Package 'echolocatoR'

October 31, 2022

Type Package

Title Automated genomic fine-mapping

Version 2.0.2

Description Automated statistical and functional fine-mapping with extensive access to genome-wide datasets.

License GPL-3

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

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

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

SystemRequirements Python (>= 3.7.0)

biocViews Genetics, FunctionalGenomics, SystemsBiology

Imports echoconda, downloadR, echodata, echotabix, echoannot, catalogueR, echoLD, echofinemap, echoplot, data.table, methods, stringr, lifecycle, cli, utf8, scales

Suggests MungeSumstats (>= 1.3.14), echodeps, BiocStyle, remotes, rmarkdown, knitr, testthat (>= 3.0.0), badger

Remotes github::RajLabMSSM/echoconda,
github::RajLabMSSM/downloadR,
github::RajLabMSSM/echodata,
github::RajLabMSSM/echotabix,
github::RajLabMSSM/echoannot,
github::RajLabMSSM/catalogueR,
github::RajLabMSSM/echoLD,
github::RajLabMSSM/echofinemap,
github::RajLabMSSM/echoplot,
github::RajLabMSSM/echodeps

RoxygenNote 7.2.1.9000

VignetteBuilder knitr

Config/testthat/edition 3

R topics documented:

`batapply` **echolocatoR**-*themed progress bar*

Description

Iterator function with **echolocatoR**-themed progress bar.

Usage

```
batapply(
  X,
  FUN = function(l) Sys.sleep(5/100),
  apply_func = lapply,
  total = length(X),
  name = NULL,
  .envir = parent.frame(),
  cli.progress_show_after = 0,
  clear = FALSE,
  color1 = cli::col_br_cyan,
```

```
  color2 = cli::col_br_magenta,
  ...
)
```

Arguments

X	a vector (atomic or list) or an expression object. Other objects (including classed objects) will be coerced by base::as.list.
FUN	the function to be applied to each element of X: see 'Details'. In the case of functions like +, %*%, the function name must be backquoted or quoted.
apply_func	Iterator function to use (default: lapply).
total	Passed to cli_progress_bar().
name	Name of the progress bar, a label, passed to cli_progress_bar().
.envir	Passed to cli_progress_bar().
cli.progress_show_after	How long to wait before showing the progress bar.
clear	Whether to remove the progress bar from the screen after it has terminated. Defaults to the cli.progress_clear option, or TRUE if unset.
color1	First color to use in the palette.
color2	Second color to use in the palette.
...	Additional arguments passed to apply_func.

Value

A (named) list.

Examples

```
out <- batapply(X = seq_len(30))
```

---

catalogueR *catalogueR*

Description

catalogueR

check_deprecated *Check deprecated arguments*

Description

Semi-automatically check all deprecated args in a given function.

Usage

```
check_deprecated(
  fun = "finemap_loci",
  pkg = "echolocatoR",
  when = "2.0.0",
  args = match.call(),
  lifecycle_fun = lifecycle::deprecate_warn,
  reassign = FALSE,
 map = list(A1_col = "colmap", A2_col = "colmap", chrom_col = "colmap", position_col =
  "colmap", effect_col = "colmap", freq_col = "colmap", gene_col = "colmap", locus_col
   = "colmap", MAF_col = "colmap", N_cases = "colmap", N_controls = "colmap",
  N_cases_col = "colmap", N_controls_col = "colmap", sample_size = "colmap", MAF_col =
  "colmap", pval_col = "colmap", stderr_col = "colmap", tstat_col = "colmap", snp_col =
  "colmap", file_sep = NULL, probe_path = NULL, chrom_type = NULL, PAINTOR_QTL_datasets
   = NULL,
    QTL_prefixes = "qtl_suffixes", proportion_cases = NULL, server = NULL,
   vcf_folder = NULL, top_SNPs = "topSNPs", PP_threshold = "credset_thresh",
  consensus_threshold = "consensus_thresh", plot.types = "plot_types", plot.Roadmap =
  "roadmap", plot.Roadmap_query = "roadmap_query", plot.XGR_libnames = "xgr_libnames",
  plot.zoom = "zoom", plot.zoom = "zoom", plot.Nott_epigenome = "nott_epigenome",
   plot.Nott_show_placseq = "nott_show_placseq")
)
```

Arguments

fun	Function to check.
pkg	Package that the function is from.
when	A string giving the version when the behaviour was deprecated.
args	Argument calls to assess.
lifecycle_fun	Which **lifecycle** function to use by default.
reassign	Attempt to reassign deprecated variables to the corresponding new variable (if applicable).
map	Mapping between old:new argument names. Use NULL if the argument is no longer used at all.

Examples

```
topSNPs <- echodata::topSNPs_Nalls2019
fullSS_path <- echodata::example_fullSS()
testthat::expect_error(
  echolocatoR::finemap_loci(
    fullSS_path = fullSS_path,
```

```
    topSNPs = topSNPs,
    loci = c("BST1","MEX3C"),
    chrom_col = "CHR",
    position_col = "BP")
)
```

---

check_genome *Check genome build*

Description

If `fullSS_genome_build==NULL` and `munged=TRUE`, infers genome build (hg19 vs. hg38) from summary statistics using get_genome_builds. This can only be done with summary statistics that have already been munged by format_sumstats. When `fullSS_genome_build` is a synonym of hg19 or hg38, this function simply returns a standardized version of the user-provided genome build.

