Package 'crandep'

January 19, 2024

Title Network Analysis of Dependencies of CRAN Packages

Version 0.3.5

Description The dependencies of CRAN packages can be analysed in a network fashion. For each package we can obtain the packages that it depends, imports, suggests, etc. By iterating this procedure over a number of packages, we can build, visualise, and analyse the dependency network, enabling us to have a bird's-eye view of the CRAN ecosystem. One aspect of interest is the number of reverse dependencies of the packages, or equivalently the in-degree distribution of the dependency network. This can be fitted by the power law and/or an extreme value mixture distribution <arXiv:2008.03073>, of which functions are provided.

Depends R (>= 3.4)

License GPL (>= 2)

URL https://github.com/clement-lee/crandep

BugReports https://github.com/clement-lee/crandep/issues

Encoding UTF-8

LazyData true

Imports xml2, rvest, stringr, dplyr, igraph, Rcpp, stats

Suggests ggplot2, tibble, visNetwork, knitr, rmarkdown

RoxygenNote 7.2.3

NeedsCompilation yes

SystemRequirements pandoc (>= 1.12.3) - http://pandoc.org

Author Clement Lee [aut, cre] (<https://orcid.org/0000-0003-1785-8671>)

Maintainer Clement Lee <clement.lee.tm@outlook.com>

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

Repository CRAN

Date/Publication 2024-01-19 12:40:11 UTC

R topics documented:

`chi_citations` *Citation network of CHI papers*

Description

A dataset containing the citations of conference papers of the ACM Conference on Human Factors in Computing Systems (CHI) from 1981 to 2019, obtained from the ACM digital library. The resulting citation network can be compared to the dependencies network of CRAN packages, in terms of network-related characteristics, such as degree distribution and sparsity.

Usage

```
chi_citations
```

Format

A data from with21951 rows and 4 variables:

from the unique identifier (in the digital library) of the paper that cites other papers

to the unique identifier of the paper that is being cited

year_from the publication year of the citing paper

year_to the publication year of the cited paper

Source

https://dl.acm.org/conference/chi

See Also

cran_dependencies

---

cran_dependencies *Dependencies of CRAN packages*

Description

A dataset containing the dependencies of various types (Imports, Depends, Suggests, LinkingTo, and their reverse counterparts) of more than 14600 packages available on CRAN as of 2020-05-09.

Usage

```
cran_dependencies
```

Format

A data frame with 211408 rows and 4 variables:

from the name of the package that introduced the dependencies

to the name of the package that the dependency is directed towards

type the type of dependency, which can take the follow values (all in lowercase): "depends", "imports", "linking to", "suggests"

reverse a boolean representing whether the dependency is a reverse one (TRUE) or a forward one (FALSE)

Source

The CRAN pages of all the packages available on https://cran.r-project.org

See Also

chi_citations

df_to_graph *Construct the giant component of the network from two data frames*

Description

Construct the giant component of the network from two data frames

Usage

```
df_to_graph(edgelist, nodelist = NULL, gc = TRUE)
```

Arguments

edgelist	A data frame with (at least) two columns: from and to
nodelist	NULL, or a data frame with (at least) one column: name, that contains the nodes to include
gc	Boolean, if 'TRUE' (default) then the giant component is extracted, if 'FALSE' then the whole graph is returned

Value

An igraph object & a connected graph if gc is 'TRUE'

Examples

```
from <- c("1", "2", "4")
to <- c("2", "3", "5")
edges <- data.frame(from = from, to = to, stringsAsFactors = FALSE)
nodes <- data.frame(name = c("1", "2", "3", "4", "5"), stringsAsFactors = FALSE)
df_to_graph(edges, nodes)
```

dmix2 *Probability mass function (PMF) of 2-component discrete extreme value mixture distribution*

Description

dmix2 returns the PMF at x for the 2-component discrete extreme value mixture distribution. The components below and above the threshold u are the (truncated) Zipf-polylog(alpha,theta) and the generalised Pareto(shape, sigma) distributions, respectively.

Usage

```
dmix2(x, u, alpha, theta, shape, sigma, phiu)
```

Arguments

x	Vector of positive integers
u	Positive integer representing the threshold
alpha	Real number, first parameter of the Zipf-polylog component
theta	Real number in (0, 1], second parameter of the Zipf-polylog component
shape	Real number, shape parameter of the generalised Pareto component
sigma	Real number, scale parameter of the generalised Pareto component
phiu	Real number in (0, 1), exceedance rate of the threshold u

Value

A numeric vector of the same length as x

See Also

Smix2 for the corresponding survival function, dpol and dmix3 for the PMFs of the Zipf-polylog and 3-component discrete extreme value mixture distributions, respectively.

---

dmix3 — *Probability mass function (PMF) of 3-component discrete extreme value mixture distribution*

Description

dmix3 returns the PMF at x for the 3-component discrete extreme value mixture distribution. The component below v is the (truncated) Zipf-polylog(alpha1,theta1) distribution, between v & u the (truncated) Zipf-polylog(alpha2,theta2) distribution, and above u the generalised Pareto(shape, sigma) distribution.

