

Package ‘spatialFDA’

November 14, 2025

Title A Tool for Spatial Multi-sample Comparisons

Version 1.3.1

URL <https://github.com/mjemons/spatialFDA>

BugReports <https://github.com/mjemons/spatialFDA/issues>

Description spatialFDA is a package to calculate spatial statistics metrics.

The package takes a SpatialExperiment object and calculates spatial statistics metrics using the package spatstat.

Then it compares the resulting functions across samples/conditions using functional additive models as implemented in the package refund.

Furthermore, it provides exploratory visualisations using functional principal component analysis, as well implemented in refund.

License GPL (>= 3) + file LICENSE

Encoding UTF-8

Depends R (>= 4.3.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, ggplot2, parallel, patchwork, purrr, refund,
SpatialExperiment, spatstat.explore, spatstat.geom,
SummarizedExperiment, methods, stats, fda, tidyr, graphics,
ExperimentHub, scales, S4Vectors

biocViews Software, Spatial, Transcriptomics

VignetteBuilder knitr

Suggests stringr, knitr, rmarkdown, BiocStyle, testthat (>= 3.0.0),
mgcv

Config/testthat/edition 3

git_url <https://git.bioconductor.org/packages/spatialFDA>

git_branch devel

git_last_commit bd8c241

git_last_commit_date 2025-11-03

Repository Bioconductor 3.23

Date/Publication 2025-11-13

Author Martin Emons [aut, cre] (ORCID:

<<https://orcid.org/0009-0000-5219-5311>>),

Samuel Gunz [aut] (ORCID: <<https://orcid.org/0000-0002-8909-0932>>),

Fabian Scheipl [aut] (ORCID: <<https://orcid.org/0000-0001-8172-3603>>),

Mark D. Robinson [aut, fnd] (ORCID:

<<https://orcid.org/0000-0002-3048-5518>>)

Maintainer Martin Emons <martin.emons@uzh.ch>

Contents

| | |
|-------------------------------------|-----------|
| .dfToppp | 2 |
| .extractMetric | 3 |
| .loadExample | 4 |
| .speToDf | 5 |
| calcCrossMetricPerFov | 6 |
| calcMetricPerFov | 7 |
| crossSpatialInference | 8 |
| extractCrossInferenceData | 10 |
| functionalGam | 11 |
| functionalPCA | 13 |
| plotCrossFOV | 14 |
| plotCrossHeatmap | 15 |
| plotCrossMetricPerFov | 16 |
| plotFbPlot | 18 |
| plotFpca | 19 |
| plotMdl | 20 |
| plotMetricPerFov | 21 |
| prepData | 23 |
| print.fpca | 24 |
| rMaxHeuristic | 25 |
| spatialInference | 26 |
| Index | 29 |

.dfToppp

Convert SpatialExperiment object to ppp object

Description

Convert SpatialExperiment object to ppp object

Usage

```
.dfToppp(df, marks = NULL, continuous = FALSE, window = NULL)
```

Arguments

| | |
|------------|--|
| df | A dataframe with the x and y coordinates from the corresponding SpatialExperiment and the ColData |
| marks | A vector of marks to be associated with the points, has to be either named 'cell_type' if you want to compare discrete celltypes or else continuous gene expression measurements are assumed as marks. |
| continuous | A boolean indicating whether the marks are continuous defaults to FALSE |
| window | An observation window of the point pattern of class owin. |

Value

A ppp object for use with spatstat functions

Examples

```
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
speSub <- subset(spe, , image_number == "138")
dfSub <- .speToDf(speSub)
pp <- .dfTopp(dfSub, marks = "cell_type")
```

.extractMetric *Compute a spatial metric on a SpatialExperiment object*

Description

A function that takes a SpatialExperiment object and computes a spatial statistics function as implemented in spatstat. The output is a spatstat object.

Usage

```
.extractMetric(  
  df,  
  selection,  
  fun,  
  marks = NULL,  
  rSeq = NULL,  
  by = NULL,  
  continuous = FALSE,  
  window = NULL,  
  ...  
)
```

Arguments

| | |
|-------------------------|---|
| <code>df</code> | A dataframe with the x and y coordinates from the corresponding <code>SpatialExperiment</code> and the <code>colData</code> |
| <code>selection</code> | the mark(s) you want to compare |
| <code>fun</code> | the <code>spatstat</code> function to compute on the point pattern object |
| <code>marks</code> | the marks to consider e.g. cell types |
| <code>rSeq</code> | the range of r values to compute the function over |
| <code>by</code> | the <code>spe colData</code> variable(s) to add to the meta data |
| <code>continuous</code> | A boolean indicating whether the marks are continuous defaults to <code>FALSE</code> |
| <code>window</code> | a observation window for the point pattern of class <code>owin</code> . |
| <code>...</code> | Other parameters passed to <code>spatstat.explore</code> functions |

