Package 'ggenealogy'

December 12, 2016

Version 0.3.0

Title Visualization Tools for Genealogical Data

Description Methods for searching through genealogical data and displaying the results. Plotting algorithms assist with data exploration and publication-quality image generation. Includes interactive genealogy visualization tools. Provides parsing and calculation methods for variables in descendant branches of interest. Uses the Grammar of Graphics.

Author Lindsay Rutter, Susan Vanderplas, Di Cook

Maintainer Lindsay Rutter <lrutter@iastate.edu>

License GPL

Depends R (>= 3.3.0)

Imports ggplot2 (>= 2.2.0), igraph (>= 0.7.1), plyr (>= 1.8.1), reshape2 (>= 1.4), plotly (>= 4.5.6), tibble (>= 1.2)

VignetteBuilder knitr

Suggests stringr (>= 0.6.2), knitr (>= 1.13), roxygen2 (>= 3.0.0), dplyr (>= 0.5.0)

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-12 18:34:18

R topics documented:

`buildAncDesCoordDF` *Returns the coordinate positions of all ancestors and descendants of a variety.*

Description

Calculates coordinates to plot each ancestors and descendant of a variety in a lineage. The x and y values describe the coordinates of the label, while the xstart, ystart, xend, and yend values describe the edges of the label.

Usage

```
buildAncDesCoordDF(df)
```

Arguments

`df`	the data frame of the ancestors and descendants of a variety (from function buildAncDesTotalDF)

See Also

buildAncList for information on determining ancestors

buildDesList for information on determining descendants

---

buildAncDesTotalDF *Returns data frame with plot coordinates of all ancestors and descendants of a variety.*

Description

Returns the data frame that includes labels and plot coordinates of all ancestors and descendants of a variety. Users can specify the maximum number of ancestors and descendants to display.

Usage

```
buildAncDesTotalDF(v1, geneal, mAnc = 3, mDes = 3)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)
mAnc	the maximum number of generations of ancestors of v1 to be displayed (in numeric format)
mDes	the maximum number of generations of descendants of v1 to be displayed (in numeric format)

See Also

buildAncList for information on determining ancestors

buildDesList for information on determining descendants

Examples

```
data(sbGeneal)
v1 <- "Essex"
buildAncDesTotalDF(v1, sbGeneal)
```

buildAncList *Returns the ancestors of a particular variety (if they exist).*

Description

This function returns a nested list of the ancestors of the inputted variety.

Usage

```
buildAncList(v1, geneal, gen = 0)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)
gen	the generation (note: This should be left as default, as any other input will not affect results anyway)

See Also

getParent for information on determining parents

Examples

```
data(sbGeneal)
getParent("Essex", sbGeneal)
buildAncList("Essex", sbGeneal)
```

buildDesList *Returns the descendants of a particular variety (if they exist).*

Description

This function returns a nested list of the descendants of the inputted variety.

Usage

```
buildDesList(v1, geneal, gen = 0)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)
gen	the generation (note: This should be left as default, as any other input will not affect results)

See Also

getChild for information on determining children

Examples

```
data(sbGeneal)
getParent("Essex", sbGeneal)
buildDesList("Essex", sbGeneal, 3)
```

---

buildEdgeTotalDF *Build the edges in the genealogy graph.*

Description

This function takes the graph object and creates a data frame object of the edges between all parent-child relationships in the graph.

Usage

```
buildEdgeTotalDF(geneal, ig, colName, bin = 12)
```

Arguments

geneal	the full genealogy (in data frame format)
ig	the graph representation of the data genealogy (in igraph format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
bin	the number of bins to determine the vertical positions of nodes (default is 12). For more information on choosing bin size, please visit the ggenealogy vignette.

See Also

dfToIG for information on producing ig from the genealogy

https://www.r-project.org for iGraph information

buildMinusPathDF *Process the genealogy graph*

Description

This function takes the spreadTotalDF object (from the buildSpreadTotalDF function) and the path object as inputs. From these objects, it creates a data frame object of the label, x, and y values of all nodes in the ful genealogy. However, the data frame object does not include the labels of the path varieties, as they will be treated differently.

Usage

