Package 'multiverse'

May 9, 2026

Version 0.6.2

Date 2024-10-07

Title Create 'multiverse analysis' in R

Maintainer Abhraneel Sarma <abhraneel@u.northwestern.edu>

Description Implement 'multiverse' style analyses (Steegen S., Tuerlinckx F, Gelman A., Vanpaemal, W., 2016) <doi:10.1177/1745691616658637> to show the robustness of statistical inference. 'Multiverse analysis' is a philosophy of statistical reporting where paper authors report the outcomes of many different statistical analyses in order to show how fragile or robust their findings are. The 'multiverse' package (Sarma A., Kale A., Moon M., Taback N., Chevalier F., Hullman J., Kay M., 2021) <doi:10.31219/osf.io/yfbwm> allows users to concisely and flexibly implement 'multiverse-style' analysis, which involve declaring alternate ways of performing an analysis step, in R and R Notebooks.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0), knitr (>= 1.3.2)

Imports dplyr (>= 0.8.1), purrr (>= 0.3.4), rlang (>= 0.4.0), R6, methods, tidyr (>= 1.0.0), tibble, magrittr, tidyselect, formatR, collections, evaluate, rstudioapi, berryFunctions, furrr, styler, distributional, jsonlite, readr

Suggests ggplot2 (>= 3.0.0), testthat (>= 3.0.0), highr, rmarkdown, covr, broom, boot, gganimate, gifski, forcats, stringr, cowplot, tidybayes, png, stringi, modelr, future

VignetteBuilder knitr

Language en-US

BugReports https://github.com/MUCollective/multiverse/issues/

URL https://mucollective.github.io/multiverse/, https://github.com/mucollective/multiverse/

Config/testthat/edition 3

NeedsCompilation no

Author Abhraneel Sarma [aut, cre],
Matthew Kay [aut],
Michael Moon [ctb],
Mark Miller [ctb],
Kyle Hwang [ctb],
Hadley Wickham [ctb],
Alex Kale [ctb],
Nathan Taback [ctb],
Fanny Chevalier [ctb],
Jessica Hullman [ctb],
Pierre Dragicevic [ctb],
Yvonne Jansen [ctb]

Repository CRAN

Date/Publication 2024-10-07 23:20:03 UTC

Contents

`multiverse-package` *'Explorable Multiverse' Data Analysis and Reports in R*

Description

Implement 'multiverse' style analyses (Steegen S., Tuerlinckx F, Gelman A., Vanpaemal, W., 2016) doi:10.1177/1745691616658637 to show the robustness of statistical inference. 'Multiverse analysis' is a philosophy of statistical reporting where paper authors report the outcomes of many different statistical analyses in order to show how fragile or robust their findings are. The 'multiverse' package (Sarma A., Kale A., Moon M., Taback N., Chevalier F., Hullman J., Kay M., 2021) doi:10.31219/osf.io/yfbwm allows users to concisely and flexibly implement 'multiverse-style' analysis, which involve declaring alternate ways of performing an analysis step, in R and R Notebooks.

`multiverse` is an R package that aims to make it easy to declare 'multiverse-style' analysis in R and R notebooks. The 'multiverse' package allows users to concisely and flexibly declare alternate ways of performing an analysis step in order to show how fragile or robust their findings are.

Details

'Multiverse style' analyses (Steegen 2016) is intended to highlight the robustness of an analysis to arbitrary decisions that are present in any data analysis. Considering all possible combinations of reasonable decisions that can be made at each step of an analysis, 'multiverse style' analysis can surface whether an outcome is an artifact of a particular idiosyncratic combination of analysis choices, or if it is robust against such arbitrary choices.

However, current tools do not support declaring 'multiverse' analysis easily, requiring users to declare custom control flows and multiple nested 'if-else' blocks. The 'multiverse' package aims to simplify the process of composing 'multiverse' analysis using a flexible and concise syntax.

To get started with the multiverse package please refer to `vignette("branch")`', `vignette("conditions")` and `vignette("multiverse-in-rmd")`. For example implementations of analysis using the multiverse package, see the case studies `vignette("durante-multiverse-analysis")` and `vignette("hurricane")`.

Author(s)

Maintainer: Abhraneel Sarma <abhraneel@u.northwestern.edu>

Authors:

- Matthew Kay <mjskay@northwestern.edu>

Other contributors:

- Michael Moon <moon@utstat.toronto.edu> [contributor]
- Mark Miller <725mrm@gmail.com> [contributor]
- Kyle Hwang <kyle.sg.hwang@gmail.com> [contributor]
- Hadley Wickham <hadley@posit.co> [contributor]
- Alex Kale <kalea@uw.edu> [contributor]

- Nathan Taback <nathan.taback@utoronto.ca> [contributor]
- Fanny Chevalier <fanny@dgp.toronto.edu> [contributor]
- Jessica Hullman <jessica.hullman@gmail.com> [contributor]
- Pierre Dragicevic <pierre.dragicevic@inria.fr> [contributor]
- Yvonne Jansen <yvonne.jansen@sorbonne-universite.fr> [contributor]

References

Steegen, Sara, Francis Tuerlinckx, Andrew Gelman, and Wolf Vanpaemel. (2016). Increasing transparency through a multiverse analysis. _Perspectives on Psychological Science_, 11(5), 702-712. doi:10.1177/1745691616658637.