Value

a `spatstat` metric object with the fov number, the number of points and the centroid of the image

Examples

```
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
speSub <- subset(spe, , image_number == "138")
dfSub <- .speToDf(speSub)
metricRes <- .extractMetric(dfSub, c("alpha", "Tc"),
  fun = "Gcross",
  marks = "cell_type", rSeq = seq(0, 50, length.out = 50),
  by = c("patient_stage", "patient_id", "image_number")
)
```

```
.loadExample
```

```
load Example dataset from Damond et al. (2019)
```

Description

load Example dataset from Damond et al. (2019)

Usage

```
.loadExample(full = FALSE)
```

Arguments

| | |
|-------------------|---|
| <code>full</code> | a boolean indicating whether to load the entire Damond et al. (2019) or only a subset |
|-------------------|---|

Value

A `SpatialExperiment` object as uploaded to `ExperimentHub()`

Examples

```
# retrieve the Damond et al. (2019) dataset
spe <- .loadExample()
```

| | |
|-----------------------|--|
| <code>.speToDf</code> | <i>Transform a <code>SpatialExperiment</code> into a dataframe</i> |
|-----------------------|--|

Description

Transform a `SpatialExperiment` into a dataframe

Usage

```
.speToDf(spe)
```

Arguments

`spe` A `SpatialExperiment` object subset to a single image

Value

A dataframe with the x and y coordinates from the corresponding `SpatialExperiment` and the col-Data

Examples

```
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
speSub <- subset(spe, , image_number == "138")
dfSub <- .speToDf(speSub)
```

calcCrossMetricPerFov *Calculate cross spatial metrics for all combinations per FOV*

Description

A function that takes a `SpatialExperiment` object as input and calculates a cross spatial metric as implemented by `spatstat` per field of view for all combinations provided by the user.

Usage

```
calcCrossMetricPerFov(
  spe,
  selection,
  subsetby = NULL,
  fun,
  marks = NULL,
  rSeq = NULL,
  by = NULL,
  ncores = 1,
  continuous = FALSE,
  assay = "exprs",
  ...
)
```

Arguments

| | |
|-------------------------|--|
| <code>spe</code> | a <code>SpatialExperiment</code> object |
| <code>selection</code> | the mark(s) you want to compare |
| <code>subsetby</code> | the <code>spe colData</code> variable to subset the data by |
| <code>fun</code> | the <code>spatstat</code> function to compute on the point pattern object |
| <code>marks</code> | the marks to consider e.g. cell types |
| <code>rSeq</code> | the range of <code>r</code> values to compute the function over |
| <code>by</code> | the <code>spe colData</code> variable(s) to add to the meta data |
| <code>ncores</code> | the number of cores to use for parallel processing, default = 1 |
| <code>continuous</code> | A boolean indicating whether the marks are continuous defaults to <code>FALSE</code> |
| <code>assay</code> | the assay which is used if <code>continuous = TRUE</code> |
| <code>...</code> | Other parameters passed to <code>spatstat.explore</code> functions |

Value

a dataframe of the `spatstat` metric objects with the radius `r`, the theoretical value of a Poisson process, the different border corrections the `fov` number, the number of points and the centroid of the image

Examples

```
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
metricRes <- calcCrossMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross", marks = "cell_type",
  rSeq = seq(0, 50, length.out = 50), by = c(
    "patient_stage", "patient_id",
    "image_number"
  ),
  ncores = 1
)
```

| | |
|------------------|---|
| calcMetricPerFov | <i>Calculate a spatial metric on a SpatialExperiment object per field of view</i> |
|------------------|---|

Description

A function that takes a `SpatialExperiment` object as input and calculates a spatial metric as implemented by `spatstat` per field of view.

Usage

```
calcMetricPerFov(
  spe,
  selection,
  subsetby,
  fun,
  marks = NULL,
  rSeq = NULL,
  by = NULL,
  continuous = FALSE,
  assay = "exprs",
  ncores = 1,
  verbose = TRUE,
  ...
)
```

Arguments

| | |
|-----------|--|
| spe | a <code>SpatialExperiment</code> object |
| selection | the mark(s) you want to compare. NOTE: This is directional. <code>c(A,B)</code> is not the same result as <code>c(B,A)</code> . |
| subsetby | the <code>spe colData</code> variable to subset the data by. This variable has to be provided, even if there is only one sample. |
| fun | the <code>spatstat</code> function to compute on the point pattern object |

| | |
|------------|---|
| marks | the marks to consider e.g. cell types |
| rSeq | the range of r values to compute the function over |
| by | the spe colData variable(s) to add to the meta data |
| continuous | A boolean indicating whether the marks are continuous defaults to FALSE |
| assay | the assay which is used if continuous = TRUE |
| ncores | the number of cores to use for parallel processing, default = 1 |
| verbose | logical indicating whether to print all information or not |
| ... | Other parameters passed to spatstat.explore functions |

Value

a dataframe of the spatstat metric objects with the radius r, the theoretical value of a Poisson process, the different border corrections the fov number, the number of points and the centroid of the image

Examples

```
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
metricRes <- calcMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross", marks = "cell_type",
  rSeq = seq(0, 50, length.out = 50), by = c(
    "patient_stage", "patient_id",
    "image_number"
  ),
  ncores = 1
)
```

crossSpatialInference *Function for Cross Spatial Inference*

Description

This function is a wrapper function around spatialInference. It calculates spatialInference results either for all cell types in marks (if selection == NULL) or for a custom subset defined in selection.

