Package 'hypegrammaR'

December 10, 2019

Title A grammar of hypothesis test driven analysis

Version 0.2.2

Description Quantitative analysis according to the IMPACT minimum standards. Accepts weights and input from kobo questionnaires.

Depends R (>= 3.5.2),
survey,
magrittr,
ggplot2,
ggthemes,
tidyr,
crayon,
grid,
stats

Imports dplyr,
reshape2,
questionr,
koboquest (>= 1.0.1),
kobostandards,
data.table,
surveyweights (>= 0.1.0),
knitr,
htmltools,
assertthat,
kableExtra,
parallel,
srvyr,
extrafont,
purrr,
scales

Remotes github::mabafaba/koboquest,
github::mabafaba/xlsformfill,
github::mabafaba/kobostandards,
github::mabafaba/composr,
github::ellieallien/surveyweights,
github::mabafaba/koboquest

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat,
rmarkdown,
tibble

VignetteBuilder knitr

R topics documented:

`hypegrammaR-package` *hypegrammaR: Implementing* R*hrefhttps://docs.google.com/document/d/1979hEu6N9d_nIHw_eXys808q81WewZ0/edit#heading=h.gjdgxsIMPACT Data Analysis Guidelines*

Description

A grammar of hypothesis driven analysis, following the idea that there is only one test

Details

Supports integration of weighted data (using the survey package) and data collected with kobotoolbox, ODK or similar. Executes the three main steps of data analysis

- summary statistics
- hypothesis tests
- preparation for visualisation

The user begins by loading the data, and if needed the questionnaire, analysisplan and sampling frame (as .csv files). To verify the correct format of these inputs, `kobostandards::check_inputs` can be used. All other functions then refer to these objects.

The two possible work-flows are: using the individual functions (in the blocks), and mapping to the resuls: using all the blocks automatically.

Mappig to the result

- `map_to_result` is the overall function that executes the three steps of data analysis
- `map_to_file` maps from the result to a csv or jpg file
- `from_analysisplan_map_to_output` applies map_to_result and map_to_file for all rows in the data analysis plan

Individual blocks

Preparing your data

- map_to_case,
- map_to_design,
- map_to_weighting

Summary statistics (examples)

- percent_with_confints_select_one,
- mean_with_confints_select_one_groups

Hypothesis tests (examples)

- hypothesis_test_t_one_sample,
- hypothesis_test_chisquared_select_one

Visualise (examples)

- barchart_percent,
- gg_heatmap_generic

Author(s)

Maintainer: Eliora Henzler <eliora.henzler@impact-initiatives.org>

`analysisplan_expand_repeat`

These are wrappers to run many analysis at once based on an analysis plan In hypegrammaR, you identify a dependent variable, an independent variable, an analysis case and a hypothesis type. The analyssiplan basically collects that information in a table that can then be applied in batch.

Description

These are wrappers to run many analysis at once based on an analysis plan In hypegrammaR, you identify a dependent variable, an independent variable, an analysis case and a hypothesis type. The analyssiplan basically collects that information in a table that can then be applied in batch.

Usage

```
analysisplan_expand_repeat(analysisplan, data)
```

Arguments

`analysisplan`	an analysis plan as a data frame
`data`	a datasets matching the analysisplan

Details

expand an analysis plan with repeat variable

an analysisplan can contain a "repeat.var" column. This function expands each analysisplan row with a repeat variable into one row per unique value in the dataset for that variable; for example if the repeat variable is "district", and in the data there are the districts "d_1" and "d_2" this will create a new analysisplan with two rows where "repeat.var.value" is "d_1" and "d_2" respectively

---

apply_data_sanitations

chaining sanitation functions

Description

chaining sanitation functions

Usage

```
apply_data_sanitations(data, dependent.var, independent.var, ...)
```

Arguments

data	the data
dependent.var	the name of the dependent variable
independent.var	the name of the independent variable
...	any number of sanitation functions
BEFORE	optional: overwrite the default checks that are always used in the beginning (not recommended, used mainly to break the recursion)
AFTER	optional: overwrite the default checks that are always used in the end (not recommended, used mainly to break the recursion)

---

assert_valid_analysisplan

check if an analysisplan is in valid format

Description

check if an analysisplan is in valid format

Usage

```
assert_valid_analysisplan(df)
```

`combine_weighting_functions` *Combine weight functions from two sampling frames*

Description

Combine weight functions from two sampling frames

Usage

```
combine_weighting_functions(weight_function_1, weight_function_2)
```

Arguments

`weight_function_1`	first weighthing function
`weight_function_2`	second weightng function

Value

returns a new function that takes a data frame as input returns a vector of weights corresponding to each row in the data frame.

