Package 'scsR'

April 10, 2015

Type Package

Title SiRNA correction for seed mediated off-target effect

Version 1.3.2

Date 2014-10-28

Author Andrea Franceschini

Maintainer Andrea Franceschini <andrea.franceschini@isb-sib.ch>, Roger Meier <roger.meier@lmsc.ethz.ch>, Christian von Mering <mering@imls.uzh.ch>

Description Corrects genome-wide siRNA screens for seed mediated off-target effect. Suitable functions to identify the effective seeds/miRNAs and to visualize their effect are also provided in the package.

License GPL-2

Depends R (>= 2.14.0), STRINGdb, methods, BiocGenerics, Biostrings, IRanges, plyr, tcltk

Imports sqldf, hash, ggplot2, graphics,grDevices, RColorBrewer

Suggests RUnit

biocViews Preprocessing

R topics documented:

`add_rank_col` *add_rank_col*

Description

This method takes in input a dataframe containing the results of an siRNA screen. Then it adds a set of column that are useful for sorting to the dataframe. At the moment the following sorting columns are provided: - column with the median value of the siRNA score for each gene - columns that comes out from the execution of the RSA sorting method (Renate Konig et al.)

Usage

```
add_rank_col(screen, reverse=FALSE,  scoreColName="score", geneColName="GeneID")
```

Arguments

`screen`	data frame containing the results of the siRNA experiment.
`reverse`	boolean specifying the direction of the sorting (from the lowest scores to the highest score or vice versa)
`scoreColName`	character vector with the name of the column that contains the score of the screen
`geneColName`	character vector withname of the column that contains the names of the genes in the screen

Value

screen data frame with sorting columns added.

Author(s)

Andrea Franceschini

References

A probability-based approach for the analysis of large-scale RNAi screens. Renate Konig et al. Nature Methods 2007

Examples

```
data(uuk_screen)
uuk_screen_ranked = add_rank_col(uuk_screen[1:100,])
```

`add_seed` *add_seed*

Description

This method takes in input a dataframe containing the results of an siRNA screen. This screen must contain the siRNA sequences in a dedicated column (the sequences have to be provided in the guide/antisense orientation). Then it adds a column with the seed of the siRNA sequences.

Usage

```
add_seed(df, seqColName="siRNA_seq", seedLength=7, startPosition=2)
```

Arguments

`df`	Dataframe containing the results of the siRNA screen.
`seqColName`	character vector with the name of the column that contains the siRNA sequences (the sequences have to be provided in the guide/antisense orientation).
`seedLength`	length of the seed in nucleotides (by default 7 bases) (integer)
`startPosition`	position in the siRNA sequence where the seed starts (by default position 2) (integer))

Value

screen data frame with the seed column added.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
seed_uuk_screen = add_seed(uuk_screen[1:100,])
```

---

benchmark_shared_hits *benchmark_shared_hits*

Description

This method can be used to benchmark sorted gene vectors (A) that comes out from a siRNA screen. The benchmark is done against other sorted gene vectors (B) that we know to contain high density of real hits (e.g. the results of a second siRNA screen performed with a different library). The benchmark is performed simply comparing the top n hits of the two lists. If the two lists contain many shared best hits than we have a strong statistical signal. Then we display the number of shared best hits for different n, in a graph (if visualize_pval variable is set to true the pvalue of the t-test is plotted instead of the number of shared hits).

Usage

```
benchmark_shared_hits(glA, glB, col, avoidIntersectL=FALSE,
                          output_file=NULL, npoints=400, title="", scaleAXPoint = 1,
                          scaleBXPoint = NULL, fixedBXPoint=400, displayRandomMultipleLines=TRUE,
                          nrandom=20, intersectGenes=TRUE, visualize_pval=FALSE, max_ylim=NULL, xlab=N
```

Arguments

glA	sorted list containing one or more sorted vectors of genes (i.e. hits of a genome wide screen sorted by significance). Each element i of glA will be benchmarked against element i of glB. In case glB contains only one element, each glA vector will be benchmarked against glB[1].
glB	sorted list containing one or more sorted vectors of genes (i.e. hits of a genome wide screen sorted by significance).
col	sorted vector of booleans (a boolean i in the vector corresponds to the shared hits of glA[i] with glB[i])
avoidIntersectL	sorted vector of colors (a color i in the vector corresponds to the shared hits line obtain intersecting glA[i] with glB[i]) To perform the benchmark we construct a background to be used (this background is given by the intersection of all the glA and glB vectors) When an element i of the vector is set to TRUE, we don't use the elements of glA[i] to compute the vector. This allows to benchmark also methods that do output only few putative good genes (instead of a sorted list of all the genes tested).
npoints	number of points on the x-axis of the graph (integer)
nrandom	number of random lines to compute (in order to infer the variation of the noise) (integer)

output_file	path to the output file where to store the graph (character vector)
title	title of the graph (character vector)
scaleAXPoint	for position x in the graph we compare the best x * scaleAXPoint best hits of the genesA vector (integer)
scaleBXPoint	for position x in the graph we compare the best x * scaleBXPoint best hits of the genesB vector (integer)
fixedBXPoint	for position x in the graph we compare the best fixedBXPoint best hits of the genesB vector (integer)
intersectGenes	specify whether to intersect the genes from the various input vectors to form a suitable background to be used for the benchmark. (boolean)
visualize_pval	specify whether a p-value (derived by an hypergeometric test) should be visualized instead of the number of shared hits. (boolean)
displayRandomMultipleLines	specify whether to display several random lines in the graph (instead of only one line that is the average of all the random lines) (boolean)
max_ylim	y upper limit (integer)
xlab	xlab (character vector)
ylab	ylab (character vector)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
data(uuk_screen_dh)

benchmark_shared_hits(
  glA=list(
    uuk_screen[1:1000,]$GeneID,
    arrange(add_rank_col(uuk_screen[1:1000,]), log_pval_rsa)$GeneID
  ),
  glB=list(uuk_screen_dh$GeneID),
  col=c("black", "blue"),
  title="UUKUNIEMI Hela Cell Killers"
)
```

---

bydf *bydf*

Description

apply a function to a group of rows in the input data frame (similar to the sql group by statememnt).

