Seurat 치트 시트

표준 Seurat 워크플로우

```
library(Seurat)
# pbmc 예시 데이터 불러오기
pbmc ← LoadData('pbmc3k')

# 사례/대조군 예시 데이터 불러오기
# 더 많은 정보는 ?ifnb 입력
ifnb ← LoadData('ifnb')

pbmc ← NormalizeData(object = pbmc) # 데이터 정규화
pbmc ← FindVariableFeatures(object = pbmc) # 고변동 특징 찾기
pbmc ← ScaleData(object = pbmc) # 데이터 스케일링
pbmc ← RunPCA(object = pbmc) # 주성분 분석(PCA) 실행
pbmc ← FindNeighbors(object = pbmc, dims = 1:30) # 이웃 찾기
pbmc ← FindClusters(object = pbmc) # 클러스터 찾기
pbmc ← RunUMAP(object = pbmc, dims = 1:30) # UMAP 실행
DimPlot(object = pbmc, reduction = 'umap') # UMAP 시각화
```

SCtransform 버전

```
pbmc ← SCTransform(object = pbmc) # SCTransform 적용
pbmc ← RunPCA(object = pbmc) # PCA 실행
pbmc ← FindNeighbors(object = pbmc, dims = 1:30) # 이웃 찾기
pbmc ← FindClusters(object = pbmc) # 클러스터 찾기
pbmc ← RunUMAP(object = pbmc, dims = 1:30) # UMAP 실행
DimPlot(object = pbmc, reduction = 'umap') # UMAP 시각화
# %>%를 사용하여 여러 명령을 연결할 수 있습니다.
pbmc ← SCTransform(pbmc) %>% RunPCA() %>% FindNeighbors(dims=1:30) %>%
  FindClusters() %>% RunUMAP(dims=1:30)
```

Seurat 객체 데이터 접근

셀, 특징, 그리고 레이어 이름

```
# 셀 및 특징 이름과 총 개수 가져오기
# 동일한 출력을 얻는 여러 방법을 보여줍니다.
# 셀 이름
colnames(pbmc)
Cells(pbmc)

# 특징(유전자) 이름
Features(pbmc)
rownames(pbmc)

# 셀/특징 수
num_cells ← ncol(pbmc)
num_features ← nrow(pbmc)

# 객체 레이어 목록
Layers(pbmc)

# 다중 모달 객체 작업
# 어세이 목록
Assays(cbmc)

# 어세이별 특징 (유전자/ADT)
Features(cbmc[["RNA"]])
Features(cbmc[["ADT"]])

# 고변동 특징 이름
VariableFeatures(pbmc)

# 고변동 특징 설정
VariableFeatures(cbmc) ← var.gene.names

# 특정 어세이에 설정
VariableFeatures(cbmc[["ADT"]]) ← var.gene.names
```

아이덴티티 클래스 레이블

```
# 셀 아이덴티티 설정 및 검색

# 메타데이터의 기존 열로 아이덴티티 클래스 설정
Idents(object = pbmc) ← 'seurat_annotations'

# 셀 아이덴티티 보기, 요약 테이블 가져오기
Idents(pbmc)
table(Idents(pbmc))

# 모든 셀에 대해 아이덴티티를 CD4 T 세포로 설정
Idents(pbmc) ← 'CD4 T cells'

# 선택된 셀 그룹에 설정
pbmc.cells ← Cells(pbmc)
Idents(object = pbmc, cells = pbmc.cells[1:10]) ← 'CD4 T cells'

# 셀 아이덴티티 클래스 가져오기
Idents(object = pbmc)
levels(x = pbmc)

# 셀 아이덴티티 클래스를 메타데이터에 저장
pbmc[['old.ident']] ← Idents(object = pbmc)
pbmc ← StashIdent(object = pbmc, save.name = 'old.ident')

# 아이덴티티 클래스 이름 변경
pbmc ← RenameIdents(object = pbmc, 'CD4 T cells' = 'T Helper cells')
```

셀 메타데이터

```
# object@meta.data에 저장된 메타데이터 데이터 프레임 보기
pbmc[[]]

# 메타데이터에서 특정 값 검색
pbmc$nCount_RNA
pbmc[[c('percent.mito', 'nFeature_RNA')]]

# 메타데이터 추가, ?AddMetaData 참조
random_group_labels ← sample(x = c('g1', 'g2'), size = ncol(x = pbmc), replace = TRUE)
pbmc$groups ← random_group_labels
```

발현 데이터 (Seurat v5에서 레이어로 저장됨)

```
# 발현 매트릭스에서 데이터 검색
# RNA 카운트 매트릭스
pbmc[["RNA"]]$counts

# 동일한 결과를 가진 대체 접근자 함수
LayerData(pbmc,assay = 'RNA',layer = 'counts')

# Seurat v4의 GetAssayData는 여전히 지원됩니다.
GetAssayData(object = pbmc, assay = 'RNA', slot = 'counts')

# ADT 카운트 매트릭스 (다중 모달 객체)
cbmc[["ADT"]]$counts

# 발현 데이터 설정
# new.data가 새로운 발현 매트릭스라고 가정
pbmc[["RNA"]]$counts ← new.data

# 동일한 결과를 가진 대체 설정자 함수
LayerData(pbmc,assay = 'RNA',layer = 'counts') ← new.data

# Seurat v4의 SetAssayData는 여전히 지원됩니다.
pbmc ← SetAssayData(object = pbmc, slot = 'counts',new.data = new.data)
```

