Package ‘mxfda’

October 7, 2024

Title A Functional Data Analysis Package for Spatial Single Cell Data

Version 0.2.2

Date 2024-10-07

Description Methods and tools for deriving spatial summary functions from single-cell imaging data and performing functional data analyses. Functions can be applied to other single-cell technologies such as spatial transcriptomics. Functional regression and functional principal component analysis methods
are in the 'refund' package <`https://cran.r-project.org/package=refund`> while calculation of the
spatial summary functions are from the 'spatstat' package <`https://spatstat.org/`>.

License MIT + file LICENSE

URL `https://github.com/julia-wrobel/mxfda/`,
`http://juliawrobel.com/mxfda/`

BugReports `https://github.com/julia-wrobel/mxfda/issues/`

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0), tidyverse, survival, ggpubr, spatialTIME, tibble, broom, refund.shiny, Seurat, SeuratObject

Imports magrittr, rlang, tidyr, purrr, dplyr, ggplot2, lifecycle, methods, stats, refund (>= 0.1-35), reshape2, mgcv, spatstat.geom, spatstat.explore, SpatEntropy, SimDesign

Depends R (>= 2.10),

LazyData true

LazyDataCompression bzip2

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Julia Wrobel [aut] (<`https://orcid.org/0000-0001-6783-1421`>),
Alex Soupir [aut, cre] (<`https://orcid.org/0000-0003-1251-9179`>)

Maintainer Alex Soupir <`alex.soupir@moffitt.org`>

Repository CRAN

Date/Publication 2024-10-07 17:40:02 UTC

Contents

`add_summary_function` *Add Summary Function*

Description

Sometimes other ways of calculating summary functions is wanted and is done in other packages, in this instance the data can be loaded into the `mxFDA` object.

Usage

```
add_summary_function(mxFDAobject, summary_function_data, metric)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`summary_function_data`	data frame with `summary_key` from `mxFDA` object as key column for summary function
`metric`	character vector with either 'uni' or 'bi' and 'k', 'l', or 'g'; e.g. 'uni g'

Value

an updated `mxFDA` object with a derived value added. See `make_mxfda()` for more details.

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

`bivariate` *bivariate*

Description

Internal function called by `extract_summary_functions` to calculate a bivariate spatial summary function for a single image.

Usage

```
bivariate(
  mximg,
  markvar,
  mark1,
  mark2,
  r_vec,
  func = c(Kcross, Lcross, Gcross, entropy),
  edge_correction,
  empirical_CSR = FALSE,
  permutations = 1000
)
```

Arguments

`mximg`	Dataframe of cell-level multiplex imaging data for a single image. Should have variables `x` and `y` to denote x and y spatial locations of each cell.
`markvar`	The name of the variable that denotes cell type(s) of interest. Character.
`mark1`	Character string that denotes first cell type of interest.
`mark2`	Character string that denotes second cell type of interest.
`r_vec`	Numeric vector of radii over which to evaluate spatial summary functions. Must begin at 0.
`func`	Spatial summary function to calculate. Options are c(Kcross, Lcross, Gcross) which denote Ripley's K, Besag's L, and nearest neighbor G function, respectively, or entropy from Vu et al, 2023.
`edge_correction`	Character string that denotes the edge correction method for spatial summary function. For Kcross and Lcross choose one of c("border", "isotropic", "Ripley", "translate", "none"). For Gcross choose one of c("rs", "km", "han")
`empirical_CSR`	logical to indicate whether to use the permutations to identify the sample-specific complete spatial randomness (CSR) estimation.
`permutations`	integer for the number of permtuations to use if empirical_CSR is `TRUE` and exact CSR not calculable

Details

[Stable]

Value

A `data.frame` containing:

`r`	the radius of values over which the spatial summary function is evaluated
`sumfun`	the values of the spatial summary function
`csr`	the values of the spatial summary function under complete spatial randomness
`fundiff`	sumfun - csr, positive values indicate clustering and negative values repulsion

Author(s)

Julia Wrobel `<julia.wrobel@emory.edu>`

Alex Soupir `<alex.soupir@moffitt.org>`

References

Xiao, L., Ruppert, D., Zipunnikov, V., and Crainiceanu, C. (2016). Fast covariance estimation for high-dimensional functional data. *Statistics and Computing*, 26, 409-421. DOI: 10.1007/s11222-014-9485-x.

Vu, T., Seal, S., Ghosh, T., Ahmadian, M., Wrobel, J., & Ghosh, D. (2023). FunSpace: A functional and spatial analytic approach to cell imaging data using entropy measures. *PLOS Computational Biology*, 19(9), e1011490.

Creed, J. H., Wilson, C. M., Soupir, A. C., Colin-Leitzinger, C. M., Kimmel, G. J., Ospina, O. E., Chakiryan, N. H., Markowitz, J., Peres, L. C., Coghill, A., & Fridley, B. L. (2021). spatialTIME and iTIME: R package and Shiny application for visualization and analysis of immunofluorescence data. *Bioinformatics* (Oxford, England), 37(23), 4584–4586. https://doi.org/10.1093/bioinformatics/btab757

`entropy` *Entropy*

Description

Entropy

Usage

