Description

Auer and Gervini developed a Bayesian graphical method to determine the number \( d \) of significant principal components; a brief overview is included in the help for the \texttt{AuerGervini} class. The output of their method is a step function that displays the maximum a posteriori (MAP) choice of \( d \) as a step function of a one-parameter family of prior distributions, and they recommend choosing the highest "long" step. The functions described here help automate the process of dividing the step lengths into "long" and "short" classes.
Usage

agDimTwiceMean(stepLength)
agDimKmeans(stepLength)
agDimKmeans3(stepLength)
agDimSpectral(stepLength)
agDimTtest(stepLength, extra=0)
agDimTtest2(stepLength)
agDimCPT(stepLength)
makeAgCpmFun(method)

Arguments

stepLength A numeric vector
method A character string describing a method supported by the detectChangePointBatch function in the cpm package.
extra Just ignore this. Don’t use it. It’s a hack to avoid having to maintain two different versions of the same code.

Details

The agDimTwiceMean function implements a simple and naive rule: a step is considered long if it is at least twice the mean length.

The agDimKmeans uses the kmeans algorithm with $k = 2$ to divide the step lengths into two classes. Starting centers for the groups are taken to be the minimum and maximum values.

The agDimKmeans3 function uses kmeans with $k = 3$, using the median as the third center. Only one of the three groups is considered "short".

The agDimSpectral applies spectral clustering (as implemented by the specc function from the kernlab package) to divide the step lengths into two groups.

The agDimTtest and agDimTtest2 functions implement two variants of a novel algorithm specialized for this particular task. The idea is to start by sorting the step lengths so that

$$L_1 \leq L_2 \leq \ldots \leq L_n.$$ 

Then, for each $i \in \{3, \ldots, N - 1\}$, we compute the mean and standard deviation of the first $i$ step lengths. Finally, one computes the likelihood that $L_{i+1}$ comes from the normal distribution defined by the first $i$ lengths. If the probability that $L_{i+1}$ is larger is less than 0.01, then it is chosen as the "smallest long step".

The novel method just described can also be viewed as a way to detect a particular kind of change point. So, we also provide the agDimCPT function that uses the changepoint detection algorithm implemented by the cpt.mean function in the changepoint package. More generally, the makeAgCpmFun allows you to use any of the changepoint models implemented as part of the detectChangePointBatch function in the cpm package.

Value

Each of the functions agDimTwiceMean, agDimKmeans, agDimKmeans3, agDimSpectral, agDimTtest, agDimTtest2, and agDimCPT returns a logical vector whose length is equal to the input stepLength. TRUE values identify "long" steps and FALSE values identify "short" steps.

The makeAgCpmFun returns a function that takes one argument (a numeric stepLength vector) and returns a logical vector of the same length.
AuerGervini-class

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Min Wang <wang.1807@osu.edu>.

References


See Also

The functions described here implement different algorithms that can be used by the agDimension function to automatically compute the number of significant principal components based on the AuerGervini approach. Several of these functions are wrappers around functions defined in other packages, including specc in the kernlab package, cpt.mean in the changepoint package, and detectChangePointBatch in the cpm package.

Examples

# simulate variances
lambda <- rev(sort(diff(sort(c(0, 1, runif(9))))))
# apply the Auer-Gervini method
ag <- AuerGervini(lambda, dd=c(3,10))
# Review the results
summary(ag)
agDimension(ag)
agDimension(ag, agDimKmeans)
agDimension(ag, agDimSpectral)
f <- makeAgCpmFun("Exponential")
agDimension(ag, f)

AuerGervini-class

Estimating Number of Principal Components Using the Auer-Gervini Method

Description

Auer and Gervini [1] described a graphical Bayesian method for estimating the number of statistically significant principal components. We have implemented their method in the AuerGervini class, and enhanced it by automating the final selection.

