Package ‘spatialTIME’

March 22, 2023

Title Spatial Analysis of Vectra Immunoflourescent Data
Version 1.3.3-3
Description Visualization and analysis of Vectra Immunoflourescent data. Options for calculating both the univariate and bivariate Ripley's K are included. Calculations are performed using a permutation-based approach presented by Wilson et al. <doi:10.1101/2021.04.27.21256104>.
License MIT + file LICENSE
Imports magrittr, dplyr, tidyr, ggplot2, scales, grDevices, purrr, rlang, plyr, spatstat.geom, spatstat.explore, RColorBrewer, furrr, future, tidyselect, crayon
Suggests knitr, rmarkdown, testthat (>= 3.0.0), gridExtra, pheatmap
VignetteBuilder knitr
URL https://github.com/FridleyLab/spatialTIME
BugReports https://github.com/FridleyLab/spatialTIME/issues
NeedsCompilation yes
RoxygenNote 7.2.3
Depends R (>= 2.10)
LazyData true
Encoding UTF-8
Author Jordan Creed [aut], Ram Thapa [aut], Christopher Wilson [aut], Alex Soupir [aut], Oscar Ospina [aut], Julia Wrobel [aut], Brooke Fridley [cph], Fridley Lab [cre]
Maintainer Fridley Lab <fridley.lab@moffitt.org>
Repository CRAN
Date/Publication 2023-03-22 14:00:02 UTC
R topics documented:

- bi_NN_G ................................. 2
- bi_ripleys_k .......................... 4
- create_mif ............................ 6
- example_clinical ..................... 7
- example.spatial ........................ 7
- example.summary ...................... 8
- merge.mifs ............................ 8
- NN_G ................................... 9
- plot.immunoflo ........................ 11
- ripley.s_k ................................ 12
- subset.mifs ............................ 14

Index 16

bi_NN_G  Bivariate Nearest Neighbor Based Measures of Spatial Clustering for IF data

Description

This function computes the nearest neighbor distribution for a particular marker relative to another marker for the observed and permuted point processes.

Usage

```r
bi_NN_G(
  mif,
  mnames,
  r_range = seq(0, 100, 50),
  num_permutations = 50,
  edge_correction = "rs",
  keep_perm_dis = FALSE,
  exhaustive = TRUE,
  workers = 1,
  overwrite = FALSE,
  xloc = NULL,
  yloc = NULL
)
```

Arguments

- `mif`  An MIF object
- `mnames`  Character vector of marker names to estimate degree of nearest neighbor distribution
- `r_range`  Numeric vector of potential r values this range must include 0. Note that the range selected is very different than count based measures. See details.
bi_NN_G

num_permutations
Numeric value indicating the number of permutations used. Default is 50.

edge_correction
Character value indicating the type of edge correction to use. Options include "rs" or "hans".

keep_perm_dis
Logical value determining whether or not to keep the full distribution of permuted G values

exhaustive
Logical. If TRUE then markers must be a vector and spatial measures will be computed all pairs of unique markers. If FALSE then markers must be a data.frame with the desired combinations.

workers
Integer value for the number of workers to spawn

overwrite
Logical value determining if you want the results to replace the current output (TRUE) or be to be appended (FALSE).

xloc
a string corresponding to the x coordinates. If null the average of XMin and XMax will be used

yloc
a string corresponding to the y coordinates. If null the average of YMin and YMax will be used

Value

Returns a data frame

anchor
Marker for which the distances are measured from

counted
Marker for which the distances are measured to

Theoretical CSR
Expected value assuming complete spatial randomness

Permutated CSR
Average observed G for the permuted point process

Observed
Observed value for the observed point process

Degree of Clustering Permuted
Degree of spatial clustering where the reference is the permuted estimate of CSR

Degree of Clustering Theoretical
Degree of spatial clustering where the reference is the theoretical estimate of CSR

Examples

#' #Create mif object
library(dplyr)
x <- create_mif(clinical_data = example_clinical %>%
  mutate(deidentified_id = as.character(deidentified_id)),
  sample_data = example_summary %>%
  mutate(deidentified_id = as.character(deidentified_id)),
  spatial_list = example_spatial,
  patient_id = "deidentified_id",
  sample_id = "deidentified_sample")

#Nearest Neighbor distribution for the colocalization of CD3+CD8+ positive
bi_ripleys_k

Bivariate Ripley's K

Description

Bivariate Ripley's K function within spatialTIME, `bi_ripleys_k` is a function that takes in a 'mIF' object, along with some parameters like marker names of interest and range of radii in which to assess bivariate clustering or colocalization. In 1.3.3.3 we have introduced the ability to forgo the need for permutations with the implementation of the exact CSR estimate. This is both faster and being the exact CSR, produces an exact degree of clustering in the spatial files.

