Package ‘gravitas’

October 13, 2022

Type Package

Title Explore Probability Distributions for Bivariate Temporal Granularities

Version 0.1.3

Depends R(>= 3.5.0)

Imports lubridate(>= 1.7.4), dplyr(>= 0.8.0), tsibble (>= 0.8.0), tibble(>= 2.1.1), rlang(>= 0.3.4), tidyr(>= 0.8.3), ggplot2 (>= 3.1.1), stats, stringr(>= 1.4.0), lvplot, ggridges, shiny, RColorBrewer, ineq, magrittr, utils

Suggests knitr, tsibbledata (>= 0.1.0), purrr (>= 0.3.2), testthat (>= 2.1.0), covr, rmarkdown, readr

VignetteBuilder knitr

Maintainer Sayani Gupta <gupta.sayani@gmail.com>

Description Provides tools for systematically exploring large quantities of temporal data across cyclic temporal granularities (deconstructions of time) by visualizing probability distributions. Cyclic time granularities can be circular, quasi-circular or aperiodic. 'gravitas' computes cyclic single-order-up or multiple-order-up granularities, check the feasibility of creating plots for any two cyclic granularities and recommend probability distributions plots for exploring periodicity in the data.

BugReports https://github.com/Sayani07/gravitas/issues

License GPL-3

URL https://github.com/Sayani07/gravitas/

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 7.1.0.9000

NeedsCompilation no

Author Sayani Gupta [aut, cre] (<https://orcid.org/0000-0003-0643-5358>),
Rob Hyndman [aut, ths] (<https://orcid.org/0000-0002-2140-5352>),
Di Cook [aut, ths] (<https://orcid.org/0000-0002-3813-7155>),
Antony Unwin [aut] (<https://orcid.org/0000-0002-5841-5757>)

Repository CRAN

Date/Publication 2020-06-25 12:10:07 UTC

R topics documented:

`create_gran` *Build dynamic temporal granularities*

Description

Create time granularities that accommodate for periodicities in data, both single and multiple order up. Periodic ones might include time granularities like minute of the day, hour of the week and aperiodic calendar categorizations may include day of the month or week of the quarter. For non-temporal data, supports only periodic deconstructions.

Usage

```
create_gran(
  .data,
  gran1 = NULL,
  hierarchy_tbl = NULL,
  label = TRUE,
  abbr = TRUE,
  ...
)
```

Arguments

.data A tsibble object.

gran1 the granularity to be created. For temporal data, any combination of "second", "minute", "qhour", "hhour", "hour", "day", "week", "fortnight ,"month", "quarter", "semester" or "year" can be chosen in the form of finer to coarser unit. For example, for the granularity hour of the week, value is "hour_week".

hierarchy_tbl A hierarchy table specifying the hierarchy of units and their relationships.

label Logical. TRUE will display the month as an ordered factor of character string such as "January", "February". FALSE will display the month as an ordered factor such as 1 to 12, where 1 stands for January and 12 for December.

abbr logical. FALSE will display abbreviated labels.

... Other arguments passed on to individual methods.

Value

A tsibble with an additional column of granularity.

Examples

```
library(dplyr)
library(ggplot2)
library(lvplot)
# Search for granularities
smart_meter10 %>%
  search_gran(highest_unit = "week")

# Screen harmonies from the search list
## Not run:
smart_meter10 %>%
  harmony(
    ugran = "day",
    filter_in = "wknd_wday"
  )

## End(Not run)
# visualize probability distribution of
# the harmony pair (wknd_wday, hour_day)
smart_meter10 %>%
dplyr::filter(customer_id == "10017936") %>%
  prob_plot(
    gran1 = "wknd_wday",
    gran2 = "hour_day",
    response = "general_supply_kwh",
    plot_type = "quantile",
    quantile_prob = c(0.1, 0.25, 0.5, 0.75, 0.9)
  ) +
  scale_y_sqrt()

#' # Compute granularities for non-temporal data
```

```
library(tsibble)
cricket_tsibble <- cricket %>%
mutate(data_index = row_number()) %>%
as_tsibble(index = data_index)

hierarchy_model <- tibble::tibble(
  units = c("index", "over", "inning", "match"),
  convert_fct = c(1, 20, 2, 1)
)
cricket_tsibble %>%
  create_gran(
    "over_inning",
    hierarchy_model
  )

  cricket_tsibble %>%
  filter(batting_team %in% c("Mumbai Indians",
                             "Chennai Super Kings"))%>%
  prob_plot("inning", "over",
  hierarchy_model,
  response = "runs_per_over",
  plot_type = "lv")

# Validate if given column in the data set
# equals computed granularity
validate_gran(cricket_tsibble,
  gran = "over_inning",
  hierarchy_tbl = hierarchy_model,
  validate_col = "over"
)
```

