Essential Commands

-Adithya Ayanam

A guide for Seurat v 3 - satija lab

Seurat Standard Workflow

The standard Seurat workflow

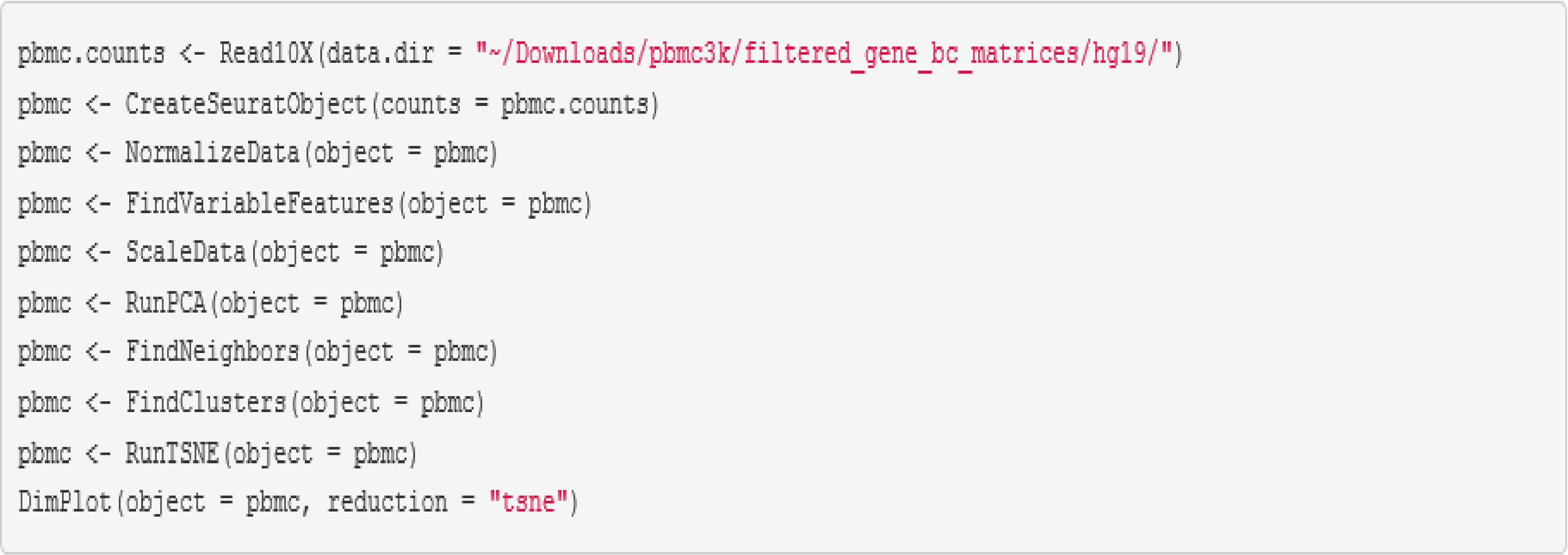
1. Examine raw single-cell expression data
2. Aims to find clusters within the data.

Process :

- Data normalization
- Variable feature selection,
- data scaling
- a PCA on variable features
- construction of a shared-nearest-neighbors graph,
- and clustering using a modularity optimizer.
- Finally, we use a t-SNE to visualize our clusters in a two-dimensional space.

Seurat Standard Workflow

Commands - Standard Workflow - Seurat v 3

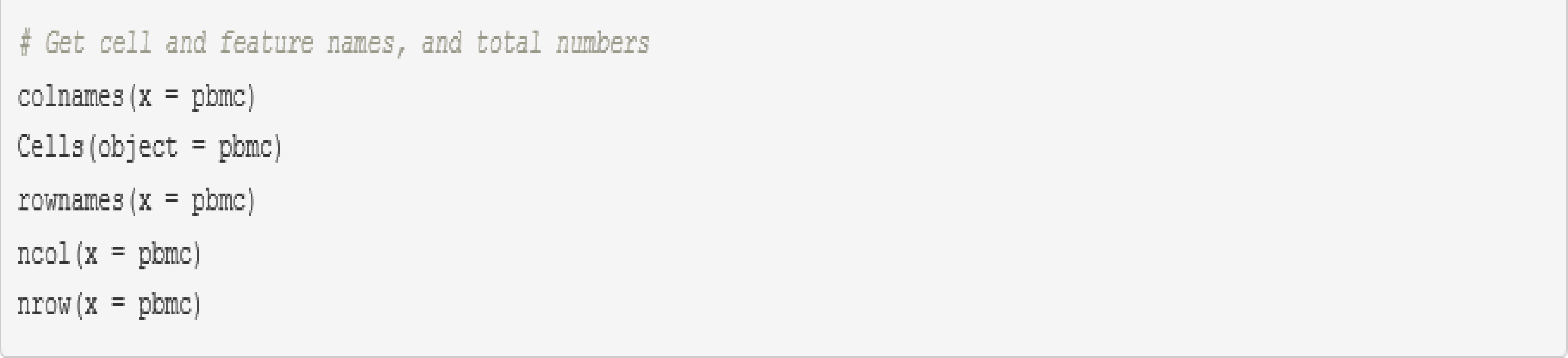


```
pbmc.counts <- Read10X(data.dir = "~/Downloads/pbmc3k/filtered_gene_bc_matrices/hg19/")
pbmc <- CreateSeuratObject(counts = pbmc.counts)
pbmc <- NormalizeData(object = pbmc)
pbmc <- FindVariableFeatures(object = pbmc)
pbmc <- ScaleData(object = pbmc)
pbmc <- RunPCA(object = pbmc)
pbmc <- FindNeighbors(object = pbmc)
pbmc <- FindClusters(object = pbmc)
pbmc <- RunTSNE(object = pbmc)
DimPlot(object = pbmc, reduction = "tsne")
```

