RAPTR VDJ Barcode Matching

Set options and working directory

Load libraries

```
library(Seurat)
library(cowplot)
library(umap)
library(dplyr)
library(Matrix)
library(readxl)
library(openxlsx)
library(ggplot2)
library(ggrepel)
library(sctransform)
library(knitr)
library(kableExtra)
library(biomaRt)
library(DESeq2)
library(escape)
library(dittoSeq)
library(GSEABase)
library(scater)
library(patchwork)
library(data.table)
```

Import marker sets for later use - not applicable here yet

{r} #gene.lists <- read_xlsx("Markers_or_signatures.xlsx") #gene.lists.names <- colnames(gene.lists) #GOI.lists <- c() #for (i in gene.lists.nam

Import the gene expression data and re-name columns so that sample info is included

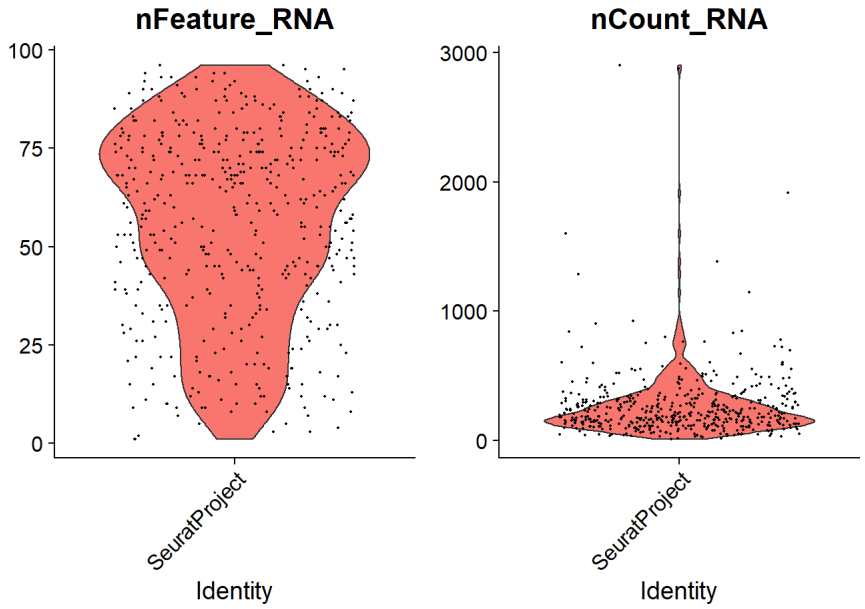
```
#aml.dat <- Read10X("filtered/aml.d5758/filtered_feature_bc_matrix/")
#fl.dat <- Read10X("filtered/fl.d5759/filtered_feature_bc_matrix/")

#colnames(aml.dat) <- paste(sapply(strsplit(colnames(aml.dat),split="-"),'[[',1L),"aml",sep="-")
#colnames(fl.dat) <- paste(sapply(strsplit(colnames(fl.dat),split="-"),'[[',1L),"fl",sep="-")
```

