Package 'monocle'

October 12, 2016

Type Package

Title Analysis tools for single-cell expression experiments.

Version 1.6.2

Date 2013-11-19

Author Cole Trapnell

Maintainer Cole Trapnell <coletrap@uw.edu>

Description Monocle performs differential expression and time-series analysis for single-cell expression experiments. It orders individual cells according to progress through a biological process, without knowing ahead of time which genes define progress through that process. Monocle also performs differential expression analysis, clustering, visualization, and other useful tasks on single cell expression data. It is designed to work with RNA-Seq and qPCR data, but could be used with other types as well.

License Artistic-2.0

Depends R (>= 2.7.0), HSMMSingleCell (>= 0.101.5), Biobase, ggplot2(>= 0.9.3.1), splines, VGAM (>= 0.9-5), igraph(>= 0.7.0), plyr

Imports BiocGenerics, cluster, combinat, fastICA, grid, irlba, matrixStats, methods, parallel, reshape2, stats, utils, limma

VignetteBuilder knitr

Suggests knitr, Hmisc

Roxygen list(wrap = FALSE)

LazyData true

biocViews Sequencing, RNASeq, GeneExpression, DifferentialExpression, Infrastructure, DataImport, DataRepresentation, Visualization, Clustering, MultipleComparison, QualityControl

NeedsCompilation no

R topics documented:

`CellDataSet` *The CellDataSet class*

Description

The main class used by Monocle to hold single cell expression data. CellDataSet extends the basic Bioconductor ExpressionSet class.

Details

This class is initialized from a matrix of expression values Methods that operate on CellDataSet objects constitute the basic Monocle workflow.

Slots

`reducedDimS`: Matrix of class `"numeric"`, containing the source values computed by Independent Components Analysis.

`reducedDimW`: Matrix of class `"numeric"`, containing the whitened expression values computed during Independent Components Analysis.

`reducedDimA`: Matrix of class `"numeric"`, containing the weight values computed by Independent Components Analysis.

`minSpanningTree`: Object of class `"igraph"`, containing the minimum spanning tree used by Monocle to order cells according to progress through a biological process.

`cellPairwiseDistances`: Matrix of class `"numeric"`, containing the pairwise distances between cells in the reduced dimension space.

`expressionFamily`: Object of class `"vglmff"`, specifying the VGAM family function used for expression responses.

`lowerDetectionLimit`: A `"numeric"` value specifying the minimum expression level considered to be true expression.

`cellPairwiseDistances` *Retrieves a matrix capturing distances between each cell in the reduced-dimensionality space*

Description

Retrieves a matrix capturing distances between each cell in the reduced-dimensionality space

Usage

```
cellPairwiseDistances(cds)
```

Arguments

`cds`	expression data matrix for an experiment

Value

A square, symmetric matrix containing the distances between each cell in the reduced-dimensionality space.

Examples

```
## Not run:
data(HSMM)
D <- cellPairwiseDistances(HSMM)

## End(Not run)
```

`cellPairwiseDistances<-` *Sets the matrix containing distances between each pair of cells used by Monocle during cell ordering. Not intended to be called directly.*

Description

Sets the matrix containing distances between each pair of cells used by Monocle during cell ordering. Not intended to be called directly.

Usage

```
cellPairwiseDistances(cds) <- value
```

Arguments

`cds`	A CellDataSet object.
`value`	a square, symmetric matrix containing pairwise distances between cells.

Value

An updated CellDataSet object

`clusterGenes` *Plots the minimum spanning tree on cells.*

Description

Plots the minimum spanning tree on cells.

