Package ‘brentlabRnaSeqTools’

June 29, 2021

Title A collection of functions to interact with the brentlab RNASeq database

Version 0.0.0.9000

Description
Process genomics data from brentlab databases, and other assorted brentlab RNASeq functions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

biocViews

Imports rlang (>= 0.4.11),
scales (>= 1.1.1),
devtools (>= 1.2.1),
sva(>= 3.34.0),
jsonlite (>= 1.7.2),
httr (>= 1.4.2),
readxl (>= 1.3.1),
matrixStats (>= 0.58.0),
tidyverse (>= 1.3.0),
dplyr (>= 1.0.5),
tidyr (>= 1.1.3),
ggplot2 (>= 3.3.3),
readr (>= 1.4.0),
stringr (>= 1.4.0),
ggbio (>= 1.34.0),
GenomicRanges (>= 1.38.0),
GenomicFeatures (>= 1.38.2),
RPostgres (>= 1.3.2),
RSQLite (>= 2.2.7),
glue (>= 1.4.2),
Rsamtools (>= 2.2.3),
DESeq2 (>= 1.26.0),
magrittr (>= 2.0.1),

RColorBrewer (>= 1.1-2),
ggExtra(>= 0.9)

URL https://github.com/cmatKhan/brentlabRnaSeqTools

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown,
covr

VignetteBuilder knitr

Depends R (>= 3.6.3),
tibble (>= 2.1.3)

Config/testthat/edition 3

R topics documented:

`archiveDatabase` *pull entire database (not counts) and save to output_dir for archival purposes*

Description

saves both the individual tables, including counts, and the combined_df

Usage

```
archiveDatabase(
  database_host,
  database_name,
  database_user,
  database_password,
  output_dir,
  archive_counts_flag = TRUE
)
```

Arguments

`database_host` if connecting to a database hosted on AWS, it might be something like ec2-54-83-201-96.compute-1.amazonaws.com

`database_name` name of the database, eg for cryptococcus kn99, the database might be named kn99_database. Check with the documentation, whoever set up the database, or get into the server and check directly database_user a user of the actual database, with some level of permissions. You'll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

database_password

password to the database user. You'll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

output_dir where to deposit a subdirectory, named by todays date in this format: 20210407, with the tables and combined_df inside. eg a mounted local directory /mnt/htcf_lts/crypto_database_archiv –> /lts/mblab/Crypto/rnaseq_data/crypto_database_archive

archive_counts_flag

boolean indicating whether or not to save the counts. default is TRUE

Value

None, writes a directory called <today's date> with tables and combined_df as .csv to output_dir

calculateCoverage *calculate coverage over a given locus*

Description

calculate coverage over a given locus

Usage

```
calculateCoverage(bamfile_path, annote_db, gene_id, strandedness, ...)
```

Arguments

bamfile_path path to a bam file @seealso brentlabRnaSeqTools::createBamPath()

annote_db a GenomicFeatures TxDb object. Maybe one made from a gtf, eg txdb = makeTxDbFromGFF("data/liftoff format = "gtf")

gene_id a gene_id of interest – must be in the gene names of the annote_db object

strandedness one of c("unstranded", "reverse") indicating the strandedness of the library. note: forward not currently supported

... additional arguments to getCoverageOverRegion()

Value

percent coverage of feature with reads above a given quality threshold and coverage depth threshold (see getCoverageOverRegion())

calculateGeneWiseMedians *calculate medians across rows of dataframe*

Description

calculate medians across rows of dataframe

Usage

```
calculateGeneWiseMedians(count_df)
```

Arguments

`count_df` could be any numeric dataframe, but in this context it will typically be a count (raw or log2) df

Value

a vector of row-wise medians (length == nrow of input df)

calculateRLE *calculate RLE of a numeric dataframe*

Description

calculate RLE of a numeric dataframe

Usage

```
calculateRLE(counts_df, log2_transformed_flag = FALSE)
```

Arguments

`counts_df` gene by samples dataframe of raw counts or logged counts (see paramter logged)

