

Package ‘sfhotspot’

October 1, 2026

Title Hot-Spot Analysis with Simple Features

Version 1.1.1

Description Identify and understand clusters of points (typically representing the locations of places or events) stored in simple-features (SF) objects. This is useful for analysing, for example, hot-spots of crime events. The package emphasises producing results from point SF data in a single step using reasonable default values for all other arguments, to aid rapid data analysis by users who are starting out. Functions available include kernel density estimation (for details, see Yip (2020) <[doi:10.22224/gistbok/2020.1.12](https://doi.org/10.22224/gistbok/2020.1.12)>), analysis of spatial association (Getis and Ord (1992) <[doi:10.1111/j.1538-4632.1992.tb00261.x](https://doi.org/10.1111/j.1538-4632.1992.tb00261.x)>) and hot-spot classification (Chainey (2020) ISBN:158948584X).

License MIT + file LICENSE

Language en-GB

URL <https://pkgs.lesscrime.info/sfhotspot/>

BugReports <https://github.com/mpjashby/sfhotspot/issues>

Encoding UTF-8

Imports classInt, cli, dbscan, ggplot2, ggspatial, isoband, rlang, sf, SpatialKDE, spdep, tibble

Depends R (>= 4.1.0)

Suggests testthat (>= 3.0.0), knitr, lubridate, rmarkdown, quarto

LazyData true

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Matt Ashby [aut, cre] (ORCID: <<https://orcid.org/0000-0003-4201-9239>>)

Maintainer Matt Ashby <matthew.ashby@ucl.ac.uk>

Repository CRAN

Date/Publication 2026-10-01 00:40:02 UTC

Contents

autoplot.hspt_c	2
autoplot.hspt_d	4
autoplot.hspt_dk	5
autoplot.hspt_g	7
autoplot.hspt_ib	8
autoplot.hspt_k	10
autoplot.hspt_n	11
autoplot.hspt_s	13
hotspot_change	15
hotspot_classify	17
hotspot_classify_params	19
hotspot_clip	20
hotspot_count	21
hotspot_dbscan	22
hotspot_dual_kde	24
hotspot_gistar	27
hotspot_grid	30
hotspot_isoband	31
hotspot_kde	33
hotspot_layer	35
hotspot_map	36
memphis_population	37
memphis_precincts	38
memphis_robberies	38
memphis_robberies_jan	39
st_transform_auto	39
Index	41

autoplot.hspt_c	<i>Plot map of hotspot classifications</i>
-----------------	--

Description

Plot the output produced by `hotspot_classify()` with reasonable defaults.

Usage

```
## S3 method for class 'hspt_c'
autoplot(
  object,
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
```

```

)

## S3 method for class 'hspt_c'
autolayer(object, ...)

## S3 method for class 'hspt_c'
hotspot_map(
  object,
  mapping = NULL,
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)

```

Arguments

object	An object with class <code>hspt_c</code> , e.g. as produced by <code>hotspot_classify()</code> .
...	Further arguments passed to <code>ggplot2::geom_sf()</code> , e.g. <code>alpha</code> .
basemap_type	A map type passed to the type argument of <code>ggspatial::annotation_map_tile()</code> , or "none" for no base map. <code>autoplot()</code> defaults to "none" and <code>hotspot_map()</code> defaults to "osm". A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any alpha value supplied in ...
basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for <code>basemap_type</code> . A non-empty string is required for a custom or otherwise unknown map type.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be NULL. Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or NULL for no additional text. Base-map attribution and any explanatory caption generated by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

`autoplot()` and `hotspot_map()` return `ggplot2::ggplot` objects. `autolayer()` returns a layer that can be added to a `ggplot2::ggplot` object.

Functions

- `autolayer(hspt_c)`: Create a ggplot layer of hotspot classifications.
- `hotspot_map(hspt_c)`: Create an automatic map of hotspot classifications.

<code>autoplot.hspt_d</code>	<i>Plot map of changes in grid counts</i>
------------------------------	---

Description

Plot the output produced by [hotspot_change\(\)](#) with reasonable defaults.

Usage

```
## S3 method for class 'hspt_d'
autoplot(
  object,
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
)
```

```
## S3 method for class 'hspt_d'
autolayer(object, ...)
```

```
## S3 method for class 'hspt_d'
hotspot_map(
  object,
  mapping = NULL,
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)
```

Arguments

<code>object</code>	An object with class <code>hspt_d</code> , e.g. as produced by hotspot_change() .
<code>...</code>	Further arguments passed to ggplot2::geom_sf() , e.g. <code>alpha</code> .
<code>basemap_type</code>	A map type passed to the <code>type</code> argument of ggspatial::annotation_map_tile() , or "none" for no base map. <code>autoplot()</code> defaults to "none" and <code>hotspot_map()</code> defaults to "osm". A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any <code>alpha</code> value supplied in <code>...</code>

basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for basemap_type. A non-empty string is required for a custom or otherwise unknown map type.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be NULL. Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or NULL for no additional text. Base-map attribution and any explanatory caption generated by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

`autoplot()` and `hotspot_map()` return `ggplot2::ggplot` objects. `autolayer()` returns a layer that can be added to a `ggplot2::ggplot` object.

Functions

- `autolayer(hspt_d)`: Create a `ggplot` layer of change in grid counts.
- `hotspot_map(hspt_d)`: Create an automatic map of changes in grid counts.

<code>autoplot.hspt_dk</code>	<i>Plot map of dual kernel-density values</i>
-------------------------------	---

Description

Plot the output produced by `hotspot_dual_kde()` using a scale appropriate to the comparison method. Ratios, logged ratios and sums use sequential scales. Only differences use a diverging scale centred on zero. Legends label the lower and upper ends of each scale as "low" and "high", respectively.

Usage

```
## S3 method for class 'hspt_dk'
autoplot(
  object,
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
)
```

```
## S3 method for class 'hspt_dk'
autolayer(object, ...)

## S3 method for class 'hspt_dk'
hotspot_map(
  object,
  mapping = NULL,
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)
```

Arguments

object	An object with class <code>hspt_dk</code> , e.g. as produced by <code>hotspot_dual_kde()</code> . The object must have a valid method attribute.
...	Further arguments passed to <code>ggplot2::geom_sf()</code> , e.g. <code>alpha</code> .
basemap_type	A map type passed to the <code>type</code> argument of <code>ggspatial::annotation_map_tile()</code> , or <code>"none"</code> for no base map. <code>autoplot()</code> defaults to <code>"none"</code> and <code>hotspot_map()</code> defaults to <code>"osm"</code> . A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any <code>alpha</code> value supplied in ...
basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or <code>NULL</code> to choose it automatically.
basemap_attribution	Attribution for the tile provider, or <code>NULL</code> to use the statement known for <code>basemap_type</code> . A non-empty string is required for a custom or otherwise unknown map type.
quiet	If <code>TRUE</code> (the default), suppress progress bars and other messages. If <code>FALSE</code> , show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be <code>NULL</code> . Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or <code>NULL</code> for no additional text. Base-map attribution and any explanatory caption generated by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

`autoplot()` and `hotspot_map()` return `ggplot2::ggplot` objects. `autolayer()` returns a layer that can be added to a `ggplot2::ggplot` object.

Functions

- `autolayer(hspt_dk)`: Create a `ggplot` layer of dual density values.

- `hotspot_map(hspt_dk)`: Create an automatic map of dual density values.

<code>autoplot.hspt_g</code>	<i>Plot map of Getis-Ord G_i^* results</i>
------------------------------	---

Description

If object contains a kde column, density is shown only in cells in which the G_i^*/G_i result passes the specified significance and sign conditions. The pvalue column is used as supplied and is not adjusted by the plotting methods. Cells that do not satisfy the conditions are transparent. When `sign = "both"`, cold-spot densities are negated for plotting and a diverging scale distinguishes cold spots from hot spots. When only one sign is shown, a medium-to-dark sequential scale avoids making the least-dense significant cells appear nearly white. If object does not contain a kde column, the G_i^*/G_i value is plotted using a diverging scale centred on zero and `critical_p` and `sign` do not affect the mapped values.