`datasanitation_design` *"high level" data sanitations do the following:*

Description

- take a design object and some information on the analysis to be conducted - apply a number of low lever data sanitations such as checking the number of unique values in a variable (data_sanitation_low_level.R) - return a logical whether the sanitation was successful plus either the design object or a message explaining why the sanitation failed

Usage

```
datasanitation_design(design, dependent.var, independent.var,
  sanitation_function)
```

Arguments

`design`	the design object
`dependent.var`	a string containing the dependent variable in the analysis case
`independent.var`	a string containing the independent variable in the analysis case
`sanitation_function`	the function containing all the checks for the analysis function in question

Details

We do this mainly because the survey package is very strict and throws a lot of errors that we are trying to catch up front the code is structured as follows:

- there's some "low level" check functions - apply_data_sanitations() can be used to chain any number of those checks - that is then used to create "high level" sanitation functions matching specific "Blocks", for example a specific summary statistic function. Applies basic sanitation to data before summary statistics or hypothesis test can be applied

Value

returns the cleaned data with a santation success or failure message

---

datasanitation_is_good_dataframe

individual low level sanitation checks ———————————————————————————

Description

low level sanitation checks make sure a very specific condition in the data is fulfilled. They are then chained together into task specific combinations of checks (see data_sanitation_high_level)

Usage

```
datasanitation_is_good_dataframe(data, ...)
```

Details

each data sanitation does the following:

1. check if the specific sanitation check is fulfilled 2. if yes, return a successfull_sanitation object which includes the data as is 3. if no, try to modify the dataset to fix the issue 4. if successful, return a successful_sanitation object which includes the dataset 5. if not successful, return a failed_sanitation object check if data is a data frame

---

datasanitation_return_empty_table

creates a summary.statistic table with all numbers NA in the standard format you would expect from map_to_summary_statistic

---

Description

creates a summary.statistic table with all numbers NA in the standard format you would expect from map_to_summary_statistic

Usage

```
datasanitation_return_empty_table(data, dependent.var,
  independent.var = NULL, message = NA)
```

Arguments

data	the input data
dependent.var	the name of the dependent variable
independent.var	the name of the indepenent variable
message	a character string explaining why we are returning an empty table

---

`from_analysisplan_map_to_output` *apply an analysis plan*

Description

Takes all usual hypegrammaR input files plus an analysis plan and maps directly to an output document

Usage

```
from_analysisplan_map_to_output(data, analysisplan, weighting = NULL,
  cluster_variable_name = NULL, questionnaire = NULL,
  labeled = FALSE, verbose = TRUE, confidence_level = 0.95)
```

Arguments

data	the data set as a data frame (load_data())
analysisplan	the analysisplan (load_analysisplan())
weighting	optional: the weighting function (use load_samplingframe() and then map_to_weighting())
cluster_variable_name	optional: the name of the variable with the cluster IDs
questionnaire	optional: the questionnaire (load_questionnaire())
labeled	do you want the resuts to display labels rather than xml names ? defaults to false, requires the questionnaire
verbose	should progress be printed to the console? (default TRUE, slightly faster if FALSE)
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Value

returns a list of hypegrammaR "result" objects (see map_to_result())

from_result_map_to_md_table *create a result table in markdown format*

Description

create a result table in markdown format

Usage

```
from_result_map_to_md_table(result)
```

Arguments

result	a result as produced by map_to_result()

Value

a character string with markdown syntax.

gg_heatmap_generic *a heatmap with the styles that we always want applied*

Description

a heatmap with the styles that we always want applied

Usage

```
gg_heatmap_generic(summary.statistic)
```

grouped_barchart_percent *Grouped barchart for percentages*

Description

Grouped barchart for percentages

Usage

```
grouped_barchart_percent(result)
```

hypothesis_test_chisquared_select_multiple

Perform a chi squared test on a select multiple question against a select one question.

Description

Perform a chi squared test on a select multiple question against a select one question.

Usage

```
hypothesis_test_chisquared_select_multiple(dependent.var,
  dependent.var.sm.cols, independent.var, design, questionnaire = NULL)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a ’select multiple’.
design	the svy design object created using map_to_design or directly with svydesign
independen.var	string with the column name in ‘data‘ of the independent variable. Should be a ’select one’ with few (<15) categories.

Value

A list with the results of the test (Chi Squared statistics, p value) or the error message.

Examples

```
## Not run: hypothesis_test_chisquared_select_one("population_group", "resp_gender", design)
```

hypothesis_test_chisquared_select_one

Based on the analysis case, map_to_hypothesis_test() will pick out one of these hypothesis test functions.

Description

they all have the same pattern

Usage

```
hypothesis_test_chisquared_select_one(dependent.var, independent.var,
  design)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a ’select one’.
design	the svy design object created using map_to_design or directly with svydesign
independen.var	string with the column name in ‘data‘ of the independent variable. Should be a ’select one’ with few (<15) categories.

Details

1. Start with a sanitation: checking if the data is ok and modifying it if necessary. see data_sanitation_high_level.R for details) 2. if sanitation failed, return an empty result 3. make a formula for the test 4. apply the test; on failure, try not to throw errors, but instead an empty result with a hopefully meaningful message 5. return a result in a standard format; all of these functions should _always_ return an object of exactly the same structure

hypothesis_test_chisquared_select_one Perform a chi squared test on a select one question against another.

Value

A list with the results of the test (Chi Squared statistics, p value) or the error message.

