

# Package ‘spatialTIME’

December 19, 2023

**Title** Spatial Analysis of Vectra Immunofluorescent Data

**Version** 1.3.4-2

**Description** Visualization and analysis of Vectra Immunofluorescent 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](https://doi.org/10.1101/2021.04.27.21256104)>.

**License** MIT + file LICENSE

**Imports** magrittr, dplyr, tidyr, ggplot2, scales, grDevices, purrr, spatstat.geom, spatstat.explore, RColorBrewer, furrr, future, tidyselect, crayon, pbmcapply, dixon, tibble, stringr

**Suggests** knitr, devtools, rmarkdown, testthat (>= 3.0.0), gridExtra, pheatmap

**VignetteBuilder** knitr

**URL** <https://github.com/FridleyLab/spatialTIME>

**BugReports** <https://github.com/FridleyLab/spatialTIME/issues>

**NeedsCompilation** no

**Config/testthat/edition** 3

**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](mailto:fridley.lab@moffitt.org)>

**Repository** CRAN

**Date/Publication** 2023-12-19 18:40:02 UTC

**R topics documented:**

bi_NN_G . . . . .	2
bi_pair_correlation . . . . .	4
bi_ripleys_k . . . . .	5
bi_ripleys_k_WSI . . . . .	6
compute_metrics . . . . .	8
create_mif . . . . .	10
dixons_s . . . . .	11
example_clinical . . . . .	13
example_spatial . . . . .	14
example_summary . . . . .	14
interaction_variable . . . . .	15
merge_mifs . . . . .	16
NN_G . . . . .	17
pair_correlation . . . . .	18
plot_immunoflo . . . . .	19
ripleys_k . . . . .	20
subset_mif . . . . .	22
<b>Index</b>	<b>24</b>

---

bi_NN_G	<i>Bivariate Nearest Neighbor G(r)</i>
---------	--

---

**Description**

Bivariate Nearest Neighbor G(r)

**Usage**

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

**Arguments**

mif	object of class 'mif' created by function 'create_mif()'
mnames	character vector of column names within the spatial files, indicating whether a cell row is positive for a phenotype
r_range	numeric vector of radii around marker positive cells which to use for G(r)
num_permutations	integer number of permutations to use for estimating core specific complete spatial randomness (CSR)
edge_correction	character vector of edge correction methods to use: "rs", "km" or "han"
keep_perm_dis	boolean for whether to summarise permutations to a single value or maintain each permutations result
workers	integer number for the number of CPU cores to use in parallel to calculate all samples/markers
overwrite	boolean whether to overwrite previous run of NN G(r) or increment "RUN" and maintain previous measurements
xloc, yloc	the x and y location columns in the spatial files that indicate the center of the respective cells

**Value**

object of class 'mif' containing a new slot under 'derived' got nearest neighbor distances

**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[1:2],
  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.")
## Not run:
x2 = bi_NN_G(mif = x, mnames = mnames_good[1:2],
  r_range = 0:100, num_permutations = 10,
  edge_correction = "rs", keep_perm_dis = FALSE,
  workers = 1, overwrite = TRUE)

## End(Not run)
```

---

bi\_pair\_correlation    *Bivariate Pair Correlation Function*

---

## Description

Bivariate Pair Correlation Function

## Usage

```
bi_pair_correlation(
  mif,
  mnames,
  r_range = NULL,
  num_permutations = 100,
  edge_correction = "translation",
  keep_permutation_distribution = FALSE,
  workers = 1,
  overwrite = FALSE,
  xloc = NULL,
  yloc = NULL,
  ...
)
```

## Arguments

mif	object of class ‘mif’
mnames	character vector or dataframe with 2 columns containing markers/marker combinations to run
r_range	numeric vector radii to measure
num_permutations	integer for the number of permutations to run
edge_correction	character string for which edge correction to implement for Ripley’s K
keep_permutation_distribution	boolean whether to summarise the permutations or keep all
workers	integer for number of cores to use when calculating
overwrite	boolean for whether to overwrite existing bivariate pair correlation results
xloc	x location column in spatial files
yloc	y location column in spatial files
...	other variables to pass to ‘[spatstat.explore::pcfcross]’

## Value

‘mif’ object with the bivariate\_pair\_correlation slot filled

## 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  $n$  cells 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 import 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

```
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,  
  xloc = NULL,  
  yloc = NULL,  
  force = FALSE  
)
```

## 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

xloc, 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

force logical whether or not to continue if sample has more than 10,000 cells

### Value

mif object with bivariate Ripley's K calculated

### 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",
  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)
```

---

bi\_ripleys\_k\_WSI

*Bivariate Ripley's K for Whole Slide Images*

---

### 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 forsgo 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  $n \times n$  cell is  $n \times n$  by  $n \times n$  in size and therefore easily consumers 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 import 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

```
bi_ripleys_k_WSI(
  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", 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**

