Package 'AwesomePackage'

September 27, 2022

Title Infer ancestry with some models and fit those models with some algorithms

Version 1.4.0

Description We use the classical PSD model for ancestor inference, which has been widely used, such as STRUCTURE (Pritchard et al. 2000, MCMC), FRAPPE (Tang et al. 2005, EM), ADMIXTURE (Alexander et al. 2009, SQP), fastSTRUCTURE (Raj et al. 2014, VI), TeraStructure (Gopalan et al. 2017, SVI). We illustrate the close relationship between the PSD model, the Poisson NMF model, the multinomial topic model and the LDA model, which can optimize the algorithm. We use Expectation-Maximization algorithm (EM), sequential quadratic programming algorithm (SQP), variational inference algorithm (VI) and stochastic variational inference algorithm (SVI) to fit the model, then illustrate the relationships and differences between these algorithms through simulation experiments and real data experiments.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

URL https://github.com/JONATHONCHOW/AwesomePackage, https://jonathonchow.github.io/AwesomePackage/

BugReports https://github.com/JONATHONCHOW/AwesomePackage/issues

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppEigen

Imports cowplot, ggplot2, MCMCpack, progress, Rcpp, stats

Suggests knitr, rmarkdown, R.rsp

VignetteBuilder knitr, R.rsp

LazyData true

R topics documented:

`data_HGDP` *Sample data from the Human Genome Diversity Project*

Description

A group of scientists at Stanford University have collaborated on a large study to understand genetic diversity in human populations. They analyzed genomic DNA from 1,043 individuals from around the world, determining their genotypes at more than 650,000 SNP loci, with the Illumina BeadStation technology. Genomic DNA samples from these fully-consenting individuals were collected by the Human Genome Diversity Project (HGDP), in a collaboration with the Centre Etude Polymorphism Humain (CEPH) in Paris. The collection they tested is referred to as the "HGDP-CEPH Human Genome Diversity Cell Line Panel". They represent 51 different populations from Africa, Europe, the Middle East, South and Central Asia, East Asia, Oceania and the Americas.

Format

`data_HGDP` is a 942 x 50000 matrix.

Source

https://cephb.fr/en/hgdp_panel.php

data_TGP *Sample data from the 1000 Genomes Project*

Description

The 1000 Genomes Project (TGP) created a catalogue of common human genetic variation, using openly consented samples from people who declared themselves to be healthy. The reference data resources generated by the project remain heavily used by the biomedical science community. The International Genome Sample Resource (IGSR) maintains and shares the human genetic variation resources built by the 1000 Genomes Project. We also update the resources to the current reference assembly, add new data sets generated from the 1000 Genomes Project samples and add data from projects working with other openly consented samples.

Format

`data_TGP` is a 1092 x 50000 matrix.

Source

https://www.internationalgenome.org/

hello_world *Hello world*

Description

Hello world and add "AwesomePackage" to NAMESPACE.

Usage

```
hello_world()
```

Value

A string "Data science is fantastic!".

Examples

```
hello_world()
```

map_HGDP — *Corresponding tables for individuals and populations of the HGDP dataset*

Description

map_HGDP gives the corresponding relationship among individuals, small populations and large populations. plot_structure uses this relationship to group data and draw structure plot.

Format

map_HGDP is a table with three columns.

Superpop Large populations, such as Asians, Africans, Europeans.

Pop Small populations, such as Chinese, British, Norwegian.

Indiv Individuals.

See Also

plot_structure

map_TGP — *Corresponding tables for individuals and populations of the TGP dataset*

Description

map_TGP gives the corresponding relationship among individuals, small populations and large populations. plot_structure uses this relationship to group data and draw structure plot.

Format

map_TGP is a table with three columns.

Superpop Large populations, such as Asians, Africans, Europeans.

Pop Small populations, such as Chinese, British, Norwegian.

Indiv Individuals.

See Also

plot_structure

plot_index_vs_K *Plot the relationship between index and K*

Description

Draw a diagram of the relationship between the index and K using package ggplot2. The index can be loglikelihood, error, ELBO, and time.

