Package ‘hdpx’

July 28, 2022

Title Hierarchical Dirichlet process for categorical count data, expanded

Version 1.0.5

Encoding UTF-8

Depends R(>= 3.5.0)

biocViews

LazyData true

Language en-US

Description Model categorical count data with hierarchical Dirichlet process (HDP) mixture models. If you want to do mutational signature discovery, you probably want to use package mSigHdp, which calls this package. This package is only supported on Linux systems. Includes functions to initialise an HDP model with a custom tree structure, perform Gibbs sampling of the posterior distribution, and analyse the output. The underlying mathematical theory is described by Teh et al. (Hierarchical Dirichlet Processes, Journal of the American Statistical Association, 2006, 101:476). This R package is based on code forked from Nicola Roberts, https://github.com/nicolaroberts/hdp. Roberts adapted the R code from open source MATLAB code and incorporated C code from Yee Whye Teh (MATLAB and C code available at http://www.stats.ox.ac.uk/~teh/research/npbayes/npbayes-r21.tgz). Subsequent changes by Rozen and Liu are mostly confined to the R code. These include (1) corrections to garbage collection in the interface to the C code and (2) a new function for computing unsigned Stirling numbers of the first kind (3) a complete re-working of the process by which ``raw clusters'' sampled in posterior chains are combined into ``components'' (sets of mutations generated by one mutational process) (4) new functions for plotting to visualize and evaluate components extracted by the new procedures. There are also revised suggestions for burnin procedures and for setting hyperparameters for the concentration parameters; see https://github.com/steverozen/mSigHdp.

License file LICENSE

URL https://github.com/steverozen/hdpx

BugReports https://github.com/steverozen/hdpx/issues

Imports lsa,
cluster,
methods,
coda,
dendextend,
Matrix,
ggplot2,
ICAMS,
reshape2,
stats,
beeswarm,
parallelDist

Suggests testthat,
RColorBrewer,
knitr,
rmarkdown,
BiocStyle,
devtools

Collate 'aaa-classes-input.R'
'aaa-classes-output.R'
'aaa-generics-input.R'
'aaa-generics-output.R'
'cull_posterior_samples.R'
'diagnostic_in_extraction.R'
'dp_activate.R'
'dp_freeze.R'
'extract_ccc_from_hdp.R'
'extract_components.R'
'globals.R'
'hdp.R'
'hdp_addconparam.R'
'hdp_adddp.R'
'hdp_burnin.R'
'hdp_getstate.R'
'hdp_init.R'
'hdp_multi_chain.R'
'hdp_posterior.R'
'hdp_posterior_sample.R'
'hdp_prior_init.R'
'hdp_quick_init.R'
'hdp_setdata.R'
'interpret_components.R'
'iterate.R'
'new_diagnostic_plot_functions.R'
'plot_chain.R'
'plot_components.R'
'RcppExports.R'
'TestScaffold.R'
'utilities.R'
'utilities_nr3.R'
'xmake.s.R'
'zzz.R'

LinkingTo Rcpp,RcppArmadillo

RoxygenNote 7.2.1

R topics documented:

`comp_categ_counts` *Get sample vs category counts for each component*

Description

Get sample vs category counts for each component

Usage

```
comp_categ_counts(x)
```

Arguments

`x`	hdpSampleChain or hdpSampleMulti

Value

List of matrices (one for each component) counting the sample-category data assignment across all DP nodes. Number of rows is the number of posterior samples, and number of columns is the number of data categories.

`comp_categ_distn` *Get mean distribution over data categories for each component*

Description

Get mean distribution over data categories for each component

Usage

```
comp_categ_distn(x)
```

Arguments

`x`	hdpSampleChain or hdpSampleMulti

Value

List with elements "mean" and "cred.int", containing matrices with the mean (and lower/upper 95% credibility interval) distribution over data categories for each component. Number of rows is the number of components, and number of columns is the number of data categories. Rows sum to 1.

`comp_cos_merge` *Get cos.merge setting*

Description

Get cos.merge setting

Usage

```
comp_cos_merge(x)
```

Arguments

`x`	hdpSampleChain or hdpSampleMulti

Value

number of components

`comp_dp_counts` *Get sample vs component counts for each DP*

Description

Get sample vs component counts for each DP

Usage

```
comp_dp_counts(x)
```

Arguments

`x`	hdpSampleChain or hdpSampleMulti

Value

List of matrices (one for each DP) counting sample-component assignment (aggregating across data categories). Number of rows is the number of posterior samples, and number of columns is the number of components.

`comp_dp_distn` *Get mean distribution over components for each DP*

Description

Get mean distribution over components for each DP

Usage

```
comp_dp_distn(x)
```

Arguments

`x`	hdpSampleChain or hdpSampleMulti

Value

List with elements "mean" and "cred.int", containing matrices with the mean (and lower/upper 95% credibility interval) distribution over components for each DP. Number of rows is the number of DPs, and number of columns is the number of components. Rows sum to 1.

`cull_posterior_samples` *Cull early posterior samples from a hdpSampleChain object*

Description

Extend the 'burn-in' period and reduce the number of posterior samples taken from a sampling chain by culling the first `ncull` posterior samples. If components have been previously calculated for this sampling chain, they will be removed and must be recalculated.

Usage

```
cull_posterior_samples(chain, ncull)
```

Arguments

`chain`	A hdpSampleChain object
`ncull`	The number of posterior samples to cull

Value

A hdpSampleChain object with the designated 'burn-in' period extended, and the number of posterior samples reduced by `ncull`

See Also

`plot_lik`, `plot_numcluster`, `plot_data_assigned`

`default_merge_raw_cluster_args`

Default arguments for combining "raw clusters" into "aggregated raw clusters".

Description

Default arguments for combining "raw clusters" into "aggregated raw clusters".

Usage

```
default_merge_raw_cluster_args()
```

Value

A list with the following elements:

* `identical.cutoff`

* `nearly.identical.cutoff`

* `clustering.cutoff`

* `discard.singletons`

`diagnostic_in_extraction`

Diagnostic plotting inside of hdp_merge_and_extract_components function. This function generates details of the raw clusters in hdp.0

Description

Diagnostic plotting inside of hdp_merge_and_extract_components function. This function generates details of the raw clusters in hdp.0

Usage

```
diagnostic_in_extraction(
  clust_hdp0_ccc,
  ncat,
  nsamp,
  nch,
  ccc,
  cdc,
  diagnostic.folder
)
```

Arguments

`clust_hdp0_ccc`	This function is deprecated.
`ncat`	Number of categories.
`nsamp`	Number of posterior samples.
`nch`	Number of posterior chains.
`ccc`	clust_categ_counts.
`cdc`	clust_dp_counts .
`diagnostic.folder`	A directory where details for hdp.0 are plotted.