```
entropy(df, r_vec, markvar)
```

Arguments

`df`	data frame with x and y columns, along with a column for point marks
`r_vec`	vector of length wanted for breaks (will be rescaled) with max value at max for measuring entropy
`markvar`	The name of the variable that denotes cell type(s) of interest. Character.

Details

[Experimental]

Value

data frame with entropy calculated for `length(r_vec)` bins within 0 to `max(r_vec)`

Author(s)

Thao Vu <`thao.3.vu@cuanschutz.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

References

Vu, T., Seal, S., Ghosh, T., Ahmadian, M., Wrobel, J., & Ghosh, D. (2023). FunSpace: A functional and spatial analytic approach to cell imaging data using entropy measures. *PLOS Computational Biology*, 19(9), e1011490.

Altieri, L., Cocchi, D., & Roli, G. (2018). A new approach to spatial entropy measures. *Environmental and ecological statistics*, 25, 95-110.

`extract_entropy` *extract_entropy*

Description

The extract_entropy() is used to compute spatial entropy at each distance interval for all cell types of interest. The goal is to capture the diversity in cellular composition, such as similar proportions across cell types or dominance of a single type, at a specific distance range. Additionally, spatial patterns, including clustered, independent, or regular, among cell types can also be acquired. In this example, we will look at the spatial heterogeneity across T cells, macrophages, and others. To focus on the local cell-to-cell interactions, we set the default maximum of the distance range (i.e., rmax) to be 400 microns. The default number of distance breaks/intervals is set to 50. Then, a sequence of distance breaks is generated by linearly decreasing from rmax to 0 on a log scale. At each distance range, partial spatial entropy and residual entropy are calculated as in Vu et al. (2023), Altieri et al. (2018). These spatial entropy functions can then be used as input functions for FPCA.

Usage

```
extract_entropy(mxFDAobject, markvar, marks, n_break = 50, rmax = 400)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`markvar`	The name of the variable that denotes cell type(s) of interest. Character.
`marks`	Character vector that denotes cell types of interest.
`n_break`	Total number of distance ranges/intervals of interest made from 0 to `rmax` for calculating entropy
`rmax`	Max distance between pairs of cells

Value

object of class `mxFDA` with a dataframe in the `multivariate_summaries` slot

`extract_fpca_object` *Extract FPCA object*

Description

Function that extracts the FPCA object created either by `run_fpca()` or `run_mfpca()` from the `mxFDA` object

Usage

```
extract_fpca_object(mxFDAobject, what)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`what`	what functional PCA data to extract, e.g. 'uni k'

Details

[Stable]

Output object can be visualized with `refund.shiny::plot_shiny()`

Value

`fpca` object created with `run_fcm()`

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load ovarian mxFDA object
data('ovarian_FDA')

#run the FPCA
ovarian_FDA = run_fpca(ovarian_FDA, metric = "uni g", r = "r", value = "fundiff",
                       lightweight = TRUE,
                       pve = .99)

#extract the fpca object
obj = extract_fpca_object(ovarian_FDA, "uni g fpca")
```

`extract_fpca_scores` *Extract FPCA scores*

Description

Extract FPCA scores

Usage

```
extract_fpca_scores(mxFDAobject, what)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`what`	what functional PCA data to extract, e.g. 'uni k'

Details

[Stable]

Value

fpca object

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load ovarian mxFDA object
data('ovarian_FDA')

#run ghe lfcm model
ovarian_FDA = run_fpca(ovarian_FDA, metric = "uni g", r = "r",
                       value = "fundiff",
                       analysis_vars = c("age", "survival_time"))

#extract uni fpc scores
fpc = extract_fpca_scores(ovarian_FDA, 'uni g fpca')
```

`extract_model` *Extract Model*

Description

Currently only extracts functional cox models not mixed functional cox models.

Usage

```
extract_model(mxFDAobject, metric, type, model_name)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`metric`	metric functional PCA data to extract, e.g. 'uni k'
`type`	one of "cox", "mcox", or "sofr" to specify the type of model to extract
`model_name`	character string of the model name to retrieve

Details

[Stable]

Value

fit functional model

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load ovarian mxFDA object
data('ovarian_FDA')

#run the lfcm model
ovarian_FDA = run_fcm(ovarian_FDA, model_name = "fit_lfcm",
                      formula = survival_time ~ age, event = "event",
                      metric = "uni g", r = "r", value = "fundiff",
                      analysis_vars = c("age", "survival_time"),
                      afcm = FALSE)

#extract model
mod = extract_model(ovarian_FDA, 'uni g', 'cox', 'fit_lfcm')
```

