=TRUE), by=1)
)
hist(
  SubjectDetails79$SimilarAgeCount,
  main   = "",
  breaks = seq(from=0, to=max(SubjectDetails79$SimilarAgeCount, na.rm=TRUE), by=1)
)
hist(
  SubjectDetails79$KidCountBio,
  main   = "",
  breaks = seq(from=0, to=max(SubjectDetails79$KidCountBio, na.rm=TRUE), by=1)
)
hist(
  SubjectDetails79$KidCountInNls,
  main   = "",
  breaks = seq(from=0, to=max(SubjectDetails79$KidCountInNls, na.rm=TRUE), by=1)
)
#hist(SubjectDetails79$Mob, main="",
#     breaks=seq.Date(
#       from=min(SubjectDetails79$Mob, na.rm=TRUE),
#       to=max(SubjectDetails79$Mob, na.rm=TRUE),
#       by="year")
#)
par(oldPar)
```

`SurveyDate` *Dataset containing survey details for each subject, for each year*

Description

Each row represents a survey that a subject completed (or didn't complete). It can be very helpful when restructuring the NLS investigator extracts into a longitudinal dataset that's aligned by age (instead of by survey wave). The Age variables can help to align other response variables across subjects. While the 'SurveySource' indicates where to look for their responses.

These variables are useful to many types of analyses (not just behavior genetics), and are provided to save users time.

Format

A data frame with 580,752 observations on the following 6 variables.

- **SubjectTag** see the variable of the same name in Links79Pair
- **SurveySource** The location of that subject's survey responses that year. Values are `NoInterview`, `Gen1`, `Gen2C` or `Gen2YA`.
- **SurveyYear** The year/wave of the survey.
- **SurveyDate** The exact date of the administered survey.

- **AgeSelfReportYears** The subject's age, according to a their own response, or their mother's response.
- **AgeCalculateYears** The subject's age, calculated from subtracting their birthday from the interview date.
- **Age** The subject's age, which uses `AgeCalculateYears` or `AgeSelfReportYears` if it's not available.

Details

The `AgeSelfReportYears` and `AgeCalculateYears` variables usually agree, but not always. The `Age` variable uses `AgeCalculateYears` (or `AgeSelfReportYears` when `AgeCalculateYears` is missing).

The exact *date* of birth isn't public (only the subject's *month* of birth). To balance the downward bias of two weeks, theri birthday is set to the 15th day of the month to produce `AgeCalculateYears`.

In the Gen2 Child dataset, self-reported age is stated by month (eg, the child is 38 months old); a constant of 0.5 months has been added to balance the downward bias. In the Gen2 YA and Gen1 datasets, self-reported age is stated by year (eg, the subject is 52 years old); a constant of 0.5 years has been added.

Author(s)

Will Beasley

Source

Gen1 information comes from the Summer 2013 release of the NLSY79 sample. Gen2 information comes from the January 2015 release of the NLSY79 Children and Young Adults sample. Data were extracted with the NLS Investigator (https://www.nlsinfo.org/investigator/).

Examples

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

summary(SurveyDate)
table(SurveyDate$SurveyYear, SurveyDate$SurveySource)
table(is.na(SurveyDate$AgeSelfReportYears), is.na(SurveyDate$AgeCalculateYears))

if( require(ggplot2) & require(plyr) ) {
  dsSourceYear <- plyr::count(SurveyDate, c("SurveyYear", "SurveySource"))
  dsSourceYear <- dsSourceYear[dsSourceYear$SurveySource != "NoInterview", ]

  ggplot(dsSourceYear, aes(x=SurveyYear, y=freq, color=SurveySource)) +
    geom_line(size=2) +
    geom_point(size=5, shape=21) +
    scale_color_brewer(palette = "Dark2") +
    theme_bw() +
    theme(legend.position=c(0,0), legend.justification=c(0,0))

  ggplot(SurveyDate, aes(x=AgeSelfReportYears, y=AgeCalculateYears)) +
    geom_abline() +
```

```
    geom_point(shape=21) +
    theme_bw()
}
```

`ValidateOutcomeDataset` *Validates the schema of datasets containing outcome variables.*

Description

The **NlsyLinks** handles a lot of the plumbing code needed to transform extracted NLSY datasets into a format that statistical routines can interpret. In some cases, a dataset of measured variables is needed, with one row per subject. This function validates the measured/outcome dataset, to ensure it posses an interpretable schema. For a specific list of the requirements, see `Details` below.

Usage

```
ValidateOutcomeDataset(dsOutcome, outcomeNames)
```

Arguments

`dsOutcome`	A base::data.frame with the measured variables
`outcomeNames`	The column names of the measure variables that eventually will be used by a statistical procedure.

