Running KKLClustering

Large-scale single-cell clustering algorithm based on K-means and optimal graph structure

Ren Jun | renjun0324@hotmail.com

Vignette built on 2021-09-07 21:31:18 with KKLClustering version 0.2.0.

KKLClustering Workflow

In order to find the best clustering result while increasing the computing speed, we designed an algorithm process that combines outlier detection based on region division and louvain clustering based on random sampling. After obtaining the original data, a certain percentage of genes can be screened. Of course, we also encourage the use of principal component analysis (PCA) results for the follow-up process.

Test Data

In order to get started quickly, we provide a small public data set GSE60361(Lake et al. 2017), which including count matrix with 2844 cell * 18877 gene and cell type labels.

```
suppressMessages(library(KKLClustering))

data(count)
knitr::kable(as.matrix(count)[1:5,1:3])
#> Loading required package: Matrix
```

	1772071015_C02	1772071017_G12	1772071017_A05
Tspan12	0	0	0
Tshz1	3	1	0
Fnbp1l	3	1	6
Adamts15	0	0	0
Cldn12	1	1	1

```
data(cellinfo)
knitr::kable(head(cellinfo,3))
```

	celltype
1772071015_C02	Interneurons
1772071017_G12	Interneurons
1772071017_A05	Interneurons

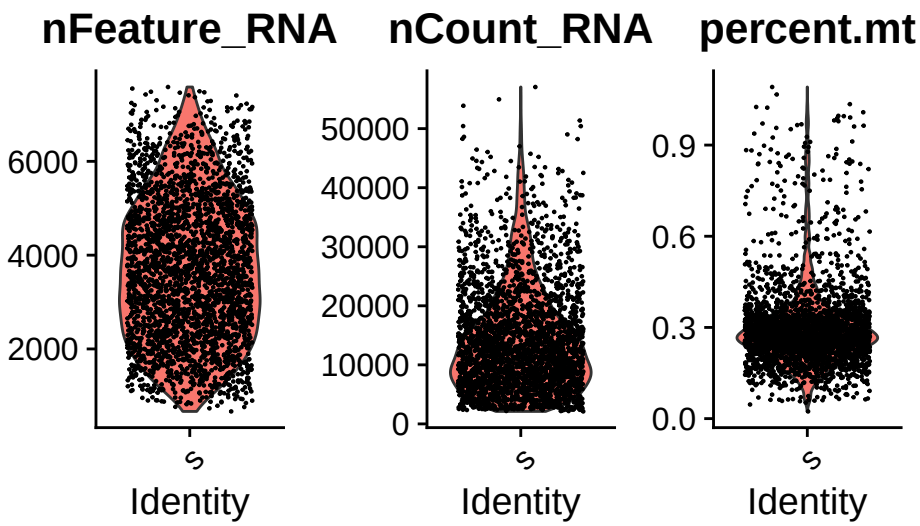
Data Preprocessing

We use Seurat(Butler et al. 2018; Satija et al. 2015) for standard filtering, and the detailed process can refer to (seurat workflow)

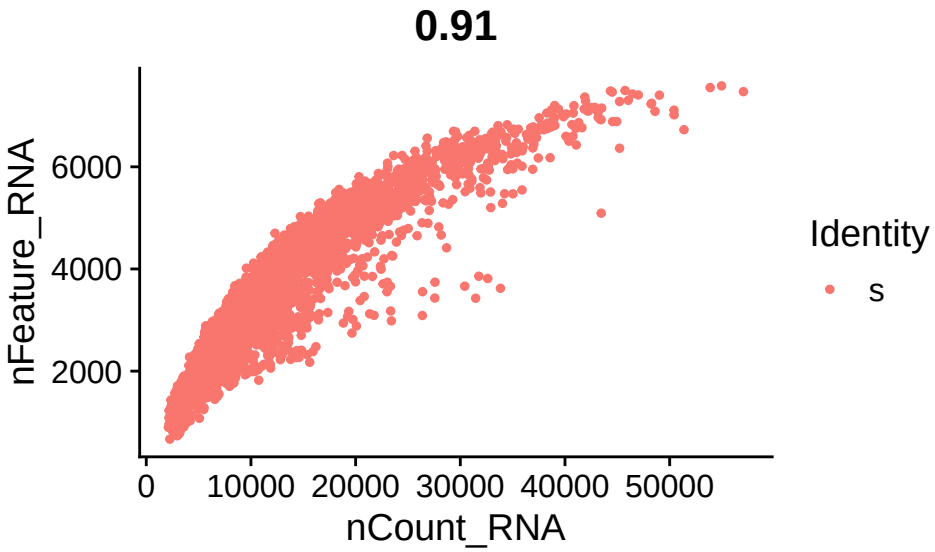
```
suppressMessages(library(Seurat))
suppressMessages(library(ggplot2))
suppressMessages(library(ggsci))

setwd("~/KKLClustering/test")

load("s.rda")
load("outlier_df.rda")
load("result.rda")
```

```
s <- CreateSeuratObject(count, min.cells = 200, min.features = 3, meta.data = cellinfo)
s <- PercentageFeatureSet(s, pattern = "^Mt", col.name = "percent.mt")
s <- SCTransform(s, variable.features.n = 2000, verbose = FALSE)

s <- RunPCA(s, features = VariableFeatures(object = s), npcs = 30, verbose = FALSE)
pca.c <- pcaMarchenkoPastur(M = length(VariableFeatures(s)),
                            N = ncol(s),
                            pca.sdev = s@reductions$pca@stdev,
                            factor = 1)

s <- RunTSNE(s, reduction = "pca", dims = which(pca.c), verbose = FALSE)
s <- RunUMAP(s, reduction = "pca", dims = which(pca.c), verbose = FALSE)
```