Usage

```
check_genome(
  fullSS_genome_build = NULL,
  munged = FALSE,
  fullSS_path = NULL,
  sampled_snps = 10000,
  names_from_paths = TRUE,
  dbSNP = 144,
  nThread = 1,
  verbose = TRUE
)
```

Arguments

`fullSS_genome_build`	Genome build of the full summary statistics (`fullSS_path`). Can be "GRCH37" or "GRCH38" or one of their synonyms.. If `fullSS_genome_build==NULL` and `munged=TRUE`, infers genome build (hg19 vs. hg38) from summary statistics using get_genome_builds.
`munged`	Whether `fullSS_path` have already been standardised/filtered full summary stats with format_sumstats. If `munged=FALSE` you'll need to provide the necessary column names to the `colmap` argument.
`fullSS_path`	Path to the full summary statistics file (GWAS or QTL) that you want to fine-map. It is usually best to provide the absolute path rather than the relative path.
`sampled_snps`	Downsample the number of SNPs used when inferring genome build to save time.
`names_from_paths`	Infer the name of each item in `sumstats_list` from its respective file path. Only works if `sumstats_list` is a list of paths.
`dbSNP`	version of dbSNP to be used (144 or 155). Default is 155.
`nThread`	Number of threads to parallelise saving across.
`verbose`	Print messages.

Value

Character string indicating genome build.

Examples

```
fullSS_path <- echodata::example_fullSS()
build <- check_genome(fullSS_genome_build="hg19",
                      fullSS_path=fullSS_path)
```

`deprecated` *deprecated*

Description

deprecated

`downloadR` *downloadR*

Description

downloadR

`echoannot` *echoannot*

Description

echoannot

`echoconda` *echoconda*

Description

echoconda

`echodata` *echodata*

Description

echodata

finemap_loci *Fine-map multiple loci*

Description

echolocatoR will automatically fine-map each locus. Uses the topSNPs data.frame to define locus coordinates.

Usage

```
finemap_loci(
  loci = NULL,
  fullSS_path,
  fullSS_genome_build = NULL,
  results_dir = file.path(tempdir(), "results"),
  dataset_name = "dataset_name",
  dataset_type = "GWAS",
  topSNPs = "auto",
  force_new_subset = FALSE,
  force_new_LD = FALSE,
  force_new_finemap = FALSE,
  finemap_methods = c("ABF", "FINEMAP", "SUSIE"),
  finemap_args = NULL,
  n_causal = 5,
  credset_thresh = 0.95,
  consensus_thresh = 2,
  fillNA = 0,
  conditioned_snps = "auto",
  priors_col = NULL,
  munged = FALSE,
  colmap = echodata::construct_colmap(munged = munged),
  compute_n = "ldsc",
  LD_reference = "1KGphase3",
  LD_genome_build = "hg19",
  leadSNP_LD_block = FALSE,
  superpopulation = "EUR",
  download_method = "axel",
  bp_distance = 5e+05,
  min_POS = NA,
  max_POS = NA,
  min_MAF = NA,
  trim_gene_limits = FALSE,
  max_snps = NULL,
  min_r2 = 0,
  remove_variants = FALSE,
  remove_correlates = FALSE,
  query_by = "tabix",
  case_control = TRUE,
  qtl_suffixes = NULL,
  plot_types = c("simple"),
  show_plot = TRUE,
```

```
zoom = "1x",
tx_biotypes = NULL,
nott_epigenome = FALSE,
nott_show_placseq = FALSE,
nott_binwidth = 200,
nott_bigwig_dir = NULL,
xgr_libnames = NULL,
roadmap = FALSE,
roadmap_query = NULL,
remove_tmps = TRUE,
conda_env = "echoR_mini",
return_all = TRUE,
use_tryCatch = TRUE,
seed = 2022,
nThread = 1,
verbose = TRUE,
top_SNPs = deprecated(),
PP_threshold = deprecated(),
consensus_threshold = deprecated(),
plot.Nott_epigenome = deprecated(),
plot.Nott_show_placseq = deprecated(),
plot.Nott_binwidth = deprecated(),
plot.Nott_bigwig_dir = deprecated(),
plot.Roadmap = deprecated(),
plot.Roadmap_query = deprecated(),
plot.XGR_libnames = deprecated(),
server = deprecated(),
plot.types = deprecated(),
plot.zoom = deprecated(),
QTL_prefixes = deprecated(),
vcf_folder = deprecated(),
probe_path = deprecated(),
file_sep = deprecated(),
chrom_col = deprecated(),
chrom_type = deprecated(),
position_col = deprecated(),
snp_col = deprecated(),
pval_col = deprecated(),
effect_col = deprecated(),
stderr_col = deprecated(),
tstat_col = deprecated(),
locus_col = deprecated(),
freq_col = deprecated(),
MAF_col = deprecated(),
A1_col = deprecated(),
A2_col = deprecated(),
gene_col = deprecated(),
N_cases_col = deprecated(),
N_controls_col = deprecated(),
N_cases = deprecated(),
N_controls = deprecated(),
proportion_cases = deprecated(),
```

```
  sample_size = deprecated(),
  PAINTOR_QTL_datasets = deprecated()
)
```