Usage

```
crossSpatialInference(
  spe,
  selection = NULL,
  fun,
  marks = NULL,
  rSeq = NULL,
  correction,
```

```

    sample_id,
    image_id,
    condition,
    continuous = FALSE,
    assay = "exprs",
    transformation = NULL,
    eps = NULL,
    delta = 0,
    family = stats::gaussian(link = "log"),
    verbose = TRUE,
    ncores = 1,
    ...
)

```

Arguments

| | |
|----------------|--|
| spe | a SpatialExperiment object |
| selection | the mark(s) you want to compare. NOTE: This is directional. c(A,B) is not the same result as c(B,A). |
| fun | the spatstat function to compute on the point pattern object |
| marks | the marks to consider e.g. cell types |
| rSeq | the range of r values to compute the function over |
| correction | the edge correction to be applied |
| sample_id | the spe colData variable to mark the sample, if not NULL this will result in a mixed model estimation |
| image_id | the spe colData variable to mark the image |
| condition | the spe colData variable to mark the condition |
| continuous | A boolean indicating whether the marks are continuous defaults to FALSE |
| assay | the assay which is used if continuous = TRUE |
| transformation | the transformation to be applied as exponential e.g. 1/2 for sqrt |
| eps | some distributional families fail if the response is zero, therefore, zeros can be replaced with a very small value eps |
| delta | the delta value to remove from the beginning of the spatial statistics functions. Can be reasonable if e.g. cells are always spaced by 10 μm . If set to "minNnDist" it will take the mean of the minimum nearest neighbour distance across all images for this cell type pair. |
| family | the distributional family for the functional GAM |
| verbose | logical indicating whether to print all information or not |
| ncores | the number of cores to use for parallel processing, default = 1 |
| ... | Other parameters passed to spatstat.explore functions for parameters concerning the spatial function calculation and to refund::pffr for the functional additive mixed model inference |

Value

a list of objects created by the function `spatialInference` with three objects: i) the dataframe with the spatial statistics results, ii) the designmatrix of the inference and iii) the fitted pffr object

Examples

```
spe <- .loadExample()
#make the condition a factor variable
colData(spe)[["patient_stage"]] <- factor(colData(spe)[["patient_stage"]])
#relevel to have non-diabetic as the reference category
colData(spe)[["patient_stage"]] <- relevel(colData(spe)[["patient_stage"]],
"Non-diabetic")

selection <- c("acinar", "ductal")
resLs <- crossSpatialInference(spe, selection, fun = "Gcross",
                              marks = "cell_type", rSeq = seq(0, 50, length.out = 50),
                              correction = "rs", sample_id = "patient_id",
                              image_id = "image_number", condition = "patient_stage",
                              algorithm = "bam",
                              ncores = 1
                              )
```

`extractCrossInferenceData`

Reshaping the Result of a Cross Spatial Inference to a Dataframe

Description

Reshaping the Result of a Cross Spatial Inference to a Dataframe

Usage

```
extractCrossInferenceData(
  resLs,
  QCMetric = "medianMinIntensity",
  QCThreshold = 0.1
)
```

Arguments

| | |
|--------------------------|---|
| <code>resLs</code> | a list with four objects: i) the dataframe with the spatial statistics results transformed and filtered as used for fitting, ii) the raw spatial statistics results, iii) the designmatrix of the inference and iv) the fitted pffr object v) the residual standard error per condition defined as the residual sum of squares divided by the number of datapoints - sum of the estimated degrees of freedom for the model parameters as well as other QC metrics |
| <code>QCMetric</code> | the metric to relate the quality of the fit too. |
| <code>QCThreshold</code> | the threshold on the QC metric. Depends on the function used. |

Value

a dataframe for plotting with ggplot2

Examples

```
spe <- .loadExample()
#make the condition a factor variable
colData(spe)[["patient_stage"]] <- factor(colData(spe)[["patient_stage"]])
#relevel to have non-diabetic as the reference category
colData(spe)[["patient_stage"]] <- relevel(colData(spe)[["patient_stage"]],
"Non-diabetic")

selection <- c("acinar", "ductal")
resLs <- crossSpatialInference(spe, selection,
                             fun = "Gcross", marks = "cell_type",
                             rSeq = seq(0, 50, length.out = 50), correction = "rs",
                             sample_id = "patient_id",
                             image_id = "image_number", condition = "patient_stage",
                             algorithm = "bam",
                             ncores = 1
                             )
df <- extractCrossInferenceData(resLs)
```

functionalGam

General additive model with functional response

Description

A function that takes the output of a metric calculation as done by calcMetricPerFov. The data has to be prepared into the correct format for the functional analysis by the prepData function. The output is a pffr object as implemented by refund.

