Package ‘refund.shiny’

November 20, 2021

Title Interactive Plotting for Functional Data Analyses

Version 0.4.1

Description
Shiny applications are produced for different types of popular functional data analyses. The functional data analyses are implemented in the refund package, then refund.shiny reads in the refund object and implements an object-specific set of plots based on the object class using S3.

Depends R (>= 3.0.1)
Imports ggplot2, tidyr, shiny (>= 0.11), reshape2, dplyr, gridExtra, lme4, plotly, refund, registr

License GPL (>= 2)
Encoding UTF-8
Repository CRAN
RoxygenNote 7.1.2
Suggests testthat

NeedsCompilation no

Author Julia Wrobel [aut, cre], Jeff Goldsmith [aut]

Maintainer Julia Wrobel <julia.wrobel@cuanschutz.edu>

Date/Publication 2021-11-20 22:50:01 UTC

R topics documented:

as_refundObj ........................................ 2
as_refundObj.matrix .................................. 3
bakeLasagna .......................................... 4
combinat .............................................. 4
createInputCall ...................................... 5
createInvLink ........................................ 5
downloadModule ...................................... 6
downloadModuleUI ................................... 6
fMBD .................................................... 7
as_refundObj

Convert data to refund objects for use in functional data analyses

Description

Very experimental function, primarily used to convert matrices storing functional data to data.frames with specific variable names.

Usage

as_refundObj(obj, ...)

Arguments

obj Object to be converted. Currently supports class matrix, formatted so that rows contain functional observations on subjects.

... additional arguments to be passed to methods.

Value

An object of classes data.frame and refund.object, the latter of which is so far not used. Columns are id (taken from the rownames of obj, if they exist), index (with behavior described above), and value (taken from entries in obj).
as_refundObj.matrix

Author(s)
Jeff Goldsmith <jeff.goldsmith@columbia.edu>

Examples

```r
## Not run:
library(ggplot2)
library(refund)
cca_df = as_refundObj(DTI$cca)
ggplot(cca_df, aes(x = index, y = value, group = id)) + geom_line()
## End(Not run)
```

Description

Convert matrices to dataframes for use in functional data analyses

Usage

```r
## S3 method for class 'matrix'
as_refundObj(obj, index = NULL, ...)
```

Arguments

- **obj**: Matrix object to be converted; rows contain functional observations on subjects.
- **index**: Time grid on which functional data are observed; defaults to NULL, which assumes an equally-spaced grid on [0,1].
- **...**: additional arguments to be passed to methods (not used).

Value

An object of classes data.frame and refund.object, the latter of which is so far not used. Columns are id (taken from the rownames of obj, if they exist), index (with behavior described above), and value (taken from entries in obj).

Author(s)
Jeff Goldsmith <jeff.goldsmith@columbia.edu>
Examples

```r
library(ggplot2)
library(refund)

cca_df = as_refundObj(DTI$cca)
ggplot(cca_df, aes(x = index, y = value, group = id)) + geom_line()
```

---

**bakeLasagna**

Create side-by-side lasagna plot and density plot

**Description**

Internal method used in conjunction with `makeLasagna()` to create side-by-side lasagna plot and distribution plot. The distribution plot gives distribution of sorting covariate.

**Usage**

```r
bakeLasagna(data, data.long, covariate = NULL)
```

**Arguments**

- **data**: Dataset for lasagna plot. Same data used in `makeLasagna()` function.
- **data.long**: Sorted longform dataset for lasagna plot output by `makeLasagna()` function.
- **covariate**: User-selected covariate for sorting the rows in the lasagna plot. Defaults to `NULL`, in which case data is sorted by row number.

**Author(s)**

Julia Wrobel <julia.wrobel@cuanschutz.edu>

---

**combinat**

Internal function from 'fda' package

**Description**

function used in method for fast modified band depth (MBD) calculation

**Usage**

```r
combinat(n, p)
```
**createInputCall**

**Arguments**
- `n` number of columns in your dataset
- `p` number of rows in your dataset

**Author(s)**
Ying Sun and Marc G.Genton

**Description**
Internal method that constructs the input calls for `plot_shiny.fosr()`. The variable name and values are passed as arguments, and a corresponding slider (for numeric) or drop-down (for factor) input is constructed.