Arguments

`loci`	Character list of loci in **Locus** col of `topSNPs`.
`fullSS_path`	Path to the full summary statistics file (GWAS or QTL) that you want to fine-map. It is usually best to provide the absolute path rather than the relative path.
`fullSS_genome_build`	Genome build of the full summary statistics (`fullSS_path`). Can be "GRCH37" or "GRCH38" or one of their synonyms.. If `fullSS_genome_build==NULL` and `munged=TRUE`, infers genome build (hg19 vs. hg38) from summary statistics using get_genome_builds.
`results_dir`	Where to store all results. **IMPORTANT!:** It is usually best to provide the absolute path rather than the relative path. This is especially important for *FINEMAP*.
`dataset_name`	The name you want to assign to the dataset being fine-mapped, This will be used to name the subdirectory where your results will be stored (e.g. *Data/GWAS/<dataset_name>*). Don't use special characters (e.g.".", "/").
`dataset_type`	The kind dataset you're fine-mapping (e.g. GWAS, eQTL, tQTL). This will also be used when creating the subdirectory where your results will be stored (e.g. *Data/<dataset_type>/Kunkle_2019*).
`topSNPs`	A data.frame with the genomic coordinates of the lead SNP for each locus. The lead SNP will be used as the center of the window when extracting subset from the full GWAS/QTL summary statistics file. Only one SNP per **Locus** should be included. At minimum, `topSNPs` should include the following columns: ***Locus*** A unique name for each locus. Often, loci are named after a relevant gene (e.g. LRRK2) or based on the name/coordinates of the lead SNP (e.g. locus_chr12_40734202) ***CHR*** The chromosome that the SNP is on. Can be "chr12" or "12" format. ***POS*** The genomic position of the SNP (in basepairs)
`force_new_subset`	By default, if a subset of the full summary stats file for a given locus is already present, then **echolocatoR** will just use the pre-existing file. Set `force_new_subset=T` to override this and extract a new subset. Subsets are saved in the following path structure: *Data/\<dataset_type\>/\<dataset_name\>/\<locus\>/Multi-finemap/\<locus\>_\<dataset_name\>_Multi-finemap.tsv.gz*
`force_new_LD`	Force new LD subset.
`force_new_finemap`	By default, if an fine-mapping results file for a given locus is already present, then **echolocatoR** will just use the preexisting file. Set `force_new_finemap=T` to override this and re-run fine-mapping.
`finemap_methods`	Which fine-mapping methods you want to use.
`finemap_args`	A named nested list containing additional arguments for each fine-mapping method. e.g. `finemap_args = list(FINEMAP=list(), PAINTOR=list(method=""))`
`n_causal`	The maximum number of potential causal SNPs per locus. This parameter is used somewhat differently by different fine-mapping tools. See tool-specific functions for details.

`credset_thresh` The minimum fine-mapped posterior probability for a SNP to be considered part of a Credible Set. For example, `credset_thresh=.95` means that all Credible Set SNPs will be 95% Credible Set SNPs.

`consensus_thresh`
The minimum number of fine-mapping tools in which a SNP is in the Credible Set in order to be included in the "Consensus_SNP" column.

`fillNA` Value to fill LD matrix NAs with.

`conditioned_snps`
Which SNPs to conditions on when fine-mapping with (e.g. *COJO*).

`priors_col` [Optional] Name of the a column in `dat` to extract SNP-wise prior probabilities from.

`munged` Whether `fullSS_path` have already been standardised/filtered full summary stats with format_sumstats. If `munged=FALSE` you'll need to provide the necessary column names to the `colmap` argument.

`colmap` Column name mappings in in `fullSS_path`. Must be a named list. Can use construct_colmap to assist with this. This function can be used in two different ways:

- `munged=FALSE` : When `munged=FALSE`, you will need to provide the necessary column names to the `colmap` argument (*default*).
- `munged=TRUE` : Alternatively, instead of filling out each argument in construct_colmap, you can simply set `munged=TRUE` if `fullSS_path` has already been munged with format_sumstats.

`compute_n` How to compute per-SNP sample size (new column "N").
If the column "N" is already present in `dat`, this column will be used to extract per-SNP sample sizes and the argument `compute_n` will be ignored.
If the column "N" is *not* present in `dat`, one of the following options can be supplied to `compute_n`:

- `0`: N will not be computed.
- `>0`: If any number >0 is provided, that value will be set as N for every row. **Note**: Computing N this way is incorrect and should be avoided if at all possible.
- `"sum"`: N will be computed as: cases (N_CAS) + controls (N_CON), so long as both columns are present.
- `"ldsc"`: N will be computed as effective sample size: Neff =(N_CAS+N_CON)*(N_CAS/(N_CA / mean((N_CAS/(N_CAS+N_CON))(N_CAS+N_CON)==max(N_CAS+N_CON)).
- `"giant"`: N will be computed as effective sample size: Neff = 2 / (1/N_CAS + 1/N_CON).
- `"metal"`: N will be computed as effective sample size: Neff = 4 / (1/N_CAS + 1/N_CON).

