Package 'cellTree'

April 12, 2022

Title Inference and visualisation of Single-Cell RNA-seq data as a hierarchical tree structure

Version 1.24.0

Authors David duVerle [aut, cre], Koji Tsuda [aut]

Encoding UTF-8

Description This packages computes a Latent Dirichlet Allocation (LDA) model of single-cell RNA-seq data and builds a compact tree modelling the relationship between individual cells over time or space.

Depends R (>= 3.3), topGO

License Artistic-2.0

LazyData true

RoxygenNote 5.0.1

VignetteBuilder knitr

URL http://tsudalab.org

Imports topicmodels, slam, maptpx, igraph, xtable, gplots

Suggests BiocStyle, knitr, HSMMSingleCell, biomaRt, org.Hs.eg.db, Biobase, tools

biocViews ImmunoOncology, Sequencing, RNASeq, Clustering, GraphAndNetwork, Visualization, GeneExpression, GeneSetEnrichment, BiomedicalInformatics, CellBiology, FunctionalGenomics, SystemsBiology, GO, TimeCourse, Microarray

NeedsCompilation no

Author David duVerle [aut, cre], Koji Tsuda [aut]

Maintainer David duVerle <dave@cb.k.u-tokyo.ac.jp>

git_url https://git.bioconductor.org/packages/cellTree

git_branch RELEASE_3_14

git_last_commit 2a079ea

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`cellTree-package` *Inference and visualisation of Single-Cell RNA-seq Data data as a hierarchical tree structure*

Description

This packages computes a Latent Dirichlet Allocation (LDA) model of single-cell RNA-seq data and build a compact tree modelling the relationship between individual cells over time or space.

Details

A typical use-case will require you to run compute.lda on your expression data, to fit an LDA model, followed by compute.backbone.tree to generate a tree structure from the LDA model.

Plotting functions ct.plot.grouping and ct.plot.topics will then help you plot the tree structure with the desired annotations, while function cell.ordering.table will output an ordered table of cells, ranked by their position in the tree.

To get further information on each topic, you can run Gene Ontology enrichment tests, using compute.go.enrichment, plot the result tables as a graph using ct.plot.go.dag or render it with LaTeX, using go.results.to.latex.

cell.ordering.table *Ranking of cells according to backbone tree structure*

Description

Produces a table of input cells ranked by their position in the backbone tree.

Usage

```
cell.ordering.table(b.tree, write.to.tex.file = NULL)
```

Arguments

b.tree	An igraph backbone tree, as returned by compute.backbone.tree.
write.to.tex.file	Boolean (optional). If not NULL, outputs LaTeX version of table to file write.to.tex.file.

Value

List of all cells, ranked by position in backbone tree, along with topic information.

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)

# Recover sampling time (in days) for each cell:
library(HSMMSingleCell)
data(HSMM_sample_sheet)
days.factor = HSMM_sample_sheet$Hours
days = as.numeric(levels(days.factor))[days.factor]

# Compute near-optimal backbone tree:
b.tree = compute.backbone.tree(HSMM_lda_model, days)
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
temp.output = tempfile()
cell.ordering.table(b.tree, write.to.tex.file = temp.output)
```

compute.backbone.tree *Backbone Tree construction*

Description

Builds a 'backbone tree' from a fitted LDA model.

Usage

