Package 'MetaPipeXUpdate'

September 8, 2023

Type Package

Title Standardizing Data Structure, Analysis Code & Reporting for Experimental Data of Between Groups Comparisons in Replication Projects

Version 0.0.0.9000

Author Jens Fuenderich

Maintainer Jens Fuenderich <Jens.Fuenderich@uni-erfurt.de>

Description The 'MetaPipeX' framework is a reaction to the diversity in reporting standards and data formats in multi-lab replication projects. It serves as a proposal to standardize the data structure, analysis code and reporting for experimental data of between groups comparisons in replication projects. 'MetaPipeX' consists of three components. A descriptive pipeline for data transformations and analyses, analysis functions that implement the pipeline and a Shiny App that utilizes the standardized structure to allow various insights into the data produced by the pipeline. While the framework maps most easily to direct replications, its scope may be broadened to conceptual replications and similar projects. Illustrations of the pipeline and the functions are available on 'github' in the Graphics folder (<https://github.com/JensFuenderich/MetaPipeX/tree/main/Supplementary_Material>) The 'MetaPipeX' R-package provides a readily available set of functions to run the standardized part of the pipeline. The package includes three analysis functions that each represent a step in the 'MetaPipeX' pipeline ('create_replication_summaries', 'merge_replication_summaries' and 'meta_analyses') and a fourth that runs the pipeline, starting at person level data ('full_pipeline'). Further, it includes a function that runs the Shiny App ('MetaPipeX::ShinyApp'). All meta-analyses use 'metafor::rma.mv' ('Viechtbauer', 2010).

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports dplyr,
DT,
ggplot2,
grDevices,
haven,
janitor,
magrittr,
mathjaxr,

metafor,
puniform,
readr,
shiny,
shinyjs,
shinythemes,
shinyWidgets,
stats,
utils

RdMacros mathjaxr

R topics documented:

`MetaPipeXUpdate-package`

The MetaPipeXUpdate Package

Description

MetaPipeXUpdate

Details

Package:	MetaPipeXUpdate
Type:	Package
Version:	0.0.1
Date:	2023-04-21
Depends:	R (>= 4.2)
License:	GPL (>=3)
URL:	https://github.com/JensFuenderich/MetaPipeX_Update

A package for analyzing and exploring multi-lab replication data in a standardized format.

Author(s)

Jens Fuenderich

`create_MetaPipeX_format` *create MetaPipeX format*

Description

create MetaPipeX format

Usage

```
create_MetaPipeX_format(
  Merged_Site_Summaries,
  Meta_Analyses,
  output_folder = NULL,
  suppress_list_output = FALSE
)
```

Arguments

`Merged_Site_Summaries`	A data frame or path to a .csv format data frame that contains the merged site summaries. Use the MetaPipeXUpdate::merge_site_summaries function to create this data format, or use the hrefhttps://github.com/JensFuenderich/MetaPipeX/blob/main/Supplem on github.
`Meta_Analyses`	A data frame or path to a .csv format data frame that contains the merged site summaries. Use the MetaPipeXUpdate::meta_analyze_MASCs function to create this data format, or use the hrefhttps://github.com/JensFuenderich/MetaPipeX/tree/main/Supplemo on github.
`output_folder`	Specify the output folder for the meta-analyses and the codebook. If no folder is specified, the function will return its output only to the R environment (unless this is suppressed under suppress_list_output).
`suppress_list_output`	A logical indicating whether results should be returned in R. If TRUE, no output is returned in R.

Details

No transformations are performed on the data in this step of the MetaPipeX pipeline.

Value

A list object containing the following components:
MetaPipeX data format A data frame with all site summary and meta-analytical statistics from all included MASCs (Meta-Analytical-Data-Collections).

codebook A codebook that applies to the data frame (merged_site_summaries).
In order to export the data structure as .csv files in a folder, output_folder has to be specified.