Value

The plots of presence of a raw cluster in each chain.

`dp_activate` *Activate DP nodes*

Description

Specify the number of starting clusters, and activate the DP nodes to be included in the posterior sampling process (`hdp_posterior`). When initialised, the DP nodes are 'held out' (not available for posterior sampling).

Usage

```
dp_activate(hdp, dpindex, initcc, seed = sample(1:10^7, 1))
```

Arguments

`hdp`	A hdpState object
`dpindex`	Indices of the DPs to activate (include all parent DPs)
`initcc`	Number of data clusters to start with (every data item is randomly assigned to a cluster to start with)
`seed`	The (integer) seed that can be set to reproduce output. Default is a random seed from 1 – 10^7, reported in the output.

Details

Note that this step can be slow and memory-intensive for very large datasets.

Value

A hdpState object with activated DPs and an initial random cluster allocation for each data item. See `hdpState-class`

See Also

`hdp_init`, `hdp_addconparam`, `hdp_adddp`, `hdp_setdata`, `hdp_posterior`

Examples

```
my_hdp <- hdp_init(ppindex=0, cpindex=1, hh=rep(1, 6), alphaa=rep(1, 3), alphab=rep(2, 3)
my_hdp <- hdp_adddp(my_hdp, 2, 1, 2)
my_hdp <- hdp_adddp(my_hdp, 10, c(rep(2, 5), rep(3, 5)), 3)
my_hdp <- hdp_setdata(my_hdp, 4:13, example_data_hdp)
# active all DPs and start with two data clusters
my_hdp <- dp_activate(my_hdp, 1:13, 2)
```

`dp_freeze` *Freeze DP nodes*

Description

Freezes previously active DP nodes. A frozen DP node is not included in posterior sampling, but its statistics *are* considered in the sampling of other active DPs. This is useful for conditioning on a previous dataset. First, set up a HDP for one dataset, run the posterior sampling chain, and then freeze all old nodes (except the top DP). Add new DP nodes with new data and run a second posterior sampling chain over the new nodes (*given* the information in the frozen nodes).

Usage

```
dp_freeze(hdp, dpindex)
```

Arguments

`hdp`	A hdpState object
`dpindex`	Indices of the DPs to freeze

Value

A hdpState object with the specified DP nodes frozen. See `hdpState-class`

See Also

`hdp_init`, `hdp_addconparam`, `hdp_adddp`, `hdp_setdata`, `dp_activate`, `hdp_posterior`

`example_data_hdp` *Fake categorical count data*

Description

Fake categorical count data with 10 samples and 6 categories. Generated from two underlying categorical data distributions with a different average mixture ratio in the first five samples from the last five samples.

Usage

```
example_data_hdp
```

Format

A numeric count matrix with 10 rows and 6 columns

`example_data_hdp_prior` *Fake categorical count data with priors*

Description

Fake categorical count data with 100 samples and 10 categories. Generated from four underlying categorical data distributions. Two of the underlying components are available as known prios in `example_known_priors`.

Usage

```
example_data_hdp_prior
```

Format

A numeric count matrix with 100 rows and 10 columns

`example_known_priors` *Example known priors*

Description

Two example prior components for the example data `example_data_hdp_prior`. 10 rows (one per data category) and 2 columns (one per prior component, each sums to 1).

Usage

```
example_known_priors
```

Format

A numeric matrix with 10 rows and 2 columns.

`extract_ccc_from_hdp` *Find the ccc and cdc that matched to a spectrum in ccc_0 and cdc_0 this function is to summarize the credint and mean of cccs and cdcs and for further diagnostic plotting.*

Description

Find the ccc and cdc that matched to a spectrum in ccc_0 and cdc_0 this function is to summarize the credint and mean of cccs and cdcs and for further diagnostic plotting.

Usage

```
extract_ccc_from_hdp(signature, ccc_0, cos.merge = 0.9)
```

Arguments

`signature`	A numerical vector representing the signature for which want to find information in `ccc_0`.
`ccc_0`	"CCCs" from sample chains. A list of lists of numerical matrices. At the top level, one list for each Gibbs sample chain. Each element of a top-level list is a numerical matrix, with columns for raw clusters and rows being mutation types. The number of rows in these matrices should be the length of `signature`.
`cos.merge`	The minimum cosine similarity for declaring a match between `signature` and a column of one of the matrices in in `ccc_0`. This should probably be the same as the cuttoff for separating clusters of mutations after divisive clustering `extract_components`.

Value

Invisibly, a list with at least the elements

- `ccc_mean`

- `ccc_credint`

`extract_components` *Combine "raw clusters" of mutations in "components" (clusters of mutations generated by one mutational process).*

Description

Combine "raw clusters" of mutations in "components" (clusters of mutations generated by one mutational process).

Usage

```
extract_components(
  sample.chains,
  merge.raw.cluster.args = default_merge_raw_cluster_args()
)
```

Arguments

`sample.chains`	An `hdpSampleChain-class` or \codehdpSampleMulti-class object or a list of `hdpSampleChain-class` objects.
`merge.raw.cluster.args`	See `default_merge_raw_cluster_args`.

Value

A list with the elements

components Aggregated clusters as a data frame. Rows represent the categories (i.e. for mutational signature analysis, the mutation type, e.g. ACT -> AGT). Columns are aggregated clusters, i.e. clusters after all "raw clusters" across all Gibbs samples have been combined according to the divisive clustering. Each cell contains number of items (for mutational signature analysis, the number of mutations) of a particular category in a particular aggregated cluster.

components.post.samples A data frame with two columns: one is the index of column in `components` and the other is the number of posterior samples that contributed to that aggregated cluster (column in `components`).

components.cdc A numerical matrix. Each row is a Dirichlet process (DP). This can either be a leaf DP, which for mutational signatures corresponds to a biological sample (for example, a tumor), or a parent or ancestor DP. Each column corresponds to the cluster in the corresponding column n `components`

each.chain.noise.clusters Deprecated.

each.chain.noise.cdc Deprecated.

multi.chains An `hdpSampleChain-class` or `hdpSampleMulti-class` object updated with component information.

nsamp The total number of posterior samples across all Gibbs sampling chains.