```
buildMinusPathDF(path, geneal, ig, colName, colNameY, bin = 12)
```

Arguments

path	path as returned from getPath() or a vector of two variety names which exist in the ig object
geneal	the full genealogy (in data frame format)
ig	the graph representation of the data genealogy (in igraph format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
colNameY	the name of the second optional column of the data frame that contains the second optional quantitative variable of interest (in character string format). This optional quantitative variable will be plotted on the vertical axis.
bin	the number of bins to determine the vertical positions of nodes (default is 12). For more information on choosing bin size, please visit the ggenealogy vignette.

See Also

https://www.r-project.org for iGraph information

getPath for information on input path building

buildPathDF *Build data frame for path representation*

Description

This function builds a dataframe of information about the path object that can later be used for visualization. The dataframe includes "label" (name of each variety) of each node, "x" (the date of the variety, the x-axis value for which the label and incoming/outgoing edges are centered), "y" (the y-axis value, which is the index of the path, incremented by unity), "xstart" (the x-axis position of the outgoing edge (leaving to connect to the node at the next largest y-value)), "xend" (the x-axis position of the outgoing edge (connected to the node at the next largest y-value)), "ystart" (the y-axis position of the outgoing edge (leaving to connect to the node at the next largest y-value), "yend" (the y-axis position of the outgoing edge (connected to the node at the next largest y-value))).

Usage

```
buildPathDF(path, geneal, colName, colNameY = "")
```

Arguments

path	path object representing the path between two vertices
geneal	the full genealogy (in data frame format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
colNameY	the name of the second optional column of the data frame that contains the second optional quantitative variable of interest (in character string format). This optional quantitative variable will be plotted on the vertical axis.

---

buildPlotTotalDF *Build all labels in the graph*

Description

This function takes the spreadTotalDF object (from the buildSpreadTotalDF function) and the path object as inputs. From these objects, it creates a data frame object of the text label positions for the varieties in the path, as well as the edges only in the varieties in the path.

Usage

```
buildPlotTotalDF(path, geneal, ig, colName, colNameY = "", bin = 12)
```

Arguments

path	path as returned from getPath() or a vector of two variety names which exist in ig
geneal	the full genealogy (in data frame format)
ig	the graph representation of the data genealogy (in igraph format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
colNameY	the name of the second optional column of the data frame that contains the second optional quantitative variable of interest (in character string format). This optional quantitative variable will be plotted on the vertical axis.
bin	the number of bins to determine the vertical positions of nodes (default is 12). For more information on choosing bin size, please visit the ggenealogy vignette

See Also

https://www.r-project.org for iGraph information

https://www.r-project.org for iGraph information

getPath for information on input path building

buildSpreadTotalDF *Build a data frame where the varieties are spread so they do not overlap*

Description

Constructs a data frame object so that varieties are spread such that they do not overlap, even though the x-axis position will represent dates.

Usage

```
buildSpreadTotalDF(geneal, ig, colName, bin = 12)
```

Arguments

geneal	the full genealogy (in data frame format)
ig	the graph representation of the data genealogy (in igraph format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
bin	the number of bins to determine the vertical positions of nodes (default is 12). For more information on choosing bin size, please visit the ggenealogy vignette

See Also

https://www.r-project.org for iGraph information

dfToIG *Process the genealogy graph*

Description

Processes the genealogy into an igraph object with appropriate vertex information, graph type, and edge weights.

Usage

```
dfToIG(geneal, vertexinfo = NULL, edgeweights = 1, isDirected = FALSE)
```

Arguments

geneal	the full genealogy (in data frame format)
vertexinfo	(default NULL) either names of columns in the genealogy which should be added to the database as vertex information or a data frame with information for all vertices such that the first column contains vertex names.
edgeweights	(default 1) name of a column which contains edge weights
isDirected	(default FALSE) should the graph be a directed graph?

See Also

https://www.r-project.org for iGraph information

`getAncestors` *Returns a list of the ancestors of a particular variety (if they exist)*

Description

This function returns a list of the ancestors of the inputted variety within and including a given number of generations

Usage

```
getAncestors(v1, geneal, gen = 3)
```

Arguments

`v1`	the label of the vertex of interest (in character string format)
`geneal`	the full genealogy (in data frame format)
`gen`	the number of generations back to include as ancestors

Examples

```
data(sbGeneal)
getParent("Essex", sbGeneal)
getAncestors("Essex", sbGeneal, 1)
getAncestors("Essex", sbGeneal, 5)
```

`getBasicStatistics` *Determine basic statistics of the graph object*

Description

Returns basic statistics of the graph object (number of nodes, number of edges, whether or not the whole graph is connected, number of components, average path length, graph diameter, etc.)

Usage