Examples

```
# create IPD for 10 MASCs (all from the same MultiLab)
sim_out <- mapply(MetaPipeXUpdate::simulate_IPD,
                  MASC_index = 1:5,
                  seed = 50 + (0:(5-1)),
                  SIMPLIFY = FALSE)
# rename list elements (the individual MASCs)
names(sim_out) <- paste("MASC", 1:5, sep = "")

# run the full pipeline to create the inputs for the arguments:
# Merged_Site_Summaries & Meta_Analyses
MetaPipeX_out <- MetaPipeXUpdate::full_pipeline(data = sim_out)

# run the function using the data frames created from the pipeline
MetaPipeXUpdate::create_MetaPipeX_format(
 Merged_Site_Summaries = MetaPipeX_out$lvl3_merged_site_summaries$Merged_Site_Summaries,
  Meta_Analyses = MetaPipeX_out$lvl4_meta_analyses$Meta_Analyses)
```

`full_pipeline` *Full Pipeline Function*

Description

\(\let\underscore_ \)

This function is built on four MetaPipeX functions (summarize_sites, merge_site_summaries, meta_analyze_MASCs, create_MetaPipeX_format). As input it expects the same specifications as the summarize_sites function. This function performs all standardized computational steps (3-6) of the MetaPipeX pipeline. For more details on the pipeline, refer to the documentation of the MetaPipeX-package.

Usage

```
full_pipeline(
  data,
  MultiLab = NULL,
  MASC = NULL,
  Data_Collection_Site = NULL,
  DV = NULL,
  Group = NULL,
  output_path = NULL,
  folder_name = NULL,
  suppress_list_output = FALSE,
  method = "REML",
  sparse = FALSE
)
```

Arguments

`data`	A data frame or list of data frames that contain the individual participant data. The function expects the relevant columns to be named consistently across all list objects. Relevant to this function are columns that represent information on the multi-lab (e.g., Many Labs 2), the MASC (meta-analytical-site-collection;

e.g., Ross1), the data collection site (the lab a data point is assigned to), the group (either the treatment or control condition) and the single data point of the dependent variable (DV) per person. A template of this data frame is available on github, as is a codebook for unambiguous identification of the abbreviations.

`MultiLab` Character vector with the name of the columns in the list elements of "data" that contain the project name(s). If *is.null(Project)* == *TRUE*, "Project" is chosen as the default.

`MASC` Character vector with the name of the columns in the list elements of "data" that contain the meta-analytical-study-collections (MASCs) name(s). If *is.null(Data_Collection_Site)* == *TRUE*, "MASC" is chosen as the default. Each MASC comprises a single target effect with implementations across multiple sites (/labs).

`Data_Collection_Site` Character vector with the name of the columns in the list elements of "data" that contain the site names (usually the name of the lab). If *is.null(Data_Collection_Site)* == *TRUE*, "Data_Collection_Site" is chosen as the default. The meta-analyses in MetaPipeXUpdate::meta_analyses() and MetaPipeXUpdate::full_pipeline() are run as random effects models in metafor::rma.mv() with "random = ~ 1 | Data_Collection_Site". Thus, the pipeline assumes a distribution of true statistics (e.g., treatment means, mean differences, standardized mean differences).

`DV` Character vector with the name of the columns in the list elements of "data" that contain the (aggregated) dependent variable. If *is.null(DV)* == *TRUE*, "DV" is chosen as the default.

`Group` Character vector with the name of the columns in the list elements of "data" that contain the (treatment/control) group identification. If *is.null(Group)* == *TRUE*, "Group" is chosen as the default. These should only contain values of 0 (control group), 1 (treatment group) and NA (unidentified).

`output_path` Specify the output path for the full documentation of the MetaPipeX pipeline. For an example of the exported structure please refer to the github repository. If no folder is specified, the function will return its output only to the R environment (unless this is suppressed under suppress_list_output).

`folder_name` Optional character string to assign a custom name to the output folder. When folder_name is not specified, the folder name is set to "MetaPipeX_Output".

`suppress_list_output` Logical. FALSE by default. If FALSE, the function will return a list output to the environment, containing the site summaries and the codebook. If TRUE, these are not returned to the environment.

