Seurat Command List

Compiled: 2023-10-31

Source: `vignettes/essential_commands.Rmd` (https://github.com/satijalab/seurat/blob/HEAD/vignettes/essential_commands.Rmd)

Standard `Seurat` workflow

```
pbmc <- NormalizeData(object = pbmc)
pbmc <- FindVariableFeatures(object = pbmc)
pbmc <- ScaleData(object = pbmc)
pbmc <- RunPCA(object = pbmc)
pbmc <- FindNeighbors(object = pbmc, dims = 1:30)
pbmc <- FindClusters(object = pbmc)
pbmc <- RunUMAP(object = pbmc, dims = 1:30)
DimPlot(object = pbmc, reduction = "umap")
```

SCtransform version

```
pbmc <- SCTransform(object = pbmc)
pbmc <- RunPCA(object = pbmc)
pbmc <- FindNeighbors(object = pbmc, dims = 1:30)
pbmc <- FindClusters(object = pbmc)
pbmc <- RunUMAP(object = pbmc, dims = 1:30)
DimPlot(object = pbmc, reduction = "umap")
```

```
# note that you can chain multiple commands together with %>%
pbmc <- SCTransform(pbmc) %>%
    RunPCA() %>%
    FindNeighbors(dims = 1:30) %>%
    FindClusters() %>%
    RunUMAP(dims = 1:30)
```

Seurat Object Data Access

Cell, feature, and layer names

```
# Get cell and feature names, and total numbers We show multiple ways to get the same out
put cell names
colnames(pbmc)
Cells(pbmc)

# feature names
Features(pbmc)
rownames(pbmc)

# number of cells/features
num_cells <- ncol(pbmc)
num_features <- nrow(pbmc)

# List of object layers
Layers(pbmc)

# working with multimodal objects list assays
Assays(cbmc)

# Assay-specific features (genes/ADT)
Features(cbmc[["RNA"]])
Features(cbmc[["ADT"]])

# Variable feature names
VariableFeatures(pbmc)
```

```
# Set variable features
VariableFeatures(cbmc) <- var.gene.names

# set for a specific assay
VariableFeatures(cbmc[["ADT"]]) <- var.gene.names
```

Identity class labels

```
# Setting and retrieving cell identities

# Set identity classes to an existing column in meta data
Idents(object = pbmc) <- "seurat_annotations"

# View cell identities, get summary table
Idents(pbmc)
table(Idents(pbmc))

# Set identity to CD4 T cells for all cells
Idents(pbmc) <- "CD4 T cells"

# Set for a selected group of cells
pbmc.cells <- Cells(pbmc)
Idents(object = pbmc, cells = pbmc.cells[1:10]) <- "CD4 T cells"

# Get cell identity classes
Idents(object = pbmc)
levels(x = pbmc)

# Stash cell identity classes in metadata
pbmc[["old.ident"]] <- Idents(object = pbmc)
pbmc <- StashIdent(object = pbmc, save.name = "old.ident")

# Rename identity classes
pbmc <- RenameIdents(object = pbmc, `CD4 T cells` = "T Helper cells")
```

Cell metadata

```
# View metadata data frame, stored in object@meta.data
pbmc[[]]

# Retrieve specific values from the metadata
pbmc$nCount_RNA
pbmc[[c("percent.mito", "nFeature_RNA")]]

# Add metadata, see ?AddMetaData
random_group_labels <- sample(x = c("g1", "g2"), size = ncol(x = pbmc), replace = TRUE)
pbmc$groups <- random_group_labels
```

Expression data (stored as layers in Seurat v5)

```
# Retrieve data in an expression matrix RNA counts matrix
pbmc[["RNA"]]$counts

# Alternate accessor function with the same result
LayerData(pbmc, assay = "RNA", layer = "counts")

# GetAssayData from Seurat v4 is still supported
GetAssayData(object = pbmc, assay = "RNA", slot = "counts")

# ADT counts matrix (multimodal object)
cbmc[["ADT"]]$counts
```

```
# Set expression data assume new.data is a new expression matrix
pbmc[["RNA"]]$counts <- new.data

# Alternate setter function with the same result
LayerData(pbmc, assay = "RNA", layer = "counts") <- new.data

# SetAssayData from Seurat v4 is still supported
pbmc <- SetAssayData(object = pbmc, slot = "counts", new.data = new.data)
```