Examples

```
## Not run: hypothesis_test_chisquared_select_one("population_group", "resp_gender", design)
```

---

hypothesis_test_empty *make an empty hypothesis test result in standard format*

Description

make an empty hypothesis test result in standard format

Usage

```
hypothesis_test_empty(dependent.var = NULL, independent.var = NULL,
  design = NULL, message = "No hypothesis test", ...)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a ’select one’.
design	the svy design object created using map_to_design or directly with svydesign
independen.var	string with the column name in ‘data‘ of the independent variable. Should be a ’select one’ with few (<15) categories.

---

hypothesis_test_t_one_sample *Perform a one sample t test of one numerical variable against hypothesised value (limit)*

Description

Perform a one sample t test of one numerical variable against hypothesised value (limit)

Usage

```
hypothesis_test_t_one_sample(dependent.var, independent.var = NULL,
  limit, design)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be numerical.
independent.var	should be null ! For other functions: string with the column name in ‘data‘ of the independent variable
limit	the value to test the dependent.var against
design	the svy design object created using map_to_design or directly with svydesign

Value

A list with the results of the test (T-value, p value, etc.) or the error message.

Examples

```
## Not run: hypothesis_test_t_two_sample("males_13_15", 4, design)
```

---

hypothesis_test_t_two_sample

Perform a two sample t test of one numerical variable across mutliple groups

---

Description

Perform a two sample t test of one numerical variable across mutliple groups

Usage

```
hypothesis_test_t_two_sample(dependent.var, independent.var, design)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be numerical.
design	the svy design object created using map_to_design or directly with svydesign
independen.var	string with the column name in ‘data‘ of the independent variable. Should be a ’select one’ with few (<15) categories.

Value

A list with the results of the test (T-value, p value, etc.) or the error message.

Examples

```
## Not run: hypothesis_test_t_two_sample("males_13_15", "resp_gender", design)
```

labels_summary_statistic *Add labels to results*

Description

Add labels to results

Usage

```
labels_summary_statistic(summary.statistic, questionnaire,
  label.dependent.var.value = T, label.independent.var.value = T,
  label.dependent.var = T, label.independent.var = T,
  independent.linebreak = T, dependent.linebreak = F)
```

Arguments

questionnaire	koboquest 'questionnaire' object; output from load_questionnaire()
result	hypegrammaR 'result' object; output from map_to_result().

Details

if the Variable wasn't found in the questionnaire, or the choice wasn't found in the corresponding list of choices, the affected values will remain unchanged.

Value

same as input, but with all variable values labeled

label_pvalue *presentable p-value format*

Description

presentable p-value format

Usage

```
label_pvalue(x, digits = 3)
```

`list_all_cases` *the "analysis case" is the minimal info required to know what hypothesis test, summary statistic and visualisation would be most appropriate. it's stored as a string with a class.*

Description

the "analysis case" is the minimal info required to know what hypothesis test, summary statistic and visualisation would be most appropriate. it's stored as a string with a class.

Usage

```
list_all_cases(implemented_only = F)
```

`load_analysisplan` *Load an analysis plan from a csv file*

Description

Load an analysis plan from a csv file

Usage

```
load_analysisplan(file = NULL, df = NULL)
```

Arguments

`file`	path to a csv file with the analysis plan
`df`	alternative to 'file', you can provide the analysis plan as a data frame

Details

The analysis plan csv file must contain the following column headers: "repeat.for.variable","research.question", "sub.research.question", "hypothesis", "independent.variable", "dependent.variable", "hypothesis.type", "independent.variable.type", "dependent.variable.type". This is mainly relevant if hypegrammaR is used in the background of an interface where people don't load and their data into R manually.

load_data *load asessment data*

Description

load asessment data

Usage

```
load_data(file)
```

Arguments

file	path to a csv file with the assessment data

Details

the data _must_ be in standard kobo format with xml style headers. This is mainly relevant if hypegrammaR is used in the background of an interface where people don't load and their data into R manually.

Value

the data from the csv files as data frame. Column header symbols are changed to lowercase alphanumeric and underscore; everything else is converted to a "."

load_questionnaire *load_questionnaire*

Description

load_questionnaire

Usage

```
load_questionnaire(data, questions, choices,
  choices.label.column.to.use = NULL)
```

Arguments

data	data frame containing the data matching the questionnaire to be loaded.
questions	data frame or file name of a csv file containing the kobo form's question sheet
choices	data frame or file name of a csv file containing the kobo form's choices sheet
choices.label.column.to.use	The choices csv file has (sometimes multiple) columns with labels. They are often called "Label::English" or similar. Here you need to provide the _name of the column_ that you want to use for labels (see example!)

Value

A list containing the original questionnaire questions and choices, the choices matched 1:1 with the data columns, and all functions created by this function relating to the specific questionnaire (they are written to the global space too, but you can use these when using multiple questionnaires in parallel.)

Examples

```
## Not run:
load_questionnaire(mydata,
                  questions.file="koboquestions.csv",
                  choices.file="kobochoices.csv",
                  choices.label.column.to.use="Label::English")

