Package 'McSwan'

May 29, 2017

Type Package

Title Multiple-collision Coalescent SWeep ANalyser

Version 1.1.1

Date 2017-05-25

Author Remi Tournebize

Maintainer Remi Tournebize <remi.tournebize@ird.fr>

Description The Multiple-collision Coalescent SWeep ANalyser (McSwan) is a genome scan approach which aims at simultaneously detecting and dating hard selective sweeps from genome sequence data of multiple individuals from (possibly) multiple populations.

License GPL (>= 2)

Imports Rcpp (>= 0.12.5),
phyclust (>= 0.1-16),
MASS (>= 7.3),
caret (>= 6.0-70),
ggplot2 (>= 2.1.0),
abc (>= 2.1),
pls (>= 2.6-0),
Matrix (>= 1.2),
reshape2 (>= 1.4.1),
roxygen2 (>= 5.0.1),
quantreg (>= 5.29),
RColorBrewer (>= 1.1)

LinkingTo Rcpp

LazyData TRUE

RoxygenNote 5.0.1

R topics documented:

`coalesce` *Coalescent simulations*

Description

Given a `referenceTable` object generated with `generate_priors`, this function performs coalescent simulations using Hudson's *MS* simulator.

Usage

```
coalesce(x, execute = TRUE, verbose = FALSE)
```

Arguments

`x`	an initialized `referenceTable` object
`execute`	(logical) whether to execute the simulations
`verbose`	(logical) verbose mode

Details

If you want to see the demographic command lines for each model before simulating, set `execute = FALSE` and `verbose = TRUE`.

Value

A reference table with all simulated site frequency spectra stored in the `SFS` slot.

References

Hudson, R.R. (2002). Generating samples under a Wright-Fisher neutral model of genetic variation. Bioinformatics, 18, 337-338.

See Also

`generate_priors`, `dim_reduction`

Examples

```
Please refer to the vignette.
```

combine.referenceTable — *Combine referenceTable objects*

Description

Combine two `referenceTable` objects prior to machine learning (i.e. prior to the `dim_reduction` call).

Usage

```
## S3 method for class 'referenceTable'
combine(V, v)
```

Arguments

`V`	a first `referenceTable` object with empty `DIMREDUC` slot
`v`	a second `referenceTable` object with empty `DIMREDUC` slot

Value

A `referenceTable` object combining the two original `referenceTable` objects.

See Also

`generate_priors`, `coalesce`, `dim_reduction`

convert_VCF — *Filters and converts a VCF into a file of per-population per-SNP allele counts*

Description

This function calls a Python script which reads in a VCF file, extracts chromosome-specific variants, filters the variants to retain only high-quality biallelic SNPs, calculates the population-specific allele counts for each retained SNP (minor or derived allele counts, depending on your `referenceTable` specifications) and writes the results into a new external file.

Usage

```
convert_VCF(vcfPath, pops, reftb, outPath, chromosome, haploidize = FALSE,
  minQUAL = NULL)
```

Arguments

`vcfPath`	path to the VCF input file
`pops`	a dataframe of sample names and population indices *or*, alternatively, an external file with the same format (see *Details*)
`reftb`	an initialized `referenceTable` object
`outPath`	path to the file in which will be written the per-population per-SNP allele counts
`chromosome`	name of the chromosome to analyze (should match a chromosome name in the VCF file)
`haploidize`	set this option to TRUE to randomly draw a single allele out of the diploid genotypes (useful to mitigate the impact of low-coverage sequencing), note that if `haploidize=TRUE` the number of chromosomes in your MS command must be adjusted!
`minQUAL`	any SNP with `QUAL < minQUAL` will be filtered out (set `minQUAL = NULL` to disable the filtering)

Details

Non biallic SNPs in the VCF file will be automatically filtered out. If there are more samples in the VCF file than samples specified in *pops*, those supernumary samples will be ignored.

The `pops` dataframe must have two columns: (i) the names of the samples, matching the names in the VCF file; (ii) the indices (integers) of the population to which each sample belongs, these indices must correspond to the indices of the populations specified in the *MS*-formatted demographic history. For instance, if the sample B101 in the VCF belongs to the third-indexed population of your demographic history, the first line will read: first column:B101, second column: 3. Note that *MS* starts population indexing at 1 (included). If `pops` is an external file, it should be formatted similarly, with tab-separated columns, no header, no rownames and without quote marks around the sample names.