See Also

Useful links:

- https://mucollective.github.io/multiverse/
- https://github.com/mucollective/multiverse/
- Report bugs at https://github.com/MUCollective/multiverse/issues/new

---

`accessors` *Accessing contents of the multiverse object*

Description

A multiverse object contains several **Object variables**. These can be accessed using convenient functions. Variables from the analysis that is being performed within the multiverse can be accessed using the `$`. Object variables such as the `code`, the expanded `parameter options table`, the `parameters` and the `conditions` can be accessed using respective functions

Usage

```
## S3 method for class 'multiverse'
multiverse$name

## S3 replacement method for class 'multiverse'
multiverse$name <- value

expand(multiverse)

## Default S3 method:
expand(multiverse)

## S3 method for class 'multiverse'
expand(multiverse)
```

```
size(multiverse)

## Default S3 method:
size(multiverse)

## S3 method for class 'multiverse'
size(multiverse)

code(multiverse, .pretty = TRUE)

## Default S3 method:
code(multiverse, .pretty = TRUE)

## S3 method for class 'multiverse'
code(multiverse, .pretty = TRUE)

parameters(multiverse)

## Default S3 method:
parameters(multiverse)

## S3 method for class 'multiverse'
parameters(multiverse)

conditions(multiverse)

## Default S3 method:
conditions(multiverse)

## S3 method for class 'multiverse'
conditions(multiverse)

extract_variable_from_universe(multiverse, idx, name)
```

Arguments

multiverse	Object of class multiverse
name	a variable name
value	a new value to be assigned
.pretty	A binary argument whether ‘code()‘ should prettify the output using the tidyverse style guide. defaults to TRUE.
idx	index of the universe in the multiverse (corresponds to the row in the table)

branch *Define multiple analysis paths for a step in the multiverse*

Description

The branch function allows the user to define multiple analysis options for a particular step in the analysis.

Arguments

parameter	A string to identify the branch. Each branch is characterised using a parameter which takes different options.
...	Different options for completing a particular step in the analysis. Each option is declared as <option_name> ~ <option_calculation>. See examples for more details.
.options	Declare a continuous value as the option of a parameter using a sequence (see examples for details), and the expanded sequence will be included as options for that parameter in the multiverse.

Details

For every step in the analysis, there may be more than one analysis option. We use branch to declare these different analysis options. Each branch is characterised by a parameter. The first argument passed into the branch is the parameter.

All the other arguments passed into branch are the different analysis options corresponding to that parameter (that particular step in the analysis process). Naturally, at least two or more options should be declared. Thus, the branch function will provide a warning if the total number arguments passed is less than three.

Please refer to vignette("branch") for more details on how to use this function to create a complete multiverse analysis.

Examples

```
library(dplyr)

# Example 1: declaring multiple options for a data processing step
set.seed(123)
x = rnorm(100, 30, 10)

# Say that you have a variable, x. You want to discretise this variable into two ordinal
# categories – high (if x >= 30) and low (if x < 30). However, another researcher might argue
# for discretising this variable into three ordinal categories – high (if x >= 40),
# medium (if 20 <= x < 40), and low (if x < 20).

M.1 = multiverse() # create a new multiverse object

inside(M.1, {
```

```
y = branch(discretisation,
        "two_levels" ~ ifelse(x < 30, "low", "high"),
        "three_levels" ~ ifelse(x < 20, "low", ifelse(x > 40, "high", "medium"))
    )
})


# Example 2: using branch with tidyverse and `%>%`
# Let's say that we have some data which indicates the amount of time spent by a user
# in four different conditions which are indexed 1, 2, 3 and 4
# (the modality column in the following dataset).
# We will first load the data and convert the column into factor from integer.

data("userlogs")
data.userlogs.raw = userlogs %>%
    mutate( modality = factor(modality) ) %>%
    arrange( modality )

M.2 = multiverse() # create a new multiverse object

inside(M.2, {
    df = data.userlogs.raw %>%
        select(modality, duration) %>%
        mutate( duration = branch( data_transform,
                            "none" ~ duration,
                            "log" ~ log(duration)))
})


# Example 3: using branch with tidyverse and `%>%`
# Consider a scenario where there are more than one alternatives to
# identifying and removing outliers

data("hurricane")

M.3 = multiverse()

# here, we perform a `filter` operation in the multiverse
inside(M.3, {
    df.filtered = hurricane %>%
        filter(branch(death_outliers,
                    "no_exclusion" ~ TRUE,
                    "most_extreme" ~ name != "Katrina",
                    "two_most_extreme" ~ !(name %in% c("Katrina", "Audrey"))
    ))
})


# Example 4: using branch as a function
# An alternate way of implementing the `branch()` function from Example 2 may be:

M.4 = multiverse()
```

```
inside(M.4, {
    duration_transform = branch(data_trans,
        "log-transformed" ~ log,
        "un-transformed" ~ identity
    )

    duration = duration_transform(data.userlogs.raw$duration)
})



# Example 5: continuous option values for a parameter

M.5 = multiverse()
inside(M.5, {
  branch(foo, "option1" ~ 1, .options = 1:10)
})

M.6 = multiverse()
# alternatively, we could specify how we want the vector to be expanded
# for continuous parameters
inside(M.6, {
  branch(foo, "option1" ~ 1, .options = seq(0, 1, by = 0.1))
})
```