```
compute.backbone.tree(lda.results, grouping = NULL,
  start.group.label = NULL, absolute.width = 0, width.scale.factor = 1.2,
  outlier.tolerance.factor = 0.1, rooting.method = NULL, only.mst = FALSE,
  grouping.colors = NULL, merge.sequential.backbone = FALSE)
```

Arguments

`lda.results`	A fitted LDA model, as returned by `compute.lda`
`grouping`	An (optional) vector of labels for each cell in the `lda.results` object. E.g. a sampling times (numeric) or tissue categories.
`start.group.label`	If a `grouping` parameter is provided, you can optionally specify the starting group. If no `start.group.label` is specified and the `grouping` vector is numeric, the lowest value will automatically be selected. Otherwise, the group with lowest mean-squared-distance between cells is selected.
`absolute.width`	Numeric (optional). Distance threshold below which a cell vertex is considered to be attached to a backbone vertex (see paper for more details). By default, this threshold is computed dynamically, based on the distance distribution for each branch.
`width.scale.factor`	Numeric (optional). A scaling factor for the dynamically-computed distance threshold (ignored if `absolute.width` is provided). Higher values will result in less branches in the backbone tree, while lower values might lead to a large number of backbone branches.
`outlier.tolerance.factor`	Numeric (optional). Proportion of vertices, out of the total number of vertices divided by the total number of branches, that can be left at the end of the backbone tree-building algorithm.
`rooting.method`	String (optional). Method used to root the backbone tree. Must be either NULL or one of: 'longest.path', 'center.start.group' or 'average.start.group'. 'longest.path' picks one end of the longest shortest-path between two vertices. 'center.start.group' picks the vertex in the starting group with lowest mean-square-distance to the others. 'average.start.group' creates a new artificial vertex, as the average of all cells in the starting group. If no value is provided, the best method is picked based on the type of grouping and start group information available.
`only.mst`	If `TRUE`, returns a simple rooted minimum-spanning tree, instead of a backbone tree.
`grouping.colors`	(Optional) vector of RGB colors to be used for each grouping.
`merge.sequential.backbone`	(Optional) whether to merge sequential backbone vertices that are close enough. This will produce a more compact backbone tree, but at the cost of extra computing time.

Details

In order to easily visualise the structural and temporal relationship between cells, we introduced a special type of tree structure dubbed 'backbone tree', defined as such:

Considering a set of vertices V and a distance function over all pairs of vertices: $d : V \times V \rightarrow R^+$, we call *backbone tree* a graph, T with backbone B, such that:

- T is a tree with set of vertices V and edges E.
- B is a tree with set of vertices $V_B \subseteq V$ and edges $E_B \subseteq E$.
- All 'vertebrae' vertices of T: $v \in V \setminus V_B$ are connected by a single edge to the closest vertex in the set of backbone vertices $v_B^* \in V_B$. I.e: $v_B^* = argmin_{v_B \in V_B} d(v_B, v)$.
- For all vertices in $V \setminus V_B$ are less than distance δ to a vertex in the backbone tree B: $\forall v \in V \setminus V_B, \exists v_B \in V_B$ such that $d(v, v_b) \leq \delta$.

In this instance, we relax the last condition to cover only 'most' non-backbone vertices, allowing for a variable proportion of outliers at distance $> \delta$ from any vertices in V_B.

We can then define the 'optimal' backbone tree to be a backbone tree such that the sum of weighted edges in the backbone subtree E_B is minimal. Finding such a tree can be easily shown to be NP-Complete (by reduction to the Vertex Cover problem), but we developed a fast heuristic relying on Minimum Spanning Tree to produce a reasonable approximation.

The resulting quasi-optimal backbone tree (simply referred to as 'the' backbone tree thereafter) gives a clear hierarchical representation of the cell relationship: the objective function puts pressure on finding a (small) group of prominent cells (the backbone) that are good representatives of major steps in the cell evolution (in time or space), while remaining cells are similar enough to their closest representative for their difference to be ignored. Such a tree provides a very clear visualisation of overall cell differentiation paths (including potential differentiation into sub-types).

Value

A igraph object with either a minimum rooted spanning-tree (if `only.mst` is `TRUE`) or a quasi-optimal backbone tree connecting all input cells. Cell topic distribution, distances and branch order are added as vertex/edge/graph attributes.

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)

# Recover sampling time (in days) for each cell:
library(HSMMSingleCell)
data(HSMM_sample_sheet)
days.factor = HSMM_sample_sheet$Hours
days = as.numeric(levels(days.factor))[days.factor]

# Compute near-optimal backbone tree:
b.tree = compute.backbone.tree(HSMM_lda_model, days)
# Plot resulting tree with sampling time as a vertex group colour:
ct.plot.grouping(b.tree)
```

`compute.go.enrichment` *Gene Ontology enrichment analysis*

Description

Computes enrichment scores for Gene Ontology terms associated with genes in each topic.