Usage

```
plot_index_vs_K(
  L,
  methods,
  index.id = c("loglik", "error", "elbo", "time"),
  start.point = 2,
  colors = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7"),
  linetypes = "solid",
  linesizes = 0.5,
  shapes = 19,
  fills = "white",
  theme = function() theme_cowplot(12),
  title = NULL
)
```

Arguments

L	A list where each element is a vector of index with respect to K.
methods	A vector of the same length as L where each element is a name of the fitting algorithm, such as "em", "sqp", "vi", "svi".
index.id	Choose index. Should be one of "loglik", "error", "elbo", "time".
start.point	The initial point of K.
colors	The colors used to draw curves.
linetypes	The line types used to draw curves.
linesizes	The line sizes used to draw curves.
shapes	The shapes used to draw points.
fills	The fill colors used to draw points.
theme	The ggplot2 theme.
title	Title of the plot.

Value

A ggplot object.

Examples

```
L <- list(c(-50,-20,-10,-1,-0.5), c(-30,-5,-1,-0.1))
plot_index_vs_K(L, c("fun","more fun"), index.id = "loglik")
L <- list(c(0.1,0.5,0.7,1.2), c(1.6,0.2,0.8,1.5,2.4))
plot_index_vs_K(L, c("fun","more fun"), index.id = "error")
L <- list(c(-10,-2,-0.5,-0.1), c(-5,-1,-0.1))
plot_index_vs_K(L, c("fun","more fun"), index.id = "elbo")
L <- list(c(10,15,20), c(12,18,30,32))
plot_index_vs_K(L, c("fun","more fun"), index.id = "time")
```

---

plot_loss *Plot the loss function*

Description

Plot the loss function against the number of iterations using package ggplot2.

Usage

```
plot_loss(
  L,
  methods,
  sample.rate,
  epsilon = 0.01,
  colors = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7"),
  linetypes = "solid",
  linesizes = 0.5,
  shapes = 19,
  fills = "white",
  theme = function() theme_cowplot(12),
  title = NULL
)
```

Arguments

L	A list where each element is a vector of loss functions with respect to the number of iterations.
methods	A vector of the same length as L where each element is a name of the fitting algorithm, such as "em", "sqp", "vi", "svi".
sample.rate	The sampling rate of the loss function.
epsilon	A small, positive number added to the vertical axis so that the logarithmic scale does not over-emphasize very small differences.
colors	The colors used to draw loss curves.
linetypes	The line types used to draw loss curves.
linesizes	The line sizes used to draw loss curves.
shapes	The shapes used to draw points at iterations.
fills	The fill colors used to draw points at iterations.
theme	The ggplot2 theme.
title	Title of the plot, such as "K = 2", "K = 3".

Value

A ggplot object.

Examples

```
L <- list(c(-100,-20,-10,-1,-0.5,-0.3,-0.1), c(-30,-5,-1,-0.1,-0.05))
plot_loss(L, c("fun","more fun"), 1)
```

`plot_structure` *Plot the population proportion*

Description

Plot the population proportion of individuals using package ggplot2.

Usage

```
plot_structure(
  P,
  pops = NULL,
  label = NULL,
  map.indiv = NULL,
  map.pop = NULL,
  gap = NULL,
  colors = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#EE2C2C", "#CC79A7",
    "#8968CD", "#FF83FA", "#EECFA1", "#A52A2A", "#4169E1", "#FFFF00", "#BFEFFF",
    "#FF1493"),
  font.size = 9,
  title = NULL,
  subtitle = NULL
)
```

Arguments

`P`	The proportion matrix.
`pops`	Population order options.
`label`	The original order of individuals. This option is only for data that needs to be grouped.
`map.indiv`	The new order of individuals. This option is only for data that needs to be grouped.
`map.pop`	The order of populations. This option is only for data that needs to be grouped.
`gap`	Gaps between groups. This option is only for data that needs to be grouped.
`colors`	Theme color options.
`font.size`	Font size used in plot.
`title`	Title of the plot, such as "EM", "SQP", "VI", "SVI".
`subtitle`	Subtitle of the plot, such as "K = 2", "K = 3".