Usage

```
clusterGenes(expr_matrix, k, method = function(x) {     as.dist((1 -
  cor(t(x)))/2) }, ...)
```

Arguments

`expr_matrix`	a matrix of expression values to cluster together
`k`	how many clusters to create
`method`	the distance function to use during clustering
`...`	extra parameters to pass to pam() during clustering

Value

a pam cluster object

Examples

```
## Not run:
full_model_fits <- fitModel(HSMM[sample(nrow(fData(HSMM_filtered)), 100),],  modelFormulaStr="expression~sm.ns(
expression_curve_matrix <- responseMatrix(full_model_fits)
clusters <- clusterGenes(expression_curve_matrix, k=4)
plot_clusters(HSMM_filtered[ordering_genes,], clusters)

## End(Not run)
```

`compareModels` *Compare model fits*

Description

Performs likelihood ratio tests on nested vector generalized additive models

Usage

```
compareModels(full_models, reduced_models)
```

Arguments

`full_models`	a list of models, e.g. as returned by fitModels(), forming the numerators of the L.R.Ts.
`reduced_models`	a list of models, e.g. as returned by fitModels(), forming the denominators of the L.R.Ts.

Value

a data frame containing the p values and q-values from the likelihood ratio tests on the parallel arrays of models.

`detectGenes` *Sets the global expression detection threshold to be used with this CellDataSet. Counts how many cells each feature in a CellDataSet object that are detectably expressed above a minimum threshold. Also counts the number of genes above this threshold are detectable in each cell.*

Description

Sets the global expression detection threshold to be used with this CellDataSet. Counts how many cells each feature in a CellDataSet object that are detectably expressed above a minimum threshold. Also counts the number of genes above this threshold are detectable in each cell.

Usage

```
detectGenes(cds, min_expr = NULL)
```

Arguments

cds	the CellDataSet upon which to perform this operation
min_expr	the expression threshold

Value

an updated CellDataSet object

Examples

```
## Not run:
data(HSMM)
HSMM <- detectGenes(HSMM, min_expr=0.1)

## End(Not run)
```

---

differentialGeneTest *Tests each gene for differential expression as a function of progress through a biological process, or according to other covariates as specified.*

Description

Tests each gene for differential expression as a function of progress through a biological process, or according to other covariates as specified.

Usage

```
differentialGeneTest(cds,
  fullModelFormulaStr = "expression~sm.ns(Pseudotime, df=3)",
  reducedModelFormulaStr = "expression~1", cores = 1)
```

Arguments

cds	a CellDataSet object upon which to perform this operation
fullModelFormulaStr	a formula string specifying the full model in differential expression tests (i.e. likelihood ratio tests) for each gene/feature.
reducedModelFormulaStr	a formula string specifying the reduced model in differential expression tests (i.e. likelihood ratio tests) for each gene/feature.
cores	the number of cores to be used while testing each gene for differential expression

Value

a data frame containing the p values and q-values from the likelihood ratio tests on the parallel arrays of models.

`fitModel` *Fits a model for each gene in a CellDataSet object.*

Description

Fits a model for each gene in a CellDataSet object.

Usage

```
fitModel(cds, modelFormulaStr = "expression~sm.ns(Pseudotime, df=3)",
  cores = 1)
```

Arguments

`cds`	the CellDataSet upon which to perform this operation
`modelFormulaStr`	a formula string specifying the model to fit for the genes.
`cores`	the number of processor cores to be used during fitting.

Details

This function fits a Tobit-family vector generalized additive model (VGAM) from the VGAM package for each gene in a CellDataSet. The default formula string speficies that the (log transformed) expression values follow a Tobit distribution with upper and lower bounds specificed by max_expr and min_expr, respectively. By default, expression levels are modeled as smooth functions of the Pseudotime value of each cell. That is, expression is a function of progress through the biological process. More complicated formulae can be provided to account for additional covariates (e.g. day collected, genotype of cells, media conditions, etc).

Value

a list of VGAM model objects

`minSpanningTree` *Retrieve the minimum spanning tree generated by Monocle during cell ordering.*

Description

Retrieves the minimum spanning tree (MST) that Monocle constructs during orderCells(). This MST is mostly used in plot_spanning_tree to help assess the accuracy of Monocle\'s ordering.