Usage

```
compute.go.enrichment(lda.results, go.db, ontology.type = "BP",
  reformat.gene.names = FALSE, bonferroni.correct = TRUE,
  p.val.threshold = if (bonferroni.correct) 0.05 else 0.01,
  go.score.class = "weight01Score", dag.file.prefix = FALSE)
```

Arguments

`lda.results`	A fitted LDA model, as returned by `compute.lda`
`go.db`	String. Genome-wide annotation with GO mapping for the appropriate organism (e.g. **org.Mm.eg.db** or **org.Hs.eg.db**).
`ontology.type`	(optional). "BP" for Biological Process, "MF" for Molecular Function, and "CC" for Cellular Component.
`reformat.gene.names`	Boolean. If set to `TRUE`, converts all gene names to capitalised lowercase.
`bonferroni.correct`	Boolean. Unless set to `FALSE`, adjust statistical testing p-value threshold for multiple testing.
`p.val.threshold`	Numeric (optional). P-value significance threshold.
`go.score.class`	String (optional). Name of the scoring method to use for the Kolmogorov-Smirnov test (e.g. "weigth01Score" or "elimScore"). See **topGO** documentation for a complete list of scoring methods.
`dag.file.prefix`	String or `FALSE`. If not set to `FALSE`, plots individual subgraphs of significant terms for each topic using the string as filename prefix.

Value

Returns a named list object with ranked tables of significantly enriched GO terms for each topic ('all'), terms that only appear in each topic ('unique') and terms that appear in less than half of the other topics ('rare'). In addition the list object contains an igraph object with the full GO DAG, annotated with each term's p-value and the significance threshold adjusted for multiple testing (Bonferroni method).

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)


# Load GO mapping database for 'homo sapiens':
library(org.Hs.eg.db)
# Compute Cellular Component GO enrichment sets for each topic:
go.results = compute.go.enrichment(HSMM_lda_model, org.Hs.eg.db, ontology.type="CC", bonferroni.correct=TRUE, p.

# Print table of terms that are only significantly enriched in each topic:
print(go.results$unique)
```

`compute.lda` *LDA model inference*

Description

This function fits a Latent Dirichlet Allocation (LDA) to single-cell RNA-seq data.

Usage