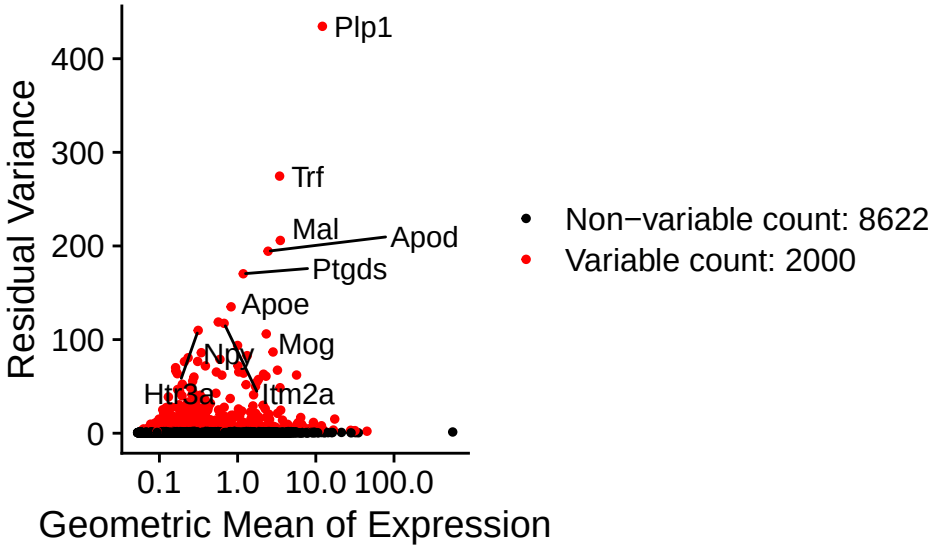
```
Idents(s) <- "s"
VlnPlot(object = s, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"))
```



