Package 'mverse'

May 23, 2022

Type Package

Title Tidy Multiverse Analysis Made Simple

Version 0.1.0

Maintainer Michael Jongho Moon <michael.moon@mail.utoronto.ca>

Description Extends 'multiverse' package
(Sarma A., Kale A., Moon M., Taback N., Chevalier F., Hullman J., Kay M., 2021)
<doi:10.31219/osf.io/yfbwm>, which allows users perform to create explorable
multiverse analysis in R. This extension provides an additional level of
abstraction to the 'multiverse' package with the aim of creating
user friendly syntax to researchers, educators, and students in statistics.
The 'mverse' syntax is designed to allow piping and takes hints from
the 'tidyverse' grammar. The package allows users to define and inspect
multiverse analysis using familiar syntax in R.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 3.6), multiverse

Imports magrittr (>= 1.5), rlang, dplyr, tidyr, tidyselect, stringr,
stats, broom, grDevices, igraph, ggraph, ggplot2, cowplot

Suggests tidyverse, scales, MASS, testthat (>= 2.1.0), pkgdown (>=
1.5.1), covr, knitr, rmarkdown, kableExtra

RoxygenNote 7.1.2

VignetteBuilder knitr

URL https://github.com/mverseanalysis/mverse/,
https://mverseanalysis.github.io/mverse/

NeedsCompilation no

Author Michael Jongho Moon [aut, cre],
Haoda Li [aut],
Mingwei Xu [aut],
Nathan Taback [aut],
Fanny Chevalier [aut],
Alison Gibbs [ctb]

Repository CRAN

Date/Publication 2022-05-23 08:10:02 UTC

R topics documented:

`add_branch_condition` *Add branch conditions to a* `mverse` *object.*

Description

This method adds one or more branch conditions to an existing `mverse` object. Branch conditions are used to specify an option in one branch dependent on an option in another branch.

Usage

```
add_branch_condition(.mverse, ...)

## S3 method for class 'mverse'
add_branch_condition(.mverse, ...)
```

Arguments

.mverse	a mverse object.
...	branch conditions.

Value

a mverse object.

See Also

Other branch condition functions: branch_condition()

Examples

```
# Define branches and add them to an \code{mverse} object.
y <- mutate_branch(alldeaths, log(alldeaths + 1))
distribution <- family_branch(poisson, gaussian)
# You can match branching options by providing the options
# the way provide them when defining branches.
match_poisson <- branch_condition(alldeaths, poisson)
mv <- mverse(hurricane) %>%
  add_mutate_branch(y) %>%
  add_family_branch(distribution) %>%
  add_branch_condition(match_poisson)
summary(mv)
# You can also condition to reject a pair of options by
# setting reject = TRUE.
match_log_lin <- branch_condition(log(alldeaths + 1), poisson, reject = TRUE)
mv <- add_branch_condition(mv, match_log_lin)
summary(mv)
```

---

add_family_branch *Add family branches to a* mverse *object.*

Description

This method adds one or more family branches to an existing mverse object. Family branches are used to define options for the analysis distributions when using glm_mverse().

Usage

```
add_family_branch(.mverse, ...)

## S3 method for class 'mverse'
add_family_branch(.mverse, ...)
```

Arguments

.mverse	a mverse object.
...	family_branch objects.

Value

The resulting mverse object.

See Also

Other family branch functions: family_branch()

Examples

```
# Define a family branch.
model_distributions <- family_branch(
  gaussian, poisson(link = "log")
)
# Create a mverse and add the branch.
mv <- create_multiverse(hurricane) %>%
  add_family_branch(model_distributions)
```

---

add_filter_branch *Add filter branches to a* mverse *object.*

Description

This method adds one or more filter branches to an existing mverse object. Filter branches are used to define options for conditions for selecting subsets of data rows.

Usage

```
add_filter_branch(.mverse, ...)

## S3 method for class 'mverse'
add_filter_branch(.mverse, ...)
```

Arguments

.mverse	a mverse object.
...	filter_branch objects.

Value

The resulting mverse object.

See Also

Other filter branch functions: `filter_branch()`

Examples

```
# Define a filter branch.
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  !Name %in% c("Katrina"),
  !Name %in% c("Katrina"),
  TRUE # include all
)
# Create a mverse and add the branch.
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers)
```

`add_formula_branch` *Add formula branches to a* `mverse` *object.*

Description

This method adds one or more formula branches to an existing `mverse` object. Formula branches are used to specify model structure options for the analysis.

