Package 'GPrank'

August 17, 2018

Title Gaussian Process Ranking of Multiple Time Series

Version 0.1.4

Date 2018-08-17

Depends R (>= 2.14.0)

Imports gptk, matrixStats, tigreBrowserWriter, RColorBrewer

LazyData true

Description Implements a Gaussian process (GP)-based ranking method which can be used to rank multiple time series according to their temporal activity levels. An example is the case when expression levels of all genes are measured over a time course and the main concern is to identify the most active genes, i.e. genes which show significant non-random variation in their expression levels. This is achieved by computing Bayes factors for each time series by comparing the marginal likelihoods under time-dependent and time-independent GP models. Additional variance information from pre-processing of the observations is incorporated into the GP models, which makes the ranking more robust against model overfitting. The package supports exporting the results to 'tigreBrowser' for visualisation, filtering or ranking.

License MIT + file LICENSE

URL https://github.com/PROBIC/GPrank

BugReports https://github.com/PROBIC/GPrank/issues

RoxygenNote 6.0.1

NeedsCompilation no

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Hande Topa [aut, cre],
Antti Honkela [aut]

Maintainer Hande Topa <hande.topa@helsinki.fi>

Repository CRAN

Date/Publication 2018-08-17 15:00:03 UTC

R topics documented:

`GPrank-package` *GPrank - Gaussian Process ranking of multiple time series*

Description

GPrank package implements our Gaussian-process-based ranking method, which can be used to rank multiple time series according to their temporal activity levels. GPrank incorporates available variance information on the observations into the GP models and achieves a higher performance than the standard GP-based ranking methods.

Details

Package:	GPrank
Type:	Package
Version:	0.1.2
Date:	2016-12-21
License:	MIT

Details regarding the usage of the package can be found in the vignette.

Author(s)

Hande Topa, Antti Honkela

Maintainer: Hande Topa <hande.topa@helsinki.fi>

References

Hande Topa, Agnes Jonas, Robert Kofler, Carolin Kosiol, Antti Honkela. Gaussian process test for high-throughput sequencing time series: application to experimental evolution. *Bioinformatics* 31(11):1762-1770, **2015**.

Hande Topa, Antti Honkela. Analysis of differential splicing suggests different modes of short-term splicing regulation. *Bioinformatics* 32(12):i147-i155, **2016**.

`apply_gpTest` *Performing Gaussian process test*

Description

Function for fitting time-dependent and time-independent GP models and computing the Bayes factors.

Usage

```
apply_gpTest(x, y, v, nullModelKernelTypes = c("white", "fixedvariance"),
  modelKernelTypes = c("rbf", "white", "fixedvariance"), y_fitted = FALSE)
```

Arguments

`x`	One-column matrix which contains the input values, i.e., time points for GP models. The values given in this vector are used in GP model, so if any transformation is needed, remember to transform them now.
`y`	One-column matrix which contains the observed values at the corresponding time points given in `x` vector.
`v`	One-column matrix which contains the fixed variances at the corresponding time points given in `x` vector, with the corresponding observations given in `y` vector.
`nullModelKernelTypes`	Character vector which contains the type of the kernels which will be used in the null, i.e., time-independent GP model. Default is c("white","fixedvariance").
`modelKernelTypes`	Character vector which contains the type of the kernels which will be used in the time-dependent GP model. Default is c("rbf","white","fixedvariance"). Kernel types: `'rbf'`, `'white'`, `'fixedvariance'`. Note that the lower bound for the length scale parameter of rbf kernel is set to the minimum distance between consecutive time points in order to mitigate potential overfitting problems.
`y_fitted`	Logical variable indicating whether the fitted y values at the observed time points will be given or not. Default is set to FALSE.