Value

No internal return. The function will write an external file. This file has a header summarizing the conversion parameters (starting with "#"), followed by the SNP-wise allele counts. For each line, there are three space-delimited fields:

- the SNP position on the chromosome;
- a SNP coefficient: always 1 if derived allele counts; can be 1 or 0.5 if minor allele counts – 0.5 corresponding to a SNP which can have two possible alternative states;
- the per-population allele counts, formatted `ac.i1.i2.[...].in` with (*i1*, *i2*, ..., *in*) the allele counts in populations (1, 2, ..., *n*).

See Also

`get_SFS`

Examples

```
Please refer to the vignette.
```

`dim_reduction` *Supervised machine learning of the selection signals*

Description

Performs supervised learning algorithms (LDA and PLSR) on the summary statistics (joint multidimensional allele frequency spectra) prior to genome scan. This function is to be run after `coalesce`.

Usage

```
dim_reduction(x, removeCollinearCols = TRUE, compress_SFS = TRUE,
  LDA_xPC = 0.1, LDA_sizePerModel = NULL, PLS_normalize = TRUE,
  PLS_ncomp = NULL, PLS_ncomp_method = "elbow", PLS_maxncomp = 100,
  PLS_plotCV = TRUE)
```

Arguments

`x`	a `referenceTable` object with non-empty `PRIORS` and `SFS` slots
`removeCollinearCols`	(logical) whether to remove collinear columns from the site frequency spectra prior to LDA & PLS dimension reduction (TRUE is recommended)
`LDA_xPC`	(float between 0 and 1) number of principal components to retain as a fraction of the original number of cells in the SFS; LDA will be trained on the PCA-projected simulations
`LDA_sizePerModel`	(integer) number of simulations per model to consider for training the LDA; *NULL* (by default) forces to use all data
`PLS_normalize`	(logical) whether to normalize the spectra prior to PLS analysis (TRUE is recommended)
`PLS_ncomp`	(integer) if NULL, automatically select the fittest number of PLS components, else an integer to manually set the number of retained components
`PLS_ncomp_method`	(string) the method to automatically select the best number of features: either "*elbow*", "*onesigma*" or "*randomization*" (cf. `selectNcomp` in package `pls`)
`PLS_maxncomp`	(integer) if automatic selection of PLS features, the maximum number of PLS components to consider before downsampling to the fittest set
`plot_PLScv`	(logical) if TRUE, plots the PLS cross-validation graphs

Value

An object of class `referenceTable` with filled-in `DIMREDUC` slot.

See Also

`coalesce`, `plsr`, `lda`

Examples

```
Please refer to the vignette.
```

`generate_priors` *Initialize the reference table*

Description

Semi-automatic initialization of the prior distributions for all model parameters.

Usage

```
generate_priors(msDemography, No, fold, windowSize = 1e+05, nSimul = 10000,
  restrictDemesTo = NULL, sweepAge = NULL)
```

Arguments

`msDemography`	(string) the neutral demographic history of the metapopulation, formatted according to Hudson's *MS* conventions, where $\theta = 4N_o\mu$ with N_o the diploid effective size; times are scaled in units of $4N_o$ and population size rescaling in units of N_o; note that the demographic history can involve unsampled (ghost) populations
`No`	(integer) the **diploid** effective size of the reference population
`fold`	(logical) whether allele counts are to be defined in reference to the **minor** alleles (`fold = TRUE`, producing **folded** site frequency spectra) or to the **derived** alleles (`fold = FALSE`), producing **unfolded** site frequency spectra. In the derived allele case, you must ensure that the SNPs in your VCF dataset are polarized relatively to some ancestral individuals (thus genotypes are defined in terms of ANCESTRAL/DERIVED instead of REF/ALT).
`windowSize`	(integer) length of the sequences (in bp) to simulate, the larger the better (> 100.000)
`nSimul`	(integer) number of simulations to perform for each model (the larger, the better but beware memory usage! previous tests suggested that 20.000 were sufficient for moderately complex demographic scenarios)
`restrictDemesTo`	(array of integers) by default, McSwan will detect population-specific sweeps across all the populations specified in the `MS` command; however, if you want to restrict the analysis to a particular subset of demes, provide their **indices** in a vector; note that population indices must correspond to their position under the `-I` switch of the `MS` command, and note that the index of the first population is **1**
`sweepAge`	(special list) prior distribution for the sweep ages (scaled in generations before present); if `NULL` and the number of populations is superior to 2, the prior distribution ranges will be automatically determined, otherwise it is mandatory to specify the distribution manually, see `Details`)