Dimensional reductions

```
# Get cell embeddings and feature loadings stored on pbmc[['pca']]@cell.embeddings
Embeddings(pbmc, reduction = "pca")

# stored in pbmc[['pca]]@feature.loadings
Loadings(pbmc, reduction = "pca")
```

```
# Create custom dimensional reduction loadings matrix is optional
new_reduction <- CreateDimReducObject(embeddings = new.embeddings, loadings = new.loading
s, key = "custom_pca")
pbmc[["custom_pca"]] <- new_reduction
```

FetchData

```
# FetchData can access anything from expression matrices, cell embeddings, or metadata Us
e the previously listed
# commands to access entire matrices Use FetchData to access individual/small groups of v
ariables
FetchData(object = pbmc, vars = c("PC_1", "nFeature_RNA", "MS4A1"), layer = "counts")
```

Subsetting and merging

Subset Seurat objects

```
# Subset Seurat object based on identity class, also see ?SubsetData
subset(x = pbmc, idents = "B")
subset(x = pbmc, idents = c("Naive CD4 T", "CD8 T"), invert = TRUE)

# Subset on the expression level of a gene/feature
subset(x = pbmc, subset = MS4A1 > 2.5)

# Subset on a combination of criteria
subset(x = pbmc, subset = MS4A1 > 2.5 & PC_1 > 5)
subset(x = pbmc, subset = MS4A1 > 2.5, idents = "B")

# Subset on a value in the object meta data
subset(x = pbmc, subset = groups == "g1")

# Downsample the number of cells per identity class
subset(x = pbmc, downsample = 100)
```

Split layers

```
# In Seurat v5, users can now split in object directly into different layers keeps expres
sion data in one object, but
# splits multiple samples into layers can proceed directly to integration workflow after
splitting layers
ifnb[["RNA"]] <- split(ifnb[["RNA"]], f = ifnb$stim)
Layers(ifnb)

# If desired, for example after intergation, the layers can be joined together again
ifnb <- JoinLayers(ifnb)
```

Split objects

```
# In line with prior workflows, you can also into split your object into a list of multip
le objects based on a metadata
# column creates a list of two objects
ifnb_list <- SplitObject(ifnb, split.by = "stim")
ifnb_list$CTRL
ifnb_list$STIM
```

Merge objects (without integration)

In Seurat v5, merging creates a single object, but keeps the expression information split into different layers for integration. If not proceeding with integration, rejoin the layers after merging.

```
# Merge two Seurat objects
merged_obj <- merge(x = ifnb_list$CTRL, y = ifnb_list$STIM)
merged_obj[["RNA"]] <- JoinLayers(merged_obj)

# Example to merge more than two Seurat objects
merge(x = pbmc1, y = list(pbmc2, pbmc3))
```

Merge objects (with integration)

See introduction to integration (/seurat/articles/integration_introduction) for more information.

```
merged_obj <- merge(x = ifnb_list$CTRL, y = ifnb_list$STIM)
merged_obj <- NormalizeData(merged_obj)
merged_obj <- FindVariableFeatures(merged_obj)
merged_obj <- ScaleData(merged_obj)
merged_obj <- RunPCA(merged_obj)
merged_obj <- IntegrateLayers(object = obj, method = RPCAIntegration, orig.reduction = "p
ca", new.reduction = "integrated.rpca",
    verbose = FALSE)

# now that integration is complete, rejoin layers
merged_obj[["RNA"]] <- JoinLayers(merged_obj)
```

Pseudobulk analysis

Group cells together, based on multiple categories

See DE vignette (/seurat/articles/de_vignette) for information on how to add the `donor_id` column to meta data.

```
# pseudobulk cells only by cell type
bulk <- AggregateExpression(ifnb, group.by = "seurat_annotations", return.seurat = TRUE)
Cells(bulk)

# pseudobulk cells by stimulation condition AND cell type
bulk <- AggregateExpression(ifnb, group.by = c("stim", "seurat_annotations"), return.seur
at = TRUE)
Cells(bulk)

# pseudobulk cells by stimulation condition AND cell type AND donor
bulk <- AggregateExpression(ifnb, group.by = c("stim", "seurat_annotations", "donor_id"),
return.seurat = TRUE)
Cells(bulk)
```

