Package ‘REPPlab’

August 28, 2016

Type Package

Title R Interface to 'EPP-Lab', a Java Program for Exploratory Projection Pursuit

Version 0.9.4

Date 2016-08-23

Author Daniel Fischer, Alain Berro, Klaus Nordhausen, Anne Ruiz-Gazen

Maintainer Daniel Fischer <daniel.fischer@luke.fi>

Depends R (>= 2.15.1), rJava, lattice, LDRTools (>= 0.2), utils, graphics

Suggests tourr, amap

Description An R Interface to 'EPP-lab' v1.0. 'EPP-lab' is a Java program for projection pursuit using genetic algorithms written by Alain Berro and S. Larabi Marie-Sainte and is included in the package. The 'EPP-lab' sources are available under https://github.com/fischuu/EPP-lab.git.

License GPL (>= 2)

Encoding latin1

NeedsCompilation no

Repository CRAN

RoxygenNote 5.0.1

Date/Publication 2016-08-28 09:17:09

R topics documented:

REPPlab-package .................................................. 2
cof.epplab ...................................................... 3
EPPlab .......................................................... 4
EPPlabAgg ....................................................... 6
EPPlabOutlier .................................................... 8
fitted.epplab ................................................... 10
pairs.epplab .................................................... 11
plot.epplab ..................................................... 11
Description

An interface that gives access to the Java program 'EPP-lab' which implements several biologically inspired optimisation algorithms and several indices for Exploratory Projection Pursuit (PP). The objective of optimizing PP indices and projecting the data on the associated one-dimensional directions is to detect hidden structures such as clusters or outliers in (possibly high dimensional) data sets. The 'EPP-lab' sources and jar-files are available under https://github.com/fischuu/EPP-lab.git. For a detailed description of 'EPP-lab', see Larabi (2011).

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>REPPlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>0.9.4</td>
</tr>
<tr>
<td>Date</td>
<td>2016-08-23</td>
</tr>
<tr>
<td>License</td>
<td>GPL</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Daniel Fischer, Alain Berro, Klaus Nordhausen, Anne Ruiz-Gazen

Maintainer: Daniel Fischer <daniel.fischer@luke.fi>

References


---

**coef.epplab**

*Extracts the Directions of an Epplab Object*

### Description

Extracts the found directions of an epplab object.

### Usage

```r
## S3 method for class 'epplab'
coef(object, which = 1:ncol(object$PPdir), ...)
```

### Arguments

- **object**: Object of class epplab.
- **which**: Specifies which directions are extracted.
- **...**: Additional parameters.

### Details

The coef function extracts the directions found from the EPPlab call.

### Author(s)

Daniel Fischer

### Examples

```r
library(tourr)
data(olive)
res <- EPPlab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMin",n.simu=10, maxiter=20)
coef(res)
```
Description

REPPlab optimizes a projection pursuit (PP) index using a Genetic Algorithm (GA) or one of two Particle Swarm Optimisation (PSO) algorithms over several runs, implemented in the Java program EPP-lab. One of the PSO algorithms is a classic one while the other one is a parameter-free extension called Tribes. The parameters of the algorithms (maxiter and individuals for GA and maxiter and particles for PSO) can be modified by the user. The PP indices are the well-known Friedman and Friedman-Tukey indices together with the kurtosis and a so-called discriminant index that is devoted to the detection of groups. At each run, the function finds a local optimum of the PP index and gives the associated projection direction and criterion value.

Usage

REPPlab(x, PPindex = "KurtosisMax", PPalg = "GA", n.simu = 20, sphere = FALSE, maxiter = NULL, individuals = NULL, particles = NULL, step_iter = 10, eps = 10^(-6))

Arguments

- **x**: Matrix where each row is an observation and each column a dimension.
- **PPindex**: The used index, see details.
- **PPalg**: The used algorithm, see details.
- **n.simu**: Number of simulation runs.
- **sphere**: Logical, sphere the data. Default is FALSE, in which case the data is only standardized.
- **maxiter**: Maximum number of iterations.
- **individuals**: Size of the generated population in GA.
- **particles**: Number of generated particles in the standard PSO algorithm.
- **step_iter**: Convergence criterium parameter, see details. (Default: 10)
- **eps**: Convergence criterium parameter, see details. (Default: 10^(-6))

Details

The function always centers the data using `colMeans` and divides by the standard deviation. Spher- ing the data is optional. If spherung is requested the function `WhitenSVD` is used, which automatical- ly tries to determine the rank of the data.