See Also

`hdp_posterior`, `hdp_multi_chain`, `plot_comp_size`, `plot_comp_distn`, `plot_dp_comp_exposur`

`hdpBase-class` *hdpBase class for the base distribution*

Description

hdpBase class for the base distribution

Usage

```
## S4 method for signature 'hdpBase'
as.list(x, ...)
```

Arguments

`x`	Object of class hdpBase
`...`	unused

Methods (by generic)

- `as.list(hdpBase)`: convert to list class

Slots

`hh` parameters for base Dirichlet distribution (pseudocounts)

`classqq` overall count matrix for data items of each category (rows) in each cluster (columns)

`numclass` number of clusters

`hdpConparam-class` *hdpConparam class for the DP concentration parameter/s*

Description

hdpConparam class for the DP concentration parameter/s

Usage

```
## S4 method for signature 'hdpConparam'
as.list(x, ...)
```

Arguments

`x`	Object of class hdpConparam
`...`	unused

Methods (by generic)

- `as.list(hdpConparam)`: convert to list class

Slots

`alphaa` shape parameter for the gamma prior over alpha

`alphab` rate parameter for the gamma prior over alpha

`numdp` number of DP nodes sharing this concentration parameter

`alpha` concentration parameter value

`totalnd` number of data items in each DP with this concentration parameter

`totalnt` number of tables in each DP with this concentration parameter

`hdpDP-class` *hdpDP class for a DP node*

Description

note that the 'items' in parent nodes are the tables of their children

Usage

```
## S4 method for signature 'hdpDP'
as.list(x, ...)

## S4 method for signature 'hdpDP'
numdata(x, ...)
```

Arguments

`x`	Object of class hdpDP
`...`	unused

Methods (by generic)

- `as.list(hdpDP)`: convert to list class
- `numdata(hdpDP)`: Get number of data items at this DP.

Slots

`datacc` cluster index for each data item

`classnd` number of items assigned to each cluster in this DP

`classnt` number of tables assigned to each cluster in this DP

`beta` weight on each cluster in this DP (including empty cluster at end)

`alpha` concentration parameter for this DP

`numdata` number of data items registered to this DP node

`datass` value of each data item

`hdpSampleChain-class` *hdpSampleChain class for posterior samples off one MCMC chain*

Description

hdpSampleChain class for posterior samples off one MCMC chain

Usage

```
## S4 method for signature 'hdpSampleChain'
as.list(x, ...)

## S4 method for signature 'hdpSampleChain'
sampling_seed(x, ...)

## S4 method for signature 'hdpSampleChain'
hdp_settings(x, ...)

## S4 method for signature 'hdpSampleChain'
final_hdpState(x, ...)

## S4 method for signature 'hdpSampleChain'
lik(x, ...)

## S4 method for signature 'hdpSampleChain'
numcluster(x, ...)

## S4 method for signature 'hdpSampleChain'
cp_values(x, ...)

## S4 method for signature 'hdpSampleChain'
clust_categ_counts(x, ...)
```

```
## S4 method for signature 'hdpSampleChain'
clust_dp_counts(x, ...)

## S4 method for signature 'hdpSampleChain'
numcomp(x)

## S4 method for signature 'hdpSampleChain'
prop.ex(x)

## S4 method for signature 'hdpSampleChain'
comp_cos_merge(x)

## S4 method for signature 'hdpSampleChain'
comp_categ_counts(x)

## S4 method for signature 'hdpSampleChain'
comp_dp_counts(x)

## S4 method for signature 'hdpSampleChain'
comp_categ_distn(x)

## S4 method for signature 'hdpSampleChain'
comp_dp_distn(x)
```

Arguments

`x`	Object of class hdpSampleChain
`...`	unused

Methods (by generic)

- `as.list(hdpSampleChain)`: Convert to list class
- `sampling_seed(hdpSampleChain)`: Get random seed used by `hdp_posterior`
- `hdp_settings(hdpSampleChain)`: Get settings of posterior sampling chain
- `final_hdpState(hdpSampleChain)`: Get hdpState object from the end of the posterior sampling chain
- `lik(hdpSampleChain)`: Get likelihood of data given model over all iterations
- `numcluster(hdpSampleChain)`: Get the number of clusters for each posterior sample
- `cp_values(hdpSampleChain)`: Get matrix of concentration parameter values for each posterior sample
- `clust_categ_counts(hdpSampleChain)`: Get category vs cluster counts for each posterior sample
- `clust_dp_counts(hdpSampleChain)`: Get dp node vs cluster counts for each posterior sample
- `numcomp(hdpSampleChain)`: Get number of extracted components for hdpSampleChain
- `prop.ex(hdpSampleChain)`: Get proportion of dataset explained (on average) for hdpSampleChain
- `comp_cos_merge(hdpSampleChain)`: Get cos.merge setting for hdpSampleChain

- `comp_categ_counts(hdpSampleChain)`: Get sample vs category counts for each component
- `comp_dp_counts(hdpSampleChain)`: Get sample vs component counts for each DP
- `comp_categ_distn(hdpSampleChain)`: Get mean distribution over data categories for each component
- `comp_dp_distn(hdpSampleChain)`: Get mean distribution over components for each DP

Slots

`seed` Random seed used by `hdp_posterior`

`settings` Settings of the posterior sampling chain: burnin, n (number of samples collected), space (iters between samples), cpiter (con param moves between iters)

`hdp` hdpState object after the final iteration

`lik` Likelihood of data given model at each iteration

`numcluster` Number of raw data clusters in each posterior sample

`cp_values` Matrix of concentration parameter values (one column for each parameter) in each posterior sample (rows).

`clust_categ_counts` List of matrices (one from each posterior sample) counting the category-cluster data assignment across all DP nodes. Number of rows is the number of categories (constant), and number of columns is the number of clusters in that posterior sample (variable).

`clust_dp_counts` List of matrices (one from each posterior sample) counting within-DP cluster assignment (aggregating across data categories). Number of rows is the number of DPs (constant), and number of columns is the number of clusters in that posterior sample (variable).

`numcomp` Number of global components

`prop.ex` (Average) proportion of dataset explained by the extracted components

`comp_cos_merge` `cos.merge` setting used by `hdp_extract_components`; deprecated

`comp_categ_counts` List of matrices (one for each component) counting the sample-category data assignment across all DP nodes. Number of rows is the number of posterior samples, and number of columns is the number of data categories.

`comp_dp_counts` List of matrices (one for each DP) counting sample-component assignment (aggregating across data categories). Number of rows is the number of posterior samples, and number of columns is the number of components.

