Package ‘iNZightTools’

September 17, 2021

Type Package
Title Tools for 'iNZight'
Version 1.12.0
Depends R (>= 4.0)
Imports chron, dplyr, forcats, glue, grDevices, haven, magrittr,
       methods, RcppTOML, readr (>= 1.2.0), readxl, srvyr, stats,
       stringr, styler, survey, tibble, tidyr, tools, lubridate,
       utils, validate, zoo
Suggests covr, jsonlite, RCurl, testthat (>= 3.0.0)

BugReports https://github.com/iNZightVIT/iNZightTools/issues
Contact inzight_support@stat.auckland.ac.nz

URL http://inzight.nz

Description Provides a collection of wrapper functions for common variable and dataset manipulation workflows primarily used by 'iNZight', a graphical user interface providing easy exploration and visualisation of data for students of statistics, available in both desktop and online versions. Additionally, many of the functions return the 'tidyverse' code used to obtain the result in an effort to bridge the gap between GUI and coding.

License GPL-3
Encoding UTF-8
Language en-GB
RoxygenNote 7.1.1
Config/testthat/edition 3

NeedsCompilation no

Author Tom Elliott [aut, cre] (https://orcid.org/0000-0002-7815-6318),
       Christoph Knopf [ctb],
       Akshay Gupta [ctb],
       Owen Jin [aut] (Tidyverse variable/data manipulation functions),
       Lushi Cai [ctb],
       Yiwen He [aut] (Dates/times and dataset manipulation),
       Daniel Barnett [aut] (Data validation)
R topics documented:

- add_suffix
- aggregateData
- aggregatedt
- appendrows
- as_survey
- as_survey_spec
- code
- collapseLevels
- combineCatVars
- convertToCat
- convert_to_datetime
- countMissing
- createNewVar
- create_varname
- deleteVars
- extract_part
- filterLevels
- filterNumeric
- filterRandom
- filterRows
- fitDesign
- fitModel
- form_class_intervals
- import_survey
- iNZightTools
- is_cat
- is_dt
- is_num
- is_preview
- is_survey
- is_svydesign
- is_svyrep
- joindata
- load_rda
- make_names
- make_survey
- missingToCat
- newdevice
- print.inzsvyspec
- rankVars
- read_meta
add_suffix

Description

When creating new variables or modifying the data set, we often add a suffix added to distinguish the new name from the original one. However, if the same action is performed twice (for example, filtering a data set), the suffix is duplicated (data.filtered.filtered). This function averts this by adding the suffix if it doesn’t exist, and otherwise appending a counter (data.filtered2).

Usage

add_suffix(name, suffix)

Arguments

name a character vector containing (original) names
suffix the suffix to add, a length-one character vector

Value

character vector of names with suffix appended
aggregateData

Aggregate data by categorical variables

Description

Aggregate a dataframe into summaries of all numeric variables by grouping them by specified categorical variables and returns the result along with tidyverse code used to generate it.

Usage

aggregateData(
  .data,
  vars,
  summaries,
  summary_vars,
  varnames = NULL,
  quantiles = c(0.25, 0.75),
  custom_funs = NULL
)

Arguments

.data a dataframe or survey design object to aggregate
vars a character vector of categorical variables in .data to group by
summaries summaries to generate for the groups generated in vars. See details.
summary_vars names of variables in the dataset to calculate summaries of
varnames name templates for created variables (see details).
quantiles if requesting quantiles, specify the desired quantiles here
custom_funs a list of custom functions (see details).

Value

aggregated dataframe containing the summaries with tidyverse code attached

Calculating variable summaries

The aggregateData function accepts any R function which returns a single-value (such as mean, var, sd, sum, IQR). The default name of new variables will be {var}_{fun}, where {var} is the variable name and {fun} is the summary function used. You may pass new names via the varnames argument, which should be either a vector the same length as summary_vars, or a named list (where the names are the summary function). In either case, use {var} to represent the variable name. e.g., {var}_mean or min_{var}.
You can also include the summary `missing`, which will count the number of missing values in the variable. It has default name `{var}_missing`.

For the quantile summary, there is the additional argument `quantiles`. A new variable will be created for each specified quantile `p`. To name these variables, use `{p}` in `varnames` (the default is `{var}_q{p}`).