`extract_spatial_summary` *Summarise spatial data in mxFDA object*

Description

Summarise spatial data in mxFDA object

Usage

```
extract_spatial_summary(mxFDAobject, columns, grouping_columns = NULL)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`columns`	character vector for column heading for cells to summarise
`grouping_columns`	character vector of other columns to use as grouping, such as region classification column

Details

[Experimental]

Currently this function is experimental as it only handles data that has text in the columns. Eventually, will be able to handle any data inputs such as those from HALO where cells are designated as positive (1) or negative (0) for a cell phenotypes.

Value

data frame with percent of total points per spatial sample `columns`. If multiple levels are present in `columns` columns, multiple output columns will be provided.

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load data
data(lung_df)

#create data frames for `mxFDA` object
clinical = lung_df %>%
  dplyr::select(image_id, patient_id, patientImage_id, gender,
         age, survival_days, survival_status, stage) %>%
  dplyr::distinct()
#make small, just need to make sure it runs
spatial = lung_df %>%
  dplyr::select(-image_id, -gender, -age, -survival_days, -survival_status, -stage) %>%
  dplyr::filter(patientImage_id %in% clinical$patientImage_id[1:10])

#create `mxFDA` object
mxFDAobject = make_mxfda(metadata = clinical,
                         spatial = spatial,
                         subject_key = "patient_id",
                         sample_key = "patientImage_id")

#get markers
markers = colnames(mxFDAobject@Spatial) %>%
  grep("pheno", ., value = TRUE)

#extract summary
df = extract_spatial_summary(mxFDAobject, markers)
```

`extract_summary_functions`

Extract Summary Functions

Description

Function to extract spatial summary functions from the `Spatial` slot of an `mxFDA` object

Usage

```
extract_summary_functions(
  mxFDAobject,
  r_vec = seq(0, 100, by = 10),
  extract_func = c(univariate, bivariate),
  summary_func = c(Kest, Lest, Gest),
  markvar,
  mark1,
  mark2 = NULL,
  edge_correction,
  empirical_CSR = FALSE,
  permutations = 1000
)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`r_vec`	Numeric vector of radii over which to evaluate spatial summary functions. Must begin at 0.
`extract_func`	Defaults to univariate, which calculates univariate spatial summary functions. Choose bivariate for bivariate spatial summary functions.
`summary_func`	Spatial summary function to calculate. Options are c(Kest, Lest, Gest) which denote Ripley's K, Besag's L, and nearest neighbor G function, respectively.
`markvar`	The name of the variable that denotes cell type(s) of interest. Character.
`mark1`	Character string that denotes first cell type of interest.
`mark2`	Character string that denotes second cell type of interest for calculating bivariate summary statistics. Not used when calculating univariate statistics.
`edge_correction`	Character string that denotes the edge correction method for spatial summary function. For Kest and Lest choose one of c("border", "isotropic", "Ripley", "translate", "none"). For Gest choose one of c("rs", "km", "han")
`empirical_CSR`	logical to indicate whether to use the permutations to identify the sample-specific complete spatial randomness (CSR) estimation. If there are not enough levels present in `markvar` column for permutations, the theoretical will be used.
`permutations`	integer for the number of permtuations to use if empirical_CSR is `TRUE` and exact CSR not calculable

Details

[Stable]

Complete spatial randomness (CSR) is the estimation or measure of a spatial summary function when the points or cells in a sample are randomly distributed, following no clustering or dispersion pattern. Some samples do have artifacts that may influence what CSR is under the distribution of points as they are found in the sample such as large regions of missing points or possibly in the case of tissue sections, necrotic tissue where cells are dead. Theoretical CSR requires points have an equal chance of occurring anywhere in the sample that these artifacts violate, necessitating the need to estimate or calculate what this CSR would be for each sample independently. Previously Wilson et al. had demonstrated cases in which sample-specific CSR was important over the use of the theoretical in calculating how much the observed deviates from expected.

Value

an object of class `mxFDA` containing the corresponding spatial summary function slot filled. See `make_mxfda()` for object structure details.

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

References

Xiao, L., Ruppert, D., Zipunnikov, V., and Crainiceanu, C. (2016). Fast covariance estimation for high-dimensional functional data. *Statistics and Computing*, 26, 409-421. DOI: 10.1007/s11222-014-9485-x.

Wilson, C., Soupir, A. C., Thapa, R., Creed, J., Nguyen, J., Segura, C. M., Gerke, T., Schildkraut, J. M., Peres, L. C., & Fridley, B. L. (2022). Tumor immune cell clustering and its association with survival in African American women with ovarian cancer. PLoS computational biology, 18(3), e1009900. https://doi.org/10.1371/journal.pcbi.1009900

Creed, J. H., Wilson, C. M., Soupir, A. C., Colin-Leitzinger, C. M., Kimmel, G. J., Ospina, O. E., Chakiryan, N. H., Markowitz, J., Peres, L. C., Coghill, A., & Fridley, B. L. (2021). spatialTIME and iTIME: R package and Shiny application for visualization and analysis of immunofluorescence data. *Bioinformatics* (Oxford, England), 37(23), 4584–4586. https://doi.org/10.1093/bioinformatics/btab757

`spatstat.explore::Kest()`

`spatstat.explore::Gest()`

`spatstat.explore::Lest()`

`spatstat.explore::Kcross()`

`spatstat.explore::Gcross()`

`spatstat.explore::Lcross()`

Examples

```
#load ovarian FDA object
data('ovarian_FDA')

