

# Package ‘Oscope’

April 12, 2018

**Type** Package

**Title** Oscope - A statistical pipeline for identifying oscillatory genes in unsynchronized single cell RNA-seq

**Version** 1.8.0

**Date** 2015-7-28

**Author** Ning Leng

**Maintainer** Ning Leng <lengning1@gmail.com>

**Depends** EBSeq, cluster, testthat, BiocParallel

**Suggests** BiocStyle

**Description** Oscope is a statistical pipeline developed to identifying and recovering the base cycle profiles of oscillating genes in an unsynchronized single cell RNA-seq experiment. The Oscope pipeline includes three modules: a sine model module to search for candidate oscillator pairs; a K-medoids clustering module to cluster candidate oscillators into groups; and an extended nearest insertion module to recover the base cycle order for each oscillator group.

**License** Artistic-2.0

**Collate** 'AbsCor.R' 'NormForSine.R' 'SineFun.R' 'FormatSineOut.R' 'Opt2Shift.R' 'SineOptim.R' 'PipeR.R' 'ImpShift.R' 'PipeShiftCDF.R' 'scanK.R' 'NISFun.R' 'CalcMV.R' 'OscopeKM.R' 'OscopeENI.R' 'OscopeSine.R' 'FlagCluster.R' 'PermuCut.R'

**BuildVignettes** yes

**biocViews** StatisticalMethod,RNASeq, Sequencing, GeneExpression

**NeedsCompilation** no

## R topics documented:

Oscope-package . . . . .	2
AbsCor . . . . .	3
CalcMV . . . . .	3
FlagCluster . . . . .	4
FormatSineOut . . . . .	5
ImpShift . . . . .	6
NISFun . . . . .	7

NormForSine . . . . .	8
Opt2Shift . . . . .	9
OscopeENI . . . . .	10
OscopeExampleData . . . . .	11
OscopeKM . . . . .	11
OscopeSine . . . . .	12
PermuCut . . . . .	13
PipeR . . . . .	14
PipeShiftCDF . . . . .	15
scanK . . . . .	16
SineFun . . . . .	17
SineOptim . . . . .	18

<b>Index</b>	<b>19</b>
--------------	-----------

---

Oscope-package	<i>Oscope - A statistical pipeline for identifying oscillatory genes in unsynchronized single cell RNA-seq</i>
----------------	--

---

## Description

Oscope is a statistical pipeline developed to identifying and recovering the base cycle profiles of oscillating genes in an unsynchronized single cell RNA-seq experiment. The Oscope pipeline includes three modules: a sine model module to search for candidate oscillator pairs; a K-medoids clustering module to cluster candidate oscillators into groups; and an extended nearest insertion module to recover the base cycle order for each oscillator group.

## Details

Package: Oscope  
 Type: Package  
 Version: 0.99.0  
 Date: 2015-7-27  
 License: Artistic-2.0

## Author(s)

Ning Leng

Maintainer: Ning Leng <lengning1@gmail.com>

## References

Leng et al. Oscope - A statistical pipeline for identifying oscillatory genes in unsynchronized single cell RNA-seq, accepted

---

AbsCor	<i>Calculate absolute correlations among gene pairs</i>
--------	---

---

**Description**

Calculate absolute correlations among gene pairs

**Usage**

```
AbsCor(DataIn, method="pearson", diagNA=TRUE)
```

**Arguments**

DataIn	input data, gene-by-sample matrix
method	"pearson" or "spearman"; default is "pearson"
diagNA	whether replace diagonal values to NA's

**Value**

Output is a gene-by-gene matrix; the  $i, j$  th entry shows the absolute correlation of the  $i$ th and  $j$ th gene.

