# Advanced scRNA-seq Cheatsheet

**The tables below consist of valuable functions or commands that will help you through this module.**

**Each table represents a different library/tool and its corresponding commands.**

**You may also be interested in the following additional cheatsheets:**

- Download the PDF for the [Introduction to R and Tidyverse cheatsheet](#)
- Download the PDF for the [Introduction to Single-Cell RNA sequencing cheatsheet](#)
- Download the PDF for the [`SCE` and `Seurat` object comparison cheatsheet](#)

> Please note that these tables are not intended to tell you all the information you need to know about each command.
>
> The hyperlinks found in each piece of code will take you to the documentation for further information on the usage of each command. Please be aware that the documentation will generally provide information about the given function's most current version (or a recent version, depending on how often the documentation site is updated). This will usually (but not always!) match what you have installed on your machine. If you have a different version of R or other R packages, the documentation may differ from what you have installed.

**Table of Contents**

- - `ggplot2` functions
  - `dplyr`, `tidyr`, `stringr`, and `tibble` functions

- Pathway analysis

  - `msigdbr`
  - `clusterProfiler` and `enrichplot`
  - `AUCell` and `GSEABase`

- `bluster`

# `scater` and `scran`

Read the `scater` package documentation, and a vignette on its usage.

Read the `scran` package documentation, and a vignette on its usage.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `scater` | `plotReducedDim()` | Plot reduced dimensions | Plot a given reduced dimension slot from a `SingleCellExperiment` object by its name |
| `scater` | `plotUMAP()` | Plot UMAP | Plot the "UMAP"-named reduced dimension slot from a `SingleCellExperiment` object |
| `scater` | `plotExpression()` | Plot expression | Plot expression values for all cells in a `SingleCellExperiment` object, using the `logcounts` assay by default |
| `scran` | `clusterCells()` | Cluster cells | Perform clustering on an SCE object using the `bluster` package |

# `miQC`

Read the `miQC` package documentation, and a vignette on its usage.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `miQC` | `mixtureModel()` | Mixture model | Fit a `miQC` mixture model to a `SingleCellExperiment` object for use in filtering |
| `miQC` | `filterCells()` | Filter cells | Filter cells from a `SingleCellExperiment` object based on a `miQC` model, returning a filtered `SingleCellExperiment` object |
| `miQC` | `plotMetrics()` | Plot metrics | Plot percent of mitochondrial reads against the number of unique genes found for each cell |
| `miQC` | `plotModel()` | Plot model | `miQC::plotMetics()` with the `miQC` fitted model overlaid |
| `miQC` | `plotFiltering()` | Plot filtering | Plot percent of mitochondrial reads against the number of unique genes found, coloring points based on whether they will be filtered out or not |

# `batchelor` and `harmony`

Read the `batchelor` package documentation, and a vignette on its usage.

Read the `harmony` package documentation, and a vignette on its usage.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `batchelor` | `MultiBatchPCA()` | Multi-batch PCA | Perform PCA across multiple gene expression matrices, weighted by batch size |
| `batchelor` | `fastMNN()` | Fast mutual nearest neighbors correction | Perform integration on an SCE object with mutual nearest neighbors using the `fastMNN` algorithm, returning an SCE object with batch-corrected principal components |
| `harmony` | `RunHarmony()` | Run the `harmony` algorithm | Perform integration with the `harmony` algorithm on a matrix of single-cell genomics cell embeddings, returning a matrix of batch-corrected principal components |

# `SingleR`

Read the `SingleR` package documentation, and an e-book on its usage.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `SingleR` | `trainSingleR()` | Train the SingleR classifier | Build a `SingleR` classifier model object from an annotated reference dataset |
| `SingleR` | `classifySingleR()` | Classify cells with SingleR | Use a `SingleR` model object to assign cell types to the cells in an `SCE` object |
| `SingleR` | `SingleR()` | Annotate scRNA-seq data | Combines `trainSingleR()` and `classifySingleR()` to assign cell types to an `SCE` object from an annotated reference dataset |

## `pheatmap` and `EnhancedVolcano`

Read the `pheatmap` package documentation.