cricket *Cricket data set for different seasons of Indian Premier League*

Description

The Indian Premier League played by teams representing different cities in India from 2008 to 2016.

Usage

```
cricket
```

Format

A tibble with 8560 rows and 11 variables.

season years representing IPL season

match_id match codes

batting_team name of batting team

bowling_team name of bowling team

inning innings of the match

over overs of the inning

wicket number of wickets in each over

dot_balls number of balls with no runs in an over

runs_per_over Runs for each over

run_rate run rate for each over

Source

https://www.kaggle.com/josephgpinto/ipl-data-analysis/data

Examples

```
data(cricket)

library(tsibble)
library(dplyr)
library(ggplot2)

# convert data set to a tsibble ----
cricket_tsibble <- cricket %>%
  mutate(data_index = row_number()) %>%
  as_tsibble(index = data_index)
# set the hierarchy of the units in a table ----
hierarchy_model <- tibble::tibble(
  units = c("index", "over", "inning", "match"),
  convert_fct = c(1, 20, 2, 1)
)
# Compute granularities ----
cricket_tsibble %>%
  create_gran("over_inning",
              hierarchy_model)
# Visualise distribution of runs across granularities ----
cricket_tsibble %>%
  filter(batting_team %in% c("Mumbai Indians",
                             "Chennai Super Kings"))%>%
  prob_plot("inning", "over",
  hierarchy_model,
  response = "runs_per_over",
  plot_type = "lv")
```

gran_advice *Advice summaries for granularities*

Description

Recommendations on plot choices, interaction, number of observations and intra or inter facet homogeneity. Important summaries before drawing distribution plots.

Usage

```
gran_advice(.data, gran1, gran2, hierarchy_tbl = NULL, ...)
```

Arguments

`.data`	a tsibble.
`gran1, gran2`	granularities.
`hierarchy_tbl`	A hierarchy table specifying the hierarchy of units and their relationships.
`...`	other arguments to be passed for appropriate labels.

Value

Summary check points before visualizing distribution across bivariate granularities

Examples

```
library(dplyr)
library(ggplot2)

smart_meter10 %>%
filter(customer_id == "10017936") %>%
  gran_advice(gran1 = "wknd_wday",
            gran2 = "hour_day")

# choosing quantile plots from plot choices
smart_meter10 %>%
filter(customer_id == "10017936") %>%
  prob_plot(
    gran1 = "wknd_wday",
    gran2 = "hour_day",
    response = "general_supply_kwh",
    plot_type = "quantile",
    quantile_prob = c(0.1, 0.25, 0.5, 0.75, 0.9)
  ) +
  scale_y_sqrt()
```

gran_obs *Cross tabulation of granularities useful for validating if number of observations are sufficient to compute probability distributions*

Description

Cross tabulation of granularities useful for validating if number of observations are sufficient to compute probability distributions

Usage

```
gran_obs(.data, gran1, gran2, hierarchy_tbl = NULL)
```

Arguments

`.data` A tsibble object.

`gran1` One of the temporal granularities to check for harmonies.

`gran2` The second temporal granularity in the pair.

`hierarchy_tbl` A hierarchy table specifying the hierarchy of units and their relationships for non-temporal case.

Value

A tibble with number of observations for each granularity.

Examples

```
library(tsibbledata)
vic_elec %>% gran_obs("hour_day", "day_week")
```

---

harmony *Get possible set of harmonies for a given tsibble*

Description

Interaction of pair of granularities, categorised as harmony and clash. harmony () screens out harmony pairs from list of all possible pairs of granularities.