`log2_transformed_flag` Default FALSE set to true if log2 transformed counts are passed

Value

rle dataframe with genes x samples. Values are the logged differences from the gene-wise medians

connectToDatabase *Connect to a remote postgresql database*

Description

Use the RPostgres package to connect to a remote postgresql database

Usage

```
connectToDatabase(
  database_host,
  database_name,
  database_user,
  database_password
)
```

Arguments

`database_host` if connecting to a database hosted on AWS, it might be something like ec2-54-83-201-96.compute-1.amazonaws.com

`database_name` name of the database, eg for cryptococcus kn99, the database might be named kn99_database. Check with the documentation, whoever set up the database, or get into the server and check directly

`database_user` a user of the actual database, with some level of permissions. You'll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

`database_password` password to the database user. You'll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

Value

A DBI connection to the remote database

Note

for information on using R environmental files, see `https://support.rstudio.com/hc/en-us/articles/360047157094-Managing-R-with-Rprofile-Renviron-Rprofile-site-Renviron-site-rsession-conf-`

Source

`https://rpostgres.r-dbi.org/`

countReadsInRanges *given a bam file path, GRanges object, and strandedness of library, return total counts*

Description

given a bam file path, GRanges object, and strandedness of library, return total counts

Usage

```
countReadsInRanges(bamfile_path, granges_of_interest, strandedness)
```

Arguments

`bamfile_path` path to bam file

`granges_of_interest`
a GRanges object

`strandedness` one of c("reverse", "unstranded")

createBamPath *create a bam path*

Description

a helper function to creat a bampath from some metadata information. Also checks if index exists

Usage

```
createBamPath(
  run_number,
  fastq_filename,
  lts_align_expr_prefix,
  bam_suffix = "_sorted_aligned_reads_with_annote.bam",
  test = FALSE
)
```

Arguments

`run_number` the run_number (mind the leading zeros for old runs) of the run

`fastq_filename` the fastq filename, preferrably without the extension or any leading path info. However, an effort has been made to deal with full paths and extensions

`lts_align_expr_prefix`
the location of the run directories. Eg, if you are mounted and on your local computer, it might be something like "/mnt/htcf_lts/lts_align_expr"

`bam_suffix` the common bam suffix for all bam files stored in /lts. Eg, it might be something like "_sorted_aligned_reads_with_annote.bam"

`test` boolean, default FALSE. Set to TRUE if testing this function

Value

a verified filepath to the bam file

createEnvPertSet *filter combined_df for environmental perturbation sample set*

Description

filter combined_df for environmental perturbation sample set

Usage

```
createEnvPertSet(combined_df)
```

Arguments

`combined_df`	the combined tables of the database, returned directly from getMetadata() (meaning, the df hasn't been augmented after pulling from the database)

Value

environmental pertubation set

createInductionSetTally *create 90 minute induction set tally*

Description

create 90 minute induction set tally

Usage

```
createInductionSetTally(
  induction_meta_qual,
  sorted_passing_induction_meta_qual,
  iqr_fltr_rle_summary,
  grant_df
)
```

Arguments

`induction_meta_qual`	the metadata of the entire set, unfiltered
`sorted_passing_induction_meta_qual`	metadata (with quality columns) filtered for manual/auto status
`iqr_fltr_rle_summary`	sorted_passing_meta_qual filtered for IQR
`grant_df`	the definition of the 90minuteInduction set. This object is available in the brentlabRnaSeqTools package

---

`createNinetyMinInductionModelMatrix`

Create the libraryProtocol + libraryDate model matrix with the earliest date of each library protocol dropped

---

Description

Create the libraryProtocol + libraryDate model matrix with the earliest date of each library protocol dropped

Usage

```
createNinetyMinInductionModelMatrix(metadata_df)
```

Arguments

`metadata_df`	the joined tables of the database (biosample to quality assess)

Value

a model matrix constructed as specified in the description

`createNinetyMinuteInductionSet`

The current definition of the 90 minute induction dataset, according to the 2016 grant summary (loaded into environment, see head(grant_df)) – single KO only