Value

Return list which contains logged Bayes factors (logBF) and the fitted GP models (nullModel & model) with the specified kernel structures. If y_fitted is set to TRUE, fitted y values of the model are returned in y_fitted as the list element.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
x=as.matrix(seq(1,10))
y=as.matrix(sin(x))
v=as.matrix(runif(10,0,0.5))
nullModelKernelTypes=c("white","fixedvariance")
modelKernelTypes=c("rbf","white","fixedvariance")
test_result=apply_gpTest(x,y,v,nullModelKernelTypes,modelKernelTypes,y_fitted=TRUE)
null_model=test_result$nullModel
model=test_result$model
logBF=test_result$logBF
y_fitted=test_result$y_fitted
```

---

bbgp_snpData *Obtaining counts data in the format of example snpData by using the sample data file.*

Description

Function for extracting the required data for the experimental evolution application.

Usage

```
bbgp_snpData(dataFileName, noHeaderLines = 1, noInfoColumns = 3,
  noOptions = 6, sep = ":")
```

Arguments

dataFileName	Name of the raw data file
noHeaderLines	Number of header lines, set to 1 by default.
noInfoColumns	Number of columns which contain information about the SNP locations. These columns are used to construct SNP IDs.
noOptions	Number of possible alterations separated by the character defined in "sep" Default is 6, assuming the form A:T:C:G:N:Del
sep	Character which separates the alterations. Default is set to ":".

Value

List of snpData which contains counts in the "counts" matrix and sequencing depth values in the "seq_depth" matrix. SNP IDs are stored as row names, and time points are stored as column names.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
# dataFileName="sampleCountsData"
# snpData=bbgp_snpData(dataFileName)
```

`bitseq_fitGPs` *Fitting GP models for the BitSeq example*

Description

Function for fitting two GP models and computing Bayes factors, i.e., the ratio of the maximum marginal likelihood estimates of the two GP models, where the models are:

- null model: GP with only white noise covariance matrix
- alternative model: GP with squared exponential and white noise covariance matrices

Optionally, log Bayes factors and the parameter estimates can be written to output files whose names are specified with fileName_logBF, fileName_ModelParams, and fileName_NullModelParams.

Usage

```
bitseq_fitGPs(gpData, fileName_logBF = NULL, fileName_ModelParams = NULL,
  fileName_NullModelParams = NULL)
```

Arguments

`gpData`	for example, output from `bitseq_rnaSeqData`.
`fileName_logBF`	name of the file which contains log Bayes factors.
`fileName_ModelParams`	name of the file which contains model parameters.
`fileName_NullModelParams`	name of the file which contains null model parameters.

Value

List of results

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

See Also

`bitseq_rnaSeqData`

Examples

```
## Not run:
RNAseqDATA
gpData=RNAseqDATA$reltr
GPfits=bitseq_fitGPs(gpData)

## End(Not run)
```

---

bitseq_plotGP *Plotting fitted GP models for the BitSeq example*

Description

Function for plotting GP profiles. If the item to be plotted has multiple items which are associated with itself, its items can be plotted on top of each other by setting `multi` to 1, for better visual comparison. Log Bayes factors are displayed in legends for at most three items which have the largest log Bayes factors.

Usage

```
bitseq_plotGP(item, GPfits, gpData, multi = 0, ylimits = NULL,
  x_ticks = NULL, x_label = NULL, y_label = NULL, plotName = NULL,
  colScale = getColorVector())
```

Arguments

`item`	Name of the item whose GP profile to be plotted.
`GPfits`	List structure obtained by `bitseq_fitGPs` which contains the time-independent and time-dependent GP models for specified (?) item.
`gpData`	List structure obtained by, for example, `bitseq_rnaSeqData` function.
`multi`	Indicator value for specifying whether multiple plots (1) or a single plot (0) will be plotted on the same frame. Default value is 0.
`ylimits`	Numeric vector which contains minimum and maximum limits for the y axis.
`x_ticks`	X-axis tick labels.
`x_label`	X-axis label.
`y_label`	Y-axis label.
`plotName`	Name of the plot containing png or pdf extension. png is recommended for the plots which will be displayed on the browser, whereas pdf is recommended for the plots which will be used in printed documents. The default plotting settings are specified in `bitseq_setPlot` function. If you would like to use your own preferred settings, remember to specify them on your own.
`colScale`	RGB color code of the plot. If not specified, colors are determined by default.