Usage

```
bydf(df, groupColName, valColName, fun, newColName="temp_by_col_name")

bydfa(df, groupColName, valColName, fun, newColName="temp_by_col_name")
```

Arguments

df	input data frame
groupColName	name of the column to be used for grouping the rows (character vector)
valColName	name of the column containing the values to be inserted in the function (character vector)
fun	function to be applied (function)
newColName	name of the colum that contains the result of the function (character vector)

Details

The methods currently depend on the type of to:

bydf apply a function to a group of rows in the input data frame (similar to the sql group by statememnt). Put the results of this function in a new data frame that is returned as output.

bydfa apply a function to a group of rows in the input data frame (similar to the sql group by statememnt). return the same data frame with an additional column with the results of the function.

Value

bydf: data frame with the function applied to the grouping bydfa: input data frame with an additional column with the results of the function applied to the grouping.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
screen=add_seed(uuk_screen[1:1000,])
screen_sd = bydf(screen, groupColName="seed7", "score", sd, "sd")
```

check_consistency *check_consistency*

Description

This method takes an siRNA screen as input and check its consistency (i.e. check that the format of the data is suitable for the usage with our scsR package). The method prints meaningful warnings for every inconsistency that can be detected

Usage

```
check_consistency(screen, scoreColName = "score", geneColName = "GeneID",
                                    seqColName="siRNA_seq")
```

Arguments

screen	Dataframe containing the results of the siRNA scree
scoreColName	name of the column that contains the score of the screen (character vector)
geneColName	name of the column that contains the gene identifier of the screen (character vector)
seqColName	name of the column that contains the siRNA sequences in the screen. (the sequences have to be provided in the guide/antisense orientation and each sequence must be in the format of a character vector, i.e. a simple string). (character vector)

Value

return the data frame passed in input with possible consistency corrections.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
uuk_screen <- check_consistency(uuk_screen)
```

compare_sorted_geneSets *compare_sorted_geneSets*

Description

This method can be used to compare the performances of two different sorted gene vectors (A1 and A2) relative to a reference vector (B). To perform the comparison we use n best hits from genesetA1 and genesetA2. n is defined as the number of elements of the smallest of the two vectors(after intersecting it with the background). For the comparison see also the enrichment_geneSet method.

Usage

```
compare_sorted_geneSets(genesetA1, genesetA2, genesetB, background, limA=NULL, limB=NULL)
```

Arguments

genesetA1	vector of sorted genes (character vector)
genesetA2	vector of sorted genes (character vector)
genesetB	vector of genes to be used as reference (character vector)
background	vector of genes to be used as background (character vector)
limA	limit the number of genes of the vector genesetA1 to the first limA genes (integer)
limB	limit the number of genes of the vector genesetB1 to the first limB genes (integer)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
data(uuk_screen_dh)
compare_sorted_geneSets(unique(uuk_screen$GeneID)[1:200],
                  unique(arrange(add_rank_col(uuk_screen), log_pval_rsa)$GeneID)[1:200],
                             unique(uuk_screen_dh$GeneID)[1:400],
                             intersect(uuk_screen$GeneID, uuk_screen_dh$GeneID)
                             )
```

create_sd_matrix *create_sd_matrix*

Description

We observed that the standard deviation of the oligos that share the same seed do change relative to their average score. In principle we could plot this information on a graph (x-axis = average of the oligos that share the same seed, y-axes = standard deviation of the oligos). We do provide this utility method to condense this information in a matrix (that reports the quantiles of the standard deviation for every score interval).

Usage

```
create_sd_matrix(screen, seedColName="seed7", scoreColName="score")
```

Arguments

screen	data frame containing the results of the siRNA experiment.
seedColName	name of the column that contains the seeds sequences of the screen (character vector)
scoreColName	name of the column that contains the score of the screen (character vector)

Value

matrix that reports the quantiles of the standard deviation for every score interval.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)


# to speed up the example we use only the first 100 rows
uuk_screen_reduced = uuk_screen[1:100,]

screen = add_seed(uuk_screen_reduced)
sd_matrix = create_sd_matrix(screen)
```

delColDf *delColDf*

Description

Delete a specific column in the data frame.

Usage

```
delColDf(df, colName)
```

Arguments

df	data frame
colName	name of the column to be deleted (character vector)

Value

input data frame with the column deleted.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
uuk_screen2 = delColDf(uuk_screen, "score")
```

delete_undefined_rows *delete_undefined_rows*

Description

method to delete the rows that contain undefined values in some specific columns.

Usage

```
delete_undefined_rows(df, colNames, quiet=FALSE)
```

Arguments

df	data frame
colNames	vector with the names of the column that must be defined (i.e. their values cannot be NULL, NA, NaN or zero-length strings) (vector of strings)
quiet	specify whether to avoid printing warnings. (boolean)

Value

data frame without the rows that contain at least one undefined value in the column list

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
screen <- delete_undefined_rows(uuk_screen, colNames=c("score", "GeneID"))
```

`enrichment_geneSet` *enrichment_geneSet*

Description

Computes the hypergeometric p-value that represents the enrichment of genesetA with genes of the genesetB.