Usage

```
add_formula_branch(.mverse, ...)

## S3 method for class 'mverse'
add_formula_branch(.mverse, ...)
```

Arguments

`.mverse`	a `mverse` object.
`...`	`formula_branch` objects.

Value

The resulting `mverse` object.

See Also

Other formula branch functions: `formula_branch()`

Examples

```
# Define a formula branch.
model_specifications <- formula_branch(
  y ~ femininity,
  y ~ femininity + hurricane_strength,
  y ~ femininity * hurricane_strength
)
# Create a mverse, add the branch.
mv <- create_multiverse(hurricane) %>%
  add_formula_branch(model_specifications)
```

`add_mutate_branch` *Add mutate branches to a* `mverse` *object.*

Description

This method adds one or more mutate branches to an existing `mverse` object. Mutate branches are used to define options for adding a new column to the analysis dataset.

Usage

```
add_mutate_branch(.mverse, ...)

## S3 method for class 'mverse'
add_mutate_branch(.mverse, ...)
```

Arguments

`.mverse`	a `mverse` object.
`...`	`mutate_branch` objects.

Value

The resulting `mverse` object.

See Also

Other mutate branch functions: `mutate_branch()`

Examples

```
# Define mutate branches.
hurricane_strength <- mutate_branch(
  # damage vs. wind speed vs.pressure
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014,
  # Standardized versions
```

```
  scale(NDAM),
  scale(HighestWindSpeed),
  -scale(Minpressure_Updated_2014),
)
y <- mutate_branch(
  alldeaths, log(alldeaths + 1)
)
# Create a mverse and add the branches.
mv <- create_multiverse(hurricane) %>%
  add_mutate_branch(hurricane_strength) %>%
  add_mutate_branch(y)
# You can also add multiple branches with a single call.
mv <- create_multiverse(hurricane) %>%
  add_mutate_branch(hurricane_strength, y)
```

`AIC` *Display the AIC and BIC score of the fitted models across the multiverse*

Description

Display the AIC and BIC score of `glm` regression results across the multiverse.

Usage

```
## S3 method for class 'glm_mverse'
AIC(object, ..., k = 2)

## S3 method for class 'glm_mverse'
BIC(object, ...)

AIC(object, ..., k = 2)

BIC(object, ...)
```

Arguments

`object`	a `glm_mverse` object.
`...`	ignored. for compatibility only.
`k`	ignored. for compatibility only.

Value

a multiverse table as a tibble

`branch_condition` *Create a new branch condition.*

Description

A branch condition conditions option `x` to depend on option `y`. When the branch condition is added to a `mverse` object, option `x` is executed only when `y` is. Use `reject = TRUE`, to negate the condition.

Usage

```
branch_condition(x, y, reject = FALSE)
```

Arguments

`x`	option 1
`y`	option 2
`reject`	if TRUE, the condition rejects universes with option 1 and option 2

Value

A `branch_condition` object.

See Also

Other branch condition functions: `add_branch_condition()`

Examples

```
# Example branches.
y <- mutate_branch(alldeaths, log(alldeaths + 1))
model <- formula_branch(y ~ femininity * strength, y ~ femininity + strength)
# Define a new branch condition.
match_poisson <- branch_condition(alldeaths, poisson)
# Define a branch condition that reject an option dependent on another.
match_log_lin <- branch_condition(log(alldeaths + 1), poisson, reject = TRUE)
```

execute_multiverse *Execute the entire multiverse.*

Description

This method executes the analysis steps defined in the `mverse` objected across the entire multiverse.

Usage

```
execute_multiverse(.mverse)

## S3 method for class 'mverse'
execute_multiverse(.mverse)
```

Arguments

`.mverse`	a `mverse` object.

Value

The resulting `mverse` object.

