Package 'tidyindex'

November 16, 2023

Type Package

Title A Tidy Data Pipeline to Construct, Compare, and Analyse Indexes

Version 0.1.0

Description Construct and analyse indexes in a pipeline tidy workflow. 'tidyindex' contains modules for transforming variables, aggregating variables across time, reducing data dimension through weighting, and fitting distributions. A manuscript describing the methodology can be found at <https://github.com/huizezhang-sherry/paper-tidyindex>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/huizezhang-sherry/tidyindex

BugReports https://github.com/huizezhang-sherry/tidyindex/issues

Imports cli, dplyr, generics, ggplot2, glue, purrr, rlang (>= 1.1.0), tidyr, tidyselect, tsibble, vctrs

RoxygenNote 7.2.3

Suggests covr, knitr, lmomco, lubridate, rmarkdown, slider, SPEI, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

VignetteBuilder knitr

NeedsCompilation no

Author H. Sherry Zhang [aut, cre, cph]
(<https://orcid.org/0000-0002-7122-1463>),
Dianne Cook [aut] (<https://orcid.org/0000-0002-3813-7155>),
Ursula Laa [aut] (<https://orcid.org/0000-0002-0249-6439>),
Nicolas Langrené [aut] (<https://orcid.org/0000-0001-7601-4618>),
Patricia Menéndez [aut] (<https://orcid.org/0000-0003-0701-6315>)

Maintainer H. Sherry Zhang <huize.zhang@monash.edu>

Repository CRAN

Date/Publication 2023-11-16 11:20:02 UTC

R topics documented:

add_paras *Add parameters to an index table object*

Description

The function joins the parameter table to the ‘paras‘ element of an index table object.

Usage

```
add_paras(data, para_tbl, by)
```

Arguments

`data`	a `idx_tbl` object
`para_tbl`	a tibble or data frame object with parameter of variables
`by`	a single column name (support tidyselect) in the ‘para_tbl‘ that maps to the variable name in the data

Value

an index object

Examples

```
init(gggi) |> add_paras(gggi_weights, by = "variable")
```

`compute_indexes` *Calculate multiple indexes at once*

Description

Calculate multiple indexes at once

Usage

```
compute_indexes(.data, ...)

## S3 method for class 'idx_res'
augment(x, .id = ”.id”, ...)
```

Arguments

`.data`	an `idx_tbl` object
`...`	Unused, included for generic consistency only
`x`	an `idx_res` object, calculated from `compute_indexes`
`.id`	a character string, the name of the first column

Value

an `idx_res` object

Examples

```
library(dplyr)
library(lmomco)
library(generics)
res <- tenterfield |>
  mutate(month = lubridate::month(ym)) |>
  init(id = id, time = ym, group = month) |>
  compute_indexes(
    spi = idx_spi(),
    spei = idx_spei(.lat = lat, .tavg = tavg),
    edi = idx_edi()
 )
```

`dimension_reduction` *The dimension reduction module*

Description

The module combines multiple variables into a new variable. The new variable can be a linear combination of the original variables, `aggregate_linear()`, or a geometric mean of the original variables, `aggregate_geometry()`, or created from an user formula input, `aggregate_manual()`.

Usage

```
dimension_reduction(data, ...)

aggregate_linear(formula, weight)

aggregate_geometrical(formula)

aggregate_manual(formula)
```

Arguments

`data`	used in `dimension_reduction()`, an `idx_tbl` object, see [tidyindex::init()]
`...`	used in `dimension_reduction()`, a dimension reduction object of `dim_red` class, currently one of `aggregate_linear()`, `aggregate_geometrical()`, or `aggregate_manual()`.
`formula`	the formula to evaluate
`weight`	used in `aggregate_linear()`, the column of the linear weights from the `roles` element in an index table object. See [tidyindex::add_paras()]

Value

an index table object

Examples

```
dt <- gggi |>
  dplyr::select(country, sex_ratio_at_birth:healthy_life_expectancy) |>
  init()

dt |>
  dimension_reduction(health = aggregate_manual(
    ~sex_ratio_at_birth * 0.693 + healthy_life_expectancy * 0.307))
dt |>
  add_paras(gggi_weights, by = variable) |>
  dimension_reduction(health = aggregate_linear(
    ~sex_ratio_at_birth:healthy_life_expectancy, weight = var_weight))
dt |>
  dimension_reduction(health = aggregate_geometrical(
    ~sex_ratio_at_birth:healthy_life_expectancy)
  )
```

distribution_fit *The distribution fit module*

Description

This module fits a distribution to the variable of interest. Currently implemented distributions are: gamma, `dist_gamma()`, generalized logistic, `dist_glo()`, generalized extreme value, `dist_gev()`, and Pearson Type III, `dist_pe3()`

