Package 'SimAdmixtR'

February 11, 2020

Type Package

Title Simulates a set of admixed diploid genotypes based on SNP allele frequencies

Version 0.1.0

Author Daniel W. Kennedy

Maintainer Daniel W. Kennedy <dan.w.kennedy@outlook.com>

Description The package imports the SNP allele frequencies from two or more origin populations, as well as the number and make-up of ancestors. It then randomly samples the ancestor genomes from the origin populations, and then, using Mendelian rules, simulates the inheritance of alleles down to the current generation. It can also repeat this process a specified number of times.
The package then outputs the set of repeated simulated genotypes as text file in the input format of the STRUCTURE program.

License MIT + file LICENSE

URL https://github.com/danwkenn/SimAdmixtR

Encoding UTF-8

LazyData true

Imports utils,
tibble,
dplyr,
magrittr,
reshape2,
purrr,
stringr,
tidyr,
shiny,
rlang

Suggests ggplot2,
knitr,
rmarkdown

RoxygenNote 6.1.1

VignetteBuilder knitr

R topics documented:

---

af_split *Internal function used for reading in allele frequency file.*

Description

Converts a string to vector of frequencies. e.g. "0.01/0.89" to c(0.01,0.89,0.1)

Usage

```
af_split(x, n_vari)
```

Arguments

`x`	character string containing numeric values separated by backslashes.
`n_vari`	number of variants.

---

back_convert.Internal *Internal function, not for use. Used to convert from numeric output to AGCT.*

Description

The output of the sim_single_sample function is integers, which index the variants of each of the SNPs. This function uses the allele frequency data-frame to convert these integers back to the variants (A,G,C,T).

Usage

```
back_convert.Internal(sim_samples, allele_freq_data)
```

Arguments

`sim_samples`	list outputted from the sim_single_sample function.
`allele_freq_data`	tibble ouputted from the read_in_af_data function.

Value

A list of `n_samples` elements, each corresponding to a sample. Each element is a matrix with a row for each allele, and a column for each SNP. The elements of the matrix are characters for nucleotides (A,C,G,T).

`genotype_finder` *Internal function which finds possible genotypes from available SNP variants.*

Description

Internal function which finds possible genotypes from available SNP variants.

Usage

```
genotype_finder(allele_freq_data)
```

`read_in_af_data` *Read in the population allele frequencies from prespecified CSV file format.*

Description

This function is used internally to read in allele frequency information and complete basic cleaning. To ensure that the correct probabilities are used in simulation, care should be taken to ensure the input file complies with the required format (see Details below), and that the order of the populations given in the input file is kept consistent with the input `ancestor_pop_label` in the simulation functions.

Usage

```
read_in_af_data(file)
```

Arguments

`file`	Path to CSV file containing the population allele frequencies.

Details

The CSV file must comply with the following format rules:

- Each row corresponds to a SNP.
- The file's first 3 columns must be the SNP ID, The chromosome and position, and the SNP variants.
- The variants must be specified by a list of nucleotides separated by backslashes '/', where the most frequent allele is first. E.g. 'A/G' specifies a SNP with the most common variant being 'A' and the least common being 'G'.
- In the following columns the derived allele frequencies in each pure population. The column names must have "(D)" at the end of their name to be recognised.

Value

A tibble data-frame with a column for the SNP ID called `snp_id`, a column for the variants called `variants` which contain character vectors for each SNP, and a column for each population of allele frequencies as sum-to-one constrained vectors.

Examples

```
#Download file to temporary directory:
temp_dir <- tempdir()
download.file(url = "https://raw.githubusercontent.com/danwkenn/SimAdmixtR/master/inst/example-files/input-
allele_frequencies.tbl <- read_in_af_data(file = paste0(temp_dir,"/input-af-example.csv"))
allele_frequencies.tbl
```

`run_App` *Run a basic User Interface to easily run simulation experiments and "download" the simulated data.*

Description

Run a basic User Interface to easily run simulation experiments and "download" the simulated data.

Usage

```
run_App()
```

Examples

```
## Not run: run_App()
```

`simulate_admixture` *Simulate genetic inheritance of SNPs based on population allele frequencies and family tree structure.*

Description

Simulate genetic inheritance of SNPs based on population allele frequencies and family tree structure.

Usage

```
simulate_admixture(n_samples, ancestor_pop_label, file)
```

Arguments

`n_samples`	The number of samples to simulate.
`ancestor_pop_label`	A vector of integers giving the highest level of the family tree, which consists only of ancestors from pure genetic populations. Integers index the population, with order given by the order in which the population dominant allele frequencies are provided in the input allele frequencies file.
`file`	File with input allele frequencies of ancestor populations. This file must be in a specific format as exemplified in ‘input-af-example.csv‘. For a complete description of the format see `read_in_af_data`.

Value

A list of `n_samples` elements, each corresponding to a sample. Each element is a matrix with a row for each allele, and a column for each SNP. The elements of the matrix are characters for nucleotides (A,C,G,T).