**Usage**

```r
createInputCall(name, variable)
```

**Arguments**
- `name` variable name
- `variable` variable values from dataset

**Author(s)**
Jeff Goldsmith <ajg2202@columbia.edu>

---

**createInvLink**

**Return inverse link function for plot_shiny.fpc()**

**Description**
Internal method that constructs the inverse link function for a generalized FPCA fit. This is used in toggling between plots on the natural scale and on the response scale.

**Usage**

```r
createInvLink(family = NULL)
```

**Arguments**
- `family` Family of the (generalized) FPCA. Currently supported families are `gaussian` and `binomial`. 
downloadModule

Description

Internal method that creates UI with buttons to download a plot as a PDF or ggplot object.

Usage

downloadModule(input, output, session, plotObject, plotName)

Arguments

- **input**: gets user input from UI
- **output**: designates output for UI
- **session**: Shiny variable for server modules
- **plotObject**: Reactive plot object defined elsewhere in the server function.
- **plotName**: Character string designating name of the plot for PDF output.

Author(s)

Julia Wrobel <jw3134@cumc.columbia.edu>

downloadModuleUI

Description

Internal method that creates UI with buttons to download a plot as a PDF or ggplot object.

Usage

downloadModuleUI(id)

Arguments

- **id**: name of module. Allows each call of this module to be uniquely identified.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>
**fMBD**

fast modified band depth calculation for fda

Method for fast modified band depth (fMBD) calculation

**Usage**

fMBD(data)

**Arguments**

- **data** name of dataset

**Author(s)**

Ying Sun and Marc G. Genton

---

**getWidth**

Get spaces between timepoints as widths for binary registration lasagna plot.

**Description**

Get spaces between timepoints as widths for binary registration lasagna plot.

**Usage**

getWidth(z)

**Arguments**

- **z** time values for a specific subject

**Author(s)**

Julia Wrobel <jw3134@cumc.columbia.edu>
makeLasagna
Pre-process data for lasagna plot

Description
Internal method that takes a dataframe of observed data with an outcome matrix and user-selected covariate, sorts outcome by the selected covariate, and assigns heights to each row based on value of the selected covariate. The resulting dataframe is used with bakeLasagna() to create lasagna plot.

Usage
makeLasagna(data, outcome, covariate = NULL)

Arguments
data Dataset for lasagna plot.
oneoutcome Matrix of values where each row represents a functional observation.
covariate User-selected covariate for sorting the rows in the lasagna plot. Defaults to NULL, in which case data is sorted by row number.

Author(s)
Julia Wrobel <julia.wrobel@cuanschutz.edu>

make_linCom
Create lincom plot for FPCA panels

Description
Produces a ggplot with mean and sliders to change weighting of each PC; allows you to obtain range of potential fitted values.

Usage
make_linCom(obj, pc_weights, response_scale = FALSE)

Arguments
obj f pca object to be plotted.
pc_weights User-selected weights for FPCs
response_scale Scale of response to be plotted. If TRUE results are plotted on response scale, if FALSE results are plotted on natural scale.
**make_muPC**

Create muPC plot for FPCA panels

**Description**

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

**Usage**

```
make_muPC(obj, pc_choice, response_scale = FALSE)
```

**Arguments**

- **obj**: fpca object to be plotted.
- **pc_choice**: FPC to be plotted.
- **response_scale**: Scale of response to be plotted. If TRUE results are plotted on response scale, if FALSE results are plotted on natural scale.

---

**mfpcaCalls**

Create input calls for plot_shiny.mfpca()

**Description**

Internal method that constructs the input calls for plot_shiny.mfpca(). The number of sliders to construct for each level is passed as an argument, and corresponding sliders for each FPC are constructed.