Usage

```
functionalGam(
  data,
  x,
  designmat,
  weights,
  formula,
  family = stats::gaussian(link = "log"),
  H = NULL,
  ...
)
```

Arguments

| | |
|------------------------|--|
| <code>data</code> | a dataframe with the following columns: <code>Y</code> = functional response; <code>sample_id</code> = sample ID; <code>image_id</code> = image ID; |
| <code>x</code> | the x-axis values of the functional response |
| <code>designmat</code> | a design matrix as defined by <code>model.matrix()</code> |
| <code>weights</code> | weights as the number of points per image. These weights are normalised by the mean of the weights in the fitting process |
| <code>formula</code> | the formula for the model. The colnames of the designmatrix have to correspond to the variables in the formula. |
| <code>family</code> | the distributional family as implemented in <code>family.mgcv</code> . For fast computation the default is set to gaussian with a log link. other interesting options can be <code>betar</code> and <code>scat</code> <ul style="list-style-type: none"> • for more information see <code>family.mgcv</code>. |
| <code>H</code> | the ridge penalty matrix passed to <code>mgcv::gam</code> |
| <code>...</code> | Other parameters passed to <code>pfpr</code> |

Value

a fitted `pfpr` object which inherits from `gam`

Examples

```
# load the pancreas dataset
library("tidyr")
library("dplyr")
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
# calculate the Gcross metric for alpha and Tc cells
metricRes <- calcMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross",
  marks = "cell_type", rSeq = seq(0, 50, length.out = 50),
  c("patient_stage", "patient_id", "image_number"), ncores = 1
)
metricRes$ID <- paste0(
  metricRes$patient_stage, "|", metricRes$patient_id,
  "|", metricRes$image_number
)
dat <- prepData(metricRes, "r", "rs", sample_id = "patient_id",
  image_id = "image_number", condition = "patient_stage")

#' # drop rows with NA
dat <- dat |> drop_na()

# create a designmatrix
condition <- dat$patient_stage
# relevel the condition - can set explicit contrasts here
condition <- relevel(condition, "Non-diabetic")
designmat <- model.matrix(~condition)
```

```

# colnames don't work with the '-' sign
colnames(designmat) <- c(
  "(Intercept)", "conditionLong_duration",
  "conditionOnset"
)
# fit the model
mdl <- functionalGam(
  data = dat, x = metricRes$r |> unique(),
  designmat = designmat, weights = dat$npoints,
  formula = formula(Y ~ conditionLong_duration +
    conditionOnset + s(patient_id, bs = "re")),
  algorithm = "bam"
)
summary(mdl)

```

functionalPCA

Functional Principal Component Analysis

Description

A function that takes as input the output of `calcMetricPerFov` which has to be converted into the correct format by `prepData`. The output is a list with the `fpca.face` output from `refund`.

Usage

```
functionalPCA(data, r, ...)
```

Arguments

| | |
|-------------------|--|
| <code>data</code> | a data object for functional data analysis containing at least the functional response <code>Y_{ij}</code> . |
| <code>r</code> | the functional domain |
| <code>...</code> | Other parameters passed to <code>fpca.sc</code> functions |

Value

a list with components of `fpca.sc`

Examples

```

# load the pancreas dataset
library("tidyr")
library("stringr")
library("dplyr")
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
# calculate the Gcross metric for alpha and Tc cells
metricRes <- calcMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross",

```

```

marks = "cell_type", rSeq = seq(0, 50, length.out = 50),
c("patient_stage", "patient_id", "image_number"), ncores = 1
)
metricRes$ID <- paste0(
  metricRes$patient_stage, "|", metricRes$patient_id,
  "|", metricRes$image_number
)
# prepare data for FDA
dat <- prepData(metricRes, "r", "rs")

# drop rows with NA
dat <- dat |> drop_na()
# create meta info of the IDs
splitData <- str_split(dat$ID, "x")
dat$condition <- factor(sapply(splitData, function(x) x[1]))
dat$patient_id <- factor(sapply(splitData, function(x) x[2]))
dat$image_id <- factor(sapply(splitData, function(x) x[3]))
# calculate fPCA
mdl <- functionalPCA(
  data = dat, r = metricRes$r |> unique()
)

```

plotCrossFOV

Creates a $n \times n$ plot of the cross metrics per sample

Description

Helper function for plotCrossMetricPerFov. It applies plotMetricPerFov to all n marks defined in the variable selection. This gives an $n \times n$ plot of all marks.