---

Description

The current definition of the 90 minute induction dataset, according to the 2016 grant summary (loaded into environment, see head(grant_df)) – single KO only

Usage

```
createNinetyMinuteInductionSet(metadata, grant_df)
```

Arguments

`metadata`	is the combined tables of the metadata database
`grant_df`	is the 2016 grant summary TODO: put this in DATA

Value

the set metadata – single KO only

`createNinetyMinuteInductionWithDoubles` *The current definition of the 90 minute induction dataset, according to the 2016 grant summary (loaded into environment, see head(grant_df)) – single and double KO*

Description

The current definition of the 90 minute induction dataset, according to the 2016 grant summary (loaded into environment, see head(grant_df)) – single and double KO

Usage

```
createNinetyMinuteInductionWithDoubles(metadata, grant_df)
```

Arguments

`metadata`	is the combined tables of the metadata database
`grant_df`	is the 2016 grant summary TODO: put this in DATA

Value

the set metadata

`createQCdatabase` *create a sqlite database to hold the 'custom' qc data*

Description

create a sqlite database to hold the 'custom' qc data

Usage

```
createQCdatabase(database_dirpath)
```

Arguments

`database_dirpath`
path to containing directory of new qc database

Value

database path

`database_info` *URLS to active databases*

Description

A list containing the urls to active databases. Named by organism (eg 'kn99' or 's288cr64')

Usage

```
database_info
```

Format

A list with named slots

kn99_host host of the database server, eg ec2-18-224-181-136.us-east-2.compute.amazonaws.com

kn99_db_name cryptococcus database name. probably something like kn99_database

kn99_urls urls to all tables in kn99 database

s288cr64_host host address of the yeast s288cr64 database, eg ec2-3-131-85-10.us-east-2.compute.amazonaws.com

s288cr64_db_name yeast database name. probably something like yeast_database

s288cr64_urls urls to all tables in s288cr64 database ...

Source

`https://rnaseq-databases-documentation.readthedocs.io/en/latest/`

`deseqObjectWithProtocolSpecificSizeFactors`

create deseq object with protocol specific size factors

Description

create deseq object with protocol specific size factors

Usage

```
deseqObjectWithProtocolSpecificSizeFactors(passing_qc1_meta_qual, raw_counts)
```

Arguments

`passing_qc1_meta_qual`
can be any metadata df, but if you're going to run deseq you may want to filter it for passing samples first

`raw_counts` a dataframe of raw counts with genes in the rows and samples in the columns. sample names must be the same as the fastqFileName column in passing_qc1_meta_qual.

Value

a deseq data object with size factors calculated within the library protocol groups

`determineLibraryStrandedness`

from the metadata libraryProtocol column, determine the library strandedness.

Description

Currently set up for cryptococcus. E7420L returns reverse, SolexaPrep returns unstranded. default return is unstranded

Usage

```
determineLibraryStrandedness(library_protocol)
```

Arguments

`library_protocol`
the library protocol of the sample (determines strandedness of the library)

Value

the strandedness of the library based on the value in the libraryProtocol column, or 'unstranded' by default

Note

: default is unstranded

`examineSingleGroupWithLibDateSizeFactors` *temporary function to examine only PBS samples, eg*

Description

gets all samples in qc_passing_metadata with size_factor_subset_param same as samples in row_filter

Usage

```
examineSingleGroupWithLibDateSizeFactors(
  qc1_passing_metadata,
  raw_counts,
  row_filter,
  size_factor_subset_param
)
```

Arguments

`qc1_passing_metadata`
metadata passing auto/manual flags

`raw_counts` must include at least the sames in the metadata

`row_filter` is a boolean vector, eg qc1_passing_env_pert_meta_qual$MEDIUM == "PBS"

`size_factor_subset_param`
eg LIBRARYDATE – the column to group the samples for size factor calculation

Value

a list with slots metadata, raw_counts, size_factors

`featureGRanges` *Given a GenomicFeatures annotation_db and a gene_id, extract an GRanges object of the cds*