Usage

AuerGervini(Lambda, dd=NULL, epsilon = 2e-16)
agDimension(object, agfun=agDimTwiceMean)

Arguments

Lambda

Either a SamplePCA object or a numerical vector of variances from a principal components analysis.

dd

A vector of length 2 containing the dimensions of the data used to created the Auer-Gervini object. If Lambda is a SamplePCA object, then the dimensions are taken from it, ignoring the dd argument.
AuerGervini-class

| epsilon   | A numeric value. Used to remove any variances that are less than epsilon; defaults to 2e-16. Should only be needed in rare cases where negative variances show up because of round-off error. |
| object    | An object of the AuerGervini class. |
| agfun     | A function that takes one argument (a vector of step lengths) and returns a logical vector of the same length (where true indicates "long" as opposed to "short" steps). |

Details

The Auer-Gervini method for determining the number of principal components is based on a Bayesian model that asserts that the vector of explained variances (eigenvalues) should have the form

\[ a_1 \leq a_2 \leq \ldots \leq a_d < a_{d+1} = a_{d+2} = \ldots = a_n \]

with the goal being to find the true dimension \( d \). They consider a set of prior distributions on \( d \in \{1, \ldots, n\} \) that decay exponentially, with the rate of decay controlled by a parameter \( \theta \). For each value of \( \theta \), one selects the value of \( d \) that has the maximum a posteriori (MAP) probability. Auer and Gervini show that the dimensions selected by this procedure write \( d \) as a non-increasing step function of \( \theta \). The values of \( \theta \) where the steps change are stored in the changePoints slot, and the corresponding \( d \)-values are stored in the dLevels slot.

Auer and Gervini go on to advise using their method as a graphical approach, manually (or visually?) selecting the highest step that is "long". Our implementation provides several different algorithms for automatically deciding what is "long" enough. The simplest (but fairly naive) approach is to take anything that is longer than twice the mean; other algorithms are described in agDimFunction.

Value

The AuerGervini function constructs and returns an object of the AuerGervini class. The agDimension function computes the number of significant principal components. The general idea is that one starts by computing the length of each step in the Auer-Gervini plot, and must then separate these into "long" and "short" classes. We provide a variety of different algorithms to carry out this process; the default algorithm in the function agDimTwiceMean defines a step as "long" if it more than twice the mean step length.

Objects from the Class

Objects should be created using the AuerGervini constructor.

Slots

- Lambda: A numeric vector containing the explained variances in decreasing order.
- dimensions: Numeric vector of length 2 containing the dimensions of the underlying data matrix.
- dLevels: Object of class numeric; see details
- changePoints: Object of class numeric; see details

Methods

- plot signature(x = "AuerGervini", y = "missing"); ...
- summary signature(object = "AuerGervini"); ...
The Broken Stick Method

Description

The Broken Stick model is one proposed method for estimating the number of statistically significant principal components.

Usage

brokenStick(k, n)
bsDimension(lambda, FUZZ = 0.005)

Arguments

k
n
lambda
FUZZ

An integer between 1 and n.
An integer; the total number of principal components.
The set of variances from each component from a principal components analysis. These are assumed to be already sorted in decreasing order. You can also supply a SamplePCA object, and the variances will be automatically extracted.
A real number; anything smaller than FUZZ is assumed to equal zero for all practical purposes.
The Broken Stick model is one proposed method for estimating the number of statistically significant principal components. The idea is to model $N$ variances by taking a stick of unit length and breaking it into $N$ pieces by randomly (and simultaneously) selecting break points from a uniform distribution.

Value

The `brokenStick` function returns, as a real number, the expected value of the $k$-th longest piece when breaking a stick of length one into $n$ total pieces. Most commonly used via the idiom `brokenStick(1:N, N)` to get the entire vector of lengths at one time.

The `bsDimension` function returns an integer, the number of significant components under this model. This is computed by finding the last point at which the observed variance is bigger than the expected value under the broken stick model by at least FUZZ.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References


See Also

Better methods to address this question are based on the Auer-Gervini method; see `AuerGervini`.