Usage

```

plotCrossFOV(
  subFov,
  theo,
  correction,
  x,
  imageId,
  ID = NULL,
  ncol = NULL,
  nrow = NULL,
  legend.position = "none",
  ...
)

```

Arguments

| | |
|--------|--|
| subFov | a subset of the dataframe to the respective fov |
| theo | logical; if the theoretical line should be plotted |

| | |
|-----------------|--|
| correction | the border correction to plot |
| x | the x-axis variable to plot |
| imageId | the ID of the image/fov |
| ID | the (optional) ID for plotting combinations |
| ncol | the number of columns for the facet wrap |
| nrow | the number of rows for the facet wrap |
| legend.position | the position of the legend of the plot |
| ... | Other parameters passed to ggplot2 functions |

Value

a ggplot object

plotCrossHeatmap *Plotting the Result of a Cross Spatial Inference*

Description

Plotting the Result of a Cross Spatial Inference

Usage

```
plotCrossHeatmap(
  resLs,
  adj.pvalue = "BH",
  coefficientsToPlot = NULL,
  QCThreshold = 1e-05,
  QCMetric = "medianMinIntensity",
  ...
)
```

Arguments

| | |
|--------------------|---|
| resLs | a list with four objects: i) the dataframe with the spatial statistics results transformed and filtered as used for fitting, ii) the raw spatial statistics results, iii) the designmatrix of the inference and iv) the fitted pffr object v) the residual standard error per condition defined as the residual sum of squares divided by the number of datapoints - sum of the estimated degrees of freedom for the model parameters as well as other QC metrics |
| adj.pvalue | a pvalue adjustment method as passed to stats::p.adjust defaults to Benjamini-Hochberg correction of the false discovery rate. |
| coefficientsToPlot | list of which coefficients to plot in the heatmap defaults to NULL in which case all coefficients are plotted |

QCThreshold the threshold on the Quality control metric. Depends on the function used.
 QCMetric the metric to relate the quality of the fit too.
 ... other parameters passed to ggplot2 functions

Value

a ggplot2 object

Examples

```
spe <- .loadExample()
#make the condition a factor variable
colData(spe)[["patient_stage"]] <- factor(colData(spe)[["patient_stage"]])
#relevel to have non-diabetic as the reference category
colData(spe)[["patient_stage"]] <- relevel(colData(spe)[["patient_stage"]],
"Non-diabetic")

selection <- c("acinar", "ductal")
resLs <- crossSpatialInference(spe, selection,
                             fun = "Gcross", marks = "cell_type",
                             rSeq = seq(0, 50, length.out = 50), correction = "rs",
                             sample_id = "patient_id",
                             image_id = "image_number", condition = "patient_stage",
                             algorithm = "bam",
                             ncores = 1
                             )
p <- plotCrossHeatmap(resLs, adj.pvalue = "BH")
```

plotCrossMetricPerFov *Plot a cross type spatial metric per field of view*

Description

This function plots the cross function between two marks output from calcMetricPerFov. It wraps around helper function and applies this function to all samples.

Usage

```
plotCrossMetricPerFov(
  metricDf,
  theo = NULL,
  correction = NULL,
  x = NULL,
  imageId = NULL,
  ID = NULL,
  nrow = NULL,
  ncol = NULL,
```

```

    legend.position = "none",
    ...
  )

```

Arguments

| | |
|-----------------|--|
| metricDf | the metric dataframe as calculated by calcMetricPerFov |
| theo | logical; if the theoretical line should be plotted |
| correction | the border correction to plot |
| x | the x-axis variable to plot |
| imageId | the ID of the image/fov |
| ID | the (optional) ID for plotting combinations |
| nrow | the number of rows for the facet wrap |
| ncol | the number of columns for the facet wrap |
| legend.position | the position of the legend of the plot |
| ... | Other parameters passed to ggplot2 functions |

Value

a ggplot object

Examples

```

# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
metricRes <- calcCrossMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross", marks = "cell_type",
  rSeq = seq(0, 50, length.out = 50), by = c(
    "patient_stage", "patient_id",
    "image_number"
  ),
  ncores = 1
)

metricRes$ID <- paste0(
  metricRes$patient_stage, "|", metricRes$patient_id
)

metricRes <- subset(metricRes, image_number %in% c(138, 139, 140))
p <- plotCrossMetricPerFov(metricRes,
  theo = TRUE, correction = "rs",
  x = "r", imageId = "image_number", ID = "ID"
)
print(p)

```

plotFbPlot

Functional boxplot of spatstat curves

Description

This function creates a functional boxplot of the spatial statistics curves. It creates one functional boxplot per aggregation category, e.g. condition.

