Package 'catalogueR'

July 5, 2020

Type Package

Title QTL data extraction and colocalization

Version 0.1.0

Description API access to eQTL Catalogue full summary stats and colocalization functions.

Encoding UTF-8

LazyData true

License MIT + file LICENSE

URL https://github.com/RajLabMSSM/catalogueR

BugReports https://github.com/RajLabMSSM/catalogueR/issues

Depends R (>= 3.6.0)

biocViews

Imports dplyr,
data.table,
parallel,
ggplot2,
readr,
stringr,
httr,
rlang,
jsonlite,
tidyverse,
coloc,
biomaRt,
wiggleplotr,
GenomicRanges,
AnnotationDbi,
EnsDb.Hsapiens.v75,
XGR

Suggests pbmcapply

RoxygenNote 7.1.0

R topics documented:

`catalogueR-package` *Rapid querying, colocalization, and plotting of summary stats from the eQTL Catalogue*

Description

The functions in **catalogueR** are partly derived from the following eQTL Catalogue tutorial. Additional eQTL Catalogue Resources:

- GitHub
- In-depth API documentation
- FTP server

Details

Notes on parallelization: There's multiple levels to parallelize on. You can only chooose one level at a time:

`multithread_qtl=T` Across QTL datasets

`multithread_loci=T` Across loci

`multithread_tabix=T` Within tabix files

You can also get a speedup by using tabix instead of the rest API Test: For 3 loci, and X QTL datasets:

RESTful API: 7.5 minutes

Tabix: 27 seconds (*clear winner! ~17x speedup) That said, if you're only query a small number of specific SNPs (rather than a large range), the RESTful API can sometimes be faster.

Author(s)

Maintainer: Brian Schilder <brian_schilder@alumni.brown.edu>

Other contributors:

- Jack Humphrey [contributor]

See Also

Useful links:

- https://github.com/RajLabMSSM/catalogueR
- Report bugs at https://github.com/RajLabMSSM/catalogueR/issues

---

annotate_tissues *Annotate QTL datasets with metadata*

Description

Annotate QTL datasets with metadata

Usage

```
annotate_tissues(dat)
```

Examples

```
dat <- annotate_tissues(dat)
```

BST1 echolocatoR *output example (BST1 locus)*

Description

An example results file after running `finemap_loci` on the *BST1* locus.

Usage

```
BST1
```

Format

data.table

SNP SNP RSID

CHR Chromosome

POS Genomic positiion (in basepairs) ...

Details

Data originally comes from the Parkinson's disease GWAS by Nalls et al. (bioRxiv).

Source

https://www.biorxiv.org/content/10.1101/388165v3

See Also

Other Nalls23andMe_2019: LRRK2, MEX3C

Examples

```
## Not run:
root_dir <- "~/Desktop/Fine_Mapping/Data/GWAS/Nalls23andMe_2019"
locus_dir <- "BST1/Multi-finemap/Multi-finemap_results.txt"
BST1 <- data.table::fread(file.path(root_dir,locus_dir))
BST1 <- update_CS_cols(finemap_dat=BST1)
BST1 <- find_consensus_SNPs(finemap_dat=BST1)
data.table::fwrite(BST1,"inst/extdata/Nalls23andMe_2019/BST1_Nalls23andMe_2019_subset.tsv.gz", sep="\t")
usethis::use_data(BST1, overwrite = T)

## End(Not run)
```

---

ensembl_to_hgnc *Convert ENSEMBL IDs to HGNC gene symbols*

Description

Convert ENSEMBL IDs to HGNC gene symbols

Usage

```
ensembl_to_hgnc(ensembl_ids, unique_only = T, verbose = T)
```

See Also

Other utils: gather_files(), hgnc_to_ensembl(), liftover()

Examples

```
ensembl_ids <- c("ENSG00000176697","ENSG00000128573","ENSG00000109743")
gene_symbols <- ensembl_to_hgnc(ensembl_ids=ensembl_ids)
```

`eQTL_Catalogue.fetch` *2. Query eQTL Catalogue datasets by region*

Description

Choose between tabix (faster for larger queries) or RESTful API (faster for small queries).

