Package 'eigenprcomp'

February 19, 2015

Type Package

Title Computes confidence intervals for principal components

Version 1.0

Date 2013-07-26

Author Francisco Juretig

Maintainer Francisco Juretig <fjuretig@yahoo.com>

Description Computes confidence intervals for the proportion explained by the first 1,2,k principal components, and computes confidence intervals for each eigenvalue. Both computations are done via nonparametric bootstrap.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2013-07-27 11:42:23

R topics documented:

boot_pr_comp 1

Index **5**

`boot_pr_comp` *Function to compute the confidence intervals for the proportion explained by each eigenvalue, and confidence intervals for each eigenvalue.*

Description

The proportion explained by each eigenvalue and confidence intervals for each eigenvalue are computed via nonparametric bootstrap. When doing PCA, a natural question that arises when choosing the amount of components is whether the chosen set of eigenvalues conform a well defined set. If two eigenvalues are statistically equal, then the space spanned by them is not correctly defined, so consequently none of them should be included in the set of chosen eigenvalues. Ideally, the chosen

set should include only eigenvalues that are statistically different between them, and also different respect to the other eigenvalues. On the other hand, it is usually important to compute a confidence interval for the proportion explained by the chosen set which is also addressed in this function.

Usage

```
boot_pr_comp(valores, size, alpha, plot = TRUE)
```

Arguments

valores	A matrix with the dataset should be specified.This dataset should have the observations as rows and the variables as columns.
size	The amount of boostrap replicates.
alpha	The alpha, which is by default 0,05.
plot	A boolean value indicating whether the plot of the eigenvalues CIs should be produced.

Value

proportions_quantiles
: The upper and lower quantile for the explained proportion. Note that for k variables, only k-1 proportions are needed since k eigenvalues always explains the total variance

proportions_used
: The boostrapped proportions used for the computation. It can be used to produce histograms

eigen_quantiles
: The upper and lower quantile for the eigenvalues

eigen_used
: The used eigenvalues for the eigenvalues

Author(s)

Francisco Juretig

Examples

```
boot_pr_comp(as.matrix(iris[,1:4]))

function(valores,size=1000,alpha=0.05,plot=TRUE){

if (is.matrix(valores)==FALSE){

return ("Oject is not a matrix")

  }else{

lengs               = length(valores[,1])
hlengs              = length(valores[1,])
store_bootstrap     = matrix(0,size,hlengs)
```

```
elements_aux        = NULL
indexes             = seq(1,lengs,1)
work_matrix         = matrix(0,lengs,hlengs)
empirical_quantiles1= matrix(0,hlengs-1,2)
store_eigen         = matrix(0,size,hlengs)
empirical_quantiles2= matrix(0,hlengs,2)

for (sim in 1:size){

resample_indexes= sample(indexes,length(indexes), replace=T)

for (ips in 1:length(resample_indexes)){
work_matrix[ips,] = valores[resample_indexes[ips],]
  }

cova           = (lengs-1)*cov(work_matrix)/(lengs)
auto_va        = unlist(eigen(cova)$values)
cum_eigen      = cumsum(auto_va)
total_var      = sum(auto_va)
percent        = (cum_eigen/total_var)

store_bootstrap[sim,] = percent
store_eigen[sim,]     = auto_va
  }

store_bootstrap      = store_bootstrap[,-hlengs]

for (iter in 1:(hlengs-1)){
empirical_quantiles1[iter,2]    = quantile(store_bootstrap[,iter],1-alpha/2)
empirical_quantiles1[iter,1]    = quantile(store_bootstrap[,iter],alpha/2)
  }

for (iter in 1:(hlengs)){
empirical_quantiles2[iter,2]    = quantile(store_eigen[,iter],1-alpha/2)
empirical_quantiles2[iter,1]    = quantile(store_eigen[,iter],alpha/2)
  }

if (plot==TRUE){

plot(0:max( empirical_quantiles2[,2]),0:max( empirical_quantiles2[,2]),
xlim=c(0,max( empirical_quantiles2)),ylim=c(0,nrow( empirical_quantiles2)),
type="n",xlab="Values",ylab="Eigenvalues")

for (q in 1:nrow(empirical_quantiles2)){
segments( empirical_quantiles2[q,1],q, empirical_quantiles2[q,2],q,lwd=2)
abline(v= empirical_quantiles2[q,1],lty = 3)
  }

  }

colnames(empirical_quantiles1)        = c(paste(alpha/2*100,"%"),
paste((1-alpha/2)*100,"%"))
colnames(empirical_quantiles2)        = c(paste(alpha/2*100,"%"),
```

```
paste((1-alpha/2)*100,"%"))
lista  = list("proportions_quantiles" = empirical_quantiles1,
"proportions_used" = store_bootstrap,
"eigen_quantiles"= empirical_quantiles2,"eigen_used"=store_eigen)

return(lista)

  }
  }
```

Index

∗Topic **Eigenvalues**
boot_pr_comp, 1
∗Topic **PCA**
boot_pr_comp, 1

boot_pr_comp, 1