Examples

```
# Define a mutate branch.
hurricane_strength <- mutate_branch(
  # damage vs. wind speed vs.pressure
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014,
  # Standardized versions
  scale(NDAM),
  scale(HighestWindSpeed),
  -scale(Minpressure_Updated_2014),
)
# Create a mverse and add the branch.
mv <- create_multiverse(hurricane) %>%
  add_mutate_branch(hurricane_strength)
# The branched variables are not populated across the multiverse yet.
# Execute the multiverse; the variables are populated after the execution.
execute_multiverse(mv)
```

`extract` *Extract branched values.*

Description

`extract` returns a tibble of selected values across the multiverse in a long format.

Usage

```
extract(...)

## S3 method for class 'mverse'
extract(
  .mverse,
  columns = NULL,
  nuni = NULL,
  frow = NULL,
  include_branch_options = TRUE,
  ...
)
```

Arguments

`...`	Ignored.
`.mverse`	a `mverse` object.
`columns`	a character vector of column names to extract.
`nuni`	a positive integer for the number of universes to extract.
`frow`	proportion of rows to extract from each universe.
`include_branch_options`	when `TRUE` (default), include the mutate statements used to specified the options for each branched columns

Details

This method extracts data values across the multiverse. You can specify a subset of data to extract using `columns`, `universe`, `nuni`, and `frow`.

You can specify the columns to extract from each universe by passing the column names as a character vector to `columns`. The default values is `NULL` extracting all columns with branches.

Use `universe` to specify a set of universes by their integer ids. Use `nuni` to specify the number of universes to extract data from. The method then selects the subset randomly. Specifying `universe` manually will override `nuni` value. By default, they are both set to `NULL` and the method returns data from all universes.

Use `frow` to randomly extract a fraction of data from each universe. The default value is `NULL` and all rows are returned as they are. Note if you select 1 the method will return shuffle rows in each universe before returning them. If `frow` is greater than 1, the method randomly samples rows with replacement.

Value

a tibble containing the selected columns across the multiverse.

Examples

```
# Define mutate branches.
hurricane_strength <- mutate_branch(
  # damage vs. wind speed vs.pressure
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014,
  # Standardized versions
  scale(NDAM),
  scale(HighestWindSpeed),
  -scale(Minpressure_Updated_2014),
)
y <- mutate_branch(
  alldeaths, log(alldeaths + 1)
)
# Create a mverse and add the branches.
mv <- create_multiverse(hurricane) %>%
  add_mutate_branch(hurricane_strength, y)
execute_multiverse(mv)
# Extract all branched columns from all universes
extract(mv)
# Specify the columns to extract from each universe using columns
# You can select both branched and non-branched columns
extract(mv, columns = c("hurricane_strength", "NDAM"))
# Specify the universe to extract from using universe
extract(mv, universe = 1)
# Specify the number of universes to extract from using nuni
# The universes are randomly selected
extract(mv, nuni = 3)
# Specify the proportion of data to extract from each universe using
# frow. The rows are randomly selected
extract(mv, frow = 0.7)
```

---

family_branch *Create a new family branch.*

Description

Create a new family branch.

Usage

```
family_branch(..., name = NULL)
```

Arguments

...	branch definition expressions.
name	Name for the new family.

Value

a family_branch object.

See Also

Other family branch functions: add_family_branch()

Examples

```
# Define a family branch.
model_distributions <- family_branch(
  gaussian, poisson(link = "log")
)
# Create a mverse and add the branch.
mv <- create_multiverse(hurricane) %>%
  add_family_branch(model_distributions)
```

---

filter_branch *Create a new filter branch.*

Description

Create a new filter branch.

Usage

```
filter_branch(..., name = NULL)
```

Arguments

...	branch definition expressions.
name	Name for the new filter.

Value

a filter_branch object.

See Also

Other filter branch functions: add_filter_branch()

Examples

```
# Define a filter branch.
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  !Name %in% c("Katrina"),
  !Name %in% c("Katrina"),
  TRUE # include all
)
# Create a mverse and add the branch.
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers)
```

`formula_branch` *Create a new formula branch.*

Description

Create a new formula branch.

Usage

```
formula_branch(..., name = NULL)
```

Arguments

`...`	branch definition expressions.
`name`	Name for the new formula.

Value

a `formula_branch` object.