Usage

```
minSpanningTree(cds)
```

Arguments

`cds`	expression data matrix for an experiment

Value

An igraph object representing the CellDataSet's minimum spanning tree.

Examples

```
## Not run:
data(HSMM)
T <- minSpanningTree(HSMM)

## End(Not run)
```

`minSpanningTree<-` *Sets the minimum spanning tree used by Monocle during cell ordering. Not intended to be called directly.*

Description

Sets the minimum spanning tree used by Monocle during cell ordering. Not intended to be called directly.

Usage

```
minSpanningTree(cds) <- value
```

Arguments

`cds`	A CellDataSet object.
`value`	an igraph object describing the minimum spanning tree.

Value

An updated CellDataSet object

`newCellDataSet` *Creates a new CellDateSet object.*

Description

Monocle requires that all data be housed in CellDataSet objects. CellDataSet extends Bioconductor's ExpressionSet class, and the same basic interface is supported. newCellDataSet() expects a matrix of relative expression values as its first argument, with rows as features (usually genes) and columns as cells. Per-feature and per-cell metadata can be supplied with the featureData and phenoData arguments, respectively. Use of these optional arguments is strongly encouraged. The CellDataSet also includes a VGAM expressionFamily object to encode the distribution that describes all genes.

Usage

```
newCellDataSet(cellData, phenoData = NULL, featureData = NULL,
  lowerDetectionLimit = 0.1, expressionFamily = VGAM::tobit(Lower =
  log10(lowerDetectionLimit), lmu = "identitylink"))
```

Arguments

`cellData`	expression data matrix for an experiment
`phenoData`	data frame containing attributes of individual cells
`featureData`	data frame containing attributes of features (e.g. genes)
`lowerDetectionLimit`	the minimum expression level that consistitutes true expression
`expressionFamily`	the VGAM family function to be used for expression response variables

Details

CellDataSet objects store a matrix of expression values. These values typically come from a program that calculates expression values from RNA-Seq reads such as Cufflinks. However, they might also be values from a single cell qPCR run or some other type of assay. By default, Monocle expects these values to be more or less log-normally distributed. If you log-transform the values before providing them to newCellDataSet, you will get bad results downstream. You can specify other VGAM family functions as an argument to this function, but this may result in undefined behavior. Expanded support for other family functions (e.g. the negative binomial) will likely appear in future versions of Monocle.

Value

a new CellDataSet object

Examples

```
## Not run:
sample_sheet_small <- read.delim("../data/sample_sheet_small.txt", row.names=1)
sample_sheet_small$Time <- as.factor(sample_sheet_small$Time)
gene_annotations_small <- read.delim("../data/gene_annotations_small.txt", row.names=1)
fpkm_matrix_small <- read.delim("../data/fpkm_matrix_small.txt")
pd <- new("AnnotatedDataFrame", data = sample_sheet_small)
fd <- new("AnnotatedDataFrame", data = gene_annotations_small)
HSMM <- new("CellDataSet", exprs = as.matrix(fpkm_matrix_small), phenoData = pd, featureData = fd)

## End(Not run)
```

`orderCells` *Orders cells according to progress through a learned biological process.*

Description

Orders cells according to progress through a learned biological process.

Usage

```
orderCells(cds, num_paths = 1, reverse = FALSE, root_cell = NULL)
```

Arguments

`cds`	the CellDataSet upon which to perform this operation
`num_paths`	the number of end-point cell states to allow in the biological process.
`reverse`	whether to reverse the beginning and end points of the learned biological process.
`root_cell`	the name of a cell to use as the root of the ordering tree.

Value

an updated CellDataSet object, in which phenoData contains values for State and Pseudotime for each cell

plot_clusters *Plots the minimum spanning tree on cells.*

Description

Plots the minimum spanning tree on cells.