`method` Optional argument to specify the estimation method of the meta-analyses (the default is "REML"). For more information, please refer to the documentation of the metafor package.

`sparse` A logical indicating whether sparse matrices should be used.

Details

General notes on the pipeline

The MetaPipeX pipeline is a tool to provide structure to the meta-analytical-analyses of multi-lab MASCs. A flowchart that depicts the whole process is available on github The yellow blocks with rounded corners are .csv files. The purple/white rectangles each refer to a step in the pipeline that is performed by a MetaPipeX function. MetaPipeXUpdate::full_pipeline() performs the steps 3-6 and returns "meta_pipe_x_data.csv" which may be provided to the MetaPipeX App for handy data

selection and basic plotting of the analysis results. Pleasre refer to github for an example of the MetaPipeX Output structure.

full_pipeline

This function executes the pipeline as follows:

- MetaPipeXUpdate::summarize_sites()
- MetaPipeXUpdate::merge_site_summaries()
- MetaPipeXUpdate::meta_analyze_MASCs()
- MetaPipeXUpdate::create_MetaPipeX_format()

Value

The output is a nested list object that represents the folder structure that is available on GitHub. For an inspection of the list output, run the example.

Examples

```
# create IPD for 10 MASCs (all from the same MultiLab)
sim_out <- mapply(MetaPipeXUpdate::simulate_IPD,
                  MASC_index = 1:5,
                  seed = 50 + (0:(5-1)),
                  SIMPLIFY = FALSE)
# rename list elements (the individual MASCs)
names(sim_out) <- paste("MASC", 1:5, sep = "")

# run the full pipeline
MetaPipeXUpdate::full_pipeline(data = sim_out)


## Not run:
All examples with additional comments are available on github:
https://github.com/JensFuenderich/MetaPipeX/tree/main/Supplementary_Material/Code_Examples

## End(Not run)
```

---

merge_site_summaries *Merging (Data Collection) Site Summaries*

Description

\(\let\underscore_ \) Function to merge the data collection site aggregates returned by MetaPipeX-Update::summarize_sites() into a single data frame. This is the second standardized function (and the fourth computational step) of the MetaPipeX pipeline. For more details on the pipeline, refer to the documentation of the MetaPipeX-package.

Usage

```
merge_site_summaries(data, output_folder = NULL, suppress_list_output = FALSE)
```

Arguments

data	The function expects the input to be a list of data frames or a path to a folder containing the data collection site summaries as .csv files. The input may either be produced by the MetaPipeXUpdate::summarizes_sites() function, or any inputs that use the data template. A template of this data frame is available on github, as is a codebook for unambiguous identification of the abbreviations.
output_folder	Define a path to which the merged (data collection) site summaries and the codebook are exported. If no path is specified, results are returned only in R.
suppress_list_output	A logical indicating whether results should be returned in R. If TRUE, no output is returned in R.

Details

No transformations are performed on the data in this step of the MetaPipeX pipeline.

Value

A list object containing the following components:
merged_site_summaries A data frame with all site summaries from the input.

codebook A codebook that applies to the data frame (merged_site_summaries).
In order to export the data structure as .csv files in a folder, output_folder has to be specified.

Examples

```
# create IPD for 10 MASCs (all from the same MultiLab)
sim_out <- mapply(MetaPipeXUpdate::simulate_IPD,
                  MASC_index = 1:5,
                  seed = 50 + (0:(5-1)),
                  SIMPLIFY = FALSE)
# rename list elements (the individual MASCs)
names(sim_out) <- paste("MASC", 1:5, sep = "")

# create site summaries
Site_Summaries <- MetaPipeXUpdate::summarize_sites(data = sim_out)

# run MetaPipeX function to merge site summaries
MetaPipeXUpdate::merge_site_summaries(data = Site_Summaries$Site_Summaries)

## Not run:
All examples with additional comments are available on github:
https://github.com/JensFuenderich/MetaPipeX/tree/main/Supplementary_Material/Code_Examples

## End(Not run)
```

`meta_analyze_MASCs` *Meta Analyses per MASC (meta-analytical study collection)*

Description

\(\let\underscore_ \) Function to run meta-analyses on the mean difference (MD) and the standardized mean difference (SMD). The meta-analyses are run with the metafor::rma.mv function (Viechtbauer, 2010). For more details on the meta-analyses, refer to the Details and Return section. This function is the third (and fifth computational step) of the MetaPipeX pipeline. For more details on the pipeline, refer to the documentation of the MetaPipeX-package.