Value

A ggplot object.

Examples

```
P <- matrix(c(0.5,0.3,0.8, 0.5,0.7,0.2), 3, 2)
plot_structure(P, title = "FUN")
```

---

psd_error *Compute the prediction error*

Description

Compute the deviance residuals under the binomial model averaged over all entries as the prediction error.

Usage

```
psd_error(G, result)
```

Arguments

G	The I x J matrix of counts; all entries of G should be taken from {0,1,2}.
result	Output of psd_fit_em, psd_fit_sqp, or psd_fit_vi.

Value

A value indicates the accuracy of the prediction.

Examples

```
G <- matrix(c(0,0,1, 0,2,1, 1,0,1, 0,1,0, 1,0,0), 3, 5)
result <- psd_fit_em(G, 2, 1e-5, 10)
psd_error(G, result)
```

---

psd_fit_em *Use EM algorithm to fit PSD model*

Description

Fit PSD model with EM algorithm, and use the loss function as a stopping criterion.

Usage

```
psd_fit_em(G, K, epsilon = 1e-05, maxiter = 500)
```

Arguments

G	The I x J matrix of counts; all entries of G should be taken from {0,1,2}.
K	An integer 2 or greater giving the matrix rank.
epsilon	Convergence criterion.
maxiter	The maximum number of iterations.

Value

A List with the following parameters:

P The population scale matrix of the individuals.

F The gene scale matrix of the populations.

Loss A vector represents the value of the loss function which records once for 10 iterations.

Iterations An integer represents the number of iterations.

Examples

```
G <- matrix(c(0,0,1, 0,2,1, 1,0,1, 0,1,0, 1,0,0), 3, 5)
psd_fit_em(G, 2, 1e-5, 10)
```

---

psd_fit_sqp *Use SQP algorithm to fit PSD model*

Description

Fit PSD model with SQP algorithm, and use the loss function as a stopping criterion.

Usage

```
psd_fit_sqp(G, K, epsilon = 1e-05, maxiter = 50, initem = 100)
```

Arguments

G	The I x J matrix of counts; all entries of G should be taken from {0,1,2}.
K	An integer 2 or greater giving the matrix rank.
epsilon	Convergence criterion.
maxiter	The maximum number of iterations.
initem	A number of iterations when using EM algorithm for initialization.

Value

A List with the following parameters:

P The population scale matrix of the individuals.

F The gene scale matrix of the populations.

Loss A vector represents the value of the loss function which records once for 10 iterations.

Iterations An integer represents the number of iterations.

Examples

```
G <- matrix(c(0,0,1, 0,2,1, 1,0,1, 0,1,0, 1,0,0), 3, 5)
psd_fit_sqp(G, 2, 1e-5, 10, 10)
```

psd_fit_svi *Use SVI algorithm to fit PSD model*

Description

Fit PSD model with SVI algorithm, and use the loss function as a stopping criterion.

Usage

```
psd_fit_svi(
  G,
  K,
  epsilon = 1e-05,
  maxiter = 5e+05,
  val_iter = 10000,
  maxdrop = 3,
  maxiter.sample = 100,
  maxiter.val = 2000,
  val_J = 0.05,
  val_I = 0.1,
  tau = 1,
  kappa = 0.5
)
```

Arguments

G	The I x J matrix of counts; all entries of G should be taken from {0,1,2}.
K	An integer 2 or greater giving the matrix rank.
epsilon	Convergence criterion.
maxiter	The maximum number of iterations.
val_iter	The number of iterations between each validation set sampling.
maxdrop	The maximum number of consecutive decreases in the loss function. Beyond this value the loop will stop.
maxiter.sample	The maximum number of iterations in the sampling section.
maxiter.val	The maximum number of iterations in the validation section.
val_J	Sample proportion of SNPs in validation set.
val_I	Sample proportion of individuals in validation set.
tau	A parameter of the descending direction of SVI algorithm.
kappa	A parameter of the descending direction of SVI algorithm.