Examples

```
#Download file to temporary directory:
temp_dir <- tempdir()
download.file(url = "https://raw.githubusercontent.com/danwkenn/SimAdmixtR/master/inst/example-files/input-
sim_data <- simulate_admixture(
n_samples = 10,
ancestor_pop_label = c(1,2,2,2),
file = paste0(temp_dir,"/input-af-example.csv")
)
```

`sim_single_locus` *Function which simulates mendelian inheritance for a single allele, beginning with a random draws of variants based on the population allele frequencies.*

Description

The function proceeds as follows:

1. For each of the ancestors, using the ancestor's population index, draw two alleles randomly from the ancestor's population allele frequency.
2. Based on Mendelian Inheritance, for each generation, randomly and with equal probability draw one allele from the mother and one allele from the father. Therefore, each generation there will half the number of people/allele pairs.
3. Proceed until there is only one sample left.

Usage

```
sim_single_locus(ancestor_pop_label, allele_freq)
```

Arguments

`ancestor_pop_label`	A vector of integers giving the highest level of the family tree, which consists only of ancestors from pure genetic populations. Integers index the population, with order given by the order in which the population dominant allele frequencies are provided in the input allele frequencies file.
`allele_freq`	A list of vectors of the population allele frequencies. Index must match the index of `ancestor_pop_label`.

Value

A matrix with two rows and a single column. The elements correspond to the two alleles for the resulting genotype at the given SNP. The values are either 1 or 2, and correspond to the two variants in order of the supplied allele frequency vectors.

Examples

```
#Single grandparent:
sim_single_locus(
ancestor_pop_label = c(2,1,1,1),
allele_freq = list(
c(0.2,0.9),c(0.9,0.2)
))
```

`sim_single_sample` *Function which simulates inheritance for a set of genes for a single simulated person.*

Description

Function which simulates inheritance for a set of genes for a single simulated person.

Usage

```
sim_single_sample(ancestor_pop_label, allele_freq_list)
```

Arguments

`ancestor_pop_label`	A vector of integers giving the highest level of the family tree, which consists only of ancestors from pure genetic populations. Integers index the population, with order given by the order in which the population dominant allele frequencies are provided in the input allele frequencies file.
`allele_freq_list`	A list of allele frequencies for the ancestor populations. The upper level must be SNP, and the lower level corresponds to population frequencies in order of the population indexes in `ancestor_pop_label`.

`stitch` *Simple function to pair up the elements of two or more equal-length lists. Function works similar to zip in python.*

Description

Simple function to pair up the elements of two or more equal-length lists. Function works similar to zip in python.

Usage

```
stitch(list1, ...)
```

Arguments

`list1`	A list.
`...`	More lists of the same length.

Value

A list of the same length as the original lists. Each element of the list is itself a list, containing two elements: the corresponding element of the first list, and the corresponding element of the second list. The function should will also work with vectors of equal length.

Examples

```
listA = list(1,2,3,4)
listB = list("A","B","C","D")
stitch(listA,listB)
```

`string_to_vector` *Simple internal function to create a vector of characters from a string.*

Description

Simple internal function to create a vector of characters from a string.

Usage

```
string_to_vector(x)
```

`summarise_admixture` *Function finds the genotype frequencies for a simulated data-list, and the most likely genotype for each SNP.*

Description

Function finds the genotype frequencies for a simulated data-list, and the most likely genotype for each SNP.

Usage

```
summarise_admixture(data_list, pop_allele_file)
```

Arguments

`data_list`	Simulated data output from simulate_admixture.
`pop_allele_file`	Path to CSV file containing the population allele frequencies.

Value

A list containing:

- `frequencies`: A list of vectors, with one element for each SNP. The vectors have a single element for each possible genotype, and the value is the observed frequency of that genotype in the simulated data.
- `most_common`: a matrix of the most common genotype found in the sample.

`write_to_structure` *Take simulated genotypes and write to a STRUCTURE input file format.*

Description

Take simulated genotypes and write to a STRUCTURE input file format.

Usage

```
write_to_structure(sim_data, structure_snp_order = NULL,
  example_structure_file = NULL, sample_names = NULL,
  output = "test", type = c("txt", "csv"), keep_extra_snps = TRUE)
```

Arguments

`sim_data`	Output from simulate_admixture
`structure_snp_order`	(Optional) The desired order of SNPs for columns of output file.
`example_structure_file`	(Optional) A sample STRUCTURE input file to extract the desired order of SNPs from. If both `example_structure_file` and `structure_snp_order` are specified, the order given by this will take precedence.
`sample_names`	(Optional) vector of names for the simulated samples.
`output`	Name for the output file.
`type`	Options for txt or csv file outputs.
`keep_extra_snps`	If there are SNPs present in the simulated data, but not in the SNP order provided by `example_structure_file` or `structure_snp_order`, should the extra SNPs be kept and appended to the table (on the right)?

Value

`NULL`

Examples

```
#Download files to temporary directory:
temp_dir <- tempdir()
download.file(url = "https://raw.githubusercontent.com/danwkenn/SimAdmixtR/master/inst/example-files/input-
download.file(url = "https://raw.githubusercontent.com/danwkenn/SimAdmixtR/master/inst/example-files/simple

sim_data <- simulate_admixture(
n_samples = 10,
ancestor_pop_label = c(1,2,2,2),
file = paste0(temp_dir,"/input-af-example.csv")
)

write_to_structure(
sim_data = sim_data,
example_structure_file = paste0(temp_dir,"/example-structure-file.csv"),
output = "example-output",
type = c("txt","csv"))
```

Index