Package: densityarea (via r-universe)

September 10, 2024

Type Package

Title Polygons of Bivariate Density Distributions

Version 0.1.0.9001

Description With bivariate data, it is possible to calculate 2-dimensional kernel density estimates that return polygons at given levels of probability. 'densityarea' returns these polygons for analysis, including for calculating their area.

License GPL (>= 3)

URL https://github.com/JoFrhwld/densityarea, https://jofrhwld.github.io/densityarea/

BugReports https://github.com/JoFrhwld/densityarea/issues

Depends R (>= 4.1)

Imports cli, dplyr, ggdensity, isoband, purrr, rlang, sf, sfheaders, tibble, vctrs

Suggests forcats, ggplot2, knitr, ragg, readr, rmarkdown, stringr, testthat (>= 3.0.0), tidyr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Repository https://jofrhwld.r-universe.dev

RemoteUrl https://github.com/jofrhwld/densityarea

RemoteRef HEAD

RemoteSha 76ee0876479e102522e157c8950dea8302607459

Contents

`density_area` *Density Area*

Description

A convenience function to get just the areas of density polygons.

Usage

```
density_area(
  x,
  y,
  probs = 0.5,
  as_sf = FALSE,
  as_list = FALSE,
  range_mult = 0.25,
  rangex = NULL,
  rangey = NULL,
  ...
)
```

Arguments

`x, y`	Numeric data dimensions
`probs`	Probabilities to compute density polygons for
`as_sf`	Should the returned values be sf::sf? Defaults to `FALSE`.
`as_list`	Should the returned value be a list? Defaults to `TRUE` to work well with tidyverse list columns
`range_mult`	A multiplier to the range of `x` and `y` across which the probability density will be estimated.
`rangex, rangey`	Custom ranges across `x` and `y` ranges across which the probability density will be estimated.
`...`	Additional arguments to be passed to `ggdensity::get_hdr()`

Details

If both `rangex` and `rangey` are defined, `range_mult` will be disregarded. If only one or the other of `rangex` and `rangey` are defined, `range_mult` will be used to produce the range of the undefined one.

Value

A list of data frames, if `as_list=TRUE`, or just a data frame, if `as_list=FALSE`.

Data frame output:

If `as_sf=FALSE`, the data frame has the following columns:

level_id An integer id for each probability level

prob The probability level (originally passed to `probs`)

area The area of the HDR polygon

sf output:

If `as_sf=TRUE`, the data frame has the following columns:

level_id An integer id for each probability level

prob The probability level (originally passed to `probs`)

geometry The `sf::st_polygon()` of the HDR

area The area of the HDR polygon

Examples

```
library(densityarea)
library(dplyr)
library(sf)

ggplot2_inst <- require(ggplot2)

# basic usage

set.seed(10)
x <- rnorm(100)
y <- rnorm(100)

density_area(x,
             y,
             probs = ppoints(50)) ->
  poly_areas_df

head(poly_areas_df)

# Plotting the relationship between probability level and area
if(ggplot2_inst){
  ggplot(poly_areas_df,
         aes(prob, area)) +
    geom_line()
}

# Tidyverse usage

data(s01)

## Data preprocessing
```

```
s01 |>
  mutate(log_F2 = -log(F2),
         log_F1 = -log(F1)) ->
  s01

### Data frame output

s01 |>
  group_by(name) |>
  reframe(density_area(log_F2,
                       log_F1,
                       probs = ppoints(10))) ->
  s01_areas_df

if(ggplot2_inst){
  s01_areas_df |>
    ggplot(aes(prob, area)) +
    geom_line()
}

### Including sf output

s01 |>
  group_by(name) |>
  reframe(density_area(log_F2,
                       log_F1,
                       probs = ppoints(10),
                       as_sf = TRUE)) |>
  st_sf() ->
  s01_areas_sf

if(ggplot2_inst){
  s01_areas_sf |>
    arrange(desc(prob)) |>
    ggplot() +
    geom_sf(aes(fill = area))
}
```

`density_polygons` *Density polygons*

Description

Given numeric vectors `x` and `y`, `density_polygons()` will return a data frame, or list of a data frames, of the polygon defining 2d kernel densities.

Usage

```
density_polygons(
  x,
```

```
  y,
  probs = 0.5,
  as_sf = FALSE,
  as_list = FALSE,
  range_mult = 0.25,
  rangex = NULL,
  rangey = NULL,
  ...
)
```

Arguments

x, y	Numeric data dimensions
probs	Probabilities to compute density polygons for
as_sf	Should the returned values be sf::sf? Defaults to FALSE.
as_list	Should the returned value be a list? Defaults to FALSE to work with dplyr::reframe()
range_mult	A multiplier to the range of x and y across which the probability density will be estimated.
rangex, rangey	Custom ranges across x and y ranges across which the probability density will be estimated.
...	Additional arguments to be passed to ggdensity::get_hdr()

Details

When using density_polygons() together with dplyr::summarise(), as_list should be TRUE.

If both rangex and rangey are defined, range_mult will be disregarded. If only one or the other of rangex and rangey are defined, range_mult will be used to produce the range of the undefined one.

Value

A list of data frames, if as_list=TRUE, or just a data frame, if as_list=FALSE.