Seurat Object Interaction

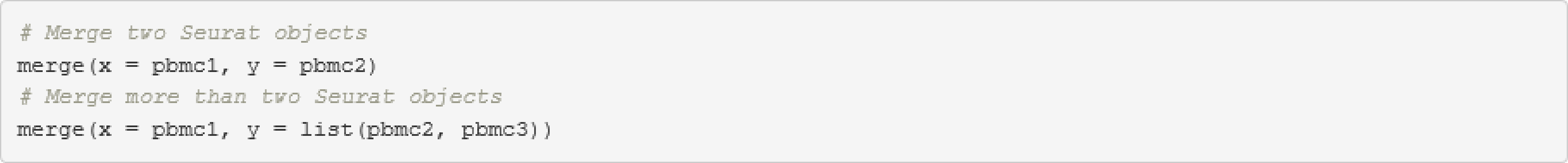
Commands - Object interaction - Seurat v 3

1. To Get cell and feature names, and total numbers



```
# Get cell and feature names, and total numbers
colnames(x = pbmc)
Cells(object = pbmc)
rownames(x = pbmc)
ncol(x = pbmc)
nrow(x = pbmc)
```

2. To Merge two datasets

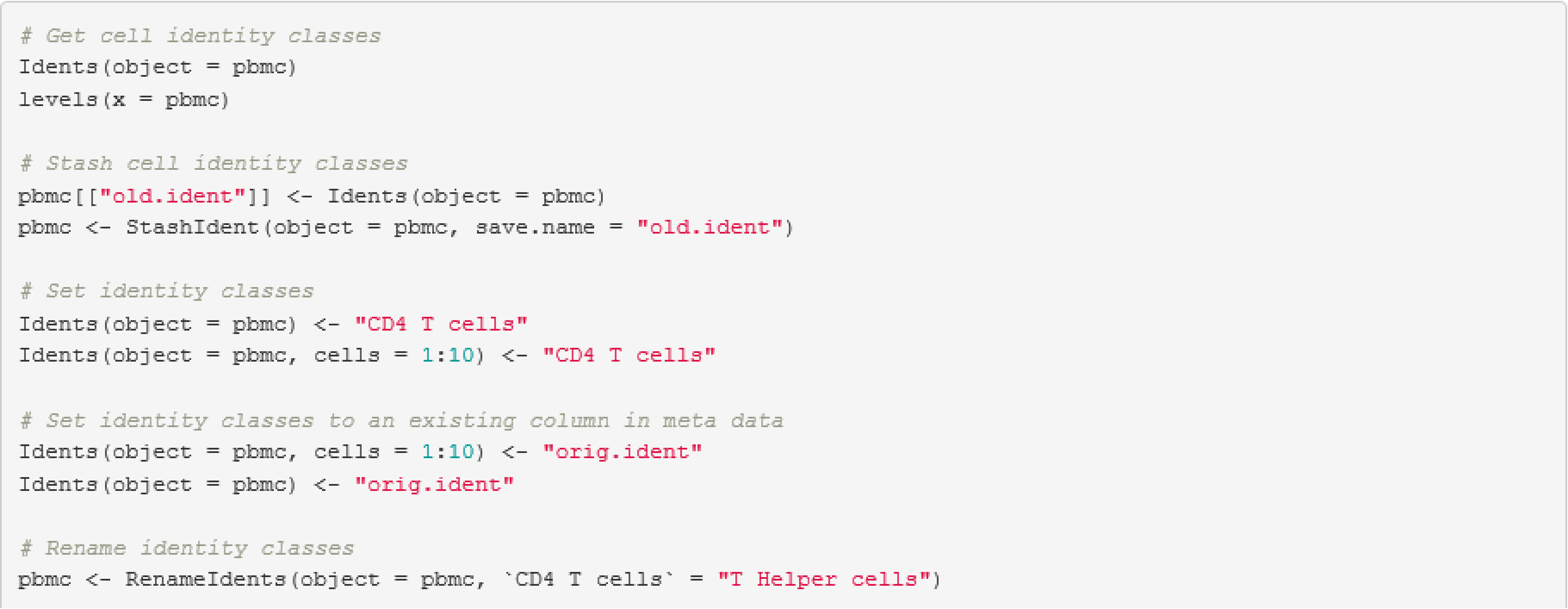


```
# Merge two Seurat objects
merge(x = pbmc1, y = pbmc2)
# Merge more than two Seurat objects
merge(x = pbmc1, y = list(pbmc2, pbmc3))
```