`LD_reference` LD reference to use:

- "1KGphase1" : 1000 Genomes Project Phase 1 (genome build: hg19).
- "1KGphase3" : 1000 Genomes Project Phase 3 (genome build: hg19).
- "UKB" : Pre-computed LD from a British European-decent subset of UK Biobank. *Genome build* : hg19
- "<vcf_path>" : User-supplied path to a custom VCF file to compute LD matrix from.
  Accepted formats: *.vcf* / *.vcf.gz* / *.vcf.bgz*
  Genome build : defined by user with `target_genome`.

- "<matrix_path>" : User-supplied path to a pre-computed LD matrix *Accepted formats*: *.rds* / *.rda* / *.csv* / *.tsv* / *.txt*
  Genome build : defined by user with `target_genome`.

`LD_genome_build`
: Genome build of the LD panel. This is automatically assigned to the correct genome build for each LD panel except when the user supplies custom vcf/LD files.

`leadSNP_LD_block`
: Only return SNPs within the same LD block as the lead SNP (the SNP with the smallest p-value).

`superpopulation`
: Superpopulation to subset LD panel by (used only if `LD_reference` is "1KGphase1" or "1KGphase3"). See popDat_1KGphase1 and popDat_1KGphase3 for full tables of their respective samples.

`download_method`
- `"axel"` : Multi-threaded
- `"wget"` : Single-threaded
- `"download.file"` : Single-threaded
- `"internal"` : Single-threaded (passed to download.file)
- `"wininet"` : Single-threaded (passed to download.file)
- `"libcurl"` : Single-threaded (passed to download.file)
- `"curl"` : Single-threaded (passed to download.file)

`bp_distance`
: Distance around the lead SNP to include.

`min_POS`
: Minimum genomic position to include.

`max_POS`
: Maximum genomic position to include.

`min_MAF`
: Minimum Minor Allele Frequency (MAF) of SNPs to include.

`trim_gene_limits`
: If a gene name is supplied to this argument (e.g. `trim_gene_limits="BST"`), only SNPs within the gene body will be included.

`max_snps`
: Maximum number of SNPs to include.

`min_r2`
: Correlation threshold for `remove_correlates`.

`remove_variants`
: A list of SNP RSIDs to remove.

`remove_correlates`
: A list of SNPs. If provided, all SNPs that correlates with these SNPs (at r2>=`min_r2`) will be removed from both `dat` and `LD` list items..

`query_by`
: Choose which method you want to use to extract locus subsets from the full summary stats file. Methods include:

  "tabix" Convert the full summary stats file in an indexed tabix file. Makes querying lightning fast after the initial conversion is done. (*default*)

  "coordinates" Extract locus subsets using min/max genomic coordinates with *awk*.

`case_control`
: Whether the summary statistics come from a case-control study (e.g. a GWAS of having Alzheimer's Disease or not) (`TRUE`) or a quantitative study (e.g. a GWAS of height, or an eQTL) (`FALSE`).

`qtl_suffixes`
: If columns with QTL data is included in `dat`, you can indicate which columns those are with one or more string suffixes (e.g. `qtl_suffixes=c(".eQTL1",".eQTL2")` to use the columns "P.QTL1", "Effect.QTL1", "P.QTL2", "Effect.QTL2").

`plot_types` Which kinds of plots to include. Options:

- "simple"Just plot the following tracks: GWAS, fine-mapping, gene models
- "fancy"Additionally plot XGR annotation tracks (XGR, Roadmap, Nott2019). '
- "LD"LD heatmap showing the 10 SNPs surrounding the lead SNP.

`show_plot` Print plot to screen.

`zoom` Zoom into the center of the locus when plotting (without editing the fine-mapping results file). You can provide either:

- The size of your plot window in terms of basepairs (e.g. `zoom=50000` for a 50kb window).
- How much you want to zoom in (e.g. `zoom="1x"` for the full locus, `zoom="2x"` for 2x zoom into the center of the locus, etc.).

You can pass a list of window sizes (e.g. `c(50000,100000,500000)`) to automatically generate multiple views of each locus. This can even be a mix of different style inputs: e.g. `c("1x","4.5x",25000)`.

`tx_biotypes` Transcript biotypes to include in the gene model track. By default (`NULL`), all transcript biotypes will be included. See get_tx_biotypes for a full list of all available biotypes

`nott_epigenome` Include tracks showing brain cell-type-specific epigenomic data from Nott et al. (2019).

`nott_show_placseq` Include track generated by NOTT2019_plac_seq_plot.

`nott_binwidth` When including Nott et al. (2019) epigenomic data in the track plots, adjust the bin width of the histograms.

`nott_bigwig_dir` Instead of pulling Nott et al. (2019) epigenomic data from the *UCSC Genome Browser*, use a set of local bigwig files.

