

# Package: cellpaintr (via r-universe)

September 23, 2026

**Type** Package

**Title** Perturbation Analysis for Cell Painting Data

**Version** 0.99.3

**Description** This package load cell painting data into R and preprocess feature to make them ready for machine learning. We use random forest to predict cell perturbations from CellProfiler features. We summarize the results in a volcano plot.

**biocViews** SingleCell, Software, Classification, FeatureExtraction, Preprocessing, Visualization

**License** LGPL-3

**Encoding** UTF-8

**LazyData** false

**Config/testthat/edition** 3

**Imports** ggplot2, readr, stringr, dplyr, parsnip, ranger, yardstick, rsample, purrr, scrapper, tibble, tidyr, ggrepel, cli

**Depends** R (>= 4.5.0), SummarizedExperiment, SingleCellExperiment

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), BiocStyle, scater, rlang

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

**URL** <https://github.com/ChristofSeiler/cellpaintr>,  
<https://christofseiler.github.io/cellpaintr/>

**BugReports** <https://github.com/ChristofSeiler/cellpaintr/issues>

**Repository** <https://biocstaging.r-universe.dev>

**Date/Publication** 2026-08-24 14:20:07 UTC

**RemoteUrl** <https://github.com/BiocStaging/cellpaintr>

**RemoteRef** HEAD

**RemoteSha** 7b091cd816d3acc68bc5071dfe76d25726922c

Contents

|                               |           |
|-------------------------------|-----------|
| aggregateYhat . . . . .       | 2         |
| calculateStats . . . . .      | 3         |
| compute_y_hat . . . . .       | 4         |
| generate_data . . . . .       | 5         |
| loadData . . . . .            | 5         |
| plotAUC . . . . .             | 6         |
| plotCellsPerImage . . . . .   | 7         |
| plotLOO . . . . .             | 7         |
| plotPCACor . . . . .          | 8         |
| plotROC . . . . .             | 9         |
| predictLOO . . . . .          | 10        |
| removeLowVariance . . . . .   | 11        |
| removeNAs . . . . .           | 12        |
| removeOutliers . . . . .      | 12        |
| removeRowColEffect . . . . .  | 13        |
| removeZeroInflation . . . . . | 14        |
| transformScale . . . . .      | 14        |
| volcanoPlot . . . . .         | 15        |
| <b>Index</b>                  | <b>17</b> |

---

|               |                                                                            |
|---------------|----------------------------------------------------------------------------|
| aggregateYhat | <i>Aggregate predicted leave-one-out probabilities over meta variables</i> |
|---------------|----------------------------------------------------------------------------|

---

Description

Aggregate predicted leave-one-out probabilities over meta variables

Usage

aggregateYhat(sce, target, group, assay\_type = "tfmfeatures")

Arguments

- sce                [SingleCellExperiment](#) object
- target            Name of target variable for prediction
- group             Grouping variable for cross-validation, e.g., patient
- assay\_type        A string specifying the assay

Value

data.frame

---

|                |                                                                                                                        |
|----------------|------------------------------------------------------------------------------------------------------------------------|
| calculateStats | <i>Aggregate predicted leave-one-out probabilities over meta variables over a list of SingleCellExperiment objects</i> |
|----------------|------------------------------------------------------------------------------------------------------------------------|

---

## Description

Aggregate predicted leave-one-out probabilities over meta variables over a list of SingleCellExperiment objects

## Usage

```
calculateStats(sce, target, group, assay_type = "tfmfeatures")
```

## Arguments

|            |                                                       |
|------------|-------------------------------------------------------|
| sce        | A <a href="#">SingleCellExperiment</a> object         |
| target     | Name of target variable for prediction                |
| group      | Grouping variable for cross-validation, e.g., patient |
| assay_type | A string specifying the assay                         |

## Value

data.frame

## Examples

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)

sce$Drug <- as.factor(sce$Drug)
sce$Drug <- relevel(sce$Drug, ref = "D1")
types <- c("AreaShape", "Intensity", "Texture")

sce_single <- predictL00(
  sce,
  target = "Drug", group = "Patient",
  interest_level = "D7", reference_level = "D1",
  types = types,
  n_threads = 1
)

calculateStats(sce_single, target = "Drug", group = "Patient")
```

---

`compute_y_hat`*Internal function to compute `y_hat` on a subset of the features*

---

## Description

Internal function to compute `y_hat` on a subset of the features

## Usage