Usage

```
dmix3(x, v, u, alpha1, theta1, alpha2, theta2, shape, sigma, phi1, phi2, phiu)
```

Arguments

x	Vector of positive integers
v	Positive integer representing the lower threshold
u	Positive integer representing the upper threshold
alpha1	Real number, first parameter of the Zipf-polylog component below v
theta1	Real number in (0, 1], second parameter of the Zipf-polylog component below v
alpha2	Real number, first parameter of the Zipf-polylog component between v & u
theta2	Real number in (0, 1], second parameter of the Zipf-polylog component between v & u

shape	Real number, shape parameter of the generalised Pareto component
sigma	Real number, scale parameter of the generalised Pareto component
phi1	Real number in (0, 1), proportion of values below v
phi2	Real number in (0, 1), proportion of values between v & u
phiu	Real number in (0, 1), exceedance rate of the threshold u

Value

A numeric vector of the same length as x

See Also

Smix3 for the corresponding survival function, dpol and dmix2 for the PMFs of the Zipf-polylog and 2-component discrete extreme value mixture distributions, respectively.

---

dpol *Probability mass function (PMF) of Zipf-polylog distribution*

Description

dpol returns the PMF at x for the Zipf-polylog distribution with parameters (alpha, theta). The distribution is reduced to the discrete power law when theta = 1.

Usage

```
dpol(x, alpha, theta, xmax = 100000L)
```

Arguments

x	Vector of positive integers
alpha	Real number greater than 1
theta	Real number in (0, 1]
xmax	Scalar (default 100000), positive integer limit for computing the normalising constant

Details

The PMF is proportional to x^(-alpha) * theta^x. It is normalised in order to be a proper PMF.

Value

A numeric vector of the same length as x

See Also

Spol for the corresponding survival function, dmix2 and dmix3 for the PMFs of the 2-component and 3-component discrete extreme value mixture distributions, respectively.

Examples

```
dpol(c(1,2,3,4,5), 1.2, 0.5)
```

`get_dep_all_packages` *Dependencies of all CRAN packages*

Description

`get_dep_all_packages` returns the data frame of dependencies of all packages currently available on CRAN.

Usage

```
get_dep_all_packages()
```

Details

Unlike `get_dep`, there is no boolean argument 'scrape', as it is much faster to obtain the dependencies of all packages via 'tools::CRAN_package_db()'.

Value

A data frame of dependencies of all CRAN packages

See Also

`get_dep` for multiple types of dependencies, and `get_graph_all_packages` for obtaining directly a network of dependencies as an igraph object

Examples

```
## Not run:
df.cran <- get_dep_all_packages()

## End(Not run)
```

get_dep_df *Multiple types of dependencies*

Description

get_dep returns a data frame of multiple types of dependencies of a package

Usage

```
get_dep_df(name, type, scrape = TRUE)

get_dep_all(name, type, scrape = TRUE)

get_dep(name, type, scrape = TRUE)
```

Arguments

name	String, name of the package
type	A character vector that contains one or more of the following dependency words: "Depends", "Imports", "LinkingTo", "Suggests", "Enhances", "Reverse depends", "Reverse imports", "Reverse linking to", "Reverse suggests", "Reverse enhances", up to letter case and space replaced by underscore. Alternatively, if 'type = "all"', all ten dependencies will be obtained.
scrape	Boolean. If 'TRUE' (default), the page of the package will be scraped. If 'FALSE', tools::CRAN_package_db() will be used. Whether the argument equals 'TRUE' or 'FALSE' should not affect the output, but only the time taken. Usually, the former is faster than the latter for a single package.

Value

A data frame of dependencies

See Also

get_dep_all_packages for the dependencies of all CRAN packages, and get_graph_all_packages for obtaining directly a network of dependencies as an igraph object

Examples

```
get_dep("dplyr", c("Imports", "Depends"))
get_dep("MASS", c("Suggests", "Depends", "Imports"), TRUE) # FALSE will give same result
```

get_graph_all_packages *Graph of dependencies of all CRAN packages*

Description

get_graph_all_packages returns an igraph object representing the network of one or more types of dependencies of all CRAN packages.

Usage

```
get_graph_all_packages(type, gc = TRUE)
```

Arguments

type	A character vector that contains one or more of the following dependency words: "Depends", "Imports", "LinkingTo", "Suggests", "Enhances", "Reverse depends", "Reverse imports", "Reverse linking to", "Reverse suggests", "Reverse enhances", up to letter case and space replaced by underscore. Alternatively, if 'types = "all"', all ten dependencies will be obtained.
gc	Boolean, if 'TRUE' (default) then the giant component is extracted, if 'FALSE' then the whole graph is returned

Value

An igraph object & a connected graph if gc is 'TRUE'

See Also

get_dep_all_packages for the dependencies of all CRAN packages in a data frame, and df_to_graph for constructing the giant component of the network from two data frames

Examples

```
## Not run:
g0.cran.depends <- get_graph_all_packages("depends")
g1.cran.imports <- get_graph_all_packages("reverse imports")

## End(Not run)
```

`mcmc_mix1` *Markov chain Monte Carlo for TZP-power-law mixture*

Description

`mcmc_mix1` returns the posterior samples of the parameters, for fitting the TZP-power-law mixture distribution. The samples are obtained using Markov chain Monte Carlo (MCMC).