See Also

Other formula branch functions: `add_formula_branch()`

Examples

```
# Define a formula branch.
model_specifications <- formula_branch(
  y ~ femininity,
  y ~ femininity + hurricane_strength,
  y ~ femininity * hurricane_strength
)
# Create a mverse, add the branch.
mv <- create_multiverse(hurricane) %>%
  add_formula_branch(model_specifications)
```

`glm.nb_mverse` *Fit negative binomial regression models across the multiverse*

Description

`glm.nb_mverse` fits `MASS::glm.nb` across the multiverse according to model specifications provided by `formula_branch`. At least one `formula_branch` must have been added.

Usage

```
glm.nb_mverse(.mverse)
```

Arguments

`.mverse`	a `mverse` object.

Value

A `mverse` object with `glm.nb` fitted.

See Also

Other model fitting functions: `glm_mverse()`, `lm_mverse()`

Examples

```
# Fitting \code{glm.nb} models across a multiverse.
hurricane_strength <- mutate_branch(
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014
)
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  TRUE # include all
)
model_specifications <- formula_branch(
  alldeaths ~ femininity,
  alldeaths ~ femininity + hurricane_strength
)
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers) %>%
  add_mutate_branch(hurricane_strength) %>%
  add_formula_branch(model_specifications) %>%
  glm.nb_mverse()
```

glm_mverse *Fit generalized linear regression models across the multiverse.*

Description

glm_mverse fits glm across the multiverse according to model specifications provided by formula_branch. At least one formula_branch must have been added. You can also specify the underlying error distribution and the link function by adding a family_branch. If no family_branch has been provided, it follows the default behaviour of glm using the Gaussian distribution with an identity link.

Usage

```
glm_mverse(.mverse)
```

Arguments

.mverse	a mverse object.

Value

A mverse object with glm fitted.

See Also

Other model fitting functions: glm.nb_mverse(), lm_mverse()

Examples

```
# Fitting \code{glm} models across a multiverse.
hurricane_strength <- mutate_branch(
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014
)
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  TRUE # include all
)
model_specifications <- formula_branch(
  alldeaths ~ femininity,
  alldeaths ~ femininity + hurricane_strength
)
model_distributions <- family_branch(poisson)
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers) %>%
  add_mutate_branch(hurricane_strength) %>%
```

```
add_formula_branch(model_specifications) %>%
add_family_branch(model_distributions) %>%
glm_mverse()
```

`hurricane` *Data on Atlantic hurricanes in the U.S. between 1950 and 2012.*

Description

A dataset for the study conducted by Jung et al. (2014) in doi: 10.1073/pnas.1402786111Female hurricanes are deadlier than male hurricanes.

Usage

```
hurricane
```

Format

A data frame with 94 rows and 12 variables:

Year Year in which the hurricane landed on U.S.

Name Name of the hurricane.

MasFem Femininity index of the hurricane name collected by Jung et al. (1 - very masculine; 11 - very feminine).

MinPressure_before Minimum pressure of the hurricane at the time of landfall in the U.S. (original).

Minpressure_Updated_2014 Minimum pressure of the hurricane at the time of landfall in the U.S. (updated).

Gender_MF Gender indicator for the hurricane name based on `MasFem` index (1 - `MasFem` > 6; 0 otherwise).

Category Hurricane category on a scale of 1 to 5, with 5 being the most severe.

alldeaths Number of fatalities.

NDAM Normalized damage in 2013 U.S. million dollars.

Elapsed.Yrs Time since hurricane.

Source Source from where the data was gathered.

HighestWindSpeed Maximum wind speed.

MasFem_MTUrk Femininity index of the hurricane name collected by Simonsohn et al.

NDAM15 Normalized damage in 2015 U.S. million dollars.

Details

The dataset was collected by Jung et al. in their study *Female hurricanes are deadlier than male hurricanes*. Their study didn't include hurricanes Katrina and Audrey which were deemed as outliers. Simonsohn et al. collected the extra data for doi: 10.1038/s415620200912zSpecification curve analysis including an additional femininity index based on an MTUrk survey and updated normalized damage amount in 2015 U.S. dollars.

This dataset includes data prepared by Jung et al. (2014) as well as those prepared by Simonsohn et al. (2020). Specifically, all data on Katrina and Audrey are from Simonsohn et al. (2020) except minimum pressure updated in 2014. They were retrieved from Continental United States Hurricane Impacts/Landfalls 1851-2021 table maintained by U.S. National Oceanic and Atmospheric Administration. Maximum wind speed, femininity index from MTUrk survey, and 2015 damage amounts are also from Simonsohn et al. (2020).