## End(Not run)
```

`load_samplingframe` *Load a sampling frame from csv*

Description

Load a sampling frame from csv

Usage

```
load_samplingframe(file)
```

Arguments

`file`	the path and name of the sampling frame csv file to load.

Details

function loads the sampling frame and can be used to make weights ith map_to_weighting(). This is mainly relevant if hypegrammaR is used in the background of an interface where people don't load and their data into R manually.

Examples

```
## Not run: sf <- load_samplingframe("./somefolder/samplingframe.csv")
```

map_to_case *Map to case*

Description

creates a string that other functions can use to know what analysis case they are dealing with

Usage

```
map_to_case(hypothesis.type, dependent.var.type = NULL,
  independent.var.type = NULL)
```

Arguments

hypothesis.type	The hypothesis type. Must be one of "group_difference" or "direct_reporting".
dependent.var.type	The type of the dependent variable as a string. must be either "numerical" or "categorical"
independent.var.type	The type of the independent variable as a string. must be either "numerical" or "categorical"

Value

a string that other functions can use to know what analysis case they are dealing with. It has a class "analysis_case" assigned

Examples

```
## Not run: map_to_case("group_difference","categorical","categorical")
```

map_to_datamerge *produce a table that can be used for indesign data merge*

Description

produce a table that can be used for indesign data merge

Usage

```
map_to_datamerge(results, rows = c("repeat.var", "repeat.var.value"),
  values = "numbers", ignore = c("se", "min", "max"),
  questionnaire = NULL)
```

Arguments

results	a list with an analysisplan and a matching list of results (as produced by from_analysisplan_map_to_o)
rows	which analysisplan columns should define what to have per row
values	default "numbers" is almost always what you want
ignore	which columns not to include
questionnaire	an object created with load_questionnaire() (used to produce labels if required)

---

map_to_design *Map to Design*

Description

creates a ‘survey‘ design object from the data

Usage

```
map_to_design(data, cluster_variable_name = NULL,
  weighting_function = NULL)
```

Arguments

data	the dataset as a sampling frame. Must match the sampling frame provided to create the ‘weighting_function‘ produced with ‘map_to_weighting()‘
weighting_function	if cluster sampling was used, what’s the name of the column in ‘data‘ that identifies the cluster?

Details

create a ‘survey‘ package design object from the data and information on the sampling strategy

Value

a ‘survey‘ package design object

Examples

```
## Not run: map_to_design(data,cluster_variable_name="cluster_id")
```

map_to_file *Save outputs to files*

Description

Save outputs to files

Usage

```
map_to_file(object, filename, ...)
```

Arguments

object	The object you want to save as a file
filename	The name of the file that is produced. The extension needs to match the type of object you want to save (csv for tables, jpg/pdf for images)

Value

the object that was given as input (unchanged).

Examples

```
## Not run: # some table:
mytable<-data.frame(a=1:10,b=1:10)
map_to_file(mytable,"mytable.csv")

# some graphic made with ggplot:
mygraphic<-ggplot(mytable,aes(a,b))+geom_point()
map_to_file(mygraphic,"visualisation.jpg")
map_to_file(mygraphic,"visualisation.pdf")
## End(Not run)
```

map_to_generic_hierarchical_html *html from resultlist with results in specified hierarchical order based on analysisplan*

Description

html from resultlist with results in specified hierarchical order based on analysisplan

Usage

```
map_to_generic_hierarchical_html(resultlist, render_result_with,
  by_analysisplan_columns = c("dependent.var"), by_prefix = c("",
  "subset:", "variable:"), level = 2, questionnaire = NULL,
  label_varnames = TRUE, dir = "./", filename)
```

Arguments

resultlist	structure like the output from from_analysisplan_map_to_output: A list with two items "analysisplan" and "results": The "analysisplan" as a data frame, where each row must match a result in a list of "results"
render_result_with	a function that takes a single result as input and returns an rmarkdown formated string
by_analysisplan_columns	vector of strings matching column names of the analysisplan. The first element becomes the main heading, the second element the sub-heading etc.
by_prefix	a prefix added at the beginnig of the headline; same length as ‘by_analysisplan_columns‘
level	the markdown header level to start with; defaults to 2 which leads to "## heading", i.e. the second header level.
questionnaire	optional; the questionnaire (koboquest::load_questionnaire())
label_varnames	wether variables names should be labeled in headings
dir	the directory in which to save the output file (absolute path or relative to current working directory)
filename	the name of the file. must end in ’.html’
type	the type of report template to use. Currently one of "full", "visual" or "summary"

---

map_to_hypothesis_test

map to hypothesis test

Description

selects an appropriate hypothesis test function based on the analysis case

Usage

```
map_to_hypothesis_test(design, dependent.var, independent.var, case,
  questionnaire = NULL, limit = NULL)
```

Arguments

case	a string uniquely identifying the analysis case. output of map_to_case().

Value

a _function_ that computes the relevant hypothesis test

map_to_labeled *Add labels to results based on the questionnaire*

Description

Add labels to results based on the questionnaire

Usage