Usage

```
mcmc_mix1(
  x,
  count,
  u_set,
  u,
  alpha1,
  theta1,
  alpha2,
  a_psiu,
  b_psiu,
  a_alpha1,
  b_alpha1,
  a_theta1,
  b_theta1,
  a_alpha2,
  b_alpha2,
  positive,
  iter,
  thin,
  burn,
  freq,
  xmax = 100000L
)
```

Arguments

`x`	Vector of the unique values (positive integers) of the data
`count`	Vector of the same length as x that contains the counts of each unique value in the full data, which is essentially rep(x, count)
`u_set`	Positive integer vector of the values u will be sampled from
`u`	Positive integer, initial value of the threshold
`alpha1`	Real number, initial value of the parameter
`theta1`	Real number in (0, 1], initial value of the parameter
`alpha2`	Real number greater than 1, initial value of the parameter

`a_psiu, b_psiu, a_alpha1, b_alpha1, a_theta1, b_theta1, a_alpha2, b_alpha2`
: Scalars, real numbers representing the hyperparameters of the prior distributions for the respective parameters. See details for the specification of the priors.

`positive`
: Boolean, is alpha positive (TRUE) or unbounded (FALSE)?

`iter`
: Positive integer representing the length of the MCMC output

`thin`
: Positive integer representing the thinning in the MCMC

`burn`
: Non-negative integer representing the burn-in of the MCMC

`freq`
: Positive integer representing the frequency of the sampled values being printed

`xmax`
: Scalar (default 100000), positive integer limit for computing the normalising constant

Details

In the MCMC, a componentwise Metropolis-Hastings algorithm is used. The threshold u is treated as a parameter and therefore sampled. The hyperparameters are used in the following priors: u is such that the implied unique exceedance probability psiu ~ Uniform(a_psi, b_psi); alpha1 ~ Normal(mean = a_alpha1, sd = b_alpha1); theta1 ~ Beta(a_theta1, b_theta1); alpha2 ~ Normal(mean = a_alpha2, sd = b_alpha2)

Value

A list: $pars is a data frame of iter rows of the MCMC samples, $fitted is a data frame of length(x) rows with the fitted values, amongst other quantities related to the MCMC

See Also

`mcmc_pol`, `mcmc_mix2` and `mcmc_mix3` for MCMC for the Zipf-polylog, and 2-component and 3-component discrete extreme value mixture distributions, respectively.

`mcmc_mix1_wrapper` *Wrapper of mcmc_mix1*

Description

Wrapper of mcmc_mix1

Usage

```
mcmc_mix1_wrapper(
  df,
  seed,
  a_psiu = 0.1,
  b_psiu = 0.9,
  m_alpha1 = 0,
  s_alpha1 = 10,
  a_theta1 = 1,
```

```
  b_theta1 = 1,
  m_alpha2 = 0,
  s_alpha2 = 10,
  positive = FALSE,
  iter = 20000L,
  thin = 1L,
  burn = 10000L,
  freq = 100L,
  xmax = 1e+05
)
```

Arguments

df	A data frame with at least two columns, x & count
seed	Integer for set.seed
a_psiu, b_psiu, m_alpha1, s_alpha1, a_theta1, b_theta1, m_alpha2, s_alpha2	Scalars, real numbers representing the hyperparameters of the prior distributions for the respective parameters. See details for the specification of the priors.
positive	Boolean, is alpha1 positive (TRUE) or unbounded (FALSE)?
iter	Positive integer representing the length of the MCMC output
thin	Positive integer representing the thinning in the MCMC
burn	Non-negative integer representing the burn-in of the MCMC
freq	Positive integer representing the frequency of the sampled values being printed
xmax	Scalar (default 100000), positive integer limit for computing the normalising constant

Value

A list returned by mcmc_mix1

---

mcmc_mix2 *Markov chain Monte Carlo for 2-component discrete extreme value mixture distribution*

Description

mcmc_mix2 returns the posterior samples of the parameters, for fitting the 2-component discrete extreme value mixture distribution. The samples are obtained using Markov chain Monte Carlo (MCMC).

Usage

```
mcmc_mix2(
  x,
  count,
  u_set,
  u,
  alpha,
  theta,
  shape,
  sigma,
  a_psiu,
  b_psiu,
  a_alpha,
  b_alpha,
  a_theta,
  b_theta,
  m_shape,
  s_shape,
  a_sigma,
  b_sigma,
  positive,
  a_pseudo,
  b_pseudo,
  pr_power,
  iter,
  thin,
  burn,
  freq,
  invt
)
```

