Package ‘dynplot’

December 7, 2021

Type Package

Title Visualising Single-Cell Trajectories

Version 1.1.2

Description Visualise a single-cell trajectory as a graph or dendrogram, as a dimensionality reduction or heatmap of the expression data, or a comparison between two trajectories as a pairwise scatterplot or dimensionality reduction projection. Saelens and Cannoodt et al. (2019) <doi:10.1038/s41587-019-0071-9>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL `https://github.com/dynverse/dynplot`

BugReports `https://github.com/dynverse/dynplot/issues`

Depends R (>= 3.0.0)

Imports assertthat, dplyr, dynutils (>= 1.0.2), dynfeature (>= 1.0.0), dyndimred (>= 1.0.0), dynwrap (>= 1.0.0), GA, ggforce, ggplot2 (>= 3.0), ggraph (>= 2.0), ggrepel, igraph, MASS, methods, patchwork, purrr, reshape2, tibble, tidyr, tidygraph, vipor

Suggests covr, hexbin, knitr, RColorBrewer, rje, rmarkdown, testthat (>= 3.0.0), uwot

VignetteBuilder knitr

Collate 'milestone_palette.R' 'add_milestone_coloring.R' 'add_cell_coloring.R' 'add_density_coloring.R' 'data.R' 'dummy_proofing.R' 'expect_ggplot.R' 'is_colour_vector.R' 'linearise_cells.R' 'mix_colors.R' 'optimize_order.R' 'package.R' 'plot_dendro.R' 'project_waypoints.R' 'plot_dimred.R' 'plot_edge_flips.R' 'plot_graph.R' 'plot_heatmap.R' 'plot_linearised_comparison.R' 'plot_onedim.R' 'plot_strip.R' 'plot_topology.R' 'theme_clean.R'

NeedsCompilation no

Author Robrecht Cannoodt [aut, cre, cph]
(<https://orcid.org/0000-0003-3641-729X>),
Wouter Saelens [aut] (<https://orcid.org/0000-0002-7114-6248>)

Maintainer Robrecht Cannoodt <rcannood@gmail.com>

Repository CRAN

Date/Publication 2021-12-07 19:00:02 UTC

R topics documented:

`add_cell_coloring` *Add colouring to a set of cells.*

Description

The cells can be coloured by a grouping (clustering), according to a feature (gene expression), closest milestone, or pseudotime from the root of the trajectory.

Usage

```
add_cell_coloring(
  cell_positions,
  color_cells = c("auto", "none", "grouping", "feature", "milestone", "pseudotime"),
  trajectory,
  grouping = NULL,
  groups = NULL,
  feature_oi = NULL,
  expression_source = "expression",
  pseudotime = NULL,
  color_milestones = c("auto", "given", "cubeHelix", "Set3", "rainbow"),
  milestones = NULL,
  milestone_percentages = NULL
)
```

Arguments

`cell_positions`	The positions of the cells, represented by a tibble. Must contain column `cell_id` (character) and may contain columns `from`, `to`, `pseudotime`, depending on the value of `color_cells`.
`color_cells`	How to color the cells. • `"auto"`: Try to figure out how to color cells depending on whether one of the `grouping`, `feature_io`, `milestones` or `pseudotime` parameters are defined. • `"none"`: Cells are not coloured. • `"grouping"`: Cells are coloured according to a grouping (e.g. clustering). Either the `grouping` parameter or `trajectory$grouping` must be a named character vector. • `"feature"`: Cells are coloured according to the values of a given feature (e.g. gene expression). Either the `expression_source` parameter or `get_expression(trajectory)` must be a matrix. Parameter `feature_oi` must also be defined. • `"milestone"` (recommended): Cells are coloured according their position in the trajectory. The positioning of the cells are determined by parameter `milestone_percentages` or else by `trajectory$milestone_percentages`. The colours of the milestones can be determined automatically or can be specified by passing a tibble containing character columns `milestone_id` and `color` (See `add_milestone_coloring()` for help in constructing this object). • `"pseudotime"`: Cells are coloured according to the pseudotime value from the root.
`trajectory`	A dynwrap trajectory.
`grouping`	A grouping of the cells (e.g. clustering) as a named character vector.
`groups`	A tibble containing character columns `group_id` and `color`. If `NULL`, this object is inferred from the `grouping` itself.
`feature_oi`	The name of a feature to use for colouring the cells.

`expression_source`
: Source of the feature expression, defaults to `get_expression(trajectory)`.

`pseudotime`
: The pseudotime from the root of the trajectory to the cells as a named numeric vector.