`xgr_libnames` Passed to XGR_plot. Which XGR annotations to check overlap with. For full list of libraries see here. Passed to the `RData.customised` argument in xRDataLoader.

`roadmap` Find and plot annotations from Roadmap.

`roadmap_query` Only plot annotations from Roadmap whose metadata contains a string or any items from a list of strings (e.g. `"brain"` or `c("brain","liver","monocytes")`).

`remove_tmps` Whether to remove any temporary files (e.g. FINEMAP output files) after the pipeline is done running.

`conda_env` Conda environment to use.

`return_all` Return a nested list of various the pipeline's outputs including plots, tables, and file paths (default: `TRUE`). If `FALSE`, instead only returns a single merged data.table containing the results from all loci.

`use_tryCatch` If an error is encountered in one locus, the pipeline will continue to try running the rest of the loci (default: `use_tryCatch=TRUE`). This avoid stopping all analyses due to errors that only affect some loci, but currently prevents debugging via traceback.

`seed` Set the seed for all functions where this is possible.

`nThread` Number of threads to parallelise saving across.

`verbose` Print messages.

top_SNPs [deprecated]

PP_threshold [deprecated]

consensus_threshold [deprecated]

plot.Nott_epigenome [deprecated]

plot.Nott_show_placseq [deprecated]

plot.Nott_binwidth [deprecated]

plot.Nott_bigwig_dir [deprecated]

plot.Roadmap [deprecated]

plot.Roadmap_query [deprecated]

plot.XGR_libnames [deprecated]

server [deprecated]

plot.types [deprecated]

plot.zoom [deprecated]

QTL_prefixes [deprecated]

vcf_folder [deprecated]

probe_path [deprecated]

file_sep [deprecated]

chrom_col [deprecated]

chrom_type [deprecated]

position_col [deprecated]

snp_col [deprecated]

pval_col [deprecated]

effect_col [deprecated]

stderr_col [deprecated]

tstat_col [deprecated]

locus_col [deprecated]

freq_col [deprecated]

MAF_col [deprecated]

A1_col [deprecated]

A2_col [deprecated]

gene_col [deprecated]

N_cases_col [deprecated]

N_controls_col [deprecated]

N_cases [deprecated]

N_controls [deprecated]

proportion_cases [deprecated]

sample_size [deprecated]

PAINTOR_QTL_datasets [deprecated]

Value

By default, returns a nested list containing grouped by locus names (e.g. `BST1`, `MEX3C`). The results of each locus contain the following elements:

- `finemap_dat` : Fine-mapping results from all selected methods merged with the original summary statistics (i.e. **Multi-finemap results**).
- `locus_plot` : A nested list containing one or more zoomed views of locus plots.
- `LD_matrix` : The post-processed LD matrix used for fine-mapping.
- `LD_plot` : An LD plot (if used).
- `locus_dir` : Locus directory results are saved in.
- `arguments` : A record of the arguments supplied to finemap_loci.

In addition, the following object summarizes the results from all the locus-specific results:

- `merged_dat` : A merged data.table with all fine-mapping results from all loci.

See Also

Other MAIN: `finemap_locus()`

Examples

```
topSNPs <- echodata::topSNPs_Nalls2019
fullSS_path <- echodata::example_fullSS(dataset = "Nalls2019")

res <- echolocatoR::finemap_loci(
  fullSS_path = fullSS_path,
  topSNPs = topSNPs,
  loci = c("BST1","MEX3C"),
  finemap_methods = c("ABF","FINEMAP","SUSIE"),
  dataset_name = "Nalls23andMe_2019",
  fullSS_genome_build = "hg19",
  bp_distance = 1000,
  munged = TRUE)
```

`finemap_locus` *Run* **echolocatoR** *pipeline on a single locus*

Description

Unlike `finemap_loci`, you don't need to provide a `topSNPs` data.frame. Instead, just manually provide the coordinates of the locus you want to fine-map.

