# 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

> 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**

## `scater`

Read the `scater` 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 |

## `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` | `HarmonyMatrix()` | Perform `harmony` integration on a matrix | Perform integration with `harmony` on either a matrix of principle components or gene expression, 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 |

| Library/Package | Piece of Code | What it's called | What it does |
|---|---|---|---|
| `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 |

## `Seurat` and `SCE` object conversion

When converting between `Seurat` and `SCE` objects, it's helpful to know how the different object types store and refer to similar information.
The table below shows different aspects of single-cell objects and how to access the associated data, assuming the default names for each type of single-cell object.
There are several differences between `Seurat` and `SCE` objects that are useful to be aware of when converting them.
Importantly, the term `"assay"` refers to different things in `SCE` vs. `Seurat` objects:

- In an `SCE` object, an `assay` is a matrix of counts, with default names `"counts"` for raw counts and `"logcounts"` for normalized counts.
- In a `Seurat` object, an `assay` instead refers to an *experiment*. The default `Seurat` assay is called `"RNA"`, and it is analogous to the "main experiment" in an `SCE` object, which is not given a particular name.
- The `Seurat` count matrices are stored within a given assay (experiment) and have default names of `"counts"` for raw counts and `"data"` for normalized counts.

In addition, by default, `SCE` reduced dimension names are capitalized (e.g., `"PCA"`), and `Seurat` reduced dimension names are in lower case (e.g., `"pca"`).

Always bear in mind that your object(s) may be named differently from the defaults as described here!

| Data aspect | `SCE` | `Seurat` |
|---|---|---|
| Raw counts matrix | `counts(sce_object)` | `seurat_obj[["RNA"]]@counts` |
| Normalized counts matrix | `logcounts(sce_object)` | `seurat_obj[["RNA"]]@data` |
| Reduced dimension: PCA matrix | `reducedDim(sce_object, "PCA)` | `seurat_obj$pca@cell.embeddings` |
| Reduced dimension: UMAP matrix | `reducedDim(sce_object, "UMAP)` | `seurat_obj$umap@cell.embeddings` |
| Cell-level metadata | `colData(sce_object)` | `seurat_obj@meta.data` |
| Feature (gene)-level metadata | `rowData(sce_object)` | `seurat_obj[["RNA"]]@meta.features` |
| Miscellaneous additional metadata | `metadata(sce_object)` | `seurat_obj@misc` |

Below, we provide some code examples below for how you can accomplish these conversions.
For all code examples below, it is assumed that the `SingleCellExperiment` library has been loaded into your R environment:

```
library(SingleCellExperiment)
```

## Converting from `Seurat` to `SCE`

The following example code assumes you have a `Seurat` object called `seurat_obj`.

```
# Convert Seurat object to SCE object
sce_object <- Seurat::as.SingleCellExperiment(seurat_obj)
```

By default, all assays (experiments) present in the `Seurat` object will be ported into the new `SCE` object.
Recall, in `Seurat`, an assay refers to an *experiment* which may be associated with multiple count matrices.
To only specify that certain assays are retained, you can optionally provide the argument `assay` with *`Seurat` assay names* to retain in the `SCE` object, for example:

```
# Convert Seurat object to SCE object, retaining only the 'RNA' experiment (assay)
sce_object <- Seurat::as.SingleCellExperiment(seurat_obj, assay = "RNA")
```

Specifying `assay` is mostly useful if there are alternative experiments, for example from CITE-Seq data, present in the `Seurat` object that you do not want to retain during `SCE` conversion.

## Converting from `SCE` to `Seurat`

The following example code assumes you are starting with an `SCE` object called `sce_object`.

The function `Seurat::as.Seurat()` can be used to convert an `SCE` object into a `Seurat` object and takes the following arguments:

- The `SCE` object to convert
- Optional named arguments with the following defaults:
  - `counts = "counts"` specifies that the `SCE` object contains a `"counts"` assay of normalized counts that should be included during conversion.
  - If there is no `"counts"` assay in the SCE object, set this argument as `counts = NULL` or rename accordingly, e.g. `counts = "whatever_assay_name_you_are_using"`.
  - `data = "logcounts"` specifies that the `SCE` object contains a `"logcounts"` assay of normalized counts that should be included during conversion.
  - If there is no `"logcounts"` assay in the SCE object, set this argument as `data = NULL` or rename accordingly, e.g. `data = "whatever_assay_name_you_are_using"`.
  - `assay = NULL` specifies that, by default, all assays (experiments) will be converted. If there are multiple assays and you wish to only convert, for example, the `"RNA"` assay, set this argument as `assay = "RNA"`.
  - `project = "SingleCellExperiment"` specifies that the `Seurat` object being created will have this associated project name. You can override this with any string of interest, e.g. `project = "sample_XYZ"`.

```
# Convert SCE object to Seurat object, assuming both
#  `counts` and  `logcounts` assays are present
seurat_object <- Seurat::as.Seurat(sce_object)

# Convert SCE object to Seurat object, where the SCE object
#  contains a `counts` but not a `logcounts` assay
seurat_object <- Seurat::as.Seurat(sce_object, data = NULL)
```

### Approaches from `ScPCA`

In addition, this documentation from the `ScPCA` introduces how to convert `SCE` objects to `Seurat` objects.
Although this documentation was written for `ScPCA` datasets, the steps generally apply to any `SCE` object.
It's worth noting that the example code provided at that link will only retain a single assay (raw `"counts"`) in the new `SCE` object, and it will not retain reduced dimension representations (e.g., PCA or UMAP).
Therefore, this example code is mostly useful at the early stages of processing before you have performed normalization and calculated reduced dimensions.