Usage

```
distribution_fit(data, ...)

dist_gamma(var, method = "lmoms", .n_boot = 1, .boot_seed = 123)

dist_glo(var, method = "lmoms", .n_boot = 1, .boot_seed = 123)

dist_gev(var, method = "lmoms", .n_boot = 1, .boot_seed = 123)

dist_pe3(var, method = "lmoms", .n_boot = 1, .boot_seed = 123)
```

Arguments

`data`	an index table object
`...`	a distribution fit object, currently implemented are `dist_gamma()`, `dist_glo()`, `dist_gev()`, and `dist_pe3()`
`var`	used in `dist_*()` functions, the variable to fit
`method`	used in `dist_*()` functions, the fitting method, currently support "lmoms" for L-moment fit
`.n_boot`	the number of bootstrap replicate, default to 1
`.boot_seed`	the seed to generate bootstrap replicate, default to 123

Value

an index table object

Examples

```
library(dplyr)
library(lmomco)
tenterfield |>
  mutate(month = lubridate::month(ym)) |>
  init(id = id, time = ym, group = month) |>
  temporal_aggregate(.agg = temporal_rolling_window(prcp, scale = 12)) |>
  distribution_fit(.fit = dist_gamma(.agg, method = "lmoms"))
```

---

gggi *Global Gender Gap Index (2023)*

Description

The Global Gender Gap Index combines 14 variables from four dimensions to measure the gender parity across 146 countries in the world.

Usage

```
gggi

gggi_weights
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 146 rows and 22 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 14 rows and 7 columns.

Details

The dataset includes country, region, GGGI score and rank, the combined four dimensions (Economic Participation and Opportunity, Educational Attainment, Health and Survival, and Political Empowerment), and variables under each dimensions. The variable composition of each dimension is as follows:

* Economic Participation and Opportunity: Labour force participation, Wage equality for similar work, Estimated earned income, Legislators, senior officials and managers, and Professional and technical workers

* Educational attainment: Literacy rate, Enrolment in primary education, Enrolment in secondary education, Enrolment in tertiary education

* Health and survival: Sex ratio at birth and Healthy life expectancy

* Political empowerment: Women in parliament, Women in ministerial positions, and Years with female head of state

Variable names are cleaned with [janitor::clean_names()].

The weight data is extracted from page 65 of the Global Gender Gap Report (see reference), see page 61 for the region classification.

References

https://www3.weforum.org/docs/WEF_GGGR_2023.pdf

`hdi` *Human Development Index (2022)*

Description

Human Development Index (2022)

Usage

```
hdi

hdi_scales
```

Format

A tibble with three columns:

id the row number

country 191 countries with computed HDI

hdi the HDI index value

life_exp life expectancy

exp_sch expected schooling

avg_sch average schooling

gni_pc GNI per capital, logged

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 4 rows and 5 columns.

References

https://hdr.undp.org/data-center/human-development-index#/indicies/HDI

init *Initialise the tidyindex pipeline*

Description

Initialise an index table object with a data frame or a tibble.

Usage

```
init(data, ...)

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

Arguments

data	a tibble or data frame to be converted into a index object
...	arguments to give variables roles, recorded in the `paras` element of the index table object, currently used for `id` and `time`
x	an index object

Value

an index table object

Examples

```
init(hdi)
init(gggi)
```

normalise *The normalise module*

Description

The normalise module takes a probability value from a distribution fit `norm_quantile()` to convert based on the normal quantile function

Usage

```
normalise(data, ...)

norm_quantile(var)
```

Arguments

`data`	an index table object
`...`	the expression to be evaluated
`var`	used in `norm_quantile()`; the variable to be converted

Value

an index table object

Examples

```
library(dplyr)
library(lmomco)
tenterfield |>
  mutate(month = lubridate::month(ym)) |>
  init(id = id, time = ym, group = month) |>
  temporal_aggregate(.agg = temporal_rolling_window(prcp, scale = 12)) |>
  distribution_fit(.fit = dist_gamma(.agg, method = "lmoms")) |>
  normalise(index = norm_quantile(.fit))
```

`rescaling` *The rescaling module*

Description

The rescale module changes the scale of the variable(s) using one of the available rescaling functions: `rescale_zscore()`, `rescale_minmax()`, and `rescale_center`.

