Package ‘spatialFDA’

July 26, 2025

Title A Tool for Spatial Multi-sample Comparisons

Version 1.1.3

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 7.3.2

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

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 5e8d7ce

git_last_commit_date 2025-07-25

Repository Bioconductor 3.22

Date/Publication 2025-07-25

Author Martin Emons [aut, cre] (ORCID:
<https://orcid.org/0009-0000-5219-5311>),
Samuel Gunz [aut] (ORCID: <https://orcid.org/0000-0002-8909-0932>),
Mark D. Robinson [aut, fnd] (ORCID:
<https://orcid.org/0000-0002-3048-5518>)

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

Contents

`.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 continous 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 <- .dfToppp(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

`df`	A `dataframe` with the x and y coordinates from the corresponding `SpatialExperiment` and the `colData`
`selection`	the mark(s) you want to compare
`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
`by`	the spe `colData` variable(s) to add to the meta data
`continuous`	A boolean indicating whether the marks are continuous defaults to FALSE
`window`	a observation window for the point pattern of class `owin`.
`...`	Other parameters passed to `spatstat.explore` 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", "beta"),
    fun = "Gcross",
    marks = "cell_type", rSeq = seq(0, 1000, length.out = 100),
    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

`full`	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()
```

`.speToDf` *Transform a SpatialExperiment into a dataframe*

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 colData

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

`spe`	a `SpatialExperiment` object
`selection`	the mark(s) you want to compare
`subsetby`	the spe `colData` variable to subset the data by
`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
`by`	the spe `colData` variable(s) to add to the meta data
`ncores`	the number of cores to use for parallel processing, default = 1
`continuous`	A boolean indicating whether the marks are continuous defaults to FALSE
`assay`	the assay which is used if `continuous = TRUE`
`...`	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 <- calcCrossMetricPerFov(spe, c("alpha", "beta"),
    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` *Calculate a spatial metric on a* `SpatialExperiment` *object per field of view*

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,
  ...
)
```

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).
`subsetby`	the spe `colData` variable to subset the data by. This variable has to be provided, even if there is only one sample.

`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
`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
`...`	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", "beta"),
    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 wrappere 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,
  subsetby,
  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"),
  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).
`subsetby`	the spe `colData` variable to subset the data by. This variable has to be provided, even if there is only one sample.
`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.
`family`	the distributional family for the functional GAM
`ncores`	the number of cores to use for parallel processing, default = 1
`...`	Other parameters passed to `spatstat.explore` functions

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,
                     subsetby = "image_number", 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
                  )
```

`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"),
  ...
)
```

Arguments

`data`	a dataframe with the following columns: Y = functional response; sample_id = sample ID; image_id = image ID;
`x`	the x-axis values of the functional response
`designmat`	a design matrix as defined by model.matrix()
`weights`	weights as the number of points per image. These weights are normalised by the mean of the weights in the fitting process
`formula`	the formula for the model. The colnames of the designmatrix have to correspond to the variables in the formula.
`family`	the distributional family as implemented in `family.mgcv`. For fast computation the default is set to `gaussian` with a log link. other interesting options can be `betar` and `scat` • for more information see `family.mgcv`.
`...`	Other parameters passed to `pffr`

Value

a fitted pffr 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 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
)
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"))
)
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

`data`	a data object for functional data analysis containing at least the functional response Y.
`r`	the functional domain
`...`	Other parameters passed to `fpca.sc` 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 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$ID, "x")
dat$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 nXn 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 `nxn` 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

`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", "beta"),
    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", "beta"),
    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` *Plot a biplot from an fPCA analysis*

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", "beta"),
    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$npoints,
    formula = formula(Y ~ conditionLong_duration +
        conditionOnset + s(patient_id, bs = "re"))
)
summary(mdl)
plotLs <- lapply(colnames(designmat), plotMdl,
    mdl = mdl,
    shift = mdl$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", "beta"),
    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", "beta"),
    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
```

`spatialInference` *Statistical Inference on Spatial Statistics Functions*

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,
  subsetby,
  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"),
    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).
`subsetby`	the spe `colData` variable to subset the data by. This variable has to be provided, even if there is only one sample.
`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.
`family`	the distributional family for the functional GAM
`ncores`	the number of cores to use for parallel processing, default = 1
`...`	Other parameters passed to `spatstat.explore` functions

Value

a list 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")
res <- spatialInference(spe, c("alpha", "beta"),
    subsetby = "image_number", 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
)
```

Index