**Author(s)**

Ning Leng

**Examples**

```
AbsCor(matrix(rnorm(10),ncol=5))
```

---

CalcMV	<i>Calculate estimated mean and variance of RNA-Seq data</i>
--------	--

---

**Description**

Calculate estimated mean and variance of RNA-Seq data

**Usage**

```
CalcMV(Data, Sizes=NULL, NormData=FALSE, MeanCutLow=100, MeanCutHigh=NULL, ApproxVal=10^-6, Plot=
```

**Arguments**

Data	input data matrix; it should be a gene-by-sample or isoform-by sample matrix
Sizes	The library size factor for each sample. the number of values in Sizes is expected to be the same as the number of columns of Data. The library size factor will be estimated using the median normalization method implemented in EBSeq if Sizes is specified as NULL.
NormData	whether the data is already normalized. If NormData=TRUE, the specification of Sizes will be ignored and no normalization will be applied.
MeanCutLow, MeanCutHigh	we suggests the users to apply Oscope on genes with high mean and high variance. By default, MeanCutLow is specified as 100, consequently only genes with mean > 100 will be used. The CalcMV function will fit a linear regression on $\log(\text{variance}) \sim \log(\text{mean})$ on these genes. Genes with variance above this line are considered as the high mean high variance genes. The upper bound of mean may be specified using MeanCutHigh. If both are specified as NULL, all of the genes will be considered when fitting the regression.
ApproxVal	Default is $10^{-6}$ . It is used to approximate the estimate of parameter q for genes/isoforms whose estimated variance is less than estimated mean. q will be estimated using $1 - \text{ApproxVal}$
Plot	if Plot = T, a mean-variance plot will be shown. The fitted line will be shown and the selected genes will be marked in green.

**Value**

Output is a list with 6 sublists : Mean: estimated means of genes/isoforms; Var: estimated variances; Median: estimated medians; GeneToUse: the high mean high variance genes (suggested input for Oscope); Q: estimated q's (without approximation); Q\_mdf: estimated q's with approximations; Phi\_mdf: estimated overdispersion parameter (phi), with approximations.

**Author(s)**

Ning Leng

**Examples**

```
exp=matrix(rnorm(100,1000,10),ncol=10)
rownames(exp)=paste0("g",1:10)
CalcMV(exp)
```

---

FlagCluster

*Flag gene clusters with small within-cluster phase differences and/or small within-cluster sine scores*

---

**Description**

Flag gene clusters with small within-cluster phase differences and/or small within-cluster sine scores

**Usage**

```
FlagCluster(SineRes, KMRes, Data, qt, thre=pi/4, qtincluster=.5, qtinpermu=.9 ,Seed=1)
```

**Arguments**

SineRes	output of OscopeSine() function
KMRes	output of KMRes() function
Data	a gene-by-sample (isoform-by-sample) matrix indicating the rescaled expression of genes/isoforms. all values should be between [-1, 1].
qt, thre	Define a gene pair's linear score as $\min(\eta, \pi - \eta)$ , in which $\eta$ is defined as phase shift mod $\pi$ . A cluster will be flagged if the qt th quantile of within-cluster linear score is less than thre.
qtincluster, qtinpermu	To define clusters with small within-cluster sine scores, for each cluster we generate permuted data of these genes (different cell permutation for each gene). We calculate the within-cluster sine scores within the cluster of permuted genes, then infer whether the sine scores in the cluster of interest are greater than those generated by the permuted genes. A cluster will be flagged if its qtincluster th quantile in the original data is less than its qtinpermu th quantile in permuted data.
Seed	seed

**Value**

Output: RemoveID: a vector of cluster numbers that are flagged; SineCompreList: sine score and permuted sine score for each cluster; LinearList: linear score of each cluster

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
tmp <- matrix(sin(rnorm(330)),ncol=11)
rownames(tmp) <- paste0("tmp",1:30)
Dat <- rbind(aa, bb, cc, tmp)
res1 <- OscopeSine(Dat)
res2 <- OscopeKM(res1, quan=.8, maxK=5)
res <- FlagCluster(res1, res2, Dat)
```

---

FormatSineOut

*Format SinFun outputs from lists to matrix*


---

**Description**

Format SinFun outputs from lists to matrix

**Usage**