```
map_to_labeled(result, questionnaire)
```

Arguments

result	hypegrammaR 'result' object; output from map_to_result().
questionnaire	koboquest 'questionnaire' object; output from load_questionnaire()

Details

if the variable wasn't found in the questionnaire, or the choice wasn't found in the corresponding list of choices, the affected values will remain unchanged.

Value

same as 'result' input, but with all variable values labeled

map_to_master_table *Make the master table of summary stats and hypothesis tests*

Description

Make the master table of summary stats and hypothesis tests

Usage

```
map_to_master_table(results_object, filename, questionnaire = NULL)
```

Arguments

results_object	a list containing one or more hypegrammaR result objects: the output of map_to_result
filename	The name of the file that is produced. The extension needs to be ".csv".
questionnaire	optional: the questionnaire obtained by load_questionnaire. Necessary is you want labeled results

Value

a dataframe containing the summary statistics and p values for each element in results.

map_to_result *Map to results from data, variable names & case*

Description

Produce summary statistics, hypothesis tests and plot objects for a hypothesis

Usage

```
map_to_result(data, dependent.var, independent.var = NULL, case,
  cluster.variable.name = NULL, weighting = function(df) {     rep(1,
  nrow(df)) }, questionnaire = NULL, confidence_level = 0.95)
```

Arguments

data	the data as a data.frame. Must match the sampling frame used to produce the 'weighting' as well as the questionnaire if applicable.
dependent.var	string with the column name in "data" of the dependent variable
case	the analysis case, created with map_to_case().
cluster.variable.name	if cluster sampling, provide the name of the variable in the dataset that denotes the cluster
weighting	A function that generates weights from a dataframe. You can create it with surveyweights::weighting_fun_from_samplingframe()
questionnaire	output from load_questionnaire()
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)
independen.var	string with the column name in 'data' of the independent variable

Details

- takes as parameters outputs from - load_data() - map_to_case() - load_samplingframe() - load_questionnaire() - output can be processed by: - map_to_labeled() - map_to_visualisation() - map_to_table() - map_to_master_table() - map_to_visualisation_heatmap()

Value

A list with the summary.statistic the hypothesis.test result

map_to_summary_statistic *Map to summary statistic*

Description

selects an appropriate summary statistic function based on the analysis case

Usage

```
map_to_summary_statistic(design, dependent.var, independent.var, case,
  questionnaire = NULL, confidence_level = 0.95)
```

Arguments

design	the design object (map_to_design())
dependent.var	the name of the dependent variable
independent.var	the name of the independent variable
case	a string uniquely identifying the analysis case. output of map_to_case().
questionnaire	the questionnaire (from load_questionnaire())
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Value

a _function_ that computes the relevant summary statistic

Examples

```
## Not run: map_to_summary_statistic("group_difference_categorical_categorical")
## Not run: my_case<- map_to_case( ... )
my_sumstat <- map_to_summary_statistic(my_case)
my_sumstat( ... )
## End(Not run)
```

map_to_summary_table *Make one big csv table from a list of results' summary satistics*

Description

Make one big csv table from a list of results' summary satistics

Usage

```
map_to_summary_table(results_object, filename, questionnaire = NULL)
```

Arguments

results_object	a list containing one or more hypegrammaR result objects: the output of map_to_result
filename	The name of the file that is produced. The extension needs to be ".csv".
questionnaire	optional: the questionnaire obtained by load_questionnaire. Necessary is you want labeled results

Value

a dataframe containing the summary statistics for each element in results.

---

map_to_table *results as a table*

Description

results as a table

Usage

```
map_to_table(result)
```

Arguments

result	a hypegrammaR ‘result‘ object produced by map_to_result

Value

a date frame with only the summary statistics

---

map_to_template *Map results to an output template*

Description

Map results to an output template

Usage

```
map_to_template(x, questionnaire = NULL, dir, type = NULL, filename,
  custom_template = NULL)
```

Arguments

x	hypegrammaR result or list of results (created with map_to_result() or from_analysisplan_map_to_ou
questionnaire	optional: the questionnaire (load_questionnaire())
dir	the directory in which to save the output file (absolute path or relative to current working directory)
type	the type of report template to use, as a string. Currently one of "full", "visual" or "summary". Can be omitted if custom template is used
filename	the name of the file. must end in '.html'
custom_template	optional: the full path to the custom template to use (must be an RMD file in the templates folder)

Details

This function does the following steps:

---

map_to_visualisation *map to visualisation*

Description

selects an appropriate visualisation function based on the analysis case

Usage

```
map_to_visualisation(result)
```

Arguments

result	a result object containing the summary statistics and hypothesis tests for the case.

Value

a _function_ that creates the relevant ggplot object

Examples