Custom functions can be passed via the `custom_funs` argument. This should be a list, and each element should have a name and either an `expr` or `fun` element. Expressions should operate on a variable `x`. The function should be a function of `x` and return a single value.

```r
cust_funs <- list(name = '{var}_width', expr = diff(range(x), na.rm = TRUE))
cust_funs <- list(name = '{var}_stderr',
  fun = function(x) {
    s <- sd(x)
    n <- length(x)
    s / sqrt(n)
  }
)
```

**Author(s)**

Tom Elliott, Owen Jin

**See Also**

code
countMissing

**Examples**

```r
aggregatedt <- aggregateData(iris, 
  vars = c("Species"),
  summaries = c("mean", "sd", "iqr")
)
cat(code(aggregatedt))
head(aggregatedt)
```

---

### aggregatedt

#### Aggregate datetimes

**Description**

Aggregate datetimes

**Usage**

`aggregatedt(.data, method, key, name)`
**Arguments**

- `.data` : dataframe or tibble to aggregate
- `method` : the type of aggregation
- `key` : the key column
- `name` : the name of the variable

**Value**

a data frame/tibble

**Author(s)**

Yiwen He

---

**appendrows**

**Append row to the dataset**

**Description**

Append row to the dataset

**Usage**

`appendrows(.data, imported_data, date = FALSE)`

**Arguments**

- `.data` : original dataset
- `imported_data` : imported dataset
- `date` : whether a "When_Added" column is required (default FALSE)

**Value**

dataset with new rows appended

**Author(s)**

Yiwen He
Description
Coerce an object to a survey design by extracting the survey object

Usage
## S3 method for class 'inazvyspec'
as_survey(.data, ...)

Arguments
.data an inazvyspec object
... additional arguments, ignored

Value
a survey design object

Description
Parse survey to survey spec

Usage
as_survey_spec(x)

Arguments
x an object which can be converted to a survey spec (e.g., survey.design)

Value
an inazvyspec file

Methods (by class)
• survey.design: Method for survey.design objects
Author(s)
Tom Elliott

code
Get Data’s Code

Description
Used to grab code from a data.frame generated by this package.

Usage
code(data)

Arguments
data
dataset you want to extract the code from

Details
This is simply a helper function to grab the contents of the ‘code’ attribute contained in the data object.

Value
The code used to generate the data.frame, if available (else NULL)

Author(s)
Tom Elliott

collapseLevels
Collapse data by values of a categorical variable

Description
Collapse several values in a categorical variable into one level

Usage
collapseLevels(.
  .data,
  var,
  levels,
  collapse = paste(levels, collapse = "_"),
  name = sprintf("%s.coll", var)
)

combineCatVars

Arguments

.data a dataframe to collapse
.var a character of the name of the categorical variable to collapse
.levels a character vector of the levels to be collapsed
.collapse name of the newly created level
.name a name for the new variable

Value

the original dataframe containing a new column of the collapsed variable with tidyverse code attached

Author(s)

Owen Jin

See Also

code

Examples

collapsed <- collapseLevels(iris, var = "Species",
    levels = c("setosa", "virginica"))
cat(code(collapsed))
head(collapsed)

Description

Combine specified categorical variables by concatenating their values into one character, and returns the result along with tidyverse code used to generate it.

Usage

combineCatVars(
    .data, vars,
    sep = ".",
    name = paste(vars, collapse = sep),
    keep_empty = FALSE
)
Arguments

.data a dataframe with the columns to be combined
vars a character vector of the categorical variables to be combined
sep the separator to combine the values of the variables in var by. "." by default
name a name for the new variable
keep_empty logical, if FALSE empty level combinations are removed from the factor

Details

When either variable is NA, the result is NA.

Value

original dataframe containing a new column of the renamed categorical variable with tidyverse code attached

Author(s)

Owen Jin

Examples

combined <- combineCatVars(warpbreaks, vars = c("wool", "tension"), sep = ".")
cat(code(combined))
head(combined)

convertToCat

Convert numeric variables to categorical

Description

Convert specified numeric variables into factors

Usage

convertToCat(.data, vars, names = paste(vars, "cat", sep = "."))

Arguments

.data a dataframe with the categorical column to convert
vars a character vector of numeric column names to convert
names a character vector of names for the created variable(s)
Value

   original dataframe containing a new column of the converted numeric variable with tidyverse code attached

Author(s)

   Owen Jin

See Also

code

Examples

   converted <- convertToCat(iris, vars = c("Petal.Width"))
   cat(code(converted))
   head(converted)

convert_to_datetime

Convert to datetime

Description

   Convert to datetime

Usage

   convert_to_datetime(.data, factorname, convname, newname)

Arguments

   .data            dataframe
   factorname      name of the variable
   convname        format
   newname         name of the new column

Value

   dataframe with datetime column

Author(s)

   Yiwen He
createNewVar

Description

Create a new variable by using a valid R expression and returns the result along with tidyverse code used to generate it.