Usage

```
enrichment_geneSet(genesetA, genesetB, background=NULL, quiet=FALSE)
```

Arguments

`genesetA`	vector of sorted genes (vector of strings)
`genesetB`	vector of sorted genes (vector of strings)
`background`	vector of genes to be used as background (vector of strings)
`quiet`	avoid print any message/warning (boolean)

Value

the hypergeometric p-value that represents the enrichment of genesetA with genes of the genesetB. (integer)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
data(uuk_screen_dh)
enrichment_geneSet(unique(uuk_screen$GeneID)[1:200],
                   unique(uuk_screen_dh$GeneID)[1:400],
                   intersect(uuk_screen$GeneID, uuk_screen_dh$GeneID))
```

enrichment_heatmap *enrichment_heatmap*

Description

Produces an heatmap showing the enriched annotations that are found in the input vectors of gene identifiers.

Usage

```
enrichment_heatmap(genesVectors, vectorsNames, output_file=NULL, title="", limit=400, species_ncbi_ta
                   enrichmentType="Process", limitMultiPicture=NULL, fdr_threshold=0.05, pvalue_thr
                   cexRow=NULL, cexCol=1, STRINGversion="9_05", selectTermsVector=NULL, iea = TRUE, s
```

Arguments

genesVectors	list containing several sorted vectors of genes (i.e. columns of the heatmap) for which to compute the enrichment in pathways (list)
vectorsNames	names of the vectors (to be displayed as column labels on the heatmap) (vector of strings)
output_file	path to an output file where to store the heatmap (this file should have the pdf extension) (character vector)
title	title of the heatmap graph (character vector)
limit	considers only the top genes in the vector (integer)
species_ncbi_taxonomy_id	ncbi taxonomy id of the organism (e.g. 9606 for Human) (integer)
enrichmentType	type of Enrichment of the heatmap (either Process or KEGG. The first tests for enrichment in GO biological processes, while the second tests for the enrichment in KEGG pathways) (character vector)
limitMultiPicture	number of rows of the heatmap before to start a new page in the pdf (integer)
fdr_threshold	considers only the rows with at least one element below this threshold (number)
pvalue_threshold	considers only the rows with at least one element below this threshold (number)
cexRow	size of the row labels (number)
cexCol	size of the columns' labels (number)
STRINGversion	specify the version of STRING to use for the enrichment annotations (by default 9_05) (character vector)
selectTermsVector	specify the terms to select. Each term must fully contain at least one string of this vector. This parameter can be used when we want to limit the output of the method, for example to fit the output image in one page of an article (vector of strings).

iea	specify whether to use Electronic Inferred Association annotations (to be used in case you are querying the GeneOntology). (boolean)
sortingMethod	specify whether a sorting method should be applied. For the moment, the only available method is rowMeans. (character vector)
avoidIntersect	specify whether a sorting method should be applied. For the moment, the only available method is rowMeans. (character vector)

Value

matrix that is used to generate the heat map

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
data(uuk_screen_dh)
 ## Not run:
heatmapMatrix = enrichment_heatmap( list( uuk_screen$GeneID,
                          arrange(add_rank_col(uuk_screen), log_pval_rsa)$GeneID,
                          uuk_screen_dh$GeneID
),
                    list("Qiagen", "Qiagen (RSA)", "Dharmacon"),
                    limit=400,
                    enrichmentType = "Process",
                    output_file=NULL,
                    title="Uuk Cell Killers",
selectTermsVector=c("cycle")
)


## End(Not run)
```

---

get_sd_quant *get_sd_quant*

Description

This method scan the quantile standard deviation matrix (produced by create-sd-matrix function) and finds the quantile of the given standard deviation and average score

Usage

```
get_sd_quant(sdval, score, sd_matrix)
```

Arguments

sdval	standard deviation (number)
score	average score (number)
sd_matrix	standard deviation quantile matrix (matrix)

Value

number from 1 to 20 that represents the quantile of the standard deviation in the given score range (1 corresponds to 0.05 percent). (integer)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 2500 rows
uuk_screen_reduced = uuk_screen[1:2500,]

screen = add_seed(uuk_screen_reduced)
sd_matrix = create_sd_matrix(screen)
quant <- get_sd_quant(0.3, 0.9, sd_matrix)
```

---

get_seed_oligos_df *get_seed_oligos_df*

Description

This function returns the screen, that is given in input, with additional columns about the possible off-targets/seed effect of each oligos. The seed effect is computed excluding the current oligo.

Usage

```
get_seed_oligos_df(screen, seedColName="seed7", scoreColName="score", geneColName="GeneID", gene_int
                     min_oligos_x_gene=4, min_oligos_x_statistics=4, random=FALSE, kolmogorovSample
```

Arguments

screen	data frame containing the results of the siRNA experiment (sorted by significance).
seedColName	specify the direction of the sorting (from the lowest scores to the highest score or vice versa) (character vector)
scoreColName	name of the column that contains the score of the screen (character vector)
geneColName	name of the column that contains the names of the genes in the screen (character vector)

gene_interval — apply the analysis only to the genes that are included in this interval (the screen must be sorted by significance and the interval has to be intended from the best hits to the worst hits). (vector of integer)

min_oligos_x_gene — minimum number of oligos that a gene must have in order to be included in the analysis (integer)

min_oligos_x_statistics — minimum number of oligos with the same seed that is required in order to apply a statistics (otherwise 0 is returned). (integer)

random — randomize the genes of the screen (boolean)

progress_bar — print progress bar (boolean)

kolmogorovSampleSize — sample size to be used for the Kolmogorov Smirnov statistics (i.e. the number of genes that we consider to be enough in order to infer the correct distribution of the genome-wide screen. The higher this number, the slower the computation). If this variable is left to NULL the Kolmogorov statistics is disabled (integer)

Value

screen, that is given in input, with additional columns about the possible off-targets/seed effect of each oligos. (data frame)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 100 rows
uuk_screen_reduced = uuk_screen[1:1000,]

uuk_screen <- add_seed(uuk_screen_reduced)
sodf = get_seed_oligos_df(uuk_screen)
```