Usage

```
finemap_locus(
  locus,
  fullSS_path,
  fullSS_genome_build = NULL,
  results_dir = file.path(tempdir(), "results"),
  dataset_name = "dataset_name",
```

```
    dataset_type = "GWAS",
    case_control = TRUE,
    topSNPs = "auto",
    force_new_subset = FALSE,
    force_new_LD = FALSE,
    force_new_finemap = FALSE,
    finemap_methods = c("ABF", "FINEMAP", "SUSIE"),
    finemap_args = NULL,
    n_causal = 5,
    credset_thresh = 0.95,
    consensus_thresh = 2,
    fillNA = 0,
    conditioned_snps = NULL,
    priors_col = NULL,
    munged = FALSE,
    colmap = echodata::construct_colmap(munged = munged),
    compute_n = "ldsc",
    LD_reference = "1KGphase3",
    LD_genome_build = "hg19",
    leadSNP_LD_block = FALSE,
    superpopulation = "EUR",
    download_method = "axel",
    bp_distance = 5e+05,
    min_POS = NA,
    max_POS = NA,
    min_MAF = NA,
    trim_gene_limits = FALSE,
    max_snps = NULL,
    min_r2 = 0,
    remove_variants = FALSE,
    remove_correlates = FALSE,
    query_by = "tabix",
    qtl_suffixes = NULL,
    plot_types = c("simple"),
    zoom = "1x",
    show_plot = TRUE,
    tx_biotypes = NULL,
    nott_epigenome = FALSE,
    nott_show_placseq = FALSE,
    nott_binwidth = 200,
    nott_bigwig_dir = NULL,
    xgr_libnames = NULL,
    roadmap = FALSE,
    roadmap_query = NULL,
    remove_tmps = TRUE,
    seed = 2022,
    conda_env = "echoR_mini",
    nThread = 1,
    verbose = TRUE,
    top_SNPs = deprecated(),
    PP_threshold = deprecated(),
    consensus_threshold = deprecated(),
```

```
  plot.Nott_epigenome = deprecated(),
  plot.Nott_show_placseq = deprecated(),
  plot.Nott_binwidth = deprecated(),
  plot.Nott_bigwig_dir = deprecated(),
  plot.Roadmap = deprecated(),
  plot.Roadmap_query = deprecated(),
  plot.XGR_libnames = deprecated(),
  server = deprecated(),
  plot.types = deprecated(),
  plot.zoom = deprecated(),
  QTL_prefixes = deprecated(),
  vcf_folder = deprecated(),
  probe_path = deprecated(),
  file_sep = deprecated,
  chrom_col = deprecated(),
  chrom_type = deprecated(),
  position_col = deprecated(),
  snp_col = deprecated(),
  pval_col = deprecated(),
  effect_col = deprecated(),
  stderr_col = deprecated(),
  tstat_col = deprecated(),
  locus_col = deprecated(),
  freq_col = deprecated(),
  MAF_col = deprecated(),
  A1_col = deprecated(),
  A2_col = deprecated(),
  gene_col = deprecated(),
  N_cases_col = deprecated(),
  N_controls_col = deprecated(),
  N_cases = deprecated(),
  N_controls = deprecated(),
  proportion_cases = deprecated(),
  sample_size = deprecated(),
  PAINTOR_QTL_datasets = deprecated()
)
```

Arguments

`locus`	Locus name to fine-map (e.g. `"BIN1"`). Can be named to indicate a specific gene within a QTL locus (e.g. `c(ENSG00000136731="BIN1")`).
`fullSS_path`	Path to the full summary statistics file (GWAS or QTL) that you want to fine-map. It is usually best to provide the absolute path rather than the relative path.
`fullSS_genome_build`	Genome build of the full summary statistics (`fullSS_path`). Can be "GRCH37" or "GRCH38" or one of their synonyms.. If `fullSS_genome_build==NULL` and `munged=TRUE`, infers genome build (hg19 vs. hg38) from summary statistics using get_genome_builds.
`results_dir`	Where to store all results. **IMPORTANT!:** It is usually best to provide the absolute path rather than the relative path. This is especially important for *FINEMAP*.

`dataset_name` The name you want to assign to the dataset being fine-mapped, This will be used to name the subdirectory where your results will be stored (e.g. *Data/GWAS/<dataset_name>*). Don't use special characters (e.g.".", "/").

`dataset_type` The kind dataset you're fine-mapping (e.g. GWAS, eQTL, tQTL). This will also be used when creating the subdirectory where your results will be stored (e.g. *Data/<dataset_type>/Kunkle_2019*).

`case_control` Whether the summary statistics come from a case-control study (e.g. a GWAS of having Alzheimer's Disease or not) (`TRUE`) or a quantitative study (e.g. a GWAS of height, or an eQTL) (`FALSE`).

`topSNPs` A data.frame with the genomic coordinates of the lead SNP for each locus. The lead SNP will be used as the center of the window when extracting subset from the full GWAS/QTL summary statistics file. Only one SNP per **Locus** should be included. At minimum, `topSNPs` should include the following columns:

- ***Locus*** A unique name for each locus. Often, loci are named after a relevant gene (e.g. LRRK2) or based on the name/coordinates of the lead SNP (e.g. locus_chr12_40734202)
- ***CHR*** The chromosome that the SNP is on. Can be "chr12" or "12" format.
- ***POS*** The genomic position of the SNP (in basepairs)

`force_new_subset`
By default, if a subset of the full summary stats file for a given locus is already present, then **echolocatoR** will just use the pre-existing file. Set `force_new_subset=T` to override this and extract a new subset. Subsets are saved in the following path structure: *Data/\<dataset_type\>/\<dataset_name\>/\<locus\>/Multi-finemap/\<locus\>_\<dataset_name\>_Multi-finemap.tsv.gz*

`force_new_LD` Force new LD subset.

`force_new_finemap`
By default, if an fine-mapping results file for a given locus is already present, then **echolocatoR** will just use the preexisting file. Set `force_new_finemap=T` to override this and re-run fine-mapping.

`finemap_methods`
Which fine-mapping methods you want to use.