**Usage**

```
mfpcaCalls(plot.npc, plotObj, percents)
```

**Arguments**

- **plot.npc**: list of 2 numeric entries giving number of sliders at each level
- **plotObj**: the mfpca object plotted in the plot_shiny.mfpca() function.
- **percents**: the percent variance calculated for each eigen values for levels 1 and 2.

**Value**

A list of numbers that indicate percent variance for selected level.

**Author(s)**

Julia Wrobel <jw3134@columbia.edu>
outliers

*Identifies outliers for plot_shiny.fosr()*

**Description**

Internal method that assigns band depth values to curves based on exact fast MBD computation (Sun & Genton, 2012). Code modified from fbplot in fda package. A dataframe of residuals is passed as an argument, and depths and outlying curves are returned.

**Usage**

```
outliers(data, factor = 1.5)
```

**Arguments**

- **data**: matrix or df of functional observations
- **factor**: a constant that determines the fences for outliers. Defaults to 1.5, as in classical definition for Tukey outliers.

**Author(s)**

Julia Wrobel <jw3134@columbia.edu>

**References**


---

plot_shiny

*plot_shiny The generic function for interactive plots of functional data analyses*

**Description**

Interactive Plotting for Functional Data

**Usage**

```
plot_shiny(obj, ...)
```
Arguments

obj object to be plotted. Currently, allowed data types are fpca mfpca 1fpca and fosr.

... additional arguments passed to plotting functions

Details

Function for interactive plotting of functional data analysis results.

This package builds on the refund package: tools in refund are used to conduct analyses and functions in this package create interactive visualizations of the results of those analyses. There are four major categories of analyses that can be viewed:

1. Functional principal components analyses implemented by fpca.sc, fpca.face, fpca.ssvd, and fpca2s. Plots show the mean +/- 2SD times each FPC; scree plots; linear combinations of score values and FPCs; reconstructions for each subject; and score scatterplots.

2. Function-on-scalar regression analyses implemented by bayes_fosr. Plots show the raw data colored by covariate values; fitted values depending on covariates; coefficient functions; and residuals.

3. Multilevel functional principal components analyses implemented by mfpca.sc. Plots show the mean +/- 2SD times each FPC; scree plots; linear combinations of score values and FPCs; reconstructions for each subject; and score scatterplots for levels 1 and 2.

4. Longitudinal functional principal components analyses

Value

This function outputs a shiny app based on the class of the input object.

Author(s)

Jeff Goldsmith <jeff.goldsmith@columbia.edu>, Julia Wrobel <julia.wrobel@cuanschutz.edu>

See Also

plot_shiny.fpca, plot_shiny.mfpca, plot_shiny.fosr

Examples

```r
## Not run:
library(refund)
library(dplyr)

##### FPCA Example on real data #####

data(cd4)
SC = fpca.sc(cd4)
plot_shiny(SC)
```
##### FoSR Example #####

```r
data(DTI)
DTI = DTI[complete.cases(DTI),]
fit.fosr = bayes_fosr(cca ~ pasat + sex, data = DTI)
plot_shiny(fit.fosr)
```

##### FoSR Example with outliers #####

```r
DTI$cca[1,] = DTI$cca[1,] + .4
DTI$cca[2,] = DTI$cca[2,] + .4

fosr.dti2 = bayes_fosr(cca ~ pasat + sex, data = DTI)
plot_shiny(fosr.dti2)
```

##### Longitudinal FoSR Examples #####

```r
data(DTI2)
class(DTI2$cca) = class(DTI2$cca)[-1]
DTI2 = subset(DTI2, select = c(cca, id, pasat))
DTI2 = DTI2[complete.cases(DTI2),]

fosr.dti3 = bayes_fosr(cca ~ pasat + re(id), data = DTI2, Kt = 10, Kp = 4, cov.method = "FPCA")
plot_shiny(fosr.dti3)
```