Description

Given a GenomicFeatures annotation_db and a gene_id, extract an GRanges object of the cds

Usage

```
featureGRanges(annotation_db, gene_id, feature)
```

Arguments

`annotation_db`	a GenomicFeatures db. You can either get this from the bioconductor resources, or create your own with a gtf
`gene_id`	the ID of a gene in the db. Eg, for cryptococcus CKF44_05222
`feature`	one of c("cds", "exon"), determins which feature to extract from the annotations

Value

an IRanges object of the given gene's exons

References

GenomicRanges::GRanges, GenomicFeatures

`fltrLowReplicateParams` *filter low replicate parameters from metadata*

Description

given a model formula, remove samples with less than a specified number of replicates from the metadata

Usage

```
fltrLowReplicateParams(metadata_df, design_formula, replicate_threshold = 2)
```

Arguments

`metadata_df`	a data frame that contains at least the model paramters of interest
`design_formula`	an R formula, eg ~libraryDate+treatment, of parameters contained in the metadata_df
`replicate_threshold`	the number of replicates below which samples will be removed. Default is 2

Value

the input metadata with samples in replicate groups with less than the specified thershold filtered out

`getBamIndexPath` *helper function to add .bai to bam path*

Description

helper function to add .bai to bam path

Usage

```
getBamIndexPath(bamfile_path)
```

Arguments

`bamfile_path` path to bamfile

`getCoverageOverRegion` *create a dataframe of coverage by nucleotide over a given locus*

Description

create a dataframe of coverage by nucleotide over a given locus

Usage

```
getCoverageOverRegion(
  bamfile_path,
  annote_db,
  gene_id,
  strandedness,
  quality_threshold = 20L,
  coverage_threshold = 0,
  lts_align_expr_prefix = Sys.getenv("LTS_ALIGN_EXPR_PREFIX"),
  bamfile_suffix = Sys.getenv("BAM_SUFFIX")
)
```

Arguments

`bamfile_path` path to a bam file @seealso brentlabRnaSeqTools::createBamPath()

`annote_db` a GenomicFeatures TxDb object. Maybe one made from a gtf, eg txdb = makeTxDbFromGFF("data/liftoff format = "gtf")

`gene_id` a gene_id of interest – must be in the gene names of the annote_db object

`strandedness` one of c("reverse", "unstranded"). NOTE: forward only strand NOT currently configured

`quality_threshold`
quality threshold above which reads will be considered. 20l is default, which is chosen b/c it is the default for HTSeq

`coverage_threshold`
minimum read count above which to consider reads. Default is 0

`lts_align_expr_prefix`
= path to the directory which stores the run_12345_samples run directories. For example, /lts/mblab/Crypto/rnaseq_data/lts_align_expr. By default, this looks in your .Renviron for a key LTS_ALIGN_EXPR_PREFIX

`bamfile_suffix`
= whatever is appended after the fastqFileName (no extension). Currently, this is "_sorted_aligned_reads_with_annote.bam". By default, this looks in your .Renviron for a key BAM_SUFFIX

References

GenomicRanges, Rsamtools

`getMetadata` *Get the combined metadata as a tibble from a remote database*

Description

Join the biosample, rnasample, s1sample, s2sample, library, fastqFiles and qualityAssessment tables (in that order, left joins) and return the result as a tibble

Use the RPostgres package to connect to a remote postgresql database, do the table joining, and return the joined metadata as a tibble. The database connection is closed

Usage

```
getMetadata(database_host, database_name, database_user, database_password)
```

Arguments

`database_host`
if connecting to a database hosted on AWS, it might be something like ec2-54-83-201-96.compute-1.amazonaws.com

`database_name`
name of the database, eg for cryptococcus kn99, the database might be named kn99_database. Check with the documentation, whoever set up the database, or get into the server and check directly

`database_user`
a user of the actual database, with some level of permissions. You’ll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

`database_password`
password to the database user. You’ll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