Usage

```
meta_analyze_MASCs(
  data,
  output_folder = NULL,
  suppress_list_output = FALSE,
  method = "REML",
  sparse = FALSE
)
```

Arguments

`data`	The function expects the input to be a data frame. The input may either be the data frame produced by the MetaPipeXUpdate::merge_site_summaries() function, or one with the same columns names. A template of this data frame is available on github, as is a codebook for unambiguous identification of the abbreviations. Further, it is possible to use a reduced version of the codebook, as meta-analyses are applied to MD and SMD only.
`output_folder`	Specify the output folder for the meta-analyses and the codebook. If no folder is specified, the function will return its output only to the R environment (unless this is suppressed under suppress_list_output).
`suppress_list_output`	A logical indicating whether results should be returned in R. If TRUE, no output is returned in R.
`method`	A character string to specify the type of model to be fitted. Default is "REML". For more details, refer to the metafor documentation.
`sparse`	A logical indicating whether sparse matrices should be used.

Details

The meta-analyses within the function are written with metafor::rma.mv (Viechtbauer, 2010). The multivariate version of the rma function is deployed to allow for the use of sparse matrices ("sparse = TRUE") for optimal performance in meta-analyses with thousands of data collection sites. They are fitted as a random-effects model with "random = ~ 1 | Data_Collection_Site" and a restricted maximum likelihood estimation ("REML"). The function runs seven meta-analyses per MASC:

- control mean (yi = C_M, V = SE_C_M)
- treatment mean (yi = T_M, V = SE_T_M)
- control group standard deviation(yi = ln_SD, V = SE_ln_SD)

- treatment group standard deviation (yi = ln_SD, V = SE_ln_SD)
- mean difference (yi = MD, V = SE_MD)
- pooled standard deviation (yi = ln_SD, V = SE_ln_SD)
- standardized mean difference (yi = SMD, V = SE_SMD)

Meta-analyses of standard deviations are performed on log-transformed standard deviations, as recommended by Nakagawa et al. (2015). All standard deviations (C_SD, T_SD, pooled_SD) are transformed using:

- R-Code
  `ln_SD = log(SD) + 1 / (2 * (n - 1))`

- Model

$$ln\hat{\tau} = lnSD + \frac{1}{2(n-1)}$$

All standard errors of standard deviations are created using:

- R-Code
  `SE_ln_SD = sqrt(1 / (2 * (n - 1)))`

- Model

$$s^2_{ln\hat{\tau}} = \sqrt{\frac{1}{2(n-1)}}$$

In order to achieve interpretable meta-analytical results (the model estimate, tau, tau^2 and the coefficient of variation), they are transformed back into the original units. The following transformation is applied to the model estimate:

- R-Code
  `Model_Est = exp(mu_ln + 0.5 * tau2_ln)`

- Model

$$\mu = e^{\mu_{ln} + 0.5\tau^2_{ln}}$$

The following transformation is applied to tau^2 (which is subsequently used to calculate tau and the coefficient of variation):

- R-Code
  `Tau2 = exp(2 * mu_ln + tau2_ln) * (exp(tau2_ln) - 1)`

- Model

$$\tau^2 = e^{2\mu_{ln} + \tau^2_{ln}}(e^{\tau^2_{ln}} - 1)$$

Value

The output is a list with two objects: A data frame with the meta-analytical results and a codebook for unambiguous identification of its columns.

meta analyses
The data frame contains information to identify each analysis (MultiLab, MASC) and statistical output from the two meta-analyses per MASC. The statistical output for each meta-analysis includes:

- A model estimate for the y of interest (Est__).
- The number of sites included in the analysis (Result__K).
- The estimated τ^2 (sigma2 from the rma.mv object) value (Tau2__).
- The estimated τ (the square root of the sigma2 from the rma.mv object) value (Tau2__).
- The estimated I^2 value. I^2 is not part of the rma.mv output object and has to be calculated from τ.

  $$I^2 = 100\frac{\hat{\tau}^2}{\hat{\tau}^2 + \tilde{v}}$$

  with

  $$\tilde{v} = \frac{(k-1)\sum w_i}{(\sum w_i) - \sum w_i^2}$$

  Transformation according to: https://wviechtb.github.io/metafor/reference/print.rma.html
- The estimated H^2 value. H^2 is not part of the rma.mv output object and has to be calculated from τ.

  $$H^2 = 100\frac{\hat{\tau}^2 + \tilde{v}}{\tilde{v}}$$

  with

  $$\tilde{v} = \frac{(k-1)\sum w_i}{(\sum w_i) - \sum w_i^2}$$

- The Q statistic (QE__).
- The p-value from the test on the Q statistic (QEp__).

codebook
A codebook that applies to the data frame (meta_analyses).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. Journal of Statistical Software, 36(3), 1-48. doi: 10.18637/jss.v036.i03 Nakagawa, S., Poulin, R., Mengersen, K., Reinhold, K., Engqvist, L., Lagisz, M., & Senior, A. M. (2015). Meta-analysis of variation: ecological and evolutionary applications and beyond. Methods in Ecology and Evolution, 6(2), 143-152. doi: 10.1111/2041-210X.12309

Examples

```
# create IPD for 10 MASCs (all from the same MultiLab)
sim_out <- mapply(MetaPipeXUpdate::simulate_IPD,
                  MASC_index = 1:5,
                  seed = 50 + (0:(5-1)),
                  SIMPLIFY = FALSE)
# rename list elements (the individual MASCs)
names(sim_out) <- paste("MASC", 1:5, sep = "")
```

```
# create site summaries
Site_Summaries <- MetaPipeXUpdate::summarize_sites(data = sim_out)

# merge site summaries
Merged_Site_Summaries <- MetaPipeXUpdate::merge_site_summaries(data = Site_Summaries$Site_Summaries)

# run the MetaPipeX function to meta-analyze all MASCs
MetaPipeXUpdate::merge_site_summaries(data = Merged_Site_Summaries$Merged_Site_Summaries)

## Not run:
All examples with additional comments are available on github:
https://github.com/JensFuenderich/MetaPipeX/tree/main/Supplementary_Material/Code_Examples

## End(Not run)
```

`ShinyApp` *MetaPipeX Shiny App*

Description

The MetaPipeX app is a GUI to provide insight into data that is in the MetaPipeX format. Make sure you are connected to the internet before running the app. For more details, please refer to the MetaPipeX tutorial. A web version of the app is available on a server of the LMU Munich.

Usage

```
ShinyApp()
```

Value

If executed this function will start a local instance of the MetaPipeX app.

`simulate_IPD` *simulate_IPD*

Description

simulate_IPD

Usage

```
simulate_IPD(MASC_index, MultiLab_index = NULL, seed = NULL)
```

Arguments

`MASC_index`	Insert (numeric or character) vector with indices.
`MultiLab_index`	Insert (numeric or character) vector with indices.
`seed`	Insert a vector with seeds, in order to create reproducible data.

Value

This function creates a list object with IPD for one MASC per list element.

Examples

```
# create IPD for 5 MASCs (all from the same MultiLab)
sim_out <- lapply(1:5, eval(MetaPipeXUpdate::simulate_IPD))
# rename list elements (the individual MASCs)
names(sim_out) <- paste("MASC", 1:5, sep = "")
```

`summarize_sites` *Creating (Data Collection) Site Summaries*

Description

\(\let\underscore_ \)

Function to compute data collection site aggregates from individual participant data. Components of the standardized mean difference and their standard errors are calculated and reported. This is the first standardized function (and the third computational step) of the MetaPipeX pipeline. For more details on the calculated statistics, refer to the Details section. For more details on the pipeline, refer to the documentation of the MetaPipeX-package.