Usage

```
plotFbPlot(metricDf, x, y, aggregateBy)
```

Arguments

| | |
|-------------|---|
| metricDf | the metric dataframe as calculated by calcMetricPerFov |
| x | the name of the x-axis of the spatial metric |
| y | the name of the y-axis of the spatial metric |
| aggregateBy | the criterion by which to aggregate the curves into a functional boxplot. Can be e.g. the condition of the different samples. |

Value

a list of base R plots

Examples

```
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
metricRes <- calcMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross", marks = "cell_type",
  rSeq = seq(0, 50, length.out = 50), by = c(
    "patient_stage", "patient_id",
    "image_number"
  ),
  ncores = 1
)
# create a unique ID for the data preparation
metricRes$ID <- paste0(
  metricRes$patient_stage, "|", metricRes$patient_id,
  "|", metricRes$image_number
)
plotFbPlot(metricRes, 'r', 'rs', 'patient_stage')
```

| | |
|----------|--|
| plotFpca | <i>Plot a biplot from an fPCA analysis</i> |
|----------|--|

Description

A function that takes the output from the functionalPCA function and returns a ggplot object of the first two dimensions of the PCA as biplot.

Usage

```
plotFpca(data, res, colourby = NULL, labelby = NULL)
```

Arguments

| | |
|----------|--|
| data | a data object for functional data analysis containing at least the functional response Y . |
| res | the output from the fPCA calculation |
| colourby | the variable by which to colour the PCA plot by |
| labelby | the variable by which to label the PCA plot by |

Value

a list with components of fpca.face

Examples

```
# load the pancreas dataset
library("tidyr")
library("stringr")
library("dplyr")
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
# calculate the Gcross metric for alpha and beta cells
metricRes <- calcMetricPerFov(spe, c("alpha", "beta"),
  subsetby = "image_number", fun = "Gcross",
  marks = "cell_type", rSeq = seq(0, 50, length.out = 50),
  c("patient_stage", "patient_id", "image_number"), ncores = 1
)
metricRes$ID <- paste0(
  metricRes$patient_stage, "|", metricRes$patient_id,
  "|", metricRes$image_number
)

# prepare data for FDA
dat <- prepData(metricRes, "r", "rs")

# drop rows with NA
dat <- dat |> drop_na()
```

```

# create meta info of the IDs
splitData <- str_split(dat$ID, "|")
dat$condition <- factor(sapply(splitData, function(x) x[1]))
dat$patient_id <- factor(sapply(splitData, function(x) x[2]))
dat$image_id <- factor(sapply(splitData, function(x) x[3]))
# calculate fPCA
mdl <- functionalPCA(
  data = dat, r = metricRes$r |> unique()
)
p <- plotFpca(
  data = dat, res = mdl, colourby = "condition",
  labelby = "patient_id"
)
print(p)

```

plotMdl

Plot a pffr model object

Description

A function that takes a pffr object as calculated in functionalGam and plots the functional coefficients. The functions are centered such that their expected value is zero. Therefore, the scalar intercept has to be added to the output with the argument shift in order to plot the coefficients in their original range.

Usage

```
plotMdl(mdl, predictor, shift = NULL)
```

Arguments

| | |
|-----------|---|
| mdl | a pffr model object |
| predictor | predictor to plot |
| shift | the value by which to shift the centered functional intercept. this will most often be the constant intercept |

Value

ggplot object of the functional estimate

Examples

```

library("tidyr")
library("stringr")
library("dplyr")
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
metricRes <- calcMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross", marks = "cell_type",

```

```

    rSeq = seq(0, 50, length.out = 50), by = c(
      "patient_stage", "patient_id",
      "image_number"
    ),
    ncores = 1
  )
  # create a unique ID for each row
  metricRes$ID <- paste0(
    metricRes$patient_stage, "x", metricRes$patient_id,
    "x", metricRes$image_number
  )

  dat <- prepData(metricRes, "r", "rs", sample_id = "patient_id",
    image_id = "image_number", condition = "patient_stage")

  #' # drop rows with NA
  dat <- dat |> drop_na()

  # create a designmatrix
  condition <- dat$patient_stage
  # relevel the condition - can set explicit contrasts here
  condition <- relevel(condition, "Non-diabetic")
  designmat <- model.matrix(~condition)
  # colnames don't work with the '-' sign
  colnames(designmat) <- c(
    "(Intercept)", "conditionLong_duration",
    "conditionOnset"
  )
  # fit the model
  mdl <- functionalGam(
    data = dat, x = metricRes$r |> unique(),
    designmat = designmat, weights = dat$npoints,
    formula = formula(Y ~ conditionLong_duration +
      conditionOnset + s(patient_id, bs = "re")),
    algorithm = "bam"
  )
  summary(mdl)
  plots <- lapply(colnames(designmat), plotMdl,
    mdl = mdl,
    shift = mdl$coefficients[["(Intercept)"]]
  )

```

plotMetricPerFov

Plot a spatial metric per field of view

Description

A function that plots the output of the function calcMetricPerFov. The plot contains one curve per FOV and makes subplots by samples.

