Package ‘densityarea’

October 2, 2023

Type Package

Title Polygons of Bivariate Density Distributions

Version 0.1.0

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

RoxygenNote 7.2.3

NeedsCompilation no

Author Josef Fruehwald [aut, cre, cph]

Maintainer Josef Fruehwald <jofrhwld@gmail.com>

Repository CRAN

Date/Publication 2023-10-02 10:20:06 UTC
**R topics documented:**

- `density_area` ......................................................... 2
- `density_polygons` .................................................. 4
- `s01` ................................................................. 8

---

**Index**

<table>
<thead>
<tr>
<th>density_area</th>
<th>Density Area</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

```r
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
```

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,
density_polygons

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 probabiltiy 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)
tidyris_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
```r
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
```r
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
```r
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
```r
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

```r
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


Index

* datasets
  s01, 8

  density_area, 2
  density_polygons, 4
  dplyr::reframe(), 5
  dplyr::summarise(), 5

  ggdensity::get_hdr(), 2, 5

  s01, 8
  sf, 5
  sf::sf, 2, 5
  sf::st_polygon(), 5
  sf::st_sf(), 5