Read the `EnhancedVolcano` package documentation, and vignette on its usage.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `pheatmap` | `pheatmap()` | Pretty heatmap | Plot a (pretty!) clustered heatmap |
| `EnhancedVolcano` | `EnhancedVolcano()` | Enhanced volcano | Plot a volcano plot to visualize differential expression analysis results |

## `DESeq2` and pseudo-bulking functions

Read the `DESeq2` package documentation, and a [vignette on its usage](#).

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `scuttle` | `aggregateAcrossCells()` | Aggregate data across groups of cells | Sum counts for each combination of features across groups of cells, commonly used to *pseudo-bulk* `SCE` counts |
| `DESeq2` | `DESeqDataSet()` | DESeq Dataset | Establish a `DESeq` object from a pseudo-bulked `SingleCellExperiment` object or a bulk `SummarizedExperiment` object |
| `DESeq2` | `estimateSizeFactors()` | Estimate size factors | Estimate size factors which are used to normalize counts for differential expression analysis |
| `DESeq2` | `rlog()` | Apply a regularized log transformation | Log2-transform counts in a `DESeq` object for differential expression analysis |
| `DESeq2` | `plotPCA()` | Sample PCA plot for transformed data | Plot sample PCA from a log-transformed `DESeq` object to check for batch effects |
| `DESeq2` | `DESeq()` | Perform differential expression analysis | Perform differential expression: Estimate size factors, transform data, estimate dispersions, and perform testing. |
| `DESeq2` | `plotDispEsts()` | Plot dispersion estimates | Plot dispersion estimates from a fitted `DESeq` object to evaluate model fit |

| | | | |
|---|---|---|---|
| `DESeq2` | `results()` | Extract results from a `DESeq` analysis | Extract results from a fitted `DESeq` object into a data frame |
| `DESeq2` | `resultsNames()` | Extract results names | Return coefficient names from a fitted `DESeq` object |
| `DESeq2` | `lfcShrink()` | Shrink log2 fold changes | Add shrunken log2-fold changes to a results table produced by `DESeq2::results()` |

## `tidyverse` functions

### `purrr` functions

Read the `purrr` package documentation and a vignette on its usage, and download the `purr` package cheatsheet.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `purrr` | `map()` | map | Apply a function across each element of list; return a list |
| `purrr` | `imap()` | imap | Apply a function across each element of list and its index/names; return a list |
| `purrr` | `map2()` | map2 | Apply a function across each element of two lists at a time; return a list |
| `purrr` | `reduce()` | Reduce | Reduce a list to a single value by applying a given function |

Note that `purrr::map()` functions can take advantage of R's new (as of version 4.1.0) anonymous function syntax:

```
# One-line syntax:
\(x) # function code goes here #

# Multi-line syntax:
\(x) {
  # function code goes      #
  # inside the curly braces #
}

# Example: Use an anonymous function with `purrr::map()`
# to get the colData's rownames for each SCE in `list_of_sce_objects`
purrr::map(
  list_of_sce_objects,
  \(x) rownames(colData(x))
)
```

## `ggplot2` functions

Read the `ggplot2` package documentation and an overall reference for `ggplot2` functions, and download the `ggplot2` package cheatsheet.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `ggplot2` | `geom_bar()` | Barplot | Creates a barplot of counts for a given categorical variable when added as a layer to a `ggplot()` object |
| `ggplot2` | `scale_fill_brewer()` | Add brewer fill scale | Apply a Brewer "fill" color palette to a categorical variable in a `ggplot()` object |
| `ggplot2` | `guides()` | Guides | Function to customize legend ("guide") appearance |
| `ggplot2` | `facet_grid()` | Facet grid | Plot individual panels using specified variables to subset the data across rows and/or columns of a grid |
| `ggplot2` | `vars()` | Vars | Helper function to specify variables to `facet_grid()` or `facet_wrap()` |
| `ggplot2` | `theme_bw()` | Black and white theme | Display `ggplot` with gridlines but a white background |
| `ggplot2` | `theme()` | Theme | Customize elements of a `ggplot` plot theme |
| `ggplot2` | `element_text()` | Element text | Customize textual elements of a `ggplot` theme |