Source

Kiju Jung, Sharon Shavitt, Madhu Viswanathan, and Joseph M. Hilbe. (2014). "Female hurricanes are deadlier than male hurricanes." *Proceedings of the National Academy of Sciences*, 111(24), 8782-8787. doi: 10.1073/pnas.1402786111

Uri Simonsohn, Joseph P. Simmons, and Leif D. Nelson. (2020). "Specification curve analysis" *Nature Human Behaviour*, 4, 1208–14. doi: 10.1038/s415620200912z

---

`lm_mverse` *Fit linear regression models across the multiverse.*

Description

`lm_mverse` fits `lm` across the multiverse according to model specifications provided by `formula_branch`. At least one `formula_branch` must have been added.

Usage

```
lm_mverse(.mverse)
```

Arguments

`.mverse`	a `mverse` object.

Value

A `mverse` object with `lm` fitted.

See Also

Other model fitting functions: `glm.nb_mverse()`, `glm_mverse()`

Examples

```
# Fitting \code{lm} models fitted across a multiverse.
hurricane_strength <- mutate_branch(
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014
)
y <- mutate_branch(
  alldeaths, log(alldeaths + 1)
)
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  TRUE # include all
)
model_specifications <- formula_branch(
  y ~ femininity,
  y ~ femininity + hurricane_strength
)
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers) %>%
  add_mutate_branch(hurricane_strength, y) %>%
  add_formula_branch(model_specifications) %>%
  lm_mverse()
```

`multiverse_tree` *Plot a multiverse tree diagram.*

Description

A multiverse tree diagram displays the branching combination of all the branches added to the given `mverse` object taking any branch conditions defined. The method also allows zooming into a subset of branches using `branches` parameter.

Usage

```
multiverse_tree(
  .mverse,
  label = FALSE,
  branches = NULL,
  label_size = NULL,
  label_angle = 0
)
```

Arguments

.mverse	A mverse object.
label	A logical. Display options as labels when TRUE.
branches	A character vector. Display a subset of branches when specified. Display all when NULL.
label_size	A numeric. Set size of option labels.
label_angle	A numeric. Rotate option labels.

Value

A ggplot object displaying the multiverse tree.

Examples

```
# Display a multiverse tree with multiple branches.
outliers <- filter_branch(!Name %in% c("Katrina", "Audrey"), TRUE)
femininity <- mutate_branch(MasFem, Gender_MF)
strength <- mutate_branch(
  NDAM, HighestWindSpeed, Minpressure_Updated_2014, log(NDAM)
)
y <- mutate_branch(alldeaths, log(alldeaths + 1))
model <- formula_branch(y ~ femininity * strength, y ~ femininity + strength)
distribution <- family_branch(poisson, gaussian)
mv <- mverse(hurricane) %>%
  add_filter_branch(outliers) %>%
  add_mutate_branch(femininity, strength, y) %>%
  add_formula_branch(model) %>%
  add_family_branch(distribution)
multiverse_tree(mv)
# Display a multiverse tree with branch conditions.
match_poisson <- branch_condition(alldeaths, poisson)
match_log_lin <- branch_condition(log(alldeaths + 1), gaussian)
add_branch_condition(mv, match_poisson)
add_branch_condition(mv, match_log_lin)
multiverse_tree(mv)
# Display a multiverse tree for a subset of branches
# with label for each option.
multiverse_tree(mv, branches = c("y", "distribution"), label = TRUE)
# adjusting size and orientation of the labels
multiverse_tree(mv, branches = c("y", "distribution"),
  label = TRUE, label_size = 4, label_angle = 45)
```

`mutate_branch` *Create a new mutate branch.*

Description

Create a new mutate branch.

Usage

```
mutate_branch(..., name = NULL)
```

Arguments

`...`	branch definition expressions.
`name`	Name for the new variable.

Value

a mutate_branch object.

See Also

Other mutate branch functions: `add_mutate_branch()`

Examples

```
# Define mutate branches.
hurricane_strength <- mutate_branch(
  # damage vs. wind speed vs.pressure
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014,
  # Standardized versions
  scale(NDAM),
  scale(HighestWindSpeed),
  -scale(Minpressure_Updated_2014),
)
# Create a mverse and add the branch.
mv <- create_multiverse(hurricane) %>%
  add_mutate_branch(hurricane_strength)
```

mverse *Create a new* mverse *object*

Description

Constructs a new mverse object which extends `multiverse::multiverse` object.