##### LFPCA Example on real data #####

```r
data(DTI)
MS <- subset(DTI, case == 1)  # subset data with multiple sclerosis (MS) case

index.na <- which(is.na(MS$cca))
Y <- MS$cca; Y[index.na] <- fpca.sc(Y)$Yhat[index.na]; sum(is.na(Y))
id <- MS$ID
visit.index <- MS$visit
t

lfpca.dti1 <- fpca.lfda(Y = Y, subject.index = id,
visit.index = visit.index, obsT = visit.time,
LongiModel.method = 'lme',
mFPCA.pve = 0.95)
plot_shiny(lfpca.dti1)

lfpca.dti2 <- fpca.lfda(Y = Y, subject.index = id,
visit.index = visit.index, obsT = visit.time,
LongiModel.method = 'fpca.sc',
mFPCA.pve = 0.80, sFPCA.pve = 0.80)
plot_shiny(lfpca.dti2)
```

## End(Not run)
plot_shiny.flc

Interactive Plotting for Functional Linear Concurrent regression

Description

Produces an interactive plot illustrating a functional linear concurrent regression analysis.

Usage

```r
## S3 method for class 'flcm'
plot_shiny(obj, xlab = "", ylab = "", title = "", ...)
```

Arguments

- `obj`: fosr object to be plotted.
- `xlab`: x axis label
- `ylab`: y axis label
- `title`: plot title
- `...`: additional arguments passed to plotting functions

Value

No object is returned. This function takes in objects of class 'fosr' and outputs a shiny application for that object.

Author(s)

Jeff Goldsmith <jeff.goldsmith@columbia.edu>, Julia Wrobel <julia.wrobel@cuanschutz.edu>

See Also

- `plot_shiny`

plot_shiny.fosr

Interactive Plotting for Functional-on-Scalar Regressions

Description

Produces an interactive plot illustrating a function-on-scalar regression analysis.

Usage

```r
## S3 method for class 'fosr'
plot_shiny(obj, xlab = "", ylab = "", title = "", ...)
```
plot_shiny.fpca

Arguments

- `obj`: fosr object to be plotted.
- `xlab`: x axis label
- `ylab`: y axis label
- `title`: plot title
- `...`: additional arguments passed to plotting functions

Value

No object is returned. This function takes in objects of class 'fosr' and outputs a shiny application for that object.

Author(s)

Jeff Goldsmith <jeff.goldsmith@columbia.edu>, Julia Wrobel <julia.wrobel@cuanschutz.edu>

See Also

- `plot_shiny`

---

plot_shiny.fpca  
Interactive Plotting for Functional Principal Component Analysis

Description

Produces an interactive plot illustrating a functional principal component analysis.

Usage

```r
## S3 method for class 'fpca'
plot_shiny(obj, xlab = "", ylab = "", title = "", thin_data = FALSE, ...)
```

Arguments

- `obj`: fpca object to be plotted.
- `xlab`: x axis label
- `ylab`: y axis label
- `title`: plot title
- `thin_data`: If TRUE data is thinned for each subject to make plotting faster. Defaults to FALSE.
- `...`: additional arguments passed to plotting functions

Value

No object is returned. This function takes in objects of class 'fpca' and outputs a shiny application for that object.
plot_shiny.lfpca

Author(s)
Julia Wrobel <julia.wrobel@cuanschutz.edu>, Jeff Goldsmith <jeff.goldsmith@columbia.edu>

See Also
plot_shiny

plot_shiny.lfpca  Interactive Plotting for Longitudinal Functional Data Analysis using FPCA

Description
Produces an interactive plot illustrating longitudinal functional data analysis (Park and Staicu, 2015).

Usage
## S3 method for class 'lfpca'
plot_shiny(obj, xlab = "", ylab = "", title = "", ...)  

Arguments
  obj  lfpca object to be plotted.
  xlab  x axis label
  ylab  y axis label
  title  plot title
  ...  additional arguments passed to plotting functions

Author(s)
So Young Park <spark13@ncsu.edu>, Ana-Maria Staicu <astaicu@ncsu.edu>

References

See Also
plot_shiny; fpca.lfda in the refund package for estimation method.
plot_shiny.mfpcaplot_shiny.registration

plot_shiny.mfpcaplot_shiny.registration

Description

Interactive Plotting for Multilevel Functional Principal Component Analysis

Description

Produces an interactive plot illustrating a multilevel functional principal component analysis.