Usage

```
plotMetricPerFov(
  metricDf,
  theo = FALSE,
  correction = NULL,
  x = NULL,
  imageId = NULL,
  ID = NULL,
  nrow = NULL,
  ncol = NULL,
  legend.position = "none",
  ...
)
```

Arguments

| | |
|-----------------|--|
| metricDf | the metric dataframe as calculated by calcMetricPerFov |
| theo | logical; if the theoretical line should be plotted |
| correction | the border correction to plot |
| x | the x-axis variable to plot |
| imageId | the ID of the image/fov |
| ID | the (optional) ID for plotting combinations |
| nrow | the number of rows for the facet wrap |
| ncol | the number of columns for the facet wrap |
| legend.position | the position of the legend of the plot |
| ... | Other parameters passed to ggplot2 functions |

Value

a ggplot object

Examples

```
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
metricRes <- calcMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross", marks = "cell_type",
  rSeq = seq(0, 50, length.out = 50), by = c(
    "patient_stage", "patient_id",
    "image_number"
  ),
  ncores = 1
)
# ceate a unique plotting ID
metricRes$ID <- paste0(
  metricRes$patient_stage, "|", metricRes$patient_id
```

```

)

p <- plotMetricPerFov(metricRes,
  correction = "rs", x = "r",
  imageId = "image_number", ID = "ID"
)
print(p)

```

```
prepData
```

Prepare data from calcMetricRes to be in the right format for FDA

Description

Prepare data from calcMetricRes to be in the right format for FDA

Usage

```
prepData(metricRes, x, y, sample_id = NULL, image_id = NULL, condition = NULL)
```

Arguments

| | |
|-----------|--|
| metricRes | a dataframe as calculated by calcMetricRes - requires the columns ID (unique identifier of each row) |
| x | the name of the x-axis of the spatial metric |
| y | the name of the y-axis of the spatial metric |
| sample_id | the spe colData variable to mark the sample |
| image_id | the spe colData variable to mark the image |
| condition | the spe colData variable to mark the condition |

Value

returns a list with three entries, the unique ID, the functional response Y and the weights

Examples

```

# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
metricRes <- calcMetricPerFov(spe, c("alpha", "Tc"),
  subsetby = "image_number", fun = "Gcross", marks = "cell_type",
  rSeq = seq(0, 50, length.out = 50), by = c(
    "patient_stage", "patient_id",
    "image_number"
  ),
  ncores = 1
)

# create a unique ID for each row
metricRes$ID <- paste0(

```

```

    metricRes$patient_stage, "|", metricRes$patient_id,
    "|", metricRes$image_number
  )
  dat <- prepData(metricRes, "r", "rs", sample_id = "patient_id",
    image_id = "image_number", condition = "patient_stage")

```

```
print.fpca
```

```
print the fPCA results
```

Description

this is a function that prints a summary of the fPCA result of class fpca

Usage

```
## S3 method for class 'fpca'
print(x, ...)
```

Arguments

```

x          the result of function functionalPCA
...       other parameters passed to base generic function print

```

Value

a formatted overview of the fPCA result

Examples

```

# load the pancreas dataset
library("tidyr")
library("stringr")
library("dplyr")
# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
# calculate the Gcross metric for alpha and beta cells
metricRes <- calcMetricPerFov(spe, c("alpha", "beta"),
  subsetby = "image_number", fun = "Gcross",
  marks = "cell_type", rSeq = seq(0, 50, length.out = 50),
  c("patient_stage", "patient_id", "image_number"), ncores = 1
)
metricRes$ID <- paste0(
  metricRes$patient_stage, "|", metricRes$patient_id,
  "|", metricRes$image_number
)
# prepare data for FDA
dat <- prepData(metricRes, "r", "rs")

# drop rows with NA

```

```

dat <- dat |> drop_na()

# create meta info of the IDs
splitData <- strsplit(dat$ID, "|", fixed = TRUE)
dat$condition <- factor(sapply(splitData, function(x) x[1]))
dat$patient_id <- factor(sapply(splitData, function(x) x[2]))
dat$image_id <- factor(sapply(splitData, function(x) x[3]))
# calculate fPCA
mdl <- functionalPCA(
  data = dat, r = metricRes$r |> unique()
)
mdl

```

rMaxHeuristic

Heuristic for the choice of rMax

Description

Heuristic for the choice of rMax

Usage

```
rMaxHeuristic(spe, subsetby, marks)
```

Arguments

| | |
|----------|---|
| spe | a SpatialExperiment object |
| subsetby | the spe colData variable to subset the data by. This variable has to be provided, even if there is only one sample. |
| marks | the marks to consider e.g. cell types |

Value

a ggplot histogram of the bounding radius of all the

Examples

```

# retrieve example data from Damond et al. (2019)
spe <- .loadExample()
p <- rMaxHeuristic(spe,
  subsetby = "image_number", marks = "cell_type"
)

```

Description

A function to perform spatial statistical inference on spatial omics data. This function works so far only on functions of radius "r".