Usage

```
rescaling(data, ...)

rescale_zscore(var, na.rm = TRUE)

rescale_minmax(var, min = NULL, max = NULL, na.rm = TRUE, censor = TRUE)

rescale_center(var, na.rm = TRUE)
```

Arguments

`data`	an index table object, see [tidyindex::init()]
`...`	used in `rescaling`, a rescaling object of class `rescale`, currently one of the `rescale_zscore()`, `rescale_minmax()`, and `rescale_center()`,
`var`	the variable(s) to rescale, accept tidyselect syntax
`na.rm`	used in `rescale_*()`, logical, whether to remove NAs
`min, max`	used in `rescale_minmax()`, the minimum and maximum value
`censor`	used in `rescale_minmax()`, logical; whether to censor points outside min and max, if provided

Value

an index table object

Examples

```
dt <- hdi |> init()
dt |> rescaling(life_exp = rescale_zscore(life_exp))
dt |> rescaling(life_exp2 = rescale_minmax(life_exp, min = 20, max = 85))
```

---

swap_values *Testing alternatives*

Description

The two functions allows you to substitute a value/expression in the pipeline with other options. These functions will evaluate the modified pipeline step, as well as its prior and subsequent steps to create different versions of the index.

Usage

```
swap_values(data, .var, .param, .values)

swap_exprs(data, .var, .exprs)
```

Arguments

`data`	an `idx_tbl` object
`.var`	the name of the variable, which the step is tested for alternatives
`.param`	the name of the parameter to swap
`.values, .exprs`	a list of values or expressions

Value

an index table

Examples

```
library(generics)
hdi_paras <- hdi_scales |>
dplyr::add_row(dimension = "Education", name = "Education",
               var = "sch", min = 0, max = 0) |>
  dplyr::mutate(weight = c(1/3, 0, 0, 1/3, 1/3),
                weight2 = c(0.1, 0, 0, 0.8, 0.1),
                weight3 = c(0.8, 0, 0, 0.1, 0.1),
                weight4 = c(0.1, 0, 0, 0.1, 0.8))
```

```
dt <- hdi |>
  init(id = country) |>
  add_paras(hdi_paras, by = var) |>
  rescaling(life_exp = rescale_minmax(life_exp, min = min, max = max)) |>
  rescaling(exp_sch = rescale_minmax(exp_sch, min = min, max = max)) |>
  rescaling(avg_sch = rescale_minmax(avg_sch, min = min, max = max)) |>
  rescaling(gni_pc = rescale_minmax(gni_pc, min = min, max = max)) |>
  dimension_reduction(sch = aggregate_manual(~(exp_sch + avg_sch)/2)) |>
  dimension_reduction(index = aggregate_linear(~c(life_exp, sch, gni_pc),
                      weight = weight))


dt2 <- dt |>
  swap_values(.var = "index", .param = weight,
            .value = list(weight2, weight3, weight4))
augment(dt2)

dt3 <- dt |>
  swap_exprs(.var = index, .exprs = list(
             aggregate_geometrical(~c(life_exp, sch, gni_pc))))
augment(dt3)
```

`temporal_aggregate` *The temporal processing module*

Description

The temporal processing module is used to aggregate data along the temporal dimension. Current available aggregation recipe includes `temporal_rolling_window`.

Usage

```
temporal_aggregate(data, ...)

temporal_rolling_window(
  var,
  scale,
  .before = 0L,
  .step = 1L,
  .complete = TRUE,
  rm.na = TRUE,
  ...
)
```

Arguments

`data`	an index table object, see [tidyindex::init()]
`...`	an temporal processing object of class `temporal_agg`

`var`	the variable to aggregate
`scale`	numeric, the scale (window) of the aggregation
`.before, .step, .complete`	see `slide_dbl`
`rm.na`	logical, whether to remove the first few rows with NAs

Value

an index table object

Examples

```
tenterfield |>
  init(time = ym) |>
  temporal_aggregate(.agg = temporal_rolling_window(prcp, scale = 12))

# multiple ids (groups), and multiple scales
queensland |>
  dplyr::filter(id %in% c("ASN00029038", "ASN00029127")) |>
  init(id = id, time = ym) |>
  temporal_aggregate(temporal_rolling_window(prcp, scale = c(12, 24)))
```

`tenterfield` *Weather data for in-situ stations in Queensland from 1990 to 2020*

Description

Weather data for in-situ stations in Queensland from 1990 to 2020

Usage

```
tenterfield

aus_climate

queensland
```

Format

A tibble with 9 columns:

id station ID, ASN000xxxxx

ym date in ‘tsibble::yearmonth‘ format

prcp aggregated monthly precipitation from daily data

tmax, tmin, tavg maximum/minimum/ average temperature

long, lat longitude and latitude of the station

name station name

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 52373 rows and 9 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 11252 rows and 9 columns.