Seurat Object Interaction

Commands - Object interaction - Seurat v 3

3. Identity Classes

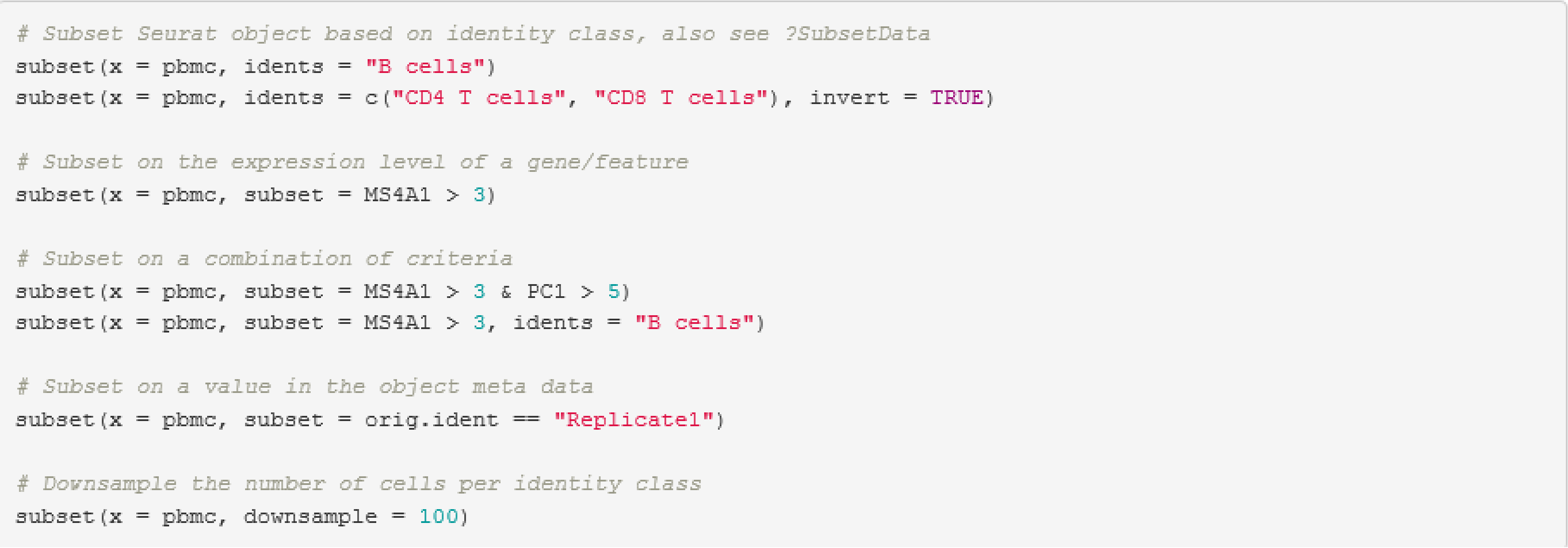


```
# Get cell identity classes
Idents(object = pbmc)
levels(x = pbmc)

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

# Set identity classes
Idents(object = pbmc) <- "CD4 T cells"
Idents(object = pbmc, cells = 1:10) <- "CD4 T cells"

# Set identity classes to an existing column in meta data
Idents(object = pbmc, cells = 1:10) <- "orig.ident"
Idents(object = pbmc) <- "orig.ident"

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

Seurat Object Interaction

Commands - Object interaction - Seurat v 3

3. Subset Command - (replacement for FilterCells command in older version)

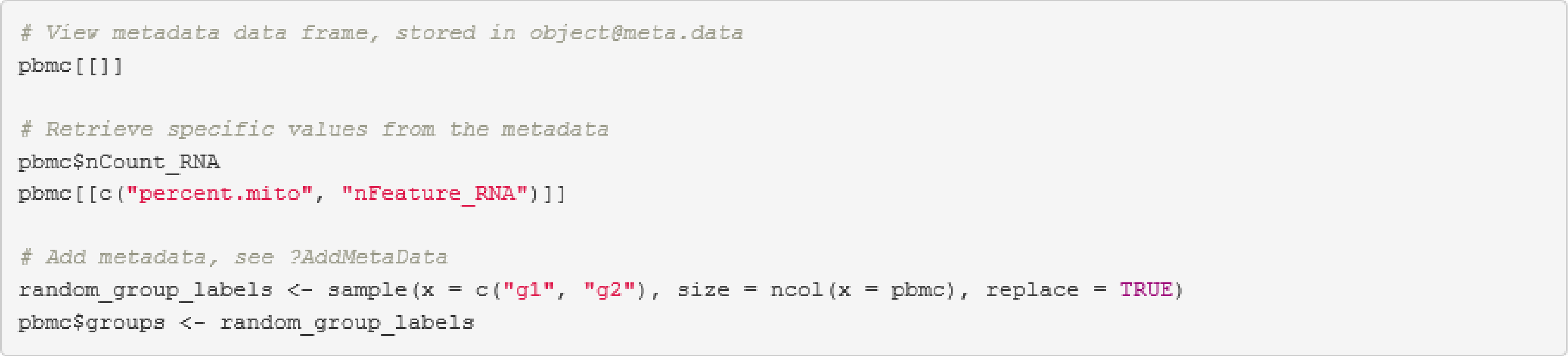