`color_milestones`
: Which palette to use for colouring the milestones

  - `auto`: Determine colours automatically. If `color` is already specified in milestones tibble, this will be used. Otherwise, the colour scheme is determined by `milestone_palette_list$auto`.
  - `given`: The `milestones` object already contains a column `color`.
  - `cubeHelix`: Use the `rje::cubeHelix()` palette.
  - `Set3`: Use the `RColorBrewer::brewer.pal(name = "Set3")` palette.
  - `rainbow`: Use the `grDevices::rainbow()` palette.

`milestones`
: Tibble containing the column `milestone_id` (character). If `color_milestones` is set to `"given"`, this tibble should also contain a column `color` (character), containing colour hex codes (e.g. `"#123456"`).

`milestone_percentages`
: The milestone percentages.

Value

A named list with following objects:

- cell_positions: The `trajectory$progressions` object with a `color` column added.
- color_scale: A ggplot colour scale to add to the downstream ggplot.
- fill_scale: A ggplot fill scale to add to the downstream ggplot.
- color_cells: The input `color_cells` value, except `"auto"` will have been replaced depending on which other parameters were passed.

`add_density_coloring` *Color cells using a background density*

Description

Color cells using a background density

Usage

```
add_density_coloring(
  cell_positions,
  color_density = c("none", "grouping", "feature"),
  trajectory,
  grouping = NULL,
  groups = NULL,
  feature_oi = NULL,
  expression_source = "expression",
  padding = 0.1,
  nbins = 1000,
  bw = 0.2,
  density_cutoff = 0.3,
  density_cutoff_label = density_cutoff/10
)
```

Arguments

`cell_positions`	The positions of the cells in 2D. Must be a tibble with character column `cell_id` and numeric columns `comp_1` and `comp_2`.
`color_density`	How to color density, can be "none", "grouping", or "feature".
`trajectory`	A dynwrap trajectory.
`grouping`	A grouping of the cells (e.g. clustering) as a named character vector.
`groups`	A tibble containing character columns `group_id` and `color`. If `NULL`, this object is inferred from the `grouping` itself.
`feature_oi`	The name of a feature to use for colouring the cells.
`expression_source`	Source of the feature expression, defaults to `get_expression(trajectory)`.
`padding`	The padding in the edges to the plot, relative to the size of the plot.
`nbins`	Number of bins for calculating the density.
`bw`	Bandwidth, relative to the size of the plot.
`density_cutoff`	Cutoff for density, the lower the larger the areas.
`density_cutoff_label`	Cutoff for density for labeling, the lower the further way from cells.

Value

A named list with objects:

- polygon: A layer to add to the ggplot.
- scale: A scale to add to the ggplot.

`add_milestone_coloring` *Add colouring to a set of milestones.*

Description

Add colouring to a set of milestones.

Usage

```
add_milestone_coloring(
  milestones = NULL,
  color_milestones = c("auto", "given", "cubeHelix", "Set3", "rainbow")
)
```

Arguments

`milestones`	Tibble containing the column `milestone_id` (character). If `color_milestones` is set to `"given"`, this tibble should also contain a column `color` (character), containing colour hex codes (e.g. `"#123456"`).
`color_milestones`	Which palette to use for colouring the milestones • `auto`: Determine colours automatically. If `color` is already specified in milestones tibble, this will be used. Otherwise, the colour scheme is determined by `milestone_palette_list$auto`. • `given`: The `milestones` object already contains a column `color`. • `cubeHelix`: Use the `rje::cubeHelix()` palette. • `Set3`: Use the `RColorBrewer::brewer.pal(name = "Set3")` palette. • `rainbow`: Use the `grDevices::rainbow()` palette.

Value

A tibble containing the input character column `milestone_id` and a character column `color` containing colour hex-codes (e.g. `"#123456"`).

`dynplot` *dynplot: Plotting Single-Cell Trajectories*

Description

Visualise a single-cell trajectory as a graph or dendrogram, as a dimensionality reduction or heatmap of the expression data, or a comparison between two trajectories as a pairwise scatterplot or dimensionality reduction projection.

`empty_plot` *Create an empty plot for spacing*

Description

Create an empty plot for spacing

Usage

```
empty_plot()
```

Value

An empty ggplot2.

Examples

```
empty_plot()
```

`example_bifurcating` *An example bifurcating dataset*

Description

An example bifurcating dataset

Usage

```
example_bifurcating
```

Format

An object of class `dynwrap::with_prior` (inherits from `dynwrap::with_expression`, `dynwrap::with_cell_waypoints`, `dynwrap::with_trajectory`, `dynwrap::data_wrapper`, `list`) of length 20.

`example_disconnected` *An example disconnected dataset*

Description

An example disconnected dataset

Usage

```
example_disconnected
```

Format

An object of class `dynwrap::with_prior` (inherits from `dynwrap::with_expression`, `dynwrap::with_cell_waypoints`, `dynwrap::with_trajectory`, `dynwrap::data_wrapper`, `list`) of length 20.