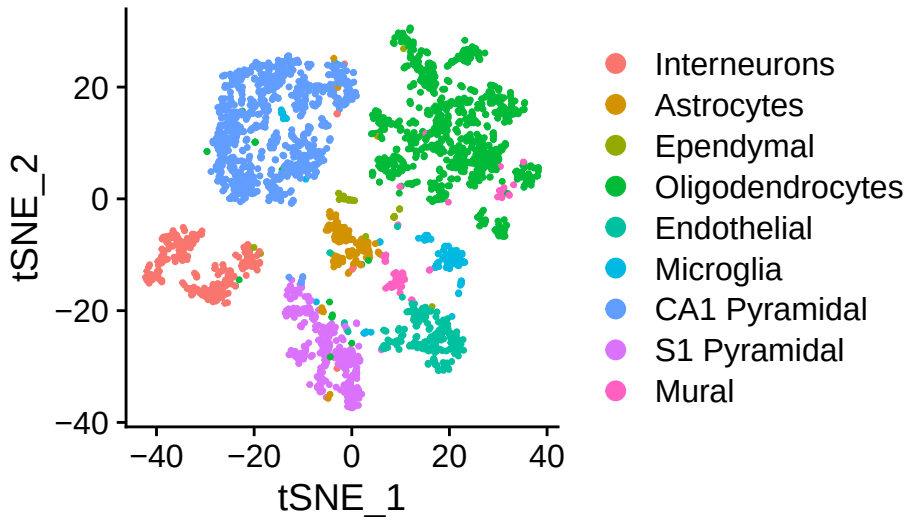

```
FeatureScatter(s, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



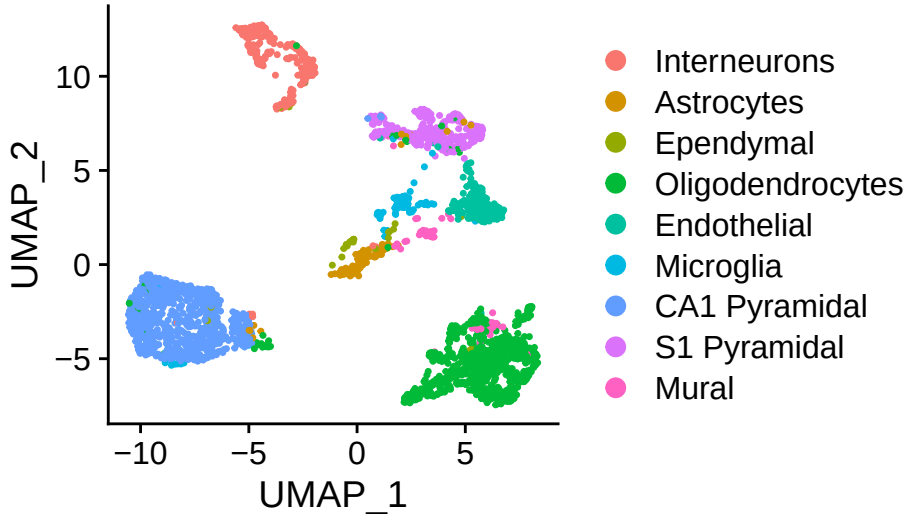

```
top10 <- head(x = VariableFeatures(s), 10)
plot1 <- VariableFeaturePlot(s)
LabelPoints(plot = plot1, points = top10, repel = TRUE)
#> When using repel, set xnudge and ynudge to 0 for optimal results
```




```
Idents(s) <- s[["celltype"]]
DimPlot(s, reduction = "tsne")
```

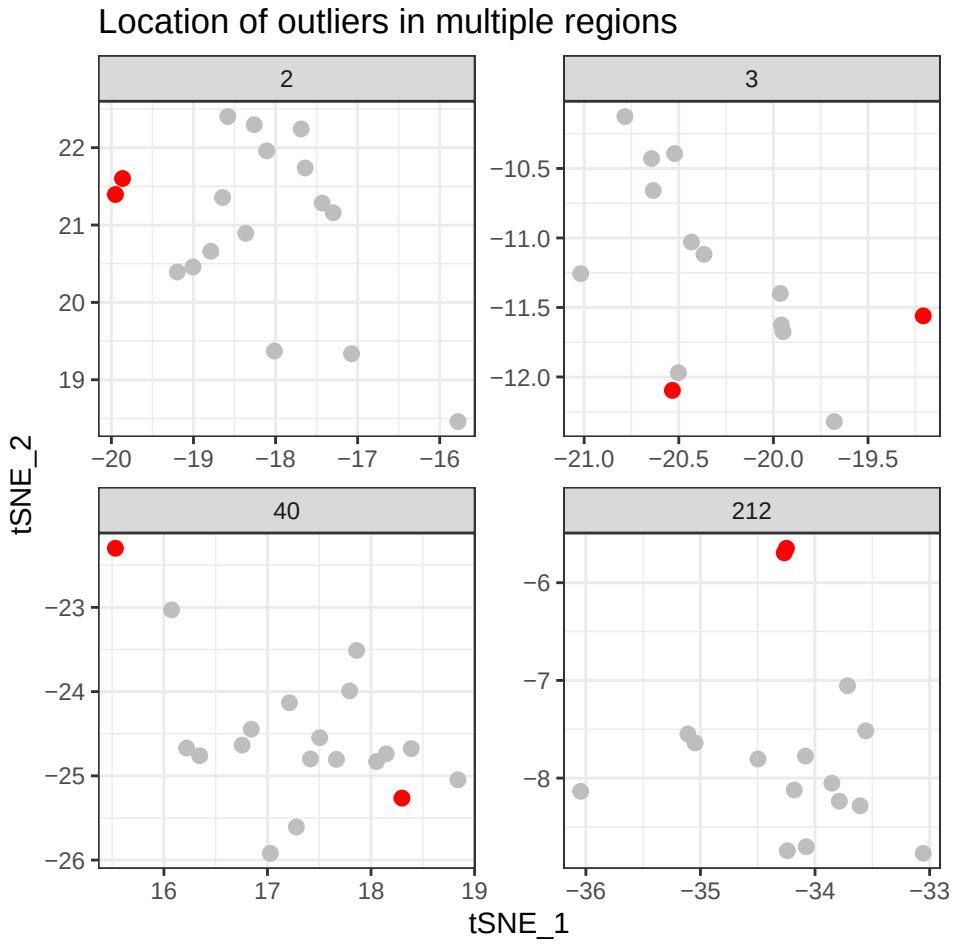