Value

A DBI connection to the remote database

Note

for information on using R environmental files, see https://support.rstudio.com/hc/en-us/articles/360047157094-Managing-R-with-Rprofile-Renviron-Rprofile-site-Renviron-site-rsession-conf-

Source

https://rpostgres.r-dbi.org/

getRawCounts *Get combined raw counts*

Description

Get combined raw counts

Usage

```
getRawCounts(database_host, database_name, database_user, database_password)
```

Arguments

`database_host` if connecting to a database hosted on AWS, it might be something like ec2-54-83-201-96.compute-1.amazonaws.com

`database_name` name of the database, eg for cryptococcus kn99, the database might be named kn99_database. Check with the documentation, whoever set up the database, or get into the server and check directly

`database_user` a user of the actual database, with some level of permissions. You'll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

`database_password`
password to the database user. You'll need to check with the database maintainer for this. It is suggested that you use a .Renviron file in your local project (make sure it is completely ignored by git, R, etc) to store this info

Value

a gene by samples dataframe of all counts

getRunNumberLeadingZero

correct run number to add leading zero where approprirate

Description

correct run number to add leading zero where approprirate

Usage

```
getRunNumberLeadingZero(run_number)
```

Arguments

`run_number`	a run number, most likely from the metadata runNumber field

getUserAuthToken

get (via a http POST request) your user authentication token from the database

Description

get (via a http POST request) your user authentication token from the database

Usage

```
getUserAuthToken(url, username, password)
```

Arguments

`url`	check the database_info variable. for configured organisms, you can find this under database_info$organism$token_auth
`username`	a valid username for the database. If you don't have one, then you'll need to ask for one to be created
`password`	password associated with your username

Value

the auth token associated with the username and password

Note

do not save your auth token in a public repository. For example, you might put it in your .Renviron and then make sure that your .Renviron is in your .gitignore. Otherwise, save it outside of a github tracked directory or otherwise ensure that it will not be pushed up to github

`grant_df` *the 2016 grant summary represented as a dataframe*

Description

also check the google sheet

Usage

```
grant_df
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 164 rows and 3 columns.

`graphYeastTimeCourse` *Plot time vs normalized counts of a given gene over n samples, faceted by librarydate and run*

Description

graph_output = graphTimeCourse(cst6_sample_metadata, 'CST6', 'YIL036W', cst6_norm_counts)

Usage

```
graphYeastTimeCourse(metadata_df, genotype_1, gene_id, norm_counts)
```

Arguments

`metadata_df`	where metadata$fastqFileName is equal in format to colnames(raw_counts)
`genotype_1`	is the entry in the metadata genotype1 column which you would like to examine
`gene_id`	is the gene id (the systematic name as opposed to the 'common name') most likely corresponding to genotype_1. This is how the correct row is extracted from the count data
`norm_counts`	MUST have rownames assigned to the gene_ids includes at least all samples in metadata

Value

a ggplot graph of the gene_id in question, faceted on run and library date

Note

see the vignette called yeast_timecourse_qc.Rmd. There is also sample data that comes with this package, so you can run the vignette verbatim to see how it works. The vignette includes code that

isNumeric *test if argument is numeric*

Description

copied directly from the limma codebase

Usage

```
isNumeric(x)
```

Arguments

x	any R object

Details

copied from the limma docs: This function is used to check the validity of arguments for numeric functions. It is an attempt to emulate the behavior of internal generic math functions. IsNumeric differs from is.numeric in that data.frames with all columns numeric are accepted as numeric.

listTables *list tables in databse*

Description

list tables in databse

Usage

```
listTables(db)
```

Arguments

db	a connection to the database

Value

all tables in database

See Also

https://www.postgresqltutorial.com/postgresql-show-tables/

`locusLog2Cpm` *calculate log2cpm for a given locus*

Description

calculate log2cpm for a given locus

Usage