`comp_categ_distn` List with elements `mean` and `cred.int`, containing matrices with the mean (and lower/upper 95% credibility interval) distribution over data categories for each component. Number of rows is the number of components, and number of columns is the number of data categories. Rows sum to 1.

`comp_dp_distn` List with elements "mean" and "cred.int", containing matrices with the mean (and lower/upper 95% credibility interval) distribution over components for each DP. Number of rows is the number of DPs, and number of columns is the number of components. Rows sum to 1.

hdpSampleMulti-class

hdpSampleMulti class for multiple independent hdpSampleChain objects for the same HDP

Description

hdpSampleMulti class for multiple independent hdpSampleChain objects for the same HDP

Usage

```
## S4 method for signature 'hdpSampleMulti'
as.list(x, ...)

## S4 method for signature 'hdpSampleMulti'
chains(x, ...)

## S4 method for signature 'hdpSampleMulti'
numcomp(x)

## S4 method for signature 'hdpSampleMulti'
prop.ex(x)

## S4 method for signature 'hdpSampleMulti'
comp_cos_merge(x)

## S4 method for signature 'hdpSampleMulti'
comp_categ_counts(x)

## S4 method for signature 'hdpSampleMulti'
comp_dp_counts(x)

## S4 method for signature 'hdpSampleMulti'
comp_categ_distn(x)

## S4 method for signature 'hdpSampleMulti'
comp_dp_distn(x)
```

Arguments

`x`	Object of class hdpSampleMulti
`...`	unused

Methods (by generic)

- `as.list(hdpSampleMulti)`: Convert to list class
- `chains(hdpSampleMulti)`: Get list of hdpSampleChain objects
- `numcomp(hdpSampleMulti)`: Get number of extracted components for hdpSampleMulti
- `prop.ex(hdpSampleMulti)`: Get proportion of dataset explained (on average) for hdp-SampleMulti

- `comp_cos_merge(hdpSampleMulti)`: Get cos.merge setting for hdpSampleMulti
- `comp_categ_counts(hdpSampleMulti)`: Get sample vs category counts for each component
- `comp_dp_counts(hdpSampleMulti)`: Get sample vs component counts for each DP
- `comp_categ_distn(hdpSampleMulti)`: Get mean distribution over data categories for each component
- `comp_dp_distn(hdpSampleMulti)`: Get mean distribution over components for each DP

Slots

`chains` List of hdpSampleChain objects storing multiple independent runs of the posterior sampling chain for the same data and HDP struct

`numcomp` Number of global components extracted by `hdp_extract_components` (not including component 0)

`prop.ex` (Average) proportion of dataset explained by the extracted components

`comp_cos_merge` `cos.merge` setting used by `hdp_extract_components`

`comp_categ_counts` List of matrices (one for each component) counting the sample-category data assignment across all DP nodes. Number of rows is the number of posterior samples, and number of columns is the number of data categories.

`comp_dp_counts` List of matrices (one for each DP) counting sample-component assignment (aggregating across data categories). Number of rows is the number of posterior samples, and number of columns is the number of components.

`comp_categ_distn` List with elements "mean" and "cred.int", containing matrices with the mean (and lower/upper 95% credibility interval) distribution over data categories for each component. Number of rows is the number of components, and number of columns is the number of data categories. Rows sum to 1.

`comp_dp_distn` List with elements "mean" and "cred.int", containing matrices with the mean (and lower/upper 95% credibility interval) distribution over components for each DP. Number of rows is the number of DPs, and number of columns is the number of components. Rows sum to 1.

`hdpState-class` *hdpState class for a Hierarchical Dirichlet Process in one state*

Description

hdpState class for a Hierarchical Dirichlet Process in one state

Usage

```
## S4 method for signature 'hdpState'
as.list(x, ...)

## S4 method for signature 'hdpState'
numdp(x, ...)

## S4 method for signature 'hdpState'
```

```
numconparam(x, ...)

## S4 method for signature 'hdpState'
base(x, ...)

## S4 method for signature 'hdpState'
conparam(x, ...)

## S4 method for signature 'hdpState'
dp(x, ...)

## S4 method for signature 'hdpState'
dpstate(x, ...)

## S4 method for signature 'hdpState'
ppindex(x, ...)

## S4 method for signature 'hdpState'
cpindex(x, ...)

## S4 method for signature 'hdpState'
numcateg(x, ...)

## S4 method for signature 'hdpState'
base_params(x, ...)

## S4 method for signature 'hdpState'
activating_seed(x, ...)

## S4 method for signature 'hdpState'
pseudoDP(x, ...)
```

Arguments

`x`	Object of class hdpState
`...`	unused

Methods (by generic)

- `as.list(hdpState)`: Convert to list class
- `numdp(hdpState)`: Get number of DPs
- `numconparam(hdpState)`: Get number of concentration parameters
- `base(hdpState)`: Get base distribution
- `conparam(hdpState)`: Get list of concentration parameters
- `dp(hdpState)`: Get list of DP nodes
- `dpstate(hdpState)`: Get state of every DP
- `ppindex(hdpState)`: Get parent process index of every DP
- `cpindex(hdpState)`: Get concentration parameter index of every DP
- `numcateg(hdpState)`: Get number of data categories

- `base_params(hdpState)`: Get parameters of the base Dirichlet distribution (like psuedocounts across categories).
- `activating_seed(hdpState)`: Get seed used to initialse clustering
- `pseudoDP(hdpState)`: Get index of frozen pseudo-data DP nodes for prior info (only if initialised via hdp_prior_init)

Slots

`numdp`	number of DP nodes in the hierarchical Dirichlet Process
`numconparam`	number of concentration parameters
`base`	base distribution (hdpBase object)
`conparam`	concentration parameters (list of hdpConparam objects)
`dp`	DP nodes (list of hdpDP objects)
`dpstate`	state of DP nodes for posterior sampling process: active (2), frozen (1), or heldout (0)
`ppindex`	parent node index for each DP
`cpindex`	concentration parameter index for each DP
`ttindex`	DP index of those sharing a concentration parameter
`initcc`	number of initial clusters
`seed_activate`	random seed used to initiate cluster membership
`pseudoDP`	(Optional) index of pseudodata nodes (only if initialised via hdp_prior_init)

`hdp_addconparam` *Add concentration parameters to a hdpState object*

Description

Add concentration parameters to a hdpState object by specifying the shape and rate parameters of the gamma prior/s. DPs using these new concentration parameters can be added with `hdp_adddp`. Data is assigned via `hdp_setdata`. When initialised, the DP nodes are 'heldout' (not available for posterior sampling) and will need to be activated (see `dp_activate`). Finally, the posterior sampling process (a Gibbs sampler) is run via `hdp_posterior`.