Usage

```
## S3 method for class 'hspt_g'
autoplot(
  object,
  critical_p = 0.05,
  sign = c("both", "hot", "cold"),
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
)

## S3 method for class 'hspt_g'
autolayer(object, critical_p = 0.05, sign = c("both", "hot", "cold"), ...)

## S3 method for class 'hspt_g'
hotspot_map(
  object,
  mapping = NULL,
  critical_p = 0.05,
  sign = c("both", "hot", "cold"),
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)
```

Arguments

object	An object with class <code>hspt_g</code> , e.g. as produced by <code>hotspot_gistar()</code> .
critical_p	A single numeric value specifying the largest p-value to treat as statistically significant when plotting density.
sign	Which significant results should show density: "both" (the default), "hot" for positive G_i^*/G_i values, or "cold" for negative values.
...	Further arguments passed to <code>ggplot2::geom_sf()</code> , e.g. <code>alpha</code> .
basemap_type	A map type passed to the type argument of <code>ggspatial::annotation_map_tile()</code> , or "none" for no base map. <code>autoplot()</code> defaults to "none" and <code>hotspot_map()</code> defaults to "osm". A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any <code>alpha</code> value supplied in ...
basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for <code>basemap_type</code> . A non-empty string is required for a custom or otherwise unknown map type.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be NULL. Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or NULL for no additional text. Base-map attribution and any explanatory caption generated by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

`autoplot()` and `hotspot_map()` return `ggplot2::ggplot` objects. `autolayer()` returns a layer that can be added to a `ggplot2::ggplot` object.

Functions

- `autolayer(hspt_g)`: Create a `ggplot` layer of Getis-Ord G_i^* results.
- `hotspot_map(hspt_g)`: Create an automatic map of Getis-Ord G_i^* results.

 autoplot.hspt_ib

Plot isobands

Description

Plot the output produced by `hotspot_isoband()` using a sequential or diverging discrete scale appropriate to the original hotspot result and selected value. Legend entries use the ordered `label` column containing concise, automatically formatted ranges.

Usage

```
## S3 method for class 'hspt_ib'
autoplot(
  object,
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
)

## S3 method for class 'hspt_ib'
autolayer(object, ...)

## S3 method for class 'hspt_ib'
hotspot_map(
  object,
  mapping = NULL,
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)
```

Arguments

object	An object with class <code>hspt_ib</code> , as produced by <code>hotspot_isoband()</code> .
...	Further arguments passed to <code>ggplot2::geom_sf()</code> , e.g. <code>alpha</code> .
basemap_type	A map type passed to the type argument of <code>ggspatial::annotation_map_tile()</code> , or "none" for no base map. <code>autoplot()</code> defaults to "none" and <code>hotspot_map()</code> defaults to "osm". A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any alpha value supplied in
basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for <code>basemap_type</code> . A non-empty string is required for a custom or otherwise unknown map type.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be NULL. Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or NULL for no additional text. Base-map attribution and any explanatory caption generated

by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

autoplot() and hotspot_map() return [ggplot2::ggplot](#) objects. autolayer() returns a layer that can be added to a [ggplot2::ggplot](#) object.

Functions

- autolayer(hspt_ib): Create a ggplot layer of isobands.
- hotspot_map(hspt_ib): Create an automatic map of isobands.

autoplot.hspt_k	<i>Plot map of kernel-density values</i>
-----------------	--

Description

Plot the output produced by [hotspot_kde\(\)](#) with reasonable default values.

Usage

```
## S3 method for class 'hspt_k'
autoplot(
  object,
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
)

## S3 method for class 'hspt_k'
autolayer(object, ...)

## S3 method for class 'hspt_k'
hotspot_map(
  object,
  mapping = NULL,
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)
```

Arguments

object	An object with class <code>hspt_k</code> , e.g. as produced by <code>hotspot_kde()</code> .
...	Further arguments passed to <code>ggplot2::geom_sf()</code> , e.g. <code>alpha</code> .
basemap_type	A map type passed to the <code>type</code> argument of <code>ggspatial::annotation_map_tile()</code> , or "none" for no base map. <code>autoplot()</code> defaults to "none" and <code>hotspot_map()</code> defaults to "osm". A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any <code>alpha</code> value supplied in ...
basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for <code>basemap_type</code> . A non-empty string is required for a custom or otherwise unknown map type.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be NULL. Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or NULL for no additional text. Base-map attribution and any explanatory caption generated by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

`autoplot()` and `hotspot_map()` return `ggplot2::ggplot` objects. `autolayer()` returns a layer that can be added to a `ggplot2::ggplot` object.

Functions

- `autolayer(hspt_k)`: Create a `ggplot` layer of kernel-density values.
- `hotspot_map(hspt_k)`: Create an automatic map of kernel-density values.

<code>autoplot.hspt_n</code>	<i>Plot map of grid counts</i>
------------------------------	--------------------------------

Description

Plot the output produced by `hotspot_count()` with reasonable default values. Weighted counts are plotted when the object contains a `sum` column; otherwise unweighted counts in the `n` column are plotted.

Usage

```
## S3 method for class 'hspt_n'
autoplot(
  object,
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
)

## S3 method for class 'hspt_n'
autolayer(object, ...)

## S3 method for class 'hspt_n'
hotspot_map(
  object,
  mapping = NULL,
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)
```

Arguments

object	An object with class <code>hspt_n</code> , e.g. as produced by <code>hotspot_count()</code> .
...	Further arguments passed to <code>ggplot2::geom_sf()</code> , e.g. <code>alpha</code> .
basemap_type	A map type passed to the type argument of <code>ggspatial::annotation_map_tile()</code> , or "none" for no base map. <code>autoplot()</code> defaults to "none" and <code>hotspot_map()</code> defaults to "osm". A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any alpha value supplied in
basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for <code>basemap_type</code> . A non-empty string is required for a custom or otherwise unknown map type.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be NULL. Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or NULL for no additional text. Base-map attribution and any explanatory caption generated

by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

autoplot() and hotspot_map() return [ggplot2::ggplot](#) objects. autolayer() returns a layer that can be added to a [ggplot2::ggplot](#) object.

Functions

- autolayer(hspt_n): Create a ggplot layer of grid counts.
- hotspot_map(hspt_n): Create an automatic map of grid counts.

autoplot.hspt_s	<i>Plot DBSCAN hotspot clusters</i>
-----------------	-------------------------------------

Description

Plot the polygon clusters produced by [hotspot_dbscan\(\)](#) with reasonable defaults. Polygons can be filled according to their count, proportion or rank, and optionally labelled with the same values.

Usage

```
## S3 method for class 'hspt_s'
autoplot(
  object,
  col_fill = c("n", "prop", "rank", "none"),
  col_label = "none",
  ...,
  basemap_type = "none",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  quiet = TRUE
)

## S3 method for class 'hspt_s'
autolayer(
  object,
  col_fill = c("n", "prop", "rank", "none"),
  col_label = "none",
  ...
)

## S3 method for class 'hspt_s'
hotspot_map(
  object,
  mapping = NULL,
```

```

col_fill = c("n", "prop", "rank", "none"),
col_label = "none",
...,
basemap_type = "osm",
basemap_zoom = NULL,
basemap_attribution = NULL,
caption = NULL,
quiet = TRUE
)

```

Arguments

object	An object with class <code>hspt_s</code> , as produced by <code>hotspot_dbscan()</code> .
col_fill	A single string specifying the column used for the fill aesthetic: "n" (the default), "prop" or "rank". Use "none" to draw unfilled polygons with the default <code>ggplot2</code> border colour. Set colour in <code>...</code> to use a different border colour.
col_label	One or more strings specifying the columns used for labels shown in each cluster: "none" (the default), or any combination of "n", "prop" and "rank". Multiple labels are separated by newlines. Proportions are formatted as percentages and ranks as ordinal numbers. Labels use a semi-transparent background chosen to contrast with the base map.
...	Static aesthetics and other arguments passed to <code>ggplot2::geom_sf()</code> , e.g. colour, fill or alpha.
basemap_type	A map type passed to the type argument of <code>ggspatial::annotation_map_tile()</code> , or "none" for no base map. <code>autoplot()</code> defaults to "none" and <code>hotspot_map()</code> defaults to "osm". A base map requires an internet connection unless the required tiles are cached. When a base map is used, the hotspot layer is drawn with <code>alpha = 0.75</code> , overriding any alpha value supplied in <code>...</code> .
basemap_zoom	The zoom level passed to <code>ggspatial::annotation_map_tile()</code> , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for <code>basemap_type</code> . A non-empty string is required for a custom or otherwise unknown map type.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.
mapping	For <code>hotspot_map()</code> , this must be NULL. Hotspot-result objects determine their own aesthetic mappings.
caption	For <code>hotspot_map()</code> , additional text to display in the map caption, or NULL for no additional text. Base-map attribution and any explanatory caption generated by the plotting method are retained, with explanatory text above this caption and attribution below it.