Usage

```
mverse(data)

create_multiverse(data)
```

Arguments

data	source dataframe.

Value

A `mverse` object with the source dataframe attached.

Examples

```
# Create a mverse object.
mv <- mverse(hurricane)
# create_multiverse() is an alias of mverse().
mv <- create_multiverse(hurricane)
```

print.branch *Print method for* *_branch *objects.*

Description

Print method for *_branch objects.

Usage

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

Arguments

x	a branch object.
...	ignored. for compatibility only.

Value

No return value, called for printing only.

soccer *Number of cards given for each referee-player pair in soccer.*

Description

A dataset containing card counts between 2,053 soccer players playing in the first male divisions of England, Germany, France, and Spain in the 2012-2013 season and 3,147 referees that these players played under in professional matches. The dataset contains other covariates including 2 independent skin tone ratings per player. Each line represents a player-referee pair.

Usage

```
soccer
```

Format

A data frame with 146,028 rows and 26 variables:

playerShort short player ID

player player name

club player club

leagueCountry country of player club (England, Germany, France, and Spain)

birthday player birthday

height player height (in cm)

weight player weight (in kg)

position detailed player position

games number of games in the player-referee dyad

victories victories in the player-referee dyad

ties ties in the player-referee dyad

defeats losses in the player-referee dyad

goals goals scored by a player in the player-referee dyad

yellowCards number of yellow cards player received from referee

yellowReds number of yellow-red cards player received from referee

redCards number of red cards player received from referee

rater1 skin rating of photo by rater 1 (5-point scale ranging from "very light skin" to "very dark skin")

rater2 skin rating of photo by rater 2 (5-point scale ranging from "very light skin" to "very dark skin")

refNum unique referee ID number (referee name removed for anonymizing purposes)

refCountry unique referee country ID number (country name removed for anonymizing purposes)

meanIAT mean implicit bias score (using the race IAT) for referee country, higher values correspond to faster white | good, black | bad associations

nIAT sample size for race IAT in that particular country

seIAT standard error for mean estimate of race IAT

meanExp mean explicit bias score (using a racial thermometer task) for referee country, higher values correspond to greater feelings of warmth toward whites versus blacks

nExp sample size for explicit bias in that particular country

seExp standard error for mean estimate of explicit bias measure

Details

The skin colour of each player was rated by two independent raters, rater1 and rater2, and the 5-point scale values were scaled to 0 to 1 - i.e., 0, 0.25, 0.5, 0.75, 1.

Source

Silberzahn, R., Uhlmann, E. L., Martin, D. P., Anselmi, P., Aust, F., Awtrey, E. C., ... Nosek, B. A. (2018, August 24). Many analysts, one dataset: Making transparent how variations in analytical choices affect results. Retrieved from https://osf.io/gvm2z/

`spec_curve` *Display a specification curve across the multiverse.*

Description

`spec_curve` returns the specification curve as proposed by Simonsohn, Simmons, and Nelson (2020) <doi:10.1038/s41562-020-0912-z>. `spec_curve` are available for `mverse` objects fitted with `lm_mverse()`, `glm_mverse()`, and `glm.nb_mverse()`. Notice that the order of universes may not correspond to the order in the summary table.

Usage

```
spec_curve(.object, var, ...)

## S3 method for class 'lm_mverse'
spec_curve(
  .object,
  var,
  conf.int = TRUE,
  conf.level = 0.95,
  option = names(multiverse::parameters(.object)),
  universe_order = FALSE,
  color_order = FALSE,
  color = NULL,
  branch_order = NULL,
  point_size = 0.25,
```

```
  grid_size = 2,
  point_alpha = 1,
  brewer_palette = "Set2",
  yaxis_text_size = 8,
  ...
)

## S3 method for class 'glm_mverse'
spec_curve(
  .object,
  var,
  conf.int = TRUE,
  conf.level = 0.95,
  option = names(multiverse::parameters(.object)),
  universe_order = FALSE,
  color_order = FALSE,
  color = NULL,
  branch_order = NULL,
  point_size = 0.25,
  grid_size = 2,
  point_alpha = 1,
  brewer_palette = "Set2",
  yaxis_text_size = 8,
  ...
)

## S3 method for class 'glm.nb_mverse'
spec_curve(
  .object,
  var,
  conf.int = TRUE,
  conf.level = 0.95,
  option = names(multiverse::parameters(.object)),
  universe_order = FALSE,
  color_order = FALSE,
  color = NULL,
  branch_order = NULL,
  point_size = 0.25,
  grid_size = 2,
  point_alpha = 1,
  brewer_palette = "Set2",
  yaxis_text_size = 8,
  ...
)
```