Usage

```
hdp_addconparam(hdp, alphaa, alphab)
```

Arguments

`hdp`	A hdpState object
`alphaa`	Shape hyperparameters for the gamma priors over the DP concentration parameters.
`alphab`	Rate hyperparameters for the gamma priors over the DP concentration parameters.

Value

A hdpState object updated with the new concentration parameters. See `hdpState-class`

See Also

hdp_init, hdp_adddp, hdp_setdata, dp_activate, hdp_posterior

Examples

```
hdp_example <- hdp_init(c(0, 1, 1), c(1, 2, 2), rep(1, 6), rep(2, 2), rep(0.5, 2))
hdp_example <- hdp_addconparam(hdp_example, rep(1, 2), rep(1, 2))
```

---

hdp_adddp *Add DPs to a hdpState object*

Description

Add DP nodes to a hdpState object and specify each parent relationship and concentration parameter. Concentration parameters can be added to a hdpState object with hdp_addconparam. Data is assigned via hdp_setdata. When initialised, the DP nodes are 'heldout' (not available for posterior sampling) and will need to be activated (see dp_activate). Finally, the posterior sampling process (a Gibbs sampler) is run via hdp_posterior.

Usage

```
hdp_adddp(hdp, numdp, ppindex, cpindex)
```

Arguments

hdp	A hdpState object
numdp	The number of DPs to add
ppindex	Index (or indices) of the parental process(es) for the new DPs.
cpindex	Index (or indices) of the concentration parameters for the new DPs.

Value

A hdpState object with the updated HDP structure. See hdpState-class

See Also

hdp_init, hdp_setdata, dp_activate, hdp_posterior

Examples

```
my_hdp <- hdp_init(ppindex=0, cpindex=1, hh=rep(1, 6), alphaa=rep(1, 3), alphab=rep(2, 3)
# add two more DPs with parent '1' and concentration parameter '2'
my_hdp <- hdp_adddp(my_hdp, 2, 1, 2)
my_hdp
hdp_example <- hdp_init(c(0, 1, 1), c(1, 2, 2), rep(1, 6), rep(2, 2), rep(0.5, 2))
# add six more DPs, three with parent '2', three with parent '3',
# and all with concentration parameter '2'
hdp_example <- hdp_adddp(hdp_example, 6, c(2, 2, 2, 3, 3, 3), 2)
hdp_example
```

`hdp_burnin` *Burnin of posterior sampling chain across activated DPs.*

Description

Run a Gibbs sampler over the activated nodes of a hierarchical Dirichlet process. Each iteration re-assigns the cluster allocation of every data item.

Usage

```
hdp_burnin(hdp, burnin, cpiter = 1, verbosity = 0)
```

Arguments

`hdp`	An `hdpState-class` object or a list representation of this. The list representation contains elements that correspond to slots in an`hdpState-class` object.
`burnin`	The number of burn-in iterations.
`cpiter`	The number of iterations of concentration parameter sampling to perform after each iteration.
`verbosity`	Verbosity of debugging statements. 0 (least verbose) – 4 (most verbose). 0 highly recommended - only change for debugging small examples.

Value

A list with the elements:

hdplist A list representation of an `hdpState-class` object.

likelihood A numeric vector with the likelihood at each iteration.

`hdp_init` *Initialise a hdpState object*

Description

Initialise a hdpState object with one or more DP nodes and their parent relationships, the parameters of the base Dirichlet distribution, and a set of hyperparameters for the gamma priors over the DP concentration parameters. Further DP nodes can be added with `hdp_adddp`, and further concentration parameters can be added with `hdp_addconparam`. Data is assigned via `hdp_setdata`. When initialised, the DP nodes are 'heldout' (not available for posterior sampling) and will need to be activated (see `dp_activate`). Finally, the posterior sampling process (a Gibbs sampler) is run via `hdp_posterior`.

Usage

```
hdp_init(ppindex, cpindex, hh, alphaa, alphab)
```

Arguments

`ppindex`	Index (or indices) of the parental process(es) for the initial DPs. The 'top' DP should have parent process '0' (the base Dirichlet distribution).
`cpindex`	Index (or indices) of the concentration parameter(s) for the initial DPs.
`hh`	Parameters of the base Dirichlet distribution (like psuedocounts across categories). Must be a vector with length equal to the number of data item categories.
`alphaa`	Shape hyperparameters for the gamma priors over the DP concentration parameters.
`alphab`	Rate hyperparameters for the gamma priors over the DP concentration parameters.

Value

A hdpState object with the initial HDP structure. See `hdpState-class`

See Also

`hdp_quick_init`, `hdp_prior_init`, `hdp_addconparam`, `hdp_adddp`, `hdp_setdata`, `dp_activate`, `hdp_posterior`

Examples

```
# initialise a HDP with just one 'top' DP node off the base distribution,
# a uniform Dirichlet base distribution over six possible data categories,
# and three possible concentration parameters to be shared across the HDP tree
# (top DP using conparam number 1), each with hyperparameters (1,2).
hdp_init(ppindex=0, cpindex=1, hh=rep(1, 6), alphaa=rep(1, 3), alphab=rep(2, 3))

# initialise a HDP with one 'top' DP node off the base distribution,
# AND two children DP nodes off that parent. The two children DPs share a different
# concentration parameter (hyperparameters are (2, 0.5)).
hdp_init(ppindex=c(0, 1, 1), cpindex=c(1, 2, 2), hh=rep(1, 6), alphaa=rep(2, 2), alphab=r
```

`hdp_multi_chain` *Gather multiple independent posterior sampling chains for the same HDP*

Description

Gather multiple independent posterior sampling chains for the same HDP

Usage

```
hdp_multi_chain(chain_list)
```

Arguments

`chain_list`	A list of hdpSampleChain objects for the same data and HDP structure, but run with different random seeds.

Value

A hdpSampleMulti object

See Also

hdp_posterior

`hdp_posterior` *Posterior sampling chain across activated DPs.*

Description

Run a Gibbs sampler over the activated DP nodes of a Hierarchichal Dirichlet Process. Each iteration re-assigns the cluster allocation of every data item. Run `burnin` iterations, and then collect `n` samples from the chain with `space` iterations between each collected sample. To plot output, see `plot_lik`, `plot_numcluster`, and `plot_data_assigned`. Can collect multiple independent HDP sampling chains in a hdpSampleMulti object via `hdp_multi_chain`.