```
# Subset Seurat object based on identity class, also see ?SubsetData
subset(x = pbmc, idents = "B cells")
subset(x = pbmc, idents = c("CD4 T cells", "CD8 T cells"), invert = TRUE)

# Subset on the expression level of a gene/feature
subset(x = pbmc, subset = MS4A1 > 3)

# Subset on a combination of criteria
subset(x = pbmc, subset = MS4A1 > 3 & PC1 > 5)
subset(x = pbmc, subset = MS4A1 > 3, idents = "B cells")

# Subset on a value in the object meta data
subset(x = pbmc, subset = orig.ident == "Replicate1")

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

Data Access

Commands - Data Access - Seurat v 3

1. MetaData - Add, View and Retrieve

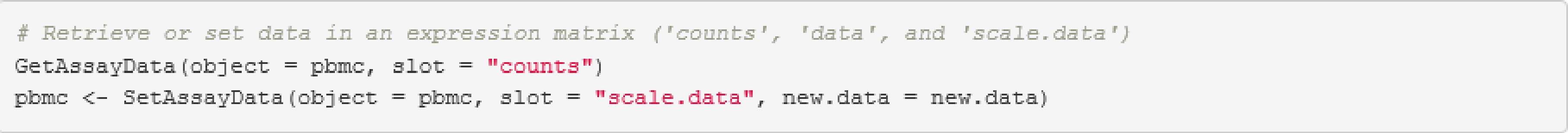


```
# View metadata data frame, stored in object@meta.data
pbmc[[]]

# 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
```

Data Access

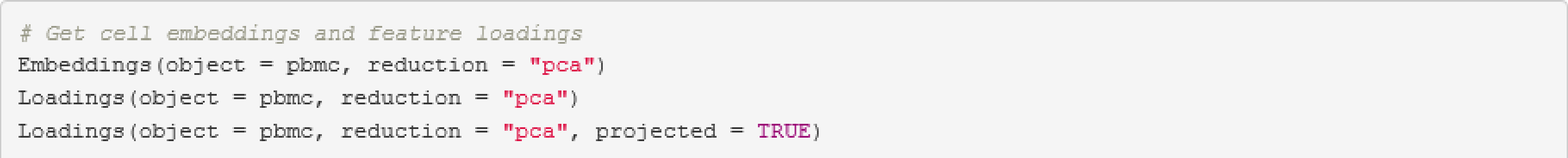
Commands - Data Access - Seurat v 3

2. Expression matrix - Set and Retrieve



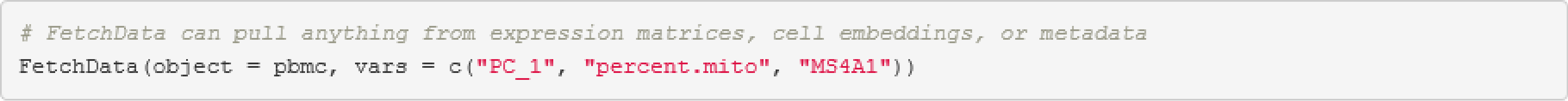
```
# Retrieve or set data in an expression matrix ('counts', 'data', and 'scale.data')
GetAssayData(object = pbmc, slot = "counts")
pbmc <- SetAssayData(object = pbmc, slot = "scale.data", new.data = new.data)
```

3. Cell Embeddings and Feature Loadings



```
# Get cell embeddings and feature loadings
Embeddings(object = pbmc, reduction = "pca")
Loadings(object = pbmc, reduction = "pca")
Loadings(object = pbmc, reduction = "pca", projected = TRUE)
```

4. Fetch Data

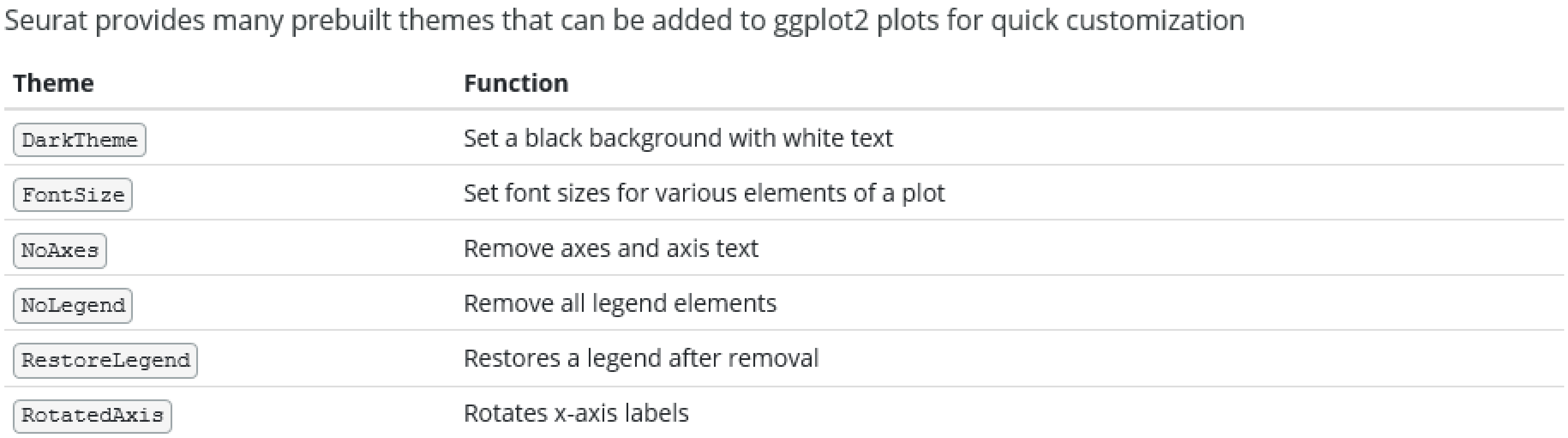


```
# FetchData can pull anything from expression matrices, cell embeddings, or metadata
FetchData(object = pbmc, vars = c("PC_1", "percent.mito", "MS4A1"))
```