`example_linear` *An example linear dataset*

Description

An example linear dataset

Usage

```
example_linear
```

Format

An object of class `dynwrap::with_prior` (inherits from `dynwrap::with_expression`, `dynwrap::with_cell_waypoints`, `dynwrap::with_trajectory`, `dynwrap::data_wrapper`, `list`) of length 20.

`example_tree` *An example tree dataset*

Description

An example tree dataset

Usage

```
example_tree
```

Format

An object of class `dynwrap::with_prior` (inherits from `dynwrap::with_expression`, `dynwrap::with_cell_waypoints`, `dynwrap::with_trajectory`, `dynwrap::data_wrapper`, `list`) of length 20.

`linearise_cells` *Prepare a trajectory for linearised visualisation.*

Description

This is an internal function and should probably not be used manually.

Usage

```
linearise_cells(
  trajectory,
  margin = 0.05,
  no_margin_between_linear = TRUE,
  one_edge = FALSE,
  equal_cell_width = FALSE
)
```

Arguments

`trajectory`	A dynwrap trajectory.
`margin`	A margin between trajectory segments.
`no_margin_between_linear`	Whether to add a margin only when a branch occurs.
`one_edge`	Whether or not to assign each cell to one cell only. This can occur when a cell is on a branching point, or in between multiple edges.
`equal_cell_width`	Whether or not to space segments according to cell count.

Value

A named list with values:

- `milestone_network`: A linearised version of `trajectory$milestone_network` with extra columns: `add_margin`, `n_margins`, `cumstart`, `cumend`, `edge_id`.
- `progressions`: A linearised version of `trajectory$progressions` with extra columns: `percentage2`, `length`, `directed`, `add_margin`, `n_margins`, `cumstart`, `cumend`, `edge_id`, `cumpercentage`.
- `margin`: The used margin (numeric).

Examples

```
linearise_cells(example_bifurcating)
```

`milestone_palette` *Get the names of valid color palettes*

Description

Get the names of valid color palettes

Usage

```
milestone_palette(name, n)

get_milestone_palette_names()
```

Arguments

`name`	The name of the palette. Must be one of `"cubeHelix"`, `"Set3"`, or `"rainbow"`.
`n`	The number of colours to be in the palette.

Value

The names of supported palettes.

Examples

```
get_milestone_palette_names()
```

`plot_dendro` *Plot a trajectory as a dendrogram*

Description

Plot a trajectory as a dendrogram

Usage

```
plot_dendro(
  trajectory,
  color_cells = c("auto", "none", "grouping", "feature", "milestone", "pseudotime"),
  grouping = NULL,
  groups = NULL,
  feature_oi = NULL,
  expression_source = "expression",
  pseudotime = NULL,
  color_milestones = c("auto", "given", "cubeHelix", "Set3", "rainbow"),
  milestones = NULL,
  milestone_percentages = NULL,
  alpha_cells = 1,
  size_cells = 2.5,
  border_radius_percentage = 0.1,
  diag_offset = 0.05,
  y_offset = 0.2,
  arrow = grid::arrow(type = "closed")
)
```

Arguments

`trajectory` A dynwrap trajectory.

`color_cells` How to color the cells.

- `"auto"`: Try to figure out how to color cells depending on whether one of the `grouping`, `feature_io`, `milestones` or `pseudotime` parameters are defined.
- `"none"`: Cells are not coloured.
- `"grouping"`: Cells are coloured according to a grouping (e.g. clustering). Either the `grouping` parameter or `trajectory$grouping` must be a named character vector.
- `"feature"`: Cells are coloured according to the values of a given feature (e.g. gene expression). Either the `expression_source` parameter or `get_expression(trajectory)` must be a matrix. Parameter `feature_oi` must also be defined.
- `"milestone"` (recommended): Cells are coloured according their position in the trajectory. The positioning of the cells are determined by parameter `milestone_percentages` or else by `trajectory$milestone_percentages`. The colours of the milestones can be determined automatically or can be specified by passing a tibble containing character columns `milestone_id` and `color` (See `add_milestone_coloring()` for help in constructing this object).
- `"pseudotime"`: Cells are coloured according to the pseudotime value from the root.

`grouping` A grouping of the cells (e.g. clustering) as a named character vector.

`groups` A tibble containing character columns `group_id` and `color`. If `NULL`, this object is inferred from the `grouping` itself.

`feature_oi` The name of a feature to use for colouring the cells.

`expression_source`
Source of the feature expression, defaults to `get_expression(trajectory)`.

`pseudotime` The pseudotime from the root of the trajectory to the cells as a named numeric vector.

`color_milestones`
Which palette to use for colouring the milestones

- `auto`: Determine colours automatically. If `color` is already specified in milestones tibble, this will be used. Otherwise, the colour scheme is determined by `milestone_palette_list$auto`.