```
compute_y_hat(  
  feature_name,  
  sce_feature,  
  starts,  
  target,  
  group,  
  interest_level,  
  reference_level,  
  weights,  
  n_threads,  
  assay_type  
)
```

## Arguments

|                              |                                                       |
|------------------------------|-------------------------------------------------------|
| <code>feature_name</code>    | String with feature name                              |
| <code>sce_feature</code>     | <a href="#">SingleCellExperiment</a>                  |
| <code>starts</code>          | Starting string                                       |
| <code>target</code>          | Name of target variable for prediction                |
| <code>group</code>           | Grouping variable for cross-validation, e.g., patient |
| <code>interest_level</code>  | Factor interest level in ‘target’ variable            |
| <code>reference_level</code> | Factor reference level in ‘target’ variable           |
| <code>weights</code>         | Weights variable when features are aggregated         |
| <code>n_threads</code>       | Number of parallel threads for fitting of models      |
| <code>assay_type</code>      | A string specifying the assay                         |

## Value

[tibble](#) data frame

---

|               |                                                                 |
|---------------|-----------------------------------------------------------------|
| generate_data | <i>Simulate CellProfiler data and write to a temporary file</i> |
|---------------|-----------------------------------------------------------------|

---

**Description**

Simulate CellProfiler data and write to a temporary file

**Usage**

```
generate_data()
```

**Value**

path to csv file

**Examples**

```
set.seed(23)
generate_data()
```

---

|          |                                                                                |
|----------|--------------------------------------------------------------------------------|
| loadData | <i>Load cell painting data from file and convert to a SingleCellExperiment</i> |
|----------|--------------------------------------------------------------------------------|

---

**Description**

Load cell painting data from file and convert to a SingleCellExperiment

**Usage**

```
loadData(cell_file)
```

**Arguments**

cell\_file      path to csv file from CellProfiler, e.g., MyExpt\_Cells.csv

**Value**

[SingleCellExperiment](#) object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
loadData(cell_file)
```

---

|         |                            |
|---------|----------------------------|
| plotAUC | <i>Plot AUC comparison</i> |
|---------|----------------------------|

---

**Description**

Plot AUC comparison

**Usage**

```
plotAUC(sce, target, group, assay_type = "tfmfeatures")
```

**Arguments**

|            |                                                       |
|------------|-------------------------------------------------------|
| sce        | A <a href="#">SingleCellExperiment</a> object         |
| target     | Name of target variable for prediction                |
| group      | Grouping variable for cross-validation, e.g., patient |
| assay_type | A string specifying the assay                         |

**Value**

[ggplot2](#) object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)

sce$Drug <- as.factor(sce$Drug)
sce$Drug <- relevel(sce$Drug, ref = "D1")
types <- c("AreaShape", "Intensity", "Texture")

sce_single <- predictL00(
  sce,
  target = "Drug", group = "Patient",
  interest_level = "D7", reference_level = "D1",
  types = types,
  n_threads = 1
)

plotAUC(sce_single, target = "Drug", group = "Patient")
```

---

|                   |                                       |
|-------------------|---------------------------------------|
| plotCellsPerImage | <i>Plot number of cells per image</i> |
|-------------------|---------------------------------------|

---

**Description**

Plot number of cells per image

**Usage**

```
plotCellsPerImage(sce, bins = 100)
```

**Arguments**

|      |                                             |
|------|---------------------------------------------|
| sce  | <a href="#">SingleCellExperiment</a> object |
| bins | Number of histogram bins                    |

**Value**

[ggplot2](#) object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
plotCellsPerImage(sce)
```

---

|         |                                                   |
|---------|---------------------------------------------------|
| plotL00 | <i>Plot predicted leave-one-out probabilities</i> |
|---------|---------------------------------------------------|

---

**Description**

Plot predicted leave-one-out probabilities

**Usage**

```
plotL00(sce, target, group, assay_type = "tfmfeatures")
```

**Arguments**

|            |                                                       |
|------------|-------------------------------------------------------|
| sce        | <a href="#">SingleCellExperiment</a> object           |
| target     | Name of target variable for prediction                |
| group      | Grouping variable for cross-validation, e.g., patient |
| assay_type | A string specifying the assay                         |