```
locusLog2Cpm(bam_path, strandedness, locus_granges, lib_size)
```

Arguments

`bam_path`	path to bam file. Note: the index with extension .bai also must exist
`strandedness`	the strandedness of the library
`locus_granges`	a granges object specifying the locus over which to count
`lib_size`	the number of reads in the library

Value

either 0, if there are no counts, or the log2cpm of the counts over the locus. counting is done via the same method as default HTSeq

`patchTable` *PATCH entries in database table*

Description

using the package httr, update entries in certain fields in given rows of a table

Usage

```
patchTable(database_table_url, auth_token, update_df, id_col)
```

Arguments

`database_table_url`	NO TRAILING '/'. eg "http://18.224.181.136/api/v1/QualityAssess"
`auth_token`	see brentlabRnaSeqTools::getUserAuthToken()
`update_df`	a dataframe, preferrably a tibble, already read in, subsetted. Columns must be correct data type for db table
`id_col`	name of the id column of the table. this number will be appended to the url to create the uri for the record

Value

a list of httr::response() objects

plotCoverageOverLocus *plot coverage over locus*

Description

ggbio plot with transcripts track and coverage track

Usage

```
plotCoverageOverLocus(
  bamfile_path,
  annote_db,
  gene_id,
  strandedness,
  quality_threshold = 20L
)
```

Arguments

`bamfile_path` path to a bam file @seealso brentlabRnaSeqTools::createBamPath()

`annote_db` a GenomicFeatures TxDb object. Maybe one made from a gtf, eg txdb = makeTxDbFromGFF("data/liftoff format = "gtf")

`gene_id` a gene_id of interest – must be in the gene names of the annote_db object

`strandedness` one of c("reverse", "unstranded"). NOTE: forward only strand NOT currently configured

`quality_threshold` quality threshold above which reads will be considered. 20l is default, which is chosen b/c it is the default for HTSeq

postCounts *post counts to database*

Description

using the package httr, post the raw count .csv, which is the compiled counts for a given run, to the database

Usage

```
postCounts(
  database_counts_url,
  run_number,
  auth_token,
  new_counts_path,
  fastq_table,
  count_file_suffix = "_read_count.tsv"
)
```

Arguments

`database_counts_url`
: eg database_info$kn99_urls$Counts database_info is a saved data object in this package

`run_number`
: the run number of this counts sheet – this is important b/c fastqFileNames aren't necessarily unique outside of their runs

`auth_token`
: see brentlabRnaSeqTools::getUserAuthToken()

`new_counts_path`
: path to the new counts csv

`fastq_table`
: a recent pull of the database fastq table

`count_file_suffix`
: the suffix appended to the fastqFileName in the count file column headings. default is "_read_count.tsv"

Value

a list of httr::response() objects

`postFastqSheet` *post new fastq sheet to database*

Description

post new fastq sheet to database

Usage

```
postFastqSheet(database_fastq_url, auth_token, new_fastq_path)
```

Arguments

`database_fastq_url`
: eg database_info$kn99_urls$FastqFiles database_info is a saved data object in this package

`auth_token`
: see brentlabRnaSeqTools::getUserAuthToken()

`new_fastq_path`
: path to new fastq sheet

postQcSheet *post new qc sheet to database*

Description

using the package httr, post the new qc sheet to the database

Usage

```
postQcSheet(database_qc_url, auth_token, run_number, new_qc_path, fastq_table)
```

Arguments

`database_qc_url`
eg database_info$kn99_urls$QualityAssess. database_info is a saved data object in this package

`auth_token` see brentlabRnaSeqTools::getUserAuthToken

`run_number` the run number of this qc sheet – this is important b/c fastqFileNames aren't necessarily unique outside of their runs

`new_qc_path` path to the new counts csv

`fastq_table` a recent pull of the database fastq table

Value

a list of httr::response() objects

Note

there can be problems with dependencies and the rename function. this is working for now, but see here for more info https://statisticsglobe.com/r-error-cant-rename-columns-that-dont-exist

proteinCodingCount *get total protein coding count from count dataframe*

Description

given a count dataframe with gene_ids as rownames and quantification in a column called raw_counts, return sum of protein coding genes

Usage