Value

`autoplot()` and `hotspot_map()` return `ggplot2::ggplot` objects. `autolayer()` returns one or more layers that can be added to a `ggplot2::ggplot` object.

Functions

- `autolayer(hspt_s)`: Create ggplot layers for DBSCAN hotspot clusters.
- `hotspot_map(hspt_s)`: Create an automatic map of DBSCAN clusters.

hotspot_change	<i>Identify change in hotspots over time</i>
----------------	--

Description

Identify change in the number of points (typically representing events) between two periods (before and after a specified date) or in two groups (e.g. on weekdays or at weekends).

Usage

```
hotspot_change(
  data,
  time = NULL,
  boundary = NULL,
  groups = NULL,
  cell_size = NULL,
  grid_type = "rect",
  grid = NULL,
  quiet = FALSE
)
```

Arguments

<code>data</code>	sf data frame containing points.
<code>time</code>	Name of the column in data containing Date or POSIXt values representing the date associated with each point. Ignored if groups is not NULL. If this argument is NULL and data contains a single column of Date or POSIXt values, that column will be used automatically.
<code>boundary</code>	A single Date or POSIXt value representing the point after which points should be treated as having occurred in the second time period. See 'Details'.
<code>groups</code>	Name of a column in data containing exactly two unique non-missing values, which will be used to identify whether each row should be counted in the first (before) or second (after) groups. Which groups to use will be determined by calling <code>sort(unique(groups))</code> . If groups is not a factor, a message will be printed confirming which value has been used for which group. See 'Details'.
<code>cell_size</code>	numeric value specifying the size of each equally spaced grid cell, using the same units (metres, degrees, etc.) as used in the sf data frame given in the data argument. Ignored if grid is not NULL. If this argument and grid are NULL (the default), the cell size will be calculated automatically (see Details).
<code>grid_type</code>	character specifying whether the grid should be made up of squares ("rect", the default) or hexagons ("hex"). Ignored if grid is not NULL.

grid	sf data frame containing points containing polygons, which will be used as the grid for which counts are made.
quiet	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.

Details

This function creates a regular two-dimensional grid of cells (unless a custom grid is specified with `grid`) and calculates the difference between the number of points in each grid cell:

- before and after a set point in time, if boundary is specified,
- between two groups of points, if a column of grouping values is specified with `groups`,
- before and after the mid-point of the dates/times present in the data, if both boundary and groups are NULL (the default).

If both boundary and groups are not NULL, the value of boundary will be ignored.

Coverage of the output data:

The grid produced by this function covers the convex hull of the input data layer. This means the result may include zero counts for cells that are outside the area for which data were provided, which could be misleading. To handle this, consider cropping the output layer to the area for which data are available. For example, if you only have crime data for a particular district, crop the output dataset to the district boundary using [st_intersection](#).

Automatic cell-size selection:

If no cell size is given then the cell size will be set so that there are 50 cells on the shorter side of the grid. If the data SF object is projected in metres or feet, the number of cells will be adjusted upwards so that the cell size is a multiple of 100.

Value

An [sf](#) tibble of regular grid cells with corresponding hot-spot classifications for each cell. This can be plotted using [autoplot](#).

See Also

[hotspot_dual_kde\(\)](#) for comparing the density of two layers, which will often be more useful than comparing counts if the point locations represent and underlying continuous distribution.

Examples

```
# Compare counts from the first half of the period covered by the data to
# counts from the second half
```

```
hotspot_change(memphis_robberies)
```

```
# Create a grouping variable, then compare counts across values of that
# variable
```

```

memphis_robberies$weekend <-
  weekdays(memphis_robberies$date) %in% c("Saturday", "Sunday")
hotspot_change(memphis_robberies, groups = weekend)

```

hotspot_classify	<i>Classify hot-spots</i>
------------------	---------------------------

Description

Classify cells in a grid based on changes in the clustering of points (typically representing events) in a two-dimensional regular grid over time.

Usage

```

hotspot_classify(
  data,
  time = NULL,
  period = NULL,
  start = NULL,
  cell_size = NULL,
  grid_type = "rect",
  grid = NULL,
  collapse = FALSE,
  params = hotspot_classify_params(),
  quiet = FALSE
)

```

Arguments

<code>data</code>	sf data frame containing points.
<code>time</code>	Name of the column in data containing Date or POSIXt values representing the date associated with each point. If this argument is NULL and data contains a single column of Date or POSIXt values, that column will be used automatically.
<code>period</code>	A character value containing a number followed by a unit of time, e.g. for example, "12 months" or "3.5 days", where the unit of time is one of second, minute, hour, day, week, month, quarter or year (or their plural forms).
<code>start</code>	A Date or POSIXt value specifying when the first temporal period should start. If NULL (the default), the first period will start at the beginning of the earliest date found in the data (if period is specified in days, weeks, months, quarters or years) or at the earliest time found in the data otherwise.
<code>cell_size</code>	numeric value specifying the size of each equally spaced grid cell, using the same units (metres, degrees, etc.) as used in the sf data frame given in the data argument. Ignored if grid is not NULL. If this argument and grid are NULL (the default), the cell size will be calculated automatically (see Details).

grid_type	character specifying whether the grid should be made up of squares ("rect", the default) or hexagons ("hex"). Ignored if grid is not NULL.
grid	sf data frame containing points containing polygons, which will be used as the grid for which counts are made.
collapse	If the range of dates in the data is not a multiple of period, the final period will be shorter than the others. In that case, should this shorter period be collapsed into the penultimate period?
params	A list of optional parameters that can affect the output. The list can be produced most easily using the hotspot_classify_params helper function.
quiet	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.

Value

An [sf](#) tibble of regular grid cells with corresponding hot-spot classifications for each cell. This can be plotted using [autoplot](#).

Hot-spots are spatial areas that contain more points than would be expected by chance; cold-spots are areas that contain fewer points than would be expected. Whether an area is a hot-spot can vary over time. This function creates a space-time cube, determines whether an area is a hot-spot for each of several consecutive time periods and uses that to classify areas according to whether they are persistent, intermittent, emerging or former hot- or cold-spots.

Hot and cold spots:

Hot- and cold-spots are identified by calculating the Getis-Ord G_i^* (gi-star) or G_i Z -score statistic for each cell in a regular grid for each time period. The corresponding p -values are adjusted once within each period using [p.adjustSP](#) and the method specified by `p_adjust_method` in `params`. They are not adjusted again across time periods, to avoid overly-conservative results. Cells are classified as follows, using the parameters provided in the `params` argument:

- *Persistent hot-/cold-spots* are cells that have been hot-/cold-spots consistently over time. Formally: if the p -value is less than `critical_p` for at least `persistent_prop` proportion of time periods.
- *Emerging hot-/cold-spots* are cells that have become hot-/cold-spots recently but were not previously. Formally: if the p -value is less than `critical_p` for at least `hotspot_prop` of time periods defined as recent by `recent_prop` but the p -value was *not* less than `critical_p` for at least `hotspot_prop` of time periods defined as non-recent by `1 - recent_prop`.
- *Former hot-/cold-spots* are cells that used to be hot-/cold-spots but have not been more recently. Formally: if the p -value was less than `critical_p` for at least `hotspot_prop` of time periods defined as non-recent by `1 - recent_prop` but the p -value was *not* less than `critical_p` for for at least `hotspot_prop` of time periods defined as recent by `recent_prop`.
- *Intermittent hot-/cold-spots* are cells that have been hot-/cold-spots, but not as frequently as persistent hotspots and not only during recent/non-recent periods. Formally: if the p -value is less than `critical_p` for at least `hotspot_prop` of time periods but the cell is not an emerging or former hotspot.
- *No pattern* if none of the above categories apply.