`durante` *Survey to study the effect of fertility on religiosity and political attitudes*

Description

A dataset containing the responses to the survey conducted by Durante, Rae, and Griskevicius (2013) in their study, *The fluctuating female vote: Politics, religion, and the ovulatory cycle*. Durante et al. study effect of fertility on religiosity and political attitudes. Steegen, Tuerlinckx, Gelman, and Vanpaemel (2016) used this dataset in their analysis to illustrate how a multiverse analysis can highlight the robustness of the conclusions reached by the original author.

Format

A data frame with 502 rows and 26 variables:

Abortion "Abortion is a women's [sic] right."

DateTesting Date of participant filling in the questionnaire.

Donate "For the next part of the study we will donate $1 to the presidential campaign of your preferred candidate. Please indicate which candidate's campaign you would like us to donate $1 to." Mitt Romney — Barack Obama"

FreeMarket "In nearly every instance, the free market allocates resources most efficiently."

Marijuana "Marijuana should be legal."

Marriage "Marriage is between a man and a woman."

PrivSocialSec "Privatize Social Security." 1 – 7

Profit "Business corporations make too much profit."

Rel1 "How much do you believe in God?"

Rel2 "I see myself as a religiously oriented person."

Rel3 "I believe that God or a Higher Power is responsible for my existence"

Relationship What is your current romantic relationship status?" (1) not dating/romantically involved with anyone, (2) dating or involved with only one partner, (3) engaged or living with my partner, (4) married, or (5) other. If participants picked response (5), they were prompted to provide a description of their relationship, which was subsequently coded into one of the four options by the original authors. The data here has already been coded into another response option.

ReportedCycleLength How many days long are your menstrual cycles? (for most women, the range is between 25-35 days) Keep in mind this is the number of days from the start of one menstrual period to the start of the next menstrual period and NOT the length of your menstrual bleeding.

RestrictAbortion "Laws should restrict abortion in all or most cases."

RichTax "The rich should pay a higher tax rate than the middle class."

StLiving "Government should ensure that all citizens meet a certain minimum standard of living"

StartDateNext Indicates the expected start date of their next menstrual period (the research material does not contain a question about the variable. However, the data file for Study 2 contained this variable.)

StartDateofLastPeriod Please give your best estimate of the date on which you started your last period (please be as precise as possible). This date was probably within the last few weeks. Sometimes thinking of where you were when you started your last period helps. For instance, was it on a weekend?, were you at work, was it during a football game?, etc. Please write the date in mm/dd/yyyy format (e.g., 8/18/2012).

StartDateofPeriodBeforeLast Please give your best estimate of the date on which you started the period before your last period (please be as precise as possible). Please write the date in mm/dd/yyyy format (e.g., 7/18/2012)."

StemCell "Stem cell research is moral and can be useful for science."

Sure1 "How sure are you about that date (StartDateofLastPeriod)?"

Sure2 How sure are you about that date (StartDateofPeriodBeforeLast)?

Vote "Imagine walking into the voting booth today. Who would you vote for in the presidential election?" Mitt Romney (republican) – Barack Obama (democrat)"

WorkerID ID of participant

Details

All questions were preceded by the prompt — "Please indicate how much you agree with the following statements"

The following items were responses to religiosity items (on a scale of 1 - 9): *Rel1*, *Rel2*, *Rel3*

The following items were responses to fiscal political attitudes items (on a scale of 1 - 7): *RichTax*, *TooMuchProfit*, *StandardLiving*, *FreeMarket*, *PrivSocialSec*

The following items were responses to social political attitudes items (on a scale of 1 - 7): *Abortion*, *Marriage*, *StemCell*, *Marijuana*, *RestrictAbortion*

In addition, the values of *StartDateofLastPeriod*, *StartDateofPeriodBeforeLast* and *StartDateNext* are missing for WorkerID 15 and 16. This impacts the calculation of *CycleDay* variable in the dataset. Steegen et al. "reconstructed this variable for their analysis after fixing some coding errors". To ensure that their results were identical to Durante et al.'s, they used the processed variable Cycle Day from the original data file (11 and 18 for WorkerIDs 15 and 16, respectively).