Usage

```
eQTL_Catalogue.fetch(
  unique_id,
  quant_method = "ge",
  infer_region = T,
  gwas_data = NULL,
  is_gwas = F,
  nThread = 1,
  use_tabix = T,
  chrom = NULL,
  bp_lower = NULL,
  bp_upper = NULL,
  multithread_tabix = F,
  add_qtl_id = T,
  convert_genes = T,
  verbose = T
)
```

Arguments

`quant_method`	eQTL Catalogue actually contains more than just eQTL data. For each dataset, the following kinds of QTLs can be queried: **gene expression QTL** `quant_method="ge"` (*default*) or `quant_method="microarray"`, depending on the dataset. **catalogueR** will automatically select whichever option is available. **exon expression QTL** **under construction** `quant_method="ex"` **transcript usage QTL** **under construction** `quant_method="tx"` **promoter, splice junction and 3' end usage QTL** **under construction** `quant_method="txrev"`
`nThread`	The number of CPU cores you want to use to speed up your queries through parallelization.
`use_tabix`	Tabix is about ~17x faster (*default:* =T) than the REST API (=*F*).
`bp_lower`	Minimum basepair position of the query window.
`bp_upper`	Maxmimum basepair position of the query window.
`verbose`	Show more (=T) or fewer (=F) messages.

See Also

Other eQTL Catalogue: `eQTL_Catalogue.iterate_fetch()`, `eQTL_Catalogue.query()`, `eQTL_Catalogue.search_m`
`fetch_restAPI()`, `fetch_tabix()`, `merge_gwas_qtl()`, `meta`

Examples

```
data("meta"); data("BST1");
gwas.qtl <- eQTL_Catalogue.fetch(unique_id=meta$unique_id[1], gwas_data=BST1)
```

`eQTL_Catalogue.iterate_fetch` *Iterate queries to* eQTL Catalogue

Description

Uses coordinates from stored summary stats files (e.g. GWAS) to determine which regions to query from *eQTL Catalogue*.

Usage

```
eQTL_Catalogue.iterate_fetch(
  sumstats_paths,
  output_dir = "./catalogueR_queries",
  qtl_id,
  quant_method = "ge",
  infer_region = T,
  use_tabix = T,
  multithread_loci = T,
  multithread_tabix = F,
  nThread = 4,
  split_files = T,
  merge_with_gwas = F,
  force_new_subset = F,
  progress_bar = F,
  genome_build = "hg19",
  verbose = T
)
```

Arguments

`sumstats_paths` A list of paths to any number of summary stats files whose coordinates you want to use to make queries to eQTL Catalogue. If you wish to add custom names to the loci, simply add these as the names of the path list (e.g. `c(BST1="<path>/<to>/<BST1_file>",LRRK2="<path>/<to>/<LRRK2_file>")`). Otherwise, loci will automatically named based on their min/max genomic coordinates.

The minimum columns in these files required to make queries include:

SNP RSID of each SNP.

CHR Chromosome (can be in "chr12" or "12" format).

POS Genomic position of each SNP.

... Optional extra columns.

`output_dir` The folder you want the merged gwas/qtl results to be saved to (set `output_dir=F` if you don't want to save the results). If `split_files=F`, all query results will be merged into one and saved as *<output_dir>/eQTL_Catalogue.tsv.gz*. If `split_files=T`, all query results will instead be split into smaller files and stored in *<output_dir>/*.

quant_method
: eQTL Catalogue actually contains more than just eQTL data. For each dataset, the following kinds of QTLs can be queried:

  gene expression QTL `quant_method="ge"` (*default*) or `quant_method="microarray"`, depending on the dataset. **catalogueR** will automatically select whichever option is available.

  exon expression QTL **under construction** `quant_method="ex"`

  transcript usage QTL **under construction** `quant_method="tx"`

  promoter, splice junction and 3' end usage QTL **under construction** `quant_method="txrev"`

use_tabix
: Tabix is about ~17x faster (*default:* =T) than the REST API (=*F*).

nThread
: The number of CPU cores you want to use to speed up your queries through parallelization.

split_files
: Save the results as one file per QTL dataset (with all loci within each file). If this is set to `=T`, then this function will return the list of paths where these files were saved. A helper function is provided to import and merge them back together in R. If this is set to `=F`, then this function will instead return one big merged data.table containing results from all QTL datasets and all loci. `=F` is not recommended when you have many large loci and/or many QTL datasets, because you can only fit so much data into memory.

merge_with_gwas
: Whether you want to merge your QTL query results with your GWAS data (convenient, but takes up more storage).

force_new_subset
: By default, **catalogueR** will use any pre-existing files that match your query. Set `force_new_subset=T` to override this and force a new query.

progress_bar
: Show progress bar during parallelization across loci. *WARNING!*: Progress bar (via `pbmclapply`) only works on Linux/Unix systems (e.g. mac) and NOT on Windows.

genome_build
: The genome build of your query coordinates (e.g. `gwas_data`). If your coordinates are in *hg19*, **catalogueR** will automatically lift them over to *hg38* (as this is the build that eQTL Catalogue uses).

verbose
: Show more (`=T`) or fewer (`=F`) messages.