```
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_WSI(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)
```

---

compute\_metrics

*Calculate Count Based Measures and NN Measures of Spatial Clustering for IF data*

---

**Description**

This function calculates count based Measures (Ripley's K, Besag L, and Marcon's M) of IF data to characterize correlation of spatial point process. For nearest neighbor calculations of 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**

```
compute_metrics(
  mif,
  mnames,
  r_range = seq(0, 100, 50),
  num_permutations = 50,
```

```

edge_correction = c("translation"),
method = c("K"),
k_trans = "none",
keep_perm_dis = FALSE,
workers = 1,
overwrite = FALSE,
xloc = NULL,
yloc = NULL,
exhaustive = T
)

```

### Arguments

mif	An MIF object
mnames	Character vector of marker names to estimate degree of spatial clustering.
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 vector indicating the type of edge correction to use. Options for count based include "translation" or "isotropic" and for nearest neighbors include "rs" or "hans".
method	Character vector indicating which count based measure (K, BiK, G, BiG) used to estimate the degree of spatial clustering. Description of the methods can be found in Details section.
k_trans	Character value of the transformation to apply to count based metrics (none, M, or L)
keep_perm_dis	Logical value determining whether or not to keep the full distribution of permuted K or 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
exhaustive	whether or not to compute all combinations of markers

### Value

Returns a data.frame	
Theoretical CSR	Expected value assuming complete spatial randomness
Permuted CSR	Average observed K, L, or M 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")

# Define the set of markers to study
mnames <- c("CD3..Opal.570..Positive", "CD8..Opal.520..Positive",
"FOXP3..Opal.620..Positive", "CD3..CD8.", "CD3..FOXP3.")

# Ripley's K and nearest neighbor G for all markers with a neighborhood size
# of 10,20,...,100 (zero must be included in the input).
```

---

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**

```
#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")
```

---

dixons\_s

*Dixon's S Segregation Statistic*


---

**Description**

This function processes the spatial files in the mif object, requiring a column that distinguishes between different groups i.e. tumor and stroma

**Usage**

```
dixons_s(
  mif,
  mnames,
  num_permutations = 1000,
  type = c("Z", "C"),
  workers = 1,
  overwrite = FALSE,
  xloc = NULL,
  yloc = NULL
)
```

**Arguments**

mif	An MIF object
mnames	vector of markers corresponding to spatial columns to check Dixon's S between
num_permutations	Numeric value indicating the number of permutations used. Default is 1000.
type	a character string for the type that is wanted in the output which can be "Z" for z-statistic results or "C" for Chi-squared statistic results
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 for Z-statistic

From

To

Obs.Count

Exp.Count

S

Z

p-val.Z

p-val.Nobs

Marker

Classifier Labeled Column Counts

Image.Tag

Returns a data frame for C-statistic

Segregation

df

Chi-sq

P.asymp

P.rand

Marker

Classifier Labeled Column Counts

Image.Tag

### 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")
```

---

example_clinical	<i>Clinical variables of 229 patients</i>
------------------	---

---

### Description

A tibble wuith clinical characteristics for 229 patients

### Usage

```
example_clinical
```

### Format

A tibble with 229 rows and 6 variables

**age** age at diagnosis

**race** self-idenitified race

**sex** patient biological sex

**status** disease status

**deidentified\_sample** sample identifier

**deidentified\_id** patient identifier

---

example_spatial	<i>Example list of 5 spatial TMA data</i>
-----------------	---

---

**Description**

A list containing 5 spatial data frames

**Usage**

```
example_spatial
```

**Format**

A list of 5 data frames:

- TMA\_[3,B].tiff
- TMA\_[6,F].tiff
- TMA\_[7,B].tiff
- TMA\_[9,K].tiff
- TMA\_[8,U].tiff

---

example_summary	<i>Marker summaries of 229 samples</i>
-----------------	--

---

**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 ...

---

 interaction\_variable *Bivariate Interaction Variable*


---

**Description**

Single-cell spatial-protein metric introduced by Steinhart et al in <https://doi.org/10.1158/1541-7786.mcr-21-0411>

**Usage**

```
interaction_variable(
  mif,
  mnames,
  r_range = NULL,
  num_permutations = 100,
  keep_permutation_distribution = FALSE,
  workers = 1,
  overwrite = FALSE,
  xloc = NULL,
  yloc = NULL
)
```

**Arguments**

mif	object of class 'mif'
mnames	a character vector or table with 2 columns indicating the from-to markers to assess
r_range	numeric vector of radii for which to calculate the interaction variable at
num_permutations	integer for how many permutations to use to derive the interaction estimate under CSR
keep_permutation_distribution	boolean for whether or not to keep all permutation results or average them
workers	integer for the number of CPU cores to use for permutations, markers, and spatial samples
overwrite	boolean for whether to overwrite existing interaction variable results
xloc	column name in spatial files containing the x location - if left NULL will average columns XMin and XMax
yloc	column name in spatial files containing the y location - if left NULL will average columns YMin and YMax

**Value**

object of class mif with the interaction variable derive slot filled

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

**Value**

Returns a new MIF object list

clinical_data	clinical information from all
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)
```