Data frame output:

If as_sf=FALSE, the data frame has the following columns:

level_id An integer id for each probability level

id An integer id for each sub-polygon within a probabilty level

prob The probability level (originally passed to probs)

x, y The values along the original x and y dimensions defining the density polygon. These will be renamed to the original input variable names.

order The original plotting order of the polygon points, for convenience.

sf output:

If as_sf=TRUE, the data frame has the following columns:

level_id An integer id for each probability level

prob The probability level (originally passed to probs)

geometry A column of sf::st_polygon()s.

This output will need to be passed to sf::st_sf() to utilize many of the features of sf.

Examples

```
library(densityarea)
library(dplyr)
library(purrr)
library(sf)

ggplot2_inst <- require(ggplot2)
tidyr_inst <- require(tidyr)

set.seed(10)
x <- c(rnorm(100))
y <- c(rnorm(100))

# ordinary data frame output
poly_df <- density_polygons(x,
                            y,
                            probs = ppoints(5))

head(poly_df)

# It's necessary to specify a grouping factor that combines `level_id` and `id`
# for cases of multimodal density distributions
if(ggplot2_inst){
  ggplot(poly_df, aes(x, y)) +
    geom_path(aes(group = paste0(level_id, id),
                  color = prob))
}

# sf output
poly_sf <- density_polygons(x,
                            y,
                            probs = ppoints(5),
                            as_sf = TRUE)

head(poly_sf)

# `geom_sf()` is from the `{sf}` package.
if(ggplot2_inst){
  poly_sf |>
    arrange(desc(prob)) |>
    ggplot() +
    geom_sf(aes(fill = prob))
}

# Tidyverse usage

data(s01)

# Data transformation
s01 <- s01 |>
  mutate(log_F1 = -log(F1),
         log_F2 = -log(F2))
```

```
## Basic usage with `dplyr::reframe()`
### Data frame output
s01 |>
  group_by(name) |>
  reframe(density_polygons(log_F2,
                           log_F1,
                           probs = ppoints(5))) ->
  speaker_poly_df

if(ggplot2_inst){
  speaker_poly_df |>
    ggplot(aes(log_F2, log_F1)) +
    geom_path(aes(group = paste0(level_id, id),
                  color = prob)) +
    coord_fixed()
}

### sf output
s01 |>
  group_by(name) |>
  reframe(density_polygons(log_F2,
                           log_F1,
                           probs = ppoints(5),
                           as_sf = TRUE)) |>
  st_sf() ->
  speaker_poly_sf

if(ggplot2_inst){
  speaker_poly_sf |>
    ggplot() +
    geom_sf(aes(color = prob),
            fill = NA)
}

## basic usage with dplyr::summarise()
### data frame output

if(tidyr_inst){
  s01 |>
    group_by(name) |>
    summarise(poly = density_polygons(log_F2,
                                      log_F1,
                                      probs = ppoints(5),
                                      as_list = TRUE)) |>
    unnest(poly) ->
    speaker_poly_df
}
### sf output

if(tidyr_inst){
  s01 |>
    group_by(name) |>
```

```
    summarise(poly = density_polygons(
      log_F2,
      log_F1,
      probs = ppoints(5),
      as_list = TRUE,
      as_sf = TRUE
    )) |>
    unnest(poly) |>
    st_sf() ->
    speaker_poly_sf
}
```

---

s01 *Vowel Space Data*

Description

This is the vowel space data from a single speaker, s01, whose audio interview and transcription are part of the Buckeye Corpus (Pitt et al. 2007). The transcript was realigned to the audio using the Montreal Forced Aligner (McAullife et al. 2022) and vowel formant data extracted with FAVE (Rosenfelder et al. 2022).

Usage

```
s01
```

Format

`s01`**:**

A dataframe with 4,245 rows and 10 columns

name Speaker id

age Speaker age (y=young, o=old)

sex Speaker sex

word Word in the transcription

vowel Arpabet transcription of the measured vowel

plt_vclass A modified Labov/Trager notation of the measured vowel

ip_vclass An IPA-like transcription of the measured vowel

F1 The measured F1 frequency (Hz)

F2 The measured F2 frequency (Hz)

dur The measured vowel duration

Source

McAuliffe, M., Fatchurrahman, M. R., GalaxieT, NTT123, Amogh Gulati, Coles, A., Veaux, C., Eren, E., Mishra, H., Paweł Potrykus, Jung, S., Sereda, T., Mestrou, T., Michaelasocolof, & Vannawillerton. (2022). MontrealCorpusTools/Montreal-Forced-Aligner: Version 2.0.1 (v2.0.1) doi:10.5281/ZENODO.6658586

Pitt, M. A., Dilley, L., Johnson, K., Kiesling, S., Raymond, W., Hume, E., & Fosler-Lussier, E. (2007). Buckeye Corpus of Conversational Speech (2nd release). Department of Psychology, Ohio State University. https://buckeyecorpus.osu.edu/

Rosenfelder, I., Fruehwald, J., Brickhouse, C., Evanini, K., Seyfarth, S., Gorman, K., Prichard, H., & Yuan, J. (2022). FAVE (Forced Alignment and Vowel Extraction) Program Suite v2.0.0 https://github.com/JoFrhwld/FAVE

Index