See Also

Other eQTL Catalogue: `eQTL_Catalogue.fetch()`, `eQTL_Catalogue.query()`, `eQTL_Catalogue.search_metadata()`, `fetch_restAPI()`, `fetch_tabix()`, `merge_gwas_qtl()`, `meta`

Examples

```
sumstats_paths <- example_sumstats_paths()
qtl_id  <- eQTL_Catalogue.list_datasets()$unique_id[1]
GWAS.QTL <- eQTL_Catalogue.iterate_fetch(sumstats_paths=sumstats_paths, qtl_id=qtl_id, force_new_subset=T, m
```

eQTL_Catalogue.list_datasets *List available eQTL datasets*

Description

Does some additional preprocessing of metadata to categorize tissue types.

Usage

```
eQTL_Catalogue.list_datasets(save_dir = F, force_new = F, verbose = F)
```

Examples

```
meta <- eQTL_Catalogue.list_datasets()
```

eQTL_Catalogue.query *Iterate queries to* eQTL Catalogue

Description

Deetermines which datasets to query using `qtl_search`. Uses coordinates from stored summary stats files (e.g. GWAS) to determine which regions to query from *eQTL Catalogue*. Each locus file can be stored separately, or merged together to form one large file with all query results.

Usage

```
eQTL_Catalogue.query(
  sumstats_paths = NULL,
  output_dir = "./catalogueR_queries",
  qtl_search = NULL,
  use_tabix = T,
  nThread = 4,
  quant_method = "ge",
  infer_region = T,
  split_files = T,
  merge_with_gwas = T,
  force_new_subset = F,
  genome_build = "hg19",
  progress_bar = T,
  verbose = T
)
```

Arguments

`sumstats_paths` A list of paths to any number of summary stats files whose coordinates you want to use to make queries to eQTL Catalogue. If you wish to add custom names to the loci, simply add these as the names of the path list (e.g. `c(BST1="<path>/<to>/<BST1_file>",LRRK2="<path>/<to>/<LRRK2_file>")`). Otherwise, loci will automatically named based on their min/max genomic coordinates.

The minimum columns in these files required to make queries include:

SNP RSID of each SNP.

CHR Chromosome (can be in "chr12" or "12" format).

POS Genomic position of each SNP.

... Optional extra columns.

`output_dir` The folder you want the merged gwas/qtl results to be saved to (set `output_dir=F` if you don't want to save the results). If `split_files=F`, all query results will be merged into one and saved as *<output_dir>/eQTL_Catalogue.tsv.gz*. If `split_files=T`, all query results will instead be split into smaller files and stored in *<output_dir>/*.

`qtl_search` This function will automatically search for any datasets that match your criterion. For example, if you search "Alasoo_2018", it will query the datasets:

- Alasoo_2018.macrophage_naive
- Alasoo_2018.macrophage_Salmonella
- Alasoo_2018.macrophage_IFNg+Salmonella

You can be more specific about which datasets you want to include, for example by searching: "Alasoo_2018.macrophage_IFNg". You can even search by tissue or condition type (e.g.c("blood","brain")) and any QTL datasets containing those substrings (case-insensitive) in their name or metadata will be queried too.

`use_tabix` Tabix is about ~17x faster (*default:* =T) than the REST API (=*F*).

`nThread` The number of CPU cores you want to use to speed up your queries through parallelization.

`quant_method` eQTL Catalogue actually contains more than just eQTL data. For each dataset, the following kinds of QTLs can be queried:

gene expression QTL `quant_method="ge"` (*default*) or `quant_method="microarray"`, depending on the dataset. **catalogueR** will automatically select whichever option is available.

exon expression QTL **under construction** `quant_method="ex"`

transcript usage QTL **under construction** `quant_method="tx"`

promoter, splice junction and 3' end usage QTL **under construction** `quant_method="txrev"`

`split_files` Save the results as one file per QTL dataset (with all loci within each file). If this is set to `=T`, then this function will return the list of paths where these files were saved. A helper function is provided to import and merge them back together in R. If this is set to `=F`, then this function will instead return one big merged data.table containing results from all QTL datasets and all loci. `=F` is not recommended when you have many large loci and/or many QTL datasets, because you can only fit so much data into memory.