Import amplicon data and rename columns so cells match gene expression data

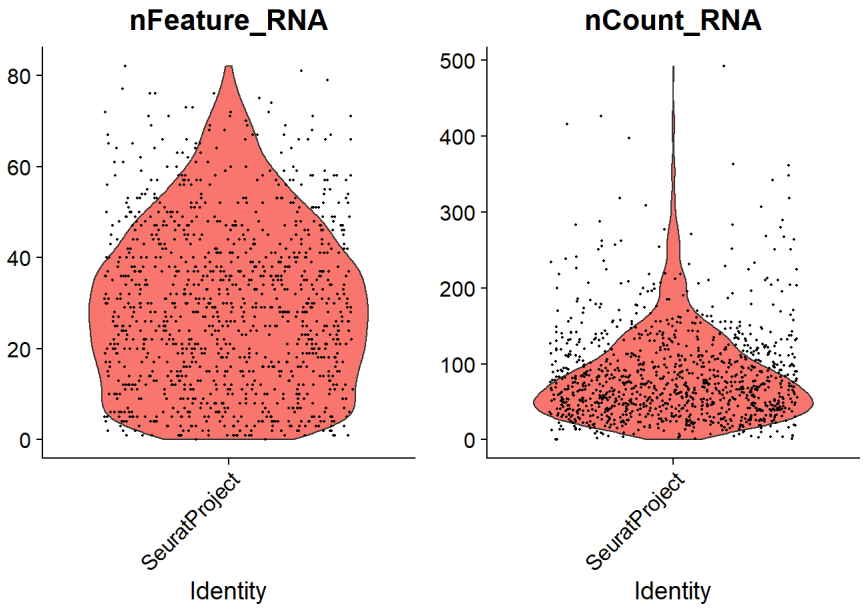
```
nCtRNAval <- 0

#d7201
d7201.umi.dat <- Read10X("../mCherry_5253T/d7201rc_cite/umi_count/",  gene.column=1)
d7201.umi <- CreateSeuratObject(counts = d7201.umi.dat)
VlnPlot(d7201.umi,features = c("nFeature_RNA", "nCount_RNA"),ncol = 2, pt.size = 0.3)
```



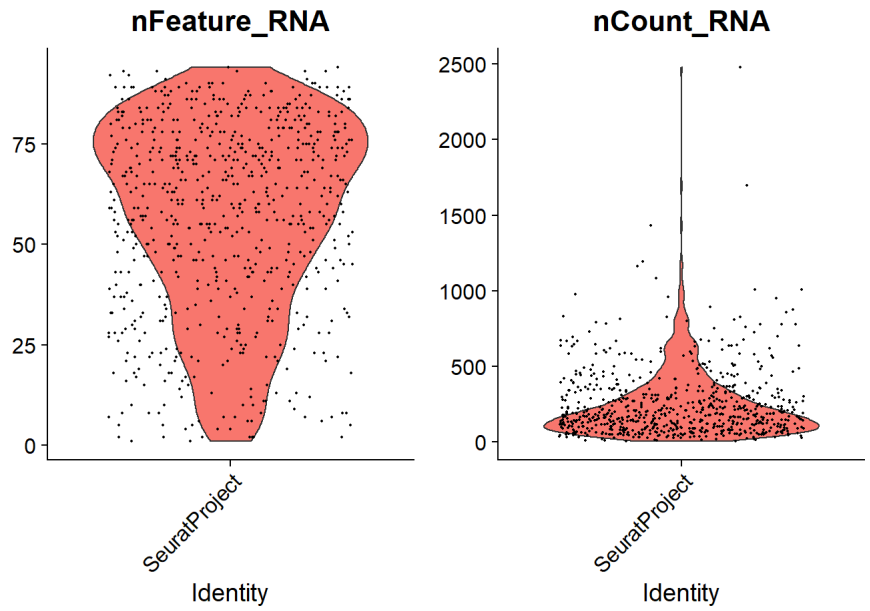

```
#d7201.umi.filt <- subset(d7201.umi, nFeature_RNA > 1 & nCount_RNA >= nCtRNAval)
d7201 <- as.data.frame(as.matrix(GetAssayData(object = d7201.umi, slot = "counts")))
d7201 <- tibble::rownames_to_column(d7201 ,"ID")
d7201 <- d7201 %>% mutate(total = rowSums(across(where(is.numeric))))
#write.xlsx(d7201, file ='d7201.umi_counts.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)

#d7202
d7202.umi.dat <- Read10X("../mCherry_5253T/d7202rc_cite/umi_count/",  gene.column=1)
d7202.umi <- CreateSeuratObject(counts = d7202.umi.dat)
VlnPlot(d7202.umi,features = c("nFeature_RNA", "nCount_RNA"),ncol = 2, pt.size = 0.3)
```