Usage

## S3 method for class 'mfpcaplot_shiny(mfpcaplot_shiny(obj, xlab = "", ylab = "", title = "", ...)Arguments

obj mfpcaplot_shiny(object to be plotted.
xlab x axis label
ylab y axis label	
title plot title
... additional arguments passed to plotting functions

Value

No object is returned. This function takes in objects of class 'mfpcaplot_shiny and outputs a shiny application for that object.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>, Jeff Goldsmith <jeff.goldsmith@columbia.edu>

See Also

plot_shiny

plot_shiny.registration

Interactive Plotting for Registration Objects

Description

Interactive Plotting for Registration Objects

Description

Produces an interactive plot illustrating functional data before and after registration. Our registration method uses FPCA, the FPCA is plotted as well.
Usage

```r
## S3 method for class 'registration'
plot_shiny(obj, xlab = "", ylab = "", title = "", thin_data = FALSE, ...)
```

Arguments

- `obj`: registration object to be plotted.
- `xlab`: x axis label
- `ylab`: y axis label
- `title`: plot title
- `thin_data`: If TRUE data is thinned for each subject to make plotting faster. Defaults to FALSE.
- `...`: additional arguments passed to plotting functions

Value

No object is returned. This function takes in objects of class 'registration' and outputs a shiny application for that object.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>

See Also

- `plot_shiny`

---

**registerLasagna**

Create lasagna plot for unregistered and registered data

Description

Get registered and unregistered lasagna plots for binary data. Note: should make this compatible for other data types as well. Requires data to have t_hat and tstar variables.

Usage

```r
registerLasagna(data)
```

Arguments

- `data`: Dataset for lasagna plot.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>
savePDF

Save Plot Object as PDF

Description

Internal method that saves plots as PDF. Can be used with all plotting methods in the package. The name of the plot object and its name to be saved under are passed in and the plot is saved as a PDF.

Usage

savePDF(title, plotName)

Arguments

- title: new name for the plot, and name of the PDF file created
- plotName: name of the ggplot object

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>

savePlot

Save Plot Object as .RData file

Description

Internal method that saves ggplot plots as .RData files. Can be used with all plotting methods in the package. The name of the plot object and its name to be saved under are passed in and the plot is saved as an RData file.

Usage

savePlot(title, plotName)

Arguments

- title: new name for the plot, and name of the RData file created.
- plotName: name of the ggplot object.

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>
tabPanelModule

download Plot as PDF or ggplot Object, modularized server

Description
Internal method that creates UI with buttons to download a plot as a PDF or ggplot object.

Usage

```r
tabPanelModule(
  input,
  output,
  session,
  plotObject = NULL,
  plotName = NULL,
  plotObject2 = NULL,
  plotName2 = NULL,
  is.plotly = FALSE,
  is.grid = FALSE
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>gets user input from UI</td>
</tr>
<tr>
<td>output</td>
<td>designates output for UI</td>
</tr>
<tr>
<td>session</td>
<td>Shiny variable for server modules.</td>
</tr>
<tr>
<td>plotObject</td>
<td>Reactive plot object defined elsewhere in the server function.</td>
</tr>
<tr>
<td>plotName</td>
<td>Character string designating name of the plot for PDF output.</td>
</tr>
<tr>
<td>plotObject2</td>
<td>Reactive plot object for the (optional) second plot.</td>
</tr>
<tr>
<td>plotName2</td>
<td>Character string designating name of the (optional) second plot for the PDF output</td>
</tr>
<tr>
<td>is.plotly</td>
<td>Indicates if plots are plotly generated. Defaults to FALSE.</td>
</tr>
<tr>
<td>is.grid</td>
<td>Indicates if plot is generated using grid.arrange() to arrange ggplot objects. If TRUE, prints plot object implicitly rather than explicitly.</td>
</tr>
</tbody>
</table>

Author(s)

Julia Wrobel <julia.wrobel@cuanschutz.edu>
tabPanelModuleUI  
*modularized UI for creating a new tab*

**Description**

Creates a UI tab with helptext, widgets for user input, a plot, and standardized layout. The default is to create one plot, but if the argument `twoPlots` is set to TRUE, then the layout allows for two plots, where each can have separate helper text and Shiny widget calls.

**Usage**