Due to the availability of whole slide images (WSI), there's a possibility users will be running bivariate Ripley's K on samples that have millions of cells. When doing this, keep in mind that a nearest neighbor matrix with *n* cell is *n* by *n* in size and therefore easily consumes high performance compute levels of RAM. To combat this, we have implemented a tiling method that performs counts for small chunks of the distance matrix at a time before finally calculating the bivariate Ripley’s K value on the total counts. When doing this there are now 2 important parameters to keep in mind. The ‘big’ parameter is the size of the tile to use. We have found 1000 to be a good number that allows for high number of cores while maintaining low RAM usage. The other important parameter when working with WSI is nlarge which is the fall over for switching to no edge correction. The spatstat.explore::Kest univariate Ripley’s K uses a default of 3000 but we have defaulted to 1000 to keep compute minimized as edge correction uses large amounts of RAM over ‘none’.

Usage

```r
bi_ripleys_k(
  mif,  
mnames,  
r_range = 0:100,  
edge_correction = "translation",  
num_permutations = 50,  
permute = FALSE,  
keep_permutation_distribution = FALSE,  
overwrite = TRUE,  
workers = 6,  
big = 1000,  
nlarge = 1000,  
xloc = NULL,  
yloc = NULL
)
```
Arguments

- **mif**: mIF object with spatial data frames, clinical, and per-sample summary information.
- **mnames**: vector of column names for phenotypes or data frame of marker combinations.
- **r_range**: vector range of radii to calculate co-localization *K*.
- **edge_correction**: character edge_correction method, one of "translation", "border", "or none".
- **num_permutations**: integer number of permutations to estimate CSR.
- **permute**: whether or not to use permutations to estimate CSR (TRUE) or to calculate exact CSR (FALSE).
- **keep_permutation_distribution**: boolean as to whether to summarise permutations to mean.
- **overwrite**: boolean as to whether to replace existing bivariate_Count if exists.
- **workers**: integer number of CPU workers to use.
- **big**: integer used as the threshold for subsetting large samples, default is 1000 either *i* or *j*.
- **nlarge**: number of cells in either *i* or *j* to flip to no edge correction - at small (relative to whole spatial region) *r* values differences in results between correction methods is negligible so running a few samples is recommended. Perhaps compute outweighs small differences in correction methods.
- **xloc**: the x and y positions that correspond to cells. If left as NULL, XMin, XMax, YMin, and YMax must be present in the spatial files.
- **yloc**: the x and y positions that correspond to cells. If left as NULL, XMin, XMax, YMin, and YMax must be present in the spatial files.

Value

- mif object with bivariate Ripley’s K calculated.

Examples

```r
x <- spatialTIME::create_mif(clinical_data = spatialTIME::example_clinical %>%
  dplyr::mutate(deidentified_id = as.character(deidentified_id)),
  sample_data = spatialTIME::example_summary %>%
  dplyr::mutate(deidentified_id = as.character(deidentified_id)),
  spatial_list = spatialTIME::example_spatial,
  patient_id = "deidentified_id",
  sample_id = "deidentified_sample")

mnames_good <- c("CD3..Opal.570..Positive","CD8..Opal.520..Positive",
  "FOXP3..Opal.620..Positive","PDL1..Opal.540..Positive",
  "PD1..Opal.650..Positive","CD3..CD8.","CD3..FOXP3.")

x2 = bi_ripleys_k(mif = x, mnames = mnames_good[1:2],
  r_range = 0:100, edge_correction = "none", permute = FALSE,
  num_permutations = 50, keep_permutation_distribution = FALSE,
  workers = 1, big = 1000)
```
create_mif  

Create Multiplex Immunofluorescent object

Description

Creates an MIF object for use in spatialIF functions

Usage

create_mif(
  clinical_data,
  sample_data,
  spatial_list = NULL,
  patient_id = "patient_id",
  sample_id = "image_tag"
)