Usage

```
harmony(
  .data,
  ugran = "year",
  lgran = NULL,
  hierarchy_tbl = NULL,
  filter_in = NULL,
  filter_out = NULL,
  facet_h = NULL,
  ...
)
```

Arguments

`.data` A tsibble object.

`ugran` Typically set as the most coarse unit required in the analysis. Default is "year".

`lgran` For "regular" tsibble, lgran is the interval of the tsibble. It needs to be specified for "irregular" time intervals. Typically serves as the finest unit required for analysis.

`hierarchy_tbl` A hierarchy table specifying the hierarchy of units and their relationships.

filter_in Choices of temporal units to be kept. Can be column names if #' required granularity already exists in the tsibble. For example, a column with public holidays which needs to be treated as granularity, can be included here for checking how it interacts with other granularities.

filter_out Choices of temporal units to be discarded.

facet_h highest level of facets allowed.

... added arguments to be passed.

Value

A tibble of harmonies and their levels.The levels can be used to decide which granularities to be plotted across x-axis/facets for exploratory aid.

Examples

```
library(tsibbledata)
vic_elec %>% harmony(lgran = "hour", ugran = "week")
```

is_harmony *Check if two temporal granularities are harmonies*

Description

Interaction of pair of granularities, categorised as harmony and clash

Usage

```
is_harmony(
  .data,
  gran1,
  gran2,
  hierarchy_tbl = NULL,
  facet_h = NULL,
  x_h = NULL
)
```

Arguments

.data A tsibble object.

gran1 One of the temporal granularities to check for harmonies.

gran2 The second temporal granularity in the pair.

hierarchy_tbl A hierarchy table specifying the hierarchy of units and their relationships.

facet_h highest level of facet variable that can be considered in harmony pair.

x_h highest level of x-axis variable that can be considered in harmony pair.

Value

TRUE if two granularties are harmonies.

Examples

```
library(tsibbledata)
vic_elec %>% is_harmony("hour_day", "day_week")
```

---

print.gran_advice *Advice summaries for granularities*

Description

Recommendations on plot choices, interaction, number of observations and intra or inter facet homogeneity. Important summaries before drawing distribution plots.

Usage

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

Arguments

x	An object of class gran_advice
...	other arguments to be passed for appropriate labels.

Value

Print check points before visualizing distribution across bivariate granularities

Examples

```
library(dplyr)
library(ggplot2)

smart_meter10 %>%
filter(customer_id == "10017936") %>%
  gran_advice(gran1 = "wknd_wday",
            gran2 = "hour_day")
```

`prob_plot` *Plotting probability distributions across granularities*

Description

Plot probability distribution of univariate series across bivariate temporal granularities.

Usage

```
prob_plot(
  .data,
  gran1 = NULL,
  gran2 = NULL,
  hierarchy_tbl = NULL,
  response = NULL,
  plot_type = NULL,
  quantile_prob = c(0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.99),
  facet_h = NULL,
  symmetric = TRUE,
  alpha = 0.8,
  threshold_nobs = NULL,
  ...
)
```

Arguments

`.data` a tsibble

`gran1` the granularity which is to be placed across facets. Can be column names if required granularity already exists in the tsibble. For example, a column with public holidays which needs to be treated as granularity, can be included here.

`gran2` the granularity to be placed across x-axis. Can be column names if required granularity already exists in the tsibble.

`hierarchy_tbl` A hierarchy table specifying the hierarchy of units and their relationships.

`response` response variable to be plotted.

`plot_type` type of distribution plot. Options include "boxplot", "lv" (letter-value), "quantile", "ridge" or "violin".

`quantile_prob` numeric vector of probabilities with value in [0,1] whose sample quantiles are wanted. Default is set to "decile" plot.

`facet_h` levels of facet variable for which facetting is allowed while plotting bivariate temporal granularities.

`symmetric` If TRUE, symmetic quantile area <- is drawn. If FALSE, only quantile lines are drawn instead of area. If TRUE, length of quantile_prob should be odd and ideally the quantile_prob should be a symmetric vector with median at the middle position.

`alpha` level of transperancy for the quantile area

`threshold_nobs` the minimum number of observations below which only points would be plotted

`...` other arguments to be passed for customising the obtained ggplot object.

Value

a ggplot object which can be customised as usual.