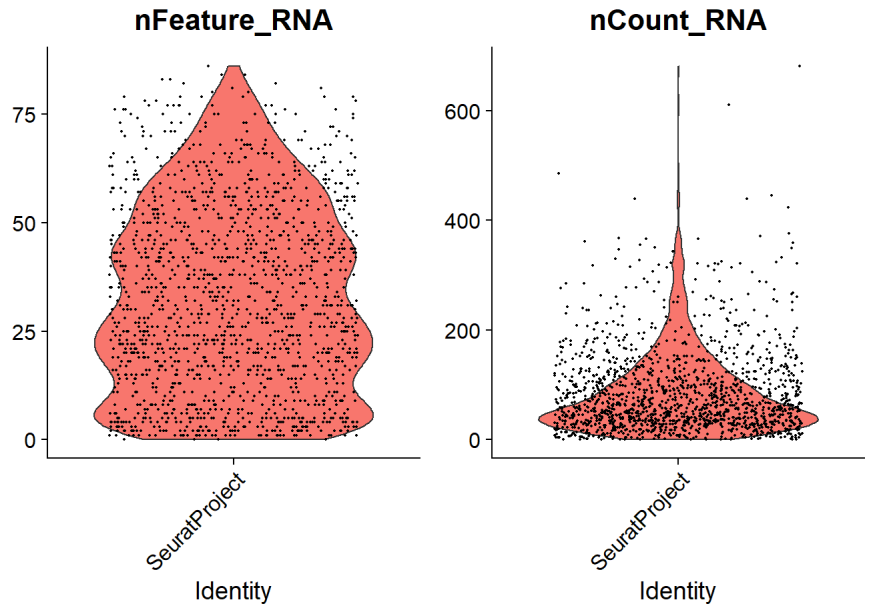

```
#d7202.umi.filt <- subset(d7202.umi, nFeature_RNA > 1 & nCount_RNA >= nCtRNAval)
d7202 <- as.data.frame(as.matrix(GetAssayData(object = d7202.umi, slot = "counts")))
d7202 <- tibble::rownames_to_column(d7202 ,"ID")
d7202 <- d7202 %>% mutate(total = rowSums(across(where(is.numeric))))
#write.xlsx(d7202, file ='d7202.umi_counts.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)

#d7203
d7203.umi.dat <- Read10X("../mCherry_5253T/d7203rc_cite/umi_count/",  gene.column=1)
d7203.umi <- CreateSeuratObject(counts = d7203.umi.dat)
VlnPlot(d7203.umi,features = c("nFeature_RNA", "nCount_RNA"),ncol = 2, pt.size = 0.3)
```



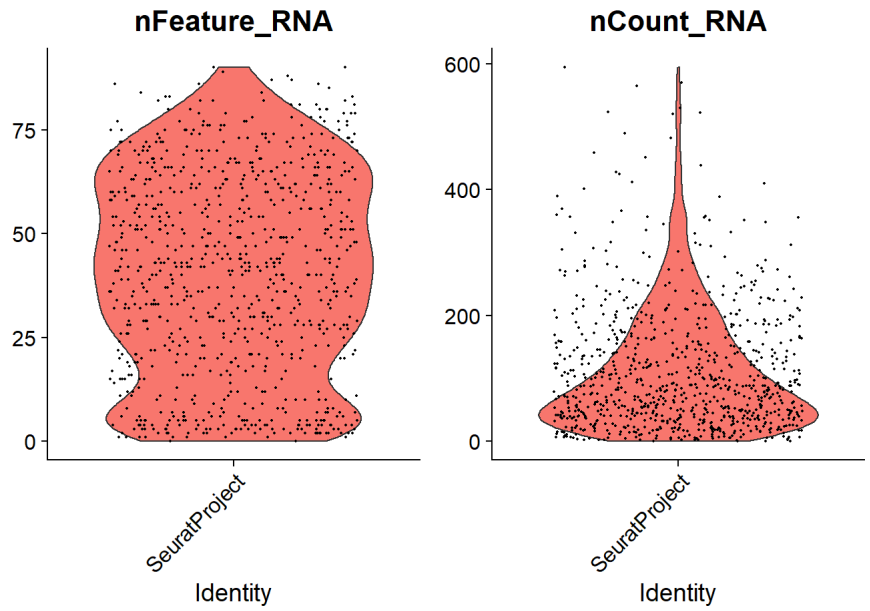

```
#d7203.umi.filt <- subset(d7203.umi, nFeature_RNA > 1 & nCount_RNA >= nCtRNAval)
d7203 <- as.data.frame(as.matrix(GetAssayData(object = d7203.umi, slot = "counts")))
d7203 <- tibble::rownames_to_column(d7203 ,"ID")
d7203 <- d7203 %>% mutate(total = rowSums(across(where(is.numeric))))
#write.xlsx(d7203, file ='d7203.umi_counts.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)

#d7204
d7204.umi.dat <- Read10X("../mCherry_5265T/d7204rc_cite/umi_count/",  gene.column=1)
d7204.umi <- CreateSeuratObject(counts = d7204.umi.dat)
VlnPlot(d7204.umi,features = c("nFeature_RNA", "nCount_RNA"),ncol = 2, pt.size = 0.3)
```