```
DimPlot(s, reduction = "umap")
```



The downsampling method based on K-means region division

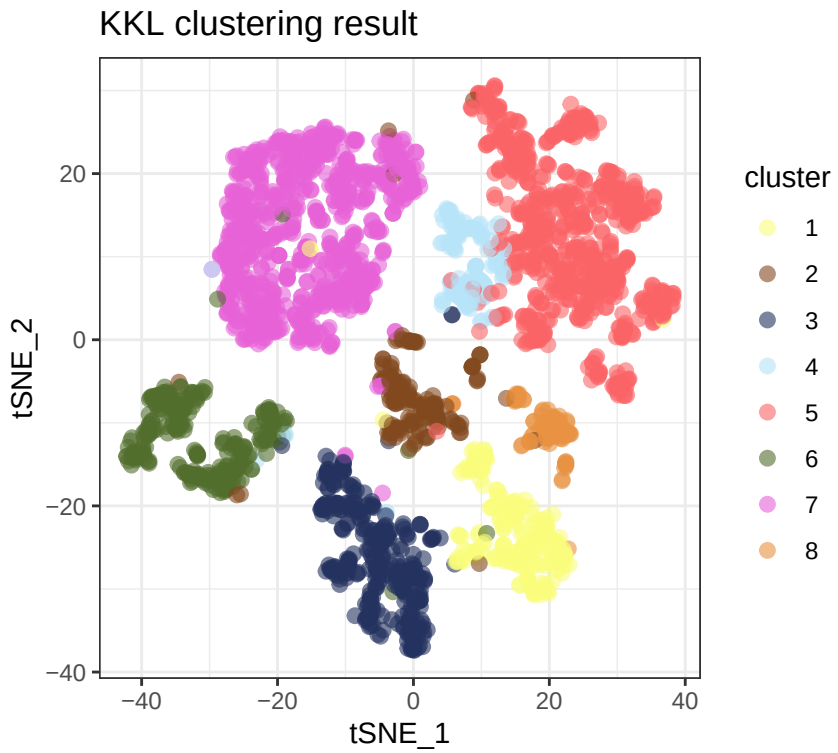
```
df2 <- outlier_df %>% dplyr::filter(cluster %in% c(2,3,40,212))
ggplot(df2, aes(x = tSNE_1, y = tSNE_2, color = I(outlier))) +
  geom_point(size = 2.3) +
  facet_wrap(~cluster, scales = "free") +
  labs(title = "Location of outliers in multiple regions") +
  theme_bw()
```



The Louvain algorithm based on the optimal KNN graph structure

The R package Clustercrit(Desgraupes 2013) provides a variety of clustering evaluation indicators, we can choose any one to evaluate different clustering results, so as to select the most KNN graph structure. If the number of K-means area divisions is less than 500, Davies_Bouldin has a good evaluation effect; otherwise, Calinski_Harabasz can be selected.

```
library(aricode)
ARI(result$cluster, result$celltype)
#> [1] 0.8486607

ggplot(result, aes(x = tSNE_1, y = tSNE_2, color = cluster)) +
  geom_point(size = 2.3, alpha = 0.6) +
  labs(title = "KKL clustering result") +
  scale_colour_rickandmorty() +
  theme_bw()
```



SessionInfo

```
sessionInfo()
#> R version 4.0.3 (2020-10-10)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 20.04 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
#> LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
#>
#> locale:
#>  [1] LC_CTYPE=zh_CN.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=zh_CN.UTF-8        LC_COLLATE=zh_CN.UTF-8
#>  [5] LC_MONETARY=zh_CN.UTF-8    LC_MESSAGES=zh_CN.UTF-8
#>  [7] LC_PAPER=zh_CN.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=zh_CN.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] parallel  stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] aricode_1.0.0       ggsci_2.9           ggplot2_3.3.3
#>  [4] SeuratObject_4.0.2  Seurat_4.0.4        Matrix_1.3-4
#>  [7] KKLClustering_0.2.0 stringr_1.4.0       igraph_1.2.6
#> [10] clusterCrit_1.2.8   dplyr_1.0.3         parallelDist_0.2.4
```

