Package 'PP3'

March 9, 2018

Type Package

Title Three-Dimensional Exploratory Projection Pursuit

Version 1.2

Date 2018-03-06

Depends R (>= 3.0)

Imports stats

Description Exploratory projection pursuit is a method to discovers structure in multivariate data. At heart this package uses a projection index to evaluate how interesting a specific three-dimensional projection of multivariate data (with more than three dimensions) is. Typically, the main structure finding algorithm starts at a random projection and then iteratively changes the projection direction to move to a more interesting one. In other words, the projection index is maximised over the projection direction to find the most interesting projection. This maximum is, though, a local maximum. So, this code has the ability to restart the algorithm from many different starting positions automatically. Routines exist to plot a density estimate of projection indices over the runs, this enables the user to obtain an idea of the distribution of the projection indices, and, hence, which ones might be interesting. Individual projection solutions, including those identified as interesting, can be extracted and plotted individually. The package can make use of the mclapply() function to execute multiple runs in parallel to speed up index discovery. Projection pursuit is similar to independent component analysis. This package uses a projection index that maximises an entropy measure to look for projections that exhibit non-normality, and operates on sphered data. Hence, information from this package is different from that obtained from principal components analysis, but the rationale behind both methods is similar. Nason, G. P. (1995) <doi:10.2307/2986135>.

License GPL (>= 2)

URL http://www.stats.bris.ac.uk/~guy

NeedsCompilation yes

Author Guy Nason [aut, cre],
Robin Sibson [ctb, ths]

Maintainer Guy Nason <G.P.Nason@bristol.ac.uk>

Repository CRAN

Date/Publication 2018-03-09 12:43:48 UTC

R topics documented:

PP3-package *Three-Dimensional Exploratory Projection Pursuit*

Description

Exploratory projection pursuit is a method to discovers structure in multivariate data. At heart this package uses a projection index to evaluate how interesting a specific three-dimensional projection of multivariate data (with more than three dimensions) is. Typically, the main structure finding algorithm starts at a random projection and then iteratively changes the projection direction to move to a more interesting one. In other words, the projection index is maximised over the projection direction to find the most interesting projection. This maximum is, though, a local maximum. So, this code has the ability to restart the algorithm from many different starting positions automatically. Routines exist to plot a density estimate of projection indices over the runs, this enables the user to obtain an idea of the distribution of the projection indices, and, hence, which ones might be interesting. Individual projection solutions, including those identified as interesting, can be extracted and plotted individually. The package can make use of the mclapply() function to execute multiple runs in parallel to speed up index discovery. Projection pursuit is similar to independent component analysis. This package uses a projection index that maximises an entropy measure to look for

projections that exhibit non-normality, and operates on sphered data. Hence, information from this package is different from that obtained from principal components analysis, but the rationale behind both methods is similar. Nason, G. P. (1995) <doi:10.2307/2986135>.

Details

The DESCRIPTION file:

Package:	PP3
Type:	Package
Title:	Three-Dimensional Exploratory Projection Pursuit
Version:	1.2
Date:	2018-03-06
Authors@R:	c(person("Guy", "Nason", role=c("aut", "cre"), email="G.P.Nason@bristol.ac.uk"), person("Robin", "Sibson",
Depends:	R (>= 3.0)
Imports:	stats
Description:	Exploratory projection pursuit is a method to discovers structure in multivariate data. At heart this package use
License:	GPL (>=2)
URL:	http://www.stats.bris.ac.uk/~guy
Author:	Guy Nason [aut, cre], Robin Sibson [ctb, ths]
Maintainer:	Guy Nason <G.P.Nason@bristol.ac.uk>

Index of help topics:

PP3-package	Three-Dimensional Exploratory Projection Pursuit
PP3init	Initialize projection pursuit code
PP3ix3dvsFromTU	Compute the projection index or its derivatives from the T and U statistics.
PP3many	Main function to carry out three-dimensional projection pursuit.
PP3slowDF3	Compute the projection index or its derivative.
beetle	The Beetle Data
getPP3index	Extract projection index values from a PP3 object.
getPP3loadings	Extract projection loadings from a PP3 object.
getPP3projdata	Extract projected data from a PP3 object.
ix3sizeplot	Plot a histogram of projection index values
pdataplot	Plots the projected data for a particular random start.
plot.PP3	Two types of plot for a PP3 object.
print.PP3	Print information about a PP3 object.
summary.PP3	Print summary information about a PP3 object.