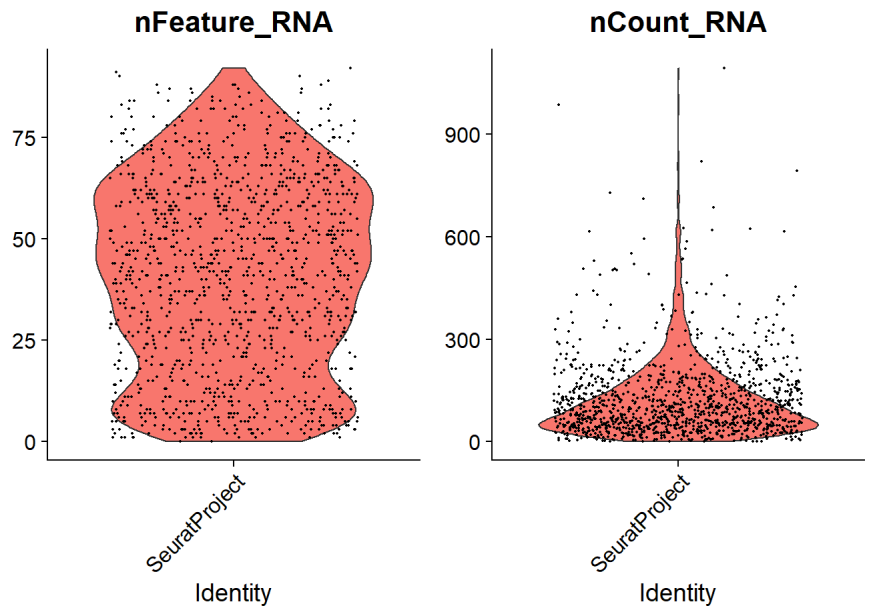

```
#d7204.umi.filt <- subset(d7204.umi, nFeature_RNA > 1 & nCount_RNA >= nCtRNAval)
d7204 <- as.data.frame(as.matrix(GetAssayData(object = d7204.umi, slot = "counts")))
d7204 <- tibble::rownames_to_column(d7204 ,"ID")
d7204 <- d7204 %>% mutate(total = rowSums(across(where(is.numeric))))
#write.xlsx(d7204, file ='d7204.umi_counts.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)

#d7205
d7205.umi.dat <- Read10X("../mCherry_5265T/d7205rc_cite/umi_count/",  gene.column=1)
d7205.umi <- CreateSeuratObject(counts = d7205.umi.dat)
VlnPlot(d7205.umi,features = c("nFeature_RNA", "nCount_RNA"),ncol = 2, pt.size = 0.3)
```




```
#d7205.umi.filt <- subset(d7205.umi, nFeature_RNA > 1 & nCount_RNA >= nCtRNAval)
d7205 <- as.data.frame(as.matrix(GetAssayData(object = d7205.umi, slot = "counts")))
d7205 <- tibble::rownames_to_column(d7205 ,"ID")
d7205 <- d7205 %>% mutate(total = rowSums(across(where(is.numeric))))
#write.xlsx(d7205, file ='d7205.umi_counts.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)

#d7206
d7206.umi.dat <- Read10X("../mCherry_5265T/d7206rc_cite/umi_count/",  gene.column=1)
d7206.umi <- CreateSeuratObject(counts = d7206.umi.dat)
VlnPlot(d7206.umi,features = c("nFeature_RNA", "nCount_RNA"),ncol = 2, pt.size = 0.3)
```




```
#d7206.umi.filt <- subset(d7206.umi, nFeature_RNA > 1 & nCount_RNA >= nCtRNAval)
d7206 <- as.data.frame(as.matrix(GetAssayData(object = d7206.umi, slot = "counts")))
d7206 <- tibble::rownames_to_column(d7206 ,"ID")
d7206 <- d7206 %>% mutate(total = rowSums(across(where(is.numeric))))
#write.xlsx(d7206, file ='d7206.umi_counts.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)
```