---

intersectAll *intersectAll*

Description

intersect several vectors that can be passed as arguments of the functino

Usage

```
intersectAll(...)
```

Arguments

...	vectors to intersect

Value

vector that results from the intersection of the input vectors

Author(s)

Andrea Franceschini

Examples

```
intersectAll(c(1,2,3,4), c(1,2), c(2,3,4))
```

---

launch_RSA *launch_RSA*

Description

launch RSA sorting method

Usage

```
launch_RSA(df, LB=-100, UB=100, reverse=FALSE, strScoreCol="", strGeneCol="Gene_ID", keepAllRSAReturn
```

Arguments

df	data frame containing the results of the siRNA experiment.
LB	RSA lower bound (look KONIG paper). (number)
UB	RSA upper bound (look KONIG paper). (number)
reverse	whether to sort in ascending or descending order. (boolean)
strScoreCol	name of the column that contains the score of the screen (character vector)
strGeneCol	name of the column that contains the names of the genes in the screen (character vector)
keepAllRSAReturnFields	specify whether you want to keep all RSA columns in the output file. (boolean)

Value

screen data frame with RSA sorting columns added.

Author(s)

Andrea Franceschini

References

A probability-based approach for the analysis of large-scale RNAi screens. Renate Konig et al. Nature Methods 2007

Examples

```
data(uuk_screen)

#extract the first 1000 lines in order to speed up the example
screen = uuk_screen[1:1000,]

screen_ranked <- launch_RSA(screen, strGeneCol="GeneID", strScoreCol="score")
```

`median_replicates` *median_replicates*

Description

perform the median of the replicates (i.e. group by oligo sequence and takes the median of the score value).

Usage

```
median_replicates(screen, seedColName = "seed7", scoreColName = "score",
                        geneColName = "GeneID", seqColName="siRNA_seq", spAvgColName = NULL)
```

Arguments

`screen`	data frame containing the results of the siRNA experiment.
`seedColName`	name of the column that contains the seed of the sequence (character vector)
`scoreColName`	name of the column that contains the score of the screen (character vector)
`geneColName`	name of the column that contains the names of the genes in the screen (character vector)
`seqColName`	name of the column that contains the names of the sequences in the screen
`spAvgColName`	name of the column that contains the names of the genes in the screen (character vector)

Value

input data frame after having performed the median of the replicates

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
mr <- median_replicates(uuk_screen)
```

`miRBase_20` *miRBase version 20 mature sequences (homo sapiens)*

Description

Mature miRNA sequences of Homo Sapiens from miRBase version .

Usage

```
data(miRBase_20)
```

Source

The microRNA Registry. Griffiths-Jones S. NAR 2004 32(Database Issue):D109-D111

`OPIrsa` *OPIrsa*

Description

look Konig paper/code for explanation about this method

Usage

```
OPIrsa(Groups,Scores,opts,Data=NULL)
```

Arguments

`Groups`	look Konig paper/code for explanation about this parameter
`Scores`	look Konig paper/code for explanation about this parameter
`opts`	look Konig paper/code for explanation about this parameter
`Data`	look Konig paper/code for explanation about this parameter

Value

look Konig paper/code for explanation about this return value

Author(s)

Andrea Franceschini

References

A probability-based approach for the analysis of large-scale RNAi screens. Renate Konig et al. Nature Methods 2007

`OPIrsaScore` *OPIrsaScore*

Description

look Konig paper in order to have information about this function

Usage

```
OPIrsaScore(I_rank, N, i_min=1, i_max=-1)
```

Arguments

`I_rank`	look Konig paper in order to have information about this parameter
`N`	look Konig paper in order to have information about this parameter
`i_min`	look Konig paper in order to have information about this parameter
`i_max`	look Konig paper in order to have information about this parameter

Value

look Konig paper in order to have information about this return value

Author(s)

Andrea Franceschini

References

A probability-based approach for the analysis of large-scale RNAi screens. Renate Konig et al. Nature Methods 2007

plot_screen_hits *plot_screen_hits*

Description

Gene-seed plot: plot the genes of the siRNA screen (x-axis) together with a representation of the effect of the seed of their oligos (circles). The position on the y-axis of the circles refers to the average score of the oligos of the gene that share the same seed. The dimension of the circles refers to the number of oligos that share the same seed in the screen (the higher the number of oligos with the same seed, the bigger is the circle). This graph can be used to look by eyes at the effect of the seeds on the genes.

Usage

```
plot_screen_hits(screen, output_file=NULL, geneScoreColName="median", seedColName="seed7",
                scoreColName="score", geneColName="GeneID", gene_interval = c(1,100),
                min_oligos_x_gene=4, min_oligos_x_statistics=4, random=FALSE, kolmogorovSampleSize=500
                ylab="score", xlab="gene", ylim=c(-4,4), graph_highest_count_thr=16, progress_bar=FALS
```

Arguments

screen	data frame containing the results of the siRNA experiment.
output_file	specify the direction of the sorting (from the lowest scores to the highest score or vice versa) (character vector)
scoreColName	name of the column that contains the score of the screen (character vector)
geneColName	name of the column that contains the names of the genes in the screen (character vector)
seedColName	name of the column that contains the seeds of the siRNA sequences in the screen (character vector). (the sequences have to be provided in the guide/antisense orientation and each sequence must be in the format of a character vector, i.e. a simple string)
geneScoreColName	name of the column that contains the global scores of the genes (i.e. the column that contains the median or the average value of the oligos) (character vector)
gene_interval	display in the graph only the genes that are included in this interval (the screen must be sorted by significance and the interval has to be intended from the best hits to the worst hits). (vector)
min_oligos_x_gene	minimum number of oligos that a gene must have in order to be included in the analysis (integer)
min_oligos_x_statistics	minimum number of oligos with the same seed that is required in order to apply a statistics (otherwise 0 is returned). (integer)
random	randomize the genes of the screen (boolean)

kolmogorovSampleSize
: sample size to be used for the Kolmogorov Smirnov statistics (i.e. the number of genes that we consider to be enough in order to infer the correct distribution of the genome-wide screen. The higher this number, the slower the computation). If this variable is left to NULL the Kolmogorov statistics is disabled (integer)

ylab
: label of the graph y-axis (character vector)

xlab
: label of the graph x-axis (character vector)

ylim
: ylim of the graph (vector)

graph_highest_count_thr
: maximum number of oligos to be used in order to display the largest circle in the graph (number)

progress_bar
: whether to show a progress bar or not (Boolean)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 100 rows
uuk_screen_reduced = uuk_screen[1:1000,]


screen = add_rank_col(add_seed(uuk_screen_reduced))
plot_screen_hits(screen)

# The screen has to be sorted. In our case it is already sorted via median.
# In order to sort the screen you can use our add_rank_col method
# example: arrange(add_rank_col(screen), median)
```