Value

GP plot(s) for the given item.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

See Also

getColorVector bitseq_setPlot bitseq_fitGPs bitseq_rnaSeqData

Examples

```
## Not run:
RNAseqDATA
gpData=RNAseqDATA$reltr
GPfits=bitseq_fitGPs(gpData)
item="ARAP2"
multi=1
ylimits=c(0,1)
x_ticks=c("0","5","10","20","40","80","160","320","640","1280")
x_label="Time (minutes)"
y_label="Expression level (rpkm)"
plotName="ARAP2_reltr.pdf"
bitseq_plotGP(item, GPfits, gpData, multi, ylimits, x_ticks, x_label, y_label, plotName)

## End(Not run)
```

---

bitseq_rnaSeqData *Obtaining data in the format of example RNAseqDATA by using BitSeq output files*

Description

Function for obtaining the means and standard deviations at the available time points.

Usage

```
bitseq_rnaSeqData(t, trFileName)
```

Arguments

t	Vector which contains the input values, i.e., time points. The file names for the corresponding time point is specified as the names of this vector.
trFileName	Name of the transcriptome file.

Value

List of GP data within the right structure.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
# t=log(c(0,5,10,20,40,80,160,320,640,1280)+5)
# names(t)=c("t0000.rpkm","t0005.rpkm","t0010.rpkm","t0020.rpkm","t0040.rpkm",
# "t0080.rpkm","t0160.rpkm","t0320.rpkm","t0640.rpkm","t1280.rpkm")
# trFileName="example_tr"
# bitseq_rnaSeqData(t,trFileName)
```

---

constructModel *Constructing GP model with the specified kernels*

Description

Function for constructing the GP model with the specified kernels (and parameters).

Usage

```
constructModel(x, y, v, kernelTypes, params = NULL)
```

Arguments

x	One-column matrix which contains the input values, i.e., time points for GP models. The values given in this vector are used in GP model, so if any transformation is needed, remember to transform them before constructing the model.
y	One-column matrix which contains the observed values at the corresponding time points given in the x vector.
v	One-column matrix which contains the fixed variances at the corresponding time points given in the x vector, with the corresponding observations given in the y vector.
kernelTypes	Character vector which contains the types of the kernels which will be used in the GP models. Kernel types: 'rbf', 'white', 'bias', 'fixedvariance'. Note that the lower bound for the length scale parameter of rbf kernel is set to the minimum distance between consecutive time points in order to mitigate potential overfitting problems.
params	Values of the kernel parameters in their transformed form. If not specified, default values are assigned.

Value

Return GP model constucted with the specified kernel settings.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
x=as.matrix(seq(1,10))
y=as.matrix(sin(x))
v=as.matrix(runif(10,0,0.5))
kernelTypes=c("rbf","white","fixedvariance")
model=constructModel(x,y,v,kernelTypes)
```

---

createDatabase *Building SQLite database*

Description

Function for building an SQLite database for displaying GP profiles of the selected items on a browser, enabling to rank them according to their Bayes factors and other provided parameters. For details of using tigreBrowser, please refer to https://github.com/PROBIC/tigreBrowser and. https://github.com/PROBIC/tigreBrowserWriter.

Usage

```
createDatabase(dbInfo, figs)
```

Arguments

dbInfo	List which contains the required information of the items that will be included in the database. Three arguments must be specified in the dbInfo: • database_name: Name of the database. • database_params: List of parameters which will be included in the database. • identifiers: Identifiers for the items which will be displayed on the browser. These identifiers should appear in the begining of the corresponding figure names, followed by an underscore and the type of the plot. For example, for identifier "geneA", multiple figures may be named as "geneA_gene.png", "geneA_abstr.png", and "geneA_reltr.png".
figs	Character vector containing the figure names.

Value

Generates an SQLite database named "$database_name.sqlite".