```
compute.lda(data, method = "maptpx", k.topics = if (method == "maptpx") 2:15
  else 4, log.scale = TRUE, sd.filter = 0.5, tot.iter = if (method ==
  "Gibbs") 200 else 1e+06, tol = if (method == "maptpx") 0.05 else 10^-5)
```

Arguments

`data`	A matrix of (non-negative) RNA-seq expression levels where each row is a gene and each column is the cell sequenced.
`method`	LDA inference method to use. Can be any unique prefix of 'maptpx', 'Gibbs' or 'VEM' (defaults to 'maptpx')
`k.topics`	Integer (optional). Number of topics to fit in the model. If `method` is 'maptpx', `k.topics` can be a vector of possible topic numbers and the the best model (evaluated on Bayes factor vs a null single topic model) will be returned.
`log.scale`	Boolean (optional). Whether the data should be log-scaled.
`sd.filter`	Numeric or `FALSE` (optional). Standard-deviation threshold below which genes should be removed from the data (no filtering if set to `FALSE`).
`tot.iter, tol`	Numeric parameters (optional) forwarded to the chosen LDA inference method's contol class.

Details

Latent Dirichlet allocation (LDA) is a generative model that allows sets of observations to be explained by unobserved groups (topics) that explain why some parts of the data are similar [Blei, 2003]. Each topic is modelled as a (Dirichlet) distribution over observations and each set of observations is also modelled as a (Dirichlet) distribution over topics. In lieu of the traditional NLP context of word occurence counts in documents, our model uses RNA-seq observation counts in single cells. Three separate LDA inference methods can be used at the moment:

- Gibbs uses Collapsed Gibbs Sampling method (implemented by Xuan-Hieu Phan and co-authors in the **topicmodels** package [Phan, 2008]) to infer the parameters of the Dirichlet distributions for a given number of topics. It gives high accuracy but is very time-consuming to run on large number of cells and genes.
- VEM uses Variational Expectation-Maximisation (as described in [Hoffman, 2010]). This method tends to converge faster than Gibbs collapsed sampling, albeit with lower accuracy.
- maptpx uses the method described in [Taddy, 2011] and implemented in package **maptpx** to estimate the parameters of the topic model for increasing number of topics (using previous estimates as a starting point for larger topic numbers). The best model (/number of topics) is selected based on Bayes factor over the Null model. Although potentially less accurate, this method provides the fastest way to train and select from a large number of models, when the number of topics is not well known.

When in doubt, the function can be ran with its default parameter values and should produce a usable LDA model in reasonable time (using the 'maptpx' inference method). The model can be further refined for a specific number of topics with slower methods. While larger models (using large number of topics) might fit the data well, there is a high risk of overfitting and it is recommended to use the smallest possible number of topics that still explains the observations well. Anecdotally, a typical number of topics for cell differentiation data (from pluripotent to fully specialised) would seem to be around 4 or 5.

Value

A LDA model fitted for `data`, of class LDA-class (for methods 'Gibbs' or 'VEM') or topics (for 'maptpx')

References

- Blei, Ng, and Jordan. "Latent dirichlet allocation." the Journal of machine Learning research 3 (2003): 993-1022.
- Hoffman, Blei and Bach (2010). "Online Learning for Latent Dirichlet Allocation." In J Lafferty, CKI Williams, J Shawe-Taylor, R Zemel, A Culotta (eds.), Advances in Neural Information Processing Systems 23, pp. 856-864. MIT Press, Cambridge, MA.
- Hornik and Grün. "topicmodels: An R package for fitting topic models." Journal of Statistical Software 40.13 (2011): 1-30.
- Phan, Nguyen and Horiguchi. "Learning to classify short and sparse text & web with hidden topics from large-scale data collections." Proceedings of the 17th international conference on World Wide Web. ACM, 2008.
- Taddy. "On estimation and selection for topic models." arXiv preprint arXiv:1109.4518 (2011).

See Also

LDA, topics, LDA_Gibbscontrol-class, CTM_VEMcontrol-class

Examples

```
# Load skeletal myoblast RNA-Seq data from HSMMSingleCell package:
library(HSMMSingleCell)
data(HSMM_expr_matrix)

# Run LDA inference using 'maptpx' method for k = 4:
 lda.results = compute.lda(HSMM_expr_matrix, k.topics=4, method="maptpx")


# Run LDA inference using 'maptpx' method for number of topics k = 3 to 6:
 lda.results = compute.lda(HSMM_expr_matrix, k.topics=3:6, method="maptpx")

# Run LDA inference using 'Gibbs' [collapsed sampling] method for number of k = 4 topics:
 lda.results = compute.lda(HSMM_expr_matrix, k.topics=4, method="Gibbs")
```

---

ct.plot.go.dag *Gene Ontology enrichment sets plotting*

Description

Plots DAG of significantly enriched terms for all topics, along with ancestor nodes.

Usage

```
ct.plot.go.dag(go.results, up.generations = 2, only.topics = NULL,
  file.output = NULL, p.val.threshold = go.results$adjusted.p.threshold,
  only.unique = FALSE, topic.colors = rainbow(length(go.results$results)))
```

Arguments

`go.results`	GO Enrichment result list object, such as returned by `compute.go.enrichment`.
`up.generations`	Integer (optional). Number of generations above significant nodes to include in the subgraph.
`only.topics`	Integer vector (optional). If not `NULL`, vector of topics that should be included in the plot (otherwise all topic enrichment sets are used).
`file.output`	String (optional). If not `NULL`, pathname of file to write the plot to.
`p.val.threshold`	Numeric (optional). P-value treshold to use to select which terms should be plotted.
`only.unique`	Only display terms that are only significant for one of the topics.
`topic.colors`	RGB colour vector (optional). Colors to use for each topic.

Value

An igraph object with the annotated GO DAG.

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)

# Load GO mapping database for 'homo sapiens':
library(org.Hs.eg.db)


# Compute GO enrichment sets for each topic:
go.results = compute.go.enrichment(HSMM_lda_model, org.Hs.eg.db, bonferroni.correct=TRUE)

go.dag.subtree = ct.plot.go.dag(go.results, up.generations = 2)
```