Coverage of the output data:

The grid produced by this function covers the convex hull of the input data layer. This means the result may include G_i^* or G_i values for cells that are outside the area for which data were provided, which could be misleading. To handle this, consider cropping the output layer to the area for which data are available. For example, if you only have crime data for a particular district, crop the output dataset to the district boundary using [st_intersection](#).

Automatic cell-size selection:

If no cell size is given then the cell size will be set so that there are 50 cells on the shorter side of the grid. If the data SF object is projected in metres or feet, the number of cells will be adjusted upwards so that the cell size is a multiple of 100.

References

Chainey, S. (2020). *Understanding Crime: Analyzing the Geography of Crime*. Redlands, CA: ESRI.

hotspot_classify_params

Control the parameters used to classify hotspots

Description

This function allows specification of parameters that affect the output from [hotspot_classify](#).

Usage

```
hotspot_classify_params(
  hotspot_prop = 0.1,
  persistent_prop = 0.8,
  recent_prop = 0.2,
  critical_p = 0.05,
  nb_dist = NULL,
  include_self = TRUE,
  p_adjust_method = NULL
)
```

Arguments

hotspot_prop	A single numeric value specifying the minimum proportion of periods for which a cell must contain significant clusters of points before the cell can be classified as a hot or cold spot of any type.
persistent_prop	A single numeric value specifying the minimum proportion of periods for which a cell must contain significant clusters of points before the cell can be classified as a persistent hot or cold spot.
recent_prop	A single numeric value specifying the proportion of periods that should be treated as being recent in the classification of emerging and former hotspots.

critical_p	A threshold p -value below which values should be treated as being statistically significant.
nb_dist	The distance around a cell that contains the neighbours of that cell, which are used in calculating the statistic. If this argument is NULL (the default), nb_dist is set as $\text{cell_size} * \sqrt{2}$ so that only the cells immediately adjacent to each cell are treated as being its neighbours.
include_self	Should points in a given cell be counted as well as counts in neighbouring cells when calculating the values of G_i^* (if include_self = TRUE, the default) or G_i (if include_self = FALSE) values? You are unlikely to want to change the default value.
p_adjust_method	The method to be used to adjust p -values for multiple comparisons using p.adjustSP . NULL (the default) uses the default method used by p.adjust (currently "holm"), but any of the character values in <code>stats::p.adjust.methods</code> may be specified. Set this argument to "none" to return unadjusted p -values. The adjustment is applied separately to the cells in each time period.

Value

A list that can be used as the input to the `params` argument to [hotspot_classify](#).

hotspot_clip	<i>Extract spatial features inside a polygon</i>
--------------	--

Description

Extract spatial features inside a polygon

Usage

```
hotspot_clip(data, boundary, quiet = FALSE, ...)
```

Arguments

data	sf data frame containing spatial features.
boundary	sf data frame containing polygons.
quiet	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.
...	Further arguments passed to st_intersection .

Details

This function is a wrapper around [st_intersection](#) that performs some additional checks and reports useful information. If `data` has a specialised result class produced by this package, that class is preserved in the clipped result. A warning is produced if clipping reduces the dimension of any geometry, such as from a polygon to a line.

Value

an SF data frame containing those spatial features that are covered by the polygons.

hotspot_count	<i>Count points in cells in a two-dimensional grid</i>
---------------	--

Description

Count points in cells in a two-dimensional grid

Usage

```
hotspot_count(
  data,
  cell_size = NULL,
  grid_type = "rect",
  grid = NULL,
  weights = NULL,
  quiet = FALSE
)
```

Arguments

data	sf data frame containing points.
cell_size	numeric value specifying the size of each equally spaced grid cell, using the same units (metres, degrees, etc.) as used in the sf data frame given in the data argument. Ignored if grid is not NULL. If this argument and grid are NULL (the default), the cell size will be calculated automatically (see Details).
grid_type	character specifying whether the grid should be made up of squares ("rect", the default) or hexagons ("hex"). Ignored if grid is not NULL.
grid	sf data frame containing polygons, which will be used as the grid for which counts are made.
weights	NULL or the name of a column in data to be used as weights for weighted counts.
quiet	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.

Details

This function counts the number of points in each cell in a regular grid. If a column name in data is supplied with the weights argument, weighted counts will also be produced.

Automatic cell-size selection:

If grid is NULL and no cell size is given, the cell size will be set so that there are 50 cells on the shorter side of the grid. If the data SF object is projected in metres or feet, the number of cells will be adjusted upwards so that the cell size is a multiple of 100.

Value

An [sf](#) tibble of regular grid cells with corresponding point counts for each cell. This can be plotted using [autoplot](#).

Examples

```
# Set cell size automatically

hotspot_count(memphis_robberies_jan)

# Transform data to UTM zone 15N so that cell_size and bandwidth can be set
# in metres
library(sf)
memphis_robberies_utm <- st_transform(memphis_robberies_jan, 32615)

# Manually set grid-cell size in metres, since the `memphis_robberies_utm`
# dataset uses a co-ordinate reference system (UTM zone 15 north) that is
# specified in metres

hotspot_count(memphis_robberies_utm, cell_size = 200)
```

hotspot_dbscan	<i>Identify hotspots using DBSCAN</i>
----------------	---------------------------------------

Description

Identify clusters of points using density-based spatial clustering and represent each cluster as a buffered convex or concave hull. Clusters are ranked by the number of points they contain.

Usage

```
hotspot_dbscan(
  data,
  eps = NULL,
  min_pts = NULL,
  density_adjust = 2,
  hull = c("concave", "convex"),
  hull_ratio = 0.75,
  transform = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

<code>data</code>	An <code>sf::sf</code> object containing point geometries.
<code>eps</code>	A single positive number specifying the DBSCAN neighbourhood radius in the units of the analysis co-ordinate reference system (CRS). If <code>NULL</code> , the radius is calculated automatically from nearest-neighbour distances. See Automatic parameter selection .
<code>min_pts</code>	A single integer specifying the minimum number of points in an <code>eps</code> neighbourhood, including the point itself, for a point to be a core point. If <code>NULL</code> , the value is calculated automatically from the number of point coordinates in <code>data</code> . See Automatic parameter selection .
<code>density_adjust</code>	A single positive number controlling automatic selection of <code>eps</code> . Ignored unless <code>eps = NULL</code> . A value of 1 (the reference neighbourhood density) uses the median nearest-neighbour distance; the default of 2 requires approximately twice that density.
<code>hull</code>	The type of hull used to represent each cluster: "convex" or "concave" (the default).
<code>hull_ratio</code>	For concave hulls, a number from zero to one specifying the fraction convex. Zero produces a maximally concave hull and one produces a convex hull. The default is 0.75. Ignored when <code>hull = "convex"</code> .
<code>transform</code>	DBSCAN uses Euclidean distances and therefore requires a projected CRS. If <code>TRUE</code> (the default), geographic input is transformed automatically with <code>st_transform_auto()</code> before analysis and the result is transformed back afterwards. If <code>FALSE</code> , geographic input produces an error.
<code>quiet</code>	If <code>TRUE</code> , suppress informative messages about automatically selected parameters and geometry preparation.
<code>...</code>	Further arguments passed to <code>dbscan::dbscan()</code> , such as <code>borderPoints</code> and nearest-neighbour search controls. The arguments <code>x</code> , <code>eps</code> , <code>minPts</code> , and <code>weights</code> cannot be supplied through <code>...</code>

Details

MULTIPOINT geometries are cast to individual points before analysis. The output `n` column counts all input point coordinates intersecting each final cluster polygon, not only points assigned to that DBSCAN cluster. DBSCAN includes border points in clusters by default; supplying `borderPoints = FALSE` through `...` instead performs DBSCAN* and excludes border points from the points used to construct the hull.

Each cluster geometry is the selected hull of all assigned points, buffered by `eps` and clipped to the convex hull of all input points. Different cluster polygons may overlap after hull construction and buffering.

Value

An `sf` tibble with class `hspt_s` and one row per non-noise cluster. It contains `cluster`, the original DBSCAN cluster identifier; `rank`, the priority rank; `n`, the number of input point coordinates intersecting the polygon; `prop`, `n` as a proportion of all input point coordinates; `prop_area`, the polygon area as a proportion of the area covered by a hull of the complete dataset; and `geometry`.

The complete-dataset hull uses the same hull, hull_ratio, buffering and clipping rules as the cluster hulls. Since cluster polygons may overlap, the sums of prop and prop_area can exceed one. Ranking is by decreasing n, then increasing polygon area, then increasing cluster identifier.

Automatic parameter selection

When min_pts = NULL, it is calculated as