```
## Not run: map_to_visualisation("result_var1")
## Not run: result_var1<- map_to_result( ... )
my_vis_fun <- map_to_visualisation(result_var1)
my_ggplot_obj<-my_vis_fun( ... )
my_ggplot_obj # plots the object
## End(Not run)
```

`map_to_visualisation_heatmap` *heatmaps are currently not part of the visualisations produced by map_to_visualisation they can't show confints but they can be really useful if lots of categories are shown (for example 8 responses in 8 groups would be 64 bars but make a neat 8x8 heatmap) Heatmaps from 'result' objects*

Description

heatmaps are currently not part of the visualisations produced by map_to_visualisation they can't show confints but they can be really useful if lots of categories are shown (for example 8 responses in 8 groups would be 64 bars but make a neat 8x8 heatmap) Heatmaps from 'result' objects

Usage

```
map_to_visualisation_heatmap(result)
```

Arguments

`result`	a hypegrammaR result object (can be made with map_to_result())

Details

to add labels, use 'myresult

Value

A hypegrammaR visualisation object, which is a list with two elements, 1) a ggplot object and 2) recommended parameters to pass to ggsave.

`map_to_weighting` *creates a weighting function from a sampling frame*

Description

creates a weighting function from a sampling frame

Usage

```
map_to_weighting(sampling.frame, data.stratum.column,
  sampling.frame.population.column = "population",
  sampling.frame.stratum.column = "stratum", data = NULL)
```

Arguments

`data.stratum.column`
: data column name that holds the record's strata names

`sampling.frame.population.column`
: sampling frame name of column holding population counts. defaults to "population"

`sampling.frame.stratum.column`
: sampling frame name of column holding stratum names. defaults to "stratum". Stratum names must match exactly values in:

`data`
: optional but recommended: you can provide an example data frame of data supposed to match the sampling frame to check if the provided variable names match and whether all strata in the data appear in the sampling frame.

`sampling.frame.file`
: data frame containing the sampling frame. should contain columns "stratum" and "population", otherwise column names must be specified.

Value

returns a new function that takes a data frame as input returns a vector of weights corresponding to each row in the data frame.

Examples

```
## Not run: # load data and sampling frames:
mydata<-read.csv("mydata.csv")
mysamplingframe<-read.csv("mysamplingframe.csv")
# create weighting function:
weighting<-weighting_fun_from_samplingframe(sampling.frame = mysamplingframe,
                                data.stratum.column = "strata_names",
                                sampling.frame.population.column = "pop",
                                sampling.frame.stratum.column = "strat_name")
# use weighting function:
mydata$weights<-weighting(mydata)

# this also works on subsets of the data:
mydata_subset<-mydata[1:100,]
subset_weights<- weighting(mydata)
## End(Not run)
```

---

mean_with_confints *Weighted means with confidence intervals*

Description

Weighted means with confidence intervals

Usage

```
mean_with_confints(dependent.var, independent.var = NULL, design,
  confidence_level = 0.95)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a numerical variable.
independent.var	should be null ! For other functions: string with the column name in ‘data‘ of the independent variable
design	the svy design object created using map_to_design or directly with svydesign
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Details

This function takes the design object and the name of your dependent variable when the latter is a numerical. It calculates the weighted mean for your variable.

Value

A table in long format of the results, with the column names dependent.var, dependent.var.value (=NA), independent.var (= NA), independent.var.value (= NA), numbers (= mean), se, min and max.

---

mean_with_confints_groups

Weighted means with confidence intervals for groups

Description

Weighted means with confidence intervals for groups

Usage

```
mean_with_confints_groups(dependent.var, independent.var, design,
  confidence_level = 0.95)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a numerical variable.
independent.var	string with the column name in ‘data‘ of the independent (group) variable. Should be a ’select one’
design	the svy design object created using map_to_design or directly with svydesign
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Details

This function takes the design object and the name of your dependent variable when the latter is a numerical. It calculates the weighted mean for your variable.

Value

A table in long format of the results, with the column names dependent.var, dependent.var.value (=NA), independent.var, independent.var.value, numbers (= mean), se, min and max.

`percent_with_confints_select_multiple` *Weighted percentages with confidence intervals for select multiple questions*

Description

Weighted percentages with confidence intervals for select multiple questions

Usage

```
percent_with_confints_select_multiple(dependent.var, dependent.var.sm.cols,
  design, na.rm = TRUE, confidence_level = 0.95)
```

Arguments

`dependent.var`	string with the column name in ‘data‘ of the dependent variable. Should be a ’select multiple.
`dependent.var.sm.cols`	a vector with the columns indices of the choices for the select multiple question. Can be obtained by calling choices_for_select_multiple(question.name, data)
`design`	the svy design object created using map_to_design or directly with svydesign
`confidence_level`	the confidence level to be used for confidence intervals (default: 0.95)

Details

this function takes the design object and the name of your dependent variable when this one is a select multiple. It calculates the weighted percentage for each category.

Value

A table in long format of the results, with the column names dependent.var, dependent.var.value, independent.var (= NA), independent.var.value (= NA), numbers, se, min and max.

percent_with_confints_select_multiple_groups

Weighted percentages with confidence intervals for groups (select multiple questions)

Description

Weighted percentages with confidence intervals for groups (select multiple questions)

Usage