```
getBasicStatistics(ig)
```

Arguments

`ig`	the graph representation of the data genealogy (in igraph format)

Examples

```
data(sbGeneal)
ig <- dfToIG(sbGeneal)
getBasicStatistics(ig)
```

`getBranchQual` *Descendant branch calculations for quantitative variable*

Description

Returns a data frame containing the names of all children of an individual of interest ("Name"). The mean and standard deviation ("Mean" and "SD") of a quantitative variable across all descendents of each child is reported. In addition, for each child, the number of its descendants is reported ("Count"), the number of its descendants who do not have a value for the quantitative variable ("NACount") is reported, and the names of all of its descendants is reported ("DesNames").

Usage

```
getBranchQual(v1, geneal, colName, rExpr, gen = 3)
```

Arguments

`v1`	the label of the vertex of interest (in character string format)
`geneal`	the full genealogy (in data frame format)
`colName`	the name of the column of the data frame that contains the qualitative variable of interest (in character string format)
`rExpr`	regular expression to be applied to the column that contains the qualitative variable of interest (in character string format). The regular expression syntax must work on a data frame column of type character. The term geneal$colName must be used in the regular expression.
`gen`	the number of generations back to include as ancestors

Examples

```
data(statGeneal)
rExpr = "geneal$colName=='The Johns Hopkins University'"
DC_JHU = getBranchQual("David Cox", statGeneal, "school", rExpr, 15)
rExpr = "geneal$colName=='UnitedKingdom'"
DC_UK = getBranchQual("David Cox", statGeneal, "country", rExpr, 15)
rExpr = "grepl('(?i)Stochastic', geneal$colName)"
DC_Stochastic = getBranchQual("David Cox", statGeneal, "thesis", rExpr, 15)
```

getBranchQuant *Descendant branch calculations for quantitative variable*

Description

Returns a data frame containing the names of all children of an individual of interest ("Name"). The mean and standard deviation ("Mean" and "SD") of a quantitative variable across all descendents of each child is reported. In addition, for each child, the number of its descendants is reported ("Count"), the number of its descendants who do not have a value for the quantitative variable ("NACount") is reported, and the names of all of its descendants is reported ("DesNames").

Usage

```
getBranchQuant(v1, geneal, colName, gen = 3)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
gen	the number of generations back to include as ancestors

Examples

```
data(statGeneal)
DC_Year <- getBranchQuant("David Cox", statGeneal, "gradYear", 15)
```

getChild *Returns the children of a particular variety (if they exist)*

Description

This function returns zero or more values that indicate the children of the inputted variety.

Usage

```
getChild(v1, geneal)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)

Examples

```
data(sbGeneal)
getChild("Tokyo", sbGeneal)
getChild("Essex", sbGeneal)
```

---

getDegree *Determine the degree between two varieties*

Description

Returns the degree (distance between unweighted edges) between two varieties, where an edge represents a parent-child relationship

Usage

```
getDegree(v1, v2, ig, geneal)
```

Arguments

v1	the label of the first vertex of interest (in character string format)
v2	the label of the second vertex of interest (in character string format)
ig	the graph representation of the data genealogy (in igraph format)
geneal	the full genealogy (in data frame format)

Examples

```
data(sbGeneal)
ig <- dfToIG(sbGeneal)
getDegree("Brim", "Bedford", ig, sbGeneal)
```

---

getDescendants *Returns a list of the descendants of a particular variety (if they exist)*

Description

This function returns a list of the descendants of the inputted variety within and including a given number of generations

Usage

```
getDescendants(v1, geneal, gen = 3)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)
gen	the number of generations back to include as descendants

Examples

```
data(sbGeneal)
getChild("Essex", sbGeneal)
getDescendants("Essex", sbGeneal, 1)
getDescendants("Essex", sbGeneal, 3)
```

---

getEdges *Returns edges (vertex names and edge weights) for the full genealogy*

Description

Returns a matrix, where each row contains information about an edge (two vertex names and edge weight, if present) of the full genealogy.

Usage

```
getEdges(ig, geneal)
```

Arguments

ig	the graph representation of the data genealogy (in igraph format)
geneal	the full genealogy (in data frame format)

Examples

```
data(sbGeneal)
ig <- dfToIG(sbGeneal)
getEdges(ig, sbGeneal)
```

getNodes *Returns the nodes for a full genealogy*

Description

Returns a character list, where rows contains names of the unique nodes in the full genealogy

Usage

```
getNodes(geneal)
```

Arguments

geneal	the full genealogy (in data frame format)

Examples

```
data(sbGeneal)
getNodes(sbGeneal)
```

getParent *Returns the parents of a particular variety (if they exist)*

Description

This function returns up to two values that indicate the parents of the inputted variety.