Arguments

clinical_data  A data frame containing patient level data with one row per participant.
sample_data  A data frame containing sample level data with one row per sample. Should at a minimum contain a 2 columns: one for sample names and one for the corresponding patient name.
spatial_list  A named list of data frames with the spatial data from each sample making up each individual data frame
patient_id  A character string indicating the column name for patient id in sample and clinical data frames.
sample_id  A character string indicating the column name for sample id in the sample data frame

Value

Returns a custom MIF

clinical  Data frame of clinical data
sample  Data frame of sample data
spatial  Named list of spatial data
derived  List of data derived using the MIF object
patient_id  The column name for sample id in the sample data frame with the clinical data
sample_id  The column name for sample id in the sample data frame to merge with the spatial data
Examples

```r
#Create mif object
library(dplyr)
x <- create_mif(clinical_data = example_clinical %>%
    mutate(deidentified_id = as.character(deidentified_id)),
    sample_data = example_summary %>%
    mutate(deidentified_id = as.character(deidentified_id)),
    spatial_list = example_spatial,
    patient_id = "deidentified_id",
    sample_id = "deidentified_sample")
```

---

**example_clinical**  \hspace{1cm} *Clinical variables of 229 patients*

---

**Description**

A tibble with clinical characteristics for 229 patients

**Usage**

```r
example_clinical
```

**Format**

A tibble with 229 rows and 6 variables

- **age**  age at diagnosis
- **race**  self-identifed race
- **sex**  patient biological sex
- **status**  disease status
- **deidentified_sample**  sample identifier
- **deidentified_id**  patient identifier

---

**example_spatial**  \hspace{1cm} *Example list of 5 spatial TMA data*

---

**Description**

A list containing 5 spatial data frames

**Usage**

```r
example_spatial
```
merge_mifs

Format
A list of 5 data frames:

• TMA_{\[3,B\]}\text{.tiff}
• TMA_{\[6,F\]}\text{.tiff}
• TMA_{\[7,B\]}\text{.tiff}
• TMA_{\[9,K\]}\text{.tiff}
• TMA_{\[8,U\]}\text{.tiff}

example_summary  Marker summaries of 229 samples

Description
A dataset containing summaries of 25 markers and 229 samples

Usage
example_summary

Format
A tibble with 229 rows and 29 variables:

deidentified_id  patient-level id
deidentified_sample  sample-level id ...

merge_mifs  Merge several MIF objects together

Description
This function merges MIF objects that were run separately so they can be used as a single MIF. MIF objects don’t *need* but *should* have the same column names in the summary file and clinical data file. The MIF objects **DO** need to have the same patient_id and sample_id.

Usage
merge_mifs(mifs = NULL, check.names = T)

Arguments
mifs  A list of MIF objects to merge together
check.names  whether to check names of spatial files and summary entries
NN_G

Nearest Neighbor Based Measures of Spatial Clustering for IF data

Description

For a given cell type, this function computes proportion of cells that have nearest neighbor less than r for the observed and permuted point processes.

Usage

```
NN_G(
  mif,
  mnames,
  r_range = seq(0, 100, 50),
  num_permutations = 50,
  edge_correction = "rs",
  keep_perm_dis = FALSE,
  workers = 1,
  overwrite = FALSE,
  xloc = NULL,
  yloc = NULL
)
```

Value

Returns a new MIF object list

- **clinical_data**: clinical information from all sample
- **sample**: cell level summary data from all
- **spatial**: contains all spatial files from all MIFs
- **derived**: appended derived variables
- **patient_id**: patient_id from the first MIF - this is why it is important to have the same patient_id for all MIFs
- **sample_id**: sample_id from the first MIF - also important for all MIFs to have the same sample_id

Examples

```
# merge several MIF objects
library(dplyr)
x <- create_mif(clinical_data = example_clinical %>%
  mutate(deidentified_id = as.character(deidentified_id)),
  sample_data = example_summary %>%
  mutate(deidentified_id = as.character(deidentified_id)),
  spatial_list = example_spatial,
  patient_id = "deidentified_id",
  sample_id = "deidentified_sample")
x <- merge_mifs(mifs = list(x, x), check.names = FALSE)
```
Arguments