#run function
ovarian_FDA = extract_summary_functions(ovarian_FDA, r_vec = 0:100,
                                        extract_func = univariate,
                                        summary_func = Gest,
                                        markvar = "immune",
                                        mark1 = "immune",
                                        edge_correction = "rs")
```

`extract_surface` *Extract Surface*

Description

Function that transforms functional models from linear or additive functional cox models into `afcmSurface` or `lfcmSurface` objects to be plotted.

Usage

```
extract_surface(
  mxFDAobject,
  metric,
  model = NULL,
  r = "r",
  value = "fundiff",
  grid_length = 100,
  analysis_vars,
  p = 0.05,
  filter_cols = NULL
)
```

Arguments

`mxFDAobject`	object of class `mxFDA` with model `model` calculated wihtin
`metric`	spatial summary function to extract surface for
`model`	character string for the name of the model for `metric` data
`r`	Character string, the name of the variable that identifies the function domain (usually a radius for spatial summary functions). Default is "r".
`value`	Character string, the name of the variable that identifies the spatial summary function values. Default is "fundiff".
`grid_length`	Length of grid on which to evaluate coefficient functions.
`analysis_vars`	Other variables used in modeling FCM fit.
`p`	numeric p-value used for predicting significant AFCM surface
`filter_cols`	a named vector of factors to filter summary functions to in `c(Derived_Column = "Level_to_Filter")` format

Value

a 4 element list of either class `lfcmSurface` or `afcmSurface` depending on the class of model

`Surface`	`data.frame` for term predictions for the surface of the metric * radius area
`Prediction`	`data.frame` for standard error of the terms for the above surface. AFCM models use the `p` to set the upper and lower standard errors of β_1

`Metric`	character of the spatial summary function used; helps keep track if running many models
`P-value`	a numeric value of the input p-value

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load ovarian mxFDA object
data('ovarian_FDA')

#run the lfcm model
ovarian_FDA = run_fcm(ovarian_FDA, model_name = "fit_lfcm",
                      formula = survival_time ~ age, event = "event",
                      metric = "uni g", r = "r", value = "fundiff",
                      analysis_vars = c("age", "survival_time"),
                      afcm = FALSE)

#extract surface
model_surface = extract_surface(ovarian_FDA, metric = 'uni g',
                                model = 'fit_lfcm',
                                analysis_vars = 'age') #variables in model
```

`filter_spatial` *Filter Spatial data*

Description

function to filter the spatial data slot of the `mxFDA` object.

Usage

```
filter_spatial(mxFDAobject, ..., based_on = "meta", force = FALSE)
```

Arguments

`mxFDAobject`	object of class `mxFDA`
`...`	expressions that return a logical TRUE/FALSE value when evaluated on columns of the meta data slot. These expressions get passed to `dplyr::filter()` so must be compatible.
`based_on`	character for which data slot to use for filtering, either 'meta', or 'spatial'. Default to 'meta'.
`force`	logical whether or not to return empty spatial data *if* filtering results in 0 rows

Value

object of class `mxFDA` with the spatial slot filtered. See `make_mxfda()` for more details on object

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

References

`dplyr::filter()`

Examples

```
#load ovarian mxFDA object
data(ovarian_FDA)

#filter ages greater than 50
ovarian_FDA_age50 = filter_spatial(ovarian_FDA, age >= 50, based_on = 'meta')
```

`lung_df` *Multiplex imaging data from a non-small cell lung cancer study.*

Description

This data is adapted from the VectraPolarisData Bioconductor package. There are multiple ROIs for each patient. Data was filtered to include only the cells in the tumor compartment.

Usage

```
lung_df
```

Format

`lung_df`**:**

A data frame with 879,694 rows and 19 columns:

image_id Image id for a given patient

patient_id Unique patient id

age Patient age at time of cancer diagnosis

survival_days Survival time from diagnosis, in days

survival_status Censoring variable, 1 = death, 0 = censor

x Cell x position

y Cell y position ...

Source

`https://bioconductor.org/packages/release/data/experiment/html/VectraPolarisData.html`

`lung_FDA` *Multiplex imaging data from a non-small cell lung cancer study*

Description

This data is adapted from the VectraPolarisData Bioconductor package. There are multiple ROIs for each patient.

Usage

```
lung_FDA
```

Format

`lung_FDA`**:**

An mxFDA object with augmented non-small cel lung cancer multiplex immunofluorescence data, and NN G(r) calculated:

Metadata information about the spatial samples with column `sample_key` column in both

Spatial cell-level information with `x` and `y` columns along with `sample_key` to link to `Metadata`

subject_key column in `Metadata` that may have multiple `sample_key` values for each, akin to patient IDs

sample_key column in both `Metadata` and `Spatial` that is a 1:1 with the samples (unique per sample)

univariate_summaries univariate summary slot with nearest neighbor G calculared

bivariate_summaries empty slot available for bivariate summaries

multiivariate_summaries empty slot available for multivariate summaries

functional_pca empty slot for functional PCA data of summaries

functional_cox empty slot for functional models

Details

Spatial summary functions of lung cancer multiplex imaging data.

This data is adapted from the VectraPolarisData Bioconductor package. Signal between the survival outcome and spatial summary functions has been augmented for teaching purposes. Spatial relationship is summarized using the nearest neighbor G function.