Details

Prior distributions must be specified using the following syntax: `list("P", arg1, arg2)` with *P* the name of the distribution function (e.g. `runif` for the uniform distribution, `rlogunif` for the log-uniform; please make sure you have quoted the function name and removed the argument brackets); *arg1* and *arg2* respectively the first and second arguments of the function (e.g. for `runif` will be the lower and upper limits of the distribution). Note that if the upper and lower limits of the distribution are distant by more than 3 units in the *log10* scale, it is recommended to use the `rlogunif` distribution.

If you manually provide the prior distributions for the **sweep ages**, you have two options:

- either specify a single `list("P", arg1, arg2)` which will be the common distribution set **for all** the demes,
- or specify a list of distribution-lists, each distribution-list corresponding to the sweep age distribution for a specific deme (indexed as they appear in the `ms` command); for instance, for 2 demes, one would specify: `list(list("rlogunif", T_1, TT_1), list("runif", T_2, TT_2))`. Please note that you must specify distributions for every deme, even if you have restricted the analysis to specific demes *via* the `restrictDemesTo` option.

Value

An object of class *referenceTable* with initialized `PRIOR` slot.

References

Hudson, R.R. (2007) `ms` - a program for generating samples under neutral models `http://home.uchicago.edu/rhudson1/source/mksamples/msdir/msdoc.pdf`

See Also

`coalesce`

Examples

```
Please refer to the vignette.
```

`generate_pseudoobs` *Generates pseudo-observed genomic fragments under arbitrary demographic histories with and without positive sweeps*

Description

Simulations performed with Ewing's *et al.* `MSMS` software.

Usage

```
generate_pseudoobs(reftb, nSimul, L, recRate, sweepingIsl = NULL,
  sweepAge = NULL, sweepPos = 0.5, Smu = NULL, sAA = 1,
  verbose = FALSE, default_sweepAge_prior_func = "runif")
```

Arguments

`reftb`	an initialized *referenceTable* object
`nSimul`	(integer) number of independent genomic fragments to simulate for each evolutionary model
`L`	(integer) genomic fragment length (in base pairs) (should be significantly higher than the `windowSize` so that McSwan can perform a sliding-window scan)
`recRate`	(special list) prior distribution for the recombination rates (see `Details`)

`sweepingIsl`	(array of integers) by default, McSwan will generate fragments under all population-specific sweep models; if you want to restrict the simulations of sweep models to some given populations, provide the population **indices** in a vector; note that population indices correspond to their position under the `-I` switch of the `MS` command, and note that the index of the first population is **1**
`sweepAge`	(special list) prior distribution for the sweep ages (scaled in generations before present) (if `NULL` and the number of demes is superior to 2, the prior distribution will be automatically determined, otherwise it is mandatory to specify the distribution manually, see `Details`)
`sweepPos`	(numeric) relative position of the beneficial mutation (eg. for $sweepPos = 0.5$, the beneficial mutation will be located at the center of the genomic region)
`Smu`	forward mutation rate for the advantageous allele (per base per generation); if `NULL` this rate will be equal to the mean mutation rate μ
`sAA`	the selection coefficient of the individual homozygote for the beneficial allele, see MSMS manual.
`verbose`	(logical) verbose mode
`default_sweepAge_prior_func`	if `sweepAge = NULL` you can force here the prior distribution of the sweep ages (e.g. "runif" or "rlogunif"), however the range of the distribution will still be set automatically

Details

Prior distributions must be specified using the following syntax: `list("P", arg1, arg2)` with *P* the name of the distribution function (e.g. `runif` for the uniform distribution, `rlogunif` for the log-uniform; please make sure you have quoted the function name and removed the argument brackets); *arg1* and *arg2* respectively the first and second arguments of the function (e.g. for `runif` will be the lower and upper limits of the distribution). Note that the log-uniform distribution will tend to favour the sampling of very recent sweeps.