Usage

```
spatialInference(
  spe,
  selection,
  fun,
  marks = NULL,
  rSeq = NULL,
  correction,
  sample_id,
  image_id,
  condition,
  continuous = FALSE,
  assay = "exprs",
  transformation = NULL,
  weights = "total",
  eps = NULL,
  delta = 0,
  family = stats::gaussian(link = "log"),
  verbose = TRUE,
  ridgepenalty = 0,
  upperDeltaProb = NULL,
  weightTransform = FALSE,
  ncores = 1,
  ...
)
```

Arguments

| | |
|------------|--|
| spe | a SpatialExperiment object |
| selection | the mark(s) you want to compare. NOTE: This is directional. c(A,B) is not the same result as c(B,A). |
| fun | the spatstat function to compute on the point pattern object |
| marks | the marks to consider e.g. cell types |
| rSeq | the range of r values to compute the function over |
| correction | the edge correction to be applied |

| | |
|-----------------|--|
| sample_id | the spe colData variable to mark the sample, if not NULL this will result in a mixed model estimation |
| image_id | the spe colData variable to mark the image |
| condition | the spe colData variable to mark the condition |
| continuous | A boolean indicating whether the marks are continuous defaults to FALSE |
| assay | the assay which is used if continuous = TRUE |
| transformation | the transformation to be applied as exponential e.g. 1/2 for sqrt or Fisher's variance-stabilising transformation if "Fisher" |
| weights | the weighting to be applied to the functional GAM. Either NULL (equal weights), total (npoints of total pattern), min (npoints of the smaller subpattern) or max (npoints of the larger subpattern) or a user defined value of same length as the number of curves to be estimated |
| eps | some distributional families fail if the response is zero, therefore, zeros can be replaced with a very small value eps |
| delta | the delta value to remove from the beginning of the spatial statistics functions. Can be reasonable if e.g. cells are always spaced by 10 μm . If set to "minNnDist" it will take the mean of the minimum nearest neighbour distance across all images for this cell type pair. |
| family | the distributional family for the functional GAM |
| verbose | logical indicating whether to print all information or not |
| ridgepenalty | a numeric value defining a ridge penalty parameter which is added to the matrix H as defined in mgcv: : gam |
| upperDeltaProb | the quantile to filter out the constant 1 part for Gest and Gcross. If NULL no upper filtering is applied. |
| weightTransform | logical indicating whether the weights (number of points) should be sqrt transformed |
| ncores | the number of cores to use for parallel processing, default = 1 |
| ... | Other parameters passed to spatstat.explore functions for parameters concerning the spatial function calculation and to refund: : pffr for the functional additive mixed model inference |

Value

a list with four objects: i) the dataframe with the spatial statistics results transformed and filtered as used for fitting, ii) the raw spatial statistics results, iii) the designmatrix of the inference and iv) the fitted pffr object v) the residual standard error per condition defined as the residual sum of squares divided by the number of datapoints - sum of the estimated degrees of freedom for the model parameters as well as other QC metrics

Examples

```
spe <- .loadExample()
#make the condition a factor variable
colData(spe)[["patient_stage"]] <- factor(colData(spe)[["patient_stage"]])
```

```
#relevel to have non-diabetic as the reference category
colData(spe)[["patient_stage"]] <- relevel(colData(spe)[["patient_stage"]],
"Non-diabetic")
res <- spatialInference(spe, c("alpha", "Tc"),
  fun = "Gcross", marks = "cell_type",
  rSeq = seq(0, 50, length.out = 50), correction = "rs",
  sample_id = "patient_id",
  image_id = "image_number", condition = "patient_stage",
  ncores = 1,
  algorithm = "bam"
)
```

Index

[.dfToppp](#), 2
[.extractMetric](#), 3
[.loadExample](#), 4
[.speToDf](#), 5

[calcCrossMetricPerFov](#), 6
[calcMetricPerFov](#), 7
[crossSpatialInference](#), 8

[extractCrossInferenceData](#), 10

[functionalGam](#), 11
[functionalPCA](#), 13

[plotCrossFOV](#), 14
[plotCrossHeatmap](#), 15
[plotCrossMetricPerFov](#), 16
[plotFbPlot](#), 18
[plotFpca](#), 19
[plotMdl](#), 20
[plotMetricPerFov](#), 21
[prepData](#), 23
[print.fpca](#), 24

[rMaxHeuristic](#), 25

[spatialInference](#), 26