Package: adaptiveGPCA (via r-universe)

September 5, 2024

Title Adaptive Generalized PCA

Version 0.1.3

Description Implements adaptive gPCA, as described in: Fukuyama, J. (2017) <arXiv:1702.00501>. The package also includes functionality for applying the method to 'phyloseq' objects so that the method can be easily applied to microbiome data and a 'shiny' app for interactive visualization.

Depends R (>= 3.1.0)

License AGPL-3

LazyData true

VignetteBuilder knitr

Suggests knitr, rmarkdown

Imports ape (>= 3.1.4), ggplot2 (>= 1.0.0), shiny (>= 1.0.0), phyloseq (>= 1.14.0)

RoxygenNote 6.0.1

Repository https://jfukuyama.r-universe.dev

RemoteUrl https://github.com/jfukuyama/adaptivegpca

RemoteRef HEAD

RemoteSha a93a3334e338a1e94128c1e9c576f982ab6a10c0

Contents

`adaptiveGPCA-package` *adaptiveGPCA: A package for structured dimensionality reduction*

Description

This package implements the methods for structured dimensionality reduction described in Fukuyama, J. (2017). The general idea is to obtain a low-dimensional representation of the data, similar to that given by PCA, which incorporates side information about the relationships between the variables. The output is similar to a PCA biplot, but the variable loadings are regularized so that similar variables are encouraged to have similar loadings on the principal axes.

Details

There are two main ways of using this package. The function `adaptivegpca` will choose how much to regularize the variables according to the similarities between them, while the function `gpcaFullFamily` produces analogous output for a range of regularization parameters. With this function, the results for the different regularization parameters are inspected with the `visualizeFullFamily` function, and the desired parameter is chosen manually.

The package also contains functionality to integrate with phyloseq: the function `processPhyloseq` takes a `phyloseq` object and creates the inputs necessary to perform adaptive gPCA on a microbiome dataset including information about the phylogenetic relationships between the bacteria.

`adaptivegpca` *Adaptive gPCA*

Description

Performs adaptive generalized PCA, a dimensionality-reduction method which takes into account similarities between the variables. See Fukuyama, J. (2017) for more details.

Usage

```
adaptivegpca(X, Q, k = 2, weights = rep(1, nrow(X)))
```

Arguments

`X`	A $n \times p$ data matrix.
`Q`	A $p \times p$ similarity matrix on the variables defining an inner product on the rows of `X`, can also be given as an eigendecomposition (formatted as the output from `eigen`).
`k`	The number of components to return.
`weights`	A vector of length n containing weights for the rows of `X`.

Value

A list containing the row/sample scores (`U`), the variable loadings (`QV`), the proportion of variance explained by each of the principal components (`vars`), the value of r that was used (`r`).

Examples

```
data(AntibioticSmall)
out.agpca = adaptivegpca(AntibioticSmall$X, AntibioticSmall$Q, k = 2)
```

`AntibioticPhyloseq` *Antibiotic time course experiment.*

Description

A phyloseq object describing a time course experiment in which three people two courses of cipro and had their gut microbiomes sampled. See Dethlefsen and Relman, PNAS (2010), at https://www.ncbi.nlm.nih.gov/pubmed/20847294 for more details.

Format

A phyloseq object.

`AntibioticSmall` *A subset of the antibiotic data*

Description

This is a smaller version of the `AntibioticPhyloseq` dataset, for use in the examples so that the running time isn't so long. It has the same samples and a randomly selected set of 200 of the taxa. It is stored as a list with three components: the normalized OTU abundances (`X`), the similarity matrix for the taxa (`Q`), and the diagonal weight matrix (`D`, the identity matrix).

Format

A list with three components.

`estimateComponents` *Estimate parameters in hierarchical model*

Description

Estimates the values of r and σ in a model $X \sim N(0, \sigma^2(rQ + (1-r)I))$.

Usage

```
estimateComponents(X, Q, Qeig = NULL)
```

Arguments

`X`	An $n \times p$ data matrix.
`Q`	A $p \times p$ matrix giving the prior variance on the rows of `X`.
`Qeig`	If the eigendecomposition of `Q` is already computed, it can be included here.

Value

A list with r and σ.

