# Package 'ASAP'

April 18, 2024

**Title** ASsessing Ancestry through Principal component analysis

**Version** 0.0.1

**Description** Estimates global ancestry proportions from Principal Component Analyses through Non-Negative Least Squares.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** dplyr,
nnls,
ggplot2

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Suggests** knitr,
rmarkdown,
testthat (>= 3.0.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

## R topics documented:

asap *asap*

## Description

asap function takes a PCA as input and analyses it via NNLS to describe admixed individuals as a mixture of sources groups.

## Usage

```
asap(pca_input, as_file, sources = NULL, admixed = NULL)
```

## Arguments

| | |
|---|---|
| pca_input | R data.frame of PCA with N PCs |
| as_file | R data.frame with two columns: POP and A/S, POP column lists all populations to be considered, A/S indicatea whether the population should be considered as Admixed ('A') or as Source ('S') |
| sources | R vector indicating the groups that should be considered as Sources |
| admixed | R vector indicating the groups that should be considered as Admixed |

## Value

Returns a list containing the ancestries proportions per each Admixed group

## Examples

```
pca = read.table('example.pca') #OR
pca = read_eigen('data/TOY.pca.evec') #OR
pca = read_flash('data/TOY_flash.pca')

example_AS = read.table('data/Example_AS', header = T)

asap(pca_input = pca, as_file = example_AS) #OR
asap(pca_input = pca, sources = c('Source1','Source2','Source3'), admixed = c('Admixed1','Admixed2')
```

nnls.mat2 *nnls.mat2*

## Description

nnls.mat2 function solves nonnegative least squares problems. It requires two matrices, one ('donors') refers to the source groups, the second ('recipients') refers to the admixed groups

## Usage

```
nnls.mat2(donors, recipients)
```

### Arguments

| | |
|---|---|
| donors | Matrix with reference groups |
| recipients | Matrix with target groups |

### Value

Returns matrix describing the admixed groups as a mixture of the source groups, along with the residuals

### Examples

```
nnls.mat2(donors = my_source_individuals_matrix,recipients = my_admixed_individuals_matrix)
```

---

## pcs_distances *pcs_distances*

---

### Description

pcs_distances allows to estimate the cumulative euclidean distances of the PCs between the Sources, and plot the results (return_plot = 'YES').

### Usage

```
pcs_distances(pca_input, output_name, sources_file, return_plot = NULL)
```

### Arguments

| | |
|---|---|
| pca_input | Dataframe or table with PCA results |
| output_name | String indicating output name |
| sources_file | Two columns table: S1 and S2. Contains the pairs of sources that will be compared. |
| return_plot | NULL by default or 'YES' to plot with ggplot2 |

### Value

Returns an output_name.csv file with the cumulative distances

Returns a output_name.pdf file with the cumulative distances

### Examples

```
sources_file = read.table('~/data/Sources_Distances', header =T)
pca = read_flash(pca_input = 'data/TOY_flash.pca')
pcs_distances(pca, '~/data/output_distances', sources_file = sources_file, return_plot = 'YES')
```

plot_asap *plot_asap*

## Description

plot_asap allows to plot ASAP assignations as barplot, using either R base (default) or ggplot2 (type_ggplot = 'YES')

## Usage

```
plot_asap(asap_result, output_name, type_ggplot = NULL)
```

## Arguments

| | |
|---|---|
| asap_result | The matrix returned by asap() |
| output_name | output name for asap pdf plot |
| type_ggplot | NULL by default or 'YES' to plot with ggplot2 |

## Value

Returns an ASAP_plot.pdf file in the working directory

read.resampling *read.resampling*

## Description

read.resampling reads all resampled PCAs and performs ASAP on each one. It returns a list containing all ASAP results per each resampling. The output can then be used in the se.resampling function, to estimate the standard error.

## Usage

```
read.resampling(path_tofiles, file_pattern, as_file, eigentype)
```

## Arguments

| | |
|---|---|
| path_tofiles | string containing directory path to the resampling files. |
| file_pattern | string containing common pattern to find the all PCAs obtained by resampling |
| as_file | R data.frame with two columns: POP and A/S, POP column lists all populations to be considered, A/S indicates whether the population should be considered as Admixed ('A') or as Source ('S') |
| eigentype | if present, PCA will be read through read_eigen() function, if absent PCA will be loaded via read_flash() |

## Value

Returns a table containing the ASAP results per each resampled set.

## Examples

```
example_as = read.table('data/Example_AS', header=TRUE)

pca_jackknife = read.resampling(path_tofiles = 'data/', file_pattern = '*_Jack*', as_file = Example_AS, eigentype)
pca_jackknife = read.resampling(path_tofiles = 'data/', file_pattern = '*_Jack*', as_file = Example_AS)
```

---

`read_eigen` *read_eigen*

---

## Description

read_eigen reads smartpca output from EIGENSTRAT, where the first column contains Family ID, the second sample ID, and the following the PCs

## Usage

```
read_eigen(pca_input)
```

## Arguments

| | |
|---|---|
| `pca_input` | points to the directory and pca.evec file |

## Value

Returns a PCA matrix with new header: POP ID PC1 PCN CC

## Examples

```
read_eigen(pca_input = 'data/TOY.pca.evec')
```

---

`read_flash` *read_flash*

---

## Description

read_flash reads PCA output from flashpca, where the first column contains Family ID, the second sample ID, and the following the PCs

## Usage

```
read_flash(pca_input)
```

### Arguments

| | |
|---|---|
| pca_input | point to the directory and flash pca file |

### Value

Returns a PCA matrix with new header: POP ID PC1 PCN

### Examples

```
read_flash(pca_input = 'data/TOY_flash.pca')
```

---

## se.resampling *se.resampling*

---

### Description

se.resampling estimates the Standard Error comparing a main ASAP run and multiple other ASAP runs, for example runs obtained with jackknife resampling. se.resampling takes three inputs: a list with the main ASAP result, a list with the resampling ASAP results, and a numeric vector containing the number of SNPs per each chromosome.

### Usage

```
se.resampling(nnls_main, nnls_resampling, chromovec)
```

### Arguments

| | |
|---|---|
| nnls_main | list obtained from the asap() function on the main set. |
| nnls_resampling | list contining multilpe asap() function results, on the resampled set. |
| chromovec | a numeric vector containing the number of SNPs per each chromosome (ie. chromovec = rep(1000,times = 22)) |

### Value

Returns a table containing the standard error per each target given the source group.

### Examples

```
se.resampling(nnls_main = ASAP_main, nnls_resampling = ASAP_resampling, chromovec)
```

write_asap *write_asap*

## Description

write_asap allows to save ASAP results in a table-like format.

## Usage

```
write_asap(asap_input, output_name)
```

## Arguments

| | |
|---|---|
| asap_input | R list returned by asap() function |
| output_name | string containing the file output name |

## Examples

```
asap_results <- asap(pca_input = pca, as_file = example_as)
write_asap(asap_input = asap_results, output_name = 'my_dir/my_asap_results.txt')
```

# Index