`merge_with_gwas`

Whether you want to merge your QTL query results with your GWAS data (convenient, but takes up more storage).

`force_new_subset`
: By default, **catalogueR** will use any pre-existing files that match your query. Set `force_new_subset=T` to override this and force a new query.

`genome_build`
: The genome build of your query coordinates (e.g. `gwas_data`). If your coordinates are in *hg19*, **catalogueR** will automatically lift them over to *hg38* (as this is the build that eQTL Catalogue uses).

`progress_bar`
: `progress_bar=T` allows progress to be monitored even when multithreading enabled. Requires R package `pbmcapply`.

`verbose`
: Show more (`=T`) or fewer (`=F`) messages.

See Also

Other eQTL Catalogue: `eQTL_Catalogue.fetch()`, `eQTL_Catalogue.iterate_fetch()`, `eQTL_Catalogue.search_m`
`fetch_restAPI()`, `fetch_tabix()`, `merge_gwas_qtl()`, `meta`

Examples

```
sumstats_paths <- example_sumstats_paths()

# Merged results
GWAS.QTL <- eQTL_Catalogue.query(sumstats_paths=sumstats_paths, qtl_search="Alasoo_2018", nThread=1, force_n
# Merged results (parallel)
GWAS.QTL <- eQTL_Catalogue.query(sumstats_paths=sumstats_paths, qtl_search="Alasoo_2018", nThread=4, force_n

# Split results
gwas.qtl <- eQTL_Catalogue.query(sumstats_paths=sumstats_paths, qtl_search="Alasoo_2018", nThread=1, force_n
GWAS.QTL <- gather_files(file_paths = gwas.qtl)
# Split results (parallel)
gwas.qtl <- eQTL_Catalogue.query(sumstats_paths=sumstats_paths, qtl_search="Alasoo_2018", nThread=4, force_n
GWAS.QTL <- gather_files(file_paths = gwas.qtl)
```

---

eQTL_Catalogue.search_metadata

Search eQTL Catalogue metadata

Description

Searches through multiple relevant metadata columns to find eQTL Catalogue datasets that match at least one of your substrings in a list. All searches are case-insensitive. If `qtl_search=NULL`, will return all available datasets.

Usage

```
eQTL_Catalogue.search_metadata(qtl_search = NULL, verbose = T)
```

See Also

Other eQTL Catalogue: `eQTL_Catalogue.fetch()`, `eQTL_Catalogue.iterate_fetch()`, `eQTL_Catalogue.query()`,
`fetch_restAPI()`, `fetch_tabix()`, `merge_gwas_qtl()`, `meta`

Examples

```
qtl_datasets <- eQTL_Catalogue.search_metadata(qtl_search=c("Alasoo_2018","monocyte"))
qtl_datasets.brain <- eQTL_Catalogue.search_metadata(qtl_search="brain")
```

`example_sumstats_paths` *Paths to example summary stats*

Description

Returns the paths to summary stats stored within *catalogueR*. Each file is the output of a locus that has been fine-mapping using *echolocatoR*. Data originally comes from the Parkinson's disease GWAS by Nalls et al. (bioRxiv).

Usage

```
example_sumstats_paths(Rlib_path = NULL)
```

Arguments

`Rlib_path`	This function will automatically find your Rlib path, but you can override this by supplying it manually.

Details

SNP SNP RSID

CHR Chromosome

POS Genomic positiion (in basepairs) ...

Source

https://www.biorxiv.org/content/10.1101/388165v3

Examples

```
sumstats_paths <- example_sumstats_paths()
```

`fetch_restAPI` *2. Query eQTL Catalogue datasets by region*

Description

2.2 Method 2: RESTful API Slower than tabix (unless you're only querying several specific SNPs).

Usage

```
fetch_restAPI(
  unique_id,
  quant_method = "ge",
  infer_region = T,
  gwas_data = NULL,
  chrom = NULL,
  bp_lower = NULL,
```

```
  bp_upper = NULL,
  is_gwas = F,
  size = NULL,
  verbose = T
)
```

Arguments

quant_method	eQTL Catalogue actually contains more than just eQTL data. For each dataset, the following kinds of QTLs can be queried: **gene expression QTL** `quant_method="ge"` (*default*) or `quant_method="microarray"`, depending on the dataset. **catalogueR** will automatically select whichever option is available. **exon expression QTL** **under construction** `quant_method="ex"` **transcript usage QTL** **under construction** `quant_method="tx"` **promoter, splice junction and 3' end usage QTL** **under construction** `quant_method="txrev"`
verbose	Show more (`=T`) or fewer (`=F`) messages.