References

Kristina M Durante, Ashley Rae and Vladas Griskevicius. (2013). "The fluctuating female vote: Politics, religion, and the ovulatory cycle." *Psychological Science* 24(6), 1007-1016.

Sara Steegen and Francis Tuerlinckx and Andrew Gelman and Wolf Vanpaemel. (2015). "Increasing transparency through a multiverse analysis." *Perspectives on Psychological Science* 11(5), 702-712.

`execute` *Execute parts of, or the entire multiverse*

Description

These are functions which allow the user to execute parts or whole of the multiverse. The user can choose to either execute the default analysis using the `execute_universe()`, or the whole of the multiverse using the `execute_multiverse()`.

Usage

```
execute_multiverse(multiverse, parallel = FALSE, progress = FALSE)

execute_universe(multiverse, .universe = 1, parallel = FALSE, progress = FALSE)
```

Arguments

`multiverse`	The multiverse object
`parallel`	Indicates whether to execute the multiverse analysis in parallel. If TRUE, multiverse makes use of `future::future` as backend to support parallel processing. Requires configuration of `future::plan`. Defaults to FALSE.
`progress`	Indicates whether to include a progress bar for each step of the execution. Defaults to FALSE.

`.universe` Indicate which universe to execute, if the user wants to execute a specific combination of the parameters using `execute_universe()`. Defaults to NULL, which will execute the first (default) analysis.

Details

Each single analysis within the multiverse lives in a separate environment. We provide convenient functions to access the results for the default analysis, as well as parts or whole of the multiverse. Each analysis can also be accessed from the multiverse table, under the results column.

Examples

```
library(dplyr)

M <- multiverse()

inside(M, {
  data <- rnorm(100, 50, 20)

  x.mean <- mean(data, trim = branch(
    trim_values,
    "trim_none" ~ 0,
    "trim_1pc" ~ 0.05,
    "trim_5pc" ~ 0.025,
    "trim_10pc" ~ 0.05
  ))
})

# Computes the analysis for all
# universes in the multiverse`
M %>%
  execute_multiverse()
```

---

export_json *Exporting results from a multiverse analysis to JSON*

Description

Exports the results of the multiverse analysis to JSON in a format which is compatible with the multiverse visualisation tool

Usage

```
export_results_json(x, term, mean, sd, dist, filename)

export_results_dist_json(x, term, dist, filename)
```

```
export_code_json(x, filename)

export_data_json(x, filename)
```

Arguments

x	a tidy tibble or data frame which contains summary statistics or distributional information of each regression coefficient parameter
term	column in the data frame, x, which contains the name of the coefficients
mean	column in the data frame, x, which contains the mean estimate for each coefficient
sd	column in the data frame, x, which contains the standard error estimate for each coefficient
dist	column in the data frame, x, which contains vectorised distributions—an object of class 'distribution' for each coefficient
filename	filename on disk (as a character string)

Details

results JSON file schema It consists of a list of objects (where each object corresponds to one analysis in the multiverse). Within this object, the results attribute contains a(nother) list of objects corresponding to each outcome variable. For e.g., here we have four coefficients (see the results of the regression model), and thus the results attribute will contain four objects. Each object has the following attributes: - 'term': name of the outcome variable - 'estimate': mean / median point estimate i.e., $\mathbb{E}(\mu)$ for any parameter μ. - 'std.error': standard error for the point estimate i.e., $\sqrt{\text{var}(\mu)}$ - 'cdf.x': a list of quantiles - 'cdf.y': a list of cumulative probability density estimates corresponding to the quantiles

In addition, it also contains the following attributes, but these are not currently used by Milliways: - 'statistic' - 'p.value' - 'conf.low' - 'conf.high'

code JSON file schema It consists of two attributes: 'code' and 'parameters'. 'code' is a list of strings consisting of the R and multiverse syntax used to implement the analysis. For readability, we use [styler] to break up the declared code. 'parameters' is an object listing the parameter names and the corresponding options for each of the parameters declared in the analysis.

data JSON file schema It consists of a list of objects, each with two attributes: 'field' and 'values'. 'field' is the name of a column corresponding to a variable in the dataset. 'values' are a list of values for that variable in the dataset.