```
FormatSineOut(result, DataInSc, ShiftRg=pi/4)
```

**Arguments**

result	Output from SineFun
DataInSc	a gene-by-sample (isoform-by-sample) matrix indicating the rescaled expression of two genes/isoforms. all values should be between [-1, 1].
ShiftRg	phase shift cutoff.

**Value**

Output is a list with 4 sublists, each shows a N-by-N matrix, in which# N is the total number of genes (isoforms). SimiMat: similarity matrix (sine scores); the sine scores are calculated by  $-\log_{10}(\epsilon^2)$ . DiffMat: dissimilarity matrix; shown are  $\epsilon^2$  for each gene pair. ShiftMat: optimal phase shift estimate for each pair of genes.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
DataInSc <- rbind(aa,bb,cc)
NumGene <- nrow(DataInSc)
Res <- sapply(1:(NumGene-1),function(i)SineFun(DataInSc, i),simplify=FALSE)
Out <- FormatSineOut(Res, DataInSc)
```

---

ImpShift	<i>Search for the optimal sample order by using the Extended Nearest Insertion</i>
----------	--

---

**Description**

Search for the optimal sample order by using the Extended Nearest Insertion

**Usage**

```
ImpShift(Data, Seq=NULL, NChun=4, RdmStart=FALSE, Ndg=3)
```

**Arguments**

Data	gene-by-sample matrix or isoform-by-sample matrix.It should be rescaled to values between [-1,1].
Seq	NULL or a vector indicates the sample order. if specified, the samples will be first reordered by this vector.
NChun	number of starting points for polynomial fitting.
RdmStart	whether the start points are randomly selected.
Ndg	degree of polynomial.

**Value**

This function performs the extended nearest insertion (ENI). The ENI algorithm searches for the optimal sample order which minimizes the MSE of sliding polynomial regression (SPR). This function will call PipeShiftCDF() function, which fits SPR to each row of the data. For each gene/isoform, SPR fits NChun polynomial curves with different starting points (samples). The samples with smaller order than the start point will be appended to follow the last sample when fitting. So each fitting consider same number of samples. If RdmStart = TRUE, the start points are randomly selected. Otherwise they are evenly sampled along the sample order. The aggregated MSE of a fit (using a specific start point) is defined as the summation of the MSEs of all genes/isoforms considered here. The MSE of the SPR is defined as the largest aggregated MSE across fits using different start points. The output returns the optimal order which provides the smallest SPR MSE.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
res <- ImpShift(rbind(aa,bb,cc), NChun=2)
```

---

NISFun

*Run Extended Nearest Insertion and 2-opt on a gene cluster identified by OscopeKM function*

---

**Description**

Run Extended Nearest Insertion and 2-opt on a gene cluster identified by OscopeKM function

**Usage**

```
NISFun(ClusterList, DataIn, i, Ndg=3, NChun=4, RdmStart=FALSE, N=20000, NCThre=1000)
```

**Arguments**

ClusterList	a list of gene clusters. Each sublist contains a vector of gene names.
DataIn	gene-by-sample matrix or isoform-by-sample matrix. It should be rescaled to values between [-1,1].
i	the cluster of interest. If the second cluster in ClusterList is of interest, specify i=2.
Ndg	degree of polynomial.
NChun	number of starting points for polynomial fitting.
RdmStart	whether the start points are randomly selected.
N, NCThre	The 2-opt algorithm will stop if N iterations has been performed or if the optimal order remains unchanged for over NCThre iterations.

**Value**

This function performs the extended nearest insertion (ENI) and 2-opt algorithm to a particular cluster identified by `OscopeKM` function. The ENI algorithm searches for the optimal sample order which minimizes the MSE of sliding polynomial regression (SPR). This function will call `PipeShiftCDF()` function, which fits SPR to each row of the data. For each gene/isoform, SPR fits NChun polynomial curves with different starting points (samples). The samples with smaller order than the start point will be appended to follow the last sample when fitting. So each fitting consider same number of samples. If `RdmStart = TRUE`, the start points are randomly selected. Otherwise they are evenly sampled along the sample order. The aggregated MSE of a fit (using a specific start point) is defined as the summation of the MSEs of all genes/isoforms considered here. The MSE of the SPR is defined as the largest aggregated MSE across fits using different start points. The output of `PipeShiftCDF()` returns the optimal order which provides the smallest SPR MSE. The 2-opt algorithm is then applied to improve the optimal order searching of the ENI. In each iteration, 2-opt algorithm will randomly choose two points (samples), the flip the samples between these two points. The new order will be adapted if it provides smaller SPR MSE. The output returns the optimal order for the cluster of interest.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
res <- NISFun(list(c("aa","bb"),"cc"), rbind(aa,bb,cc),i=1, NChun=2, N=50)
```