If you manually provide the prior distributions for the **sweep ages**, you have to respect the following format:

- specify a list of distribution-sublists, each distribution-sublist corresponding to the sweep age distribution for the specific deme(s) you provided in the `sweepingIsl` argument (indexed as they appear in the `ms` command) (if `sweepingIsl = NULL` you will need to provide the distribution-sublists for **all** demes); for instance, for `sweepingIsl = c(1,2)`, one would specify: `list(list("rlogunif", T_1, TT_1), list("runif", T_2, TT_2))`.

Value

An object of class `validationTable` containing positional allele counts in the `SFS` slot.

References

Ewing et Hermisson (2010) MSMS: a coalescent simulation program including recombination, demographic structure and selection at a single locus. *Bioinformatics*

getmode — *Calculates the mode of a univariate distribution*

Usage

```
getmode(x, na.rm = TRUE, ...)
```

Arguments

`x`	a vector of numeric values

Value

A numeric value corresponding to the mode of the empirical distribution.

get_SFS — *Import a file of per-population per-SNP allele counts*

Usage

```
get_SFS(inputFile, reftb)
```

Arguments

`inputFile`	a file of per-population per-SNP allele counts, previously generated with `convert_VCF`
`reftb`	an initialized `referenceTable` object

Details

The function needs the *MS*-formatted demographic model contained in the `referenceTable` object to generate the template of the joint allele frequency spectra.

Value

Returns a list object of class `observedDataset`, containing three elements:

- `template` the template of the joint allele frequency spectra
- `obsData` a dataframe of SNP positions and allele counts
- `folded` whether those are minor (`TRUE`) or derived (`FALSE`) allele counts

See Also

`convert_VCF`

Examples

```
Please refer to the vignette.
```

gscan *Ensemble genome scan for sweep detection & sweep age estimation*

Usage

```
gscan(X, reftb, firstPos = NULL, lastPos = NULL, minSNP = 10,
  windowSizes = seq(10000, 2e+05, length.out = 20), nSteps = 20,
  discard_extraRange = TRUE)
```

Arguments

`X`	an observed dataset obtained with `get_SFS()` or a pseudo-observed dataset obtained with `generate_pseudoobs()`
`reftb`	a `referenceTable` object, with non-empty `PRIORS`, `SFS` and `DIMREDUC` slots
`firstPos`	(integer) the first position of the genome scan (NULL to automatically start at the first known position)
`lastPos`	(integer) the last position of the genome scan (NULL to automatically stop at the last known position)
`minSNP`	(integer) minimum number of SNPs in the window to consider it valid for inference
`windowSizes`	(array of integers) a vector of window lengths over which genome scans will be performed iteratively, you may consider using the R function seq(minimum_length, maximum_length, length.out = number_of_values) to sample a set of equally-spaced lengths
`nSteps`	(integer) number of overlapping shifts (the higher the number, the more overlapping windows will be introduced, increasing the scan resolution)
`discard_extraRange`	(logical) if TRUE, will discard sweep age estimations going beyond the prior range (TRUE is recommended)

Value

A list to be further processed with the `thin()` function.

See Also

`get_SFS`, `generate_pseudoobs`, `thin`

print.observedDataset *Print a summary of an* `observedDataset` *object content*

Usage

```
print.observedDataset(X)
```

Arguments

`X`	a `observedDataset` object (generated with `get_SFS`)

See Also

`get_SFS`

`print.referenceTable` *Print a summary of a* `referenceTable` *object content*

Usage

```
print.referenceTable(X)
```

Arguments

`X`	a `referenceTable` object (generated with `generate_priors`)

See Also

`generate_priors`

`print.validationTable` *Print a summary of a* `validationTable` *object content*

Usage

```
print.validationTable(X)
```

Arguments

`X`	a `validationTable` object (generated with `generate_pseudoobs`)

See Also

`generate_pseudoobs`

`rlogunif` *Generates log10-uniform distributions*

Description

This function samples values out of a log10-uniform distribution.

Usage

```
rlogunif(n, a, b)
```

Arguments

`n`	number of values to sample
`a`	lower limit of the distribution
`b`	upper limit of the distribution

Details

`Y` follows a log10-uniform distribution <=> `Y=10^(x)` where `x` follows a uniform distribution between `log10(a)` and `log10(b)`.

Value

A vector of *n* values drawn from a log10-uniform distribution.