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
BF=c(3,10,2)
FoldChange=c(0.5,3,5)
dbParams=list("BF"=BF,"Fold change"=FoldChange)
identifiers=c("geneA","geneB","geneC")
dbInfo=list(database_name="testdb","database_params"=dbParams,"identifiers"=identifiers)
figs=c("geneA_gene.png","geneA_abstr.png","geneA_reltr.png","geneB_gene.png",
"geneB_abstr.png","geneB_reltr.png","geneC_gene.png","geneC_abstr.png","geneC_reltr.png")
for (i in seq(1,9)) {
examplefig=figs[i]
     png(examplefig)
     plot(c(0, 1), c(0, 1))
     dev.off()
}
createDatabase(dbInfo,figs)
```

---

getColorVector *Extracting distinctive colors from RColorBrewer package*

Description

Function for obtaining a vector of RGB color codes where the colors are as distinctive as possible from each other. This function extracts colors from Paired (12 colors), Set2 (8 colors), and Dark2 (8 colors) palettes provided in the RColorBrewer package. If you need more than 28 colors, please remember to add your own palette.

Usage

```
getColorVector(opacity = 0.7, display = FALSE)
```

Arguments

opacity	Factor modifying the opacity, valued between [0,1]. Default is set to 0.7.
display	By default set to FALSE; if TRUE, displays the color palette.

Value

Return vector of 28 distinctive RGB color codes.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
color_vector=getColorVector()
```

`getModelKernParams` *Getting the values of the kernel parameters of the GP model*

Description

Function for getting the kernel parameter values of the GP model. In order to get the transformed values, set transformed to TRUE.

Usage

```
getModelKernParams(model, transformed = FALSE)
```

Arguments

`model`	GP model
`transformed`	Logical variable indicating whether the transformed values of the parameters are desired or not. Default is set to FALSE.

Value

Return vector of values of the kernel parameters of the GP model.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
x=as.matrix(seq(1,10))
y=as.matrix(sin(x))
v=as.matrix(runif(10,0,0.5))
kernelTypes=c("rbf","white","fixedvariance")
model=constructModel(x,y,v,kernelTypes)
params=getModelKernParams(model)
```

getYlimits *Setting limits for the y-axis*

Description

Function for setting decent y-axis limits.

Usage

```
getYlimits(Y, V, predY = NULL, predV = NULL, minRng = 0.5)
```

Arguments

Y	Matrix containing observed values; items in rows, time points in columns.
V	Matrix containing variances; items in rows, time points in columns.
predY	Matrix containing predicted values; items in rows, time points in columns.
predV	Matrix containing variances of the predicted values; items in rows, time points in columns.
minRng	Minimum range for the y axis in the plot. Default value is set to 0.5.

Value

Return y-axis limits.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
Y=matrix(c(1,2,3,4,5,6),2,3)
V=0.1*Y
predY=matrix(c(1,2,3,4,5,6),2,3)
predV=0.1*predY
y_lims=getYlimits(Y,V,predY,predV)
```

get_bbgpMeanStd *Computing means and standard deviations for the BBGP (beta binomial Gaussian process) model*

Description

Function for obtaining the posterior means and standard deviations for the frequencies (counts divided by sequencing depth) by using the counts and sequencing depth values in a beta binomial model. Parameters (alpha and beta) of the model are set to 1 by default, which keeps symmetry between f and (1-f), where f denotes the frequency valued between (0,1).

Usage

```
get_bbgpMeanStd(x, counts, seq_depth, alpha = 1, beta = 1)
```

Arguments

x	Time vector
counts	Vector containing the counts data at the given time points.
seq_depth	Vector containing the sequencing depth values at the given time points.
alpha	alpha parameter of the beta binomial model.
beta	beta parameter of the beta binomial model.