Examples

```
data(AntibioticSmall)
estimateComponents(AntibioticSmall$X, AntibioticSmall$Q)
```

`gpca` *gPCA*

Description

Performs standard gPCA with `k` components on a data matrix `X` with row inner product `Q` and weights `D`.

Usage

```
gpca(X, Q, D = rep(1, nrow(X)), k)
```

Arguments

`X`	A data matrix of size $n \times p$.
`Q`	An inner product matrix for the rows, either as a $p \times p$ matrix or an eigendecomposition of such a matrix.
`D`	Sample weights, a vector of length n.
`k`	The number of components to return.

Value

A list with variable loadings on the principal axes (`QV`), sample/row scores (`U`), the fraction of the variance explained by each of the axes (`vars`).

Examples

```
data(AntibioticSmall)
out.gpca = gpca(AntibioticSmall$X, AntibioticSmall$Q, k = 2)
```

`gpcaFullFamily` *Make a sequence of ordinations*

Description

Creates a sequence of gPCA data representations. One end of the sequence ($r = 0$) doesn't do any regularization according to the variable structure (and so is just standard PCA), and the other ($r = 1$) does a maximal amount of regularization according to the variable structure.

Usage

```
gpcaFullFamily(X, Q, weights = rep(1, nrow(X)), k = 2, rvec = (0:100)/100,
  findReflections = TRUE, returnLong = FALSE, sampledata = NULL,
  variabledata = NULL)
```

Arguments

`X`	A data matrix of size $n \times p$.
`Q`	A $p \times p$ similarity matrix defining an inner product on the rows of `X`.
`weights`	A vector of weights for the rows of `X`.
`k`	The number of components to compute for each ordination.
`rvec`	The values of r for which to make the ordinations.
`findReflections`	Whether or not flip the axes so as to make neighboring ordinations as close as possible. If `k` is very large this should be false since all possible axis combinations are searched over.
`returnLong`	Return a long data frame with the samples/variables instead of a list of data frames.
`sampledata`	Extra sample data to be included along with the sample scores.
`variabledata`	Extra variable data to be included along with the variable loadings.

Value

A list containing elements for the sample points (`locationList`), the species points (`speciesList`), and the variance fractions (`varsList`). Each element is itself a list of data frames (location/species points) or of vectors (for the variances).

Examples

```
data(AntibioticSmall)
out.ff = gpcaFullFamily(AntibioticSmall$X, AntibioticSmall$Q, k = 2)
```

---

inspectTaxonomy *Shiny gadget for tree/taxonomy inspection*

Description

Shiny gadget that allows users to visualize the scores of the taxa on the agpca axes, their positions on the phylogenetic tree, and their taxonomic assignments.

Usage

```
inspectTaxonomy(agpcafit, physeq, axes = c(1, 2), br.length = FALSE,
  height = 600)
```

Arguments

`agpcafit`	An agpca object, created either by the function `adaptivegpca` or by `visualizeFullFamily`.
`physeq`	A phyloseq object with a tree and a taxonomy table.
`axes`	The axes to plot, must be a vector of two whole numbers.
`br.length`	Plot the tree with the branch lengths?
`height`	The height, in pixels, of the plotting region.

Value

The function will open a browser window showing the tree and the locations of the taxa on the selected agpca axes. "Brushing" over the plot will highlight the positions of the selected taxa on the tree and list their taxonomic assignments. Clicking the "done" button will exit the app and return a data frame containing the positions of the selected taxa on the agpca axes, the taxonomic assignments of the selected taxa, and their names.

Examples

```
## Not run:
data(AntibioticPhyloseq)
pp = processPhyloseq(AntibioticPhyloseq)
out.agpca = adaptivegpca(pp$X, pp$Q, k = 2)
treeInspect(out.agpca, AntibioticPhyloseq)

## End(Not run)
```

`plot.adaptivegpca` *Plot an adaptivegpca object*

Description

Plots the output from `adaptivegpca`, either a scree plot, the samples, or the variables.