Usage

```
plot_clusters(cds, clustering, drawSummary = TRUE, sumFun = mean_cl_boot,
  ncol = NULL, nrow = NULL, row_samples = NULL, callout_ids = NULL)
```

Arguments

cds	CellDataSet for the experiment
clustering	a clustering object produced by clusterCells
drawSummary	whether to draw the summary line for each cluster
sumFun	whether the function used to generate the summary for each cluster
ncol	number of columns used to layout the faceted cluster panels
nrow	number of columns used to layout the faceted cluster panels
row_samples	how many genes to randomly select from the data
callout_ids	a vector of gene names or gene ids to manually render as part of the plot

Value

a ggplot2 plot object

Examples

```
## Not run:
full_model_fits <- fitModel(HSMM_filtered[sample(nrow(fData(HSMM_filtered)), 100),],  modelFormulaStr="expressi
expression_curve_matrix <- responseMatrix(full_model_fits)
clusters <- clusterGenes(expression_curve_matrix, k=4)
plot_clusters(HSMM_filtered[ordering_genes,], clusters)

## End(Not run)
```

plot_genes_in_pseudotime *Plots expression for one or more genes as a function of pseudotime*

Description

Plots expression for one or more genes as a function of pseudotime

Usage

```
plot_genes_in_pseudotime(cds_subset, min_expr = NULL, cell_size = 0.75,
  nrow = NULL, ncol = 1, panel_order = NULL, color_by = "State",
  trend_formula = "adjusted_expression ~ sm.ns(Pseudotime, df=3)",
  label_by_short_name = TRUE)
```

Arguments

cds_subset	CellDataSet for the experiment
min_expr	the minimum (untransformed) expression level to use in plotted the genes.
cell_size	the size (in points) of each cell used in the plot
nrow	the number of rows used when laying out the panels for each gene's expression
ncol	the number of columns used when laying out the panels for each gene's expression
panel_order	the order in which genes should be layed out (left-to-right, top-to-bottom)
color_by	the cell attribute (e.g. the column of pData(cds)) to be used to color each cell
trend_formula	the model formula to be used for fitting the expression trend over pseudotime
label_by_short_name	label figure panels by gene_short_name (TRUE) or feature id (FALSE)

Value

a ggplot2 plot object

Examples

```
## Not run:
data(HSMM)
my_genes <- row.names(subset(fData(HSMM), gene_short_name %in% c("CDK1", "MEF2C", "MYH3")))
cds_subset <- HSMM[my_genes,]
plot_genes_in_pseudotime(cds_subset, color_by="Time")

## End(Not run)
```

plot_genes_jitter *Plots expression for one or more genes as a jittered, grouped points*

Description

Plots expression for one or more genes as a jittered, grouped points

Usage

```
plot_genes_jitter(cds_subset, grouping = "State", min_expr = 0.1,
  cell_size = 0.75, nrow = NULL, ncol = 1, panel_order = NULL,
  color_by = NULL, plot_trend = FALSE, label_by_short_name = TRUE)
```

Arguments

cds_subset	CellDataSet for the experiment
grouping	the cell attribute (e.g. the column of pData(cds)) to group cells by on the horizontal axis
min_expr	the minimum (untransformed) expression level to use in plotted the genes.
cell_size	the size (in points) of each cell used in the plot
nrow	the number of rows used when laying out the panels for each gene's expression
ncol	the number of columns used when laying out the panels for each gene's expression
panel_order	the order in which genes should be layed out (left-to-right, top-to-bottom)
color_by	the cell attribute (e.g. the column of pData(cds)) to be used to color each cell
plot_trend	whether to plot a trendline tracking the average expression across the horizontal axis.
label_by_short_name	label figure panels by gene_short_name (TRUE) or feature id (FALSE)

Value

a ggplot2 plot object

Examples

```
## Not run:
data(HSMM)
MYOG_ID1 <- HSMM[row.names(subset(fData(HSMM), gene_short_name %in% c("MYOG", "ID1"))),]
plot_genes_jitter(MYOG_ID1, grouping="Media", ncol=2)

## End(Not run)
```

`plot_genes_positive_cells` *Plots the number of cells expressing one or more genes as a barplot*