```
min(n, 50, max(5, ceiling(sqrt(n))))
```

where n is the number of point coordinates after empty geometries have been removed and MULTIPOINT geometries have been expanded. For datasets with two to four coordinates, all coordinates are required. At least two coordinates are needed for automatic selection. This rule increases the evidence required to identify a hotspot in larger datasets, while the upper limit prevents the required number of neighbours becoming excessively large.

When eps = NULL, the function calculates the distance from every point to its (min_pts - 1)th nearest other point. The median of those distances is a typical local neighbourhood radius. The value used for clustering is

```
eps = median_neighbour_distance / sqrt(density_adjust).
```

min_pts - 1 other points are used because DBSCAN counts the focal point itself. Since circular area is proportional to the square of its radius, the default density_adjust = 2 searches for the required number of points in approximately half the typical neighbourhood area, corresponding to approximately twice the typical local point density. Larger values identify denser concentrations; values below one allow less-dense concentrations.

Automatic values provide a starting point for exploratory analysis. DBSCAN results can be sensitive to both parameters, so users should consider whether the resulting neighbourhood size and minimum density are meaningful for their application.

Examples

```
# The default concave hulls require GEOS 3.11 or later.
if (utils::compareVersion(sf::sf_extSoftVersion()[["GEOS"]], "3.11.0") >= 0) {
  hotspot_dbscan(memphis_robberies_jan)
}

hotspot_dbscan(
  memphis_robberies_jan,
  density_adjust = 3,
  hull = "convex"
)
```

Description

Estimate the relationship between the kernel density of two layers of points

Usage

```
hotspot_dual_kde(
  x,
  y,
  cell_size = NULL,
  grid_type = "rect",
  bandwidth = NULL,
  bandwidth_adjust = 1,
  method = "ratio",
  grid = NULL,
  weights = NULL,
  transform = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

<code>x, y</code>	<code>sf</code> data frames containing points.
<code>cell_size</code>	numeric value specifying the size of each equally spaced grid cell, using the same units (metres, degrees, etc.) as used in the <code>sf</code> data frame given in the <code>x</code> argument. Ignored if <code>grid</code> is not <code>NULL</code> . If this argument and <code>grid</code> are <code>NULL</code> (the default), the cell size will be calculated automatically (see Details).
<code>grid_type</code>	character specifying whether the grid should be made up of squares ("rect", the default) or hexagons ("hex"). Ignored if <code>grid</code> is not <code>NULL</code> .
<code>bandwidth</code>	either a single numeric value specifying the bandwidth to be used in calculating the kernel density estimates, or a list of exactly 2 such values. If this argument is <code>NULL</code> (the default), a single bandwidth will be determined automatically using the result of <code>bandwidth.nrd</code> called on the co-ordinates of the points in <code>x</code> , then used for both <code>x</code> and <code>y</code> . If this argument is <code>list(NULL, NULL)</code> , separate bandwidths will be determined automatically for <code>x</code> and <code>y</code> based on each layer. For lon/lat data that are transformed automatically, bandwidths are specified or calculated in the units of the projected co-ordinate reference system.
<code>bandwidth_adjust</code>	single positive numeric value by which the value of bandwidth for both <code>x</code> and <code>y</code> will be multiplied, or a list of two such values. Useful for setting the bandwidth relative to the default.
<code>method</code>	character specifying the method by which the densities, <code>d()</code> , of <code>x</code> and <code>y</code> will be related: <code>ratio</code> (the default) calculates the density of <code>x</code> divided by the density of <code>y</code> , i.e. $d(x) / d(y)$. <code>log</code> calculates the natural logarithm of the density of <code>x</code> divided by the density of <code>y</code> , i.e. $\log(d(x) / d(y))$.

	<p><code>diff</code> calculates the difference between the density of <code>x</code> and the density of <code>y</code>, i.e. $d(x) - d(y)$.</p> <p><code>sum</code> calculates the sum of the density of <code>x</code> and the density of <code>y</code>, i.e. $d(x) + d(y)$. The result of this calculation will be returned in the <code>kde</code> column of the return value.</p>
<code>grid</code>	<code>sf</code> data frame containing polygons, which will be used as the grid for which densities are estimated. Both <code>x</code> and <code>y</code> must overlap the grid and use the same co-ordinate reference system.
<code>weights</code>	NULL (the default) or a vector of length two giving either NULL or the name of a column in each of <code>x</code> and <code>y</code> to be used as weights for weighted counts and KDE values.
<code>transform</code>	the underlying SpatialKDE package cannot calculate kernel density for lon/lat data, so this must be transformed to use a projected co-ordinate reference system. If this argument is TRUE (the default) and <code>sf::st_is_longlat(x)</code> is TRUE, <code>x</code> , <code>y</code> and <code>grid</code> (if provided) will be transformed to a common projected co-ordinate reference system chosen using <code>st_transform_auto</code> before bandwidths and kernel densities are estimated. The result is returned using the original co-ordinate reference system. Set this argument to FALSE to suppress automatic transformation of the data.
<code>quiet</code>	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.
<code>...</code>	Further arguments passed to <code>kde</code> .

Value

An `sf` tibble of grid cells with corresponding point counts and dual kernel density estimates for each cell. The result has classes `hspt_dk` and `hspt_k`, and a method attribute containing the comparison method used.

The output can be plotted with reasonable defaults using `autoplot()`. The colour scale is selected according to the comparison method.

This function creates a regular two-dimensional grid of cells (unless a custom grid is specified with `grid`), calculates the density of points in each cell for each of `x` and `y` using functions from the SpatialKDE package, then produces a value representing a relation between the two densities. The count of points in each cell is also returned.

Dual kernel density values can be useful for understanding the relationship between the distributions of two sets of point locations. For example:

- The ratio between two densities representing the locations of burglaries and the locations of houses can show the distribution of the risk (incidence rate) of burglaries. The logged ratio may be useful to show relationships where one set of points has an extremely skewed distribution.
- The difference between two densities can show the change in distributions between two points in time.
- The sum of two densities can be used to estimate the total density of two types of point, e.g. the locations of occurrences of two diseases.

Coverage of the output data:

The grid produced by this function covers the convex hull of the points in `x`. This means the result may include KDE values for cells that are outside the area for which data were provided, which could be misleading. To handle this, consider cropping the output layer to the area for which data are available. For example, if you only have crime data for a particular district, crop the output dataset to the district boundary using [st_intersection](#).

Automatic cell-size selection:

If no cell size is given then the cell size will be set so that there are 50 cells on the shorter side of the grid. If the `x` SF object is projected in metres or feet, the number of cells will be adjusted upwards so that the cell size is a multiple of 100.