The main routine is PP3many. This package carries out three-dimensional projection pursuit on multivariate data set. It can be thought of as an alternative to principal components analysis, where interesting views of the data are presented in a three-dimensional projected space. This package

can directly produce a true three-dimensional solution and, not, a combination of, e.g. three one-dimensional views. This permits the elucidation of more complex structures. The three-dimensional solution can be used to produce colour pixel values enabling interesting contrast display of colour images from multispectal ones.

Author(s)

NA

Maintainer: NA

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

PP3many

Examples

```
#
# See extended example in PP3many
#
```

---

beetle *The Beetle Data*

Description

The Lubischew flea beetle data consisting of six observations measured on 74 flea beetles on three species. The rows dimnames attribute contains the species id.

Usage

```
data(beetle)
```

Format

A 74x6 data matrix.

Source

Lubischew, A.A. (1962) On the use of discriminant functions in taxonomy. *Biometrics*, **18**, 455-477.

References

Lubischew, A.A. (1962) On the use of discriminant functions in taxonomy. *Biometrics*, **18**, 455-477.

Examples

```
data(beetle)
#
# Here is the sample mean of this multivariate data set
#
apply(beetle, 2, mean)
# [1] 177.25676 123.95946  50.35135 134.81081  12.98649  95.37838
```

`getPP3index` *Extract projection index values from a PP3 object.*

Description

Extract project index value vector from a PP3 object.

Usage

```
getPP3index(x, number)
```

Arguments

`x`	PP3 object to get indices from.
`number`	Identification number of solution. Optional

Details

If `number` is missing then the whole projection index vector is returned. If `number` is specified then the projection index associated with that identification number is returned.

Value

Either a vector of indices, or a single index, depending on whether `number` is not specified or it is, respectively.

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

`getPP3loadings`, `getPP3projdata`, `PP3many`

Examples

```
#
# See help for PP3many, this contains an example of using getPP3index.
#
```

`getPP3loadings` *Extract projection loadings from a PP3 object.*

Description

Extract projection loadings (directions) from a PP3 object.

Usage

```
getPP3loadings(PP3, number)
```

Arguments

`PP3`	The PP3 object.
`number`	which projection solution you want to get.

Details

Gets projection directions associated with a particular solution from the `nrandstarts` that `PP3many` produces.

Value

A matrix of dimensional 3xK, where K was the original dimensionality of the input data set. Note: they are not an orthogonal set, but the back transform of an orthogonal set (back transformed from the inverse sphering transform).

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

`getPP3index`, `PP3many`

Examples

```
#
# See example for getPP3index in the help for PP3many, this function
# works similarly but returns loadings rather than indices.
#
```

---

getPP3projdata *Extract projected data from a PP3 object.*

Description

Extract projected data solution from a PP3 object.

Usage

```
getPP3projdata(PP3, number)
```

Arguments

`PP3`	The PP3 object you wish to get information from
`number`	The solution number you want to know about

Details

The projected data associated with a projection solution (out of the `nrandstarts` that were executed by `PP3many`)) can be extracted using this function.

Value

The three-dimensional projected data associated with projection solution `number`. This is a matrix of three rows (one for each projected direction) versus N columns, where N is the number original cases.

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

`PP3many`

Examples

```
#
# See example for getPP3index in the help for PP3many. This function
# works similarly except that projected observations are returned.
```

---

ix3sizeplot *Plot a histogram of projection index values*

Description

Plot a histogram of projection index values from a PP3 object, and superimpose control set of random projection indices for comparison.

Usage

```
ix3sizeplot(PP3manyobj, main = "Projection Index Value Histogram", nbig = 10)
```

Arguments

`PP3manyobj`	PP3 object to plot
`main`	Main title of plot
`nbig`	The number of big projection indices, and their identification numbers to plot.

Details

See the help page for plot.PP3 which calls this function.

Value

No specific return value, graphics output.