```
percent_with_confints_select_multiple_groups(dependent.var,
  dependent.var.sm.cols, independent.var, design, na.rm = TRUE,
  confidence_level = 0.95)
```

Arguments

dependent.var	string with the column name in 'data' of the dependent variable. Should be a 'select multiple.
dependent.var.sm.cols	a vector with the columns indices of the choices for the select multiple question. Can be obtained by calling choices_for_Select_multiple(question.name, data)
independent.var	string with the column name in 'data' of the independent (group) variable. Should be a 'select one'
design	the svy design object created using map_to_design or directly with svydesign
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Details

this function takes the design object and the name of your dependent variable when this one is a select multiple. It calculates the weighted percentage for each category.

Value

A table in long format of the results, with the column names dependent.var, dependent.var.value, independent.var (= NA), independent.var.value (= NA), numbers, se, min and max.

percent_with_confints_select_one

Weighted percentages with confidence intervals

Description

Weighted percentages with confidence intervals

Usage

```
percent_with_confints_select_one(dependent.var, independent.var = NULL,
  design, na.rm = TRUE, confidence_level = 0.95)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a ’select one’
independent.var	should be null ! For other functions: string with the column name in ‘data‘ of the independent variable
design	the svy design object created using map_to_design or directly with svydesign
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Details

this function takes the design object and the name of your dependent variable when this one is a select one. It calculates the weighted percentage for each category.

Value

A table in long format of the results, with the column names dependent.var, dependent.var.value, independent.var, independent.var.value, numbers, se, min and max.

Examples

```
## Not run: percent_with_confints_select_one("population_group", design)
```

---

percent_with_confints_select_one_groups

Weighted percentages with confidence intervals for groups

Description

Weighted percentages with confidence intervals for groups

Usage

```
percent_with_confints_select_one_groups(dependent.var, independent.var,
  design, na.rm = TRUE, confidence_level = 0.95)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a ’select one’
independent.var	string with the column name in ‘data‘ of the independent (group) variable. Should be a ’select one’
design	the svy design object created using map_to_design or directly with svydesign
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Details

this function takes the design object and the name of your dependent variable when this one is a select one. It calculates the weighted percentage for each category in each group of the independent variable.

Value

A table in long format of the results, with the column names dependent.var, dependent.var.value, independent.var, independent.var.value, numbers, se, min and max.

Examples

```
## Not run: percent_with_confints_select_one_groups("population_group", "resp_gender", design)
```

---

reach_style_barchart *not used*

Description

not used

Usage

```
reach_style_barchart(group, percent, error_min = NULL,
  error_max = NULL, horizontal = T)
```

---

reach_style_color_beige *reach brand beiges*

Description

reach brand beiges

Usage

```
reach_style_color_beige(lightness = 1)
```

---

reach_style_color_beiges *Reach brand beige triples*

Description

Reach brand beige triples

Usage

```
reach_style_color_beiges()
```

reach_style_color_darkgrey

Reach brand dark greys

Description

Reach brand dark greys

Usage

```
reach_style_color_darkgrey(lightness = 1)
```

reach_style_color_darkgreys

Reach brand dark grey triples

Description

Reach brand dark grey triples

Usage

```
reach_style_color_darkgreys()
```

reach_style_color_lightgrey

reach brand light greys

Description

reach brand light greys

Usage

```
reach_style_color_lightgrey(lightness = 1)
```

reach_style_color_lightgreys

Reach brand light greys triples

Description

Reach brand light greys triples

Usage

```
reach_style_color_lightgreys()
```

reach_style_color_red *Reach brand reds*

Description

Reach brand reds

Usage

```
reach_style_color_red(lightness = 1)
```

reach_style_color_reds *Reach brand reds triples*

Description

Reach brand reds triples

Usage

```
reach_style_color_reds()
```

read.csv.auto.sep *loading function with automatic default*

Description

loading function with automatic default

Usage

```
read.csv.auto.sep(file, stringsAsFactors = F, ...)
```

Arguments

file	path to a csv file with the assessment data

Details

the file is loaded with stringsAsFactors = F and with column names in alphanumeric lowercase. This is mainly relevant if hypegrammaR is used in the background of an interface where people don't load and their data into R manually.

Value

the data from the csv files as data frame. Column header symbols are changed to lowercase alphanumeric and underscore; everything else is converted to a "."

resultlist_recursive_markdown

Rmarkdown from resultlist in specified hierarchical order

Description

Rmarkdown from resultlist in specified hierarchical order

Usage

```
resultlist_recursive_markdown(resultlist,
  by_analysisplan_columns = c("dependent.var"), by_prefix = c("",
  "subset:", "variable:"), level = 2, render_result_with,
  questionnaire = NULL, label_varnames = TRUE)
```

Arguments

resultlist	structure like the output from from_analysisplan_map_to_output: A list with two items "analysisplan" and "results": The "analysisplan" as a data frame, where each row must match a result in a list of "results"
by_analysisplan_columns	vector of strings matching column names of the analysisplan. The first element becomes the main heading, the second element the sub-heading etc.
by_prefix	a prefix added at the beginnig of the headline; same length as 'by_analysisplan_columns'
level	the markdown header level to start with; defaults to 2 which leads to "## heading", i.e. the second header level.
render_result_with	a function that takes a single result as input and returns an rmarkdown formated string
questionnaire	optional; the questionnaire (koboquest::load_questionnaire())
label_varnames	wether variables names should be labeled in headings

results_rbind_summary_statistic

bind the summary statistics tables from a list of results into a single data.frame

Description

bind the summary statistics tables from a list of results into a single data.frame

Usage