`set_session` *Initialize the working session*

Usage

```
set_session(tempDir, pythonPath = "python", javaPath = "java")
```

Arguments

`tempDir`	path/name of a temporary directory (will be recursively created if it does not exist; if the directory already exists, its content will be overridden)
`pythonPath`	path to the `python` executable, simply give "python" if python is callable as an environment variable
`javaPath`	path to the `java` executable, simply give "java" if python is callable as an environment variable

Details

If python is not set as an environment variable, please see <https://docs.python.org/2/using/windows.html>, alternatively you can specify the python executable ("java.exe" for instance on Windows). Java is to be specified only if you simulate pseudo-observed datasets for the validation procedure, otherwise you can leave its default value.

Examples

```
Please refer to the vignette.
```

`summary.observedDataset` *Genomic SNP density in the observed dataset*

Description

Given a window length, computes the distribution of within-window SNP counts in the observed dataset. This function is useful to get an idea about the SNP density in the VCF to suggest appropriate window size ranges.

Usage

```
summary.observedDataset(obs, windowSize)
```

Arguments

`obs`	an `observedDataset` object, generated with `get_SFS`
`windowSize`	(integer) a window size (in base pairs)

Value

A summary of the distribution of the within-window SNP counts.

See Also

`generate_priors`

Examples

```
load(system.file("data", "UNFOLDED_POLARIZED_2dSFS_CEU-LWK_20-20_chr2", package = "McSwan"))
summary(obs, windowSize = 10000)
```

`summary.validationResult` *Summary of the sliding validation*

Usage

```
summary.validationResult(X, valtb, file)
```

Arguments

`X`	a `validationResult` object (i.e. returned by `gscan`)
`valtb`	a `validationTable` object (generated by `generate_pseudoobs`, the same object used in the previous `gscan` call)
`file`	path to a file where validation graphics will be plotted

Value

A list of 6 elements:

- `absolute.confusion.matrix`: confusion matrix with absolute counts
- `absolute.relative.matrix`: confusion matrix with relative counts
- `performance.rates`: specificity (TNR), sensitivity (TPR) and false discovery rate (FDR)
- `number.of.sweeps`: number of sweep contigs detected in each POD (ideally, should be 0 for the neutral model and 1 for the selective models)
- `sweep.inclusiveness`: number of times the detected sweeps encompass the beneficial mutation
- `NRMSE`: normalized/standardized root mean square error

Examples

```
Please refer to the vignette.
```

`thin` *Merge contiguous swept loci (tiling path inference)*

Description

This function merges contiguous regions detected as having experienced a selective sweep (i.e. following a `gscan`).

Usage

```
thin(scanResult, reftb, X, max_L = 1e+06, signif.threshold = 0.5,
  maxIter = 1000, stat = getmode)
```

Arguments

`scanResult`	a list returned by `gscan`
`reftb`	an initialized `referenceTable` object
`X`	the observed or pseudo-observed dataset used in the `gscan` call
`max_L`	(integer) the "horizon distance" (in base pairs), ie. the maximum distance from any position above which loci detected as under selection cannot be assumed to be related to the same sweep region
`signif.threshold`	(single or vector of numeric values) (between 0 and 1) a selection score cutoff above which we decide that a SNP is positive selected; if unique value, this cutoff will be used for all populations; alternatively, you can give a vector of cutoff values which will be specifically used for each population (e.g. signif.threshold=c(0.5, 0.2))
`maxIter`	(integer) maximum number of iterations used for the tiling path inference
`stat`	(function) a statistic used to center the SNP-wise sweep age estimates (`getmode` was shown to be the least biased but you may also try `median` or `mean`)

Value

Returns a dataframe of the estimated sweep contigs. One sweep region per line, and in columns:

- **`sweep.center`** the center of the sweep region contig
- **`sweep.lbound`** the first position of the sweep region contig
- **`sweep.rbound`** the last position of the sweep region contig
- **`BF`** aggregated Bayes Factor of the sweep region contig
- **`deme`** the population detected to have experienced a selective sweep (given as "*i*" and the index of the population in the *MS* command)
- **`sweepAge`** the point estimate for the sweep age
- **`sweepAge.IC.low`** lower boundary of the 95% confidence interval for the sweep age estimation
- **`sweepAge.IC.up`** upper boundary of the 95% confidence interval for the sweep age estimation
- **`sweepAge.postDistrib`** posterior distribution of the sweep age estimation

See Also

`gscan`

Examples

```
Please refer to the vignette.
```

Index