Value

Return list containing the posterior means and standard deviations of the frequencies at the time points where sequencing depth is larger than zero. x vector is updated so that it excludes the time points with zero sequencing depth, i.e. time points at which no data have been observed. Posterior means and standard deviations and the updated x vecor are assigned to the list elements named 'posteriorMean', 'posteriorStd', and 'time', respectively.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
x=c(1,2,3,4,5)
counts=c(12,54,32,0,34)
seq_depth=c(50,70,35,0,40)
bbgp=get_bbgpMeanStd(x,counts,seq_depth)
x=bbgp$time # updated time vector
y=bbgp$posteriorMean # posterior means
v=bbgp$posteriorStd^2 # posterior variances
```

plotGP *Plotting fitted GP models*

Description

Function for plotting fitted GP model within its confidence region of 2 standard deviations.

Usage

```
plotGP(model, col_item = "gray", ylimits = NULL, write_xticks = TRUE,
  write_yticks = TRUE, jitterx = FALSE)
```

Arguments

model	GP model to be plotted.
col_item	RGB color code which will be used for the color of the GP plot.
ylimits	Numeric vector which contains minimum and maximum limits for the y axis.
write_xticks	Boolean: whether to write x ticks and labels or not
write_yticks	Boolean: whether to write y ticks and labels or not
jitterx	Boolean: whether to jitter duplicated x values or not

Value

Creates the plot of the fitted GP model.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
x=as.matrix(seq(1,10))
y=as.matrix(sin(x))
v=as.matrix(runif(10,0,0.5))
kernelTypes=c("rbf","white","fixedvariance")
model=constructModel(x,y,v,kernelTypes)
col_item=getColorVector()[1]
ylimits=c(min(y)-0.1,max(y)+0.1)
plotGP(model,col_item,ylimits)
```

RNAseqDATA *Sample data obtained from example BitSeq output files*

Description

The original data was introduced in (Honkela et al., 2015) and can be accessed in Gene Expression Omnibus (GEO) database (www.ncbi.nlm.nih.gov/geo) with accession no. GSE62789.

Usage

```
data(RNAseqDATA)
```

Format

An object of class `list` of length 3.

Details

This data set contains mean and standard deviation information on the expression levels of 5 transcripts (which were originated from 2 genes) at 10 time points (0, 5, 10, 20, 40, 80, 160, 320, 640, 1280 mins) for three settings: 'gene', 'abstr' (absolute transcript), and 'reltr' (relative transcript) expression levels. In addition, the fields 'gene_mapping' and 'time_mapping' includes information which is useful to match the genes with transcripts and the time points with data files, respectively.

References

Antti Honkela, Jaakko Peltonen, Hande Topa, Iryna Charapitsa, Filomena Matarese, Korbinian Grote, Hendrik G. Stunnenberg, George Reid, Neil D. Lawrence, Magnus Rattray. Genome-wide modeling of transcription kinetics reveals patterns of RNA production delays. *PNAS* 112(42):13115-13120, **2015**.

Examples

```
data(RNAseqDATA)
gpData=RNAseqDATA$gene
gpData=RNAseqDATA$abstr
gpData=RNAseqDATA$reltr
```

setInitParams *Initializing kernel parameters*

Description

Function for setting initial parameters to reasonable values after performing a grid search over parameters within their pre-set ranges. Loglikelihood is computed for each combination of parameter values on the grid, and those which lead to the highest loglikelihood value are set as initial values.

Usage

```
setInitParams(model, grid_size = 5)
```

Arguments

model	GP model.
grid_size	Size of the grid over which the search for maximum likelihood will be done. Default value is 5.

Value

Return GP model with the parameters initialized at reasonable values.

Author(s)

Hande Topa, <hande.topa@helsinki.fi>

Examples

```
x=as.matrix(seq(1,10))
y=as.matrix(sin(x))
v=as.matrix(runif(10,0,0.5))
kernelTypes=c("rbf","white","fixedvariance")
model=constructModel(x,y,v,kernelTypes)
model=setInitParams(model)
```

snpData *Sample data for demonstrating the application of experimental evolution.*

Description

This sample data contains 5 replicates of counts and sequencing depth information for 5 SNPs at the generations (0, 10, 20, 30, 40, 50, 60).

Usage

```
data(snpData)
```

Format

An object of class `list` of length 2.

Examples

```
data(snpData)
counts=snpData$counts
seq_depth=snpData$seq_depth
```

Index