Currently the function provides the following projection pursuit indices: KurtosisMax, Discriminant, Friedman, FriedmanTukey, KurtosisMin.

Three algorithms can be used to find the projection directions. These are a Genetic Algorithm GA and two Particle Swarm Optimisation algorithms PSO and Tribe.
Since the algorithms might find local optima they are run several times. The function sorts the found directions according to the optimization criterion.

The different algorithms have different default settings. It is for GA: `maxiter=50` and `individuals=20`. For PSO: `maxiter=20` and `particles=50`. For Tribe: `maxiter=20`.

For GA, the size of the generated population is fixed by the user (`individuals`). The algorithm is based on a tournament section of three participants. It uses a 2-point crossover with a probability of 0.65 and the mutation operator is applied to all the individuals with a probability of 0.05. The termination criterion corresponds to the number of generations and is also fixed by the user (`maxiter`).

For PSO, the user can give the number of initial generated particles and also the maximum number of iterations. The other parameters are fixed following Clerc (2006) and using a "cosine" neighborhood adapted to PP for the PSO algorithm. For Tribes, only the maximum number of iterations needs to be fixed. The algorithm proposed by Cooren and Clerc (2009) and adapted to PP using a "cosine neighborhood" is used.

The algorithms stop as soon as one of the two following conditions holds: the maximum number of iterations is reached or the relative difference between the index value of the present iteration and the value of iteration \( i - \text{step}\_\text{iter} \) is less than \( \text{eps} \). In the last situation, the algorithm is said to converge and `EPPlab` will return the number of iterations needed to attain convergence. If the convergence is not reached but the maximum number of iterations is attained, the function will return some warnings. The default values are 10 for `step\_iter` and \( 1\text{E}-06 \) for `eps`. Note that if few runs have not converged this might not be problem and even non-converged projections might reveal some structure.

**Value**

A list with class 'epplab' containing the following components:

- `PPdir` Matrix containing the PP directions as columns, see details.
- `PPindexVal` Vector containing the objective criterion value of each run.
- `PPindex` Name of the used projection index.
- `PPiter` Vector containing the number of iterations of each run.
- `PPconv` Boolean vector. Is TRUE if the run converged and FALSE else.
- `PPalg` Name of the used algorithm.
- `maxiter` Maximum number of iterations, as given in function call.
- `x` Matrix containing the data (centered!).
- `sphere` Logical
- `transform` The transformation matrix from the whitening or standardization step.
- `backtransform` The back-transformation matrix from the whitening or standardization step.
- `center` The mean vector of the data

**Author(s)**

Daniel Fischer, Klaus Nordhausen
References


Examples

```r
library(tourr)
library(amap)
data(olive)
olivePP <- EPPlab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMax",n.simu=5, maxiter=20)
summary(olivePP)

X <- lubisch[1:70,2:7]ownames(X) <- lubisch[1:70,1]
res <- EPPlab(X,PPalg="PSO",PPindex="FriedmanTukey",n.simu=15, maxiter=20,sphere=TRUE)
print(res)
summary(res)
fitted(res)
plot(res)
pairs(res)
predict(res, data=lubisch[71:74,2:7])
```

---

**EPPlabAgg**

*Function to Aggregate Directions From epplab Objects*

**Description**

Function that automatically aggregates the projection directions from one or more epplab objects. Three options are available on how to choose the final projection which can have a rank larger than one. The parameter `x` can either be a single object or a list of epplab objects. Options for `method` are `inverse`, `sq.inverse` and `cumulative`. 
Usage

EPPlabAgg(x, method = "cumulative", percentage = 0.85)

Arguments

x An object of class epplab or a list of epplab objects.
method The type of method, see details. Options are inverse, sq.inverse and cumulative.
percentage Threshold for the relative eigenvalue sum to retain, see details.