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

plot.PP3

Examples

```
#
# See example in PP3many, which calls plot.PP3, which calls this function
#
```

---

pdataplot *Plots the projected data for a particular random start.*

Description

This function should be called via the plot.PP3 function with a number argument specified.

Usage

```
pdataplot(PP3manyobj, number, colvec = 1, lab = NULL, ...)
```

Arguments

PP3manyobj	The PP3 object you want to plot projected data for.
number	The identification number that identifies which of the random start projection solutions you want to plot,
colvec	Optional vector of colour indices to plot for each observation.
lab	Labels, one for each observation. Optional.
...	Other arguments to the plot function.

Details

Described in more detail in the help for plot.PP3 function when the number argument is specified.

Value

No specific value

Author(s)

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

plot.PP3, PP3many

Examples

```
#
# See example in help for PP3many
#
```

plot.PP3 *Two types of plot for a PP3 object.*

Description

Either plot projected data (if number is specified) or histogram of projection indices over all random starts

Usage

```
## S3 method for class 'PP3'
plot(x, number, main, nbig = 10, colvec = 1, lab = NULL, ...)
```

Arguments

x	PP3 object that you wish to plot.
number	If missing, then the plot information concerning all of the project indices over all random starts (using function ix3sizeplot) or, if specified, plots the projected data associated with the number random start outcome (using function pdataplot.
main	Argument for main title.
nbig	If plotting information on all indices, this argument controls the number of the largest projection indices and their identification number.
colvec	If plotting the projected data, you can supply a vector of colour indices to colour each observation in the plot
lab	As for colvec, but permits a text label for each observation.
...	Other arguments to supply to the plot.

Details

This function can produce two types of plot. The first type, if the number argument is not supplied, produces a histogram of projection index values (in black) and superimposes the pseudo-index value density estimate, and median, upper quartile, 0.9 quantile and maximum values as vertical dotted lines.

If the number argument is specified, it should be one of the projection random start numbers and then the projected data associated with that maximised projection index is plotted. The points in the plot can be coloured (using colvec) or a text label supplied (using lab) argument.

Value

No specific value.

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

ix3sizeplot, pdataplot, PP3many

Examples

```
#
# See example in help for \code{\link{PP3many}}
#
```

`PP3init` *Initialize projection pursuit code*

Description

Initialize projection pursuit code. This function only need be executed once. Then subsequent calls to projection index calculation can reuse the results of this initialization many times.

Usage

```
PP3init(xm, action = 0, limit = 3)
```

Arguments

`xm`	Data matrix containing K rows (variables) and N columns (observations).
`action`	Possible outlier action. See help to PP3many for description.
`limit`	Possible outlier action. See help to PP3many for description.

Details

This function spheres the data and calculates third- and fourth-order moment quantities, which can be used for subsequent index calculation.

Value

A list with the following components.

COPYN	An integer less than or equal to the number of columns of the input matrix. The number of points that were kept after outlier processing (for index computation only).
ansT	The three-dimensional T summary statistic array
ansU	The four-dimensional U summary statistic array

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

PP3many

Examples

```
#
# Not designed for direct user use, but here is an example
#
#
# The flea beetle data
#
data(beetle)
#
# Initialise the PP3 system for this data
#
tmp <- PP3init(t(beetle))
#
# This object contains the ansT and ansU third- and fourth-order tensors
#
```

PP3ix3dvsFromTU *Compute the projection index or its derivatives from the T and U statistics.*

Description

Compute the projection index or its derivatives from the T and U statistics.

Usage

```
PP3ix3FromTU(the.init, avec, bvec, cvec, maxrow, k, maxcol, n, text)
PP3ix3dvsFromTU(the.init, avec, bvec, cvec, maxrow, k, maxcol, n, text, type = "value")
```

Arguments

the.init	Initialization information, such as from PP3init.
avec	First projection vector of length K
bvec	Second projection vector, of length K
cvec	Third projection vector, of length K
maxrow	Maximum number of variables
k	Actual number of variables (usually maxrow and k are the same.)
maxcol	Maximum number of observations
n	Actual number of observations. (usually maxcol and n are the same.)
text	Integer. If equal to one then information messages from the FORTRAN code are produced. If equal to zero, then they are not.
type	What type of information to return. If set to "deriv" then the derivative vector is returned. If set of "value" then the projection index is returned.