- `given`: The `milestones` object already contains a column `color`.
- `cubeHelix`: Use the `rje::cubeHelix()` palette.
- `Set3`: Use the `RColorBrewer::brewer.pal(name = "Set3")` palette.
- `rainbow`: Use the `grDevices::rainbow()` palette.

`milestones`	Tibble containing the column `milestone_id` (character). If `color_milestones` is set to `"given"`, this tibble should also contain a column `color` (character), containing colour hex codes (e.g. `"#123456"`).
`milestone_percentages`	The milestone percentages.
`alpha_cells`	The alpha of the cells
`size_cells`	The size of the cells
`border_radius_percentage`	The fraction of the radius that is used for the border
`diag_offset`	The x-offset (percentage of the edge lenghts) between milestones
`y_offset`	The size of the quasirandom cell spreading in the y-axis
`arrow`	The type and size of arrow in case of directed trajectories. Set to NULL to remove arrow altogether.

Value

A dendrogram ggplot of the trajectory.

Examples

```
data(example_tree)
plot_dendro(example_tree)
plot_dendro(example_tree, color_cells = "pseudotime")
plot_dendro(
  example_tree,
  color_cells = "grouping",
  grouping = dynwrap::group_onto_nearest_milestones(example_tree)
)
```

`plot_dimred` *Plot a trajectory in a (given) dimensionality reduction*

Description

Plot a trajectory in a (given) dimensionality reduction

Usage

```
plot_dimred(
  trajectory,
 color_cells = c("auto", "none", "grouping", "feature", "milestone", "pseudotime"),
  dimred = ifelse(dynwrap::is_wrapper_with_dimred(trajectory), NA,
    dyndimred::dimred_landmark_mds),
  plot_trajectory = dynwrap::is_wrapper_with_trajectory(trajectory) &&
    !plot_milestone_network,
  plot_milestone_network = FALSE,
  label_milestones = dynwrap::is_wrapper_with_milestone_labelling(trajectory),
  alpha_cells = 1,
  size_cells = 2.5,
  border_radius_percentage = 0.1,
  size_milestones = 6,
  size_transitions = 2,
  hex_cells = ifelse(length(trajectory$cell_ids) > 10000, 100, FALSE),
  grouping = NULL,
  groups = NULL,
  feature_oi = NULL,
  color_milestones = c("auto", "given", "cubeHelix", "Set3", "rainbow"),
  milestones = NULL,
  milestone_percentages = NULL,
  pseudotime = NULL,
  expression_source = "expression",
  arrow = grid::arrow(type = "closed", length = unit(0.1, "inches")),
  color_density = c("none", "grouping", "feature"),
  padding = 0.1,
  nbins = 1000,
  bw = 0.2,
  density_cutoff = 0.3,
  density_cutoff_label = density_cutoff/10,
  waypoints = dynwrap::select_waypoints(trajectory),
  trajectory_projection_sd = sum(trajectory$milestone_network$length) * 0.05,
  color_trajectory = "none"
)
```

Arguments

`trajectory`	A dynwrap trajectory.
`color_cells`	How to color the cells. • `"auto"`: Try to figure out how to color cells depending on whether one of the `grouping`, `feature_io`, `milestones` or `pseudotime` parameters are defined. • `"none"`: Cells are not coloured. • `"grouping"`: Cells are coloured according to a grouping (e.g. clustering). Either the `grouping` parameter or `trajectory$grouping` must be a named character vector.

- `"feature"`: Cells are coloured according to the values of a given feature (e.g. gene expression). Either the `expression_source` parameter or `get_expression(trajectory)` must be a matrix. Parameter `feature_oi` must also be defined.
- `"milestone"` (recommended): Cells are coloured according their position in the trajectory. The positioning of the cells are determined by parameter `milestone_percentages` or else by `trajectory$milestone_percentages`. The colours of the milestones can be determined automatically or can be specified by passing a tibble containing character columns `milestone_id` and `color` (See `add_milestone_coloring()` for help in constructing this object).
- `"pseudotime"`: Cells are coloured according to the pseudotime value from the root.

`dimred` Can be

- A function which will perform the dimensionality reduction, see `dyndimred::list_dimred_method`
- A matrix with the dimensionality reduction, with cells in rows and dimensions (*comp_1*, *comp_2*, ...) in columns

`plot_trajectory`
Whether to plot the projected trajectory on the dimensionality reduction

`plot_milestone_network`
Whether to plot the projected milestone network on the dimensionality reduction

`label_milestones`
How to label the milestones. Can be TRUE (in which case the labels within the trajectory will be used), "all" (in which case both given labels and milestone_ids will be used), a named character vector, or FALSE

`alpha_cells` The alpha of the cells

`size_cells` The size of the cells

`border_radius_percentage`
The fraction of the radius that is used for the border

`size_milestones`
The size of the milestones

`size_transitions`
The size of the trajectory segments

`hex_cells` The number of hexes to use, to avoid overplotting points. Default is FALSE if number of cells <= 10000.