Arguments

x	Vector of the unique values (positive integers) of the data
count	Vector of the same length as x that contains the counts of each unique value in the full data, which is essentially rep(x, count)
u_set	Positive integer vector of the values u will be sampled from
u	Positive integer, initial value of the threshold
alpha	Real number greater than 1, initial value of the parameter
theta	Real number in (0, 1], initial value of the parameter
shape	Real number, initial value of the parameter
sigma	Positive real number, initial value of the parameter
a_psiu, b_psiu, a_alpha, b_alpha, a_theta, b_theta, m_shape, s_shape, a_sigma, b_sigma	Scalars, real numbers representing the hyperparameters of the prior distributions for the respective parameters. See details for the specification of the priors.
positive	Boolean, is alpha positive (TRUE) or unbounded (FALSE)?

a_pseudo	Positive real number, first parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
b_pseudo	Positive real number, second parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
pr_power	Real number in [0, 1], prior probability of the discrete power law (below u)
iter	Positive integer representing the length of the MCMC output
thin	Positive integer representing the thinning in the MCMC
burn	Non-negative integer representing the burn-in of the MCMC
freq	Positive integer representing the frequency of the sampled values being printed
invt	Vector of the inverse temperatures for Metropolis-coupled MCMC; default c(1.0) i.e. no Metropolis-coupling

Details

In the MCMC, a componentwise Metropolis-Hastings algorithm is used. The threshold u is treated as a parameter and therefore sampled. The hyperparameters are used in the following priors: u is such that the implied unique exceedance probability psiu ~ Uniform(a_psi, b_psi); alpha ~ Normal(mean = a_alpha, sd = b_alpha); theta ~ Beta(a_theta, b_theta); shape ~ Normal(mean = m_shape, sd = s_shape); sigma ~ Gamma(a_sigma, scale = b_sigma). If pr_power = 1.0, the discrete power law (below u) is assumed, and the samples of theta will be all 1.0. If pr_power is in (0.0, 1.0), model selection between the polylog distribution and the discrete power law will be performed within the MCMC.

Value

A list: $pars is a data frame of iter rows of the MCMC samples, $fitted is a data frame of length(x) rows with the fitted values, amongst other quantities related to the MCMC

See Also

mcmc_pol and mcmc_mix3 for MCMC for the Zipf-polylog and 3-component discrete extreme value mixture distributions, respectively.

---

mcmc_mix2_wrapper *Wrapper of mcmc_mix2*

Description

Wrapper of mcmc_mix2

Usage

```
mcmc_mix2_wrapper(
  df,
  seed,
  a_psiu = 0.001,
  b_psiu = 0.9,
  m_alpha = 0,
  s_alpha = 10,
  a_theta = 1,
  b_theta = 1,
  m_shape = 0,
  s_shape = 10,
  a_sigma = 1,
  b_sigma = 0.01,
  a_pseudo = 10,
  b_pseudo = 1,
  pr_power = 0.5,
  positive = FALSE,
  iter = 20000L,
  thin = 20L,
  burn = 100000L,
  freq = 1000L,
  mc3 = FALSE,
  invts = 0.001^((0:8)/8)
)
```

Arguments

df	A data frame with at least two columns, x & count
seed	Integer for set.seed
a_psiu, b_psiu, m_alpha, s_alpha, a_theta, b_theta, m_shape, s_shape, a_sigma, b_sigma	Scalars, real numbers representing the hyperparameters of the prior distributions for the respective parameters. See details for the specification of the priors.
a_pseudo	Positive real number, first parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
b_pseudo	Positive real number, second parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
pr_power	Real number in [0, 1], prior probability of the discrete power law (below u)
positive	Boolean, is alpha positive (TRUE) or unbounded (FALSE)?
iter	Positive integer representing the length of the MCMC output
thin	Positive integer representing the thinning in the MCMC
burn	Non-negative integer representing the burn-in of the MCMC
freq	Positive integer representing the frequency of the sampled values being printed
mc3	Boolean, is Metropolis-coupled MCMC to be used?
invts	Vector of the inverse temperatures for Metropolis-coupled MCMC; ignored if mc3 = FALSE

Value

A list returned by `mcmc_mix2`

`mcmc_mix3` *Markov chain Monte Carlo for 3-component discrete extreme value mixture distribution*

Description

`mcmc_mix3` returns the posterior samples of the parameters, for fitting the 3-component discrete extreme value mixture distribution. The samples are obtained using Markov chain Monte Carlo (MCMC).

Usage

```
mcmc_mix3(
  x,
  count,
  v_set,
  u_set,
  v,
  u,
  alpha1,
  theta1,
  alpha2,
  theta2,
  shape,
  sigma,
  a_psi1,
  a_psi2,
  a_psiu,
  b_psiu,
  a_alpha1,
  b_alpha1,
  a_theta1,
  b_theta1,
  a_alpha2,
  b_alpha2,
  a_theta2,
  b_theta2,
  m_shape,
  s_shape,
  a_sigma,
  b_sigma,
  powerlaw1,
  positive1,
  positive2,
```

```
  a_pseudo,
  b_pseudo,
  pr_power2,
  iter,
  thin,
  burn,
  freq,
  invt
)
```