---

 NN\_G
 

---

*Univariate Nearest Neighbor G(r)*


---

**Description**

Univariate Nearest Neighbor G(r)

**Usage**

```

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

```

**Arguments**

mif	object of class 'mif' created by function 'create_mif()'
mnames	character vector of column names within the spatial files, indicating whether a cell row is positive for a phenotype
r_range	numeric vector of radii around marker positive cells which to use for G(r)
num_permutations	integer number of permutations to use for estimating core specific complete spatial randomness (CSR)
edge_correction	character vector of edge correction methods to use: "rs", "km" or "han"
keep_perm_dis	boolean for whether to summarise permutations to a single value or maintain each permutations result
workers	integer number for the number of CPU cores to use in parallel to calculate all samples/markers
overwrite	boolean whether to overwrite previous run of NN G(r) or increment "RUN" and maintain previous measurements
xloc, yloc	the x and y location columns in the spatial files that indicate the center of the respective cells

**Value**

object of class 'mif' containing a new slot under 'derived' got nearest neighbor distances

**Examples**

```

library(dplyr)
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 = NN_G(mif = x, mnames = mnames_good[1:2],
  r_range = 0:100, num_permutations = 10,
  edge_correction = "rs", keep_perm_dis = FALSE,
  workers = 1, overwrite = TRUE)

```

---

pair\_correlation

*Univariate Pair Correlation Function*


---

**Description**

Implementation of the univariate pair correlation function from spatstat

**Usage**

```

pair_correlation(
  mif,
  mnames,
  r_range = NULL,
  num_permutations = 100,
  edge_correction = "translation",
  keep_permutation_distribution = FALSE,
  workers = 1,
  overwrite = FALSE,
  xloc = NULL,
  yloc = NULL,
  ...
)

```

**Arguments**

mif	object of class ‘mif’
mnames	character vector of marker names
r_range	numeric vector including 0. If ignored, ‘spatstat’ will decide range

num\_permutations integer indicating how many permutations to run to determine CSR estimate  
 edge\_correction character string of edge correction to apply to Ripley's K estimation  
 keep\_permutation\_distribution boolean for whether to keep the permutations or not  
 workers integer for number of threads to use when calculating metrics  
 overwrite boolean whether to overwrite existing results in the univariate\_pair\_correlation slot  
 xloc column name of single x value  
 yloc column name of single y value  
 ... other parameters to provide 'spatstat::pcf'  
 The Pair Correlation Function uses the derivative of Ripley's K so it does take slightly longer to calculate  
 'xloc' and 'yloc', if NULL, will be calculated from columns 'XMax', 'XMin', 'YMax', and 'YMin'.

**Value**

mif object with with the univariate\_pair\_correlation derived slot filled or appended to

---

plot_immunoflo	<i>Generate plot of TMA point process</i>
----------------	---

---

**Description**

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

**Usage**

```

plot_immunoflo(
  mif,
  plot_title,
  mnames,
  mcolors = NULL,
  cell_type = NULL,
  filename = NULL,
  path = NULL,
  xloc = NULL,
  yloc = 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 ???
xloc, yloc	columns in the spatial files containing the x and y locations of cells. Default is 'NULL' which will result in 'xloc' and 'yloc' being calculated from 'XMin'/'YMin' and 'XMax'/'YMax'

**Value**

mif object and the ggplot objects can be viewed from the derived slot of the mif object

**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")

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]]
```

## 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

```
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

<code>mif</code>	object of class <code>'mif'</code> created with <code>'create_mif'</code>
<code>mnames</code>	cell phenotype markers to calculate Ripley's K for
<code>r_range</code>	radius range (including 0)
<code>num_permutations</code>	number of permutations to use to estimate CSR. If <code>'keep_perm_dis'</code> is set to <code>FALSE</code> , this will be ignored
<code>edge_correction</code>	edge correction method to pass to <code>'Kest'</code> . can take one of <code>"translation"</code> , <code>"isotropic"</code> , <code>"none"</code>
<code>method</code>	not used currently
<code>permute</code>	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",
  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

*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**

```
## 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")

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, 13
- example\_spatial, 14
- example\_summary, 14

bi\_NN\_G, 2

bi\_pair\_correlation, 4

bi\_ripleys\_k, 5

bi\_ripleys\_k\_WSI, 6

compute\_metrics, 8

create\_mif, 10

dixons\_s, 11

example\_clinical, 13

example\_spatial, 14

example\_summary, 14

interaction\_variable, 15

merge\_mifs, 16

NN\_G, 17

pair\_correlation, 18

plot\_immunoflo, 19

ripleys\_k, 20

subset\_mif, 22