`grouping` A grouping of the cells (e.g. clustering) as a named character vector.

`groups` A tibble containing character columns `group_id` and `color`. If `NULL`, this object is inferred from the `grouping` itself.

`feature_oi` The name of a feature to use for colouring the cells.

`color_milestones`
Which palette to use for colouring the milestones

- `auto`: Determine colours automatically. If `color` is already specified in milestones tibble, this will be used. Otherwise, the colour scheme is determined by `milestone_palette_list$auto`.

- `given`: The `milestones` object already contains a column `color`.
- `cubeHelix`: Use the `rje::cubeHelix()` palette.
- `Set3`: Use the `RColorBrewer::brewer.pal(name = "Set3")` palette.
- `rainbow`: Use the `grDevices::rainbow()` palette.

`milestones`	Tibble containing the column `milestone_id` (character). If `color_milestones` is set to `"given"`, this tibble should also contain a column `color` (character), containing colour hex codes (e.g. `"#123456"`).
`milestone_percentages`	The milestone percentages.
`pseudotime`	The pseudotime from the root of the trajectory to the cells as a named numeric vector.
`expression_source`	Source of the expression
`arrow`	The type and size of arrow in case of directed trajectories. Set to NULL to remove arrow altogether.
`color_density`	How to color density, can be "none", "grouping", or "feature".
`padding`	The padding in the edges to the plot, relative to the size of the plot.
`nbins`	Number of bins for calculating the density.
`bw`	Bandwidth, relative to the size of the plot.
`density_cutoff`	Cutoff for density, the lower the larger the areas.
`density_cutoff_label`	Cutoff for density for labeling, the lower the further way from cells.
`waypoints`	The waypoints to use for projecting. Can by generated using `dynwrap::select_waypoints()`.
`trajectory_projection_sd`	The standard deviation of the Gaussian kernel to be used for projecting the trajectory. This is in the order of magnitude as the lengths of the milestone_network. The lower, the more closely the trajectory will follow the cells.
`color_trajectory`	How to color the trajectory, can be "nearest" for coloring to nearest cell, or "none".

Value

A dimensionality reduction ggplot of the data.

Examples

```
data(example_bifurcating)
plot_dimred(example_bifurcating)


# plotting with umap
if (requireNamespace("uwot", quietly = TRUE)) {
  plot_dimred(example_bifurcating, dimred = dyndimred::dimred_umap)
}
```

```
# using a custom dimred
dimred <- dyndimred::dimred_mds(example_bifurcating$expression)
plot_dimred(example_bifurcating, dimred = dimred)

# coloring cells by pseudotime
plot_dimred(example_bifurcating, color_cells = "pseudotime")

# coloring cells by cluster
plot_dimred(
  example_bifurcating,
  color_density = "grouping",
  grouping = dynwrap::group_onto_nearest_milestones(example_bifurcating)
)
```

---

plot_graph *Plot a trajectory as a graph*

Description

Plot a trajectory as a graph

Usage

```
plot_graph(
  trajectory,
 color_cells = c("auto", "none", "grouping", "feature", "milestone", "pseudotime"),
  color_milestones = c("auto", "given", "cubeHelix", "Set3", "rainbow"),
  grouping = NULL,
  groups = NULL,
  feature_oi = NULL,
  pseudotime = NULL,
  expression_source = "expression",
  milestones = NULL,
  milestone_percentages = NULL,
  size_trajectory = 3,
  size_milestones = 8,
  alpha_cells = 1,
  size_cells = 2.5,
  border_radius_percentage = 0.1,
  arrow = grid::arrow(length = grid::unit(1, "cm"), type = "closed"),
  label_milestones = dynwrap::is_wrapper_with_milestone_labelling(trajectory),
  plot_milestones = FALSE,
  adjust_weights = FALSE
)
```

Arguments

`trajectory` The trajectory as created by `infer_trajectory()` or `add_trajectory()`

`color_cells` How to color the cells.

- `"auto"`: Try to figure out how to color cells depending on whether one of the `grouping`, `feature_io`, `milestones` or `pseudotime` parameters are defined.
- `"none"`: Cells are not coloured.
- `"grouping"`: Cells are coloured according to a grouping (e.g. clustering). Either the `grouping` parameter or `trajectory$grouping` must be a named character vector.
- `"feature"`: Cells are coloured according to the values of a given feature (e.g. gene expression). Either the `expression_source` parameter or `get_expression(trajectory)` must be a matrix. Parameter `feature_oi` must also be defined.
- `"milestone"` (recommended): Cells are coloured according their position in the trajectory. The positioning of the cells are determined by parameter `milestone_percentages` or else by `trajectory$milestone_percentages`. The colours of the milestones can be determined automatically or can be specified by passing a tibble containing character columns `milestone_id` and `color` (See `add_milestone_coloring()` for help in constructing this object).
- `"pseudotime"`: Cells are coloured according to the pseudotime value from the root.