Details

The T and U third- and fourth-order moment related statistics are computed using PP3init. The projection index and its derivative are computed using these. This function can return either statistic. If you just want the index then you can supply the "value" argument, but it might be quicker to use PP3ix3FromTU which is called by the user-friendly PP3fastIX3.

Value

If type is set to "value" then a single value of the projection index is returned. If type is set to "deriv" then a vector, of length 3xK, with the derivative with respect to avec, bvec and cvec is returned.

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

PP3init, PP3fastIX3, PP3slowDF3

Examples

```
#
# Not for direct user use, but here is an example
#
#
# Load flea beetle data
#
data(beetle)
#
# Initialize T and U tensors
#
b.init <- PP3init(t(beetle))
#
# Get number of cases and dimensions
#
b.n <- nrow(beetle)
b.k <- ncol(beetle)
fortran.messages <- 0
#
# Select arbitrary projection vectors
#
b.pva <- c(1, rep(0, b.k-1))
b.pvb <- c(0, 1, rep(0, b.k-2))
b.pvc <- c(0, 0, 1, rep(0, b.k-3))
#
# Now compute the projection index for this data for this direction
#
answer <- PP3ix3FromTU(the.init=b.init, avec=b.pva, bvec=b.pvb,
cvec=b.pvc, maxrow=b.k, k=b.k, maxcol=b.n, n=b.n,
text=fortran.messages)
#
# Print out answer
#
answer
# [1] 13.49793
#
# Now compute the projection index derivatives for this data for this
```

```
# direction
#
answer <- PP3ix3dvsFromTU(the.init=b.init, avec=b.pva, bvec=b.pvb,
cvec=b.pvc, maxrow=b.k, k=b.k, maxcol=b.n, n=b.n,
text=fortran.messages, type="deriv")
#
# Print out answer
#
answer
# [1]  0.000000e+00  0.000000e+00  0.000000e+00 -1.283695e-15  0.000000e+00
# [6]  0.000000e+00  0.000000e+00 -4.649059e-16  0.000000e+00  2.910680e+00
#[11] -4.941646e+00 -1.232917e+00  1.057721e-01 -4.608611e+00 -8.286708e-01
#[16] -1.602697e-01  1.724654e+00 -2.029220e+00
#
# The answer is a vector of length 3xb.k = 18. The first b.k=6 entries
# correspond to the derivative wrt b.pva, the next b.k=6 entries to
# b.pvb, and the last b.k=6 entries to b.pvc.
```

PP3many *Main function to carry out three-dimensional projection pursuit.*

Description

Given a multivariate data set this function applies exploratory projection pursuit to find an interesting three-dimensional projection of the input data.

Usage

```
PP3many(xm, nrandstarts = 100, lapplyfn = lapply, action = 0, limit = 3, text = 0)
```

Arguments

xm	Data matrix. Note: ROWS correspond to variables, and so the input matrix here is (probably) the transpose of what you might expect (and is accepted by principal components functions). This might change in the future.
nrandstarts	Number of random starts.
lapplyfn	By default this is lapply. If you can use the parallel package then you can replace this with mclapply to get a speed up. The latter is faster if you know how to use the parallel package and you often need to set the `options(mc.cores=x)` option to a value of x giving the number of codes you wish to use.
action	Integer taking the value 0, 1, 2 or 3. This controls how multivariate outliers are treated. If 0, then no action is taken, the multivariate set is supplied unmolested to the projection pursuit routine. If 1, then outliers are removed. If 2,3 , then outliers are moved towards the centre according to log or square root trimming (the precise formulae are described in the trimsu FORTRAN code, and stem from Jones and Sibson (1987). The outliers are only removed, or trimmed, for the purposes of index calculation. They are retained in the final solution and projected according to the

projection direction decided by the projection pursuit algorithm computed on the non-outlier (majority) portion of the data.

`limit` This argument is only used if `action` is 1, 2 or 3 and outliers need to be dealt with. If observations are greater in distance than `limit` then they are considered to be outliers.

`text` Integer. Has no effect if set to zero. If set to one the FORTRAN code will print out informational messages.