- **mif**: An MIF object
- **mnames**: Character vector of marker names to estimate degree of nearest neighbor distribution
- **r_range**: Numeric vector of potential r values this range must include 0.
- **num_permutations**: Numeric value indicating the number of permutations used. Default is 50.
- **edge_correction**: Character value indicating the type of edge correction to use. Options include "rs" or "hans".
- **keep_perm_dis**: Logical value determining whether or not to keep the full distribution of permuted G values
- **workers**: Integer value for the number of workers to spawn
- **overwrite**: Logical value determining if you want the results to replace the current output (TRUE) or be to be appended (FALSE).
- **xloc**: a string corresponding to the x coordinates. If null the average of XMin and XMax will be used
- **yloc**: a string corresponding to the y coordinates. If null the average of YMin and YMax will be used

Value

Returns a data.frame

- **Theoretical CSR**: Expected value assuming complete spatial randomness
- **Permuted CSR**: Average observed G for the permuted point process
- **Observed**: Observed value for the observed point process
- **Degree of Clustering Permuted**: Degree of spatial clustering where the reference is the permuted estimate of CSR
- **Degree of Clustering Theoretical**: Degree of spatial clustering where the reference is the theoretical estimate of CSR

Examples

```r
#Create mif object
library(dplyr)
x <- create_mif(clinical_data = example_clinical %>%
  mutate(deidentified_id = as.character(deidentified_id)),
sample_data = example_summary %>%
  mutate(deidentified_id = as.character(deidentified_id)),
spatial_list = example_spatial,
patient_id = "deidentified_id",
sample_id = "deidentified_sample")
```
# Define the set of markers to study
markers <- c("CD3..Opal.570..Positive","CD8..Opal.520..Positive", 
"FOXP3..Opal.620..Positive","CD3..CD8."","CD3..FOXP3."")

# Nearest Neighbor distribution for all markers with a neighborhood size 
# of 10,20,...,100 (zero must be included in the input).

x <- NN_G(mif = x, mnames = markers[1:2], num_permutations = 1, 
edge_correction = 'rs', r = seq(0,100,10),
keep_perm_dis = FALSE, workers = 1)

---

**plot_immunoflo**  
*Generate plot of TMA point process*

**Description**

This function generates plot of point process in rectangular or circular window.

**Usage**

```r
plot_immunoflo(
  mif,  
  plot_title,  
  mnames,  
  mcolors = NULL,  
  cell_type = NULL,  
  filename = NULL,  
  path = NULL
)
```

**Arguments**

- `mif`  
  MIF object created using `create_MIF()`.
- `plot_title`  
  Character string or vector of character strings of variable name(s) to serve as plot title(s).
- `mnames`  
  Character vector containing marker names.
- `mcolors`  
  Character vector of color names to display markers in the plot.
- `cell_type`  
  Character vector of cell type
- `filename`  
  Character string of file name to store plots. Plots are generated as single .pdf file.
- `path`  
  Different path than file name or to use in conjunction with filename ??

**Value**

mif object and the ggplot objects can be viewed form the derived slot of the mif object
**Examples**

```r
#Create mif object
library(dplyr)
x <- create_mif(clinical_data = example_clinical %>%
  mutate(deidentified_id = as.character(deidentified_id)),
  sample_data = example_summary %>%
  mutate(deidentified_id = as.character(deidentified_id)),
  spatial_list = example_spatial,
  patient_id = "deidentified_id",
  sample_id = "deidentified_sample")

mnames_good <- c("CD3..Opal.570..Positive","CD8..Opal.520..Positive",
  "FOXP3..Opal.620..Positive","PDL1..Opal.540..Positive",
  "PD1..Opal.650..Positive","CD3..CD8.","CD3..FOXP3.")

x <- plot_immunoflo(x, plot_title = "deidentified_sample", mnames = mnames_good,
  cell_type = "Classifier.Label")

x[['derived']][['spatial_plots']][[4]]
```

---

**ripleys_k**

*Calculate Ripley's K*

**Description**

*ripleys_k()* calculates the empirical Ripley's K measurement for the cell types specified by *mnames* in the mIF object. This is very useful when exploring the spatial clustering of single cell types on TMA cores or ROI spots following processing with a program such as HALO for cell phenotyping.

In the ‘ripleys_k’ function, there is the ability to perform permutations in order to assess whether the clustering of a cell type is significant, or the ability to derive the exact CSR and forgo permutations for much faster sample processing. Permutations can be helpful if the significance of clustering was not to be identified - run 1000 permutations and if observed is outside 95-percentile then significant clustering. We, however, recommend using the exact CSR estimate due to speed.