`color_milestones`
Which palette to use for colouring the milestones

- `auto`: Determine colours automatically. If `color` is already specified in milestones tibble, this will be used. Otherwise, the colour scheme is determined by `milestone_palette_list$auto`.
- `given`: The `milestones` object already contains a column `color`.
- `cubeHelix`: Use the `rje::cubeHelix()` palette.
- `Set3`: Use the `RColorBrewer::brewer.pal(name = "Set3")` palette.
- `rainbow`: Use the `grDevices::rainbow()` palette.

`grouping` A grouping of the cells (e.g. clustering) as a named character vector.

`groups` A tibble containing character columns `group_id` and `color`. If `NULL`, this object is inferred from the `grouping` itself.

`feature_oi` The name of a feature to use for colouring the cells.

`pseudotime` The pseudotime from the root of the trajectory to the cells as a named numeric vector.

`expression_source`
Source of the feature expression, defaults to `get_expression(trajectory)`.

`milestones` Tibble containing the column `milestone_id` (character). If `color_milestones` is set to `"given"`, this tibble should also contain a column `color` (character), containing colour hex codes (e.g. `"#123456"`).

`milestone_percentages`	The milestone percentages.
`size_trajectory`	The size of the transition lines between milestones.
`size_milestones`	The size of milestones.
`alpha_cells`	The alpha of the cells.
`size_cells`	The size of the cells.
`border_radius_percentage`	The fraction of the radius that is used for the border.
`arrow`	The type and size of arrow in case of directed trajectories. Set to NULL to remove arrow altogether.
`label_milestones`	How to label the milestones. Can be TRUE (in which case the labels within the trajectory will be used), "all" (in which case both given labels and milestone_ids will be used), a named character vector, or FALSE
`plot_milestones`	Whether to plot the milestones.
`adjust_weights`	Whether or not to rescale the milestone network weights

Value

A graph ggplot of a trajectory.

Examples

```
data(example_disconnected)
plot_graph(example_disconnected)
plot_graph(example_disconnected, color_cells = "pseudotime")
plot_graph(
  example_disconnected,
  color_cells = "grouping",
  grouping = dynwrap::group_onto_nearest_milestones(example_disconnected)
)

data(example_tree)
plot_graph(example_tree)
```

`plot_heatmap` *Plot expression data along a trajectory*

Description

NOTE: When using RStudio, the heatmap might not show inside the plot area, but will be visible once you click the ’Zoom’ button.

Usage

```
plot_heatmap(
  trajectory,
  expression_source = "expression",
  features_oi = 20,
  clust = "ward.D2",
  margin = 0.02,
  color_cells = NULL,
  milestones = NULL,
  milestone_percentages = trajectory$milestone_percentages,
  grouping = NULL,
  groups = NULL,
  cell_feature_importances = NULL,
  heatmap_type = c("tiled", "dotted"),
  scale = dynutils::scale_quantile,
  label_milestones = TRUE
)
```

Arguments

`trajectory` A dynwrap trajectory.

`expression_source` Source of the feature expression, defaults to `get_expression(trajectory)`.

`features_oi` The features of interest, either the number of features or a vector giving the names of the different features

`clust` The method to cluster the features, or a hclust object

`margin` A margin between trajectory segments.

`color_cells` How to color the cells.

- `"auto"`: Try to figure out how to color cells depending on whether one of the `grouping`, `feature_io`, `milestones` or `pseudotime` parameters are defined.
- `"none"`: Cells are not coloured.
- `"grouping"`: Cells are coloured according to a grouping (e.g. clustering). Either the `grouping` parameter or `trajectory$grouping` must be a named character vector.
- `"feature"`: Cells are coloured according to the values of a given feature (e.g. gene expression). Either the `expression_source` parameter or `get_expression(trajectory)` must be a matrix. Parameter `feature_oi` must also be defined.
- `"milestone"` (recommended): Cells are coloured according their position in the trajectory. The positioning of the cells are determined by parameter `milestone_percentages` or else by `trajectory$milestone_percentages`. The colours of the milestones can be determined automatically or can be specified by passing a tibble containing character columns `milestone_id` and `color` (See `add_milestone_coloring()` for help in constructing this object).

- `"pseudotime"`: Cells are coloured according to the pseudotime value from the root.

`milestones` Tibble containing the column `milestone_id` (character). If `color_milestones` is set to `"given"`, this tibble should also contain a column `color` (character), containing colour hex codes (e.g. `"#123456"`).