---

NormForSine

*Rescale the gene/isoform expression matrix*

---

**Description**

Rescale the gene/isoform expression matrix

**Usage**

```
NormForSine(Data, qt1=.05, qt2=.95)
```

**Arguments**

<code>Data</code>	input gene-by-sample matrix or isoform-by-sample matrix
<code>qt1, qt2</code>	thresholds for outlier adjustment. For each gene/isoform, values $\leq$ qt1 th quantile ( $\geq$ qt2 th quantile) will be pushed to qt1 th quantile (qt2 th quantile) prior to the scaling. default values are 0.05 and 0.95.

**Value**

The output will be a gene-by-sample or isoform-by-sample matrix. For each gene/isoform, the expressions will be scaled linearly to  $[-1,1]$



**Author(s)**

Ning Leng

**Examples**

```
NormForSine(matrix(rnorm(10), nrow=2))
```

---

 Opt2Shift

*Run the 2-opt algorithm to improve the optimal order searching of the Extended Nearest Insertion*

---

**Description**

Run the 2-opt algorithm to improve the optimal order searching of the Extended Nearest Insertion

**Usage**

```
Opt2Shift(Data, N=20000, Seq, Ndg=3, NChun=4, NCThre=1000, RdmStart=FALSE)
```

**Arguments**

Data	gene-by-sample matrix or isoform-by-sample matrix. It should be rescaled to values between [-1,1].
N, NCThre	The 2-opt algorithm will stop if N iterations has been performed or if the optimal order remains unchanged for over NCThre iterations.
Seq	a vector indicates the sample order obtained from the ENI.
Ndg	degree of polynomial.
NChun	number of starting points for polynomial fitting.
RdmStart	whether the start points are randomly selected.

**Value**

This function performs the the 2-opt algorithm to improve the optimal order searching of the Extended Nearest Insertion (ENI). In each iteration, the function will randomly choose two points (samples), the flip the samples between these two points. The new order will be adapted if it provides smaller SPR MSE. The output returns the optimal order and its SPR MSE.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
res <- ImpShift(rbind(aa,bb,cc), NChun=2)
res2 <- Opt2Shift(rbind(aa,bb,cc), NChun=2, N=50, Seq=res)
```

OscopeENI

*Search for the optimal sample order for different gene clusters***Description**

Search for the optimal sample order for different gene clusters

**Usage**

```
OscopeENI(KMRes, Data, ClusterUse=NULL, Ndg=3, NChun=4, RdmStart=FALSE,
N=20000, NCThre=1000, parallel=FALSE, parallelParam=NULL)
```

**Arguments**

KMRes	output of OscopeKM() function.
Data	gene-by-sample matrix or isoform-by-sample matrix. It should be rescaled to values between [-1,1].
ClusterUse	a vector indicating what clusters are of interest. For example, by setting ClusterUse = c(1,2,3), only the top 3 clusters will be considered while recovering the base cycle order. If ClusterUse=NULL, all clusters will be used.
Ndg	degree of polynomial.
NChun	number of starting points for polynomial fitting.
RdmStart	whether the start points are randomly selected.
N, NCThre	The 2-opt algorithm will stop if N iterations has been performed or if the optimal order
parallel	whether apply parallel computing. if it is TRUE, BiocParallel will be called.
parallelParam	a SnowParam object to specify the clusters. If it is NULL, the default will be set as SnowParam(workers = 5, type = "SOCK") remains unchanged for over NCThre iterations.

**Value**

This function performs the extended nearest insertion (ENI) and 2-opt algorithm to all clusters (or a subset of picked clusters) identified by OscopeKM function. The function will recover independent orders to each of the clusters. For each cluster, the ENI algorithm will be applied to search for the optimal sample order which minimizes the MSE of sliding polynomial regression (SPR). This function will call PipeShiftCDF() function, which fits SPR to expression of each gene/isoform within a cluster. For each gene/isoform, SPR fits NChun polynomial curves with different starting points (samples). The samples with smaller order than the start point will be appended to follow the last sample when fitting. So each fitting consider same number of samples. If RdmStart = TRUE, the start points are randomly selected. Otherwise they are evenly sampled along the sample order. The aggregated MSE of a fit (using a specific start point) is defined as the summation of the MSEs of all genes/isoforms considered here. The MSE of the SPR is defined as the largest aggregated MSE across fits using different start points. The output of PipeShiftCDF() returns the optimal order which provides the smallest SPR MSE. The 2-opt algorithm will then be applied to improve the optimal order searching of the ENI. In each iteration, the 2-opt algorithm will randomly choose two points (samples), the flip the samples between these two points. The new order will be adapted if it provides smaller SPR MSE. The output returns the optimal order for each cluster of interest. It is a list with multiple sublists, in which each sublist includes the recovered order of the corresponding

cluster in ClusterUse. If ClusterUse is not specified, the k th sublist shows the recovered order in KMRes

### Author(s)

Ning Leng

### Examples

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
dd <- sin(seq(1.2,2.2,.1))
res <- OscopeENI(list(c1=c("aa","bb"),c2=c("cc","dd")), rbind(aa,bb,cc,dd), NChun=2, N=50)
```