Usage

createNewVar(.data, new_var = "new.variable", R_exp)

Arguments

.data a dataframe to which to add a new variable to
new_var a character of the new variable name. "new.variable" by default
R_exp a character of a valid R expression which can generate a vector of values

countMissing

Count missing values

Description

Count missing values

Usage

countMissing(var, na.rm = FALSE)

Arguments

var the vector to sum up the number of missing values
na.rm ignore this

Value

the number of missing values for that vector

Author(s)

Owen Jin

See Also

aggregateData

createNewVar

Create new variables

Description

Create a new variable by using a valid R expression and returns the result along with tidyverse code used to generate it.

Usage

createNewVar(.data, new_var = "new.variable", R_exp)

Arguments

.data a dataframe to which to add a new variable to
new_var a character of the new variable name. "new.variable" by default
R_exp a character of a valid R expression which can generate a vector of values
create_varname

Value

original dataframe containing the new column created from R_exp with tidyverse code attached

Author(s)

Owen Jin

See Also

code

Examples

created <- createNewVar(iris, new_var = "Sepal.Length_less_Sepal.Width",
   "Sepal.Length - Sepal.Width")
cat(code(created))
head(created)

----------------------------------------
create_varname     Create variable name
----------------------------------------

Description

Convert a given string to a valid R variable name, converting spaces to underscores (_) instead of dots.

Usage

create_varname(x)

Arguments

  x     a string to convert

Value

a string, which is also a valid variable name

Author(s)

Tom Elliott

Examples

create_varname("a new variable")
create_varname("8d4-2q5")
### deleteVars

**Delete variables**

**Description**
Delete variables from a dataset

**Usage**
```r
deleteVars(.data, vars)
```

**Arguments**
- `.data` dataset
- `vars` variables to delete

**Value**
dataset without chosen variables

**Author(s)**
Tom Elliott

### extract_part

**Extract part of a datetimes variable**

**Description**
Extract part of a datetimes variable

**Usage**
```r
extract_part(.data, varname, part, name)
```

**Arguments**
- `.data` dataframe
- `varname` name of the variable
- `part` part of the variable wanted
- `name` name of the new column

**Value**
dataframe with extracted part column
**filterLevels**

*Filter data by levels of a categorical variable*

**Description**

Filter a dataframe by some levels of one categorical variable and returns the result along with tidyverse code used to generate it.

**Usage**

```r
filterLevels(.data, var, levels)
```

**Arguments**

- `.data`: a dataframe or survey design object to filter
- `var`: character of the column in `.data` to filter by
- `levels`: a character vector of levels in `var` to filter by

**Value**

filtered dataframe with tidyverse code attached

**Author(s)**

Yiwen He

**See Also**

code

**Examples**

```r
filtered <- filterLevels(iris, var = "Species",
                        levels = c("versicolor", "virginica"))
cat(code(filtered))
head(filtered)
```
**filterNumeric**

Filter data by levels of a numeric variable

**Description**

Filter a dataframe by some boolean condition of one numeric variable and returns the result along with tidyverse code used to generate it.

**Usage**

```r
filterNumeric(.data, var, op, num)
```

**Arguments**

- `.data` a dataframe or survey design object to filter
- `var` character of the column in `.data` to filter by
- `op` a logical operator of "\(\leq\),\(\), \(\geq\),\(\), \(\approx\) or \(!=\) for the boolean condition
- `num` a number for which the `op` applies to

**Value**

filtered dataframe with tidyverse code attached

**Author(s)**

Owen Jin, Tom Elliott

**See Also**

- `code`

**Examples**

```r
filtered <- filterNumeric(iris, var = "Sepal.Length", op = "\(\leq\)", num = 5)
cat(code(filtered))
head(filtered)

require(survey)
data(api)
svy <- svydesign(~dnum+snum, weights = ~pw, fpc = ~fpc1+fpc2, data = apiclus2)
(svy_filtered <- filterNumeric(svy, var = "api00", op = "\(<\)", num = 700))
cat(code(svy_filtered))
```
filterRandom

*Random sampling without replacement*

**Description**

Take a specified number of groups of observations with fixed group size by sampling without replacement and returns the result along with tidyverse code used to generate it.

**Usage**

```
filterRandom(.data, n, sample_size)
```

**Arguments**

- `.data` a dataframe to sample from
- `n` the number of groups to generate
- `sample_size` the size of each group specified in `n`

**Value**

a dataframe containing the random samples with tidyverse code attached

**Author(s)**

Owen Jin

**See Also**

- `code`

**Examples**

```r
filtered <- filterRandom(iris, n = 5, sample_size = 3)
cat(code(filtered))
head(filtered)
```
filterRows

Filter data by row numbers

Description
Filter a dataframe by slicing off specified rows and returns the result along with tidyverse code used to generate it.