Visualization

Commands - visualization- Seurat v 3

- **All plotting functions will return a ggplot2 plot by default**
- **Allow easy customization with ggplot2.**

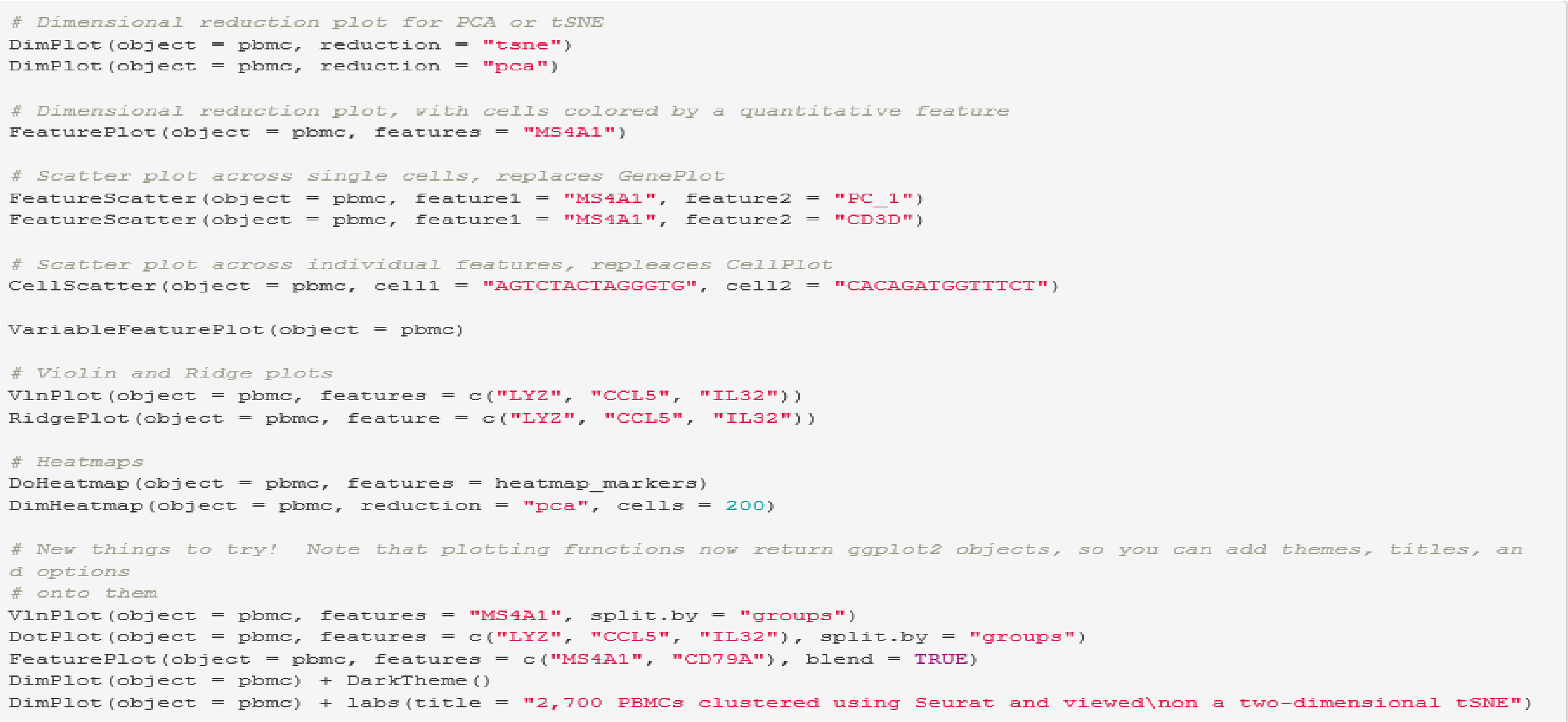


Seurat provides many prebuilt themes that can be added to ggplot2 plots for quick customization