---

OscopeExampleData	<i>Simulated gene level data set with 600 genes and 30 cells.</i>
-------------------	---

---

### Description

Simulated gene expression to evaluate Oscope. 600 genes and 30 cell are simulated. The expression mean of the genes are randomly simulated in the range of 10-10000.

### Format

GeneExampleData is a matrix with 600 genes (rows) and 30 cells (columns).

### Examples

```
data(OscopeExampleData)
str(OscopeExampleData)
```

---

OscopeKM	<i>Oscope K medoid module</i>
----------	-------------------------------

---

### Description

Oscope K medoid module

### Usage

```
OscopeKM(SineRes, quan=.95,cut=NULL,maxK=NULL,minSize=0, maxSize=200, fixK=NULL, rawscale=TRUE)
```

**Arguments**

SineRes	output of OscopeSine function.
quan	only gene pairs with similarity score $\geq$ quan th quantile will be considered in the clustering analyses. Default is 0.95.
cut	pre-defined cutoff. Gene pairs with similarity score $\geq$ cut will be considered in cluster analyses. If cut is defined, quan will be ignored.
maxK	max number of clusters to consider (scan). if numbC=NULL, it will be calculated as [number of gene considered]/10
minSize,maxSize	Only clusters with minSize $\leq$ cluster size $\leq$ maxSize are reported in output.
fixK	if fixK is specified, the k-medoids algorithm will be applied with fixK clusters.
rawscale	Recall the input is the similarity matrix ( $-\log_{10}(\text{distance from the sine model})$ ). the k-medoids clustering will be applied using (-Input) as distance. If rawscale is defined as TRUE, the k-medoids clustering will be applied using $-10^{\text{Input}}$ as distance.

**Value**

OscopeKM() calls scanK() function, which runs k-medoid clustering with varying number of clusters (k). The k is varied from 2 to maxK. The input should be the output of OscopeSine() function. scanK() function will cluster genes in gene pairs with high similarity score (the threshold can be defined using parameter quan). To select the top genes, the function first calculate the max similarity score for each gene, then select the genes with high max score.

The output object shows members in each cluster. clusters are sorted by median similarity score within cluster.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
tmp <- matrix(sin(rnorm(330)),ncol=11)
rownames(tmp) <- paste0("tmp",1:30)
Dat <- rbind(aa, bb, cc, tmp)
res1 <- OscopeSine(Dat)
res2 <- OscopeKM(res1, quan=.8, maxK=5)
```

---

OscopeSine

*Apply sine model on the full set of genes or isoforms*

---

**Description**

Apply sine model on the full set of genes or isoforms

**Usage**