**Value**

`ggplot2` object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)

sce$Drug <- as.factor(sce$Drug)
sce$Drug <- relevel(sce$Drug, ref = "D1")
types <- c("AreaShape", "Intensity", "Texture")

sce_single <- predictL00(
  sce,
  target = "Drug", group = "Patient",
  interest_level = "D7", reference_level = "D1",
  types = types,
  n_threads = 1
)

plotL00(sce_single, target = "Drug", group = "Patient")
```

---

plotPCACor

*Plot number of cells per image*

---

**Description**

Plot number of cells per image

**Usage**

```
plotPCACor(
  sce,
  filter_by = 1,
  top = 20,
  pcs = seq(5),
  assay_type = "tfmfeatures"
)
```

**Arguments**

|            |                                             |
|------------|---------------------------------------------|
| sce        | <a href="#">SingleCellExperiment</a> object |
| filter_by  | PC to use for feature selection             |
| top        | Number of top features to select            |
| pcs        | Number of PCs to plot                       |
| assay_type | A string specifying the assay               |

**Value**

`ggplot2` object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce, robust = TRUE)
sce <- scater::runPCA(sce, exprs_values = "tfmfeatures", ncomponents = 10)
plotPCACor(sce, filter_by = 1)
```

---

plotROC

*Plot ROC curves*

---

**Description**

Plot ROC curves

**Usage**

```
plotROC(sce, target, group, assay_type = "tfmfeatures")
```

**Arguments**

|            |                                                       |
|------------|-------------------------------------------------------|
| sce        | <code>SingleCellExperiment</code> object              |
| target     | Name of target variable for prediction                |
| group      | Grouping variable for cross-validation, e.g., patient |
| assay_type | A string specifying the assay                         |

**Value**

`ggplot2` object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)

sce$Drug <- as.factor(sce$Drug)
sce$Drug <- relevel(sce$Drug, ref = "D1")
types <- c("AreaShape", "Intensity", "Texture")

sce_single <- predictL00(
  sce,
```

```

    target = "Drug", group = "Patient",
    interest_level = "D7", reference_level = "D1",
    types = types,
    n_threads = 1
)

plotROC(sce_single, target = "Drug", group = "Patient")

```

---

|            |                                     |
|------------|-------------------------------------|
| predictLOO | <i>Predict target from features</i> |
|------------|-------------------------------------|

---

## Description

Predict target from features

## Usage

```

predictLOO(
  sce,
  target,
  group,
  interest_level,
  reference_level,
  types = NULL,
  channels = NULL,
  weights = NULL,
  n_threads = 1,
  assay_type = "tfmfeatures"
)

```

## Arguments

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| sce             | <a href="#">SingleCellExperiment</a> object           |
| target          | Name of target variable for prediction                |
| group           | Grouping variable for cross-validation, e.g., patient |
| interest_level  | Factor interest level in ‘target’ variable            |
| reference_level | Factor reference level in ‘target’ variable           |
| types           | Vector of strings of feature types                    |
| channels        | Vector of strings of feature channels                 |
| weights         | Weights variable when features are aggregated         |
| n_threads       | Number of parallel threads for fitting of models      |
| assay_type      | A string specifying the assay                         |

**Value**

[tibble](#) data frame

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)

sce$Drug <- as.factor(sce$Drug)
sce$Drug <- releval(sce$Drug, ref = "D1")
types <- c("AreaShape", "Intensity", "Texture")

sce_single <- predictL00(
  sce,
  target = "Drug", group = "Patient",
  interest_level = "D7", reference_level = "D1",
  types = types,
  n_threads = 1
)
```

---

|                   |                                     |
|-------------------|-------------------------------------|
| removeLowVariance | <i>Filter low variance features</i> |
|-------------------|-------------------------------------|

---

**Description**

Filter low variance features

**Usage**

```
removeLowVariance(sce, threshold = 0, robust = FALSE)
```

**Arguments**

|           |                                                               |
|-----------|---------------------------------------------------------------|
| sce       | <a href="#">SingleCellExperiment</a> object                   |
| threshold | Keep features that have larger variance than this threshold   |
| robust    | If true use median absolute deviation, otherwise use variance |

**Value**

[SingleCellExperiment](#) object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- removeLowVariance(sce)
```

---

|           |                                           |
|-----------|-------------------------------------------|
| removeNAs | <i>Remove cells with missing features</i> |
|-----------|-------------------------------------------|

---

**Description**

Remove cells with missing features

**Usage**

```
removeNAs(sce)
```

**Arguments**

sce [SingleCellExperiment](#) object

**Value**

[SingleCellExperiment](#) object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- removeNAs(sce)
```