Description

Plots the number of cells expressing one or more genes as a barplot

Usage

```
plot_genes_positive_cells(cds_subset, grouping = "State", min_expr = 0.1,
  nrow = NULL, ncol = 1, panel_order = NULL, plot_as_fraction = TRUE,
  label_by_short_name = TRUE)
```

Arguments

`cds_subset`	CellDataSet for the experiment
`grouping`	the cell attribute (e.g. the column of pData(cds)) to group cells by on the horizontal axis
`min_expr`	the minimum (untransformed) expression level to use in plotted the genes.
`nrow`	the number of rows used when laying out the panels for each gene's expression
`ncol`	the number of columns used when laying out the panels for each gene's expression
`panel_order`	the order in which genes should be layed out (left-to-right, top-to-bottom)
`plot_as_fraction`	whether to show the percent instead of the number of cells expressing each gene
`label_by_short_name`	label figure panels by gene_short_name (TRUE) or feature id (FALSE)

Value

a ggplot2 plot object

Examples

```
## Not run:
data(HSMM)
MYOG_ID1 <- HSMM[row.names(subset(fData(HSMM), gene_short_name %in% c("MYOG", "ID1"))),]
plot_genes_positive_cells(MYOG_ID1, grouping="Media", ncol=2)

## End(Not run)
```

plot_spanning_tree *Plots the minimum spanning tree on cells.*

Description

Plots the minimum spanning tree on cells.

Usage

```
plot_spanning_tree(cds, x = 1, y = 2, color_by = "State",
  show_tree = TRUE, show_backbone = TRUE, backbone_color = "black",
  markers = NULL, show_cell_names = FALSE, cell_name_size = 1)
```

Arguments

cds	CellDataSet for the experiment
x	the column of reducedDimS(cds) to plot on the horizontal axis
y	the column of reducedDimS(cds) to plot on the vertical axis
color_by	the cell attribute (e.g. the column of pData(cds)) to map to each cell's color
show_tree	whether to show the links between cells connected in the minimum spanning tree
show_backbone	whether to show the diameter path of the MST used to order the cells
backbone_color	the color used to render the backbone.
markers	a gene name or gene id to use for setting the size of each cell in the plot
show_cell_names	draw the name of each cell in the plot
cell_name_size	the size of cell name labels

Value

a ggplot2 plot object

Examples

```
## Not run:
data(HSMM)
plot_spanning_tree(HSMM)
plot_spanning_tree(HSMM, color_by="Pseudotime", show_backbone=FALSE)
plot_spanning_tree(HSMM, markers="MYH3")

## End(Not run)
```

`reducedDimA` *Get the weights needed to lift cells back to high dimensional expression space.*

Description

Retrieves the weights that transform the cells' coordinates in the reduced dimension space back to the full (whitened) space.

Usage

```
reducedDimA(cds)
```

Arguments

`cds`	A CellDataSet object.

Value

A matrix that when multiplied by a reduced-dimension set of coordinates for the CellDataSet, recovers a matrix in the full (whitened) space

Examples

```
## Not run:
data(HSMM)
A <- reducedDimA(HSMM)

## End(Not run)
```

`reducedDimA<-` *Get the weights needed to lift cells back to high dimensional expression space.*

Description

Sets the weights transform the cells' coordinates in the reduced dimension space back to the full (whitened) space.

Usage

```
reducedDimA(cds) <- value
```

Arguments

`cds`	A CellDataSet object.
`value`	A whitened expression data matrix

Value

An updated CellDataSet object

`reducedDimS` *Retrieves the coordinates of each cell in the reduced-dimensionality space generated by calls to reduceDimension.*

Description

Reducing the dimensionality of the expression data is a core step in the Monocle workflow. After you call reduceDimension(), this function will return the new coordinates of your cells in the reduced space.

Usage

```
reducedDimS(cds)
```

Arguments

`cds`	A CellDataSet object.