Details

Exploratory projection pursuit is a method introduced by Friedman and Tukey (1974) which projects multivariate K-dimensional data down onto a L-dimensional subspace, using a projection A (an LxK-dimensional matrix). A projection index, I, is devised to measure how interesting the projection is and hence usually I=I(A). A numerical optimiser is then employed to find the projection A that maximises the index of interestingness.

A goal of exploratory projection pursuit is to find projections that highlight interesting structure and clustering. Often, this method can discover interesting structures that existing methods miss, as it uses a different measure of interestingness.

As with many other optimisation problems of this sort the numerical optimiser does not find the global maximum, but a local maximum. In any case, the global maximum might not correspond to the only interesting projection direction, so it is worth exploring many projections that result in large projection indices. One way of obtaining these is to run projection pursuit from several random starting directions. Although this does not guarantee to find all interesting local optima, it will find many of them.

This code implements the three-dimensional version of the moment projection index proposed by Jones and Sibson, (1987). Description of the three-dimensional version can be found in Nason (1995). This version first centres the data (removes its mean), and then spheres it (transforms its covariance matrix to the identity). The aim of sphering is to ensure that any structure discovered by projection pursuit is not related to anything that could be found by principal components analysis (because principal components "looks for structure" contained in the covariance matrix, and there will not be any if the covariance matrix is the identity). The moment index is an approximation to the entropy index, which looks for departures in the empirical distribution of the projected data from standard normality. The heuristic is that anything that is far from normal is 'interesting'. Hence, clustered, bi-, tri- or multimodal is not normal and hence interesting and so clustered data is deemed to be interesting by the moment index. The moment index has been criticised for being sensitive to outliers. More discussion of that, and designs of robust indices can be found in Nason (2001). However, recent numerical experimentation shows that the moment index indeed does find projections with outliers, but these can be easily and quickly discounted; and the method does routinely find interesting projections, when they are present.

Value

A `PP3` class object, which is a list with the following components.

`ix3` A vector containing the optimised projection index correspond to each of the `nrandstarts` random starts.

`info` A list of `nrandstarts` lists. Each item in the list contains a list with information obtained during the optimisation process. The components of each list are `par` the final optimisation parameters corresponding to the optimal projection direction (if an optimum was found); `value` the associated optimal projection index; `counts` the number of evaluations of the projection index and, separately, for the gradient; `convergence` a code indicating the outcome of the optimisation; `message` a message that describes the convergence outcome. The last three components are information produced by the R `optim` function and are described more fully there.

`pdata.list` The data projected according to the optimal projection direction resulting from each random start. A list containing `nrandstarts` matrices, each one corresponding to the optimisation resulting from each random start. Each matrix has three rows (corresponding to the three projection directions) and a number of columns equal to the number of multivariate cases (or observations, the number of columns of the input matrix). The `plot.PP3` function uses this information to produce plots of the data projected according projection solutions, when its `number` argument is specified.

`pseudp.vals` It can be hard to evaluate a given optimal projection index value and know whether it can be judged as large. One way of assessing it is to compare it to a set of projection indices computed (without optimisation) on another set of random projection directions. It is known that most arbitrary projections result in uninteresting projections, so a collection of `pseudp.vals` acts as a set of projection index control values (NOT p-values) to which the real `ix3` values can be compared to. The `plot.PP3` functoin uses the `pseudp.vals` to draw a red density estimate of the control values (and also plots median, upper quartile, 0.9 quantile, maximum) along with a histogram of the optimised `ix3` values, and one can easily see what should be considered to be large such values.

`origvarnames` The names of the original variables (names of rows) from the original data matrix. This is taken from the `dimnames` component of the data matrix.

Note

Nearly all of PP3 was written by Guy Nason for his PhD (1992). Nason is grateful too, and worked under the guidance of the late, great, Robin Sibson. Robin also wrote a carefully written eigendecomposition subroutine, which is part of this package.