Function to add vdj metadata to a seurat object

This was downloaded from

https://github.com/satijalab/seurat/issues/1435 (https://github.com/satijalab/seurat/issues/1435)

```
add_clonotype <- function(tcr_location, seurat_obj){
    tcr <- read.csv(paste(tcr_location,"filtered_contig_annotations.csv", sep=""))

    # Remove the -1 at the end of each barcode.
    # Subsets so only the first line of each barcode is kept,
    # as each entry for given barcode will have same clonotype.
    tcr$barcode <- gsub("-1", "", tcr$barcode)
    tcr <- tcr[!duplicated(tcr$barcode), ]

    # Only keep the barcode and clonotype columns.
    # We'll get additional clonotype info from the clonotype table.
    tcr <- tcr[,c("barcode", "raw_clonotype_id")]
    names(tcr)[names(tcr) == "raw_clonotype_id"] <- "clonotype_id"

    # Clonotype-centric info.
    clono <- read.csv(paste(tcr_location,"clonotypes.csv", sep=""))

    # Slap the AA sequences onto our original table by clonotype_id.
    tcr <- merge(tcr, clono[, c("clonotype_id", "cdr3s_aa")])

    # Reorder so barcodes are first column and set them as rownames.
    tcr <- tcr[, c(2,1,3)]
    rownames(tcr) <- tcr[,1]
    tcr[,1] <- NULL

    # Add to the Seurat object's metadata.
    clono_seurat <- AddMetaData(object=seurat_obj, metadata=tcr)
    return(clono_seurat)
}
```

Add vdj metadata to citeseq seurat objects

```
d7201.umi.vdj <- add_clonotype("../d7201_vdj/outs/",d7201.umi)
d7202.umi.vdj <- add_clonotype("../d7202_vdj/outs/",d7202.umi)
d7203.umi.vdj <- add_clonotype("../d7203_vdj/outs/",d7203.umi)
d7204.umi.vdj <- add_clonotype("../d7204_vdj/outs/",d7204.umi)
d7205.umi.vdj <- add_clonotype("../d7205_vdj/outs/",d7205.umi)
d7206.umi.vdj <- add_clonotype("../d7206_vdj/outs/",d7206.umi)
```