Usage

```
hdp_posterior(
  hdp,
  burnin,
  n,
  space,
  cpiter = 1,
  seed = sample(1:10^7, 1),
  verbosity = 0
)
```

Arguments

`hdp`	A hdpState object
`burnin`	The number of burn-in iterations.
`n`	The number of posterior samples to collect.
`space`	The number of iterations between collected samples.
`cpiter`	The number of iterations of concentration parameter sampling to perform after each iteration.
`seed`	The (integer) seed that can be set to reproduce output. Default is a random seed from 1 – 10^7, reported in the output.
`verbosity`	Verbosity of debugging statements. 0 (least verbose) – 4 (most verbose). 0 highly recommended - only change for debugging small examples.

Value

A hdpSampleChain object with the salient information from each posterior sample. See `hdpSampleChain-class`

See Also

`hdp_multi_chain`, `cull_posterior_samples`, `plot_lik`, `plot_numcluster`, `plot_data_assigne`

Examples

```
my_hdp <- hdp_init(ppindex=0, cpindex=1, hh=rep(1, 6), alphaa=rep(1, 3), alphab=rep(2, 3)
my_hdp <- hdp_adddp(my_hdp, 2, 1, 2)
my_hdp <- hdp_adddp(my_hdp, 10, c(rep(2, 5), rep(3, 5)), 3)
my_hdp <- hdp_setdata(my_hdp, 4:13, example_data_hdp)
my_hdp <- dp_activate(my_hdp, 1:13, 2)
my_hdp_chain <- hdp_posterior(my_hdp, 100, 100, 10)
```

`hdp_posterior_sample` *Posterior sampling chain across activated DPs.*

Description

Run a Gibbs sampler over the burnin chains from `hdp_burnin`. Each iteration re-assigns the cluster allocation of every data item. Run `burnin` iterations, and then collect `n` samples from the chain with `space` iterations between each collected sample. To plot output, see `plot_lik`, `plot_numcluster`, and `plot_data_assigned`. Can collect multiple independent HDP sampling chains in a hdpSampleMulti object via `hdp_multi_chain`.

Usage

```
hdp_posterior_sample(
  post.input,
  post.n,
  post.space,
  post.cpiter = 1,
  seed = sample(1:10^7, 1),
  post.verbosity = 0,
  checkpoint = F
)
```

Arguments

`post.input`	An S4 object from `hdp_burnin`.
`post.n`	The number of posterior samples to collect.
`post.space`	The number of iterations between collected samples.
`post.cpiter`	The number of iterations of concentration parameter sampling to perform after each iteration.
`seed`	The (integer) seed that can be set to reproduce output. Default is a random seed from 1 – 10^7, reported in the output.
`post.verbosity`	Verbosity of debugging statements. 0 (least verbose) – 4 (most verbose). 0 highly recommended - only change for debugging small examples.
`checkpoint`	If `TRUE`, a checkpoint will be saved for every 10 posterior samples

Value

A hdpSampleChain object with the salient information from each posterior sample. See `hdpSampleChain-class`

See Also

hdp_multi_chain, cull_posterior_samples, plot_lik, plot_numcluster, plot_data_assigne

---

hdp_prior_init — *Initialise a HDP structure incorporating prior knowledge*

Description

Initialise a hdpState object incorporating prior knowlegde of some components (categorical data distributions). The structure has one top parent DP node with no associated data ('active' and available for posterior sampling), and one child DP node per prior component ('frozen' and held out from posterior sampling).

Usage

```
hdp_prior_init(prior_distn, prior_pseudoc, hh, alphaa, alphab)
```

Arguments

`prior_distn`	Matrix of prior distributions (columns must each sum to 1, number of rows matches number of data categories)
`prior_pseudoc`	Vector of pseudocounts contributed by each prior distribution
`hh`	Parameters of the base Dirichlet distribution. Must be a vector with length equal to the number of data item categories.
`alphaa`	Shape hyperparameters for the gamma priors over the DP concentration parameters.
`alphab`	Rate hyperparameters for the gamma priors over the DP concentration parameters.

Value

A hdpState object with one frozen node per prior component. See hdpState-class

See Also

hdp_init

Examples

```
# example dataset with 10 data categories, and 100 samples.
# Two components are known a priori.
hdp <- hdp_prior_init(example_known_priors, rep(1000, 2), hh=rep(1, 10),
             alphaa=c(1,1), alphab=c(1,1))
hdp <- hdp_addconparam(hdp, alphaa=c(1,1), alphab=c(1,1))
hdp <- hdp_adddp(hdp, 101, c(1, rep(4, 100)), c(3, rep(4, 100)))
hdp <- hdp_setdata(hdp, 5:104, example_data_hdp_prior)
hdp <- dp_activate(hdp, 4:104, initcc=4, seed=81479)
hdp <- hdp_posterior(hdp, burnin=2000, n=50, space=50, cpiter=3, seed=1e6)
```

`hdp_quick_init` *Initialise a simple, default HDP structure*

Description

Initialise a hdpState object with a basic default structure of one top parent DP node with no associated data, and one child DP node per row of data. Every DP node shares the same concentration parameter, and will automatically be 'activated' (made available for posterior samplig). The base distribution is a uniform Dirichlet with psuedocount 1 in each data category. Can immediately run `hdp_posterior` to collect posterior samples. To define a custom HDP structure, see `hdp_init` and `hdp_prior_init`.

Usage

```
hdp_quick_init(data, initcc = 2, alphaa = 1, alphab = 1)
```

Arguments

`data`	A `data.frame` or `matrix` of counts with one row for every sample and one column for every data category.
`initcc`	Number of initial data clusters (every data item is randomly assigned to a cluster to start with).
`alphaa`	Shape hyperparameter for the gamma prior over the concentration parameter.
`alphab`	Rate hyperparameter for the gamma prior over the concentration parameter.

Value

A hdpState object with a basic default structure. See `hdpState-class`

See Also

`hdp_init`, `hdp_posterior`, `hdp_prior_init`

Examples

```
my_quick_hdp <- hdp_quick_init(example_data_hdp)
my_quick_hdp_chain <- hdp_posterior(my_quick_hdp, 100, 50, 10, 5)
```

`hdp_setdata` *Assign data to DP nodes in a hdpState object*

Description

Assign data to 'heldout' (state is 0) DP nodes in a hdpState object. 'Heldout' DPs are not available for posterior sampling, and will need to be activated (see `dp_activate`). The posterior sampling process (a Gibbs sampler) is run via `hdp_posterior`.