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

getPP3index,getPP3loadings, getPP3projdata,plot.PP3, print.PP3, summary.PP3

Examples

```
#
# The flea beetle data
#
data(beetle)
#
# Run projection pursuit with 100 random starts (normally, you'd use MANY
# more random starts, e.g. 1000 or more. Here, we keep the number small to
# help CRAN
#
#
# N.b. I am going to set.seed here, so results match what you might see
# when trying THESE functions, but, in general, you can ignore set.seed
# or set it to your favourite value
#
set.seed(1)

beetle.PP3 <- PP3many(t(beetle), nrandstarts=100)
#
# Look at the output
#
beetle.PP3
#Class 'PP3' : Three-dimensional Projection Pursuit Object:
#  ~~~  : List with 5 components with names
# ix3 info pdata.list pseudp.vals origvarnames
#
#Number of random start(s):  100
#Maximum projection index is  22.02255  achieved by  1  random start(s).
#(Partial) list of those starts achieving max are:  90
#
#summary(.):
#----------
#   Summary statistics of projection index
#  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#       11.51   14.81   16.18   16.34   17.89   22.02
#   Summary statistics of pseudo p-values
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 8.592  10.885  12.361  12.466  13.992  18.210
#
# The print out shows that 100 random starts were executed and the max
# projection index was 22.02255 and only one of those random starts found
# this (sometimes more than one random start converges to the same maximum).
#
# The index number of the run which found the maximum was 90 (this number
# can be useful later to access the maximum).
#
# The summary gives the summary statistics of the 100 projection index
# values found. The max is the same as above, but also the distribution
```

```
# can be discerned.
#
# The distribution of the pseudo-p-values (NOT actual p-values) is presented
# after that. These are the projection indices computed purely on random
# directions, not the optimised versions and so you can think of them as
# null values to compare the earlier optimised values. E.g. the maximum of
# the pseudo-projection indices is 18.21, so any actual optimised projection
# index larger than this might be interesting.
#
# Now produce a plot (using all projection index info on 100 runs):
#
## Not run: plot(beetle.PP3)
#
# This produces (a) a histogram of the projection indices (b) a red density
# estimate of the pseudo-projection indices and (c) the median, upper quartile,
# 0.9 quantile and maximum of the pseudos as red dotted vertical lines. The red
# information corresponds to a kind of null, and so projection indices larger
# than these values might be interesting. The plot also produces some text:
#
#Big Projection Indices
#Maximum Psuedo p-value:  18.2103
#Index Number and associated projection indices
#      90       74        4       87       75       54       13        6
#22.02255 21.36578 21.29397 20.86531 19.59663 19.42427 19.34596 19.26520
#      23       60
#19.22810 19.16459
#
# This is a list of the 10 biggest projection indices and their respective
# identity numbers (which one of the random starts generated it). These
# can be used in the plot function with a number argument to generate further
# information/plots about the projection solution. Note, the number of
# biggest projection indices can be controlled with the nbig argument of
# plot.PP3.
#
# Now suppose we wanted to look at the projection solution 74, which had the
# second-biggest projection index. We can plot the projected data with the
# following command:
#
## Not run: plot(beetle.PP3, number=74, colvec=dimnames(beetle)[[1]])
#
# The colvec supplies the group structure so the different species can
# be coloured differently. The label argument permits you to put text
# labels there, per point as well as colours.
#
# You can extract information from the beetle.PP3 object using the
# extractor functions, getPP3index, getPP3projdata and the variable loadings
# using getPP3loading. For example,
#
getPP3index(beetle.PP3, 74)
#      74
#21.36578
#
# gets the second largest projection index. The third-largest can be obtained
```

```
# by replacing 74 by 4, etc.
```

PP3slowDF3 *Compute the projection index or its derivative.*

Description

Computes the projection index or its derivative with respect to the input projection directions. Function is a simple wrapper for call to PP3ix3FromTU or PP3ix3dvsFromTU.

Usage

```
PP3fastIX3(Pvec, the.init, maxrow, k, maxcol, n, text)
PP3slowDF3(Pvec, the.init, maxrow, k, maxcol, n, text)
```

Arguments

Pvec	The projection direction. Here, a three-dimensional projection direction (matrix with three columns) is stacked into a single vector.
the.init	Projection index initialization info. From the function PP3init
maxrow	Maximum number of rows (usually equal to k) or variables.
k	Actual number of rows/variables.
maxcol	Maximum number of observations (usually equal to n).
n	Number of observations.
text	Integer. If set to 1 then the FORTRAN code prints out information messages. If set to 0, then it doesn't.