Details

Denote $p_i, i = 1, ..., m$, the projection vectors contained in the list of epplab objects and $P_i, i = 1, ..., m$, the corresponding orthogonal projection matrices (each having rank one). The method cumulative is based on the eigenvalue decomposition of $P_w = \frac{1}{m} \sum_{i=1}^{m} P_i$ and transforms as 0 the eigenvectors such that the corresponding relative eigenvalues sum is at least percentage. The number of eigenvectors retained corresponds to the rank $k$ and $P$ is the corresponding orthogonal projection matrix. The methods inverse and sq.inverse are automatic rules to choose the number of eigenvectors to retain as implemented by the function AOP.

Value

A list with class 'epplabagg' containing the following components:

- $P$ The estimated average orthogonal projection matrix.
- $O$ An orthogonal matrix on which $P$ is based upon.
- $k$ The rank of the average orthogonal projection matrix.
- eigenvalues The relevant eigenvalues, see details. Only given if method="cumulative".

Author(s)

Daniel Fischer, Klaus Nordhausen, Anne Ruiz-Gazen

References


See Also

EPPlab, AOP

Examples

library(tourr)
data(olive)
# To keep the runtime short, maxiter and n.simu were chosen very
# small for demonstration purposes, real life applications would
# rather choose larger values, e.g. n.simu=100, maxiter=200


```r
olivePP.kurt.max <- 
  EPPlab(olive[,3:10], PPalg="PSO", PPindex="KurtosisMax", n.simu=10, maxiter=20)

olivePP.fried <-
  EPPlab(olive[,3:10], PPalg="PSO", PPindex="Friedman", n.simu=10, maxiter=20)

olivePPs <- list(olivePP.kurt.max, olivePP.fried)

EPPlabAgg(olivePP.kurt.max)$k
EPPlabAgg(olivePPs, "cum", 0.99)$k

pairs(olivePP.kurt.max$x %>% EPPlabAgg(olivePPs, "cum", 0.99)$0, 
      col=olive[,2], pch=olive[,1])

olivAOP.sq <- EPPlabAgg(olivePPs, "inv")
oliveProj <- olivePP.kurt.max$x %>% olivAOP.sq$0
plot(density(oliveProj))
rug(oliveProj[olive$region==1], col=1)
rug(oliveProj[olive$region==2], col=2)
rug(oliveProj[olive$region==3], col=3)
```

---

**EPPlabOutlier**

*Function to Find Outliers for an epplab Object*

**Description**

Function to decide wether observations are considered outliers or not in specific projection directions of an epplab object.

**Usage**

```r
EPPlabOutlier(x, which = 1:ncol(x$PPdir), k = 3, location = mean, scale = sd)
```

**Arguments**

- `x` An object of class epplab.
- `which` The directions in which outliers should be searched. The default is to look at all.
- `k` Numeric value to decide when an observation is considered an outlier or not. Default is 3. See details.
- `location` A function which gives the univariate location as an output. The default is `mean`.
- `scale` A function which gives the univariate scale as an output. The default is `sd`. 
Details

Denote $location_j$ as the location of the $j$th projection direction and analogously $scale_j$ as its scale. Then an observation $x$ is an outlier in the $j$th projection direction, if $|x - location_j| \geq k \cdot scale_j$.

Naturally it is best to use for this purpose robust location and scale measures like median and mad for example.

Value

A list with class 'epplabOutlier' containing the following components:

- **outlier**: A matrix with only zeros and ones. A value of 1 classifies the observation as an outlier in this projection direction.
- **k**: The factor $k$ used.
- **location**: The name of the location estimator used.
- **scale**: The name of the scale estimator used.
- **PPindex**: The name of the PPindex used.
- **PPalg**: The name of the PPalg used.

Author(s)

Klaus Nordhausen

References


See Also

EPPlab

Examples

```r
# creating data with 3 outliers
n <- 300
p <- 10
X <- matrix(rnorm(n*p), ncol=p)
X[1,1] <- 9
X[2,4] <- 7
X[3,6] <- 8
# giving the data rownames, obs.1, obs.2 and obs.3 are the outliers.
rownames(X) <- paste("obs",1:n,sep=".")

PP <- EPPlab(X, PPalg="PSO", PPindex="KurtosisMax", n.simu=20, maxiter=20)
OUT <- EPPlabOutlier(PP, k = 3, location = median, scale = mad)
OUT
```
Description

Calculates the projections of the data object onto the directions from an underlying ebb1ab object.