Value

a JSON file or (if a filepath is not specified) a dataframe for the results file and a list for the code file

Examples

```
library(dplyr)
library(tidyr)

M = multiverse()
```

```
inside(M, {
  df = tibble(
    x = rnorm(100),
    y = x * 0.5 + rnorm(100, 0, 2)
 )

 # alternatives to remove outlier
 df.filtered = df %>%
   filter(
     branch(outlier_exclusion,
        "2SD" ~ abs(y - mean(y)) > 2*sd(y),
        "3SD" ~ abs(y - mean(y)) > 3*sd(y)
     )
 )

  fit = lm(y ~ x, data = df)
  res = broom::tidy(fit)
})

execute_multiverse(M)

multiverse::expand(M) %>%
  extract_variables(res) %>%
  unnest(res) %>%
  export_results_json(term, estimate, std.error)
```

`extract_variables` *Extract variables and objects from the multiverse*

Description

This is a wrapper function for extracting one or more variables and objects declared from within the multiverse in a tidy format.

Usage

```
extract_variables(x, ..., .results = .results)

## S3 method for class 'multiverse'
extract_variables(x, ..., .results = .results)

## S3 method for class 'data.frame'
extract_variables(x, ..., .results = .results)
```

Arguments

x	either a multiverse object or a dataframe (created using `expand()`) from a multiverse object. See usage.
...	one or more variable (or object) names to be extracted from the multiverse object. The names can be quoted or unquoted.
.results	(Optional) if the .results column which stores the environments for each unique analysis has been changed, specify the new name of the column. Defaults to `.results`

Details

In a typical analysis, the user will declare variables and objects inside the multiverse object. However, it might be difficult to access the variables and objects, hence we provide convenient wrappers in the form of `extract_variables()`.

If the user wants to extract one or more literals (strings, integers, doubles or logicals of length 1) then each variables is separated out into its own column. If the user wants to extract one or more vectors (or lists) then each such variable will be extracted in its own list column. If the user wants to extract one or more dataframe then they a column of type data frame (or tibble) would be created (which is a special instance of a list column).

Examples

```
library(dplyr)

M <- multiverse()

inside(M, {
  data <- rnorm(100, 50, 20)

  x.mean <- mean(data, trim = branch(
    trim_values,
    "trim_none" ~ 0,
    "trim_1pc" ~ 0.05,
    "trim_5pc" ~ 0.025,
    "trim_10pc" ~ 0.05
  ))

  y <- sd(data)
})

# Extracts the relevant variable from the multiverse
M %>%
  extract_variables(x.mean)

# if you want to filter the multiverse before extracting variables from it
# you ca first create the table and manipulate it before extracting variables
expand(M) %>%
  extract_variables(x.mean)

# you can extract more than one variable from the multiverse simultaneously
```

```
# these variables will be new columns in the dataset
expand(M) %>%
  extract_variables(x.mean, y)
```

`get_code` *Code corresponding to a single analysis*

Description

Given a particular set of options for each parameter, extracts the code for performing a single analysis from the code used to declare the multiverse. This function is called automatically and not exported.

Usage

```
get_code(.code, .assgn = NULL)
```

Arguments

`.code`	Code that is passed to the multiverse. This is not stripped of calls such as `branch_assert()`.
`.assgn`	A list containing the assignments for each defined parameter in the multiverse

Details

For a particular parameter assignment (i.e. one set of options that each defined parameter in the multiverse takes), this function rewrites the code passed into the multiverse to output the corresponding code for that set of parameter values — the analysis for a single universe.

This is primarily going to be called by other functions, and perhaps not going to be as useful to the user for anything other than inspecting the rewritten code.

`hurricane` *Survey to study the effect of fertility on religiosity and political attitudes*

Description

A dataset for the study conducted by Jung et al. (2014) in their study, *Female hurricanes are deadlier than male hurricanes*.

Format

A data frame with 502 rows and 26 variables:

Year Year in which the hurricane occurred

Name Name given to the hurricane

MasFem A score of how Masculine or Feminine a hurricane is, on a scale of 1 - 11 where 1 is the most masculine and 11 is the most feminine. Each hurricane was rated by 9 independent coders

MinPressure_before The minimum pressure of hurricanes at the time of landfall in the United States obtained from NOAA (www.aoml.noaa.gov/hrd/hurdat/All_U.S._Hurricanes.html)

Minpressure_Updated_2014 The minimum pressure of hurricanes at the time of landfall in the United States obtained from NOAA (www.aoml.noaa.gov/hrd/hurdat/All_U.S._Hurricanes.html)

Gender_MF "Marriage is between a man and a woman."