Includes only spatial samples that had 10 or more radii with calculable G function

Source

https://bioconductor.org/packages/release/data/experiment/html/VectraPolarisData.html

---

`make_mxfda` *Make mxFDA class object*

Description

Used to create an object of class `mxFDA` that can be used with the mxfda package for functional data analysis.

Usage

```
make_mxfda(metadata, spatial = NULL, subject_key, sample_key)
```

Arguments

`metadata`	metadata with columns `subject_key` and `sample_key`
`spatial`	spatial information, either list or df, with column `sample_key`. `Spatial` can be empty if inputting data already derived. See `add_summary_function()` for more details.
`subject_key`	column name in `Metadata` for subject ID
`sample_key`	column linking `Metadata` to `Spatial` data

Details

[Stable]

Value

S4 object of class `mxFDA`

`Metadata`	slot of class `data.frame` that contains sample and subject level information
`Spatial`	slot of class `data.frame` that contains point level information within samples. An example would be cells belonging to TMA cores
`subject_key`	slot of class `character` that corresponds to a column in the `Metadata` slot that groups samples at a subject level. An example would be "*patient_id*"
`sample_key`	slot of class `character` that corresponds to a column both in the `Metadata` and `Spatial` slots that links samples to characteristics
`univariate_summaries`	slot of class `list` where univariate summary functions calculated on `Spatial` would be stored
`bivariate_summaries`	slot of class `list` where bivariate summary functions calculated on `Spatial` would be stored
`multiivariate_summaries`	slot of class `list` where entropy summary functions calculated on `Spatial` would be stored

`functional_pca`	slot of class `list` where FPCA results are stored
`functional_mpca`	slot of class `list` where MFPCA results are stored
`functional_cox`	slot of class `list` where functional cox model results are stored
`functional_mcox`	slot of class `list` where mixed functional cox model results are stored
`scalar_on_function`	slot of class `list` where functional models are fit to scalar responses

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#select sample metadata
clinical = lung_df %>%
  dplyr::select(image_id, patient_id, patientImage_id,
                gender, age, survival_days, survival_status, stage) %>%
  dplyr::distinct()
#select the spatial information
spatial = lung_df %>%
  dplyr::select(-image_id, -gender, -age, -survival_days, -survival_status, -stage)
sample_id_column = "patientImage_id"
#create the mxFDA object
mxFDAobject = make_mxfda(metadata = clinical,
                         spatial = spatial,
                         subject_key = "patient_id",
                         sample_key = sample_id_column)
```

`ovarian_FDA` *Multiplex imaging data from an ovarian cancer tumor microarray*

Description

This data is adapted from the VectraPolarisData Bioconductor package and comes from a tumor-microarray of tissue samples from 128 patients with ovarian cancer. There is one patient per subject.

Usage

```
ovarian_FDA
```

Format

`ovarian_FDA`**:**

An mxFDA object with augmented ovarian cancer multiplex immunofluorescence data, and NN G(r) calculated:

Metadata	information about the spatial samples with column `sample_key` column in both
Spatial	cell-level information with `x` and `y` columns along with `sample_key` to link to `Metadata`
subject_key	column in `Metadata` that may have multiple `sample_key` values for each, akin to patient IDs
sample_key	column in both `Metadata` and `Spatial` that is a 1:1 with the samples (unique per sample)
univariate_summaries	univariate summary slot with nearest neighbor G calculared
bivariate_summaries	empty slot available for bivariate summaries
multiivariate_summaries	empty slot available for multivariate summaries
functional_pca	empty slot for functional PCA data of summaries
functional_cox	empty slot for functional models

Details

Spatial summary functions of ovarian cancer multiplex imaging data.

This data is adapted from the VectraPolarisData Bioconductor package. Signal between the survival outcome and spatial summary functions has been augmented for teaching purposes. Spatial relationship is summarized using the nearest neighbor G function.

Source

https://bioconductor.org/packages/release/data/experiment/html/VectraPolarisData.html

---

`plot.afcmSurface` *Plot afcm object*

Description

Plot afcm object

Usage

```
## S3 method for class 'afcmSurface'
plot(x, ...)
```

Arguments

`x`	object of class `afcmSurface` to be plotted
`...`	currently ignored

Value

object compatable with ggplot2

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

`plot.lfcmSurface` *Plot lfcm surface*

Description

Plot lfcm surface

Usage

```
## S3 method for class 'lfcmSurface'
plot(x, ...)
```

Arguments

`x`	object of class `lfcmSurface` to be plotted
`...`	currently ignored

Value

object compatable with ggplot2

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

`plot.mxFDA` *Plot mxFDA object*

Description

Plot mxFDA object

Usage