Usage
filterRows(.data, rows)

Arguments
- .data: a dataframe or a survey design object to filter
- rows: a numeric vector of row numbers to slice off

Value
filtered dataframe with tidyverse code attached

Author(s)
Owen Jin

See Also
code

Examples
filtered <- filterRows(iris, rows = c(1,4,5))
cat(code(filtered))
head(filtered)

fitDesign

Fit a survey design

Description
Fit a survey design to an object

Usage
fitDesign(svydes, dataset.name)
**fitModel**

**Arguments**

- `svydes` a design
- `dataset.name` a dataset name

**Value**

a survey object

**Author(s)**

Tom Elliott

---

**fitModel** *Fit models*

---

**Description**

Wrapper function for ‘lm’, ‘glm’, and ‘svyglm’.

**Usage**

```r
fitModel(
  y,
  x,
  data,
  family = "gaussian",
  link = switch(family,
    gaussian = "gaussian",
    binomial = "logit",
    poisson = "log",
    negbin = "log"),
  design = "simple",
  svydes = NA,
  ...
)
```

**Arguments**

- `y` character string representing the response,
- `x` character string of the explanatory variables,
- `data` name of the object containing the data.
- `family` gaussian, binomial, poisson (so far, no others will be added)
- `link` the link function to use
- `design` data design specification. one of `simple`, `survey` or `experiment`
- `svydes` a vector of arguments to be passed to the svydesign function, excluding data (defined above)
- `...` further arguments to be passed to lm, glm, svyglm, such as offset, etc.
Value

A model call formula (using lm, glm, or svyglm)

Author(s)

Tom Elliott

form_class_intervals  Form Class Intervals

Description

Create categorical intervals from a numeric variable.

Usage

form_class_intervals(
  .data,
  variable,
  method = c("equal", "width", "count", "manual"),
  n_intervals = 4L,
  interval_width,
  format = "[a,b]",
  range = NULL,
  format.lowest = ifelse(isinteger, "< a", "<= a"),
  format.highest = "> b",
  break_points = NULL,
  name = sprintf("%s.f", variable)
)

Arguments

.data  the data set
variable  name of the variable to convert
method  one of ‘equal’ for equal-width intervals, ‘width’ for intervals of a specific width, ‘count’ for equal-count intervals, and ‘manual’ to specify break points manually
n_intervals  for methods ‘equal’ and ‘count’, this is the number of intervals to create
interval_width  for method ‘width’, this is the width of intervals
format  the format for intervals; use ‘a’ and ‘b’ to represent the min/max of each interval, respectively.
range  the range of the data; use this to adjust the labels (e.g., for continuous data, set this to floor/ceiling of the min/max of the data to get prettier intervals). If range does not cover the range of the data, values outside will be placed into ‘less than a’ and ‘greater than b’ categories
format.lowest  values lower than the min of range will have this label format
import_survey

format.highest values higher than the max of range will have this label format
break_points for method 'manual', specify breakpoints here (as a numeric vector)
name the name of the new variable in the resulting data set

Value

a dataframe with an additional column with categorical class intervals

Author(s)

Tom Elliott

Examples

form_class_intervals(iris, 'Sepal.Length', 'equal', 5L)

Description

The survey information should be in TOML format, with fields corresponding to survey design components. For example,

strata = strata_var
clusters = cluster_var
weights = wt_var

Usage

import_survey(file, data)

Arguments

file the file containing survey information (see Details)
data optional, if supplied the survey object will be created with the supplied data. Can be either a data.frame-like object, or a path to a data set which will be imported using iNZightTools::smart_read.

Details

For replicate weight designs, vectors (if necessary) are declared with square brackets, like so:

repweights = ['w01', 'w02', 'w03', 'w04', ..., 'w20']

although this would be better expressed using a regular expression,
repweights = '^w[0-2]' 

which matches all variables starting with a w followed by digits between 0 and 2 (inclusive). Additionally, the information can contain a file specification indicating the path to the data, which will be imported using iNZightTools::smart_read if it exists in the same directory as file, or alternatively a URL to a data file that will be downloaded.

Value

a inzsvyspec object containing the design parameters and, if data supplied, the created survey object

Author(s)

Tom Elliott

Description

The iNZightTools package contains a suite of helper functions for iNZight, mostly to make GUI development easier to provide some type of consistency across desktop and shiny versions.

Author(s)

Tom Elliott et al.

See Also

iNZight

is_cat

Is factor check

Description

This function checks if a variable a factor.