```
proteinCodingCount(counts, protein_coding_gene_ids)
```

Arguments

`counts`	a dataframe with gene_ids in the rownames and (at minimum) a quantification column called raw_counts
`protein_coding_gene_ids`	a list of gene ids considered protein coding (must correspond with counts rownames)

`qcGenotypeAndMarkerCoverage`

calcluate genotype and marker coverages for a given metadata df

---

Description

this produces a sheet that can be used to fill the 'custom' qc table for the brentlab kn99 and yeast databases

Usage

```
qcGenotypeAndMarkerCoverage(metadata_df, annote_db, bam_prefix, bam_suffix)
```

Arguments

`metadata_df`	a metadata sheet with at least the columns fastqFileName, genotype1, genotype2, runNumber, libraryProtocol
`annote_db`	a connection to a local database – see createQCdatabase()
`bam_prefix`	the directory that contains the sequencing runs,, eg if locally mounted maybe "/mnt/htcf_lts/lts_align_expr"
`bam_suffix`	the stuff appended to the end of the fastqFileName (minus the .fastq.gz). This might just be ".bam", but could be something like "_sorted_aligned_reads_with_annote.bam"

Value

a sample sheet with the following columns: fastqFileName, fastqFileNumber, runNumber, column_param, locus, coverage

Note

together, the bam_prefix/run_runNumber_samples/align/fastqFileName_bam_suffix form the path to the bamfile

qualityAssessmentFilter *filter for manual passes (overrides auto fail) and automatic passes (unless auto failed)*

Description

filter for manual passes (overrides auto fail) and automatic passes (unless auto failed)

Usage

```
qualityAssessmentFilter(metadata)
```

Arguments

`metadata`	dataframe from the database

Value

a metadata dataframe with column names cast to upper

readInData *read in columnar data*

Description

given a csv, tsv or excel sheet, use the right function to read in the data

Usage

```
readInData(path)
```

Arguments

`path`	path to a csv, tsv or xlsx

removeOneRedoIqr *progressively remove max IQR sample and recalculate*

Description

progressively remove max IQR sample and recalculate

Usage

```
removeOneRedoIqr(sample_set, logged_norm_counts)
```

Arguments

`sample_set`	the metadata
`logged_norm_counts`	counts on log2 scale and normalized

removeParameterEffects *remove some effects from the counts*

Description

subtract effect from norm counts of a single factor from coef x design. coef is in normalized log space. dds must have been created with model.matrix

Usage

```
removeParameterEffects(deseq_object, col_indicies)
```

Arguments

`deseq_object`	a deseq data object REQUIRED: the object must have been created with a model.matrix rather than a formula for the design argument
`col_indicies`	a numeric vector corresponding to the column indicies of the batch parameters you'd like to remove

Value

a log2 scale gene by samples matrix with desired effects removed

Note

works for both formula and model.matrix designs in the dds object

`rleByReplicateGroup` *calculate RLE by replicate groups*

Description

calculate RLE by replicate groups

Usage

```
rleByReplicateGroup(replicates_vector, gene_quants, log2_transformed_flag)
```

Arguments

`replicates_vector`
: a list of lists where each sublist represents a replicate group. Entries must be a metadata parameter, such as fastqFileName, that corresponds to the columns of the counts. Suggestion: use something like these dplyr functions to create the list of lists group_by() %>% group_split %>% pull(fastqFileName)

`gene_quants`
: a gene x sample dataframe with values as some sort of gene quantification (eg normed counts, or log2(norm_counts) with some effect removed), possibly already logged (@see already_logged_flag)

`log2_transformed_flag`
: a boolean where TRUE means the counts are already in log2 space

Value

a list of dataframes for each replicate group in replicateS_sample_list, each with dimensions gene x sample. values are RLE of the gene in a given sample

References

rlePlotCompareEffectRemoved() to plot the norm counts and removedEffect 'counts' on the same plot

`rlePlot` *plot RLE for a given column filter (eg, metadatametadata$MEDIUM == 'PBS'$FASTQFILENAME would give a list of fastqFileNames to filter)*

Description

plot RLE for a given column filter (eg, metadatametadata$MEDIUM == 'PBS'$FASTQFILENAME would give a list of fastqFileNames to filter)

Usage