Usage

```
summarize_sites(
  data,
  MultiLab = NULL,
  MASC = NULL,
  Data_Collection_Site = NULL,
  DV = NULL,
  Group = NULL,
  output_folder = NULL,
  suppress_list_output = FALSE
)
```

Arguments

`data`	A data frame or list of data frames that contain the individual participant data. The function expects the relevant columns to be named consistently across all list objects. Relevant to this function are columns that represent information on the multi-lab (e.g., Many Labs 2), the MASC (meta-analytical-site-collection; e.g., Ross1), the data collection site (the lab a data point is assigned to), the group (either the treatment or control condition) and the single data point of the dependent variable (DV) per person. A template of this data frame is available on github, as is a codebook for unambiguous identification of the abbreviations.
`MultiLab`	Character vector with the name of the columns in the list elements of "data" that contain the multi-lab name(s). If *is.null(MultiLab) == TRUE*, "MultiLab" is chosen as the default.

`MASC` Character vector with the name of the columns in the list elements of "data" that contain the MASCs name(s). If *is.null(MASC) == TRUE*, "MASC" is chosen as the default. Each MASC comprises a target-effect with implementations across multiple data collection sites (/labs).

`Data_Collection_Site` Character vector with the name of the columns in the list elements of "data" that contain the data collection site names. If *is.null(Data_Collection_Site) == TRUE*, "Data_Collection_Site" is chosen as the default. The meta-analyses in MetaPipeXUpdate::meta_analyses() and MetaPipeXUpdate::full_pipeline() are run as random effects models in metafor::rma.mv() with "random = ~ 1 | Data_Collection_Site". Thus, the pipeline assumes a distribution of true statistics (e.g., treatment means, mean differences, standardized mean differences).

`DV` Character vector with the name of the columns in the list elements of "data" that contain the (aggregated) dependent variable. If *is.null(DV) == TRUE*, "DV" is chosen as the default.

`Group` Character vector with the name of the columns in the list elements of "data" that contain the (treatment/control) group identification. If *is.null(Group) == TRUE*, "Group" is chosen as the default. These should only contain values of 0 (control group), 1 (treatment group) and NA (unidentified).

`output_folder` Specify the output folder for the summaries and the codebook. If no folder is specified, the function will return its output only to the R environment (unless this is suppressed under suppress_list_output).

`suppress_list_output` Logical. FALSE by default. If FALSE, the function will return a list output to the environment, containing the data collection site summaries and the codebook. If TRUE, these are not returned to the environment.

Details

Data_Collection_Site Statistics

All components of the standardized mean difference and their standard errors are returned by the function. Each standard error is returned to enable a meta-analysis on each component. The components and their standard errors are implemented as follows. Unless other sources are provided, effect size statistics are calculated according to Borenstein et al., 2009. The metafor::escalc function was used for SMD and MD (Viechtbauer, 2010).
mean (M)

- R-Code
```
## apply the function
# treatment group mean (T_M):
mean(treatment_group$DV)
# control group mean (C_M):
mean(control_group$DV)
```

- Model

  treatment group mean (T_M):

$$\bar{x}_T = \frac{1}{n} \sum_{i \in T} x$$

  control group mean (C_M):

$$\bar{x}_C = \frac{1}{n} \sum_{i \in C} x$$

standard error of the mean (SE_T_M, SE_C_M)

- R-Code

```
## define the function
SE_of_mean_fct <- function(x){
estimated_sd <- sqrt(sum((x-mean(x))^2)/(length(x)-1))
SE_of_mean <- sd(x) / sqrt(length(x))
return(SE_of_mean)}

## apply the function
# standard error of treatment group mean (SE_T_M):
SE_of_mean_fct(treatment_group$DV)
# standard error of control group mean (SE_C_M):
SE_of_mean_fct(control_group$DV)
```