Usage

is_cat(x)

Arguments

x the variable to check
**is_dt**

**Value**

logical, TRUE if the variable is a factor

**Author(s)**

Tom Elliott

---

**is_dt**  
*Is datetime check*

**Description**

This function checks if a variable a date/time/datetime

**Usage**

```
is_dt(x)
```

**Arguments**

- `x`  
  the variable to check

**Value**

logical, TRUE if the variable is a datetime

**Author(s)**

Tom Elliott

---

**is_num**  
*Is numeric check*

**Description**

This function checks if a variable is numeric, or could be considered one. For example, dates and times can be treated as numeric, so return TRUE.

**Usage**

```
is_num(x)
```

**Arguments**

- `x`  
  the variable to check
is_survey

Value
logical, TRUE if the variable is numeric

Author(s)
Tom Elliott

is_preview

Description
Checks if the complete file was read or not.

Usage
is_preview(df)

Arguments
df data to check

Value
logical

is_survey

Description
Check if object is a survey object (either standard or replicate design)

Usage
is_survey(x)

Arguments
x object to be tested

Value
logical

Author(s)
Tom Elliott
is_svydesign

Check if object is a survey object (created by svydesign())

Description
Check if object is a survey object (created by svydesign())

Usage
is_svydesign(x)

Arguments
x object to be tested

Value
logical

Author(s)
Tom Elliott

is_svyrep

Check if object is a replicate survey object (created by svrepdesign())

Description
Check if object is a replicate survey object (created by svrepdesign())

Usage
is_svyrep(x)

Arguments
x object to be tested

Value
logical

Author(s)
Tom Elliott
joindata  

Join data with another dataset

Description
Join data with another dataset

Usage

```
joindata(
  .data, 
  imported_data, 
  origin_join_col, 
  import_join_col, 
  join_method, 
  left, 
  right
)
```

Arguments

- `.data`: Original data
- `imported_data`: Imported dataset
- `origin_join_col`: column selected from the original data
- `import_join_col`: column selected from the imported dataset
- `join_method`: function used to join the two datasets
- `left`: suffix name assigned to the original dataset
- `right`: suffix name assigned to the imported dataset

Value

joined dataset

load_rda  

Load object(s) from an Rdata file

Description
Load object(s) from an Rdata file

Usage

```
load_rda(file)
```
make_names

Arguments

  file           path to an rdata file

Value

  list of data frames, plus code

Author(s)

  Tom Elliott

See Also

  save_rda

Description

  Helper function to create new variable names that are unique given a set of existing names (in a data
  set, for example). If a variable name already exists, a number will be appended.

Usage

  make_names(new, existing = character())

Arguments

  new           a vector of proposed new variable names
  existing      a vector of existing variable names

Value

  a vector of unique variable names

Author(s)

  Tom Elliott

Examples

  make_names(c("var_x", "var_y"), c("var_x", "var_z"))
**make_survey**

*Make a survey object*

**Description**
Construct a survey object from a data set and an `inzsvyspec` object.

**Usage**

```r
make_survey(.data, spec)
```

**Arguments**

- `.data` a data.frame
- `spec` a `inzsvyspec` object

**Value**

a `inzsvyspec` object with the survey design loaded

**Author(s)**

Tom Elliott

---

**missingToCat**

*Convert missing values to categorical variables*

**Description**

Turn `<NA>`’s into a "missing" character; hence numeric variables will be converted to categorical variables with any numeric values will be converted to "observed", and returns the result along with tidyverse code used to generate it.

**Usage**

```r
missingToCat(.data, vars, names = paste0(vars, "_miss"))
```

**Arguments**

- `.data` a dataframe with the columns to convert its missing values into categorical
- `vars` a character vector of the variables in `.data` for conversion of missing values to categorical
- `names` a vector of names for the new variables
Value

original dataframe containing new columns of the converted variables for the missing values with tidyverse code attached

Author(s)

Owen Jin

See Also

code

Examples

missing <- missingToCat(iris, vars = c("Species", "Sepal.Length"))
cat(code(missing))
head(missing)

newdevice

Open a New Graphics Device

Description

Opens a new graphics device

Usage

newdevice(width = 7, height = 7, ...)

Arguments

width the width (in inches) of the new device
height the height (in inches) of the new device
... additional arguments passed to the new device function

Details

Depending on the system, difference devices are better. The windows device works fine (for now), only attempt to speed up any other devices that we’re going to be using. We speed them up by getting rid of buffering.

Author(s)

Tom Elliott
print.inzsvyspec  Print iNZigh Survey Spec

Description
Print iNZight Survey Spec

Usage
## S3 method for class 'inzsvyspec'
print(x, ...)