```
#> [13] sampling_2.9
#>
#> loaded via a namespace (and not attached):
#>   [1] Rtsne_0.15            colorspace_2.0-0      deldir_0.2-3
#>   [4] ellipsis_0.3.1        ggridges_0.5.3        spatstat.data_2.1-0
#>   [7] farver_2.0.3          leiden_0.3.6          listenv_0.8.0
#>  [10] ggrepel_0.9.0         codetools_0.2-18      splines_4.0.3
#>  [13] knitr_1.30            polyclip_1.10-0       jsonlite_1.7.2
#>  [16] ica_1.0-2             cluster_2.1.0         png_0.1-7
#>  [19] uwot_0.1.10           shiny_1.5.0           sctransform_0.3.2
#>  [22] spatstat.sparse_2.0-0 compiler_4.0.3        httr_1.4.2
#>  [25] assertthat_0.2.1      fastmap_1.0.1         lazyeval_0.2.2
#>  [28] later_1.1.0.1         htmltools_0.5.1.1     tools_4.0.3
#>  [31] gtable_0.3.0          glue_1.4.2            RANN_2.6.1
#>  [34] reshape2_1.4.4        Rcpp_1.0.7            scattermore_0.7
#>  [37] vctrs_0.3.6           nlme_3.1-151          lmtest_0.9-38
#>  [40] xfun_0.20             globals_0.14.0        mime_0.9
#>  [43] lpSolve_5.6.15        miniUI_0.1.1.1        lifecycle_0.2.0
#>  [46] irlba_2.3.3           goftest_1.2-2         future_1.21.0
#>  [49] MASS_7.3-53           zoo_1.8-8             scales_1.1.1
#>  [52] spatstat.core_2.3-0   promises_1.1.1        spatstat.utils_2.2-0
#>  [55] RColorBrewer_1.1-2    yaml_2.2.1            reticulate_1.18
#>  [58] pbapply_1.4-3         gridExtra_2.3         rpart_4.1-15
#>  [61] stringi_1.5.3         highr_0.8             rlang_0.4.10
#>  [64] pkgconfig_2.0.3       matrixStats_0.57.0    evaluate_0.14
#>  [67] lattice_0.20-41       ROCR_1.0-11           purrr_0.3.4
#>  [70] tensor_1.5            labeling_0.4.2        patchwork_1.1.1
#>  [73] htmlwidgets_1.5.3     cowplot_1.1.1         tidyselect_1.1.0
#>  [76] parallelly_1.23.0     RcppAnnoy_0.0.18      plyr_1.8.6
#>  [79] magrittr_2.0.1        R6_2.5.0              generics_0.1.0
#>  [82] DBI_1.1.0             withr_2.4.0           mgcv_1.8-33
#>  [85] pillar_1.4.7          fitdistrplus_1.1-3    survival_3.2-7
#>  [88] abind_1.4-5           tibble_3.0.5          future.apply_1.7.0
#>  [91] crayon_1.3.4          KernSmooth_2.23-18    spatstat.geom_2.2-2
#>  [94] plotly_4.9.3          rmarkdown_2.6         grid_4.0.3
#>  [97] data.table_1.13.6     digest_0.6.27         xtable_1.8-4
#> [100] tidyr_1.1.2           httpuv_1.5.4          RcppParallel_5.0.2
#> [103] munsell_0.5.0         viridisLite_0.3.0
```

References

Butler, Andrew, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. 2018. "Integrating Single-Cell Transcriptomic Data Across Different Conditions, Technologies, and Species." *Nature Biotechnology* 36 (5): 411–20.

Desgraupes, Bernard. 2013. "Clustering Indices." *University of Paris Ouest-Lab Modal'X* 1: 34.

Lake, Blue B, Simone Codeluppi, Yun C Yung, Derek Gao, Jerold Chun, Peter V Kharchenko, Sten Linnarsson, and Kun Zhang. 2017. "A Comparative Strategy for Single-Nucleus and Single-Cell Transcriptomes Confirms Accuracy in Predicted Cell-Type Expression from Nuclear Rna." *Scientific Reports* 7 (1): 1–8.

Satija, Rahul, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. 2015. "Spatial Reconstruction of Single-Cell Gene Expression Data." *Nature Biotechnology* 33 (5): 495–502.