- Model

$$\hat{\sigma}_{\bar{x}} = \frac{\hat{\sigma}_x}{\sqrt{n}} = \sqrt{\frac{\frac{1}{n-1}\sum_{i=1}^{n}(x-\bar{x})^2}{n}}$$

standard deviation (T_SD, C_SD)

- R-Code

```
## apply the function
# treatment group standard deviation (T_SD):
sd(treatment_group$DV)
# control group standard deviation (C_SD):
sd(control_group$DV)
```

- Model

$$\hat{\sigma} = \sqrt{\frac{\sum(x-\bar{x}^2)}{n-1}}$$

standard error of the standard deviation (SE_T_SD, SE_C_SD)

- R-Code

```
## define the function
SE_SD_fct <- function(x){
SE_SD <- sd(x) / sqrt(2*(length(x)-1)) # for large n
return(SE_SD) }

## apply the function
# standard error of the treatment group standard deviation (SE_T_SD):
SE_SD_fct(treatment_group$DV)
# standard error of the control group standard deviation (SE_C_SD):
SE_SD_fct(control_group$DV)
```

- Model

$$\hat{\sigma}_{\hat{\sigma}} = \frac{\hat{\sigma}_x}{\sqrt{2(n-1)}} = \sqrt{\frac{\frac{1}{n-1}\sum_{i=1}^{n}(x-\bar{x})^2}{2(n-1)}}$$

$\hat{\sigma}_{\hat{\sigma}}$ is a simplified version of $\sigma_{K_n S}$ in Ahn & Fessler (2003). The authors demonstrate that for n > 10 it is reasonable to use Kn = 1. As for the overwhelming majority of samples n > k may be assumed, we excluded the term K_n.

mean difference (MD)

- R-Code

```
## apply the function
metafor::escalc(
measure = "MD",
m1i = mean(treatment_group$DV),
m2i = mean(control_group$DV),
sd1i = sd(treatment_group$DV),
sd2i = sd(control_group$DV),
n1i = length(treatment_group$DV),
n2i = length(control_group$DV),
vtype = "HO" # assuming homoscedasticity
)$yi
```

- Model

$$D = \bar{x}_T - \bar{x}_C$$

standard error of mean difference (SE_MD)

- R-Code

```
## apply the function
metafor::escalc(
measure = "MD",
m1i = mean(treatment_group$DV),
m2i = mean(control_group$DV),
sd1i = sd(treatment_group$DV),
sd2i = sd(control_group$DV),
n1i = length(treatment_group$DV),
n2i = length(control_group$DV),
vtype = "HO" # assuming homoscedasticity
)$vi
```

- Model

$$\hat{\sigma}_{\bar{x}_T - \bar{x}_C} = \sqrt{\frac{n_T + n_C}{n_T n_C} \sigma^2_{TC}} = \sqrt{\frac{n_T + n_C}{n_T n_C} \frac{\sum_{i=1}^{n} (x_T - \bar{x}_T)^2 + \sum_{i=1}^{n} (x_C - \bar{x}_C)^2}{n_T + n_C - 2}}$$

pooled standard deviation (pooled_SD)

- R-Code

```
## define the function
pooled_SD_fct <- function(t,c){
pooled_SD <- sqrt((
(sum((t-mean(t))^2))+ # sample sum of squares sums treatment group
```

```
(sum((c-mean(c))^2)) # sample sum of squares control group)/
(length(t) + length(c) -2) # n+n-2
) # end of sqrt
return(pooled_SD)}
## apply the function
pooled_SD_fct(treatment_group$DV, control_group$DV)
```

- Model

$$\hat{\sigma}_{TC} = \sqrt{\frac{\sum_{i=1}^{n}(x_T - \bar{x}_T)^2 + \sum_{i=1}^{n}(x_C - \bar{x}_C)^2}{n_T + n_C - 2}}$$

standard error of pooled standard deviation (SE_pooled_SD)