Usage

```
hdp_setdata(hdp, dpindex, data)
```

Arguments

`hdp`	A hdpState object
`dpindex`	Indices of the DPs to assign data to (in same order as rows of `data`)
`data`	A `data.frame` or `matrix` of counts with one row for every sample (same order as `dpindex`) and one column for every data category.

Value

A hdpState object updated with the new data. See `hdpState-class`

See Also

`hdp_init`, `hdp_adddp`, `dp_activate`, `hdp_posterior`

Examples

```
example_data_hdp
my_hdp <- hdp_init(ppindex=0, cpindex=1, hh=rep(1, 6), alphaa=rep(1, 3), alphab=rep(2, 3)
my_hdp <- hdp_adddp(my_hdp, 2, 1, 2)
my_hdp <- hdp_adddp(my_hdp, 10, c(rep(2, 5), rep(3, 5)), 3)
my_hdp <- hdp_setdata(my_hdp, 4:13, example_data_hdp)
dp(my_hdp)
```

`interpret_components` *Separate high and low confidence components (aggregated clusters) and exposures*

Description

Separate high and low confidence components (aggregated clusters) and exposures

Usage

```
interpret_components(
  multi.chains.retval,
  high.confidence.prop = 0.9,
  verbose = FALSE
)
```

Arguments

`multi.chains.retval`	A list that contains all the elements returned by `extract_components`.
`high.confidence.prop`	Components found in $>=$ `high.confidence.prop` proportion of posterior samples are high confidence components.
`verbose`	if TRUE, generate progress messages.

Value

In the information that follows, a "component" is the union of multiple raw clusters of mutations (in the case of mutational signature analysis). Invisibly, a list with the following elements:

high_confidence_components A data frame containing the components found in >= `high.confidence.prop` of posterior samples. Each column is a component; in the case of mutational signatures the rows are mutation types.

high_confidence_components_post_number A data frame in which the first column contains the index of a column in `high_confidence_components` and the second column contains the number of posterior samples that contributed to that component.

high_confidence_components_cdc A matrix in which each row corresponds to one of the Dirichlet processes, and each column corresponds to one component in `high_confidence_components`. In the case of mutational signature analysis, most of the columns correspond to an input biological sample (e.g. individual tumor).

low_confidence_components Analogous to `high_confidence_compents` except for components with constituent raw clusters found in < `high.confidence.prop` posterior samples.

low_confidence_components_post_number Analogous to `high_confidence_components_post_number`.

low_confidence_components_cdc Analogous to `high_confidence_components_cdc`.

`numcomp` *Get number of extracted components*

Description

Get number of extracted components

Usage

```
numcomp(x)
```

Arguments

`x`	hdpSampleChain or hdpSampleMulti

Value

number of components

`plotchain` *Diagnostic plots for HDP posterior sampling chain*

Description

Diagnostic plots for HDP posterior sampling chain

Usage

```
plot_lik(
  chain,
  start = 1,
  end = length(lik(chain)),
  col_lik = "blue",
  col_burn = "red",
  xlab = "Iteration",
  ylab = "Likelihood",
  ...
)

plot_numcluster(
  chain,
  col = "blue",
  xlab = "Sample",
  ylab = "Number of raw clusters",
  ...
)

plot_data_assigned(
  chain,
  legend = TRUE,
  col_early = "hotpink",
  col_late = "skyblue3",
  dat_prop = 0.995,
  xlab = "Number of raw clusters",
  ylab = "Cumulative prop. of data assigned",
  ...
)
```

Arguments

`chain`	A hdpSampleChain object
`start`	The starting iteration to plot from (default 1)
`end`	The final iteration to plot to (default is end of chain)
`col_lik`	Plot colour of likelihood (default blue)
`col_burn`	Plot colour of burnin (default red)
`xlab`	Horizontal axis label
`ylab`	Vertical axis label
`...`	Other arguments to plot

`col`	Plot colour for numcluster (default blue)
`legend`	Logical - should a legend be included? (default TRUE)
`col_early`	Color ramp side for early posterior samples
`col_late`	Color ramp side for late posterior samples
`dat_prop`	Extend horiztonal axis to dat_prop proportion of data assigned

---

`plotcomp` *Plot extracted components*

Description

Plot extracted components

Plot hdp signature exposure in each sample

Usage

```
plot_comp_size(
  hdpsample,
  legend = TRUE,
  col_a = "hotpink",
  col_b = "skyblue3",
  xlab = "Component",
  ylab = "Number of data items",
  ...
)

plot_comp_distn(
  hdpsample,
  comp = NULL,
  cat_names = NULL,
  grouping = NULL,
  col = "grey70",
  col_nonsig = NULL,
  show_group_labels = FALSE,
  cred_int = TRUE,
  weights = NULL,
  plot_title = NULL,
  group_label_height = 1.05,
  cex.cat = 0.7,
  ...
)

plot_dp_comp_exposure(
  hdpsample,
  input.catalog,
  ex.signature,
  col_comp,
  dpnames = NULL,
  main_text = NULL,
  incl_numdata_plot = TRUE,
  incl_nonsig = TRUE,
  incl_comp0 = TRUE,
  ylab_numdata = "Number of data items",
  ylab_exp = "Component exposure",
  leg.title = "Component",
  cex.names = 0.6,
  cex.axis = 0.7,
  mar = c(1, 4, 2, 0.5),
  oma = c(1.5, 1.5, 1, 1),
  ...
)
```

Arguments

`hdpsample`	A hdpSampleChain or hdpSampleMulti.
`legend`	Logical - should a legend be included? (default TRUE)
`col_a`	Color ramp side for early posterior samples (if hdpSampleChain) or first chain (if hdpSampleMulti)
`col_b`	Color ramp side for late posterior samples (if hdpSampleChain) or last chain (if hdpSampleMulti)
`xlab`	Horizontal axis label
`ylab`	Vertical axis label
`...`	Other arguments to plot
`comp`	(Optional) Number(s) of the component(s) to plot (from 0 to the max component number). The default is to plot all components.
`cat_names`	(Optional) Data category names to label the horizontal axis
`grouping`	(Optional) A factor indicating data category groups.
`col`	Either a single colour for all data categories, or a vector of colours for each group (in the same order as the levels of the grouping factor)
`col_nonsig`	(Optional) Colour for any data category whose 95% credibility interval overlaps with zero (if set, overrides col argument)
`show_group_labels`	Logical - should group labels be added to the top horizontal axis? (default FALSE) (only works if categories alreayd come in orders)
`cred_int`	Logical - should 95% credibility intervals be plotted? (default TRUE)
`weights`	(Optional) Weights over the data categories to adjust their relative contribution (multiplicative)
`plot_title`	(Optional) Character vector of custom plot titles (one for each component plotted)
`group_label_height`	Multiplicative factor from top of plot for group label placement
`cex.cat`	Expansion factor for the (optional) cat_names
`input.catalog`	input catalog for samples
`ex.signature`	extracted signature from hdp
`col_comp`	Colours of each component, from 0 to the max number