Arguments

.object	a glm.nb_mverse object.

var	name for the variable to show.
...	ignored.
conf.int	when TRUE (default), the estimate output includes the confidence intervals.
conf.level	the confidence level of the confidence interval returned using conf.int = TRUE. Default value is 0.95.
option	a vector of branches to show the options included.
universe_order	when TRUE, order the universes according to the order in the summary table.
color_order	when TRUE, the estimated value will be ordered according to the color.
color	an expression to indicate how colors are assigned to markers. By default, colors are assigned based on 'p.value <= 0.05'.
branch_order	name for the branch to order.
point_size	size of points on the top plot.
grid_size	size of points on the bottom plot.
point_alpha	alpha level of points and point ranges.
brewer_palette	name of colorbrewer palette for the plot.
yaxis_text_size	text size of y-axis label

Value

a specification curve plot for the estimates

Source

Uri Simonsohn, Joseph P. Simmons, and Leif D. Nelson. (2020). "Specification curve analysis" *Nature Human Behaviour*, 4, 1208–14. doi: 10.1038/s415620200912z

Examples

```
# Display a specification curve for \code{lm} models
# fitted across the multiverse.
femininity <- mutate_branch(
  MasFem > 6, MasFem > mean(MasFem)
)
model <- formula_branch(
  alldeaths ~ femininity,
  alldeaths ~ femininity + HighestWindSpeed
)
mv <- mverse(hurricane) %>%
  add_mutate_branch(femininity) %>%
  add_formula_branch(model) %>%
  lm_mverse()
spec_curve(mv, var = "femininityTRUE")
# plot based on 90% confidence interval
spec_curve(mv, var = "femininityTRUE", color = p.value < .1)
```

```
# Display a specification curve for \code{glm} models
# fitted across the multiverse.
femininity <- mutate_branch(
  MasFem > 6, MasFem > mean(MasFem)
)
model <- formula_branch(
  alldeaths ~ femininity,
  alldeaths ~ femininity + HighestWindSpeed
)
fam <- family_branch(gaussian)
mv <- mverse(hurricane) %>%
  add_mutate_branch(femininity) %>%
  add_formula_branch(model) %>%
  add_family_branch(fam) %>%
  glm_mverse()
spec_curve(mv, var = "femininityTRUE")
# plot based on 90% confidence interval
spec_curve(mv, var = "femininityTRUE", color = p.value < .1)



# Display a specification curve for \code{glm.nb} models
# fitted across the multiverse.
femininity <- mutate_branch(
  MasFem > 6, MasFem > mean(MasFem)
)
model <- formula_branch(
  alldeaths ~ femininity,
  alldeaths ~ femininity + HighestWindSpeed
)
mv <- mverse(hurricane) %>%
  add_mutate_branch(femininity) %>%
  add_formula_branch(model) %>%
  glm.nb_mverse()
spec_curve(mv, var = "femininityTRUE")
# plot based on 90% confidence interval
spec_curve(mv, var = "femininityTRUE", color = p.value < .1)
```

`summary` *Display the multiverse table with results.*

Description

This method returns the multiverse table displaying all universes defined by the multiverse. Each row corresponds to a universe and each column represents a branch.

Usage

```
## S3 method for class 'mverse'
summary(object, ...)

## S3 method for class 'lm_mverse'
summary(object, conf.int = TRUE, conf.level = 0.95, output = "estimates", ...)

## S3 method for class 'glm_mverse'
summary(object, conf.int = TRUE, conf.level = 0.95, output = "estimates", ...)

## S3 method for class 'glm.nb_mverse'
summary(object, conf.int = TRUE, conf.level = 0.95, output = "estimates", ...)
```

Arguments

object	a glm.nb_mverse object.
...	Ignored.
conf.int	When TRUE (default), the estimate output includes the confidence intervals.
conf.level	The confidence level of the confidence interval returned using conf.int = TRUE. Default value is 0.95.
output	The output of interest. The possible values are "estimates" ("e"), "df", "deviance" ("de"), and "aic" ("bic"). Alternatively, the first letters may be used. Default value is "estimates".