See Also

Other eQTL Catalogue: `eQTL_Catalogue.fetch()`, `eQTL_Catalogue.iterate_fetch()`, `eQTL_Catalogue.query()`, `eQTL_Catalogue.search_metadata()`, `fetch_tabix()`, `merge_gwas_qtl()`, `meta`

Examples

```
data("meta"); data("BST1");
qtl.subset <- fetch_restAPI(unique_id=meta$unique_id[1], gwas_data=BST1)
```

`fetch_tabix` *2. Query eQTL Catalogue datasets by region*

Description

2.1 Method 1: Tabix Faster alternative to REST API.

Usage

```
fetch_tabix(
  unique_id,
  quant_method = "ge",
  infer_region = T,
  gwas_data = NULL,
  chrom = NULL,
  bp_lower = NULL,
  bp_upper = NULL,
  is_gwas = F,
  nThread = 4,
  verbose = T
)
```

Arguments

quant_method	eQTL Catalogue actually contains more than just eQTL data. For each dataset, the following kinds of QTLs can be queried: **gene expression QTL** `quant_method="ge"` (*default*) or `quant_method="microarray"`, depending on the dataset. **catalogueR** will automatically select whichever option is available. **exon expression QTL** **under construction** `quant_method="ex"` **transcript usage QTL** **under construction** `quant_method="tx"` **promoter, splice junction and 3' end usage QTL** **under construction** `quant_method="txrev"`
nThread	The number of CPU cores you want to use to speed up your queries through parallelization.
verbose	Show more (`=T`) or fewer (`=F`) messages.

See Also

Other eQTL Catalogue: `eQTL_Catalogue.fetch()`, `eQTL_Catalogue.iterate_fetch()`, `eQTL_Catalogue.query()`, `eQTL_Catalogue.search_metadata()`, `fetch_restAPI()`, `merge_gwas_qtl()`, `meta`

Examples

```
data("meta"); data("BST1");
qtl.subset <- fetch_tabix(unique_id=meta$unique_id[1], gwas_dat=BST1)
```

---

find_consensus_SNPs *Find Consensus SNPs in* echolocatoR *output*

Description

Find Consensus SNPs in *echolocatoR* output

Usage

```
find_consensus_SNPs(
  finemap_dat,
  verbose = T,
  credset_thresh = 0.95,
  consensus_thresh = 2,
  sort_by_support = T,
  exclude_methods = NULL
)
```

Examples

```
data("BST1")
BST1 <- find_consensus_SNPs(finemap_dat=BST1)
```

gather_files *Merge files from a list of paths*

Description

Merge a list of files into one by stacking them on top of each other (i.e. rbind).

Usage

```
gather_files(file_paths, nThread = 4, verbose = T)
```

See Also

Other utils: `ensembl_to_hgnc()`, `hgnc_to_ensembl()`, `liftover()`

Examples

```
sumstats_paths <- example_sumstats_paths()
merged_dat <- gather_files(file_paths=sumstats_paths)
```

get_colocs *Run coloc on GWAS-QTL object*

Description

Run coloc on GWAS-QTL object

Usage

```
get_colocs(
  qtl.egene,
  gwas.region,
  merge_by_rsid = T,
  PP_threshold = 0.8,
  verbose = T
)
```

See Also

Other coloc: `COLOC.report_summary()`, `run_coloc()`

hgnc_to_ensembl *Convert HGNC gene symbols to ENSEMBL IDs*

Description

Convert HGNC gene symbols to ENSEMBL IDs

Usage

```
hgnc_to_ensembl(gene_symbols, unique_only = T, verbose = T)
```

See Also

Other utils: `ensembl_to_hgnc()`, `gather_files()`, `liftover()`

Examples

```
gene_symbols <- c("BDNF","FOXP2","BST1")
ensembl_ids <- hgnc_to_ensembl(gene_symbols)
```

liftover *Lift genome across builds*

Description

Lift genome across builds

Usage

```
liftover(gwas_data, build.conversion = "hg19.to.hg38", verbose = T)
```

Arguments

`build_conversion`
: "hg19.to.hg38" (*default*) or "hg38.to.hg19.