Examples

```
library(tsibbledata)
library(ggplot2)
library(tsibble)
library(lvplot)
library(dplyr)

smart_meter10 %>%
filter(customer_id %in% c("10017936")) %>%
prob_plot(
  gran1 = "day_week", gran2 = "hour_day",
  response = "general_supply_kwh", plot_type = "quantile",
  quantile_prob = c(0.1, 0.25, 0.5, 0.75, 0.9),
  symmetric = TRUE,
   outlier.colour = "red",
  outlier.shape = 2, palette = "Dark2"
)

cricket_tsibble <- cricket %>%
  mutate(data_index = row_number()) %>%
  as_tsibble(index = data_index)

hierarchy_model <- tibble::tibble(
  units = c("index", "over", "inning", "match"),
  convert_fct = c(1, 20, 2, 1)
)

cricket_tsibble %>%
  prob_plot("inning", "over",
    hierarchy_tbl = hierarchy_model,
    response = "runs_per_over",
    plot_type = "lv"
  )
```

`run_app` *Runs a shiny app demonstrating functionalities of gravitas*

Description

Runs a local shiny app that demonstrates how distributions of univariate time series can be explored across bivariate time granularities

Usage

```
run_app()
```

Value

opens a local shiny app

Author(s)

Sayani Gupta

Examples

```
## Not run:
run_app()

## End(Not run)
```

---

search_gran *Search for granularities*

Description

Get set of possible granularities that can be considered exhaustively depending on the frequency of the data.

Usage

```
search_gran(
  .data,
  lowest_unit = NULL,
  highest_unit = NULL,
  hierarchy_tbl = NULL,
  filter_in = NULL,
  filter_out = NULL,
  ...
)
```

Arguments

.data	A tsibble object.
lowest_unit	Typically set as the finest unit required for analysis. For "regular" tsibble, lgran is the interval of the tsibble. It needs to be specified for "irregular" time intervals.For non-temporal data, default is the first unit specified in the hierarchy table.
highest_unit	Typically set as the most coarse unit required for analysis.For temporal data, default is "year" and for non-temporal data, default is set as the last unit specified in the hierarchy table.

`hierarchy_tbl`	A hierarchy table specifying the hierarchy of units and their relationships.
`filter_in`	Choices of temporal units to be kept.
`filter_out`	Choices of temporal units to be discarded.
`...`	Other arguments to be passed.

Value

Set of possible granularities.

Examples

```
library(tsibbledata)
vic_elec %>% search_gran(lowest_unit = "hour", highest_unit = "month")
```

`smart_meter10` *Smart meter data for ten households*

Description

Customer Trial data conducted as part of Smart Grid Smart City (SGSC) project (2010-2014) based in Newcastle, New South Wales and areas in Sydney. It contains half hourly interval meter readings (KWh) of electricity consumption of households.

Usage

```
smart_meter10
```

Format

A tsibble with 259, 235 rows and 3 columns.

customer_id household ID

reading_datetime Date time for which data is recorded (index)

general_supply_kwh electricity supplied to this household

Source

`https://data.gov.au/dataset/ds-dga-4e21dea3-9b87-4610-94c7-15a8a77907ef/details?q=smart-meter`

validate_gran *Validate created granularities with existing columns*

Description

Validate created granularities with existing columns

Usage

```
validate_gran(
  .data,
  gran = NULL,
  hierarchy_tbl = NULL,
  validate_col = NULL,
  ...
)
```

Arguments

`.data`	A tsibble object.
`gran`	the granularity to be created for validation.
`hierarchy_tbl`	A hierarchy table.
`validate_col`	A column in the data which acts as validator.
`...`	Other arguments passed on to individual methods.

Value

A tsibble with an additional column of granularity.

Examples

```
library(dplyr)
library(tsibble)
cricket_tsibble <- cricket %>%
  mutate(data_index = row_number()) %>%
  as_tsibble(index = data_index)

hierarchy_model <- tibble::tibble(
  units = c("index", "ball", "over", "inning", "match"),
  convert_fct = c(1, 6, 20, 2, 1)
)
cricket_tsibble %>% validate_gran(
  gran = "over_inning",
  hierarchy_tbl = hierarchy_model,
  validate_col = "over"
)
```

Index