`milestone_percentages`
The milestone percentages.

`grouping` A grouping of the cells (e.g. clustering) as a named character vector.

`groups` A tibble containing character columns `group_id` and `color`. If `NULL`, this object is inferred from the `grouping` itself.

`cell_feature_importances`
The importances of every feature in every cell, as returned by `dynfeature::calculate_cell_feature_`

`heatmap_type` The type of heatmap, either tiled or dotted

`scale` Whether to rescale the expression, can be a function or boolean

`label_milestones`
How to label the milestones. Can be TRUE (in which case the labels within the trajectory will be used), "all" (in which case both given labels and milestone_ids will be used), a named character vector, or FALSE

Value

A heatmap ggplot of an expression dataset with trajectory.

Examples

```
data(example_bifurcating)
plot_heatmap(example_bifurcating)
```

---

plot_linearised_comparison

Compare two trajectories as a pseudotime scatterplot

Description

Compare two trajectories as a pseudotime scatterplot

Usage

```
plot_linearised_comparison(
  traj1,
  traj2,
  reorder = TRUE,
  margin = 0.05,
  reorder_second_by = c("mapping", "optimisation")
)
```

Arguments

`traj1`	The first trajectory
`traj2`	The second trajectory
`reorder`	Whether to reorder the trajectory
`margin`	A margin between trajectory segments.
`reorder_second_by`	How to reorder the second trajectory, either by mapping the milestones from both trajectories (`mapping`), or by trying to correlate the orderings between the two trajectories (`optimisation`)

Value

A scatterplot comparison ggplot of two linearised trajectories.

Examples

```
data(example_bifurcating)
plot_linearised_comparison(example_bifurcating, example_bifurcating)
```

`plot_onedim` *Plot a trajectory as a one-dimensional set of connected segments*

Description

Plot a trajectory as a one-dimensional set of connected segments

Usage

```
plot_onedim(
  trajectory,
  color_cells = c("auto", "none", "grouping", "feature", "milestone", "pseudotime"),
  grouping = NULL,
  groups = NULL,
  feature_oi = NULL,
  pseudotime = NULL,
  expression_source = "expression",
  color_milestones = c("auto", "given", "cubeHelix", "Set3", "rainbow"),
  milestones = NULL,
  milestone_percentages = NULL,
  alpha_cells = 1,
  size_cells = 2.5,
  border_radius_percentage = 0.1,
  orientation = 1,
  margin = 0.05,
  linearised = linearise_cells(trajectory, margin, one_edge = TRUE),
  quasirandom_width = 0.2,
  plot_cells = TRUE,
  label_milestones = dynwrap::is_wrapper_with_milestone_labelling(trajectory),
  arrow = grid::arrow(type = "closed")
)
```

Arguments

`trajectory`	A dynwrap trajectory.
`color_cells`	How to color the cells. • `"auto"`: Try to figure out how to color cells depending on whether one of the `grouping`, `feature_io`, `milestones` or `pseudotime` parameters are defined. • `"none"`: Cells are not coloured. • `"grouping"`: Cells are coloured according to a grouping (e.g. clustering). Either the `grouping` parameter or `trajectory$grouping` must be a named character vector. • `"feature"`: Cells are coloured according to the values of a given feature (e.g. gene expression). Either the `expression_source` parameter or `get_expression(trajectory)` must be a matrix. Parameter `feature_oi` must also be defined. • `"milestone"` (recommended): Cells are coloured according their position in the trajectory. The positioning of the cells are determined by parameter `milestone_percentages` or else by `trajectory$milestone_percentages`. The colours of the milestones can be determined automatically or can be specified by passing a tibble containing character columns `milestone_id` and `color` (See `add_milestone_coloring()` for help in constructing this object). • `"pseudotime"`: Cells are coloured according to the pseudotime value from the root.
`grouping`	A grouping of the cells (e.g. clustering) as a named character vector.
`groups`	A tibble containing character columns `group_id` and `color`. If `NULL`, this object is inferred from the `grouping` itself.
`feature_oi`	The name of a feature to use for colouring the cells.
`pseudotime`	The pseudotime from the root of the trajectory to the cells as a named numeric vector.
`expression_source`	Source of the feature expression, defaults to `get_expression(trajectory)`.
`color_milestones`	Which palette to use for colouring the milestones • `auto`: Determine colours automatically. If `color` is already specified in milestones tibble, this will be used. Otherwise, the colour scheme is determined by `milestone_palette_list$auto`. • `given`: The `milestones` object already contains a column `color`. • `cubeHelix`: Use the `rje::cubeHelix()` palette. • `Set3`: Use the `RColorBrewer::brewer.pal(name = "Set3")` palette.