Usage

```r
## S3 method for class 'epplab'
fitted(object, which = 1, ...)
```

Arguments

- `object`: Object of class `epplab`.
- `which`: Onto which direction should the new data be projected.
- `...`: Additional parameters

Details

The default projection direction is the direction with the best objective criterion.

Value

A matrix, having in each column the projection onto the direction of a certain run, and in each row the projected value.

Author(s)

Daniel Fischer

Examples

```r
library(tourr)
data(olive)
res <- epplab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMin",n.simu=10, maxiter=20)

# Projection to the best direction
fitted(res)

# Projection to the 1,2,5 best directions:
fitted(res,which=c(1,2,5))
```
Description

Plots a scatterplot matrix of fitted scores of an epplab object.

Usage

```r
## S3 method for class 'epplab'
pairs(x, which = 1:10, ...)
```

Arguments

- `x` Object of class `epplab`.
- `which` Which simulation runs should be taken into account.
- `...` Graphical parameters, see also `par()`.

Details

The option `which` can restrict the output to certain simulation runs. In case of many simulations, this might improve the readability.

Author(s)

Daniel Fischer

Examples

```r
library(tourr)
data(olive)
res <- EPLab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMin",n.simu=10, maxiter=20)
pairs(res)
```
Usage

```r
## S3 method for class 'epplab'
plot(x, type = "kernel", angles = "radians",
     kernel = "biweight", which = 1:10, as.table = TRUE, ...)
```

Arguments

- `x`: Object of class `epplab`
- `type`: Type of plot, values are "kernel", "histogram" and "angles".
- `angles`: Values are "degree" and "radians", if `type="angles"`.
- `kernel`: Type of kernel, passed on to `density`.
- `which`: Which simulation runs should be taken into account.
- `as.table`: A logical flag that controls the order in which panels should be displayed.
- `...`: Graphical parameters, see also `xyplot`, `densityplot` and `histogram`.

Details

The option `which` can restrict the output to certain simulation runs. In case of many simulations, this might improve the readability.

For `type="kernel"`, the default, it plots a kernel density estimate for each of the chosen directions.
In the case of `type="histogram"` the corresponding histograms. For `type="angles"` it plots the angles of the first chosen direction against all others. Whether the angles are given in degrees or radians, depends on the value of `angles`.

Author(s)

Daniel Fischer, Klaus Nordhausen

See Also

`xyplot`, `densityplot`, `histogram`, `density`

Examples

```r
library(tourr)
data(olive)
res <- EPPlab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMin",n.simu=10, maxiter=20)

# Plot with kernel estimator
plot(res)

# Just the best 5 and then 8
plot(res,which=c(1:5,8))

# Plot as histogram
plot(res,type="histogram")
```
# Plot an angles plot
plot(res,type="angles")

plot.epplabOutlier  

## Plot for an epplabOutlier Object

### Description
Visualizes which observations are considered as outliers and how often for an epplabOutlier object.

### Usage

```r
## S3 method for class 'epplabOutlier'
plot(x, col = c("white", "black"), outlier = TRUE, xlab = "", ylab = "", ...)  
```

### Arguments

- **x**: Object of class epplabOutlier.
- **col**: Which colors should be used for non-outliers and outliers. Default is white and black.
- **outlier**: Logical if only observations considered as outliers at least once should be plotted or all. Default is TRUE. xlab Possible x axis label. Default is none. ylab Possible x axis label. Default is none.
- **xlab**: Sets the x-axis label.
- **ylab**: Sets the y-axis label.
- **...**: Graphical parameters passed on to `image`.

### Author(s)
Daniel Fischer and Klaus Nordhausen

### See Also

- `epplabOutlier`, `image`

### Examples

```r
# creating data with 3 outliers
n <- 300
p <- 10
X <- matrix(rnorm(n*p),ncol=p)
X[1,1] <- 9
X[2,4] <- 7
```
predict.epplab

Calculates projections for a new Data Object

Description

Calculates the projections of a new data object onto the directions from an existing epplab object.

Usage

## S3 method for class 'epplab'
predict(object, which = 1, data = NULL, ...)

Arguments

- **object**: Object of class epplab.
- **which**: Onto which direction should the new data be projected.
- **data**: The new data object
- **...**: Additional parameters

Details

The default projection direction is the direction with the best objective criterion. In case that no data is given to the function, the fitted scores for the original data will be returned.