```
## S3 method for class 'mxFDA'
plot(x, filter_cols = NULL, ...)
```

Arguments

`x`	object of class `mxFDA` to be plotted
`filter_cols`	column key to filter
`...`	additional paramters including `y`, `what`, and `sampleID` to inform whats to be plotted

Details

[Stable]

If there are multiple metrics that are included in the derived table, an extra parameter `filter_cols` in the format of `c(Derived_Column = "Level_to_Filter")` will return curves from the `Derived_Column` with the level `Level_to_Filter`

When plotting mFPCA objects, additional arguments `level1` and `level2` help indicate which FPCA from level 1 and level 2 to plot

Value

object of class `ggplot` compatible the `ggplot2` aesthetics

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#set seed
set.seed(333)
#plotting summary
data("ovarian_FDA")
plot(ovarian_FDA, y = 'fundiff', what = 'uni g')
#running fpca
ovarian_FDA = run_fpca(ovarian_FDA, metric = "uni g", r = "r", value = "fundiff",
                       lightweight = TRUE,
                       pve = .99)
```

```
#plot fpca
plot(ovarian_FDA, what = 'uni g fpca', pc_choice = 1)
```

`plot.sofr` *Plot sofr object*

Description

Plot sofr object

Usage

```
## S3 method for class 'sofr'
plot(x, ...)
```

Arguments

`x`	object of class `sofr` to be plotted
`...`	currently ignored

Value

object compatable with ggplot2

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

`plot_fpc` *Create plot of mean +/- scaled eigenfunctions from FPCA*

Description

Produces a ggplot with mean plus or minus two standard deviations of a selected FPC.

Usage

```
plot_fpc(obj, pc_choice)
```

Arguments

`obj`	fpca object to be plotted.
`pc_choice`	FPC to be plotted.

Details

[Superseded]

Value

object of class ggplot

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

`plot_mfpc` *Create plot of mean +/- scaled eigenfunctions from FPCA*

Description

Produces a ggplot with mean plus or minus two standard deviations of a selected FPC.

Usage

```
plot_mfpc(obj, pc_choice_level1, pc_choice_level2)
```

Arguments

`obj`	fpca object to be plotted.
`pc_choice_level1`, `pc_choice_level2`	FPC to be plotted.

Details

[Superseded]

Value

list of objects of class ggplot

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

`run_fcm` *Run Function Cox Models*

Description

Fit a functional Cox regression model.

Usage

```
run_fcm(
  mxFDAobject,
  model_name,
  formula,
  event = "event",
  metric = "uni k",
  r = "r",
  value = "fundiff",
  afcm = FALSE,
  smooth = FALSE,
  filter_cols = NULL,
  ...,
  knots = NULL
)
```

Arguments

`mxFDAobject`	Dataframe of spatial summary functions from multiplex imaging data, in long format. Can be estimated using the function `extract_summary_functions` or provided separately.
`model_name`	character string to give the fit model in the functional cox slot
`formula`	Formula to be fed to mgcv in the form of survival_time ~ x1 + x2. Does not contain functional predictor. Character valued. Data must contain censoring variable called "event".
`event`	character string for the column in Metadata that contains 1/0 for the survival event
`metric`	name of calculated spatial metric to use
`r`	Character string, the name of the variable that identifies the function domain (usually a radius for spatial summary functions). Default is "r".
`value`	Character string, the name of the variable that identifies the spatial summary function values. Default is "fundiff".
`afcm`	If TRUE, runs additive functional Cox model. If FALSE, runs linear functional cox model. Defaults to linear functional cox model.
`smooth`	Option to smooth data using FPCA. Defaults to FALSE.
`filter_cols`	a named vector of factors to filter summary functions to in `c(Derived_Column = "Level_to_Filter")` format

`...`	Optional other arguments to be passed to `fpca.face`
`knots`	Number of knots for defining spline basis.

Details

[Stable]

Value

A `list` which is a linear or additive functional Cox model fit. See `mgcv::gam` for more details.

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load ovarian mxFDA object
data('ovarian_FDA')

#run the lfcm model
ovarian_FDA = run_fcm(ovarian_FDA, model_name = "fit_lfcm",
                      formula = survival_time ~ age, event = "event",
                      metric = "uni g", r = "r", value = "fundiff",
                      afcm = FALSE)
```

`run_fpca` *run_fpca*

Description

This is a wrapper for the function `fpca.face` from the `refund` package. EXPAND

Usage

```
run_fpca(
  mxFDAobject,
  metric = "uni k",
  r = "r",
  value = "fundiff",
  knots = NULL,
  analysis_vars = NULL,
  lightweight = FALSE,
  filter_cols = NULL,
  ...
)
```

Arguments

`mxFDAobject`	object of class `mxFDA` created by `make_mxfda` with metrics derived with `extract_summary_functions`
`metric`	name of calculated spatial metric to use
`r`	Character string, the name of the variable that identifies the function domain (usually a radius for spatial summary functions). Default is "r".
`value`	Character string, the name of the variable that identifies the spatial summary function values. Default is "fundiff".
`knots`	Number of knots for defining spline basis.Defaults to the number of measurements per function divided by 2.
`analysis_vars`	Optional list of variables to be retained for downstream analysis.
`lightweight`	Default is FALSE. If TRUE, removes Y and Yhat from returned FPCA object. A good option to select for large datasets.
`filter_cols`	a named vector of factors to filter summary functions to in `c(Derived_Column = "Level_to_Filter")` format
`...`	Optional other arguments to be passed to `fpca.face`

Details

[Stable]

The `filter_cols` parameter is useful when the summary function was input by the user using `add_summary_function()` and the multiple marks were assessed; a column called "Markers" with tumor infiltrating lymphocytes as well as cytotoxic T cells. This parameter allows for filtering down to include only one or the other.

Value

A `mxFDA` object with the `functional_pca` slot filled for the respective spatial summary function containing:

`mxfundata`	The original dataframe of spatial summary functions, with scores from FPCA appended for downstream modeling
`fpc_object`	A list of class "fpca" with elements described in the documentation for `refund::fpca.face`

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

References

Xiao, L., Ruppert, D., Zipunnikov, V., and Crainiceanu, C. (2016). Fast covariance estimation for high-dimensional functional data. *Statistics and Computing*, 26, 409-421. DOI: 10.1007/s11222-014-9485-x.

Examples