차원 축소

```
# 셀 임베딩 및 특징 로딩 가져오기
# pbmc[["pca"]]@cell.embeddings에 저장됨
Embeddings(pbmc, reduction = 'pca')

# pbmc[["pca]]@feature.loadings에 저장됨
Loadings(pbmc, reduction = 'pca')

# 사용자 정의 차원 축소 생성
# 로딩 매트릭스는 선택 사항입니다.
new_reduction ← CreateDimReducObject(embeddings = new.embeddings,loadings = new.loadings, key = "custom_pca")
pbmc[["custom_pca"]] ← new_reduction
```

FetchData

```
# FetchData는 발현 매트릭스, 셀 임베딩 또는 메타데이터의 모든 것에 접근할 수 있습니다.
# 전체 매트릭스에 접근하려면 이전에 나열된 명령을 사용하십시오.
# 개별/소그룹 변수에 접근하려면 FetchData를 사용하십시오.
FetchData(object = pbmc, vars = c('PC_1', 'nFeature_RNA', 'MS4A1'),layer = 'counts')
```

부분 집합화 및 병합

Seurat 객체 부분 집합화

```
# 아이덴티티 클래스를 기반으로 Seurat 객체 부분 집합화, ?SubsetData 참조
subset(x = pbmc, idents = 'B')
subset(x = pbmc, idents = c('Naive CD4 T', 'CD8 T'), invert = TRUE)

# 유전자/특징의 발현 수준에 따라 부분 집합화
subset(x = pbmc, subset = MS4A1 > 2.5)

# 여러 기준의 조합에 따라 부분 집합화
subset(x = pbmc, subset = MS4A1 > 2.5 & PC_1 > 5)
subset(x = pbmc, subset = MS4A1 > 2.5, idents = 'B')

# 객체 메타데이터의 값에 따라 부분 집합화
subset(x = pbmc, subset = groups == "g1")

# 아이덴티티 클래스당 셀 수 다운샘플링
subset(x = pbmc, downsample = 100)
```

레이어 분할

```
# Seurat v5에서는 사용자가 이제 객체를 다른 레이어로 직접 분할할 수 있습니다.
# 발현 데이터는 하나의 객체에 유지되지만, 여러 샘플을 레이어로 분할합니다.
# 레이어를 분할한 후 통합 워크플로우를 바로 진행할 수 있습니다.
ifnb[["RNA"]] <- split(ifnb[["RNA"]],f = ifnb$stim)
Layers(ifnb)

# 원하는 경우, 예를 들어 통합 후 레이어를 다시 결합할 수 있습니다.
ifnb <- JoinLayers(ifnb)
```

객체 분할

```
# 이전 워크플로우와 마찬가지로, 메타데이터 열을 기반으로 객체를 여러 객체의 목록으로 분할할 수도 있습니다.
# 두 개의 객체 목록을 생성합니다.
ifnb_list <- SplitObject(ifnb,split.by = 'stim')
ifnb_list$CTRL
ifnb_list$STIM
```

객체 병합 (통합 없이)

Seurat v5에서는 병합 시 단일 객체를 생성하지만, 발현 정보는 통합을 위해 다른 레이어로 분할된 상태로 유지됩니다. 통합을 진행하지 않을 경우, 병합 후 레이어를 다시 결합하십시오.

```
# 두 Seurat 객체 병합
merged_obj <- merge(x = ifnb_list$CTRL, y = ifnb_list$STIM)
merged_obj[["RNA"]] <- JoinLayers(merged_obj)

# 두 개 이상의 Seurat 객체를 병합하는 예시
merge(x = pbmc1, y = list(pbmc2, pbmc3))
```

객체 병합 (통합 포함)

자세한 정보는 [통합 소개](https://satijalab.org/seurat/articles/integration_introduction.html)를 참조하십시오.

```
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 = "pca", new.reduction = 'integrated.rpca',
  verbose = FALSE)

# 이제 통합이 완료되었으므로 레이어를 다시 결합합니다.
merged_obj[["RNA"]] <- JoinLayers(merged_obj)
```

유사 벌크 분석 (Pseudobulk analysis)