---

plot_seeds_methods *plot seeds utility methods*

Description

Plots informations about the effect of the seed on the screen

Usage

```
plot_effective_seeds_head(screen, seedColName="seed7", scoreColName="score", enhancer_analysis=FALSE

plot_seeds_oligo_count(screen, seedColName="seed7", scoreColName="score", output_file=NULL)
```

Arguments

screen	data frame containing the results of the siRNA experiment.
seedColName	name of the column that contains the seed of the siRNA oligo sequences of the screen (character vector) (the sequences have to be provided in the guide/antisense orientation and each sequence must be in the format of a character vector, i.e. a simple string)
scoreColName	name of the column that contains the score of the screen (character vector)
enhancer_analysis	if set to true plot the seeds that cause the oligos to have an higher score instead of a lower score. (boolean)
min_oligos_x_seed	minimum number of oligos that seed must have in order to be considered (integer)
number_of_seeds	maximum number of seeds to represent in the graph (by default the top 20 seeds are shown) (integer)
output_file	name of the pdf file where to store the graph (character vector)
color	color of the bars that represent the seeds (character vector)
colorBG	color of the bars that represent the noise (i.e. analysis executed on randomized data) (character vector)
xlim	xlim of the graph (number)
title	title of the graph (number)

Details

The methods currently depend on the type of `to`:

plot_effective_seeds_head barplot that represents the most effective seeds as bar (the length of the bars corresponds to the average score of the oligos that contain that seed). A background bar is shown under every seed. We obtain these bar simply randomizing the score column of the screen (and they well represent the noise level).

plot_seeds_oligo_count For each seed that is found in the siRNA screen, plots the number of oligos that contain that seed.

plot_seed_score_sd For each seed plot its average score and its standard deviation.

plot_screen_seeds_count For each siRNA oligo, plot the number of the other oligos in the screen that share the same seed.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 2500 rows
uuk_screen_reduced = uuk_screen[1:5000,]

plot_effective_seeds_head(add_seed(uuk_screen_reduced))
```

`randomizeInner` *randomizeInner*

Description

randomize an inner field (e.g. the scores of the oligos of a gene), keeping unaltered the order of the outer field (e.g. the genes)

Usage

```
randomizeInner(df, baseColStr, sortColStr, reverse = FALSE)
```

Arguments

`df`	input data frame
`baseColStr`	name of the column that represents the outer field (e.g. the genes) (character vector)
`sortColStr`	name of the column that represents the inner field (e.g. the scores of the oligos of a gene) (character vector)
`reverse`	specify the direction of the sorting (boolean)

Value

data frame with the randomized rows.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 1000 rows
uuk_screen_reduced = uuk_screen[1:1000,]

screen <- randomizeInner(arrange(uuk_screen_reduced, GeneID), "GeneID", "score")
```

randomSortOnVal *randomSortOnVal*

Description

randomize the order of the rows, on the values of a column (e.g. randomized the rows, keeping close the rows having the same GeneID... i.e. sort the Genes of the screen in a random way).

Usage

```
randomSortOnVal(screen, strColVal)
```

Arguments

screen	data frame containing the results of the siRNA experiment.
strColVal	column of which the values have to be kept close to each other (character vector)

Value

screen data frame sorted randomly on the defined column.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 1000 rows
uuk_screen_reduced = uuk_screen[1:1000,]

screen = randomSortOnVal(uuk_screen_reduced, "GeneID")
```

---

removeSharedOffTargets *removeSharedOffTargets*

Description

remove from an siRNA genome wide screen (screenA) all the oligos with a seed that is contained also in a second screen (screenB) in oligos designed to target the same genes (i.e. if an oligo X that target gene K in screenA is found to have the same seed as an oligo Y in screenB that targets gene K, then oligo X is removed from screenA). In this way we can remove from a screen all the oligos that have potentially the same type of off-targets as those in another screen. We suggest to perform thie step before running a benchmark on the shared hits (because we don't want the benchmark to count shared hits that are generated by possible shared off-target effects)

Usage

```
removeSharedOffTargets(screenA, screenB, seedColName="seed7",
                                         geneColName="GeneID",
                                         seqColName="siRNA_seq",
                                         removeGenes=FALSE)
```

Arguments

screenA	screen to be filtered of the oligos that share the seed with oligos that target the same gene in a screenB
screenB	screenB
seedColName	name of the column that contains the seeds (character vector)
geneColName	name of the column that contains the gene identifiers (character vector)
seqColName	name of the column that contains the oligo sequences (character vector)
removeGenes	specify whether to remove just the oligos or the entire gene, as you would probably like to do when screenA is a pooled library (i.e. remove all the oligos that refer to a gene, even if only one oligos contains a seed that is common to oligos of screenB that refer to the same gene) (boolean)

Value

return screenA filtered of the oligos that are similar in seed to those of screenB. (data frame)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
data(uuk_screen_dh)

# reduce the size of the input datasets in order to make the example faster
# (you should not perform this operation in a real case)
uuk_screen=head(uuk_screen, n=2500)
uuk_screen_dh=head(uuk_screen_dh, n=2500)

uuk_qi = removeSharedOffTargets(add_seed(uuk_screen), add_seed(uuk_screen_dh))
```

