Package 'echolocatoR'

August 28, 2022

Type Package

Title Automated genomic fine-mapping

Version 2.0.0

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 echodata,
echotabix,
echoannot,
echoconda,
echoLD,
echoplot,
echofinemap,
catalogueR,
downloadR,
R.utils,
dplyr,
data.table,
ggplot2,
patchwork,
stringr,
lifecycle,
cli

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

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

RoxygenNote 7.2.1.9000

VignetteBuilder knitr

Config/testthat/edition 3

R topics documented:

`deprecated` *deprecated*

Description

deprecated

`Directory_info` *Retrieve the location of summary stats files*

Description

Retrieve the location of summary stats files

Usage

```
Directory_info(dataset_name, variable = "fullSS.local")
```

echoannot *echoannot*

Description

echoannot

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", "POLYFUN_SUSIE"),
  finemap_args = NULL,
  n_causal = 5,
  credset_thresh = 0.95,
  consensus_thresh = 2,
  fillNA = 0,
  conditioned_snps = "auto",
  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",
PAINTOR_QTL_datasets = NULL,
case_control = TRUE,
qtl_prefixes = NULL,
plot_types = c("simple"),
zoom = "1x",
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,
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()
)
```

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.
`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.

`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 *COJO*.

`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").

  - `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 (genome build: defined by user with `target_genome`).

`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*.

`PAINTOR_QTL_datasets`
: A list of QTL datasets to be used when conducting joint functional fine-mapping with *PAINTOR*.

`qtl_prefixes`
: If columns with QTL data is included in `dat`, you can indicate which columns those are with one or more string prefixes.

`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)`.

`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.

`nThread` Number of threads to parallelize over.

`verbose` Print messages.

Value

A merged data.frame 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 = 10000,
  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", "POLYFUN_SUSIE"),
  finemap_args = NULL,
  n_causal = 5,
  credset_thresh = 0.95,
  consensus_thresh = 2,
  fillNA = 0,
  conditioned_snps = 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",
  PAINTOR_QTL_datasets = NULL,
  qtl_prefixes = NULL,
  plot_types = c("simple"),
  zoom = "1x",
```

```
  nott_epigenome = FALSE,
  nott_show_placseq = FALSE,
  nott_binwidth = 200,
  nott_bigwig_dir = NULL,
  xgr_libnames = NULL,
  roadmap = FALSE,
  roadmap_query = NULL,
  show_plot = TRUE,
  remove_tmps = TRUE,
  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()
)
```

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.
`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.
`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 *COJO*.

`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").

- `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_C
/ 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 (genome build: defined by user with `target_genome`).

`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*.

`PAINTOR_QTL_datasets`
A list of QTL datasets to be used when conducting joint functional fine-mapping with *PAINTOR*.

`qtl_prefixes` If columns with QTL data is included in `dat`, you can indicate which columns those are with one or more string prefixes.

`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)`.

`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")).
show_plot	Print plot to screen.
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.
nThread	Number of threads to parallelize over.
verbose	Print messages.

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=10000,
  munged = TRUE)
```

snps_to_condition *Identify SNPs to condition on*

Description

When running conditional analyses (e.g. *GCTA-COJO*), this functions automatically identifies SNP to condition on.

Usage

```
snps_to_condition(conditioned_snps, topSNPs, loci)
```

Index