Arguments
x              a inzsvyspec object
...            additional arguments, ignored

Author(s)
Tom Elliott

---

rankVars  Rank the data of a numeric variables

Description
Rank the values of a numeric variable in descending order, and returns the result along with tidyverse code used to generate it. Ties are broken as such: eg. values = 5, 6, 6, 7; rank = 1, 2, 2, 3

Usage
rankVars(.data, vars)

Arguments
.data               a dataframe with the variables to rank
vars                a character vector of numeric variables in .data to rank

Value
the original dataframe containing new columns with the ranks of the variables in var with tidyverse code attached
read_meta

Author(s)
Owen Jin

See Also
code

Examples
ranked <- rankVars(iris, vars = c("Sepal.Length", "Petal.Length"))
cat(code(ranked))
head(ranked)

Description
This function will read a CSV file with iNZight metadata in the header. This allows plain text CSV files to be supplied with additional comments that describe the structure of the data to make import and data handling easier.

Usage
read_meta(file, preview = FALSE, column_types, ...)

Arguments
file the plain text file with metadata
preview logical, if TRUE only the first 10 rows are returned
column_types optional column types
...
more arguments

Details
The main example is to define factor levels for an integer variable in large data sets.

Value
a data frame

Author(s)
Tom Elliott
renameLevels

Description

Rename the levels of a categorical variable, and returns the result along with tidyverse code used to generate it.

Usage

renameLevels(.data, var, to_be_renamed, name = sprintf("%s.rename", var))

Arguments

.data a dataframe with the column to be renamed
var a character of the categorical variable to rename
to_be_renamed a list of the old level name assigned to the new level name; i.e., ‘list(new level name’ = ‘old level name’)
name a name for the new variable
renameVars

Value
original dataframe containing a new column of the renamed categorical variable with tidyverse code attached

Author(s)
Owen Jin

See Also
code

Examples
renamed <- renameLevels(iris, var = "Species",
  to_be_renamed = list(set = "setosa", ver = "versicolor"))
cat(code(renamed))
head(renamed)

renameVars .data, to_be_renamed_list

Description
Rename column names and returns the result along with tidyverse code used to generate it.

Usage
renameVars(.data, to_be_renamed_list)

Arguments
.data a dataframe with columns to rename
to_be_renamed_list a list of the new column names assigned to the old column names ie. list('old column names' = 'new column names')

Value
original dataframe containing new columns of the renamed columns with tidyverse code attached

Author(s)
Owen Jin

See Also
code
Examples

renamed <- renameVars(iris,
  to_be_renamed_list = list(Species = "Type", Petal.Width = "P.W")))
cat(code(renamed))
head(renamed)

reorderLevels

Reorder a categorical

Description

Reorder the factors of a categorical variable either manually or frequency

Usage

reorderLevels(
  .data,
  var,
  new_levels = NULL,
  freq = FALSE,
  name = sprintf("%s.reord", var)
)

Arguments

.data a dataframe to reorder
var a categorical variable to reorder
new_levels a character vector of the new factor order. Only specify if freq = FALSE
freq logical, If freq = FALSE (default), will manually reorder using new_levels. If freq = TRUE, will reorder based of descending frequency of the factor levels
name name for the new variable

Value

original dataframe containing a new column of the reordered categorical variable with tidyverse code attached

Author(s)

Owen Jin

See Also

code
**reshape_data**

Reshaping dataset from wide to long or from long to wide

**Description**

Reshaping dataset from wide to long or from long to wide

**Usage**

```r
reshape_data(.data, col1, col2, cols, key, value, check)
```

**Arguments**

- `.data`: dataset
- `col1`: column to spread out (for long to wide)
- `col2`: values to be put in the spread out column (for long to wide)
- `cols`: columns(s) to gather together (for wide to long)
- `key`: name for new column containing old column names (for wide to long)
- `value`: name for new column containing old column values (for wide to long)
- `check`: check whether to use long to wide or wide to long

**Value**

Reshaped dataset

**Author(s)**

Yiwen He
**save_rda**

*Save an object with, optionally, a (valid) name*

**Description**

Save an object with, optionally, a (valid) name

**Usage**

```r
save_rda(data, file, name)
```

**Arguments**

- `data`: the data frame to save
- `file`: where to save it
- `name`: optional, the name the data will have in the rda file

**Value**

logical, should be TRUE, along with code for the save

**Author(s)**

Tom Elliott

**See Also**

`load_rda`

---

**selectVars**

*Select variables from a dataset*

**Description**

Select a (reordered) subset of variables from a subset.