References

Yin, P. (2020). Kernels and Density Estimation. *The Geographic Information Science & Technology Body of Knowledge* (1st Quarter 2020 Edition), John P. Wilson (ed.). doi:[doi:10.22224/gistbok/2020.1.12](https://doi.org/10.22224/gistbok/2020.1.12)

Examples

```
# See also the examples for `hotspot_kde()` for examples of how to specify
# `cell_size`, `bandwidth`, etc.

library(sf)

# Transform data to UTM zone 15N so that cell_size and bandwidth can be set
# in metres
memphis_robberies_utm <- st_transform(memphis_robberies, 32615)
memphis_population_utm <- st_transform(memphis_population, 32615)

# Calculate burglary risk based on residential population. `weights` is set
# to `c(NULL, population)` so that the robberies layer is not weighted and
# the population layer is weighted according to the number of residents in
# each census block.

hotspot_dual_kde(
  memphis_robberies_utm,
  memphis_population_utm,
  bandwidth = list(NULL, NULL),
  weights = c(NULL, population)
)
```

Description

Identify hotspot and coldspot locations, that is cells in a regular grid in which there are more/fewer points than would be expected if the points were distributed randomly.

Usage

```
hotspot_gistar(
  data,
  cell_size = NULL,
  grid_type = "rect",
  kde = TRUE,
  bandwidth = NULL,
  bandwidth_adjust = 1,
  grid = NULL,
  weights = NULL,
  nb_dist = NULL,
  include_self = TRUE,
  p_adjust_method = NULL,
  transform = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

<code>data</code>	sf data frame containing points.
<code>cell_size</code>	numeric value specifying the size of each equally spaced grid cell, using the same units (metres, degrees, etc.) as used in the <code>sf</code> data frame given in the <code>data</code> argument. Ignored if <code>grid</code> is not <code>NULL</code> . If this argument and <code>grid</code> are <code>NULL</code> (the default), the cell size will be calculated automatically (see Details).
<code>grid_type</code>	character specifying whether the grid should be made up of squares ("rect", the default) or hexagons ("hex"). Ignored if <code>grid</code> is not <code>NULL</code> .
<code>kde</code>	<code>TRUE</code> (the default) or <code>FALSE</code> indicating whether kernel density estimates (KDE) should be produced for each grid cell.
<code>bandwidth</code>	numeric value specifying the bandwidth to be used in calculating the kernel density estimates. If this argument is <code>NULL</code> (the default), the bandwidth will be specified automatically using the mean result of <code>bandwidth.nrd</code> called on the <code>x</code> and <code>y</code> co-ordinates separately.
<code>bandwidth_adjust</code>	single positive numeric value by which the value of <code>bandwidth</code> is multiplied. Useful for setting the bandwidth relative to the default.
<code>grid</code>	sf data frame containing polygons, which will be used as the grid for which counts are made.
<code>weights</code>	<code>NULL</code> or the name of a column in <code>data</code> to be used as weights for weighted counts, G_i^* / G_i statistics and KDE values.

<code>nb_dist</code>	The distance around a cell that contains the neighbours of that cell, which are used in calculating the statistic. If this argument is NULL (the default), <code>nb_dist</code> is set as <code>cell_size * sqrt(2)</code> so that only the cells immediately adjacent to each cell are treated as being its neighbours.
<code>include_self</code>	Should points in a given cell be counted as well as counts in neighbouring cells when calculating the values of G_i^* (if <code>include_self = TRUE</code> , the default) or G_i (if <code>include_self = FALSE</code>) values? You are unlikely to want to change the default value.
<code>p_adjust_method</code>	The method to be used to adjust p -values for multiple comparisons using <code>p.adjustSP</code> . NULL (the default) uses the default method used by <code>p.adjust</code> (currently "holm"), but any of the character values in <code>stats::p.adjust.methods</code> may be specified. Set this argument to "none" to return unadjusted p -values.
<code>transform</code>	the underlying SpatialKDE package cannot calculate kernel density for lon/lat data, so this must be transformed to use a projected co-ordinate reference system. If this argument is TRUE (the default) and <code>sf::st_is_longlat(data)</code> is TRUE, data (and grid if provided) will be transformed automatically using <code>link{st_transform_auto}</code> before the kernel density is estimated and transformed back afterwards. Set this argument to FALSE to suppress automatic transformation of the data.
<code>quiet</code>	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.
<code>...</code>	Further arguments passed to <code>kde</code> or ignored if <code>kde = FALSE</code> .

Details

This function calculates the Getis-Ord G_i^* (gi-star) or G_i Z -score statistic for identifying clusters of point locations. The underlying implementation uses the `localG` function to calculate the Z scores and then `p.adjustSP` function to adjust the corresponding p -values for multiple comparison. The function also returns counts of points in each cell and (by default but optionally) kernel density estimates using the `kde` function. If `weights` is supplied, the G_i^*/G_i statistics are calculated from the weighted counts; otherwise, they are calculated from the unweighted counts.

Coverage of the output data:

The grid produced by this function covers the convex hull of the input data layer. This means the result may include G_i^* or G_i values for cells that are outside the area for which data were provided, which could be misleading. To handle this, consider cropping the output layer to the area for which data are available. For example, if you only have crime data for a particular district, crop the output dataset to the district boundary using `st_intersection`.

Automatic cell-size selection:

If no cell size is given then the cell size will be set so that there are 50 cells on the shorter side of the grid. If the data SF object is projected in metres or feet, the number of cells will be adjusted upwards so that the cell size is a multiple of 100.

Value

An `sf` tibble with class `hspt_g` containing regular grid cells with corresponding point counts, G_i or G_i^* values and (optionally) kernel density estimates for each cell. Values greater than zero indi-

cate more points than would be expected for randomly distributed points and values less than zero indicate fewer points. Critical values of G_i and G_i^* are given in the manual page for [localG](#).

The output from this function can be plotted with reasonable defaults using [autoplot\(\)](#). When KDE values are returned, the default map shows density only in areas identified as having significantly more or fewer points than expected by chance. When `kde = FALSE`, the G_i^*/G_i statistic is plotted on a diverging scale centred on zero.

References

Getis, A. & Ord, J. K. (1992). The Analysis of Spatial Association by Use of Distance Statistics. *Geographical Analysis*, 24(3), 189-206. doi:[doi:10.1111/j.15384632.1992.tb00261.x](#)

Examples

```
library(sf)

# Transform data to UTM zone 15N so that cell_size and bandwidth can be set
# in metres
memphis_robberies_utm <- st_transform(memphis_robberies_jan, 32615)

# Automatically set grid-cell size, bandwidth and neighbour distance

hotspot_gistar(memphis_robberies_utm)

# Manually set grid-cell size in metres, since the `memphis_robberies`
# dataset uses a co-ordinate reference system (UTM zone 15 north) that is
# specified in metres

hotspot_gistar(memphis_robberies_utm, cell_size = 200)

# Automatically set grid-cell size and bandwidth for lon/lat data. The data
# will be transformed automatically before KDE values are calculated and the
# result will then be transformed back to the original CRS.

hotspot_gistar(memphis_robberies)
```

hotspot_grid

Create either a rectangular or hexagonal two-dimensional grid

Description

Create either a rectangular or hexagonal two-dimensional grid

Usage

```
hotspot_grid(data, cell_size = NULL, grid_type = "rect", quiet = FALSE, ...)
```

Arguments

data	sf data frame.
cell_size	numeric value specifying the size of each equally spaced grid cell, using the same units (metres, degrees, etc.) as used in the sf data frame given in the data argument. If this argument is NULL (the default), the cell size will be calculated automatically (see Details).
grid_type	character specifying whether the grid should be made up of squares ("rect", the default) or hexagons ("hex").
quiet	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.
...	Further arguments passed to <code>link[sf]{st_make_grid}</code> .

Value

A simple features tibble containing polygons representing grid cells.

The grid will be based on the convex hull of data, expanded by a buffer of `cell_size / 2` to ensure all the points in data fall within the resulting grid.

hotspot_isoband	<i>Convert a hotspot grid to isobands</i>
-----------------	---

Description

Generalise values in a regular square grid into polygon bands. The result is an [sf::sf](#) object with one row for each non-empty band and can be plotted with [autoplot\(\)](#) or [hotspot_layer\(\)](#).

Usage