Usage

```
## S3 method for class 'adaptivegpca'
plot(x, type = c("scree", "samples", "variables"),
  axes = c(1, 2), ...)
```

Arguments

`x`	An object of class `adaptivegpca`
`type`	What type of plot to make. `scree` will make a scree plot showing the eigenvalues, `samples` will plot the samples, and `variables` will plot the variables.
`axes`	Which axes to plot.
`...`	Not used.

Examples

```
data(AntibioticSmall)
out.agpca = adaptivegpca(AntibioticSmall$X, AntibioticSmall$Q, k = 2)
plot(out.agpca)
plot(out.agpca, type = "samples")
plot(out.agpca, type = "variables")
```

`print.adaptivegpca` *Print an adaptivegpca object*

Description

Print an adaptivegpca object

Usage

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

Arguments

`x`	`adaptivegpca` object.
`...`	Not used.

`processPhyloseq` *Make the input matrices for adaptive gPCA*

Description

Takes a phyloseq object and creates the matrices necessary to do adaptive gPCA.

Usage

```
processPhyloseq(physeq, ca = FALSE)
```

Arguments

`physeq`	A `phyloseq` object, from the phyloseq package.
`ca`	If TRUE, do the normalization as for correspondence analysis (transform counts to relative abundances, compute sample weights, center the relative abundances according to the sample weights). Otherwise, simply center the data.

Value

A list of the matrix to perform adaptive gPCA on (`X`), the species similarity matrix (`Q`), and the sample weights (`weights`).

Examples

```
data(AntibioticPhyloseq)
pp = processPhyloseq(AntibioticPhyloseq)
```

`varianceOnEvecs` *Variance along eigenvectors of Q*

Description

Project the sample points stored in the rows of `X` along the eigenvectors of `Q` and find the variance along each of the projections.

Usage

```
varianceOnEvecs(X, Q)
```

Arguments

`X`	An $n \times p$ data matrix, each row corresponding to a sample.
`Q`	A $p \times p$ similarity matrix, either as a matrix or as its eigendecomposition (the output from `eigen`).

Value

A vector containing the variance of the samples along each of the eigenvectors of `Q`.

Examples

```
data(AntibioticSmall)
voe = varianceOnEvecs(AntibioticSmall$X, AntibioticSmall$Q)
```

`visualizeFullFamily` *Shiny gadget for adaptive gPCA*

Description

Shiny gadget that shows the ordinations from an entire family of gPCAs and returns a gPCA object with the one selected by the user.

Usage

```
visualizeFullFamily(fullFamily, sample_data = NULL,
  sample_mapping = aes_string(x = "Axis1", y = "Axis2"),
  sample_facet = NULL, var_data = NULL, var_mapping = aes_string(x =
  "Axis1", y = "Axis2"), layout = c(2, 6))
```

Arguments

`fullFamily`	The output from `gpcaFullFamily`
`sample_data`	Optional data used for plotting the samples
`sample_mapping`	An aesthetic mapping to be passed to `ggplot` for plotting the samples
`sample_facet`	A `ggplot` faceting command used for faceting the samples.
`var_data`	Optional data used for plotting the variables
`var_mapping`	An aesthetic mapping to be passed to `ggplot` for plotting the variables
`layout`	A vector of length 2. The first number gives the number of columns (out of 12) for the sidebar, the second number gives the number of columns (out of 12) for the sample plot in the main panel.

Value

This function will open a 'shiny' app in a browser window. You can investigate the results for different values of r with this app. Once you press the 'done' button, the app will close and the function will return an R object containing the results for the value of r (the regularization parameter) that was chosen in the app. The returned object is a list containing the variable loadings on the principal axes (`QV`), the sample/row scores (`U`), and the fraction of the variance explained by each of the axes (`vars`).

Examples

```
## Not run:
data(AntibioticPhyloseq)
pp = processPhyloseq(AntibioticPhyloseq)
out.ff = gpcaFullFamily(pp$X, Q = pp$Q, D = pp$D, k = 2)
out.agpca = visualizeFullFamily(out.ff,
    sample_data = sample_data(AntibioticPhyloseq),
    sample_mapping = aes(x = Axis1, y = Axis2, color = condition),
    var_data = tax_table(AntibioticPhyloseq),
    var_mapping = aes(x = Axis1, y = Axis2, color = Phylum))

## End(Not run)
```

Index