Value

A matrix having in each column the projection onto the direction of a certain run and in each row the projected value.

Author(s)

Daniel Fischer
Examples

library(tourr)
data(olive)
res <- EPPlab(olive[,3:10], PPalg="PSO", PPindex="KurtosisMin", n.simu=10, maxiter=20)

newData <- matrix(rnorm(80), ncol=8)
# Projection on the best direction
predict(res, data=newData)

# Projection on the best 3 directions
predict(res, which=1:3, data=newData)

# Similar with function fitted() when no data is given:
predict(res)
fitted(res)

print.epplab

Print an epplab Object

Description

Prints an epplab object.

Usage

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

Arguments

x Object of class epplab.

... Additional parameters

Details

The print function displays the result with the best value in the objective criterion.

Author(s)

Daniel Fischer
print.epplabOutlier

Examples

```r
library(tourr)
data(olive)
res <- EPPLab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMin",n.simu=10, maxiter=20)
print(res)
```

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

Arguments

- `x`: Object of class `epplabOutlier`.
- `...`: Additional parameters

Details

The `print` function displays only the outlier matrix from the `epplabOutlier` object.

Author(s)
Klaus Nordhausen

Examples

```r
# creating data with 3 outliers
n <- 300
p <- 10
x <- matrix(rnorm(n*p),nrow=n)
x[1,9] <- 9
x[2,7] <- 7
x[3,8] <- 8
# giving the data rownames, obs.1, obs.2 and obs.3 are the outliers.
rownames(x) <- paste("obs",1:n,sep=".")
PP<-EPPLab(X,PPalg="PSO",PPindex="KurtosisMax",n.simu=20, maxiter=20)
OUT<-EPPlabOutlier(PP, k = 3, location = median, scale = mad)
OUT
```
Description

The data set contains 55 variables measured during the production process of 520 units. The problem for this data is to identify the produced items with a fault not detected by standard tests.

Format

A data frame with 520 observations on 55 variables.

Source

The data was anonymized to keep confidentiality.

Examples

data(ReliabilityData)
summary(ReliabilityData)

screeplot.epplab

Creating a Screeplot for an epplab Object

Description

Plots the objective criteria of an epplab object versus the simulation runs.

Usage

## S3 method for class 'epplab'
sccreplot(x, type = "lines", which = 1:10, main = "", ylab = "Objective criterion", xlab = "Simulation run", ...)

Arguments

- `x` Object of class epplab.
- `type` Type of screeplot, values are "barplot" and "lines".
- `which` Which simulation runs should be taken into account.
- `main` Main title of the plot.
- `ylab` Y-axis label.
- `xlab` X-axis label.
- `...` Graphical parameters, see also par()
Details

The option which can restrict the output to certain simulation runs. In case of many simulations, this might improve the readability. The barplot option is often not meaningful, because the objective criteria are usually large and bars begin by default at zero.

Author(s)

Daniel Fischer

Examples

```r
library(tourr)
data(olive)
res <- EPPLab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMin",n.simu=10, maxiter=20)
screeplot(res)

# Pretty useless:
screeplot(res,type="barplot")
screeplot(res,which=1:5)
```

summary.epplab  
Summarize an epplab Object

Description

Summarizes and prints an epplab object in an informative way.

Usage

```r
## S3 method for class 'epplab'
summary(object, which = 1:10, digits = 4, ...)
```

Arguments

- `object` Object of class epplab.
- `which` Summary for which simulation runs
- `digits` Number of displayed decimal digits
- `...` Additional parameters

Details

The option which can restrict the output to certain simulation runs. In case of many simulations, this might improve the readability.
**summary.epplabOutlier**

**Author(s)**

Daniel Fischer

**Examples**

```r
library(tourr)
data(olive)
res <- EPPLab(olive[,3:10],PPalg="PSO",PPindex="KurtosisMin",n.simu=10, maxiter=20)
summary(res)
```

**Description**

Summarizes and prints an `epplabOutlier` object in an informative way.

**Usage**

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

**Arguments**

- `object` Object of class `epplabOutlier`.
- `...` Additional parameters

**Details**

The main information provided here is a table with names of the observations which are considered outliers and in how many PP directions they are considered outliers. This function is useful if the data has been given row names.