- rainbow: Use the grDevices::rainbow() palette.

milestones	Tibble containing the column milestone_id (character). If color_milestones is set to "given", this tibble should also contain a column color (character), containing colour hex codes (e.g. "#123456").
milestone_percentages	The milestone percentages.
alpha_cells	The alpha of the cells
size_cells	The size of the cells
border_radius_percentage	The fraction of the radius that is used for the border
orientation	Whether to plot the connections in the top (1) or bottom (-1)
margin	A margin between trajectory segments.
linearised	The linearised milestone network and progressions
quasirandom_width	The width of the quasirandom cell spreading
plot_cells	Whether to plot the cells
label_milestones	How to label the milestones. Can be TRUE (in which case the labels within the trajectory will be used), "all" (in which case both given labels and milestone_ids will be used), a named character vector, or FALSE
arrow	The type and size of arrow in case of directed trajectories. Set to NULL to remove arrow altogether.

Value

A linearised (non-)linear trajectory.

Examples

```
data(example_linear)
plot_onedim(example_linear)
plot_onedim(example_linear, label_milestones = TRUE)

data(example_tree)
plot_onedim(example_tree)
```

---

plot_strip *Plot strip*

Description

Plot strip

Usage

```
plot_strip(traj1, traj2, margin = 0.05, reorder = TRUE)
```

Arguments

`traj1`	The first trajectory
`traj2`	The second traj
`margin`	A margin between trajectory segments.
`reorder`	Whether to reorder

Value

A scatterplot comparison ggplot of two linearised trajectories.

Examples

```
data(example_bifurcating)
plot_strip(example_bifurcating, example_bifurcating)
```

`plot_topology` *Plot the topology of a trajectory*

Description

Plot the topology of a trajectory

Usage

```
plot_topology(
  trajectory,
  color_milestones = c("auto", "given", "cubeHelix", "Set3", "rainbow"),
  milestones = NULL,
  layout = NULL,
  arrow = grid::arrow(type = "closed", length = unit(0.4, "cm"))
)
```

Arguments

`trajectory` A dynwrap trajectory.

`color_milestones`
Which palette to use for colouring the milestones

- `auto`: Determine colours automatically. If `color` is already specified in milestones tibble, this will be used. Otherwise, the colour scheme is determined by `milestone_palette_list$auto`.
- `given`: The `milestones` object already contains a column `color`.
- `cubeHelix`: Use the `rje::cubeHelix()` palette.
- `Set3`: Use the `RColorBrewer::brewer.pal(name = "Set3")` palette.
- `rainbow`: Use the `grDevices::rainbow()` palette.

`milestones`	Tibble containing the column `milestone_id` (character). If `color_milestones` is set to `"given"`, this tibble should also contain a column `color` (character), containing colour hex codes (e.g. `"#123456"`).
`layout`	The type of layout to create. See `ggraph::ggraph()` for more info.
`arrow`	The type and size of arrow in case of directed trajectories. Set to NULL to remove arrow altogether.

Value

A topology ggplot of a trajectory.

Examples

```
data(example_disconnected)
plot_topology(example_disconnected)

data(example_tree)
plot_topology(example_tree)
```

`project_waypoints_coloured`
Project the waypoints

Description

Project the waypoints

Usage

```
project_waypoints_coloured(
  trajectory,
  cell_positions,
  edge_positions = NULL,
  waypoints = dynwrap::select_waypoints(trajectory),
  trajectory_projection_sd = sum(trajectory$milestone_network$length) * 0.05,
  color_trajectory = "none"
)
```

Arguments

`trajectory`	A dynwrap trajectory.
`cell_positions`	The positions of the cells in 2D. Must be a tibble with character column `cell_id` and numeric columns `comp_1` and `comp_2`.
`edge_positions`	The positions of the edges.
`waypoints`	The waypoints to use for projecting. Can by generated using `dynwrap::select_waypoints()`.
`trajectory_projection_sd`	The standard deviation of the gaussian kernel.

`color_trajectory`
How to color the trajectory, can be "nearest" for coloring to nearest cell, or "none".

Value

A named list containing items:

- segments: A tibble containing columns `comp_1` (numeric), `comp_2` (numeric), `waypoint_id` (character), `milestone_id` (character), `from` (character), `to` (character) `percentage` (numeric), `group` (factor), and `arrow` (logical).

`theme_clean` *We like our plots clean*

Description

We like our plots clean

Usage

```
theme_clean()
```

Value

A ggplot2 theme.

Examples

```
data(example_bifurcating)
g <- plot_dimred(example_bifurcating)
g + theme_clean()
```

`theme_graph` *We like our plots clean*

Description

We like our plots clean

Usage

```
theme_graph()
```

Value

A ggplot2 theme.

Examples

```
data(example_bifurcating)
g <- plot_dimred(example_bifurcating)
g + theme_graph()
```

Index