Arguments

x	Vector of the unique values (positive integers) of the data
count	Vector of the same length as x that contains the counts of each unique value in the full data, which is essentially rep(x, count)
v_set	Positive integer vector of the values v will be sampled from
u_set	Positive integer vector of the values u will be sampled from
v	Positive integer, initial value of the lower threshold
u	Positive integer, initial value of the upper threshold
alpha1	Real number greater than 1, initial value of the parameter
theta1	Real number in (0, 1], initial value of the parameter
alpha2	Real number greater than 1, initial value of the parameter
theta2	Real number in (0, 1], initial value of the parameter
shape	Real number, initial value of the parameter
sigma	Positive real number, initial value of the parameter
a_psi1, a_psi2, a_psiu, b_psiu, a_alpha1, b_alpha1, a_theta1, b_theta1, a_alpha2, b_alpha2, a_theta2, b_t	Scalars, real numbers representing the hyperparameters of the prior distributions for the respective parameters. See details for the specification of the priors.
powerlaw1	Boolean, is the discrete power law assumed for below v?
positive1	Boolean, is alpha1 positive (TRUE) or unbounded (FALSE)?
positive2	Boolean, is alpha2 positive (TRUE) or unbounded (FALSE)?
a_pseudo	Positive real number, first parameter of the pseudoprior beta distribution for theta2 in model selection; ignored if pr_power2 = 1.0
b_pseudo	Positive real number, second parameter of the pseudoprior beta distribution for theta2 in model selection; ignored if pr_power2 = 1.0
pr_power2	Real number in [0, 1], prior probability of the discrete power law (between v and u)
iter	Positive integer representing the length of the MCMC output
thin	Positive integer representing the thinning in the MCMC
burn	Non-negative integer representing the burn-in of the MCMC
freq	Positive integer representing the frequency of the sampled values being printed
invt	Vector of the inverse temperatures for Metropolis-coupled MCMC; default c(1.0) i.e. no Metropolis-coupling

Details

In the MCMC, a componentwise Metropolis-Hastings algorithm is used. The thresholds v and u are treated as parameters and therefore sampled. The hyperparameters are used in the following priors: psi1 / (1.0 - psiu) ~ Beta(a_psi1, a_psi2); u is such that the implied unique exceedance probability psiu ~ Uniform(a_psi, b_psi); alpha1 ~ Normal(mean = a_alpha1, sd = b_alpha1); theta1 ~ Beta(a_theta1, b_theta1); alpha2 ~ Normal(mean = a_alpha2, sd = b_alpha2); theta2 ~ Beta(a_theta2, b_theta2); shape ~ Normal(mean = m_shape, sd = s_shape); sigma ~ Gamma(a_sigma, scale = b_sigma). If pr_power2 = 1.0, the discrete power law (between v and u) is assumed, and the samples of theta2 will be all 1.0. If pr_power2 is in (0.0, 1.0), model selection between the polylog distribution and the discrete power law will be performed within the MCMC.

Value

A list: $pars is a data frame of iter rows of the MCMC samples, $fitted is a data frame of length(x) rows with the fitted values, amongst other quantities related to the MCMC

See Also

mcmc_pol and mcmc_mix2 for MCMC for the Zipf-polylog and 2-component discrete extreme value mixture distributions, respectively.

---

mcmc_mix3_wrapper *Wrapper of mcmc_mix3*

Description

Wrapper of mcmc_mix3

Usage

```
mcmc_mix3_wrapper(
  df,
  seed,
  a_psi1 = 1,
  a_psi2 = 1,
  a_psiu = 0.001,
  b_psiu = 0.9,
  m_alpha = 0,
  s_alpha = 10,
  a_theta = 1,
  b_theta = 1,
  m_shape = 0,
  s_shape = 10,
  a_sigma = 1,
  b_sigma = 0.01,
  a_pseudo = 10,
  b_pseudo = 1,
```

```
  pr_power2 = 0.5,
  powerlaw1 = FALSE,
  positive1 = FALSE,
  positive2 = TRUE,
  iter = 20000L,
  thin = 20L,
  burn = 100000L,
  freq = 1000L,
  mc3 = FALSE,
  invts = 0.001^((0:8)/8)
)
```

Arguments

df	A data frame with at least two columns, x & count
seed	Integer for set.seed
a_psi1, a_psi2, a_psiu, b_psiu, m_alpha, s_alpha, a_theta, b_theta, m_shape, s_shape, a_sigma, b_sigma	Scalars, real numbers representing the hyperparameters of the prior distributions for the respective parameters. See details for the specification of the priors.
a_pseudo	Positive real number, first parameter of the pseudoprior beta distribution for theta2 in model selection; ignored if pr_power2 = 1.0
b_pseudo	Positive real number, second parameter of the pseudoprior beta distribution for theta2 in model selection; ignored if pr_power2 = 1.0
pr_power2	Real number in [0, 1], prior probability of the discrete power law (between v and u)
powerlaw1	Boolean, is the discrete power law assumed for below v?
positive1	Boolean, is alpha1 positive (TRUE) or unbounded (FALSE)?
positive2	Boolean, is alpha2 positive (TRUE) or unbounded (FALSE)?
iter	Positive integer representing the length of the MCMC output
thin	Positive integer representing the thinning in the MCMC
burn	Non-negative integer representing the burn-in of the MCMC
freq	Positive integer representing the frequency of the sampled values being printed
mc3	Boolean, is Metropolis-coupled MCMC to be used?
invts	Vector of the inverse temperatures for Metropolis-coupled MCMC; ignored if mc3 = FALSE

Value

A list returned by mcmc_mix3

mcmc_pol *Markov chain Monte Carlo for Zipf-polylog distribution*

Description

mcmc_pol returns the samples from the posterior of alpha and theta, for fitting the Zipf-polylog distribution to the data x. The samples are obtained using Markov chain Monte Carlo (MCMC). In the MCMC, a Metropolis-Hastings algorithm is used.