```
OscopeSine(DataInSc, parallel=FALSE, parallelParam=NULL)
```

**Arguments**

**DataInSc** a gene-by-sample (isoform-by-sample) matrix indicating the rescaled expression of two genes/isoforms. all values should be between [-1, 1].

**parallel** whether apply parallel computing. if it is TRUE, BiocParallel will be called.

**parallelParam** a SnowParam object to specify the clusters. If it is NULL, the default will be set as SnowParam(workers = 5, type = "SOCK") remains unchanged for over NCThre iterations.

**Value**

Output is a list with 4 sublists, each shows a N-by-N matrix, in which N is the total number of genes (isoforms). **SimiMat**: similarity matrix (sine scores); the sine scores are calculated by  $-\log_{10}(\epsilon^2)$ . **DiffMat**: dissimilarity matrix; shown are  $\epsilon^2$  for each gene pair. **ShiftMat**: optimal phase shift estimate for each pair of genes.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
OscopeSine(rbind(aa,bb,cc))
```

---

PermuCut

*Define sine score cutoff using permuted data*

---

**Description**

Define sine score cutoff using permuted data

**Usage**

```
PermuCut(Data, NumPermu=1000)
```

**Arguments**

**Data** a gene-by-sample (isoform-by-sample) matrix indicating the rescaled expression of genes/isoforms. all values should be between [-1, 1].

**NumPermu** number of permuted genes to generate.

**Value**

Output contains a vector of numbers. Each number presents max sine score of a given permuted gene.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
tmp <- matrix(sin(rnorm(330)),ncol=11)
rownames(tmp) <- paste0("tmp",1:30)
Dat <- rbind(aa, bb, cc, tmp)
res1 <- PermuCut(Dat,100)
```

---

**PipeR***Calculate residual of polynomial fit*

---

**Description**

Calculate residual of polynomial fit

**Usage**

```
PipeR(Data,Ndg=3,Method="Poly")
```

**Arguments**

Data	gene-by-sample matrix or isoform-by-sample matrix.It should be rescaled to values bwteen [-1,1].
Ndg	degree of polynomial.
Method	only polynomial fitting ("Poly") is available now.

**Value**

The function will fit polynomial curve to each row of the data. The output returns the MSE of each row (gene/isoform).

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
res <- PipeR(rbind(aa,bb,cc))
```

---

PipeShiftCDF	<i>Calculate residual of the sliding polynomial regression</i>
--------------	--

---

**Description**

Calculate residual of the sliding polynomial regression

**Usage**

```
PipeShiftCDF(Data, Ndg=3, NChun=4, RdmStart=FALSE)
```

**Arguments**

Data	gene-by-sample matrix or isoform-by-sample matrix. It should be rescaled to values between [-1,1].
Ndg	degree of polynomial.
NChun	number of starting points for polynomial fitting.
RdmStart	whether the start points are randomly selected.

**Value**

The function will fit sliding polynomial regression (SPR) to each row of the data. For each gene/isoform, SPR fits NChun polynomial curves with different starting points (samples). The samples with smaller order than the start point will be appended to follow the last sample when fitting. So each fitting consider same number of samples. If RdmStart = TRUE, the start points are randomly selected. Otherwise they are evenly sampled along the sample order. The aggregated MSE of a fit (using a specific start point) is defined as the summation of the MSEs of all genes/isoforms considered here. The output returns the MSE of the SPR, which is the largest aggregated MSE across fits using different start points.

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
res <- PipeShiftCDF(rbind(aa,bb,cc), NChun=2)
```

---

 scanK

*Run k-medoid algorithm with varying k on similarity matrix*


---

**Description**

Run k-medoid algorithm with varying k on similarity matrix

**Usage**

```
scanK(SimiMatIn, quan=.95,cut=NULL, maxK=NULL,minSize=0, maxSize=200, fixK=NULL, rawscale=FALSE)
```

**Arguments**

SimiMatIn	gene-by-gene similarity matrix
quan	only gene pairs with similarity score $\geq$ quan th quantile will be considered in the cluster analyses. Default is 0.95.
cut	pre-defined cutoff. Gene pairs with similarity score $\geq$ cut will be considered in cluster analyses. If cut is defined, quan will be ignored.
maxK	max number of clusters to consider (scan). if numbC=NULL, it will be calculated as [number of gene considered]/10.
minSize,maxSize	Only clusters with minSize $\leq$ cluster size $\leq$ maxSize are reported in output.
fixK	if fixK is specified, the k-medoids algorithm will be applied with fixK clusters.
rawscale	Recall the input is the similarity matrix ( $-\log_{10}(\text{distance from the sine model})$ ). the k-medoids clustering will be applied using (-Input) as distance. If rawscale is defined as TRUE, the k-medoids clustering will be applied using $-10^{\text{Input}}$ as distance.

**Value**

scanK() function runs k-medoid clustering with varying number of clusters (k). The k is varied from 2 to maxK. The input of scanK() function should be a similarity matrix. scanK() function will cluster genes in gene pairs with high similarity score (the threshold can be defined using parameter quan). To select the top genes, the function first calculate the max similarity score for each gene, then select the genes with high max score.

The output object is a list with 4 sublists: membOut: members in each cluster. clusters are sorted by median similarity score within cluster;

MedCor: median similarity score for each cluster;

Mat: input similarity matrix;

filteredMat: similarity matrix, only showing the top genes used in clustering;

Kcluster: cluster indicator of each top gene.

**Author(s)**

Ning Leng



**Examples**