---

renameColDf *renameColDf*

Description

rename the column of a data frame

Usage

```
renameColDf(df, colOldName, colNewName)
```

Arguments

df	input data frame
colOldName	name of the column that has to be changed (character vector)
colNewName	new column name (character vector)

Value

input data frame with the name of the column changed

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
screen <- renameColDf(uuk_screen, "score", "my_z_score")
```

---

replace_non_null_elements

replace_non_null_elements

Description

replace the element of the input vector with the element of the replacementVector (whenever these elements are not empy/null)

Usage

```
replace_non_null_elements(inputVect, replacementVect)
```

Arguments

inputVect	data frame containing the results of the siRNA experiment.
replacementVect	replacement vector (vector)

Value

input vector with the replaced values.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 1000 rows
uuk_screen_reduced = uuk_screen[1:1000,]

# replace all the scores with 1, except the first 100 scores of the vector
nv <- replace_non_null_elements(uuk_screen_reduced$score, c(rep(NA, 100), rep(1, nrow(uuk_screen_reduced)-100) ))
```

`seeds_analysis` *seeds_analysis*

Description

Create a data frame with several information on the effect of each seed in the genome-wide siRNA screen. The average score of that seed is reported, together with the number of oligos that contain that seed. Besides, suitable statistics are performed in order to estimate the p-value that the seed has an effect on the phenotype: - Hypergeometric test (i.e. the probability that the seeds has more hits than expected by chance) - Kolmogorov Smirnov test (i.e. the probability that you can obtain such high oligo scores by chance sampling from the entire score vector in the screen). In addition we also report the human miRNAs that have the same seeds as the oligos (given that you could be interested to test them in the lab).

Usage

```
seeds_analysis(screen, seedColName="seed7", scoreColName="score", hit_th_val=NULL,
                                enhancer_analysis=FALSE, spAvgColName=NULL,
                                minCount=NULL, ks_enabled=FALSE, miRBase=NULL)
```

Arguments

`screen`	data frame containing the results of the siRNA experiment.
`seedColName`	name of the column that contains the seeds of the siRNA oligo sequences. (character vector) (the sequences have to be provided in the guide/antisense orientation and each sequence must be in the format of a character vector, i.e. a simple string)
`scoreColName`	name of the column that contains the score of the screen (character vector)
`hit_th_val`	if the score of an oligo is below this threshold we define it as an hit. This is then used to compute a p-value with an hypergeometric test for the seed. If this value is left to NULL, the best 10 percent of the oligos are considered as hits. (number)
`enhancer_analysis`	specify the direction of the analysis. When this variable is set to FALSE it means that we are looking at the seeds that decrease the score of the oligos (when it is set to TRUE, it means we are looking at the seeds that increase the score of the oligos). (booleam)

spAvgColName	it is possible to specify the name of one column on which we want to perform the average, when we group for the seeds (for example, other than looking at the phenotype we may want to know the effect on the seed also on the cell number and display it on the same table). (vector of strings)
minCount	minimum number of oligos in which a seed must be present in order to be reported in the output table (integer)
ks_enabled	specify whether you want to compute also a Kolmogorov Smirnov test on the score of the seed. (boolean)
miRBase	data frame object containing the human miRNAs and their sequences. The names of the columns must be "miRNA" and "seq" (data frame)

Value

return a data frame with several information on the effect of each seed in the genome-wide siRNA screen.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)
data(miRBase_20)

# to speed up the example we use only the first 1000 rows
uuk_screen_reduced = uuk_screen[1:1000,]

seeds = seeds_analysis( add_seed(uuk_screen_reduced), miRBase = miRBase_20)
```

---

seed_correction *seed_correction*

Description

This method assumes that the seed effect acts in an "additive" way to the on-target signal. For example if we have an oligo X that has score -2, the method computes the average score of the other oligos in the libraries that contain the same seed as the oligo X. If for example this average score turns out to be -1.5, we can just subtract this score to the original oligo score to obtain the new "corrected" score ((-2) - (-1.5) = -0.5). However, the method assumes also that the correction factor (-1.5 in the previous example) should be multiplied by a coefficient c that reflects "how much" we suppose the effect is really additive ((-2) - c*(-1.5)). The coefficient c can be a constant (e.g. 0.5) or it can vary depending on the behavior of the oligos that share the seed. In particular, we observed the the last approach to be the most successful. Hence for our algorithm we set c = 0.4 + s. s is a factor proportional to the distance of the standard deviation of the oligos that share the same seed with respect to the standard deviation that is expected, given their average score. This is because we observed that the expected standard deviation of the oligos that share a seed strictly

depends on the average score as it can be seen using our plot-seed-score-sd function. In particular s = sd_correction_coeff * quantile_std (sd_correction_coeff is a constant, by default set to 0.6, and quantile_std is the quantile of the standard deviation of the seeds that have an average score in the same interval as that of the oligos having the seed of the oligo X).

Usage