**Usage**

```r
selectVars(.data, keep)
```

**Arguments**

- `data`: the dataset
- `keep`: vector of variable names to keep
separate

Value

a data frame with tidyverse code attribute

Author(s)

Tom Elliott

Examples

```
selectVars(iris, c("Sepal.Length", "Species", "Sepal.Width"))
```

---

**Description**

Separate columns

**Usage**

```
separate(.data, col, left, right, sep, check)
```

**Arguments**

- `.data` dataset
- `col` column to be separated
- `left` name for the separated left column
- `right` name for the separated right column
- `sep` separator used to separate columns
- `check` method of separating

**Value**

separated dataset

**Author(s)**

Yiwen He, Tom Elliott
**sheets**

*List of available sheets from a file*

**Description**

List of available sheets from a file

**Usage**

`sheets(x)`

**Arguments**

- `x` a dataframe from `smart_read`

**Value**

vector of sheet names, or NULL

**Author(s)**

Tom Elliott

---

**smart_read**

*iNZight Smart Read*

**Description**

A simple function that magically imports a file, irrespective of type.

**Usage**

```r
smart_read(
  file,
  ext = tools::file_ext(file),
  preview = FALSE,
  column_types = NULL,
  ...
)
```

**Arguments**

- `file` the file path to read
- `ext` file extension, namely "csv" or "txt"
- `preview` logical, if TRUE only the first few rows of the data will be returned
- `column_types` vector of column types (see `readr::read_csv`)
- `...` additional parameters passed to `read_*` functions
Details
The smart read function understands the following:

- delimited (.csv, .txt)
- excel files (.xls, .xlsx)
- spss files (.sav)
- stata files (.dta)
- SAS files (.sas7bdat, .xpt)
- R data files (.rds)
- JSON files (.json)

Value
a dataframe with attributes

Author(s)
Tom Elliott

sortVars | Sort data by variables

Description
Sorts a dataframe by one or more variables, and returns the result along with tidyverse code used to generate it.

Usage
sortVars(.data, vars, asc = rep(TRUE, length(vars)))

Arguments
- .data a dataframe to sort
- vars a character vector of variable names to sort by
- asc logical, same length as vars. If TRUE (default), sorted in ascending order, otherwise descending.

Value
data.frame with tidyverse code attached

Author(s)
Owen Jin
See Also

code

Examples

```r
sorted <- sortVars(iris, vars = c("Sepal.Width", "Sepal.Length"),
  asc = c(TRUE, FALSE))
cat(code(sorted))
head(sorted)
```

---

### stackVars

**Stack variables**

Collapse columns by converting from a wide to a long format and returns the result along with tidyverse code used to generate it.

#### Usage

```r
stackVars(.data, vars, key = "stack.variable", value = "stack.value")
```

#### Arguments

- `.data` a dataframe to stack
- `vars` a character vector of variables to stack
- `key` name of the new column for the stacked variables. "stack.variable" by default
- `value` name of the new column for the stacked values of the stacked. "stack.value" by default

#### Value

stacked dataframe with tidyverse code attached

#### Author(s)

Owen Jin

#### See Also

code
Examples

```r
standardized <- standardizeVars(iris, var = c("Sepal.Width", "Petal.Width"))
cat(code(standardized))
head(standardized)
```

---

**standardizeVars**  
*Standardize the data of a numeric variable*

**Description**

Centre then divide by the standard error of the values in a numeric variable

**Usage**

```r
standardizeVars(.data, vars, names = paste(sep = ".", vars, "std"))
```

**Arguments**

- `.data`  
a dataframe with the columns to standardize
- `vars`  
a character vector of the numeric variables in `.data` to standardize
- `names`  
names for the created variables

**Value**

the original dataframe containing new columns of the standardized variables with tidyverse code attached

**Author(s)**

Owen Jin, Tom Elliott

**See Also**

code

**Examples**

```r
standardized <- standardizeVars(iris, var = c("Sepal.Width", "Petal.Width"))
cat(code(standardized))
head(standardized)
```
**survey_IQR**  
*Interquartile range function for surveys*

**Description**

Calculates the interquartile range from complex survey data. A wrapper for taking differences of `svyquantile` at 0.25 and 0.75 quantiles, and meant to be called from within `summarize` (see `srvyr` package).

**Usage**

```r
survey_IQR(x, na.rm = TRUE)
```

**Arguments**

- `x`: A variable or expression
- `na.rm`: logical, if TRUE missing values are removed

**Value**

a vector of interquartile ranges

**Author(s)**

Tom Elliott

**Examples**

```r
library(survey)
library(srvyr)
data(api)

dstrata <- apistrat %>%
as_survey(strata = stype, weights = pw)

dstrata %>%
  summarise(api99_iqr = survey_IQR(api99))
```
**tidy_all_code**