Details

PP3fastIX3 computes the index only, and PP3slowDF3 computes the derivatives of the projection index with respect to the current projection direction (or, rather the Gram-Schmidt orthonormalised version). The word 'slow' does not mean slow, but refers to the fact that this routine also computes the projection index, but slowly because the derivatives are also being computed.

Value

PP3fastIX3 computes the projection index with respect to the input projection direction. PP3slowDF3 computes a numeric vector, of the same length as Pvec containing the derivative of the projection index with respect to every entry of Pvec.

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

PP3ix3dvsFromTU, PP3many

Examples

```
#
# Not designed for simple user use
#
# Since these functions are simple wrappers for PP3ix3FromTU and
# PP3ix3dvsFromTU, please consult their help functions. All these
# functions do is take a single projection vector and then split it into
# three to provide three separate projection vectors for the called
# functions.
#
```

---

print.PP3 *Print information about a PP3 object.*

Description

Print information about a PP3 object.

Usage

```
## S3 method for class 'PP3'
print(x, ...)
```

Arguments

x	PP3 object to print
...	Other arguments (not used)

Details

Prints information about a PP3 object. Primarily, the names of the components, the number of random starts that were used, the biggest (maximised) projection index that was found, and how many random starts achieved the max, and the index numbers of those. Then summary.PP3 is applied.

Value

No specific value

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

PP3many, summary.PP3

Examples

```
#
# The flea beetle data
#
data(beetle)
#
# Run projection pursuit with 10 random starts (usually MUCH more than this,
# but this example will be run on installation and testing and hence I
# want to minimize computational load. A more reasonable value is 1000)
#
beetle.PP3 <- PP3many(t(beetle), nrandstarts=10)
#
# Output from summary
#
print(beetle.PP3)
#Class 'PP3' : Three-dimensional Projection Pursuit Object:
#       ~~~  : List with 5 components with names
#              ix3 info pdata.list pseudp.vals origvarnames
#
#Number of random start(s):  10
#Maximum projection index is  20.39497  achieved by  1  random start(s).
#(Partial) list of those starts achieving max are:  9
#
#summary(.):
#----------
#Summary statistics of projection index
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  13.84   15.36   17.50   17.30   19.08   20.39
#Summary statistics of pseudo p-values
```

```
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  11.14   11.78   12.77   13.59   15.31   17.63
```

---

summary.PP3 *Print summary information about a PP3 object.*

Description

Print summary statistics about a PP3 object.

Usage

```
## S3 method for class 'PP3'
summary(object, ...)
```

Arguments

object	PP3 object
...	Other arguments (which aren't used)

Details

This applies the usual summary default function (which calculates summary statistics on a vector of values) to two vectors. The first is to the vector of maximised projection indices; the intention is so one can see what kinds of values the large ones take. The second application is to the pseudo projection indices, those computed on random directions without optimisation. Essentially, real projection indices that are larger than the maximum pseudo indices might be interesting and worth looking at with, e.g. the plot.PP3 function.

Value

Nothing explicit is returned

Author(s)

G. P. Nason

References

Friedman, J.H. and Tukey, J.W. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, **23**, 881-890.

Jones, M.C. and Sibson, R. (1987) What is projection pursuit? (with discussion) *J. R. Statist. Soc. A*, **150**, 1-36.

Nason, G. P. (1995) Three-dimensional projection pursuit. *J. R. Statist. Soc. C*, **44**, 411-430.

Nason, G. P. (2001) Robust projection indices. *J. R. Statist. Soc. B*, **63**, 551-567.

See Also

PP3many, plot.PP3

Examples

```
#
# The flea beetle data
#
data(beetle)
#
# Run projection pursuit with 10 random starts (usually MUCH more than this,
# but this example will be run on installation and testing and hence I
# want to minimize computational load. A more reasonable value is 1000)
#
beetle.PP3 <- PP3many(t(beetle), nrandstarts=10)
#
# Output from summary
#
summary(beetle.PP3)
#Summary statistics of projection index
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  13.84   15.36   17.50   17.30   19.08   20.39
#Summary statistics of pseudo p-values
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  11.14   11.78   12.77   13.59   15.31   17.63
```

Index