# `dplyr`, `tidyr`, `stringr`, and `tibble` functions

Read the full documentation and download cheatsheets (where available) for these `tidyverse` packages at the following links:

- `dplyr` documentation and `dplyr` cheatsheet
- `tidyr` documentation and `tidyr` cheatsheet
- `stringr` documentation and `stringr` cheatsheet
- `tibble` documentation

| Library/Package | Piece of Code | What it's called | What it does |
| --- | --- | --- | --- |
| `dplyr` | `pull()` | Pull | Extract a single column from a data frame into a stand-alone vector |
| `dplyr` | `count()` | Count | Count the number of observations in each group of a data frame |
| `dplyr` | `left_join()` | Left join | Joins two data frames together, retaining only rows present in the first ("left") argument to the function |
| `dplyr` | `relocate()` | Relocate | Change column order in a data frame by relocating one or more columns |
| `dplyr` | `case_when()` | Case when | Return a value based on a set of `TRUE` / `FALSE` comparisons; a vectorized `if-else` |
| `tidyr` | `pivot_longer()` | Pivot longer | Convert a "wide" format data frame to a "long" format data frame |
| `tibble` | `as_tibble()` | As tibble | Convert an object to a tibble |
| `stringr` | `str_detect()` | String detect | Returns `TRUE` / `FALSE` if a string contains a given substring |
| `stringr` | `str_starts()` | String starts | Returns `TRUE` / `FALSE` if a string starts with a given substring |

# Pathway analysis

`msigdbr`

Read the `msigdbr` package documentation and its vignette.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `msigdbr` | `msigdbr_species()` | List `msigdbr`-supported species | Lists the species `msigdbr` supports |
| `msigdbr` | `msigdbr()` | Retrieve gene set | Retrieves gene sets and member genes in long data frame format |

## `clusterProfiler` and `enrichplot`

Read the `clusterProfiler` package documentation (PDF).

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `clusterProfiler` | `GSEA()` | Gene Set Enrichment Analysis (GSEA) | Performs a universal gene set enrichment analysis on given preranked (sorted) named vector of statistics, where the names in the vector are gene identifiers of gene sets |
| `enrichplot` | `gseaplot()` | GSEA plot | Produces a plot displaying the distribution of gene set and enrichment score |

## `AUCell` and `GSEABase`

Read the `AUCell` package documentation and its vignette.

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `GSEABase` | `GeneSet()` | Gene set | Constructs a gene set as a `GeneSet` object for use with `AUCell` |
| `GSEABase` | `GeneSetCollection()` | Gene set collection | Constructs a collection of gene sets as a `GeneSetCollection` object for use with `AUCell` |
| `AUCell` | `AUCell_buildRankings()` | Build cell rankings | Builds a ranking of genes for each cell that is used to calculate the recovery curve |
| `AUCell` | `AUCell_calcAUC()` | Calculate AUC | Calculates the area under the recovery curve (AUC) for each gene set in each cell |
| `AUCell` | `AUCell_exploreThresholds()` | Explore thresholds | Calculates thresholds in AUC values that can be used to assign cells; optionally makes assignments and produces histograms |

## `bluster`

Read the `bluster` package documentation and vignettes on its usage:

- Flexible clustering for Bioconductor
- Assorted clustering diagnostics

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `bluster` | `clusterRows()` | Cluster rows of a matrix | Perform clustering using a variety of algorithms on a matrix-like object |
| `bluster` | `KmeansParam()` | K-means clustering parameters | Set up parameters to run clustering using `kmeans()` within `scran::clusterCells()` or `bluster::clusterRows()` |
| `bluster` | `NNGraphParam()` | Graph-based clustering parameters | Set up parameters for nearest-neighbor (NN) graph-based clustering algorithms within `scran::clusterCells()` or `bluster::clusterRows()` |
| `bluster` | `approxSilhouette()` | Approximate silhouette width | Calculate an approximate silhouette width for each cell given a set of clusters |
| `bluster` | `neighborPurity()` | Compute neighborhood purity | Calculate neighborhood purity for each cell given a set of clusters |
| `bluster` | `bootstrapStability()` | Assess cluster stability by bootstrapping | Generate cluster bootstrap replicates to estimate cluster robustness to sampling noise |