join cite counts and vdj meta

```
td7201 <- as.data.frame(t(as.matrix(GetAssayData(object = d7201.umi, slot = "counts"))))
td7201 <- tibble::rownames_to_column(td7201,"ID")

vdj7201 <- as.data.frame(as.data.frame(d7201.umi.vdj@meta.data))

vdj7201 <- tibble::rownames_to_column(vdj7201 ,"ID")
vdj7201.export <- plyr::join(vdj7201, td7201, by="ID", type="full")
write.xlsx(vdj7201.export, file ='d7201.export.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)
###########
td7202 <- as.data.frame(t(as.matrix(GetAssayData(object = d7202.umi, slot = "counts"))))
td7202 <- tibble::rownames_to_column(td7202,"ID")

vdj7202 <- as.data.frame(as.data.frame(d7202.umi.vdj@meta.data))

vdj7202 <- tibble::rownames_to_column(vdj7202 ,"ID")
vdj7202.export <- plyr::join(vdj7202, td7202, by="ID", type="full")
write.xlsx(vdj7202.export, file ='d7202.export.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)
############
td7203 <- as.data.frame(t(as.matrix(GetAssayData(object = d7203.umi, slot = "counts"))))
td7203 <- tibble::rownames_to_column(td7203,"ID")

vdj7203 <- as.data.frame(as.data.frame(d7203.umi.vdj@meta.data))

vdj7203 <- tibble::rownames_to_column(vdj7203 ,"ID")
vdj7203.export <- plyr::join(vdj7203, td7203, by="ID", type="full")
write.xlsx(vdj7203.export, file ='d7203.export.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)
############
td7204 <- as.data.frame(t(as.matrix(GetAssayData(object = d7204.umi, slot = "counts"))))
td7204 <- tibble::rownames_to_column(td7204,"ID")

vdj7204 <- as.data.frame(as.data.frame(d7204.umi.vdj@meta.data))

vdj7204 <- tibble::rownames_to_column(vdj7204 ,"ID")
vdj7204.export <- plyr::join(vdj7204, td7204, by="ID", type="full")
write.xlsx(vdj7204.export, file ='d7204.export.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)
############
td7205 <- as.data.frame(t(as.matrix(GetAssayData(object = d7205.umi, slot = "counts"))))
td7205 <- tibble::rownames_to_column(td7205,"ID")

vdj7205 <- as.data.frame(as.data.frame(d7205.umi.vdj@meta.data))

vdj7205 <- tibble::rownames_to_column(vdj7205 ,"ID")
vdj7205.export <- plyr::join(vdj7205, td7205, by="ID", type="full")
write.xlsx(vdj7205.export, file ='d7205.export.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)
############
td7206 <- as.data.frame(t(as.matrix(GetAssayData(object = d7206.umi, slot = "counts"))))
td7206 <- tibble::rownames_to_column(td7206,"ID")

vdj7206 <- as.data.frame(as.data.frame(d7206.umi.vdj@meta.data))

vdj7206 <- tibble::rownames_to_column(vdj7206 ,"ID")
vdj7206.export <- plyr::join(vdj7206, td7206, by="ID", type="full")
write.xlsx(vdj7206.export, file ='d7206.export.xlsx', rowNames = FALSE, colNames = TRUE, overwrite = TRUE)
```