---

|                |                                                            |
|----------------|------------------------------------------------------------|
| removeOutliers | <i>Remove cells if not enough or too many in one image</i> |
|----------------|------------------------------------------------------------|

---

**Description**

Remove cells if not enough or too many in one image

**Usage**

```
removeOutliers(sce, min, max)
```

**Arguments**

sce                [SingleCellExperiment](#) object  
 min               Remove cells below that number  
 max               Remove cells above that number

**Value**

[SingleCellExperiment](#) object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- removeOutliers(sce, min = 0, max = 300)
```

---

|                    |                                                                       |
|--------------------|-----------------------------------------------------------------------|
| removeRowColEffect | <i>Remove row and column effect using the median polish procedure</i> |
|--------------------|-----------------------------------------------------------------------|

---

**Description**

Remove row and column effect using the median polish procedure

**Usage**

```
removeRowColEffect(sce, assay_type = "tfmfeatures")
```

**Arguments**

sce                [SingleCellExperiment](#) object  
 assay\_type        A string specifying the assay

**Value**

[SingleCellExperiment](#) object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)
row <- substr(sce$Well, 1, 1)
row <- factor(row, levels = LETTERS[1:length(unique(row))])
col <- substr(sce$Well, 2, 3)
col <- factor(as.numeric(col))
sce$Row <- row
```

```
sce$Col <- col
sce <- removeRowColEffect(sce)
```

---

removeZeroInflation      *Remove zero-inflated features*

---

### Description

Remove zero-inflated features

### Usage

```
removeZeroInflation(sce, proportion = 0.2)
```

### Arguments

sce                      [SingleCellExperiment](#) object  
 proportion              Remove features exceeding this zero-inflation proportion

### Value

[SingleCellExperiment](#) object

### Examples

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- removeZeroInflation(sce)
```

---

transformScale              *Transform, center, and scale features*

---

### Description

Transform, center, and scale features

### Usage

```
transformScale(sce, robust = FALSE)
```

### Arguments

sce                      [SingleCellExperiment](#) object  
 robust                  If true robust z-score, otherwise standard z-score

**Value**

`SingleCellExperiment` object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)
```

---

volcanoPlot

*Plot predicted leave-one-out probabilities*


---

**Description**

Plot predicted leave-one-out probabilities

**Usage**

```
volcanoPlot(
  sce,
  target,
  group,
  p_cutoff = NULL,
  fc_cutoff = 1,
  assay_type = "tfmfeatures"
)
```

**Arguments**

|            |                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------|
| sce        | A <code>SingleCellExperiment</code> object                                                         |
| target     | Name of target variable for prediction                                                             |
| group      | Grouping variable for cross-validation, e.g., patient                                              |
| p_cutoff   | Cut-off for statistical significance. A horizontal line will be drawn at $-\log_{10}(p\_cutoff)$ . |
| fc_cutoff  | Cut-off for absolute log2 fold-change. A vertical lines will be drawn at fc_cutoff.                |
| assay_type | A string specifying the assay                                                                      |

**Value**

`ggplot2` object

**Examples**

```
set.seed(23)
cell_file <- generate_data()
sce <- loadData(cell_file)
sce <- transformScale(sce)

sce$Drug <- as.factor(sce$Drug)
sce$Drug <- releval(sce$Drug, ref = "D1")
types <- c("AreaShape", "Intensity", "Texture")

sce_single <- predictL00(
  sce,
  target = "Drug", group = "Patient",
  interest_level = "D7", reference_level = "D1",
  types = types,
  n_threads = 1
)

volcanoPlot(sce_single,
  target = "Drug", group = "Patient",
  p_cutoff = 0.05, fc_cutoff = 0.5
)
```

# Index

aggregateYhat, [2](#)

calculateStats, [3](#)  
compute\_y\_hat, [4](#)

generate\_data, [5](#)  
ggplot2, [6–9](#), [15](#)

loadData, [5](#)

plotAUC, [6](#)  
plotCellsPerImage, [7](#)  
plotL00, [7](#)  
plotPCACor, [8](#)  
plotROC, [9](#)  
predictL00, [10](#)

removeLowVariance, [11](#)  
removeNAs, [12](#)  
removeOutliers, [12](#)  
removeRowColEffect, [13](#)  
removeZeroInflation, [14](#)

SingleCellExperiment, [2–15](#)

tibble, [4](#), [11](#)  
transformScale, [14](#)

volcanoPlot, [15](#)