Usage

```
getParent(v1, geneal)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)

Examples

```
data(sbGeneal)
getParent("Tokyo", sbGeneal)
getParent("Essex", sbGeneal)
```

getPath *Determine the path between two varieties*

Description

Determines the shortest path between the two inputted vertices, and takes into account whether or not the graph is directed. If there is a path, the list of vertices of the path will be returned. If there is not a path, a list of character(0) will be returned. Note: For a directed graph, the direction matters. However, this function will check both directions and return the path if it exists.

Usage

```
getPath(v1, v2, ig, geneal, colName, silent = FALSE, isDirected = FALSE)
```

Arguments

v1	the label of the first vertex of interest (in character string format)
v2	the label of the second vertex of interest (in character string format)
ig	the graph representation of the data genealogy (in igraph format)
geneal	the full genealogy (in data frame format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
silent	whether or not to print output (defaults to false)
isDirected	whether or not the graph is directed (defaults to false)

Examples

```
data(sbGeneal)
ig <- dfToIG(sbGeneal)
getPath("Brim", "Bedford", ig, sbGeneal, "devYear")
getPath("Tokyo", "Volstate", ig, sbGeneal, "yield")
```

getPathOnly *Determine the path between two varieties*

Description

Determines the shortest path between the two inputted vertices, and takes into account whether or not the graph is directed. If there is a path, the list of vertices of the path will be returned. If there is not a path, a list of character(0) will be returned. Note: For a directed graph, the direction matters. However, this function will check both directions and return the path if it exists.

Usage

```
getPathOnly(v1, v2, ig, geneal, silent = FALSE, isDirected = FALSE)
```

Arguments

v1	the label of the first vertex of interest (in character string format)
v2	the label of the second vertex of interest (in character string format)
ig	the graph representation of the data genealogy (in igraph format)
geneal	the full genealogy (in data frame format)
silent	whether or not to print output (defaults to false)
isDirected	whether or not the graph is directed (defaults to false)

---

getVariable *Determine the date of a variety*

Description

Returns the documented date of the inputted variety

Usage

```
getVariable(v1, geneal, colName)
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)

Examples

```
data(sbGeneal)
getVariable("Essex", sbGeneal, "devYear")
getVariable("Tokyo", sbGeneal, "yield")
```

isChild *Determine if a variety is a child of another*

Description

Returns a boolean variable for whether the first variety is a child of the second variety

Usage

```
isChild(child, parent, geneal)
```

Arguments

child	possible child variety
parent	possible parent variety
geneal	the full genealogy (in data frame format)

Examples

```
data(sbGeneal)
isChild("Essex", "Young", sbGeneal)
isChild("Young", "Essex", sbGeneal)
```

isParent *Determine if a variety is a parent of another*

Description

Returns a boolean variable for whether the second variety is a parent of the first variety

Usage

```
isParent(child, parent, geneal)
```

Arguments

child	possible child variety
parent	possible parent variety
geneal	the full genealogy (in data frame format)

Examples

```
data(sbGeneal)
isParent("Essex", "Young", sbGeneal)
isParent("Young", "Essex", sbGeneal)
```

nodeToDF — *Returns the data frame representation of all ancestors and descendants of a variety*

Description

Converts the list-style-genealogy to a data frame, where each variety has an id value and references its' parent's id value. ID value ranges correspond to generation. It is possible that with more complex genealogical structures the range of id values may need to expand to reduce the probability of two varieties being assigned the same id value.