**Author(s)**

Klaus Nordhausen

**Examples**

```r
# creating data with 3 outliers
n <- 300
p <- 10
X <- matrix(rnorm(n*p),ncol=p)
X[1,1] <- 9
X[2,4] <- 7
```
WhitenSVD

Whitening Data Using Singular Value Decomposition

Description

The function whitens a data matrix using the singular value decomposition.

Usage

WhitenSVD(x, tol = 1e-06)

Arguments

x

A numeric data frame or data matrix with at least two rows.

tol

Tolerance value to decide the rank of the data. See details for further information. If set to NULL it will be ignored.

Details

The function whitens the data so that the result has mean zero and identity covariance matrix using the function svd. The data can have here less observations than variables and svd will determine the rank of the data automatically as the number of singular values larger than the largest singular value times tol. If tol=NULL the rank is set to the number of singular values, which is not advised when one or more singular values are close to zero.

The output contains among others as attributes the singular values and the matrix needed to back-transform the whitened data to its original space.

Value

The whitened data.

Author(s)

Klaus Nordhausen

See Also

svd, cov, colMeans
Examples

# more observations than variables
X <- matrix(rnorm(200), ncol=4)
A <- WhitenSVD(X)
round(colMeans(A), 4)
round(cov(A), 4)
# how to backtransform
summary(sweep(A %*% (attr(A, "backtransform")), 2, attr(A, "center"), "+") - X)

# fewer observations than variables
Y <- cbind(rexp(4), runif(4), rnorm(4), runif(4), rnorm(4), rt(4, 4))
B <- WhitenSVD(Y)
round(colMeans(B), 4)
round(cov(B), 4)
# how to backtransform
summary(sweep(B %*% (attr(B, "backtransform")), 2, attr(B, "center"), "+") - Y)
Index

*Topic datasets
ReliabilityData, 17

*Topic hplot
  pairs.epplab, 11
  plot.epplab, 11
  plot.epplabOutlier, 13
  screeplot.epplab, 17

*Topic methods
  coef.epplab, 3
  fitted.epplab, 10
  pairs.epplab, 11
  plot.epplab, 11
  plot.epplabOutlier, 13
  predict.epplab, 14
  print.epplab, 15
  print.epplabOutlier, 16
  summary.epplab, 18
  summary.epplabOutlier, 19

*Topic multivariate
  EPPlabAgg, 6
  EPPlabOutlier, 8
  REPPlab-package, 2
  WhitenSVD, 20

*Topic print
  coef.epplab, 3
  fitted.epplab, 10
  predict.epplab, 14
  print.epplab, 15
  print.epplabOutlier, 16
  summary.epplab, 18
  summary.epplabOutlier, 19

AOP, 7

cov, 20

density, 12
densityplot, 12

EPPlab, 4, 7, 9
EPPlabAgg, 6
EPPlabOutlier, 8, 13

fitted, epplab-method (fitted.epplab), 10
fitted-method (fitted.epplab), 10
fitted.epplab, 10

histogram, 12

image, 13

mad, 9

mean, 8

median, 9

pairs, epplab-method (pairs.epplab), 11
pairs-method (pairs.epplab), 11
pairs.epplab, 11
plot.epplab-method (plot.epplab), 11
plot.epplabOutlier-method
  (plot.epplabOutlier), 13
plot-method (plot.epplab), 11
plot.epplab, 11
plot.epplabOutlier, 13
predict, epplab-method (predict.epplab), 14
predict-method (predict.epplab), 14
predict.epplab, 14
print, epplab-method (print.epplab), 15
print, epplabOutlier-method
  (print.epplabOutlier), 16
print-method (print.epplab), 15
print.epplab, 15
print.epplabOutlier, 16

R/EPP-lab-package (REPPlab-package), 2
ReliabilityData, 17
REPPlab-package, 2

screeplot.epplab-method
  (screeplot.epplab), 17
screeplot-method(screeplot.epplab), 17
screeplot.epplab, 17
sd, 8
summary.epplab-method(summary.epplab), 18
summary.epplabOutlier-method
  (summary.epplabOutlier), 19
summary-method(summary.epplab), 18
summary.epplab, 18
summary.epplabOutlier, 19
svd, 20
WhitenSVD, 4, 20

xyplot, 12