Category Category labels on a scale of 1 - 5, with 5 being the most severe or extreme

alldeaths Total number of deaths caused by the hurricane. Information on death tolls of hurricanes were obtained primarily from monthly weather reports in the digital archive of the National Oceanic and Atmospheric Administration (www.aoml.noaa.gov/hrd/hurdat/mwr_pdf/)

HighestWindSpeed The maximum wind speed of hurricanes at the time of landfall in the United States obtained from NOAA (www.aoml.noaa.gov/hrd/hurdat/All_U.S._Hurricanes.html). This data is only available for storms after 1979

NDAM normalized damage (in million $)

Source Source from where the data was gathered

Elapsed.Yrs Time since hurricane

Details

This dataset was collated by Jung et. al in their study *Female hurricanes are deadlier than male hurricanes* They hypothesised that hurricanes with more feminine names might be perceived as less dangerous and hence lead to people taking fewer precautionary measures, resulting in more death and damages.

References

Kiju Jung and Sharon Shavitta and Madhu Viswanathana and Joseph M. Hilbed. (2014). "Female hurricanes are deadlier than male hurricanes." *Proceedings of the National Academy of Sciences*, 111(24), 8782-8787.

Yang Liu and Alex Kale and Tim Althoff and Jeff Heer. (2020). Boba: Authoring and Visualizing Multiverse Analyzes. *arXiv preprint* arXiv: 2007.05551 .

`inside` *Pass code into the multiverse*

Description

Add code to the multiverse using the function using a function call, or an assignment operator, which is a wrapper around the function

Usage

```
inside(multiverse, .expr, .label = NULL, .execute_default = TRUE)
```

Arguments

`multiverse`	A multiverse object. A multiverse object is an S3 object which can be defined using `multiverse()`
`.expr`	R syntax. All the operations that the user wants to perform within the multiverse can be passed. Since it accepts a single argument, chunks of code can be passed using '{}'. See example for details.
`.label`	It is extracted automatically from the code block of type `multiverse` when run in an RMarkdown document. This should be used only within an RMarkdown document. Defaults to NULL.
`.execute_default`	Should the default multiverse be executed as part of this call?

Details

The inside function can only access variables which can be accessed at the same environment where the multiverse object was declared in.

To perform a multiverse analysis, we will need to write code to be executed within the multiverse. The `inside()` functions allows us to do this. Use `inside()` to pass any code to the specified multiverse, which is captured as an expression. To define multiple analysis options in the code passed to the multiverse, use the `branch()` function.

Value

a multiverse object

See Also

`branch()` for more details on how to declare multiple analysis options.

The `inside` function only stores the code, and does not execute any code at this step. To execute, we provide separate functions. See `execute()` for executing the code.

Instead of using the `inside()` function, an alternate implementation of the multiverse is using the assignment operator, '<-' (please refer to examples below).

Note: the `inside()` function can only access variables which can be accessed at the same level as the multiverse object. Since `inside()` is merely an interface to add analysis to the multiverse object, even if it is being called by another function, it is actually manipulating the multiverse object, which will have a different parent environment from where `inside()` is called, and hence not have access to variables which might be accessible in the environment within the function from where `inside()` is called.

Examples

```
M.1 <- multiverse()

# using `inside` to declare multiverse code
inside(M.1, {
  data <- rnorm(100, 50, 20)

  x.mean <- mean(data, trim = branch(
    trim_values,
    "trim_none" ~ 0,
    "trim_1pc" ~ 0.05,
    "trim_5pc" ~ 0.025,
    "trim_10pc" ~ 0.05
  ))
})

# declaring multiple options for a data processing step (calculating a new variable)
data(durante)
df <- durante

inside(M.1, {
  df <- df  %>%
   mutate( ComputedCycleLength = StartDateofLastPeriod - StartDateofPeriodBeforeLast ) %>%
    mutate( NextMenstrualOnset = branch(menstrual_calculation,
                               "mc_option1" ~ StartDateofLastPeriod + ComputedCycleLength,
                               "mc_option2" ~ StartDateofLastPeriod + ReportedCycleLength,
                                     "mc_option3" ~ StartDateNext
  ))
})
```

`mulitiverse_code_block` *Create custom code blocks for multiverse analysis*

Description

An easier way to interact with a multiverse object by using a custom code block

Details

This is a custom code block that allows the users to interface directly with a created multiverse object. This allows users to implement analysis in RMarkdown without using auxiliary functions such as `inside()`. However, `inside()` is still required to define and execute multiverse analyses in a RScript using this package. See `vignette("multiverse-in-rmd")` for more details.

In RStudio, you can create a shortcut for this using RStudio AddIns (we recommend `Cmd + Opt + M` in Mac and `Ctrl + Alt + M` in Windows). To add a shortcut, go to Tools > AddIns > Browse AddIns... > Keyboard shortcuts. Search for *insert multiverse code chunk* and add a keyboard shortcut to this function. Once you have set this up, the keyboard shortcut will create a code block in any RMarkdown document.