- R-Code

```
## define the function
SE_pooled_SD_fct <- function(t,c){
pooled_SD <- sqrt((
(sum((t-mean(t))^2))+ # sample sum of squares sums treatment group
(sum((c-mean(c))^2)) # sample sum of squares control group)/
(length(t) + length(c) -2) # n+n-2
) # end of sqrt
SE_pooled_SD <- pooled_SD/sqrt(2*(length(t)+length(c)-1))
return(SE_pooled_SD)}
## apply the function
SE_pooled_SD_fct(treatment_group$DV, control_group$DV)
```

- Model

$$\hat{\sigma}_{\hat{\sigma}_{TC}} = \frac{\hat{\sigma}_{TC}}{\sqrt{2(n_T + n_C - 1)}}$$

The standard error is equivalent to that of the standard deviation. For further information, refer to the "standard error of the standard deviation" section.

standardized mean difference (SMD)

- R-Code

```
## apply the function
metafor::escalc(
measure = "SMD",
m1i = mean(treatment_group$DV),
m2i = mean(control_group$DV),
sd1i = sd(treatment_group$DV),
sd2i = sd(control_group$DV),
n1i = length(treatment_group$DV),
n2i = length(control_group$DV),
vtype = "LS2" # Borenstein variance
)$yi
## apply the function
```

- Model

$$g = d\left(1 - \frac{3}{4(n_T + n_C - 2) - 1}\right)$$

with

$$d = \frac{\bar{x}_T - \bar{x}_C}{\sqrt{\frac{\sum_{i=1}^{n}(x_T - \bar{x}_T)^2 + \sum_{i=1}^{n}(x_C - \bar{x}_C)^2}{n_T + n_C - 2}}}$$

standard error of standardized mean difference (SE_SMD)

- R-Code

```
## apply the function
sqrt(metafor::escalc(
measure = "SMD",
m1i = mean(treatment_group$DV),
m2i = mean(control_group$DV),
sd1i = sd(treatment_group$DV),
sd2i = sd(control_group$DV),
n1i = length(treatment_group$DV),
n2i = length(control_group$DV),
vtype = "LS2" # Borenstein variance
)$vi)
## apply the function
```

- Model

$$\hat{\sigma}_g = \sqrt{\hat{\sigma}_d^2 \left(1 - \frac{3}{4(n_T + n_C - 2) - 1}\right)^2}$$

with

$$\hat{\sigma}_d^2 = \frac{n_T + n_C}{n_T n_C} + \frac{d^2}{2(n_T + n_C)}$$

Value

The function summarize_collection_sites returns a list consisting of two elements: A codebook and a list of data frames. Each data frame contains all summary statistics for the according MASC (/effect). The summary statistics returned (including their standard error) are the means and standard deviations for control and experimental groups, pooled standard deviations, raw mean differences and standardized mean differences (Hedge's g according to Borenstein et al., 2009).

References

Ahn, S., & Fessler, J. A. (2003). Standard errors of mean, variance, and standard deviation estimators. EECS Department, The University of Michigan, 1(2).

Borenstein, M., Hedges, L. V., Higgins, J. P. T., & Rothstein, H. R. (2009). Introduction to Meta-Analysis John Wiley & Sons. Ltd, Chichester, UK. 10.1002/9780470743386

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. Journal of Statistical Software, 36(3), 1-48. doi: 10.18637/jss.v036.i03

Examples

```
# create IPD for 10 MASCs (all from the same MultiLab)
sim_out <- mapply(MetaPipeXUpdate::simulate_IPD,
                  MASC_index = 1:5,
                  seed = 50 + (0:(5-1)),
                  SIMPLIFY = FALSE)
# rename list elements (the individual MASCs)
names(sim_out) <- paste("MASC", 1:5, sep = "")

# run the MetaPipeX function to create site summaries
MetaPipeXUpdate::summarize_sites(data = sim_out)


## Not run:
All examples with additional comments are available on github:
https://github.com/JensFuenderich/MetaPipeX/tree/main/Supplementary_Material/Code_Examples

## End(Not run)
```

Index