`dpnames`	(Optional) Names of the DP nodes
`main_text`	(Optional) Text at top of plot
`incl_numdata_plot`	Logical - should an upper barplot indicating the number of data items per DP be included? (Default TRUE)
`incl_nonsig`	Logical - should components whose credibility intervals include 0 be included (per DP)? (Default TRUE)
`incl_comp0`	Logical - should component zero be plotted? (Default TRUE)
`ylab_numdata`	Vertical axis label for numdata plot
`ylab_exp`	Vertical exis label for exposure plot
`leg.title`	Legend title
`cex.names`	Expansion factor for bar labels (dpnames) in exposure plot
`cex.axis`	Expansion factor for vertical-axis annotation
`mar`	See ?par
`oma`	See ?par

`plot_chain_hdpsig_exp` *Plot hdp signature exposure on each chain*

Description

Plot hdp signature exposure on each chain

Usage

```
plot_chain_hdpsig_exp(hdpsample, chains, legend = TRUE)
```

Arguments

`hdpsample`	A hdpSampleChain or hdpSampleMulti object.
`chains`	A hdpSampleChain or hdpSampleMulti object in the list representation
`legend`	Logical - should a legend be included? (default TRUE)

`plot_component_posterior_samples` *Plot the distribution of raw clusters highly similar as the component in posterior chains*

Description

Plot the distribution of raw clusters highly similar as the component in posterior chains

Usage

```
plot_component_posterior_samples(components, retval)
```

Arguments

`components`	A matrix that containing components with each row corresponding a category and each column corresponding a component
`retval`	An object return from `extract_ccc_from_hdp`

`plot_component_with_credint` *Plot signatures and their 95% credible intervals*

Description

Plot signatures and their 95% credible intervals

Usage

```
plot_component_with_credint(
  retval,
  cat_names = NULL,
  col = "grey70",
  cred_int = TRUE,
  weights = NULL,
  group_label_height = 1.05,
  cex.cat = 0.7
)
```

Arguments

`retval`	an object return from `extract_ccc_from_hdp`.
`cat_names`	names displayed on x-axis, e.g. SBS96 mutation classes
`col`	Either a single colour for all data categories, or a vector of colours for each group (in the same order as the levels of the grouping factor)
`cred_int`	Logical - should 95% credibility intervals be plotted? (default TRUE)
`weights`	(Optional) Weights over the data categories to adjust their relative contribution (multiplicative)

`group_label_height`	Multiplicative factor from top of plot for group label placement
`cex.cat`	Expansion factor for the (optional) cat_names

`prop.ex` *Get proportion of dataset explained (on average)*

Description

Get proportion of dataset explained (on average)

Usage

```
prop.ex(x)
```

Arguments

`x`	hdpSampleChain or hdpSampleMulti

Value

number of components

`TestScaffold1` *Debugging scaffold for c code in hdpx/hdp*

Description

Debugging scaffold for c code in hdpx/hdp

Usage

```
TestScaffold1(
  input.catalog,
  CPU.cores = 1,
  seedNumber = 1,
  K.guess,
  multi.types = FALSE,
  verbose = TRUE,
  num.posterior = 4,
  post.burnin = 4000,
  post.n = 50,
  post.space = 50,
  post.cpiter = 3,
  post.verbosity = 0,
  cos.merge = 0.9,
  min.sample = 1,
  checkpoint.aft.post = NULL
)
```

Arguments

`input.catalog`	Input spectra catalog as a matrix or in `ICAMS` format.
`CPU.cores`	Number of CPUs to use in running `hdp_posterior`; this is used to parallelize running the posterior sampling chains, so there is no point in making this larger than `num.posterior`.
`seedNumber`	An integer that is used to generate separate random seeds for each call to `dp_activate`, and each call of `hdp_posterior`; please see the code on how this is done. But repeated calls with same value of `seedNumber` and other inputs should produce the same results.
`K.guess`	Suggested initial value of the number of signatures, passed to `dp_activate` as `initcc`.
`multi.types`	A logical scalar or a character vector. If `FALSE`, hdp will regard all input spectra as one tumor type. If `TRUE`, hdp will infer tumor types based on the string before "::" in their names. e.g. tumor type for "SA.Syn.Ovary-AdenoCA::S.500" would be "SA.Syn.Ovary-AdenoCA" If `multi.types` is a character vector, then it should be of the same length as the number of columns in `input.catalog`, and each value is the name of the tumor type of the corresponding column in `input.catalog`, e.g. `c("SA.Syn.Ovary-AdenoC`
`verbose`	If `TRUE` then `message` progress information.
`num.posterior`	Number of posterior sampling chains; can set to 1 for testing.
`post.burnin`	Pass to `hdp_posterior` `burnin`.
`post.n`	Pass to `hdp_posterior` `n`.
`post.space`	Pass to `hdp_posterior` `space`.
`post.cpiter`	Pass to `hdp_posterior` `cpiter`.
`post.verbosity`	Pass to `hdp_posterior` `verbosity`.
`cos.merge`	The cosine similarity threshold for merging raw clusters from the posterior sampling chains into "components" i.e. signatures.
`min.sample`	A "component" (i.e. signature) must have at least this many samples.
`checkpoint.aft.post`	If non-`NULL`, a file path to checkpoint the list of values returned from the calls to `hdp_posterior` as a .Rdata file.

Value

The list of sample changes returned by `hdp_posterior`.

`xmake.s` *Return a function to calculate the unsigned Stirling numbers of the first kind.*

Description

Return a function to calculate the unsigned Stirling numbers of the first kind.

Usage

```
xmake.s()
```

Value

A function to calculate a vector of unsigned Stirling numbers, $s(n, k), k = 1...n$, each divided by the maximum Stirling number in the series. The returned function is a closure with state that includes a list of all the unsigned Stirling number series $<=$ the argument, n,

i.e. $[s(1, 1)], [s(2, 1), s(2, 2)], ..., [s(n, 1), ..., s(n, n)]$. Memory usage could be substantial, but the stored state does not include the many trailing zeros in the vectors. For this to work within the hdp (https://github.com/nicolaroberts/hdp) package the function returned *must* be called `stir.closure`.

Index