Value

A List with the following parameters:

P The population scale matrix of the individuals.

Loss A vector represents the value of the loss function which records once for 10 iterations.

MaxLoss Maximum loss function value. Unlike other algorithms, we observe the loss function on the validation set. Therefore, monotonicity is not guaranteed, that is, the maximum value does not necessarily occur at the end, so the maximum value needs to be recorded.

Iterations An integer represents the number of iterations.

Examples

```
# Refer to Articles in AwesomePackage.
```

`psd_fit_vi` *Use VI algorithm to fit PSD model*

Description

Fit PSD model with VI algorithm, and use the loss function as a stopping criterion.

Usage

```
psd_fit_vi(G, K, epsilon = 1e-05, maxiter = 500)
```

Arguments

`G`	The I x J matrix of counts; all entries of G should be taken from {0,1,2}.
`K`	An integer 2 or greater giving the matrix rank.
`epsilon`	Convergence criterion.
`maxiter`	The maximum number of iterations.

Value

A List with the following parameters:

`P` The population scale matrix of the individuals.

`F` The gene scale matrix of the populations.

`Loss` A vector represents the value of the loss function which records once for 10 iterations.

`Iterations` An integer represents the number of iterations.

Examples

```
G <- matrix(c(0,0,1, 0,2,1, 1,0,1, 0,1,0, 1,0,0), 3, 5)
psd_fit_vi(G, 2, 1e-5, 10)
```

`psd_loglikelihood` *Compute the loglikelihood*

Description

Compute the maximum loglikelihood function of the PSD model.

Usage

```
psd_loglikelihood(G, result)
```

Arguments

G	The I x J matrix of counts; all entries of G should be taken from {0,1,2}.
result	Output of psd_fit_em, psd_fit_sqp, or psd_fit_vi.

Value

A value indicates the accuracy of the prediction.

Examples

```
G <- matrix(c(0,0,1, 0,2,1, 1,0,1, 0,1,0, 1,0,0), 3, 5)
result <- psd_fit_em(G, 2, 1e-5, 10)
psd_loglikelihood(G, result)
```

---

psd_loglikelihood_svi *Compute the loglikelihood of svi method in validation set*

Description

Compute the maximum loglikelihood function of the PSD model in validation set using svi method.

Usage

```
psd_loglikelihood_svi(G, K, P, maxiter = 2000)
```

Arguments

G	The I x J matrix of counts; all entries of G should be taken from {0,1,2}.
K	An integer 2 or greater giving the matrix rank.
P	The P in the output list of psd_fit_svi.
maxiter	The maximum number of iterations in validation set.

Value

A value indicates the accuracy of the prediction in validation set.

Examples

```
# Refer to Articles in AwesomePackage.
```

psd_simulation *Simulate data of PSD model*

Description

Simulate gene data of PSD model including P, F, G.

Usage

```
psd_simulation(
  I,
  J,
  K,
  type.id = c("A", "B"),
  parm_alpha = 0.1,
  parm_sd = 2,
  parm_F = NULL,
  data = data_HGDP
)
```

Arguments

I	The number of individuals to simulate.
J	The number of SNPs to simulate.
K	The number of populations to simulate.
type.id	Choose type. Should be one of "A", "B".
parm_alpha	A parameter of the normal distribution in the simulation of the first type of P.
parm_sd	A parameter of the normal distribution in the simulation of the second type of P.
parm_F	A parameter of the beta distribution in the simulation of F. It has the same length as K.
data	The data needed to be provided in order to collect the suballele frequencies of real SNPs during the simulation of F.

Value

A List with the following parameters:

G The I x J simulated matrix of counts.

P The I x K simulated population scale matrix of the individuals.

F The K x J simulated gene scale matrix of the populations.

Examples

```
data_simuA <- psd_simulation(60, 250, 3, type.id = "A", parm_F = c(0.1, 0.05, 0.01))
plot_structure(data_simuA$P, pops = rep(3:1))
data_simuB <- psd_simulation(100, 500, 5, type.id = "B")
plot_structure(data_simuB$P, pops = rep(5:1))
```

Index