Code Block Options

The multiverse code blocks require two named arguments:

1. label: this is a unique identifier for each code block. If the same label is used for two different code blocks, the code associated with the previous block in the multiverse will be overwritten by the subsequent one. If a code block is created using the keyboard shortcut, it will auto-generate a (unique) label
2. inside: the multiverse object this code block will be associated with. Defaults to "M"

`multiverse` *Create a new multiverse object*

Description

Constructs a new multiverse object which enables conducting a multiverse analysis

Usage

```
multiverse()

is_multiverse(x)

is.multiverse(x)
```

Arguments

`x`	An object

Details

To perform a multiverse analysis, the user needs to first constructs a new multiverse object. The user can declare multiple analysis pathways within this multiverse object, and execute these analyses. The multiverse object contains the following slots:

- code: This slot stores the user's code for conducting a multiverse analysis. The user can define multiple analysis pathways at each step in the analysis using the 'branch' call. It supports tidyverse syntax.
- parameters: This slot contains a named list of lists. It contains each parameter defined in the code using the 'branch()' function, and the options defined for each parameter (as a list).
- conditions: This slot contain a list of conditions: if any of the parameter values are conditional on a specific value of another parameter, these can be defined in the code using 'branch_assert()' or 'branch_exclude()'.
- current_parameter_assignment: This slot is a list which contains a single option assigned for each parameter defined in the 'code'.
- multiverse_table: This slot contains a table (in implementation, a tibble which is a rectangular data structure) where each column of the table will be a unique parameter. The table will contains every possible combination of options for each parameter — the number of rows corresponds to the number of different analysis paths. The table also contains, for each row, a list of option assignments for each parameter ('parameter_assignment' column), code for executing that particular analysis (of type 'expression'), and environments where each code will be executed.

Value

An empty multiverse object

'TRUE' if the object inherits from the 'multiverse' class.

'TRUE' if the object inherits from the 'multiverse' class.

Examples

```
M <- multiverse()
```

`parse_multiverse` *Parse the multiverse syntax to identify branches*

Description

In a multiverse, the user can define different values that a parameter can take using the `branch()` call. The `parse_multiverse()` identifies the `branch()` calls defined in the analysis syntax and parses them into a list of parameters and the corresponding values that each parameter can take. This function is called automatically and not exported.

Usage

```
parse_multiverse(.multiverse, .expr, .code, .label)
```

Arguments

.multiverse	The multiverse object which will contain the analysis
.expr	The expression that is being parsed
.code	All the code that has been passed to the multiverse
.label	The label of the code block or inside call which was used to pass the code being parsed into the multiverse

Value

parse_multiverse() returns a list of lists. the list of parameters and the list of conditions. The list of parameters is a named list which defines all the values that each defined parameter can take. The list of conditions defines, if any of the parameter values are conditional on a specific value of another parameter, the condition.

---

print *Accessing contents of the multiverse object*

Description

prints objects of class multiverse. Provides a quick overview of the multiverse by listing the parameters and conditions declared, to get an overview of the number of combinations. The function outputs upto 20 of the parameters declared along with their option names (upto 5), and upto 10 of the conditions declared.

Usage

```
## S3 method for class 'multiverse'
print(x, ...)
```

Arguments

x	An object of class multiverse
...	further arguments passed to or from other methods. Currently ignored.

style_multiverse_code *Stylises the code of multiverse for printing*

Description

Stylises the code declared with a multiverse object according to the tidyverse style guide.

Usage

```
style_multiverse_code(.code)
```

Arguments

.code	a quoted (or unevaluated) expression surrounded; usually the expression within a single multiverse code block or inside function.

Details

Since multiverse captures unevaluated expressions, the formatting is not preserved. However, if we want to view the code that is delcared for a particular multiverse analysis, the lack of formatting would lead to difficulty in readability. ‘style_multiverse_code‘ is primarily intended for internal use to stylise the output based on the tidyverse style guide and custom rules necessary for multiverse.

Value

an object of class vertical

userlogs *Userlogs*

Description

Data collected by Jansen et al. in their study, *Evaluating the Efficiency of Physical Visualizations* which investigated factors contributing to the efficiency of physical visualizations.

Format

A data frame with 512 observations and 19 variables:

subject Subject identifier

group Group / experiment session number in which the participant was involved in the experiment

formerSubject Yes/No. Whether subject had participated in a previous experiment conducted by Jansen et al.

conditionrank

modalityname Name of the modality of interaction. One of "Virtual Mouse", "Virtual prop", "Physical touch", "Physical no-touch"

repetition 1/2. Whether the participant was interacting with the visualization for the first time. Participants interacted with visualizations of two different datasets and this variable stores the order.

modality Index of the modality of interaction

question Index of question asked to the participant as part of the experiment. Each participant was asked 4 questions. All questions involved a comparison task.

trial Each participant performed 32 trials. Participants answered 4 questions for 8 different datasets which resulted in 32 trials per participant.

datasetname Each participant was presented with 8 different datasets through the visualizations. These were: "army", "carmortality", "education" "externaldebt", "grosscapital", "health", "hiv", "military"

readingTime Time taken by participant to read the visualization in seconds

error If the question was answered correctly by the participants.

duration If the question was answered correctly by the participants.

perceivedDifficulty Self reported perceived difficulty of the task by each participant.

perceivedTime Self reported perceived time taken to perform the task by each participant.