```
#load ovarian mxFDA object
data('ovarian_FDA')

#run the FPCA
ovarian_FDA = run_fpca(ovarian_FDA, metric = "uni g", r = "r", value = "fundiff",
                       lightweight = TRUE,
                       pve = .99)
```

---

run_mfcm *Run function Cox models for data with multiple samples per subject*

Description

Fit a functional Cox regression model when there are multiple functions per subject, which arise from multiple samples per subject. It is not necessary for all subjects to have the same number of samples.The function first performs a multilevel functional principal components analysis (MFPCA) decomposition to the spatial summary function. Then, the average curve for each subject is used in a functional Cox model (FCM). Variation around each subject's mean is captured by calculating the standard deviation of the level 2 scores from MFPCA, then including this as a scalar variable in the FCM called "level2_score_sd".

Usage

```
run_mfcm(
  mxFDAobject,
  model_name,
  formula,
  event = "event",
  metric = "uni k",
  r = "r",
  value = "fundiff",
  afcm = FALSE,
  filter_cols = NULL,
  pve = 0.99,
  ...,
  knots = NULL
)
```

Arguments

`mxFDAobject`	Dataframe of spatial summary functions from multiplex imaging data, in long format. Can be estimated using the function `extract_summary_functions` or provided separately.
`model_name`	character string to give the fit model in the functional cox slot

`formula`	Formula to be fed to mgcv in the form of survival_time ~ x1 + x2. Does not contain functional predictor. Character valued. Data must contain censoring variable called "event".
`event`	character string for the column in Metadata that contains 1/0 for the survival event
`metric`	name of calculated spatial metric to use
`r`	Character string, the name of the variable that identifies the function domain (usually a radius for spatial summary functions). Default is "r".
`value`	Character string, the name of the variable that identifies the spatial summary function values. Default is "fundiff".
`afcm`	If TRUE, runs additive functional Cox model. If FALSE, runs linear functional cox model. Defaults to linear functional cox model.
`filter_cols`	a named vector of factors to filter summary functions to in `c(Derived_Column = "Level_to_Filter")` format
`pve`	Proportion of variance explained by multilevel functional principal components analysis in mfpca step
`...`	Optional other arguments to be passed to `fpca.face`
`knots`	Number of knots for defining spline basis.

Details

[Stable]

Value

A `list` which is a linear or additive functional Cox model fit. See `mgcv::gam` for more details.

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load ovarian mxFDA object
data('lung_FDA')

# run the lfcm model
lung_FDA = run_mfcm(lung_FDA, model_name = "fit_mlfcm",
                      formula = survival_days ~ age,
                      event = "survival_status",
                      metric = "uni g", r = "r", value = "fundiff",
                      pve = 0.99,
                      afcm = FALSE)
```

`run_mfpca` *run_fpca*

Description

This is a wrapper for the function `mfpca.face` from the `refund` package. EXPAND

Usage

```
run_mfpca(
  mxFDAobject,
  metric = "uni k",
  r = "r",
  value = "fundiff",
  knots = NULL,
  lightweight = FALSE,
  ...
)
```

Arguments

`mxFDAobject`	object of class `mxFDA` created by `make_mxfda()` with metrics derived with `extract_summary_functions`
`metric`	name of calculated spatial metric to use
`r`	Character string, the name of the variable that identifies the function domain (usually a radius for spatial summary functions). Default is "r".
`value`	Character string, the name of the variable that identifies the spatial summary function values. Default is "fundiff".
`knots`	Number of knots for defining spline basis.Defaults to the number of measurements per function divided by 2.
`lightweight`	Default is FALSE. If TRUE, removes Y and Yhat from returned mFPCA object. A good option to select for large datasets.
`...`	Optional other arguments to be passed to `mfpca.face`

Details

[Stable]

Value

A `mxFDA` object with the `functional_mpca` slot for the respective spatial summary function containing:

`mxfundata`	The original dataframe of spatial summary functions, with scores from FPCA appended for downstream modeling
`fpc_object`	A list of class "fpca" with elements described in the documentation for `refund::fpca.face`

Author(s)

unknown <`first.last@domain.extension`>

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

References

Xiao, L., Ruppert, D., Zipunnikov, V., and Crainiceanu, C. (2016). Fast covariance estimation for high-dimensional functional data. *Statistics and Computing*, 26, 409-421. DOI: 10.1007/s11222-014-9485-x.