여러 범주를 기반으로 셀 그룹화

`donor_id` 열을 메타데이터에 추가하는 방법에 대한 정보는 [DE 안내서](https://satijalab.org/seurat/articles/de_vignette.html)를 참조하십시오.

```
# 셀 유형별로만 유사 벌크화
bulk <- AggregateExpression(ifnb,group.by ='seurat_annotations', return.seurat = TRUE)
Cells(bulk)

# 자극 조건 및 셀 유형별로 유사 벌크화
bulk <- AggregateExpression(ifnb,group.by = c('stim','seurat_annotations'), return.seurat = TRUE)
Cells(bulk)

# 자극 조건, 셀 유형 및 기증자별로 유사 벌크화
bulk <- AggregateExpression(ifnb,group.by = c('stim','seurat_annotations',"donor_id"), return.seurat = TRUE)
Cells(bulk)
```

Seurat에서의 시각화

Seurat에는 방대한 ggplot2 기반 플로팅 라이브러리가 있습니다. 모든 플로팅 함수는 기본적으로 ggplot2 플롯을 반환하므로 ggplot2를 사용하여 쉽게 사용자 정의할 수 있습니다.

```
# 차원 축소 플롯
DimPlot(object = pbmc, reduction = 'pca')

# 정량적 특징으로 셀에 색을 입힌 차원 축소 플롯
# 가능한 경우 UMAP으로 기본 설정됩니다.
FeaturePlot(object = pbmc, features = "MS4A1")

# 단일 셀 전반의 산점도
FeatureScatter(object = pbmc, feature1 = "MS4A1", feature2 = "PC_1")
FeatureScatter(object = pbmc, feature1 = "MS4A1", feature2 = "CD3D")

# 개별 특징 전반의 산점도, CellPlot을 대체합니다.
CellScatter(object = pbmc, cell1 = "AGTCTACTAGGGTG", cell2 = "CACAGATGGTTTCT")

VariableFeaturePlot(object = pbmc)

# 바이올린 및 릿지 플롯
VlnPlot(object = pbmc, features = c("LYZ", "CCL5", "IL32"))
RidgePlot(object = pbmc, feature = c("LYZ", "CCL5", "IL32"))

# 히트맵 (scale.data 슬롯 시각화)
DimHeatmap(object = pbmc,reduction = 'pca', cells = 200)

# 표준 워크플로우
pbmc <- ScaleData(pbmc,features = heatmap_markers)
DoHeatmap(object = pbmc,features = heatmap_markers)

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

# 그룹당 최대 100개의 셀을 포함하는 히트맵
DoHeatmap(pbmc,heatmap_markers,cells = subset(pbmc,downsample = 100))

# 플로팅 함수는 이제 ggplot2 객체를 반환하므로 테마, 제목 및 옵션을 추가할 수 있습니다.
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 = 'Seurat을 사용하여 클러스터링하고\n두 차원 UMAP에 표시된 2,700개 PBMC')
```

Seurat은 ggplot2 플롯에 추가하여 빠르게 사용자 정의할 수 있는 많은 사전 빌드된 테마를 제공합니다.

```
# 플로팅 헬퍼 함수는 DimPlot, FeaturePlot, CellScatter, FeatureScatter와 같은 ggplot2 기반 산점도에서 작동합니다.
plot <- DimPlot(object = pbmc) + NoLegend()

# HoverLocator는 이전의 `do.hover` 인수를 대체합니다.
```

```
# FetchData와 원활하게 작동하도록 설계된
`information` 인수를 통해 추가 데이터를 표시
할 수도 있습니다.
HoverLocator(plot = plot, information =
FetchData(object = pbmc, vars =
c("ident", "PC_1", "nFeature_RNA")))

# FeatureLocator는 이전의 `do.identify`를
대체합니다.
select.cells <- FeatureLocator(plot =
plot)

# ggplot 객체에 포인트 레이블 지정
LabelPoints(plot = plot, points =
TopCells(object = pbmc[["pca"]]), repel
= TRUE)
```

다중 어세이 특징

Seurat을 사용하면 단일 셀 수준에서 다양한 어세이(예: CITE-seq의 ADT 카운트 또는 통합/배치 보정 데이터) 간에 쉽게 전환할 수 있습니다. 대부분의 함수는 이제 어세이 매개변수를 받지만, 반복적인 문장을 피하기 위해 기본 어세이를 설정할 수 있습니다.

```
cbmc <- CreateSeuratObject(counts =
cbmc.rna)
# ADT 데이터 추가
cbmc[['ADT']] <-
CreateAssayObject(counts = cbmc.adt)
# 사용할 어세이를 지정하여 분석 실행
NormalizeData(object = cbmc, assay =
'RNA')
NormalizeData(object = cbmc, assay =
'ADT', method = 'CLR')

# 기본 어세이 검색 및 설정
DefaultAssay(object = cbmc)
DefaultAssay(object = cbmc) <- 'ADT'
DefaultAssay(object = cbmc)

# 키를 사용하여 두 어세이에서 특징 발현 가져오
기
FetchData(object = cbmc, vars =
c('rna_CD3E', 'adt_CD3'))

# 키를 사용하여 여러 어세이에서 데이터 플로팅
FeatureScatter(object = cbmc, feature1 =
"rna_CD3E", feature2 = "adt_CD3")
```