References

Yvonne Jansen and Pierre Dragicevic and Jean-Daniel Fekete. (2013) "Evaluating the Efficiency of Physical Visualizations." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* pp. 2593-2602.

`vis_correlation` *Participants responses in the study by Harrison et al., "Ranking Visualizations of correlation according to Weber's law"*

Description

A dataset containing the aggregated responses of the participants. In this study, participants were asked to make multiple judgements about the correlation.

Format

A data frame with 502 rows and 26 variables:

participant Participant identifier

vis Type of visualization used

rdirection The direction of the slope of the line (positive or negative)

sign The direction of the slope of the line (1 or -1)

visandsign A combination variable of visualization type and rdirection

rbase The coefficient of correlation using which the stimuli was generated

approach NA

jnd Estimated JND value for that participant

condition Condition that the participant was placed in (which is a combination of vis, rbase and approach)

References

Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay and Fanny Chevalier. (2019). "Increasing the transparency of research papers with explorable multiverse analyses." *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* pp. 1-15.

Kay, Matthew, and Jeffrey Heer. (2016). "Beyond Weber's law: A second look at ranking visualizations of correlation." *IEEE transactions on visualization and computer graphics* 22.1: 469-478.

Lane Harrison, Fumeng Yang, Steven Franconeri, and Remco Chang. (2014). "Ranking visualizations of correlation using Weber's law." *IEEE transactions on visualization and computer graphics* 20.12: 1943-1952.

`%when%` *Define constraints for combinations of analysis paths in the multiverse*

Description

Users can specify conditions for which a particular analysis path may or may not be valid using either the `%when%` operator or the `branch_assert` function.

Arguments

`Logical`	predicates defined in terms of the parameters and their options in the multiverse.

Details

A user can specify multiple different analysis options at each step of the analysis process using `branch()`. However, it is possible that the values of certain parameters might be conditional on the values of certain other parameters.

The conditional or the "implies" relationship between two statements, *A* and *A* has the meaning, "if *A* is true, then *B* is also true." We evaluate this relationship using classical logic: $A \implies B$ is an abbreviation for $\neg A \lor B$

See Also

`vignette("conditions")`

Examples

```
M.1 <- multiverse()

# There are different ways to specifying conditions
# One way is to use the %when% operator
# the %when% operator can be specified after the option name
inside(M.1, {
    df <- data.frame (x = 1:10 ) %>%
        mutate( y = branch( values_y, TRUE, FALSE )) %>%
        mutate( z = branch(values_z,
            "constant" ~ 5,
            "linear" ~ x + 1,
            "sum" %when% (values_y == TRUE) ~ (x + y)
        ))
})

# or it can be specified after the expression for computing the option value
inside(M.1, {
    df <- data.frame (x = 1:10 ) %>%
        mutate( y = branch( values_y, TRUE, FALSE )) %>%
        mutate( z = branch(values_z,
            "constant" ~ 5,
            "linear" ~ x + 1,
            "sum" ~ (x + y) %when% (values_y == TRUE)
        ))
})

# an advantage of the '%when' operator is that it can also be used when the
# option names are not specified for branches.
# when option names are not specified for branches, option names are assigned to
# the branches. For character, logical or numeric expressions, option names are of the
# same type (i.e. character, logical or numeric expressions respectively)
# For expressions of type symbol or call, options names are characters strings
# containing the expression.
# see the next two examples:
M.2 <- multiverse()

inside(M.2, {
 df <- data.frame (x = 1:10 ) %>%
   mutate( y = branch( values_y, TRUE, FALSE )) %>%
   mutate( z = branch(values_z,
          5,
          x + 1,
          (x + y) %when% (values_y == TRUE)
   ))
})

M.3 <- multiverse()
inside(M.3, {
 df <- data.frame (x = 1:10 ) %>%
   filter( branch( values_x,
       TRUE,
```

```
        x > 2 | x < 6
    )) %>%
    mutate( z = branch(values_z,
           5,
           x + 1,
           (x^2) %when% (values_x == 'x > 2 | x < 6')
    ))
})

# or it can be specified after the expression for computing the option value
M.4 <- multiverse()
inside(M.4, {
    df <- data.frame (x = 1:10 ) %>%
        mutate( y = branch( values_y, TRUE, FALSE )) %>%
        mutate( z = branch(values_z,
            "constant" ~ 5,
            "linear" ~ x + 1,
            "sum" ~ x + y
        )) %>%
        branch_assert( values_z != "sum" | values_y == TRUE )
})
```

Index