Some things to be aware of when computing the exact Ripley's K estimate, if your spatial file is greater than the 'big' size, the edge correction will be converted to 'none' in order to save on resources and compute time. Due to the introduction of Whole Slide Imaging (WSI), this can easily be well over 1,000,000 cells, and calculating edge correction for these spatial files will not succeed when attempting to force an edge correction on it.

**Usage**

```r
ripleys_k(
  mif,
  mnames,
  r_range = seq(0, 100, 1),
  num_permutations = 50,
  edge_correction = "translation",
```
method = "K",
permute = FALSE,
keep_permutation_distribution = FALSE,
workers = 1,
overwrite = FALSE,
xloc = NULL,
yloc = NULL,
big = 10000
)

Arguments

mif object of class ‘mif’ created with ‘create_mif’
mnames cell phenotype markers to calculate Ripley’s K for
r_range radius range (including 0)
num_permutations number of permutations to use to estimate CSR. If ‘keep_perm_dis’ is set to FALSE, this will be ignored
edge_correction edge correction method to pass to ‘Kest’. can take one of "translation", "isotropic", "none"
method not used currently
permute whether to use CSR estimate or use permutations to determine CSR
keep_permutation_distribution whether to find mean of permutation distribution or each permutation calculation
workers number of cores to use for calculations
overwrite whether to overwrite the ‘univariate_Count’ slot within ‘mif$derived’
xloc the location of the center of cells. If left ‘NULL’, ‘XMin’, ‘XMax’, ‘YMin’, and ‘YMax’ must be present.
yloc the location of the center of cells. If left ‘NULL’, ‘XMin’, ‘XMax’, ‘YMin’, and ‘YMax’ must be present.
big the number of cells at which to flip from an edge correction method other than ‘none’ to ‘none’ due to size

Value

object of class ‘mif’

Examples

x <- spatialTIME::create_mif(clinical_data = spatialTIME::example_clinical %>%
dplyr::mutate(deidentified_id = as.character(deidentified_id)),
sample_data = spatialTIME::example_summary %>%
dplyr::mutate(deidentified_id = as.character(deidentified_id)),
spatial_list = spatialTIME::example_spatial,
patient_id = "deidentified_id",
...
subset_mif

Subset mif object on cellular level

Description

This function allows to subset the mif object into compartments. For instance a mif object includes all cells and the desired analysis is based on only the tumor or stroma compartment then this function will subset the spatial list to just the cells in the desired compartment.

Usage

subset_mif(mif, classifier, level, markers)

Arguments

mif An MIF object
classifier Column name for spatial dataframe to subset
level Determines which level of the classifier to keep.
markers vector of

Value

mif object where the spatial list only as the cell that are the specified level.

Examples

```r
# Create mif object
library(dplyr)
x <- create_mif(clinical_data = example_clinical %>%
    mutate(deidentified_id = as.character(deidentified_id)),
sample_data = example_summary %>%
    mutate(deidentified_id = as.character(deidentified_id)),
sample_id = "deidentified_sample")
mnames = x$spatial[[1]] %>%
colnames() %>%
grep("Pos|CD", ., value =TRUE) %>%
grep("Cyto|Nucle", ., value =TRUE, invert =TRUE)
x2 = ripleys_k(mif = x,
    mnames = mnames[1],
r_range = seq(0, 100, 1),
num_permutations = 100,
edge_correction = "translation",
method = "K",
permute = FALSE,
keep_permutation_distribution =FALSE,
workers = 1,
overwrite =TRUE)
```
subset_mif

```r
spatial_list = example_spatial,
patient_id = "deidentified_id",
sample_id = "deidentified_sample")

markers = c("CD3..Opal.570..Positive", "CD8..Opal.520..Positive",
"FOXP3..Opal.620..Positive", "PDL1..Opal.540..Positive",
"PD1..Opal.650..Positive", "CD3..CD8.", "CD3..FOXP3."

mif_tumor = subset_mif(mif = x, classifier = 'Classifier.Label',
level = 'Tumor', markers = markers)
```
Index

* datasets
  example_clinical, 7
  example_spatial, 7
  example_summary, 8

bi_NN_G, 2
bi_ripleys_k, 4
create_mif, 6

dataset
example_clinical, 7
example_spatial, 7
example_summary, 8
merge_mifs, 8

plot_immunoflo, 11
ripleys_k, 12
subset_mif, 14