Usage

```
mcmc_pol(
  x,
  count,
  alpha,
  theta,
  a_alpha,
  b_alpha,
  a_theta,
  b_theta,
  a_pseudo,
  b_pseudo,
  pr_power,
  iter,
  thin,
  burn,
  freq,
  invt,
  xmax = 100000L
)
```

Arguments

x	Vector of the unique values (positive integers) of the data
count	Vector of the same length as x that contains the counts of each unique value in the full data, which is essentially rep(x, count)
alpha	Real number greater than 1, initial value of the parameter
theta	Real number in (0, 1], initial value of the parameter
a_alpha	Real number, mean of the prior normal distribution for alpha
b_alpha	Positive real number, standard deviation of the prior normal distribution for alpha
a_theta	Positive real number, first parameter of the prior beta distribution for theta; ignored if pr_power = 1.0
b_theta	Positive real number, second parameter of the prior beta distribution for theta; ignored if pr_power = 1.0

a_pseudo	Positive real number, first parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
b_pseudo	Positive real number, second parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
pr_power	Real number in [0, 1], prior probability of the discrete power law
iter	Positive integer representing the length of the MCMC output
thin	Positive integer representing the thinning in the MCMC
burn	Non-negative integer representing the burn-in of the MCMC
freq	Positive integer representing the frequency of the sampled values being printed
invt	Vector of the inverse temperatures for Metropolis-coupled MCMC; default c(1.0) i.e. no Metropolis-coupling
xmax	Scalar (default 100000), positive integer limit for computing the normalising constant

Value

A list: $pars is a data frame of iter rows of the MCMC samples, $fitted is a data frame of length(x) rows with the fitted values, amongst other quantities related to the MCMC

See Also

mcmc_mix2 and mcmc_mix3 for MCMC for the 2-component and 3-component discrete extreme value mixture distributions, respectively.

---

mcmc_pol_wrapper *Wrapper of mcmc_pol*

Description

Wrapper of mcmc_pol

Usage

```
mcmc_pol_wrapper(
  df,
  seed,
  alpha_init = 1.5,
  theta_init = 0.5,
  m_alpha = 0,
  s_alpha = 10,
  a_theta = 1,
  b_theta = 1,
  a_pseudo = 10,
  b_pseudo = 1,
  pr_power = 0.5,
```

```
  iter = 20000L,
  thin = 20L,
  burn = 100000L,
  freq = 1000L,
  mc3 = FALSE,
  invts = 0.001^((0:8)/8)
)
```

Arguments

df	A data frame with at least two columns, x & count
seed	Integer for set.seed
alpha_init	Real number greater than 1, initial value of the parameter
theta_init	Real number in (0, 1], initial value of the parameter
m_alpha	Real number, mean of the prior normal distribution for alpha
s_alpha	Positive real number, standard deviation of the prior normal distribution for alpha
a_theta	Positive real number, first parameter of the prior beta distribution for theta; ignored if pr_power = 1.0
b_theta	Positive real number, second parameter of the prior beta distribution for theta; ignored if pr_power = 1.0
a_pseudo	Positive real number, first parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
b_pseudo	Positive real number, second parameter of the pseudoprior beta distribution for theta in model selection; ignored if pr_power = 1.0
pr_power	Real number in [0, 1], prior probability of the discrete power law
iter	Positive integer representing the length of the MCMC output
thin	Positive integer representing the thinning in the MCMC
burn	Non-negative integer representing the burn-in of the MCMC
freq	Positive integer representing the frequency of the sampled values being printed
mc3	Boolean, is Metropolis-coupled MCMC to be used?
invts	Vector of the inverse temperatures for Metropolis-coupled MCMC

Value

A list returned by mcmc_pol

obtain_u_set_mix1 *Obtain set of thresholds with high posterior density for the TZP-power-law mixture model*

Description

obtain_u_set_mix1 computes the profile posterior density of the threshold u, and subsets the thresholds (and other parameter values) with high profile values i.e. within a certain value from the maximum posterior density. The set of u can then be used for mcmc_mix1.