```
seed_correction(screen, seedColName="seed7", scoreColName="score",
                                geneColName="GeneID", fixed_correction_coeff=0.4,
                        sd_correction_coeff=0.6, min_siRNAs_x_seed=3, progress_bar=FALSE)
```

Arguments

screen	data frame containing the results of the siRNA experiment.
seedColName	name of the column that contains the seed of the siRNA oligo sequences. (character vector) (the sequences have to be provided in the guide/antisense orientation and each sequence must be in the format of a character vector, i.e. a simple string)
scoreColName	name of the column that contains the scores of the screen. (character vector)
geneColName	name of the column that contains the identifiers of the genes in the screen (character vector)
fixed_correction_coeff	This coefficient is summed to the sd_correction_coefficient to obtain the final coefficient that is multiplyed to the correction factor (this final number must always be between 0 and 1). (character vector)
sd_correction_coeff	correction coefficient that is calibrated using the standard deviation of the oligos (i.e. oligos having a much lower standard deviation than expected are corrected multiplying the correction factor exactly by this coefficient... while oligos that have a normal or high standard deviation are corrected multiplying the correction factor by a number that is always lower than this variable, and that depends to the quantile of the standard deviation). This coefficient (after calibration) is summed to the fixed coefficient to obtain the final coefficient that is multiplyed to the correction factor (this final number must always be between 0 and 1). (number)
min_siRNAs_x_seed	This variable specify the minimum number of oligos that need to be present in the screen with the same seed as the oligo that we are correcting (the oligo that we are correcting is not included in the count). (number)
progress_bar	set this parameter to TRUE to show a progress bar. (boolean)

Value

screen data frame with the score of the oligos corrected for the seed effect.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# To reduce the execution time in the example we trim the real dataset to contain only the first 500 rows
# However in any real case the entire content of a genome-wide screen should be provided in as input.
screen=uuk_screen[1:500,]

screen_corrected = seed_correction(add_seed(screen))
```

`seed_correction_pooled`

seed_correction_pooled

Description

This method assumes that the seed effect acts in an "additive" way to the on-target signal. For example if we have an oligo X that has score -2, the method computes the average score of the other oligos in the libraries that contain the same seed as the oligo X. If for example this average score turns out to be -1.5, we can just subtract this score to the original oligo score to obtain the new "corrected" score ((-2) - (-1.5) = -0.5). However, the method assumes also that the correction factor (-1.5 in the previous example) should be multiplied by a coefficient c that reflects "how much" we suppose the effect is really additive ((-2) - c*(-1.5)). The coefficient c can be a constant (e.g. 0.5) or it can vary depending on the behavior of the oligos that share the seed. In particular, we observed the the last approach to be the most successful. Hence for our algorithm we set c = 0.4 + s. s is a factor proportional to the distance of the standard deviation of the oligos that share the same seed with respect to the standard deviation that is expected, given their average score. This is because we observed that the expected standard deviation of the oligos that share a seed strictly depends on the average score as it can be seen using our plot-seed-score-sd function. In particular s = sd_correction_coeff * quantile_std (sd_correction_coeff is a constant, by default set to 0.6, and quantile_std is the quantile of the standard deviation of the seeds that have an average score in the same interval as that of the oligos having the seed of the oligo X).

Usage

```
seed_correction_pooled(screen, seedColName="seed7", scoreColName="score",
                                geneColName="GeneID", fixed_correction_coeff=0.4,
                    sd_correction_coeff=0.6, min_siRNAs_x_seed=4, poolSize=4, enhancer_analysis=NULL
                                use_all_seeds=TRUE, progress_bar=FALSE)
```

Arguments

screen	data frame containing the results of the siRNA experiment.
seedColName	name of the column that contains the seed of the siRNA oligo sequences. (character vector) (the sequences have to be provided in the guide/antisense orientation and each sequence must be in the format of a character vector, i.e. a simple string)
scoreColName	name of the column that contains the scores of the screen. (character vector)

`geneColName` name of the column that contains the identifiers of the genes in the screen (character vector)

`fixed_correction_coeff`
This coefficient is summed to the sd_correction_coefficient to obtain the final coefficient that is multiplyed to the correction factor (this final number must always be between 0 and 1). (character vector)

`sd_correction_coeff`
correction coefficient that is calibrated using the standard deviation of the oligos (i.e. oligos having a much lower standard deviation than expected are corrected multiplying the correction factor exactly by this coefficient... while oligos that have a normal or high standard deviation are corrected multiplying the correction factor by a number that is always lower than this variable, and that depends to the quantile of the standard deviation). This coefficient (after calibration) is summed to the fixed coefficient to obtain the final coefficient that is multiplyed to the correction factor (this final number must always be between 0 and 1). (number)

`min_siRNAs_x_seed`
This variable specify the minimum number of oligos that need to be present in the screen with the same seed as the oligo that we are correcting (the oligo that we are correcting is not included in the count). (number)

`poolSize` number of siRNAs in each pool/well (integer)

`enhancer_analysis`
whether you are looking to find the genes having an high z-score (true) or a low z-score(false) (boolean)

`use_all_seeds` use all the seeds in the pool (and not only the seed having maximum score) (boolean)

`progress_bar` set this parameter to TRUE to show a progress bar. (boolean)

Value

screen data frame with the score of the oligos corrected for the seed effect.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen_dh)

# To reduce the execution time in the example we trim the real dataset to contain only the first 500 rows
# However in any real case the entire content of a genome-wide screen should be provided in as input.
screen=uuk_screen_dh[1:500,]

screen_corrected = seed_correction_pooled(add_seed(screen))
```

seed_removal *seed_removal*

Description

In certain cases we may want to select only the siRNA oligos that we are sure to NOT have strong detectable seed effect. This is different than using our seed_correction method because we don't correct the scores based on an additivity assumption, but simply remove the oligos that shows a detectable off-target effect. In principle such function should lead to the identification of few very reliable hits, but we will loose several potential hits (that can probably be detected using our seed_correction method). Hence we suggest the user to use first this seed_removal function, and then also our seed_correction method.

Usage