**Description**

Tidy code with correct indents and limit the code to the specific width

**Usage**

```r
tidy_all_code(x, width = 80, indent = 4, outfile, incl_library = TRUE)
```

**Arguments**

- `x`: character string or file name of the file containing messy code
- `width`: the width of a line
- `indent`: how many spaces for one indent
- `outfile`: the file name of the file containing formatted code
- `incl_library`: logical, if true, the output code will contain library name

**Value**

formatted code, optionally written to `outfile`

**Author(s)**

Tom Elliott, Lushi Cai

---

**transformVar**

**Transform data of a numeric variable**

**Description**

Transform the values of a numeric variable by applying a mathematical function

**Usage**

```r
transformVar(
  .data,
  var,
  transformation,
  name = sprintf("%s.%s", transformation, var)
)
```
Arguments

.data a dataframe with the variables to transform
var a character of the numeric variable in .data to transform
transformation a name of a valid mathematical function that can be applied to numeric values, eg. "log", "exp", "sqrt". For squaring, use "square"; for inverting, use "reciprocal"
name the name of the new variable

Value

the original dataframe containing a new column of the transformed variable with tidyverse code attached

Author(s)

Owen Jin

See Also

code

Examples

transformed <- transformVar(iris, var = "Petal.Length",
transformation = "log")
cat(code(transformed))
head(transformed)

---

unite Unite columns in a dataset

Description

Unite columns in a dataset

Usage

unite(.data, name, col, sep)

Arguments

.data dataset
name name for the new united column
col a vector of column names
sep separator used in between the united columns
**url_to_temp**  

**Value**  
united dataset  

**Author(s)**  
Yiwen He  

---

**description**  
**Download URL to temp file**  

**Description**  
Download URL to temp file  

**Usage**  
url_to_temp(url)  

**Arguments**  
url where the file lives on the internet  

**Value**  
the location of a (temporary) file location  

**Author(s)**  
Tom Elliott  

---

**validation_details**  
**Details of Validation Rule Results**  

**Description**  
Generates the more detailed text required for the details section in iNZValidateWin.  

**Usage**  
validation_details(cf, v, var, id.var, df)
validation_summary

**Arguments**

- **cf**: Confrontation object from `validate::confront()`
- **v**: Validator that generated `cf`
- **var**: Rule name to give details about
- **id.var**: Variable name denoting a unique identifier for each observation
- **df**: The dataset that was confronted

**Value**

A character vector giving each line of the summary detail text

**Author(s)**

Daniel Barnett

---

validation_summary  Validation Confrontation Summary

**Description**

Generates a summary of a confrontation which gives basic information about each validation rule tested.

**Usage**

`validation_summary(cf)`

**Arguments**

- **cf**: Confrontation object from `validate::confront()`

**Value**

A `data.frame` with number of tests performed, number of passes, number of failures, and failure percentage for each validation rule.

**Author(s)**

Daniel Barnett
vartype

**Description**
Get variable type name

**Usage**
vartype(x)

**Arguments**
x vector to be examined

**Value**
character vector of the variable's type

**Author(s)**
Tom Elliott

%notin%

**Description**
Anti value matching

**Usage**
x %notin% table

**Arguments**
x vector of values to be matched
table vector of values to match against

**Value**
A logical vector of same length as `x`, indicating if each element does **not** exist in the table.
Index

* iNZight
  iNZightTools, 22
  %notin%, 47
add_suffix, 3
aggregateData, 4, 12
aggregatedt, 5
appendrows, 6
as_survey, 7
as_survey_spec, 7
code, 5, 8, 9, 11, 13, 15–18, 29, 31, 33, 34, 40, 41, 44
collapseLevels, 8
combineCatVars, 9
convert_to_datetime, 11
convertToCat, 10
countMissing, 5, 12
create_varname, 13
createNewVar, 12
deleteVars, 14
extract_part, 14
filterLevels, 15
filterNumeric, 16
filterRandom, 17
filterRows, 18
fitDesign, 18
fitModel, 19
form_class_intervals, 20
import_survey, 21
iNZightTools, 22
is_cat, 22
is_dt, 23
is_num, 23
is_preview, 24
is_survey, 24
is_svydesign, 25
is_svyrep, 25
joindata, 26
load_rda, 26, 36
make_names, 27
make_survey, 28
missingToCat, 28
newdevice, 29
print.inzsvyspec, 30
rankVars, 30
read_meta, 31
read_text, 32
renameLevels, 32
renameVars, 33
reorderLevels, 34
reshape_data, 35
save_rda, 27, 36
selectVars, 36
separate, 37
sheets, 38
smart_read, 38
sortVars, 39
srvyr, 42
stackVars, 40
standardizeVars, 41
survey_IQR, 42
tidy_all_code, 43
transformVar, 43
unite, 44
url_to_temp, 45
validation_details, 45
validation_summary, 46
vartype, 47