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

Visualization

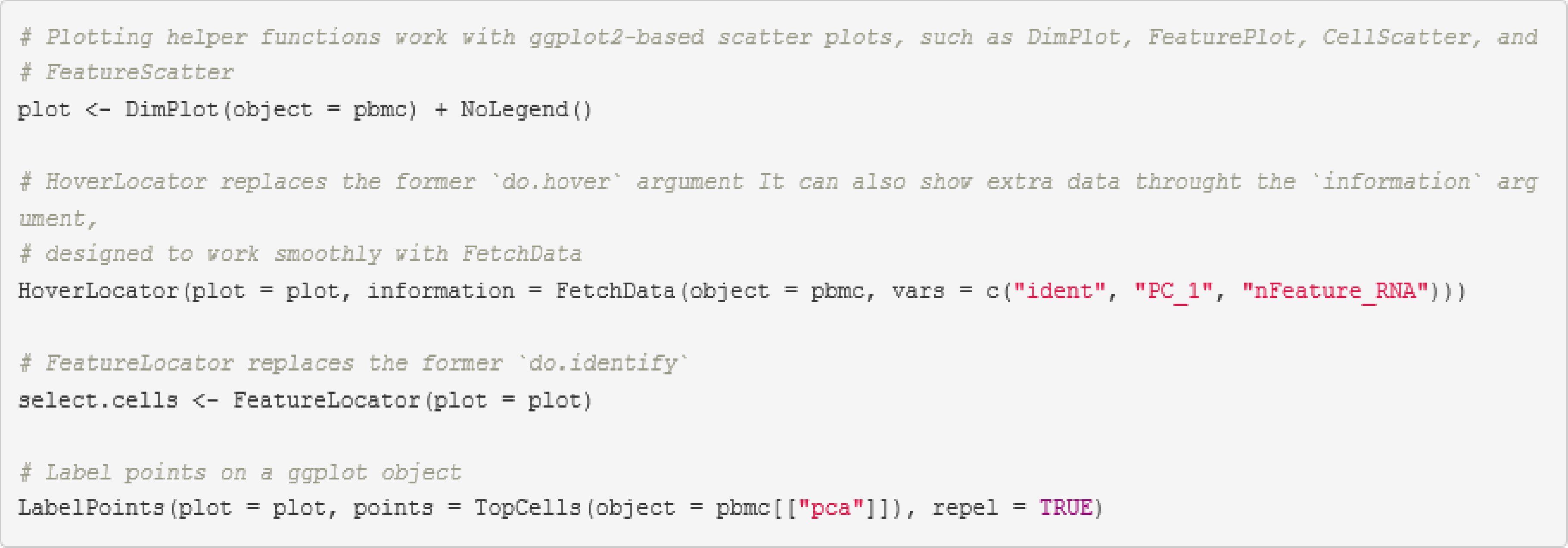
Commands - visualization- Seurat v 3 -All ggplot functions



```
# Dimensional reduction plot for PCA or tSNE
DimPlot(object = pbmc, reduction = "tsne")
DimPlot(object = pbmc, reduction = "pca")

# Dimensional reduction plot, with cells colored by a quantitative feature
FeaturePlot(object = pbmc, features = "MS4A1")

# Scatter plot across single cells, replaces GenePlot
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
DoHeatmap(object = pbmc, features = heatmap_markers)
DimHeatmap(object = pbmc, reduction = "pca", cells = 200)

# New things to try!  Note that plotting functions now return ggplot2 objects, so you can add themes, titles, an
d 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 tSNE")
```

Visualization

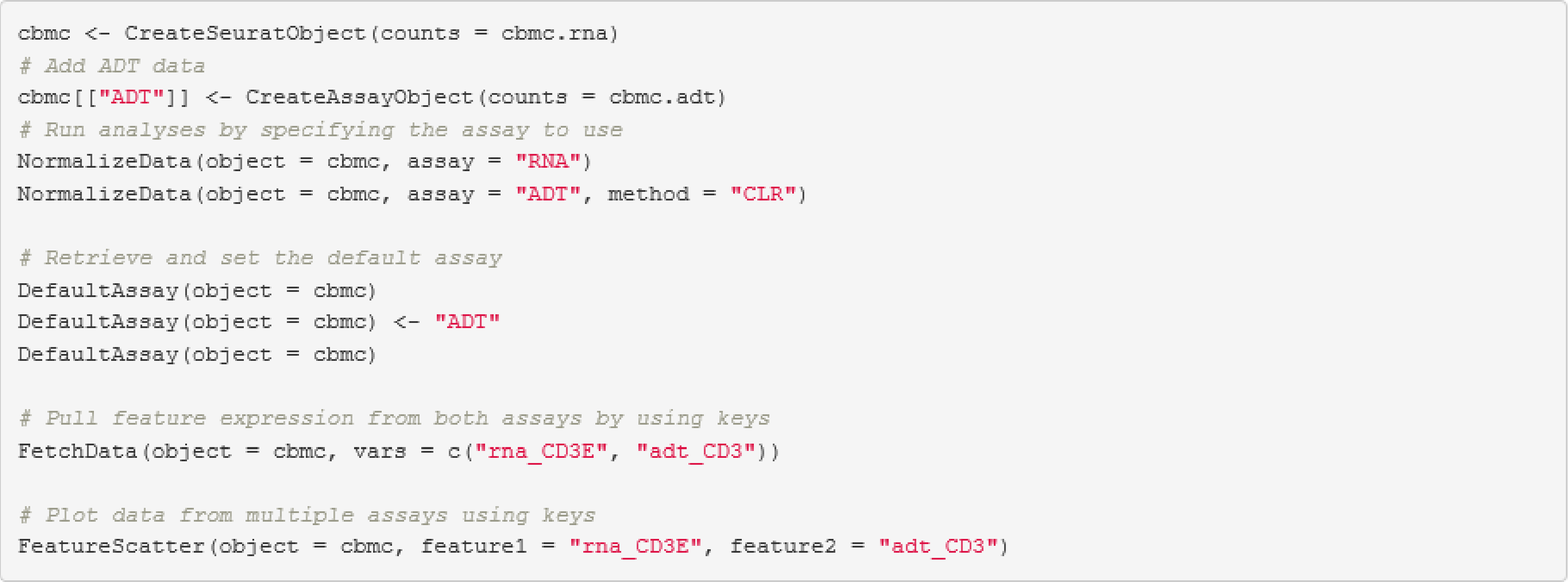
Commands - visualization- Seurat v 3 -All ggplot functions