```r
tabPanelModuleUI(
  id,  
  tabTitle,
  icon = NULL,
  calls = NULL,
  helperText = NULL,
  twoPlots = FALSE,
  calls2 = NULL,
  helperText2 = NULL,
  title2 = NULL,
  brushName = NULL,
  is.plotly = FALSE
)
```

**Arguments**

- **id**: Name of module. Allows each call of this module to be uniquely identified.
- **tabTitle**: Title of the tab, visible in UI.
- **icon**: Optional icon to appear on the tab. This attribute is only valid when using a `tabPanel` within a `navbarPage`.
- **calls**: Unevaluated expression that stores Shiny widgets (for example, a call to a sliderInput function) for the tab.
- **helperText**: Optional help text for the tab.
- **twoPlots**: defaults to FALSE, and layout is generated for one plot. If TRUE, layout is generated for two plots.
- **calls2**: Unevaluated expression that stores Shiny widgets for the (optional) second plot.
- **helperText2**: Optional help text for the (optional) second plot.
- **title2**: plot title for the (optional) second plot.
- **brushName**: character vector indicating the name of brush if you want brushing for the plot. For use in score scatterplots for `plot_shiny.fpca()` and `plot_shiny.mfpca()`.
- **is.plotly**: Indicates if plots are plotly generated. Defaults to FALSE.

**Author(s)**

Julia Wrobel <julia.wrobel@cuanschutz.edu>
thin_functional_data

---

**thin_functional_data**  
*Thin functional data*

**Description**

Takes a dense functional dataset in long form and thins it so that there are 100 observations per subject, equally spaced.

**Usage**

```r
thin_functional_data(Y, length_out = 100)
```

**Arguments**

- `Y`: functional dataframe
- `length_out`: number of points per subject for dataframe that is returned

**Author(s)**

Julia Wrobel <julia.wrobel@cuanschutz.edu>

---

**varPercent**  
*Calculate percent variance of eigenvalues for plot_shiny.mfPCA()*

**Description**

Internal method that calculates percent variance of eigenvalues for specified level (1, 2, or total) for `plot_shiny.mfPCA()`. The desired level is passed in as an argument (level = 12 for total) and a list of percent variances is returned.

**Usage**

```r
varPercent(level, plotObj)
```

**Arguments**

- `level`: numeric, 1 or 2 for levels 1 or 2, respectively, 12 to calculate total variance.
- `plotObj`: the mfPCA object plotted in the `plot_shiny.mfPCA()` function.

**Value**

a list of numbers that indicate percent variance for selected level.

**Author(s)**

Julia Wrobel <julia.wrobel@cuanschutz.edu>
Index

as_refundObj, 2
as_refundObj.matrix, 3
tabPanelModule, 19
tabPanelModuleUI, 20
thin_functional_data, 21
varPercent, 21

bakeLasagna, 4
bayes_fosr, 11

combinat, 4
createInputCall, 5
createInvLink, 5

downloadModule, 6
downloadModuleUI, 6

fMBD, 7
fpca.face, 11
fpca.sc, 11
fpca.ssvd, 11
fpca2s, 11

getWidth, 7

make_linCom, 8
make_muPC, 9
makeLasagna, 8
mfpca.sc, 11
mfpcaCalls, 9

outliers, 10

plot_shiny, 10, 13–17
plot_shiny.flcm, 13
plot_shiny.fosr, 11, 13
plot_shiny.fpca, 11, 14
plot_shiny.lfpca, 15
plot_shiny.mfpca, 11, 16
plot_shiny.registration, 16

registerLasagna, 17

savePDF, 18
savePlot, 18