Visualization in `Seurat`

Seurat has a vast, ggplot2-based plotting library. All plotting functions will return a ggplot2 plot by default, allowing easy customization with ggplot2.

```
# Dimensional reduction plot
DimPlot(object = pbmc, reduction = "pca")

# Dimensional reduction plot, with cells colored by a quantitative feature Defaults to UM
AP if available
FeaturePlot(object = pbmc, features = "MS4A1")

# Scatter plot across single cells
FeatureScatter(object = pbmc, feature1 = "MS4A1", feature2 = "PC_1")
FeatureScatter(object = pbmc, feature1 = "MS4A1", feature2 = "CD3D")

# Scatter plot across individual features, repleaces CellPlot
CellScatter(object = pbmc, cell1 = "AGTCTACTAGGGTG", cell2 = "CACAGATGGTTTCT")

VariableFeaturePlot(object = pbmc)

# Violin and Ridge plots
VlnPlot(object = pbmc, features = c("LYZ", "CCL5", "IL32"))
RidgePlot(object = pbmc, feature = c("LYZ", "CCL5", "IL32"))
```

```
# Heatmaps (visualize scale.data slot)
DimHeatmap(object = pbmc, reduction = "pca", cells = 200)

# standard workflow
pbmc <- ScaleData(pbmc, features = heatmap_markers)
DoHeatmap(object = pbmc, features = heatmap_markers)

# sctransform workflow
pbmc <- GetResidual(pbmc, features = heatmap_markers)
DoHeatmap(object = pbmc, features = heatmap_markers)

# heatmap with maximum of 100 cells per group
DoHeatmap(pbmc, heatmap_markers, cells = subset(pbmc, downsample = 100))
```

```
# New things to try!  Note that plotting functions now return ggplot2 objects, so you can
add themes, titles, and
# options onto them
VlnPlot(object = pbmc, features = "MS4A1", split.by = "groups")
DotPlot(object = pbmc, features = c("LYZ", "CCL5", "IL32"), split.by = "groups")
FeaturePlot(object = pbmc, features = c("MS4A1", "CD79A"), blend = TRUE)
DimPlot(object = pbmc) + DarkTheme()
DimPlot(object = pbmc) + labs(title = "2,700 PBMCs clustered using Seurat and viewed\non
a two-dimensional UMAP")
```

Seurat provides many prebuilt themes that can be added to ggplot2 plots for quick customization

Theme	Function
`DarkTheme`	Set a black background with white text
`FontSize`	Set font sizes for various elements of a plot
`NoAxes`	Remove axes and axis text
`NoLegend`	Remove all legend elements
`RestoreLegend`	Restores a legend after removal
`RotatedAxis`	Rotates x-axis labels

```
# Plotting helper functions work with ggplot2-based scatter plots, such as DimPlot, Featu
rePlot, CellScatter, and
# FeatureScatter
plot <- DimPlot(object = pbmc) + NoLegend()

# HoverLocator replaces the former `do.hover` argument It can also show extra data throug
ht the `information` argument,
# designed to work smoothly with FetchData
HoverLocator(plot = plot, information = FetchData(object = pbmc, vars = c("ident", "PC_
1", "nFeature_RNA")))

# FeatureLocator replaces the former `do.identify`
select.cells <- FeatureLocator(plot = plot)

# Label points on a ggplot object
LabelPoints(plot = plot, points = TopCells(object = pbmc[["pca"]]), repel = TRUE)
```

Multi-Assay Features

With Seurat, you can easily switch between different assays at the single cell level (such as ADT counts from CITE-seq, or integrated/batch-corrected data). Most functions now take an assay parameter, but you can set a Default Assay to avoid repetitive statements.

```
cbmc <- CreateSeuratObject(counts = cbmc.rna)
# Add ADT data
cbmc[["ADT"]] <- CreateAssayObject(counts = cbmc.adt)
# Run analyses by specifying the assay to use
NormalizeData(object = cbmc, assay = "RNA")
NormalizeData(object = cbmc, assay = "ADT", method = "CLR")

# Retrieve and set the default assay
DefaultAssay(object = cbmc)
DefaultAssay(object = cbmc) <- "ADT"
DefaultAssay(object = cbmc)

# Pull feature expression from both assays by using keys
FetchData(object = cbmc, vars = c("rna_CD3E", "adt_CD3"))

# Plot data from multiple assays using keys
FeatureScatter(object = cbmc, feature1 = "rna_CD3E", feature2 = "adt_CD3")
```

Additional resources

Users who are particularly interested in some of the technical changes to data storage in Seurat v5 can explore the following resources:

- SeuratObject manual (https://mojaveazure.github.io/seurat-object/)
- Seurat v5 and Assay5 introductory vignette (/seurat/articles/seurat5_essential_commands)

▶ **Session Info**

Developed by Rahul Satija, Satija Lab and Collaborators.

Site built with pkgdown (https://pkgdown.r-lib.org/) 2.0.7.