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

# HoverLocator replaces the former `do.hover` argument It can also show extra data throught the `information` arg
ument,
# 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

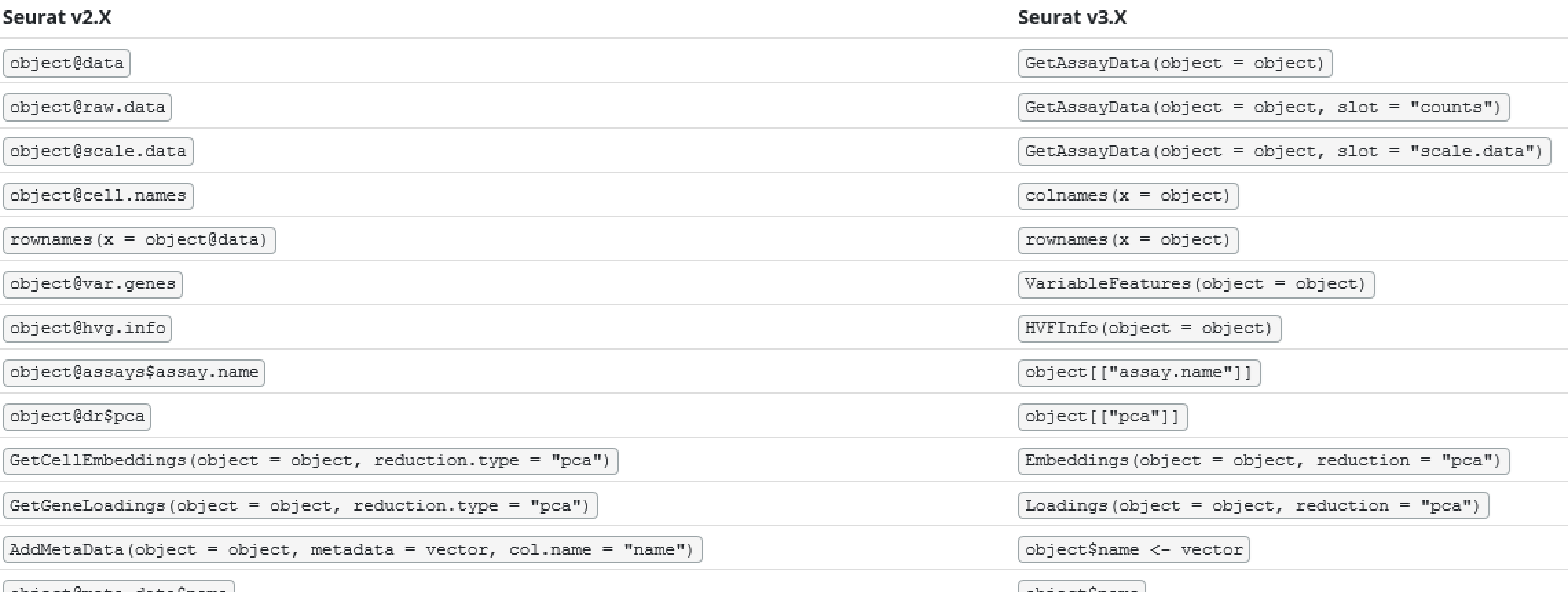
Commands - multi-assay features Seurat v 3 -All ggplot functions



```
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")
```