```
results_rbind_summary_statistic(results)
```

Arguments

results	a list of results as produced by from_analysisplan_map_to_output, or a list of outputs from map_to_result()

Value

a single data.frame with the summary statistics from all results in the input list

---

results_subset *subset a list of results based on analysis parameters*

Description

subset a list of results based on analysis parameters

Usage

```
results_subset(results, repeat.vars = NULL, repeat.var.values = NULL,
  dependent.vars = NULL, logical = NULL)
```

Arguments

results	list of results (output from 'from_analysisplan_map_to_output()')
repeat.vars	optional: vector of character strings: keeps only results where repeat.var in this list
repeat.var.values	optional: vector of character strings: keeps only results where repeat.var.vaues in this list
dependent.vars	optional: vector of character strings: keeps only results where dependent.var in this list
logical	optional: subset by a logical vector (same length as list of results)

Details

if multiple parameters are given to subset by, only those are kept where all conditions apply

Value

a resultlist in same format as from_analysisplan_map_to_output() only including those results with matching analysis parameters

---

rmdrs_pretty_summary_table *style result summary statistics for markdown outputs*

Description

style result summary statistics for markdown outputs

style result summary statistics for markdown outputs

Usage

```
rmdrs_pretty_summary_table(summary.table)

rmdrs_pretty_summary_table(summary.table)
```

Arguments

summary.table	a summary.statistic data frame from a result object (see map_to_result)
summary.table	a summary.statistic data frame from a result object (see map_to_result)

Value

a data.frame simmilar to the summary.statistic data.frame with print-ready column names and order df: data.frame from: columns to rename to: new column names order: integer vector with new column order

a data.frame simmilar to the summary.statistic data.frame with print-ready column names and order df: data.frame from: columns to rename to: new column names order: integer vector with new column order

---

sanitise_data

these functions should be depreciated eventually, but for now still in use in hypothesis_tests.R they should go because now there is a proper system to apply data sanitation within each summary statistic and hypothesis test function. take the data and some information about the planned analysis and prepare the data to ensure analysis doesn't fail

---

Description

these functions should be depreciated eventually, but for now still in use in hypothesis_tests.R they should go because now there is a proper system to apply data sanitation within each summary statistic and hypothesis test function. take the data and some information about the planned analysis and prepare the data to ensure analysis doesn't fail

Usage

```
sanitise_data(data, dependent.var, independent.var, case)
```

Arguments

data	data
dependent.var	the name of the dependent variable
independent.var	the name of the independent variable
case	the analysis case (see map_to_case())

Value

a list with two items: - 1. 'data': the sanitised dataset - 2. 'success' a logical value, TRUE if the data can be analysed,FALSE if the sanitation was unsuccessful

if '$success' is FALSE, there may be a '$message' with more information, and the data will be NULL.

summary_statistic_mode_select_one
Weighted means with confidence intervals for groups

Description

Weighted means with confidence intervals for groups

Usage

```
summary_statistic_mode_select_one(dependent.var, independent.var, design,
  confidence_level = 0.95)
```

Arguments

dependent.var	string with the column name in ‘data‘ of the dependent variable. Should be a select_one or a select_multiple.
independent.var	string with the column name in ‘data‘ of the independent (group) variable. Should be a ’select one’
design	the svy design object created using map_to_design or directly with svydesign
confidence_level	the confidence level to be used for confidence intervals (default: 0.95)

Details

This function takes the design object and the name of your dependent variable, and returns the most frequent answer for each category in independent.var

Value

A table in long format of the results, with the column names dependent.var, dependent.var.value (=NA), independent.var, independent.var.value, numbers (= mean), se, min and max.

---

to_alphanumeric_lowercase
convert text to alphanumeric lowercase. It’ similar to make.names() but more tailored to kobo. NOTE: there’s a general dilemma on making sure that variable names always match between data headers and questionnaire etc. we introduced this to try to be able to standardise all column headers as much as possible HOWEVER in hindsight, this is causing lots of issues; even though it prevents a lot of mistakes, it makes things less predictable. Ideally we should revert this; basically remove this function _everywhere_, and expect people to load the data with "check.names = F" and have precise names.

Description

convert text to alphanumeric lowercase. It' similar to make.names() but more tailored to kobo. NOTE: there's a general dilemma on making sure that variable names always match between data headers and questionnaire etc. we introduced this to try to be able to standardise all column headers as much as possible HOWEVER in hindsight, this is causing lots of issues; even though it prevents a lot of mistakes, it makes things less predictable. Ideally we should revert this; basically remove this function _everywhere_, and expect people to load the data with "check.names = F" and have precise names.

Usage

```
to_alphanumeric_lowercase(x)
```

Index