Value

A matrix, where rows are cell coordinates and columns correspond to dimensions of the reduced space.

Examples

```
## Not run:
data(HSMM)
S <- reducedDimS(HSMM)

## End(Not run)
```

`reducedDimS<-` *Set embedding coordinates of each cell in a CellDataSet.*

Description

This function sets the coordinates of each cell in a new (reduced-dimensionality) space. Not intended to be called directly.

Usage

```
reducedDimS(cds) <- value
```

Arguments

cds	A CellDataSet object.
value	A matrix of coordinates specifying each cell's position in the reduced-dimensionality space.

Value

An update CellDataSet object

---

reducedDimW *Get the whitened expression values for a CellDataSet.*

Description

Retrieves the expression values for each cell (as a matrix) after whitening during dimensionality reduction.

Usage

```
reducedDimW(cds)
```

Arguments

cds	A CellDataSet object.

Value

A matrix, where each row is a set of whitened expression values for a feature and columns are cells.

Examples

```
## Not run:
data(HSMM)
W <- reducedDimW(HSMM)

## End(Not run)
```

reducedDimW<- *Get the whitened expression values for a CellDataSet.*

Description

Sets the whitened expression values for each cell prior to dimensionality reduction. Not intended to be called directly.

Usage

```
reducedDimW(cds) <- value
```

Arguments

cds	A CellDataSet object.
value	A whitened expression data matrix

Value

An updated CellDataSet object

reduceDimension *Computes a projection of a CellDataSet object into a lower dimensional space*

Description

Computes a projection of a CellDataSet object into a lower dimensional space

Usage

```
reduceDimension(cds, max_components = 2, use_irlba = TRUE,
  pseudo_expr = 1, batch = NULL, covariates = NULL, ...)
```

Arguments

cds	the CellDataSet upon which to perform this operation
max_components	the dimensionality of the reduced space
use_irlba	Whether to use the IRLBA package for ICA reduction.
pseudo_expr	amount to increase expression values before dimensionality reduction
batch	a vector of labels specifying batch for each cell, the effects of which will be removed prior to dimensionality reduction.
covariates	a numeric vector or matrix specifying continuous effects to be removed prior to dimensionality reduction
...	additional arguments to pass to the dimensionality reduction function

Details

Currently, Monocle supports dimensionality reduction with Independent Component Analysis (ICA).

Value

an updated CellDataSet object

---

responseMatrix *Response values*

Description

Generates a matrix of response values for a set of fitted models

Usage

```
responseMatrix(models)
```

Arguments

models	a list of models, e.g. as returned by fitModels()

Value

a matrix where each row is a vector of response values for a particular feature's model, and columns are cells.

---

selectNegentropyGenes *Filter genes with extremely high or low negentropy*

Description

Filter genes with extremely high or low negentropy

Usage

```
selectNegentropyGenes(cds, lower_negentropy_bound = "0%",
  upper_negentropy_bound = "99%", expression_lower_thresh = 0.1,
  expression_upper_thresh = Inf)
```

Arguments

cds	a CellDataSet object upon which to perform this operation
lower_negentropy_bound	the centile below which to exclude to genes
upper_negentropy_bound	the centile above which to exclude to genes
expression_lower_thresh	the expression level below which to exclude genes used to determine negentropy
expression_upper_thresh	the expression level above which to exclude genes used to determine negentropy

Value

a vector of gene names

Examples

```
## Not run:
reasonableNegentropy <- selectNegentropyGenes(HSMM, "07%", "95%", 1, 100)

## End(Not run)
```

---

setOrderingFilter *Sets the features (e.g. genes) to be used for ordering cells in pseudotime.*

Description

Sets the features (e.g. genes) to be used for ordering cells in pseudotime.

Usage

```
setOrderingFilter(cds, ordering_genes)
```

Arguments

cds	the CellDataSet upon which to perform this operation
ordering_genes	a vector of feature ids (from the CellDataSet's featureData) used for ordering cells

Value

an updated CellDataSet object

Index