```
rlePlot(deseq_object, model_matrix, column_filter, title)
```

Arguments

`deseq_object`	a deseq object with results from the DESeq() call
`model_matrix`	the deseq_object model matrix
`column_filter`	a vector of fastqFileNames (or whatever the columns – samples – are called)
`title`	of the plots

Value

list with slots norm_count_rle and effect_removed_rle

`rlePlotCompareEffectRemoved` *plots output of rleSummaryByReplicateGroup*

Description

plots output of rleSummaryByReplicateGroup

Usage

```
rlePlotCompareEffectRemoved(
  norm_counts_rle,
  removed_effect_rle,
  metadata_df,
  title
)
```

Arguments

`norm_counts_rle`	output of calculateRLE (maybe one of the sublists in rleByReplicateGroup())
`removed_effect_rle`	see norm_counts_rle, but after removing some batch effects
`metadata_df`	metadata with at least FASTQFILENAME and LIBRARYDATE
`title`	title of the plot

Value

a ggplot with both the norm counts (more transparent) and removedEffect 'counts' on the same plot

`rlePlot_helper` *the actual plotting function for rlePlot*

Description

the actual plotting function for rlePlot

Usage

```
rlePlot_helper(count_df, log2_transformed_flag, title)
```

Arguments

`count_df`	counts in gene x sample
`log2_transformed_flag`	boolean where TRUE indicates the counts are in log2 space
`title`	title of the output plot

Value

a ggplot

`rleSummary` *rleSummary calculates summary statistics of rleFullTable*

Description

rleSummary calculates summary statistics of rleFullTable

Usage

```
rleSummary(rle_table_full)
```

Arguments

`rle_table_full`	the output of rleFullTable

Value

a dataframe sample by rle summary statistics

runSVA *run SVA*

Description

run SVA

Usage

```
runSVA(raw_counts, null_model_matrix, full_model_matrix)
```

Arguments

`raw_counts`	raw gene counts in the shape gene x samples where ncols matches nrow of metadata (samples == samples)
`null_model_matrix`	a model matrix respresenting only the batch effects. Could possibly be intercept only
`full_model_matrix`	the full model describing the experiment. Critically, this includes the parameter of interest

run_numbers_with_leading_zero *A named list containing a run number without a leading zero, eg 647, with the value being the same runnumber with a leading 0, eg 0647.*

Description

this is best remedied in the database itself by forcing the column to be a string and adding the 0s

Usage

```
run_numbers_with_leading_zero
```

Format

An object of class `list` of length 13.

`selectQaColumns` *select fastqFileName, fastqFileNumber, and a pre-determined set of QC columns from a metadata df*

Description

select fastqFileName, fastqFileNumber, and a pre-determined set of QC columns from a metadata df

Usage

```
selectQaColumns(metadata)
```

Arguments

`metadata` a metadata df with at least the columns listed in the select statement (see source code – notably, must include interquartile range). Column names will be cast to uppper and returned as uppers

Note

the column names for the metadata will be cast to upper and returned in upper

must include Interquartile range. Think about removing this – user could merge with IQR df after selecting these cols

`strandedScanBamParam` *create coverage scanbamparam object*

Description

helper function to create ScanBamParam object with appropriate strandedness information

Usage

```
strandedScanBamParam(locus_granges, strandedness, quality_threshold = 20L)
```

Arguments

`locus_granges` a granges object for a given gene (or some other feature on only one strand)

`strandedness` one of c("reverse", "unstranded"). NOTE: forward only strand NOT currently configured

`quality_threshold`
quality threshold above which reads will be considered. 20l is default, which is chosen b/c it is the default for HTSeq

testBamPath *test bam path*

Description

test bam path

Usage

```
testBamPath(metadata_df)
```

Index