Ver 2 and Ver 3 Differences

Commands - Differences- Seurat v 3 -All ggplot functions

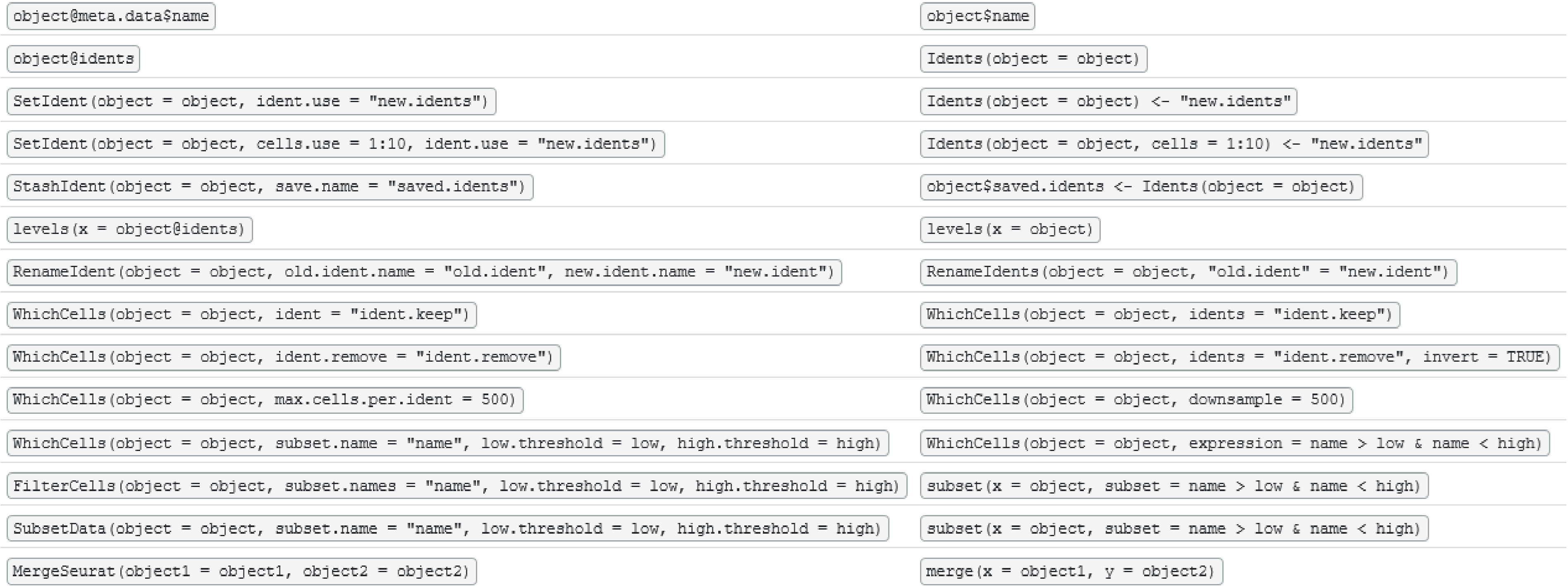


Seurat v2.X	Seurat v3.X
`object@data`	`GetAssayData(object = object)`
`object@raw.data`	`GetAssayData(object = object, slot = "counts")`
`object@scale.data`	`GetAssayData(object = object, slot = "scale.data")`
`object@cell.names`	`colnames(x = object)`
`rownames(x = object@data)`	`rownames(x = object)`
`object@var.genes`	`VariableFeatures(object = object)`
`object@hvg.info`	`HVFInfo(object = object)`
`object@assays$assay.name`	`object[["assay.name"]]`
`object@dr$pca`	`object[["pca"]]`
`GetCellEmbeddings(object = object, reduction.type = "pca")`	`Embeddings(object = object, reduction = "pca")`
`GetGeneLoadings(object = object, reduction.type = "pca")`	`Loadings(object = object, reduction = "pca")`
`AddMetaData(object = object, metadata = vector, col.name = "name")`	`object$name <- vector`
`object@meta.data$name`	`object$name`

Ver 2 and Ver 3 Differences

Commands - Differences- Seurat v 3 -All ggplot functions

Ver 2



`object@meta.data$name`	`object$name`
`object@idents`	`Idents(object = object)`
`SetIdent(object = object, ident.use = "new.idents")`	`Idents(object = object) <- "new.idents"`
`SetIdent(object = object, cells.use = 1:10, ident.use = "new.idents")`	`Idents(object = object, cells = 1:10) <- "new.idents"`
`StashIdent(object = object, save.name = "saved.idents")`	`object$saved.idents <- Idents(object = object)`
`levels(x = object@idents)`	`levels(x = object)`
`RenameIdent(object = object, old.ident.name = "old.ident", new.ident.name = "new.ident")`	`RenameIdents(object = object, "old.ident" = "new.ident")`
`WhichCells(object = object, ident = "ident.keep")`	`WhichCells(object = object, idents = "ident.keep")`
`WhichCells(object = object, ident.remove = "ident.remove")`	`WhichCells(object = object, idents = "ident.remove", invert = TRUE)`
`WhichCells(object = object, max.cells.per.ident = 500)`	`WhichCells(object = object, downsample = 500)`
`WhichCells(object = object, subset.name = "name", low.threshold = low, high.threshold = high)`	`WhichCells(object = object, expression = name > low & name < high)`
`FilterCells(object = object, subset.names = "name", low.threshold = low, high.threshold = high)`	`subset(x = object, subset = name > low & name < high)`
`SubsetData(object = object, subset.name = "name", low.threshold = low, high.threshold = high)`	`subset(x = object, subset = name > low & name < high)`
`MergeSeurat(object1 = object1, object2 = object2)`	`merge(x = object1, y = object2)`