Details

When you pass a mverse objected fitted with model, the summary table includes results of the fitted models across the multiverse.

Value

a multiverse table as a tibble.

Examples

```
# Displaying the multiverse table without any fitted values.
hurricane_strength <- mutate_branch(
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014
)
mv <- create_multiverse(hurricane) %>%
  add_mutate_branch(hurricane_strength)
summary(mv)
## Displaying after adding a a filter branch.
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
```

```
  !Name %in% c("Katrina"),
  !Name %in% c("Katrina"),
  TRUE # include all
)
mv <- add_filter_branch(mv, hurricane_outliers)
summary(mv)



# Displaying the multiverse table with \code{lm} models fitted.
hurricane_strength <- mutate_branch(
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014
)
y <- mutate_branch(
  alldeaths, log(alldeaths + 1)
)
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  TRUE # include all
)
model_specifications <- formula_branch(
  y ~ femininity,
  y ~ femininity + hurricane_strength
)
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers) %>%
  add_mutate_branch(hurricane_strength, y) %>%
  add_formula_branch(model_specifications) %>%
  lm_mverse()
summary(mv)



# Displaying the multiverse table with \code{glm} models fitted.
hurricane_strength <- mutate_branch(
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014
)
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  TRUE # include all
)
model_specifications <- formula_branch(
  alldeaths ~ femininity,
  alldeaths ~ femininity + hurricane_strength
)
model_distributions <- family_branch(poisson)
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers) %>%
  add_mutate_branch(hurricane_strength) %>%
```

```
  add_formula_branch(model_specifications) %>%
  add_family_branch(model_distributions) %>%
  glm_mverse()
summary(mv)



# Displaying the multiverse table with \code{glm.nb} models fitted.
hurricane_strength <- mutate_branch(
  NDAM,
  HighestWindSpeed,
  Minpressure_Updated_2014
)
hurricane_outliers <- filter_branch(
  !Name %in% c("Katrina", "Audrey", "Andrew"),
  TRUE # include all
)
model_specifications <- formula_branch(
  alldeaths ~ femininity,
  alldeaths ~ femininity + hurricane_strength
)
mv <- create_multiverse(hurricane) %>%
  add_filter_branch(hurricane_outliers) %>%
  add_mutate_branch(hurricane_strength) %>%
  add_formula_branch(model_specifications) %>%
  glm.nb_mverse()
summary(mv)
```

`ttest` *Performs one or two sample t-tests on data columns.*

Description

`ttest_mverse` performs t-tests across the multiverse. If x or y is specified, then performs one and two sample t-tests on specified columns of the data. If both x and y are NULL, then performs t.test based on the formula branches.

Usage

```
ttest_mverse(
  .mverse,
  x = NULL,
  y = NULL,
  alternative = "two.sided",
  mu = 0,
  paired = FALSE,
  var.equal = FALSE,
  conf.level = 0.95
)
```

Arguments

.mverse	a mverse object.
x	(optional) column name of data within mverse object
y	(optional) column name of data within mverse object
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
mu	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
paired	a logical indicating whether you want a paired t-test.
var.equal	a logical variable indicating whether to treat the two variances as being equal.
conf.level	confidence level of the interval.

Value

A ttest_mverse object.

Examples

```
# Performing a unpaired two sample t-test.
mv <- mverse(soccer)
x <- mutate_branch(
  ((rater1 + rater2) / 2) > mean((rater1 + rater2) / 2),
  ifelse(rater1 > rater2, rater1, rater2) >
    mean(ifelse(rater1 > rater2, rater1, rater2))
)
y <- mutate_branch(
  redCards, yellowCards, yellowReds
)
two_sample_form <- formula_branch(y ~ x)
mv <- mv %>%
  add_mutate_branch(x, y) %>%
  add_formula_branch(two_sample_form)
ttest_mverse(mv)
```

Index