Examples

```r
brokenStick(1:10, 10)
sum(brokenStick(1:10, 10))
fakeVar <- c(30, 20, 8, 4, 3, 2, 1)
bsDimension(fakeVar)
```

---

**compareAgDimMethods**  
*Compare Methods to Divide Steps into "Long" and "Short"*

**Description**

Auer and Gervini developed a Bayesian graphical method to determine the number $d$ of significant principal components; a brief overview is included in the help for the `AuerGervini` class. The output of their method is a step function that displays the maximum a posteriori (MAP) choice of $d$ as a step function of a one-parameter family of prior distributions, and they recommend choosing the highest "long" step. The functions described here help automate the process of dividing the step lengths into "long" and "short" classes.

**Usage**

```r
compareAgDimMethods(object, agfuns)
```
**Arguments**

object: An object of the `AuerGervini` class

agfuns: A list of functions

**Details**

This method simply iterates over the list of functions that implement different algorithms/methods to determine the PC dimension.

**Value**

Returns an integer vector of the same length as the list of `agfuns`, containing the number of significant principal components computed by each method.

**Author(s)**

Kevin R. Coombes <krc@silicovore.com>, Min Wang <wang.1807@osu.edu>.

**References**


**See Also**

`AuerGervini`, `agDimension`.

**Examples**

```r
# simulate variances
lambda <- rev(sort(diff(sort(c(0, 1, runif(9)))))),
# apply the Auer-Gervini method
ag <- AuerGervini(lambda, dd=c(3,10))
# try different methods
agfuns <- list(twice=agDimTwiceMean,
    km=agDimKmeans,
    cpt=agDimCPT)
compareAgDimMethods(ag, agfuns)
```

---

**Description**

TODO: Stuff goes here...

**Usage**

`rndLambdaF(data, B = 1000, alpha = 0.05, ...)`
Arguments

data  TODO
B     TODO
alpha TODO
...  TODO

Details
TODO

Value
TODO

Author(s)
Kevin R. Coombes <krc@silicovore.com>, Min Wang <wang.1807@osu.edu>.

References
TODO

spca-data  Sample PCA Dataset

Description
This data set contains an object of the class \link{SamplePCA}. This object results from performing a principal components analysis on a simulated data set where the true

Usage
data(spca)

Format
A \code{SamplePCA} object based on a simulated data matrix with 204 rows and 14 columns, with true "principal component dimension" equal to one. That is, there should be one significant principal component.

Source
Simulations are described in detail in the \pkg{Thresher} package, which depends on the \pkg{PCDimension} package.

See Also
The \pkg{ClassDiscovery} package contains the \code{SamplePCA} class and functions.
Index

*Topic **classes**
  AuerGervini-class, 3

*Topic **cluster**
  brokenStick, 5

*Topic **datasets**
  spca-data, 8

*Topic **models**
  agDimFunction, 1
  AuerGervini-class, 3
  brokenStick, 5
  compareAgDimMethods, 6
  rndLambdaF, 7
  agDimCPT (agDimFunction), 1
  agDimension, 3, 7
  agDimension (AuerGervini-class), 3
  agDimFunction, 1, 4, 5
  agDimKmeans (agDimFunction), 1
  agDimKmeans3 (agDimFunction), 1
  agDimSpectral (agDimFunction), 1
  agDimTtest (agDimFunction), 1
  agDimTtest2 (agDimFunction), 1
  agDimTwiceMean, 4
  agDimTwiceMean (agDimFunction), 1
  AuerGervini, 1, 3, 6, 7
  AuerGervini (AuerGervini-class), 3
  AuerGervini-class, 3
  brokenStick, 5
  bsDimension (brokenStick), 5
  compareAgDimMethods, 6
  cpt.mean, 2, 3
  detectChangePointBatch, 2, 3
  kmeans, 2
  makeAgCpmFun (agDimFunction), 1
  plot, AuerGervini, missing-method
    (AuerGervini-class), 3
  rndLambdaF, 7
  SamplePCA, 5, 8
  spca (spca-data), 8
  spca-data, 8
  specc, 2, 3
  summary, AuerGervini-method
    (AuerGervini-class), 3