```

aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
tmp <- matrix(sin(rnorm(330)),ncol=11)
rownames(tmp) <- paste0("tmp",1:30)
Dat <- rbind(aa, bb, cc, tmp)
res1 <- OscopeSine(Dat)
res2 <- scanK(res1$SimiMat, quan=.8, maxK=5)

```

SineFun

*Apply sine model on one particular gene vs. other genes***Description**

Apply sine model on one particular gene vs. other genes

**Usage**

```
SineFun(DataInSc, i)
```

**Arguments**

<code>DataInSc</code>	a gene-by-sample (isoform-by-sample) matrix indicating the rescaled expression of two genes/isoforms. all values should be between [-1, 1].
<code>i</code>	the gene (isoform) of interest. The function will apply the sine model on gene (isoform) <code>i</code> vs. gene (isoform) <code>j</code> for all $j > i$ . Gene (isoform) <code>i</code> ( <code>j</code> ) is defined as the gene (isoform) shown in the <code>i</code> ( <code>j</code> ) th row. <code>i</code> should be smaller than the total number of genes (isoforms).

**Value**

Output is a list with two sublists, each shows the optimal phi's (shift) and epsilon's (value).  $N-i$  entries will be included in each sublist ( $N$  is the total number of genes/isoforms). The  $k$ th entry indicates results of gene (isoform) `i` vs. `i+k`.

**Author(s)**

Ning Leng

**Examples**

```

aa <- sin(seq(0,1,.1))
bb <- sin(seq(0.5,1.5,.1))
cc <- sin(seq(0.9,1.9,.1))
SineFun(rbind(aa,bb,cc), 1)

```

---

SineOptim

*Function for searching optimal phase shift*

---

**Description**

Function for searching optimal phase shift

**Usage**

```
SineOptim(Pairdata)
```

**Arguments**

Pairdata            a sample-by-2 matrix indicating the rescaled expression of two genes/isoforms.  
                     all values should be between [-1, 1].

**Value**

Output provides the optimal phi (shift) and its corresponding epsilon<sup>2</sup> (value) of the sine model.  
 $\text{epsilon\_g1,g2}^2 = \sum_s [X_{g1,s}^2 + X_{g2,s}^2 - 2X_{g1,s}X_{g2,s} \cos(\text{phi\_g1,g2}) - \sin^2(\text{phi\_g1,g2})]^2$

**Author(s)**

Ning Leng

**Examples**

```
aa <- sin(seq(0,1,.1))  
bb <- sin(seq(0.5,1.5,.1))  
SineOptim(cbind(aa,bb))
```

# Index

## \*Topic **datasets**

OscopeExampleData, [11](#)

## \*Topic **package**

Oscope-package, [2](#)

AbsCor, [3](#)

CalcMV, [3](#)

FlagCluster, [4](#)

FormatSineOut, [5](#)

ImpShift, [6](#)

NISFun, [7](#)

NormForSine, [8](#)

Opt2Shift, [9](#)

Oscope (Oscope-package), [2](#)

Oscope-package, [2](#)

OscopeENI, [10](#)

OscopeExampleData, [11](#)

OscopeKM, [11](#)

OscopeSine, [12](#)

PermuCut, [13](#)

PipeR, [14](#)

PipeShiftCDF, [15](#)

scanK, [16](#)

SineFun, [17](#)

SineOptim, [18](#)