See Also

Other utils: `ensembl_to_hgnc()`, `gather_files()`, `hgnc_to_ensembl()`

Examples

```
data("BST1")
gr.lifted <- liftover(gwas_data=BST1, build.conversion="hg19.to.hg38")
```

LRRK2 echolocatoR *output example (LRRK2 locus)*

Description

An example results file after running `finemap_loci` on the *LRRK2* locus.

Usage

```
LRRK2
```

Format

data.table

SNP SNP RSID

CHR Chromosome

POS Genomic positiion (in basepairs) ...

Details

Data originally comes from the Parkinson's disease GWAS by Nalls et al. (bioRxiv).

Source

https://www.biorxiv.org/content/10.1101/388165v3

See Also

Other Nalls23andMe_2019: `BST1`, `MEX3C`

Examples

```
## Not run:
root_dir <- "~/Desktop/Fine_Mapping/Data/GWAS/Nalls23andMe_2019"
locus_dir <- "LRRK2/Multi-finemap/Multi-finemap_results.txt"
LRRK2 <- data.table::fread(file.path(root_dir,locus_dir))
LRRK2 <- update_CS_cols(finemap_dat=LRRK2)
LRRK2 <- find_consensus_SNPs(finemap_dat=LRRK2)
data.table::fwrite(LRRK2,"inst/extdata/Nalls23andMe_2019/LRRK2_Nalls23andMe_2019_subset.tsv.gz", sep="\t")
usethis::use_data(LRRK2, overwrite = T)

## End(Not run)
```

merge_gwas_qtl *Merge GWAS data (query) and QTL data (results)*

Description

Merge GWAS data (query) and QTL data (results)

Usage

```
merge_gwas_qtl(gwas_data, qtl.subset, verbose = T)
```

Arguments

verbose	Show more (=T) or fewer (=F) messages.

See Also

Other eQTL Catalogue: eQTL_Catalogue.fetch(), eQTL_Catalogue.iterate_fetch(), eQTL_Catalogue.query(), eQTL_Catalogue.search_metadata(), fetch_restAPI(), fetch_tabix(), meta

meta *eQTL Catalogue dataset metadata*

Description

List of all queryable tabix-indexed eQTL Catalogue datasets and their associated systems/tissues/cell types.

Usage

```
meta
```

Format

An object of class data.table (inherits from data.frame) with 242 rows and 11 columns.

See Also

Other eQTL Catalogue: eQTL_Catalogue.fetch(), eQTL_Catalogue.iterate_fetch(), eQTL_Catalogue.query(), eQTL_Catalogue.search_metadata(), fetch_restAPI(), fetch_tabix(), merge_gwas_qtl()

Examples

```
## Not run:
meta <- eQTL_Catalogue.list_datasets(force_new=T)
meta <- meta %>% dplyr::mutate(ftp_path= gsub("Fairfax_2014_monocyte","Fairfax_2014",ftp_path))
usethis::use_data(meta, overwrite=T)

## End(Not run)
```

MEX3C echolocatoR *output example (MEX3C locus)*

Description

An example results file after running `finemap_loci` on the *MEX3C* locus.

Usage

```
MEX3C
```

Format

data.table

SNP SNP RSID

CHR Chromosome

POS Genomic positiion (in basepairs) ...

Details

Data originally comes from the Parkinson's disease GWAS by Nalls et al. (bioRxiv).

Source

https://www.biorxiv.org/content/10.1101/388165v3

See Also

Other Nalls23andMe_2019: `BST1`, `LRRK2`

Examples

```
## Not run:
root_dir <- "~/Desktop/Fine_Mapping/Data/GWAS/Nalls23andMe_2019"
locus_dir <- "MEX3C/Multi-finemap/Multi-finemap_results.txt"
MEX3C <- data.table::fread(file.path(root_dir,locus_dir))
MEX3C <- update_CS_cols(finemap_dat=MEX3C)
MEX3C <- find_consensus_SNPs(finemap_dat=MEX3C)
data.table::fwrite(MEX3C,"inst/extdata/Nalls23andMe_2019/MEX3C_Nalls23andMe_2019_subset.tsv.gz", sep="\t")
usethis::use_data(MEX3C, overwrite = T)

## End(Not run)
```

run_coloc *Iteratively run coloc on GWAS-QTL objects*

Description

Iteratively run coloc on GWAS-QTL objects

Usage

```
run_coloc(
  gwas.qtl_paths,
  save_path = "./coloc_results.tsv.gz",
  nThread = 3,
  top_snp_only = T,
  split_by_group = F
)
```

See Also

Other coloc: `COLOC.report_summary()`, `get_colocs()`

Index