---

ct.plot.grouping *Plot cell tree with grouping information*

Description

Plots a backbone tree (or MST) that was computed with `compute.backbone.tree`, displaying each cell's grouping.

Usage

```
ct.plot.grouping(tree, file.output = NULL, show.labels = FALSE,
  force.recompute.layout = FALSE, height = 20, width = 10,
  vertebrae.distance = 0, backbone.vertex.size = 0, vert.vertex.size = 0)
```

Arguments

`tree`	An igraph tree, as returned by `compute.backbone.tree`
`file.output`	String (optional). Path of a file where the plot should be saved in PDF format (rendered to screen if omitted).
`show.labels`	Boolean (optional). Whether to write each cell's row number next to its vertex.
`force.recompute.layout`	Boolean (optional). If set to `TRUE`, recomputes the graph's layout coordinates even when present.
`height`	Numeric (optional). Height and width (in inches) of the plot.
`width`	Numeric (optional). Height and width (in inches) of the plot.
`vertebrae.distance`	Numeric (optional). If non-zero: forces a specific plotting distance (in pixels) between backbone cells and related peripheral cells ('vertebrae').

backbone.vertex.size
: Numeric (optional). Diameter (in pixels) of backbone and vertebrae cell vertices.

vert.vertex.size
: Numeric (optional). Diameter (in pixels) of backbone and vertebrae cell vertices.

Value

An updated igraph object with x and y vertex coordinate attributes.

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)

# Recover sampling time (in days) for each cell:
library(HSMMSingleCell)
data(HSMM_sample_sheet)
days.factor = HSMM_sample_sheet$Hours
days = as.numeric(levels(days.factor))[days.factor]

# Compute near-optimal backbone tree:
b.tree = compute.backbone.tree(HSMM_lda_model, days)
# Plot resulting tree with sampling time as a vertex group colour:
ct.plot.grouping(b.tree)
```

---

ct.plot.heatmap *Gene Expression Heatmap*

Description

Plots a heatmap of gene expression, with cells ordered according to the structure computed by compute.backbone.tree.

Usage

```
ct.plot.heatmap(data, b.tree, log.scale = TRUE, sd.filter = 0.7,
  reorder.genes = TRUE)
```

Arguments

data	A matrix of (non-negative) RNA-seq expression levels where each row is a gene and each column is the cell sequenced.
b.tree	igraph object returned by compute.backbone.tree.
log.scale	Boolean (optional). Whether the data should be log-scaled.
sd.filter	Numeric or FALSE (optional). Standard-deviation threshold below which genes should be removed from the data (no filtering if set to FALSE).
reorder.genes	Boolean (optional). Whether the gene rows should be reordered using a dendrogram of their mean value.

Value

`data` object reordered according to the backbone tree, such as used to plot the heatmap.

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)

# Recover sampling time (in days) for each cell:
library(HSMMSingleCell)
data(HSMM_sample_sheet)
days.factor = HSMM_sample_sheet$Hours
days = as.numeric(levels(days.factor))[days.factor]

# Compute near-optimal backbone tree:
b.tree = compute.backbone.tree(HSMM_lda_model, days)
# Plot heatmap:
data(HSMM_expr_matrix)
ct.plot.heatmap(HSMM_expr_matrix[1:2000,], b.tree, reorder.genes=FALSE)
```

---

ct.plot.topics *Plot cell tree with topic distributions*

Description

Plots a backbone tree (or MST) that was computed with `compute.backbone.tree`, displaying each cell's topic distribution as a pie chart.