`finemap_args` A named nested list containing additional arguments for each fine-mapping method. e.g. `finemap_args = list(FINEMAP=list(), PAINTOR=list(method=""))`

`n_causal` The maximum number of potential causal SNPs per locus. This parameter is used somewhat differently by different fine-mapping tools. See tool-specific functions for details.

`credset_thresh` The minimum fine-mapped posterior probability for a SNP to be considered part of a Credible Set. For example, `credset_thresh=.95` means that all Credible Set SNPs will be 95% Credible Set SNPs.

`consensus_thresh`
The minimum number of fine-mapping tools in which a SNP is in the Credible Set in order to be included in the "Consensus_SNP" column.

`fillNA` Value to fill LD matrix NAs with.

`conditioned_snps`
Which SNPs to conditions on when fine-mapping with (e.g. *COJO*).

`priors_col` [Optional] Name of the a column in `dat` to extract SNP-wise prior probabilities from.

`munged` Whether `fullSS_path` have already been standardised/filtered full summary stats with format_sumstats. If `munged=FALSE` you'll need to provide the necessary column names to the `colmap` argument.

`colmap` Column name mappings in in `fullSS_path`. Must be a named list. Can use construct_colmap to assist with this. This function can be used in two different ways:

- `munged=FALSE` : When `munged=FALSE`, you will need to provide the necessary column names to the `colmap` argument (*default*).
- `munged=TRUE` : Alternatively, instead of filling out each argument in construct_colmap, you can simply set `munged=TRUE` if `fullSS_path` has already been munged with format_sumstats.

`compute_n` How to compute per-SNP sample size (new column "N").
If the column "N" is already present in `dat`, this column will be used to extract per-SNP sample sizes and the argument `compute_n` will be ignored.
If the column "N" is *not* present in `dat`, one of the following options can be supplied to `compute_n`:

- `0`: N will not be computed.
- `>0`: If any number >0 is provided, that value will be set as N for every row. **Note**: Computing N this way is incorrect and should be avoided if at all possible.
- `"sum"`: N will be computed as: cases (N_CAS) + controls (N_CON), so long as both columns are present.
- `"ldsc"`: N will be computed as effective sample size: Neff =(N_CAS+N_CON)*(N_CAS/(N_CA / mean((N_CAS/(N_CAS+N_CON))(N_CAS+N_CON)==max(N_CAS+N_CON)).
- `"giant"`: N will be computed as effective sample size: Neff = 2 / (1/N_CAS + 1/N_CON).
- `"metal"`: N will be computed as effective sample size: Neff = 4 / (1/N_CAS + 1/N_CON).

`LD_reference` LD reference to use:

- "1KGphase1" : 1000 Genomes Project Phase 1 (genome build: hg19).
- "1KGphase3" : 1000 Genomes Project Phase 3 (genome build: hg19).
- "UKB" : Pre-computed LD from a British European-decent subset of UK Biobank. *Genome build* : hg19
- "<vcf_path>" : User-supplied path to a custom VCF file to compute LD matrix from.
  Accepted formats: *.vcf* / *.vcf.gz* / *.vcf.bgz*
  Genome build : defined by user with `target_genome`.
- "<matrix_path>" : User-supplied path to a pre-computed LD matrix *Accepted formats*: *.rds* / *.rda* / *.csv* / *.tsv* / *.txt*
  Genome build : defined by user with `target_genome`.

`LD_genome_build`
Genome build of the LD panel. This is automatically assigned to the correct genome build for each LD panel except when the user supplies custom vcf/LD files.

`leadSNP_LD_block`
Only return SNPs within the same LD block as the lead SNP (the SNP with the smallest p-value).

`superpopulation`
: Superpopulation to subset LD panel by (used only if `LD_reference` is "1KGphase1" or "1KGphase3"). See popDat_1KGphase1 and popDat_1KGphase3 for full tables of their respective samples.

`download_method`
: - `"axel"` : Multi-threaded
  - `"wget"` : Single-threaded
  - `"download.file"` : Single-threaded
  - `"internal"` : Single-threaded (passed to download.file)
  - `"wininet"` : Single-threaded (passed to download.file)
  - `"libcurl"` : Single-threaded (passed to download.file)
  - `"curl"` : Single-threaded (passed to download.file)

`bp_distance`
: Distance around the lead SNP to include.

`min_POS`
: Minimum genomic position to include.

`max_POS`
: Maximum genomic position to include.

`min_MAF`
: Minimum Minor Allele Frequency (MAF) of SNPs to include.

`trim_gene_limits`
: If a gene name is supplied to this argument (e.g. `trim_gene_limits="BST"`), only SNPs within the gene body will be included.

`max_snps`
: Maximum number of SNPs to include.

`min_r2`
: Correlation threshold for `remove_correlates`.

`remove_variants`
: A list of SNP RSIDs to remove.

`remove_correlates`
: A list of SNPs. If provided, all SNPs that correlates with these SNPs (at r2>=`min_r2`) will be removed from both `dat` and `LD` list items..

`query_by`
: Choose which method you want to use to extract locus subsets from the full summary stats file. Methods include:

  "tabix" Convert the full summary stats file in an indexed tabix file. Makes querying lightning fast after the initial conversion is done. (*default*)

  "coordinates" Extract locus subsets using min/max genomic coordinates with *awk*.