Details

The `dsOutcome` parameter must:

1. Have a non-missing value.
2. Contain at least one row.
3. Contain a column called 'SubjectTag' (case sensitive).
4. Have the SubjectTag column containing only positive numbers.
5. Have the SubjectTag column where all values are unique (ie, two rows/subjects cannot have the same value).

The `outcomeNames` parameter must:

1. Have a non-missing value
2. Contain only column names that are present in the `dsOutcome` data frame.

Value

Returns `TRUE` if the validation passes. Returns an error (and associated descriptive message) if it false.

Author(s)

Will Beasley

Examples

```
library(NlsyLinks) #Load the package into the current R session.
ds <- ExtraOutcomes79
outcomeNames <- c("MathStandardized", "WeightZGenderAge")
ValidateOutcomeDataset(dsOutcome=ds, outcomeNames=outcomeNames) #Returns TRUE.
outcomeNamesBad <- c("MathMisspelled", "WeightZGenderAge")
#ValidateOutcomeDataset(dsOutcome=ds, outcomeNames=outcomeNamesBad) #Throws error.
```

ValidatePairLinks *Validates the schema of a links for pairs of relatives*

Description

A helper function that verifies the linking dataset contains (A) the essential columns exist, and (B) at least one row. It is called by `CreatePairLinks`.

Typical use of **NlsyLinks** will not require this function, since a valid paired links are supplied for each supported sample (ie, Links79Pair).

The **NlsyLinks** uses several types of linking schemas. This function validates the type where each relative subject has their own row.

The following four columns must be present: (1) `Subect1Tag`, (2) `Subect2Tag`, (3) `R`, and (4) `MultipleBirth`. They must have a `numeric` mode/datatype.

Usage

```
ValidatePairLinks(linksPair)
```

Arguments

`linksPair`	The base::data.frame to validate.

Value

Returns `TRUE` if the validation passes. Returns an error (and associated descriptive message) if it false.

See Also

Links79Pair, Links79PairExpanded,

Examples

```
dsSingleLinks <- data.frame(
  ExtendedID        = c(  1  , 1,   1  , 2),
  SubjectTag_S1     = c(101, 101, 102, 201),
  SubjectTag_S2     = c(102, 103, 103, 202),
  R                 = c( .5, .25, .25,  .5),
  RelationshipPath  = rep("Gen2Siblings", 4)
)
dsSingleOutcomes <- data.frame(
  SubjectTag = c(101, 102, 103, 201, 202),
  DV1        = c( 11,  12,  13,  41,  42),
  DV2        = c( 21,  22,  23,  51,  52)
)
dsDouble <- CreatePairLinksDoubleEntered(
  outcomeDataset         = dsSingleOutcomes,
  linksPairDataset       = dsSingleLinks,
  outcomeNames           = c("DV1", "DV2"),
  validateOutcomeDataset = TRUE
)
dsDouble #Show the 8 rows in the double-entered pair links
summary(dsDouble) #Summarize the variables

ValidatePairLinksAreSymmetric(dsDouble) #Should return TRUE.
```

`ValidatePairLinksAreSymmetric` *Verifies that the pair relationships are symmetric.*

Description

For certain analyses, the pairs links (which can be considered a type of sparse matrix) need to be symmetric. For instance, if there is a row for Subjects 201 and 202 with R=0.5, there should be a second row for Subjects 202 and 201 with R=0.5.

This validation function is useful to some types of DF methods and some spatially-inspired methods.

Usage

```
ValidatePairLinksAreSymmetric(linksPair)
```

Arguments

`linksPair`	The base::data.frame object that should be symmetric

Value

Returns TRUE if symmetric. Throw an error with `base::stop()` if asymmetric.

Author(s)

Will Beasley

See Also

CreatePairLinksDoubleEntered()

Examples

```
dsSingleLinks <- data.frame(
  ExtendedID       = c(  1,   1,   1  , 2),
  SubjectTag_S1    = c(101, 101, 102, 201),
  SubjectTag_S2    = c(102, 103, 103, 202),
  R                = c( .5, .25, .25,  .5),
  RelationshipPath = rep("Gen2Siblings", 4)
)
dsSingleOutcomes <- data.frame(
  SubjectTag = c(101, 102, 103, 201, 202),
  DV1        = c( 11,  12,  13,  41,  42),
  DV2        = c( 21,  22,  23,  51,  52)
)
dsDouble <- CreatePairLinksDoubleEntered(
  outcomeDataset         = dsSingleOutcomes,
  linksPairDataset       = dsSingleLinks,
  outcomeNames           = c("DV1", "DV2"),
  validateOutcomeDataset = TRUE
)
dsDouble #Show the 8 rows in the double-entered pair links
summary(dsDouble) #Summarize the variables

ValidatePairLinksAreSymmetric(dsDouble) #Should return TRUE.
```

Index