Usage

```
ct.plot.topics(tree, file.output = NULL, show.labels = FALSE,
  force.recompute.layout = FALSE, height = 20, width = 10,
  vertebrae.distance = 0, backbone.vertex.size = 0, vert.vertex.size = 0)
```

Arguments

`tree`	An igraph tree, as returned by `compute.backbone.tree`
`file.output`	String (optional). Path of a file where the plot should be saved in PDF format (rendered to screen if omitted).
`show.labels`	Boolean (optional). Whether to write each cell's row number next to its vertex.
`force.recompute.layout`	Boolean (optional). If set to `TRUE`, recomputes the graph's layout coordinates even when present.
`height, width`	Numeric (optional). Height and width (in inches) of the plot.
`vertebrae.distance`	Numeric (optional). If non-zero: forces a specific plotting distance (in pixels) between backbone cells and related peripheral cells ('vertebrae').

`backbone.vertex.size, vert.vertex.size`
: Numeric (optional). Diameter (in pixels) of backbone and vertebrae cell vertices.

Value

An updated igraph object with x and y vertex coordinate attributes.

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)

# Recover sampling time (in days) for each cell:
library(HSMMSingleCell)
data(HSMM_sample_sheet)
days.factor = HSMM_sample_sheet$Hours
days = as.numeric(levels(days.factor))[days.factor]

# Compute near-optimal backbone tree:
b.tree = compute.backbone.tree(HSMM_lda_model, days)
# Plot resulting tree with sampling time as a vertex group colour:
ct.plot.grouping(b.tree)
```

---

get.cell.dists *Cell Pairwise-Distance Matrix*

Description

Computes the pairwise distance between cells, based on the topic histograms form a fitted LDA model.

Usage

```
get.cell.dists(lda.results)
```

Arguments

`lda.results` A fitted LDA model, as returned by `compute.lda`

Details

Distances between histograms are computed using the Chi-square distance

Value

A square matrix of pairwise distance between cells in the input model.

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)

# Compute cell pairwise distances:
b.tree = get.cell.dists(HSMM_lda_model)
```

`go.results.to.latex` *LaTeX output for Gene Ontology Enrichment results*

Description

Outputs or writes result tables of Gene Ontology enrichment testing in LaTeX format.

Usage

```
go.results.to.latex(go.results, tex.file.name = "go_terms.tex",
  topic.colors = rainbow(length(go.results$all)))
```

Arguments

`go.results`	GO Enrichment result list object, such as returned by `compute.go.enrichment`.
`tex.file.name`	String (optional). If not `NULL`: name of the file to save to.
`topic.colors`	RGB colour vector (optional). Colors to use for each topic.

Value

GO enrichment results in LaTeX format

Examples

```
# Load pre-computed LDA model for skeletal myoblast RNA-Seq data from HSMMSingleCell package:
data(HSMM_lda_model)


# Load GO mapping database for 'homo sapiens':
library(org.Hs.eg.db)
# Compute GO enrichment sets for each topic:
go.results = compute.go.enrichment(HSMM_lda_model, org.Hs.eg.db, bonferroni.correct=TRUE)

# Output LaTeX tables for GO results
latex.table = go.results.to.latex(go.results, tex.file.name='go_results.tex')

# [Optional] compile LaTeX to PDF (need to have LaTeX binaries installed):
library('tools')
texi2pdf('go_results.tex')
library('Biobase')
openPDF('go_results.pdf')
```

`HSMM_lda_model` *Pre-computed LDA model for HSMMSingleCell data*

Description

Pre-computed LDA model for HSMMSingleCell data

Usage

```
HSMM_lda_model
```

Format

An object of class `list` of length 3.

Value

LDA model obtained by running `compute.lda` on the **HSMMSingleCell** package's data. This demo model can be used with all functions in this package that require a fitted LDA model, such as `compute.backbone.tree`.

See examples for `compute.lda` for more details.

See Also

compute.lda

Index