```
hotspot_isoband(
  data,
  value = NULL,
  breaks = 5,
  style = "equal",
  critical_p = 0.05,
  quiet = FALSE
)
```

Arguments

data	An sf::sf object containing a regular square grid and at least one numeric column, typically produced by a <code>hotspot_*</code> () function.
value	The unquoted name of the numeric column to convert. If NULL, a suitable column is inferred from the <code>sfhotspot</code> class of data, or from the sole numeric column in an otherwise unrecognised SF object.

breaks	Either a single positive integer giving the requested number of bands (five by default), or a strictly increasing numeric vector giving the band boundaries. If multiple boundaries are supplied, style is silently ignored.
style	Method used to calculate breaks when breaks has length one. Values other than "pconventions" are passed to the style argument of <code>classInt::classIntervals()</code> . "pconventions" uses two-sided normal-theory thresholds corresponding to p-values of 0.1, 0.05, 0.01 and 0.001 and ignores breaks.
critical_p	For output from <code>hotspot_gistar()</code> containing a kde column, the largest p-value for a cell to be included. The default is 0.05. This argument is ignored for other inputs.
quiet	If TRUE, suppress informative messages about automatically selected values and potentially unhelpful intervals.

Details

The default style = "equal" divides the observed range into equal-width intervals, providing a discrete analogue of a linear continuous colour scale. Extreme values can make equal-width bands uninformative; a message is produced when at least 90% of finite values fall in one calculated band.

For change values, G_i^*/G_i statistics, and dual-KDE differences or logged ratios, automatically calculated breaks place zero at a boundary or at the centre of a band. Explicit break vectors are never altered.

When data is output from `hotspot_gistar()` and contains KDE values, KDE is converted and cells with `pvalue >= critical_p` are excluded before the bands are calculated. Without KDE values, the G_i^*/G_i statistic is converted.

Values are interpolated between grid-cell centres. Lattice positions absent from a grid clipped to its original analysis area are treated as missing. Consequently, the isobands need not cover the complete area of every outer grid cell.

Value

An sf tibble with class `hspt_ib`. It has one row per non-empty band and columns lower, upper, band, label, and geometry. The band and label columns are ordered factors; band contains technical interval notation and label contains ranges formatted for display.

Examples

```
memphis_robberies_jan |>
  hotspot_kde() |>
  hotspot_isoband() |>
  autoplot()
```

hotspot_kde

*Estimate two-dimensional kernel density of points***Description**

Estimate two-dimensional kernel density of points

Usage

```
hotspot_kde(
  data,
  cell_size = NULL,
  grid_type = "rect",
  bandwidth = NULL,
  bandwidth_adjust = 1,
  grid = NULL,
  weights = NULL,
  transform = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

<code>data</code>	sf data frame containing points.
<code>cell_size</code>	numeric value specifying the size of each equally spaced grid cell, using the same units (metres, degrees, etc.) as used in the <code>sf</code> data frame given in the <code>data</code> argument. Ignored if <code>grid</code> is not <code>NULL</code> . If this argument and <code>grid</code> are <code>NULL</code> (the default), the cell size will be calculated automatically (see Details).
<code>grid_type</code>	character specifying whether the grid should be made up of squares ("rect", the default) or hexagons ("hex"). Ignored if <code>grid</code> is not <code>NULL</code> .
<code>bandwidth</code>	numeric value specifying the bandwidth to be used in calculating the kernel density estimates. If this argument is <code>NULL</code> (the default), the bandwidth will be determined automatically using the result of bandwidth.nrd called on the co-ordinates of <code>data</code> .
<code>bandwidth_adjust</code>	single positive numeric value by which the value of <code>bandwidth</code> is multiplied. Useful for setting the bandwidth relative to the default.
<code>grid</code>	sf data frame containing polygons, which will be used as the grid for which densities are estimated.
<code>weights</code>	<code>NULL</code> or the name of a column in <code>data</code> to be used as weights for weighted counts and KDE values.
<code>transform</code>	the underlying <code>SpatialKDE</code> package cannot calculate kernel density for lon/lat data, so this must be transformed to use a projected co-ordinate reference system. If this argument is <code>TRUE</code> (the default) and <code>sf::st_is_longlat(data)</code>

	is TRUE, data (and grid if provided) will be transformed automatically using <code>link{st_transform_auto}</code> before the kernel density is estimated and transformed back afterwards. Set this argument to FALSE to suppress automatic transformation of the data.
<code>quiet</code>	if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.
<code>...</code>	Further arguments passed to <code>kde</code> .

Details

This function creates a regular two-dimensional grid of cells (unless a custom grid is specified with `grid`) and calculates the density of points in each cell on that grid using functions from the `SpatialKDE` package. The count of points in each cell is also returned.

Coverage of the output data:

The grid produced by this function covers the convex hull of the input data layer. This means the result may include KDE values for cells that are outside the area for which data were provided, which could be misleading. To handle this, consider cropping the output layer to the area for which data are available. For example, if you only have crime data for a particular district, crop the output dataset to the district boundary using `st_intersection`.

Automatic cell-size selection:

If no cell size is given then the cell size will be set so that there are 50 cells on the shorter side of the grid. If the data `SF` object is projected in metres or feet, the number of cells will be adjusted upwards so that the cell size is a multiple of 100.

Value

An `sf` tibble of grid cells with corresponding point counts and kernel density estimates for each cell. This can be plotted using `autoplot`.

References

Yin, P. (2020). Kernels and Density Estimation. *The Geographic Information Science & Technology Body of Knowledge* (1st Quarter 2020 Edition), John P. Wilson (ed.). doi:[doi:10.22224/gistbok/2020.1.12](https://doi.org/10.22224/gistbok/2020.1.12)

Examples

```
library(sf)

# Transform data to UTM zone 15N so that cell_size and bandwidth can be set
# in metres
memphis_robberies_utm <- st_transform(memphis_robberies_jan, 32615)

# Automatically set grid-cell size, bandwidth and neighbour distance

hotspot_kde(memphis_robberies_utm)
```

```
# Manually set grid-cell size and bandwidth in metres, since the
# `memphis_robberies_utm` dataset uses a co-ordinate reference system (UTM
# zone 15 north) that is specified in metres

hotspot_kde(memphis_robberies_utm, cell_size = 200, bandwidth = 1000)
```

hotspot_layer

Create a ggplot layer from hotspot results

Description

hotspot_layer() is a student-friendly wrapper around `ggplot2::autolayer()`. It uses the plotting method for the class of object, so results from different hotspot_*() functions are displayed using the appropriate variable and aesthetic mapping.

Usage

```
hotspot_layer(object, ...)
```

Arguments

object	An object returned by an sfhotspot analysis.
...	Arguments passed to the corresponding <code>ggplot2::autolayer()</code> method.

Details

Use hotspot_layer() when combining hotspot results with other ggplot2 layers. To create a complete map with suitable scales, labels and an optional base map, use `hotspot_map()` instead.

Value

A ggplot2 layer, or a list of layers when the selected plotting method requires more than one layer.

See Also

`hotspot_map()` for creating a complete map and `ggplot2::autolayer()` for the ggplot2 generic that this function wraps.

Examples

```
robbery_counts <- hotspot_count(
  memphis_robberies,
  cell_size = 0.01,
  quiet = TRUE
)

ggplot2::ggplot() +
```

```
hotspot_layer(robbery_counts) +
  ggplot2::scale_fill_distiller(palette = "Blues", direction = 1) +
  ggplot2::theme_void()
```

hotspot_map

Create an automatic map

Description

hotspot_map() creates a map from an sf object or the result of an sfhotspot analysis using reasonable defaults. Maps include an OpenStreetMap base map by default and are returned as [ggplot2::ggplot](#) objects, so they can be customised with ggplot2 layers, scales and themes.