Usage

```
obtain_u_set_mix1(
  df,
  positive = FALSE,
  u_max = 2000L,
  log_diff_max = 11,
  alpha1_init = 0.01,
  theta1_init = exp(-1),
  alpha2_init = 2,
  a_psiu = 0.1,
  b_psiu = 0.9,
  m_alpha1 = 0,
  s_alpha1 = 10,
  a_theta1 = 1,
  b_theta1 = 1,
  m_alpha2 = 0,
  s_alpha2 = 10,
  xmax = 1e+05
)
```

Arguments

df	A data frame with at least two columns, x & count
positive	Boolean, is alpha1 positive (TRUE) or unbounded (FALSE, default)?
u_max	Positive integer for the maximum threshold
log_diff_max	Positive real number, the value such that thresholds with profile posterior density not less than the maximum posterior density - log_diff_max will be kept
alpha1_init	Scalar, initial value of alpha1
theta1_init	Scalar, initial value of theta1
alpha2_init	Scalar, initial value of alpha2
a_psiu, b_psiu, m_alpha1, s_alpha1, a_theta1, b_theta1, m_alpha2, s_alpha2	Scalars, hyperparameters of the priors for the parameters
xmax	Scalar (default 100000), positive integer limit for computing the normalising constant

Value

A list: `u_set` is the vector of thresholds with high posterior density, `init` is the data frame with the maximum profile posterior density and associated parameter values, `profile` is the data frame with all thresholds with high posterior density and associated parameter values, `scalars` is the data frame with all arguments (except df)

See Also

`mcmc_mix1_wrapper` that wraps `obtain_u_set_mix1` and `mcmc_mix1`, `obtain_u_set_mix2` for the equivalent function for the 2-component mixture model

---

`obtain_u_set_mix2` *Obtain set of thresholds with high posterior density for the 2-component mixture model*

Description

`obtain_u_set_mix2` computes the profile posterior density of the threshold u, and subsets the thresholds (and other parameter values) with high profile values i.e. within a certain value from the maximum posterior density. The set of u can then be used for `mcmc_mix2`.

Usage

```
obtain_u_set_mix2(
  df,
  powerlaw = FALSE,
  positive = FALSE,
  u_max = 2000L,
  log_diff_max = 11,
  alpha_init = 0.01,
  theta_init = exp(-1),
  shape_init = 0.1,
  sigma_init = 1,
  a_psiu = 0.001,
  b_psiu = 0.9,
  m_alpha = 0,
  s_alpha = 10,
  a_theta = 1,
  b_theta = 1,
  m_shape = 0,
  s_shape = 10,
  a_sigma = 1,
  b_sigma = 0.01
)
```

Arguments

df	A data frame with at least two columns, x & count
powerlaw	Boolean, is the power law (TRUE) or polylogarithm (FALSE, default) assumed?
positive	Boolean, is alpha positive (TRUE) or unbounded (FALSE, default)?
u_max	Positive integer for the maximum threshold
log_diff_max	Positive real number, the value such that thresholds with profile posterior density not less than the maximum posterior density - log_diff_max will be kept
alpha_init	Scalar, initial value of alpha
theta_init	Scalar, initial value of theta
shape_init	Scalar, initial value of shape parameter
sigma_init	Scalar, initial value of sigma
a_psiu, b_psiu, m_alpha, s_alpha, a_theta, b_theta, m_shape, s_shape, a_sigma, b_sigma	Scalars, hyperparameters of the priors for the parameters

Value

A list: u_set is the vector of thresholds with high posterior density, init is the data frame with the maximum profile posterior density and associated parameter values, profile is the data frame with all thresholds with high posterior density and associated parameter values, scalars is the data frame with all arguments (except df)

See Also

mcmc_mix2_wrapper that wraps obtain_u_set_mix2 and mcmc_mix2, obtain_u_set_mix1 for the equivalent function for the TZP-power-law mixture model

---

obtain_u_set_mix3 *Obtain set of thresholds with high posterior density for the 3-component mixture model*

Description

obtain_u_set_mix3 computes the profile posterior density of the thresholds v & u, and subsets the thresholds (and other parameter values) with high profile values i.e. within a certain value from the maximum posterior density. The sets of v & u can then be used for mcmc_mix3.

Usage

```
obtain_u_set_mix3(
  df,
  powerlaw1 = FALSE,
  powerlaw2 = FALSE,
  positive1 = FALSE,
  positive2 = TRUE,
```

```
  log_diff_max = 11,
  v_max = 100L,
  u_max = 2000L,
  alpha_init = 0.01,
  theta_init = exp(-1),
  shape_init = 1,
  sigma_init = 1,
  a_psi1 = 1,
  a_psi2 = 1,
  a_psiu = 0.001,
  b_psiu = 0.9,
  m_alpha = 0,
  s_alpha = 10,
  a_theta = 1,
  b_theta = 1,
  m_shape = 0,
  s_shape = 10,
  a_sigma = 1,
  b_sigma = 0.01
)
```