Usage

```
nodeToDF(tlist, branch = 0, par.id = NA, id = 1)
```

Arguments

tlist	list of varieties
branch	of particular variety in the genealogy
par.id	the id of the parent
id	id offset

plotAncDes — *Returns the image object to show the ancestors and descendants of a variety*

Description

Returns the image object to show the ancestors and descendants of a variety, with the variety highlighted, if desired

Usage

```
plotAncDes(v1, geneal, mAnc = 3, mDes = 3, vColor = "#D35C79")
```

Arguments

v1	the label of the vertex of interest (in character string format)
geneal	the full genealogy (in data frame format)
mAnc	the maximum number of generations of ancestors of v1 to be displayed (in numeric format)
mDes	the maximum number of generations of descendants of v1 to be displayed (in numeric format)
vColor	the color of the text of the main variety

Examples

```
data(sbGeneal)
plotAncDes("Tokyo", sbGeneal, vColor = "red")
plotAncDes("Essex", sbGeneal, 2, 3, "blue") + ggplot2::labs(x = "Generation index", y = "")
```

`plotDegMatrix` *Returns the image object to show the heat map of degrees between the inputted set of vertices*

Description

Returns the image object to show the heat map of degrees between the inputted set of vertices

Usage

```
plotDegMatrix(varieties, ig, geneal)
```

Arguments

`varieties`	subset of varieties used to generate the heat map
`ig`	the graph representation of the data genealogy (in igraph format)
`geneal`	the full genealogy (in data frame format)

See Also

https://www.r-project.org for iGraph information

Examples

```
data(sbGeneal)
ig <- dfToIG(sbGeneal)
varieties <- c("Bedford", "Calland", "Narow", "Pella", "Tokyo", "Young", "Zane")
p <- plotDegMatrix(varieties, ig, sbGeneal)
p + ggplot2::scale_fill_continuous(low = "white", high = "darkgreen")
```

plotPath *Construct the graphic object of the path*

Description

This function takes the path as input and outputs an ggplot2 object. The image will correctly position the node labels with x-axis representing the node date, and y-axis representing the node path index. Edges between two nodes represent parent-child relationships between those nodes. For visual appeal, there is a grey box that outlines the node label, as well as an underline and overline for each label.

Usage

```
plotPath(path, geneal, colName, colNameY = "", fontFace = 1)
```

Arguments

path	object created from function getPath
geneal	the full genealogy (in data frame format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
colNameY	the name of the second optional column of the data frame that contains the second optional quantitative variable of interest (in character string format). This optional quantitative variable will be plotted on the vertical axis.
fontFace	fontface for the two nodes of interest (1=plain, 2=bold, 3=italic, 4=bold-italic), DEFAULT is 1

See Also

getPath for information on input path building

Examples

```
data(sbGeneal)
ig <- dfToIG(sbGeneal)
pathTN <- getPath("Tokyo", "Narow", sbIG, sbGeneal, "devYear")
plotPath(pathTN, sbGeneal, "devYear")

sbFilt <- sbGeneal[complete.cases(sbGeneal[1:3]),]
sbFiltIG <- dfToIG(sbFilt)
pathCL <- getPath("Clark", "Lawrence", sbFiltIG, sbFilt, "yield")
plotPath(pathCL, sbFilt, "devYear", "yield") + ggplot2::xlab("Dev Year") + ggplot2::ylab("Yield")
```

plotPathOnAll *Plot a path between two vertices over the full genealogy*

Description

This function requires a path and the ig object, and plots the full genealogy with the path highlighted. The image will correctly position the node labels with x-axis representing the node date, and y-axis representing the node path index. Light grey edges between two nodes represent parent-child relationships between those nodes. To enhance the visual understanding of how the path-of-interest fits into the entire graph structure, the nodes within the path are labelled in boldface, and connected with light-green boldfaced edges.

Usage

```
plotPathOnAll(path, geneal, ig, colName, colNameY = "", bin = 12,
  edgeCol = "gray84", pathEdgeCol = "seagreen", nodeSize = 3,
  pathNodeSize = 3, pathNodeFont = "bold", nodeCol = "black",
  animate = FALSE)
```

Arguments

path	path as returned from getPath() or a vector of two variety names which exist in ig
geneal	the full genealogy (in data frame format)
ig	the graph representation of the data genealogy (in igraph format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
colNameY	the name of the second optional column of the data frame that contains the second optional quantitative variable of interest (in character string format). This optional quantitative variable will be plotted on the vertical axis.
bin	the number of bins to determine the vertical positions of nodes (default is 12). For more information on choosing bin size, please visit the ggenealogy vignette
edgeCol	color of the non-path edges, default is "gray84"
pathEdgeCol	color of the path edges, default is "seagreen"
nodeSize	text size of the non-path node labels, default is 3
pathNodeSize	text size of the path node labels, default is 3
pathNodeFont	font face of text of the path node labels ("plain", "italic", "bold", "bold.italic"), default is "bold"
nodeCol	color of the non-path node labels, default is black
animate	if the plot will have interactive capabilities, default is FALSE