Examples

```
#load data
data(lung_FDA)

#run mixed fpca
lung_FDA = run_mfpca(lung_FDA, metric = 'uni g')
```

`run_sofr` *Run Scalar on Function Regression*

Description

Fit a scalar-on-function regression model. Uses refund::pfr under the hood for computations, and stores results in the mxfda object.

Usage

```
run_sofr(
  mxFDAobject,
  model_name,
  formula,
  family = "gaussian",
  metric = "uni k",
  r = "r",
  value = "fundiff",
  smooth = FALSE,
  filter_cols = NULL,
  ...,
  knots = NULL
)
```

Arguments

`mxFDAobject`	Dataframe of spatial summary functions from multiplex imaging data, in long format. Can be estimated using the function `extract_summary_functions` or provided separately.
`model_name`	character string to give the fit model
`formula`	Formula to be fed to mgcv in the form of outcome ~ x1 + x2. Does not contain functional predictor. Character valued.
`family`	Exponential family distribution to be passed to `mgcv::gam`. Defaults to "gaussian". Select "binomial" for binary outcome.
`metric`	Name of calculated spatial metric to use
`r`	Character string, the name of the variable that identifies the function domain (usually a radius for spatial summary functions). Default is "r".
`value`	Character string, the name of the variable that identifies the spatial summary function values. Default is "fundiff".
`smooth`	Option to smooth data using FPCA. Defaults to FALSE.
`filter_cols`	a named vector of factors to filter summary functions to in `c(Derived_Column = "Level_to_Filter")` format
`...`	Optional other arguments to be passed to `fpca.face`
`knots`	Number of knots for defining spline basis.

Details

[Stable]

Value

A `list` which is a linear or additive functional Cox model fit. See `mgcv::gam` for more details.

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

Examples

```
#load ovarian mxFDA object
data('ovarian_FDA')

# run scalar on function regression model with a continuous outcome (age)
ovarian_FDA = run_sofr(ovarian_FDA,
                       model_name = "fit_sofr",
                       formula = age~stage,
                       metric = "uni g", r = "r", value = "fundiff")

# run scalar on function regression model with a binary outcome (stage)
# also known as functional logistic regression
```

```
ovarian_FDA = run_sofr(ovarian_FDA,
                       model_name = "fit_sofr",
                       formula = stage~age,
                       family = "binomial",
                       metric = "uni g", r = "r", value = "fundiff")
```

`summary.mxFDA` *Summary method for object of class* `mxFDA`

Description

Summary method for object of class `mxFDA`

Usage

```
## S3 method for class 'mxFDA'
summary(object, ...)
```

Arguments

`object`	object of class `mxFDA`
`...`	unused currently

Details

[Stable]

Value

summary of object to the R console

Author(s)

Alex Soupir <`alex.soupir@moffitt.org`>

`univariate` *univariate*

Description

Internal function called by `extract_summary_functions()` to calculate a univariate spatial summary function for a single image.

Usage

```
univariate(
  mximg,
  markvar,
  mark1,
  mark2,
  r_vec,
  func = c(Kest, Lest, Gest),
  edge_correction,
  empirical_CSR = FALSE,
  permutations = 1000
)
```

Arguments

`mximg`	Dataframe of cell-level multiplex imaging data for a single image. Should have variables `x` and `y` to denote x and y spatial locations of each cell.
`markvar`	The name of the variable that denotes cell type(s) of interest. Character.
`mark1`	dummy filler, unused
`mark2`	dummy filler, unused
`r_vec`	Numeric vector of radii over which to evaluate spatial summary functions. Must begin at 0.
`func`	Spatial summary function to calculate. Options are c(Kest, Lest, Gest) which denote Ripley's K, Besag's L, and nearest neighbor G function, respectively.
`edge_correction`	Character string that denotes the edge correction method for spatial summary function. For Kest and Lest choose one of c("border", "isotropic", "Ripley", "translate", "none"). For Gest choose one of c("rs", "km", "han")
`empirical_CSR`	logical to indicate whether to use the permutations to identify the sample-specific complete spatial randomness (CSR) estimation.
`permutations`	integer for the number of permtuations to use if empirical_CSR is `TRUE` and exact CSR not calculable

Details

[Stable]

Value

A `data.frame` containing:

`r`	the radius of values over which the spatial summary function is evaluated
`sumfun`	the values of the spatial summary function
`csr`	the values of the spatial summary function under complete spatial randomness
`fundiff`	sumfun - csr, positive values indicate clustering and negative values repulsion

Author(s)

Julia Wrobel <`julia.wrobel@emory.edu`>

Alex Soupir <`alex.soupir@moffitt.org`>

References

Creed, J. H., Wilson, C. M., Soupir, A. C., Colin-Leitzinger, C. M., Kimmel, G. J., Ospina, O. E., Chakiryan, N. H., Markowitz, J., Peres, L. C., Coghill, A., & Fridley, B. L. (2021). spatialTIME and iTIME: R package and Shiny application for visualization and analysis of immunofluorescence data. *Bioinformatics* (Oxford, England), 37(23), 4584–4586. https://doi.org/10.1093/bioinformatics/btab757

Index