`theme_benchmark` *A ggplot2 theme for benchmarking the index series*

Description

A ggplot2 theme for benchmarking the index series

Usage

```
theme_benchmark(yintercept = -2, linetype = "dashed")
```

Arguments

`yintercept`	intercept
`linetype`	linetype

Value

a ggplot2 object

Examples

```
if (require("ggplot2", quietly = TRUE) ){
dplyr::tibble(x = 1:100, y = rnorm(100, sd = 2)) |>
  ggplot(aes(x = x, y =y )) +
  geom_line() +
  theme_benchmark()
}
```

`trans_thornthwaite` *Drought-related index functions*

Description

The functions are used for quick computing of some common drought indexes built from wrappers of the underlying modules. For more customised needs, users may build their own indexes from the modules.

Usage

```
trans_thornthwaite(var, lat, na.rm = FALSE, verbose = TRUE)

idx_spi(data, .prcp, .dist = dist_gamma(), .scale = 12)

idx_spei(
  data,
  .tavg,
  .lat,
  .prcp,
  .pet_method = trans_thornthwaite(),
  .scale = 12,
  .dist = dist_glo()
)

idx_rdi(
  data,
  .tavg,
  .lat,
  .prcp,
  .pet_method = trans_thornthwaite(),
  .scale = 12
)

idx_edi(data, .tavg, .lat, .prcp, .scale = 12)
```

Arguments

`var` the variable to be transformed, see [tidyindex::variable_trans()] and [SPEI::thornthwaite()]

`lat, na.rm, verbose`
see [SPEI::thornthwaite]

`data` an `id_tbl` object

`.dist` the distribution used for distribution fit, see [tidyindex::distribution_fit()]

`.scale` the temporal aggregation scale, see [tidyindex::temporal_aggregation()]

`.tavg, .lat, .prcp`
variables to be used in the index calculation, see Details

`.pet_method` the method used for calculating potential evapotranspitation, currently only `trans_thornthwaite()`

Details

Below explains the steps wrapped in each index and the intermediate variables created.

The `idx_spi()` function performs

1. a temporal aggregation on the input precipitation series, `.prcp`, as `.agg`,
2. a distribution fit step on the aggregated precipitation , `.agg`, as `.fit`, and
3. a normalising step on the fitted values, `.fit`, as `.index`

The `idx_spei()` function performs

1. a variable transformation step on the inut average temperature, `.tavg`, to obtain the potential evapotranspiration, `.pet`,
2. a dimension reduction step to calculate difference series, `.diff`, between the input precipitation series, `.prcp`, and `.pet`,
3. a temporal aggregation step on the difference series, `.diff`, as `.agg`,
4. a distribution fit step on the aggregated series, `.agg`, as `.fit`, and
5. a normalising step on the fitted value, `.fit`, to obtain `.index`.

The `idx_rdi()` function performs

1. a variable transformation step on the input average temperature, `.tavg`, to obtain potential evapotranspiration `.pet`,
2. a dimension reduction step to calculate the ratio of input precipitation, `.prcp`, to `.pet` as `.ratio`,
3. a temporal aggregation step on the ratio series, `.ratio`, as `.agg`
4. a variable transformation step to take the log10 of the aggregated series, `.agg`, as `.y`, and
5. a rescaling step to rescale `.y` by zscore to obtain `.index`.

The `idx_edi()` function performs

1. a dimension reduction step to aggregate the input precipitation series, `prcp`, as `.mult`,
2. a temporal aggregation step on the aggregated precipitation series (`.mult`) as `.ep`, and
3. a rescaling step to rescale `.ep` by zscore to obtain `.index`.

Value

an index table object

Examples

```
library(dplyr)
library(lmomco)
dt <- tenterfield |>
  mutate(month = lubridate::month(ym)) |>
  init(id = id, time = ym, group = month)

dt |> idx_spi()
dt |> idx_spi(.scale = c(12, 24))
dt |> idx_spei(.lat = lat, .tavg = tavg)
dt |> idx_rdi(.lat = lat, .tavg = tavg)
dt |> idx_edi(.lat = lat, .tavg = tavg)
```

variable_trans — *The variable transformation module*

Description

The variable transformation module is used to transform a single variable in the index table object. The transformation is specified by a variable transformation object of class `var_trans`, created by `trans_*` functions. Currently, the following transformation functions are supported: `trans_log10`, `trans_quadratic`, `trans_square_root`, and `trans_cubic_root`.

Usage

```
variable_trans(data, ...)

trans_log10(var)

trans_quadratic(var)

trans_square_root(var)

trans_cubic_root(var)
```

Arguments

`data`	an index table object
`...`	an variable transformation recipe of class `var_trans`, created by `trans_*` function, the transformation recipe to be evaluated
`var`	used in `trans_*` functions, the variable to be transformed

Value

an index table object

Examples

```
hdi |> init() |> variable_trans(gni_pc = trans_log10(gni_pc))
```

Index