Arguments

`df`	A data frame with at least two columns, degree & count
`powerlaw1`	Boolean, is the power law (TRUE) or polylogarithm (FALSE, default) assumed for the left tail?
`powerlaw2`	Boolean, is the power law (TRUE) or polylogarithm (FALSE, default) assumed for the middle bulk?
`positive1`	Boolean, is alpha positive (TRUE) or unbounded (FALSE, default) for the left tail?
`positive2`	Boolean, is alpha positive (TRUE) or unbounded (FALSE, default) for the middle bulk?
`log_diff_max`	Positive real number, the value such that thresholds with profile posterior density not less than the maximum posterior density - `log_diff_max` will be kept
`v_max`	Positive integer for the maximum lower threshold
`u_max`	Positive integer for the maximum upper threshold
`alpha_init`	Scalar, initial value of alpha
`theta_init`	Scalar, initial value of theta
`shape_init`	Scalar, initial value of shape parameter
`sigma_init`	Scalar, initial value of sigma
`a_psi1, a_psi2, a_psiu, b_psiu, m_alpha, s_alpha, a_theta, b_theta, m_shape, s_shape, a_sigma, b_sigma`	Scalars, hyperparameters of the priors for the parameters

Value

A list: `v_set` is the vector of lower thresholds with high posterior density, `u_set` is the vector of upper thresholds with high posterior density, `init` is the data frame with the maximum profile posterior density and associated parameter values, `profile` is the data frame with all thresholds with high posterior density and associated parameter values, `scalars` is the data frame with all arguments (except df)

See Also

`mcmc_mix3_wrapper` that wraps `obtain_u_set_mix3` and `mcmc_mix3`

`Smix2` *Survival function of 2-component discrete extreme value mixture distribution*

Description

`Smix2` returns the survival function at x for the 2-component discrete extreme value mixture distribution. The components below and above the threshold u are the (truncated) Zipf-polylog(alpha,theta) and the generalised Pareto(shape, sigma) distributions, respectively.

Usage

```
Smix2(x, u, alpha, theta, shape, sigma, phiu)
```

Arguments

`x`	Vector of positive integers
`u`	Positive integer representing the threshold
`alpha`	Real number, first parameter of the Zipf-polylog component
`theta`	Real number in (0, 1], second parameter of the Zipf-polylog component
`shape`	Real number, shape parameter of the generalised Pareto component
`sigma`	Real number, scale parameter of the generalised Pareto component
`phiu`	Real number in (0, 1), exceedance rate of the threshold u

Value

A numeric vector of the same length as x

See Also

`dmix2` for the corresponding probability mass function, `Spol` and `Smix3` for the survival functions of the Zipf-polylog and 3-component discrete extreme value mixture distributions, respectively.

`Smix3` *Survival function of 3-component discrete extreme value mixture distribution*

Description

`Smix3` returns the survival function at x for the 3-component discrete extreme value mixture distribution. The component below v is the (truncated) Zipf-polylog(alpha1,theta1) distribution, between v & u the (truncated) Zipf-polylog(alpha2,theta2) distribution, and above u the generalised Pareto(shape, sigma) distribution.

Usage

```
Smix3(x, v, u, alpha1, theta1, alpha2, theta2, shape, sigma, phi1, phi2, phiu)
```

Arguments

`x`	Vector of positive integers
`v`	Positive integer representing the lower threshold
`u`	Positive integer representing the upper threshold
`alpha1`	Real number, first parameter of the Zipf-polylog component below v
`theta1`	Real number in (0, 1], second parameter of the Zipf-polylog component below v
`alpha2`	Real number, first parameter of the Zipf-polylog component between v & u
`theta2`	Real number in (0, 1], second parameter of the Zipf-polylog component between v & u
`shape`	Real number, shape parameter of the generalised Pareto component
`sigma`	Real number, scale parameter of the generalised Pareto component
`phi1`	Real number in (0, 1), proportion of values below v
`phi2`	Real number in (0, 1), proportion of values between v & u
`phiu`	Real number in (0, 1), exceedance rate of the threshold u

Value

A numeric vector of the same length as x

See Also

`dmix3` for the corresponding probability mass function, `Spol` and `Smix2` for the survival functions of the Zipf-polylog and 2-component discrete extreme value mixture distributions, respectively.

Spol *Survival function of Zipf-polylog distribution*

Description

Spol returns the survival function at x for the Zipf-polylog distribution with parameters (alpha, theta). The distribution is reduced to the discrete power law when theta = 1.

Usage

```
Spol(x, alpha, theta, xmax = 100000L)
```

Arguments

x	Vector of positive integers
alpha	Real number greater than 1
theta	Real number in (0, 1]
xmax	Scalar (default 100000), positive integer limit for computing the normalising constant

Value

A numeric vector of the same length as x

See Also

dpol for the corresponding probability mass function, Smix2 and Smix3 for the survival functions of the 2-component and 3-component discrete extreme value mixture distributions, respectively.

Examples

```
Spol(c(1,2,3,4,5), 1.2, 0.5)
```

topo_sort_kahn *Return a sorted vector of nodes id*

Description

Return a sorted vector of nodes id

Usage

```
topo_sort_kahn(g, random = FALSE)
```

Arguments

g	An igraph object of a DAG
random	Boolean, whether the order of selected nodes is randomised in the process

Value

A data frame with two columns: "id" is the names of nodes in g, and "id_num" is the topological ordering

Examples

```
df0 <- data.frame(from = c("a", "b"), to = c("b", "c"), stringsAsFactors = FALSE)
g0 <- igraph::graph_from_data_frame(df0, directed = TRUE)
topo_sort_kahn(g0)
```

Index