Usage

```
hotspot_map(object, mapping = NULL, ...)
```

```
## S3 method for class 'sf'
hotspot_map(
  object,
  mapping = NULL,
  ...,
  basemap_type = "osm",
  basemap_zoom = NULL,
  basemap_attribution = NULL,
  caption = NULL,
  quiet = TRUE
)
```

Arguments

object	An sf object or an object returned by an sfhotspot analysis.
mapping	An aesthetic mapping created by ggplot2::aes() , or NULL for no aesthetic mappings. This argument can be used only for ordinary sf objects, not objects with an hspt_* class created by one of the hotspot_*() family of functions.
...	For ordinary sf objects, fixed aesthetics and other arguments passed to ggplot2::geom_sf() . For hotspot-result objects, arguments passed to the corresponding ggplot2::autoplot() method.
basemap_type	A map type passed to the type argument of ggspatial::annotation_map_tile() , or "none" for no base map.
basemap_zoom	The zoom level passed to ggspatial::annotation_map_tile() , or NULL to choose it automatically.
basemap_attribution	Attribution for the tile provider, or NULL to use the statement known for basemap_type. A non-empty string is required for a custom or otherwise unknown map type.

caption	Additional text to display in the map caption, or NULL for no additional text. Any explanatory caption generated by the plotting method is retained above this text and the base-map attribution is retained below it.
quiet	If TRUE (the default), suppress progress bars and other messages. If FALSE, show tile-download progress and other messages.

Details

For an ordinary `sf` object, mapping supplies aesthetic mappings for the feature layer and ... supplies fixed values and other arguments to `ggplot2::geom_sf()`. Hotspot-result objects determine their own aesthetic mappings, so mapping cannot be supplied for those classes.

Value

A `ggplot2::ggplot` object.

Examples

```
hotspot_map(memphis_robberies, basemap_type = "none")
hotspot_map(
  memphis_robberies,
  mapping = ggplot2::aes(colour = offense_type),
  basemap_type = "none",
  caption = "Memphis robberies"
)
```

memphis_population	<i>Populations of census blocks in Memphis in 2020</i>
--------------------	--

Description

A dataset containing records of populations associated with the centroids of census blocks in Memphis, Tennessee, in 2020.

Usage

```
memphis_population
```

Format

A simple-features tibble with 10,393 rows and three variables:

geoid the census GEOID for each block

population the number of people residing in each block

geometry the co-ordinates of the centroid of each block, stored in simple-features point format

Source

US Census Bureau. Census 2020, Redistricting Data summary file. <https://www.census.gov/programs-surveys/decennial-census/about/rdo/summary-files.html>

memphis_precincts	<i>Memphis Police Department Precincts</i>
-------------------	--

Description

A dataset containing the boundaries of Memphis Police Department precincts.

Usage

```
memphis_precincts
```

Format

A simple-features tibble with 9 rows and two variables:

precinct the precinct name

geometry the boundary of each precinct, stored in simple-features polygon format

Licence: Public domain <https://data.memphistn.gov/d/tdws-78iq>

Source

City of Memphis <https://data.memphistn.gov/d/rqqz-pj4u>

memphis_robberies	<i>Personal robberies in Memphis in 2019</i>
-------------------	--

Description

A dataset containing records of personal robberies recorded by police in Memphis, Tennessee, in 2019.

Usage

```
memphis_robberies
```

Format

A simple-features tibble with 2,245 rows and four variables:

uid a unique identifier for each robbery

offense_type the type of crime (always 'personal robbery')

date the date and time at which the crime occurred

geometry the co-ordinates at which the crime occurred, stored in simple-features point format

Source

Crime Open Database, <https://osf.io/zyaqn/>

memphis_robberies_jan *Personal robberies in Memphis in January 2019*

Description

A dataset containing records of personal robberies recorded by police in Memphis, Tennessee, in January 2019. This dataset is too small for some types of analysis but is included for testing purposes.

Usage

```
memphis_robberies_jan
```

Format

A simple-features tibble with 206 rows and four variables:

uid a unique identifier for each robbery

offense_type the type of crime (always 'personal robbery')

date the date and time at which the crime occurred

geometry the co-ordinates at which the crime occurred, stored in simple-features point format

Source

Crime Open Database, <https://osf.io/zyaqn/>

st_transform_auto *Toggle between lon/lat and UTM co-ordinates*

Description

It is often useful to convert geographic data to a projected co-ordinate system, e.g. for specifying a buffer distance in metres. `st_transform_auto` converts between lon/lat co-ordinates and the UTM or UPS zone for the centroid of the data.

Usage

```
st_transform_auto(data, check = TRUE, quiet = FALSE)
```

Arguments

data An `sf` object.

check Should the function check if the chosen UTM zone covers all the data? If so then a warning is issued if the centroid of any row in the data is more than 1° outside the area covered by the UTM zone.

quiet if set to TRUE, messages reporting the values of any parameters set automatically will be suppressed. The default is FALSE.

Value

SF object transformed to new CRS.

Index

* datasets

- memphis_population, 37
- memphis_precincts, 38
- memphis_robberies, 38
- memphis_robberies_jan, 39

- autolayer.hspt_c (autoplot.hspt_c), 2
- autolayer.hspt_d (autoplot.hspt_d), 4
- autolayer.hspt_dk (autoplot.hspt_dk), 5
- autolayer.hspt_g (autoplot.hspt_g), 7
- autolayer.hspt_ib (autoplot.hspt_ib), 8
- autolayer.hspt_k (autoplot.hspt_k), 10
- autolayer.hspt_n (autoplot.hspt_n), 11
- autolayer.hspt_s (autoplot.hspt_s), 13
- autoplot, 16, 18, 22, 34
- autoplot(), 26, 30, 31
- autoplot.hspt_c, 2
- autoplot.hspt_d, 4
- autoplot.hspt_dk, 5
- autoplot.hspt_g, 7
- autoplot.hspt_ib, 8
- autoplot.hspt_k, 10
- autoplot.hspt_n, 11
- autoplot.hspt_s, 13

- bandwidth.nrd, 25, 28, 33

- classInt::classIntervals(), 32

- dbscan::dbscan(), 23

- ggplot2::aes(), 36
- ggplot2::autolayer(), 35
- ggplot2::autoplot(), 36
- ggplot2::geom_sf(), 3, 4, 6, 8, 9, 11, 12, 14, 36, 37
- ggplot2::ggplot, 3, 5, 6, 8, 10, 11, 13, 14, 36, 37
- ggspatial::annotation_map_tile(), 3–6, 8, 9, 11, 12, 14, 36

- hotspot_change, 15
- hotspot_change(), 4
- hotspot_classify, 17, 19, 20
- hotspot_classify(), 2, 3
- hotspot_classify_params, 18, 19
- hotspot_clip, 20
- hotspot_count, 21
- hotspot_count(), 11, 12
- hotspot_dbscan, 22
- hotspot_dbscan(), 13, 14
- hotspot_dual_kde, 24
- hotspot_dual_kde(), 5, 6, 16
- hotspot_gistar, 27
- hotspot_gistar(), 8, 32
- hotspot_grid, 30
- hotspot_isoband, 31
- hotspot_isoband(), 8, 9
- hotspot_kde, 33
- hotspot_kde(), 10, 11
- hotspot_layer, 35
- hotspot_layer(), 31
- hotspot_map, 36
- hotspot_map(), 35
- hotspot_map.hspt_c (autoplot.hspt_c), 2
- hotspot_map.hspt_d (autoplot.hspt_d), 4
- hotspot_map.hspt_dk (autoplot.hspt_dk), 5
- hotspot_map.hspt_g (autoplot.hspt_g), 7
- hotspot_map.hspt_ib (autoplot.hspt_ib), 8
- hotspot_map.hspt_k (autoplot.hspt_k), 10
- hotspot_map.hspt_n (autoplot.hspt_n), 11
- hotspot_map.hspt_s (autoplot.hspt_s), 13

- kde, 26, 29, 34

- localG, 29, 30

- memphis_population, 37
- memphis_precincts, 38

memphis_robberies, [38](#)
memphis_robberies_jan, [39](#)

p.adjust, [20](#), [29](#)
p.adjustSP, [18](#), [20](#), [29](#)

sf, [15–18](#), [20–22](#), [25](#), [26](#), [28](#), [29](#), [31](#), [33](#), [34](#), [39](#)
sf::sf, [23](#), [31](#)
st_intersection, [16](#), [19](#), [20](#), [27](#), [29](#), [34](#)
st_transform_auto, [26](#), [39](#)
st_transform_auto(), [23](#)