See Also

https://www.r-project.org for iGraph information

getPath for information on input path building

Examples

```
data(sbGeneal)
sb <- sbGeneal[complete.cases(sbGeneal[1:3]),]
ig <- dfToIG(sb)
pathCL <- getPath("Clark", "Lawrence", ig, sb, "yield")
plotPathOnAll(pathCL, sb, ig, "yield", bin = 3, pathEdgeCol = "red") + ggplot2::xlab("Yield")
plotPathOnAll(pathCL, sb, ig, "yield", "devYear") + ggplot2::xlab("Yield") + ggplot2::ylab("Year")
```

---

plotVariableMatrix *Returns the image object to show the heat map of dates between the inputted set of vertices*

Description

Returns the image object to show the heat map of dates between the inputted set of vertices

Usage

```
plotVariableMatrix(varieties, geneal, colName, xLab = "Variety",
  yLab = "Variety", legendLab = "Difference in variable")
```

Arguments

varieties	subset of varieties used to generate the heat map
geneal	the full genealogy (in data frame format)
colName	the name of the column of the data frame that contains the quantitative variable of interest (in character string format)
xLab	string label on the x axis (default is "Variety")
yLab	string label on the y axis (default is "Variety")
legendLab	string label on the legend (default is "Degree")

Examples

```
data(sbGeneal)
varieties <- c("Bedford", "Calland", "Narow", "Pella", "Tokyo", "Young", "Zane")
p <- plotVariableMatrix(varieties, sbGeneal, "devYear", "Variety", "Variety", "Difference")
p + ggplot2::scale_fill_continuous(low = "white", high = "darkgreen")
```

sbGeneal *Soybean data*

Description

This data set contains soy bean genealogical information maintained by the United States Department of Agriculture to be used by plant breeders, geneticists, bioinformaticians, pathologists,and many other research workers.

Usage

```
data(sbGeneal)
```

Format

a RData instance, 1 row per each child-parent relationship between soybean varieties

Details

Soybean genealogical data

This data contains information on copy number variants, single nucleotide polymorphisms, protein content, and yield, of soybeans. The available data consists of a data frame structure that contains 412 direct child-parent relationships between pairs of soybean varieties. These data were collected from field trials, genetic studies, and United States Department of Agriculture (USDA) bulletins, and date as early as the first decade of the 1900s.

- child name of child soybean variety
- devYear year child variety was introduced
- yield protein yield
- yearImputed whether or not the introduced year of the child variety was imputed
- parent name of parent soybean variety

References

"Pedigrees of soybean cultivars released in the United States and Canada." Theodore Hyivitz, C.A. Newell, S.G. Carmer. College of Agriculture, University of Illinois at Urbana-Champaign (1977).

`statGeneal` *Academic statistics data*

Description

This data set contains academic genealogical information from the Mathematics Genealogy Project, a web-based database serviced by the North Dakota State University Department of Mathematics and the American Mathematical Society. Specifically, this data set represents a subset of the Mathematical Genealogy Project that contains all the parent-child relationships where both parent and child received an advanced degree of statistics as of June 6, 2015.

Usage

```
data(statGeneal)
```

Format

a `RData` instance, 1 row per each individual in the Mathematics Genealogy Project with an advanced degree in statistics. If the child had a parent who was also listed as having received an advanced degree in statistics in the Mathematics Genealogy project, then the row also contains the parental information.

Details

Academic statistics genealogical data

This data contains information on copy number variants, single nucleotide polymorphisms, protein content, and yield, of soybeans. The available data consists of a data frame structure that contains 412 direct child-parent relationships between pairs of soybean varieties. These data were collected from field trials, genetic studies, and United States Department of Agriculture (USDA) bulletins, and date as early as the first decade of the 1900s.

- child name of the individual who received an advanced degree in statistics
- parent name of the individual who mentored the child and also received an advanced degree in statistics. If the child has no such parent, then this field is an empty string
- gradYear year the child received their advanced degree in statistics
- country country from which the child received their advanced degree in statistics
- school school from which the child received their advanced degree in statistics
- thesis title of the thesis the child submitted to receive their advanced degree in statistics. If this information is not available, then this field is an empty string

References

North Dakota State University and American Mathematical Society (2010). The Mathematics Genealogy Project. Archived Web Site. Retrieved from the Library of Congress, Accessed on March 6, 2015, URL http://www.genealogy.math.ndsu.nodak.edu.

Index