`qtl_suffixes`
: If columns with QTL data is included in `dat`, you can indicate which columns those are with one or more string suffixes (e.g. `qtl_suffixes=c(".eQTL1",".eQTL2")` to use the columns "P.QTL1", "Effect.QTL1", "P.QTL2", "Effect.QTL2").

`plot_types`
: Which kinds of plots to include. Options:

  - "simple"Just plot the following tracks: GWAS, fine-mapping, gene models
  - "fancy"Additionally plot XGR annotation tracks (XGR, Roadmap, Nott2019). '
  - "LD"LD heatmap showing the 10 SNPs surrounding the lead SNP.

`zoom`
: Zoom into the center of the locus when plotting (without editing the fine-mapping results file). You can provide either:

  - The size of your plot window in terms of basepairs (e.g. `zoom=50000` for a 50kb window).
  - How much you want to zoom in (e.g. `zoom="1x"` for the full locus, `zoom="2x"` for 2x zoom into the center of the locus, etc.).

You can pass a list of window sizes (e.g. `c(50000,100000,500000)`) to automatically generate multiple views of each locus. This can even be a mix of different style inputs: e.g. `c("1x","4.5x",25000)`.

`show_plot` Print plot to screen.

`tx_biotypes` Transcript biotypes to include in the gene model track. By default (`NULL`), all transcript biotypes will be included. See get_tx_biotypes for a full list of all available biotypes

`nott_epigenome` Include tracks showing brain cell-type-specific epigenomic data from Nott et al. (2019).

`nott_show_placseq`
Include track generated by NOTT2019_plac_seq_plot.

`nott_binwidth` When including Nott et al. (2019) epigenomic data in the track plots, adjust the bin width of the histograms.

`nott_bigwig_dir`
Instead of pulling Nott et al. (2019) epigenomic data from the *UCSC Genome Browser*, use a set of local bigwig files.

`xgr_libnames` Passed to XGR_plot. Which XGR annotations to check overlap with. For full list of libraries see here. Passed to the `RData.customised` argument in xRDataLoader.

`roadmap` Find and plot annotations from Roadmap.

`roadmap_query` Only plot annotations from Roadmap whose metadata contains a string or any items from a list of strings (e.g. `"brain"` or `c("brain","liver","monocytes")`).

`remove_tmps` Whether to remove any temporary files (e.g. FINEMAP output files) after the pipeline is done running.

`seed` Set the seed for all functions where this is possible.

`conda_env` Conda environment to use.

`nThread` Number of threads to parallelise saving across.

`verbose` Print messages.

`top_SNPs` [deprecated]

`PP_threshold` [deprecated]

`consensus_threshold`
[deprecated]

`plot.Nott_epigenome`
[deprecated]

`plot.Nott_show_placseq`
[deprecated]

`plot.Nott_binwidth`
[deprecated]

`plot.Nott_bigwig_dir`
[deprecated]

`plot.Roadmap` [deprecated]

`plot.Roadmap_query`
[deprecated]

`plot.XGR_libnames`
[deprecated]

`server` [deprecated]

plot.types [deprecated]
plot.zoom [deprecated]
QTL_prefixes [deprecated]
vcf_folder [deprecated]
probe_path [deprecated]
file_sep [deprecated]
chrom_col [deprecated]
chrom_type [deprecated]
position_col [deprecated]
snp_col [deprecated]
pval_col [deprecated]
effect_col [deprecated]
stderr_col [deprecated]
tstat_col [deprecated]
locus_col [deprecated]
freq_col [deprecated]
MAF_col [deprecated]
A1_col [deprecated]
A2_col [deprecated]
gene_col [deprecated]
N_cases_col [deprecated]
N_controls_col [deprecated]
N_cases [deprecated]
N_controls [deprecated]
proportion_cases [deprecated]
sample_size [deprecated]
PAINTOR_QTL_datasets [deprecated]

Details

The primary functions of **echolocatoR** that expedite fine-mapping by wrapping many other **echolocatoR** functions into one. Encompasses steps including:

Subset & standardize Extract subsets of the full summary stats GWAS or QTL file and reformat them to be compatible with **echolocatoR**'s various functions

Calculate linkage disequilibrium Download and prepare the necessary LD matrix.

Fine-map Run various fine-mapping tools and merge the results into a single multi-finemap data.frame.

Plot Summarise the results in a multi-track plot for each locus.

See Also

Other MAIN: `finemap_loci()`

Examples

```
topSNPs <- echodata::topSNPs_Nalls2019
fullSS_path <- echodata::example_fullSS(dataset = "Nalls2019")

res <- echolocatoR::finemap_locus(
  fullSS_path = fullSS_path,
  topSNPs = topSNPs,
  locus = "BST1",
  finemap_methods = c("ABF","FINEMAP","SUSIE"),
  dataset_name = "Nalls23andMe_2019",
  fullSS_genome_build = "hg19",
  bp_distance = 1000,
  munged = TRUE)
```

`lfm` *lfm*

Description

lfm

Index