```
seed_removal(screen,seedColName="seed7", scoreColName="score", geneColName="GeneID",
                    min_siRNAs_x_seed=4, remove_unrepresented_seeds=TRUE, lower_bound_threshold = -0.5
                    higher_bound_threshold = 0.5, min_oligos_x_gene_threshold = 2, useMedian=FALSE, rem
```

Arguments

screen	data frame containing the results of the siRNA experiment.
seedColName	name of the column that contains the seed of the siRNA oligo sequences. (character vector) (the sequences have to be provided in the guide/antisense orientation and each sequence must be in the format of a character vector, i.e. a simple string)
scoreColName	name of the column that contains the score of the screen (character vector)
geneColName	name of the column that contains the names of the genes in the screen (character vector)
min_siRNAs_x_seed	minimum number of oligos x seed that are required in order to execute the analysis (i.e. compute their average score and remove them if the score is outside the boundaries defined in the lower_bound_threshold and higher_bound_threshold parameters) (integer)
remove_unrepresented_seeds	if set to TRUE, we remove all the seeds that are found in the screen in less than the number of oligos specified in the min_siRNAs_x_seed parameter. You should use this option if you think that it is not advisable to rely on siRNAs having seeds that are present in only few oligos, because we are not able to estimate precisely their seed effect and hence we cannot detect whether they have strong off-target effect. (boolean)
lower_bound_threshold	lower bound on the average(or median) score of the seed to be considered effective (number)

higher_bound_threshold
: higher bound on the average(or median) score of the seed to be considered effective (number)

min_oligos_x_gene_threshold
: minimum number of siRNAs that each gene must have in order to be reported. Using this function we end up removing several seeds, and hence some genes remain with few oligos (1 or 2). You could think that this low number is not enough to be able to detect an effect on the gene, and hence you may like to remove these genes setting this variable appropriately. (integer)

useMedian
: specify whether to use the mean or the median to compute the score of each seed (boolean)

removeGenes
: specify whether to remove the genes for which at least one siRNA targeting that gene has been found effective. This approach is suggested for the pooled libraries. (boolean)

include_current_gene
: lower bound on the average score of the seed to be considered effective (boolean)

progress_bar
: lower bound on the average score of the seed to be considered effective (boolean)

Value

screen data frame where we removed the siRNAs that show a detectable seed effect.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 1000 rows
uuk_screen_reduced = uuk_screen[1:1000,]

screen_corrected = seed_removal(add_seed(uuk_screen_reduced))
```

---

sortInner *sortInner*

Description

sorts an inner field (e.g. the scores of the oligos of a gene), keeping unaltered the order of the outer field (e.g. the genes)

Usage

```
sortInner(df, baseColStr, sortColStr, reverse = FALSE)
```

Arguments

df	input data frame.
baseColStr	name of the column that represents the outer field (e.g. the genes) (character vector)
sortColStr	name of the column that represents the inner field (e.g. the scores of the oligos of a gene) (character vector)
reverse	specify the direction of the sorting (boolean)

Value

data frame with sorted rows.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 1000 rows
uuk_screen_reduced = uuk_screen[1:1000,]

uuk_screen_innerSorted <- sortInner(uuk_screen_reduced, "GeneID", "score")
```

---

split_df *split_df*

Description

You can use this function to extract only certain rows from the data frame. Give a value in the strIdCol, we include in the output data frame only the rows specified in the linesToGet vector (e.g. for every gene extract only the first two oligos)

Usage

```
split_df(df, strIdCol, linesToGet)
```

Arguments

df	input data frame
strIdCol	name of the column containing the identifiers of the groups (character vector)
linesToGet	vector containing the numbers of the lines to retrieve (vector)

Value

subset of the input data frame (only the rows requested are included in the subset)

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

# to speed up the example we use only the first 1000 rows
uuk_screen_reduced = uuk_screen[1:1000,]


uuk_screen_firstOligo <- split_df(uuk_screen_reduced, "GeneID", c(1))
```

`transcribe_seqs` *transcribe_seqs*

Description

transcribe the sequences that are present in a specific column of the input data frame

Usage

```
transcribe_seqs(df, seqColName="siRNA_seq", toDNA=FALSE, progress_bar=FALSE)
```

Arguments

`df`	input data frame.
`seqColName`	name of the column that contains the sequences (character vector) (each sequence must be in the format of a character vector, i.e. a simple string)
`toDNA`	choose whether to transcribe to DNA (i.e. put T instead of U) (boolean)
`progress_bar`	choose whether disable printing warnings/messages. (boolean)

Value

data frame given in input, but with the sequences transcribed.

Author(s)

Andrea Franceschini

Examples

```
data(uuk_screen)

input_screen = head(uuk_screen, n=10)
uuk_screen_transcribed = transcribe_seqs(input_screen)
```

uuk_screen *cell number phenotype of the Uukuniemi QIAGEN siRNA genome wide screen*

Description

Cell Number phenotype of a genome wide siRNA screen that have been performed on Hela DC-SIGN cells that have been infected with the Uukuniemi virus. The screen has been performed using a QIAGEN unpooled library having 4 siRNA oligos x gene.

Usage

```
data(uuk_screen)
```

Format

Data frame with 72249 observations on the following 3 variables.

`GeneID` a numeric vector

`siRNA_seq` a character vector

`score` a numeric vector

Source

Roger Meier et al. 2014

uuk_screen_dh *cell number phenotype of the Uukuniemi DHARMACON siRNA genome wide screen*

Description

Cell Number phenotype of a genome wide siRNA screen that have been performed on Hela DC-SIGN cells that have been infected with the Uukuniemi virus. The screen has been performed using a DHARMACON pooled library with pools of 4 siRNA oligos x gene.

Usage

```
data(uuk_screen_dh)
```

Format

Data frame with 70304 observations on the following 3 variables.

`GeneID` a numeric vector

`siRNA_seq` a character vector

`score` a numeric vector

Source

Roger Meier et al. 2014

Index