write session info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] data.table_1.14.0          patchwork_1.1.1
##  [3] scater_1.20.1              scuttle_1.2.0
##  [5] SingleCellExperiment_1.14.1 GSEABase_1.54.0
##  [7] graph_1.70.0               annotate_1.70.0
##  [9] XML_3.99-0.6               AnnotationDbi_1.54.1
## [11] dittoSeq_1.4.1             escape_1.2.0
## [13] DESeq2_1.32.0              SummarizedExperiment_1.22.0
## [15] Biobase_2.52.0             MatrixGenerics_1.4.0
## [17] matrixStats_0.59.0         GenomicRanges_1.44.0
## [19] GenomeInfoDb_1.28.0        IRanges_2.26.0
## [21] S4Vectors_0.30.0           BiocGenerics_0.38.0
## [23] biomaRt_2.48.1             kableExtra_1.3.4
## [25] knitr_1.33                 sctransform_0.3.2
## [27] ggrepel_0.9.1              ggplot2_3.3.5
## [29] openxlsx_4.2.4             readxl_1.3.1
## [31] Matrix_1.3-3               dplyr_1.0.7
## [33] umap_0.2.7.0               cowplot_1.1.1
## [35] SeuratObject_4.0.2         Seurat_4.0.3
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.1                reticulate_1.20
##   [3] tidyselect_1.1.1          RSQLite_2.2.7
##   [5] htmlwidgets_1.5.3         grid_4.1.0
##   [7] BiocParallel_1.26.0       Rtsne_0.15
##   [9] ScaledMatrix_1.0.0        munsell_0.5.0
##  [11] codetools_0.2-18          ica_1.0-2
##  [13] future_1.21.0             miniUI_0.1.1.1
##  [15] withr_2.4.2               colorspace_2.0-1
##  [17] filelock_1.0.2            highr_0.9
##  [19] rstudioapi_0.13           ROCR_1.0-11
##  [21] tensor_1.5                listenv_0.8.0
##  [23] labeling_0.4.2            GenomeInfoDbData_1.2.6
##  [25] polyclip_1.10-0           farver_2.1.0
##  [27] pheatmap_1.0.12           bit64_4.0.5
##  [29] rhdf5_2.36.0              parallelly_1.26.1
##  [31] vctrs_0.3.8               generics_0.1.0
##  [33] xfun_0.23                 BiocFileCache_2.0.0
##  [35] R6_2.5.0                  ggbeeswarm_0.6.0
##  [37] rsvd_1.0.5                msigdbr_7.4.1
##  [39] locfit_1.5-9.4            rhdf5filters_1.4.0
##  [41] bitops_1.0-7              spatstat.utils_2.2-0
##  [43] cachem_1.0.5              DelayedArray_0.18.0
##  [45] assertthat_0.2.1          promises_1.2.0.1
##  [47] scales_1.1.1              beeswarm_0.4.0
##  [49] gtable_0.3.0              beachmat_2.8.0
##  [51] globals_0.14.0            goftest_1.2-2
##  [53] rlang_0.4.11              genefilter_1.74.0
##  [55] systemfonts_1.0.2         splines_4.1.0
##  [57] lazyeval_0.2.2            spatstat.geom_2.2-0
##  [59] yaml_2.2.1                reshape2_1.4.4
##  [61] abind_1.4-5               httpuv_1.6.1
##  [63] tools_4.1.0               ellipsis_0.3.2
##  [65] spatstat.core_2.2-0       jquerylib_0.1.4
##  [67] RColorBrewer_1.1-2        ggridges_0.5.3
##  [69] Rcpp_1.0.6                plyr_1.8.6
##  [71] sparseMatrixStats_1.4.0   progress_1.2.2
##  [73] zlibbioc_1.38.0           purrr_0.3.4
##  [75] RCurl_1.98-1.3            prettyunits_1.1.1
##  [77] rpart_4.1-15              openssl_1.4.4
##  [79] deldir_0.2-10             viridis_0.6.1
##  [81] pbapply_1.4-3             zoo_1.8-9
##  [83] cluster_2.1.2             magrittr_2.0.1
##  [85] RSpectra_0.16-0           scattermore_0.7
```

```
##  [87] lmtest_0.9-38             RANN_2.6.1
##  [89] fitdistrplus_1.1-5        GSVA_1.40.1
##  [91] hms_1.1.0                 mime_0.11
##  [93] evaluate_0.14             xtable_1.8-4
##  [95] gridExtra_2.3             compiler_4.1.0
##  [97] tibble_3.1.2              KernSmooth_2.23-20
##  [99] crayon_1.4.1              htmltools_0.5.1.1
## [101] mgcv_1.8-36               later_1.2.0
## [103] tidyr_1.1.3               geneplotter_1.70.0
## [105] DBI_1.1.1                 dbplyr_2.1.1
## [107] MASS_7.3-54               rappdirs_0.3.3
## [109] babelgene_21.4            igraph_1.2.6
## [111] pkgconfig_2.0.3           plotly_4.9.4.1
## [113] spatstat.sparse_2.0-0     xml2_1.3.2
## [115] svglite_2.0.0             vipor_0.4.5
## [117] bslib_0.2.5.1             webshot_0.5.2
## [119] XVector_0.32.0            rvest_1.0.0
## [121] stringr_1.4.0             digest_0.6.27
## [123] RcppAnnoy_0.0.18          spatstat.data_2.1-0
## [125] Biostrings_2.60.1         rmarkdown_2.9
## [127] cellranger_1.1.0          leiden_0.3.8
## [129] uwot_0.1.10               DelayedMatrixStats_1.14.0
## [131] curl_4.3.1                shiny_1.6.0
## [133] lifecycle_1.0.0           nlme_3.1-152
## [135] jsonlite_1.7.2            Rhdf5lib_1.14.1
## [137] BiocNeighbors_1.10.0      limma_3.48.1
## [139] viridisLite_0.4.0         askpass_1.1
## [141] fansi_0.5.0               pillar_1.6.1
## [143] lattice_0.20-44           KEGGREST_1.32.0
## [145] fastmap_1.1.0             httr_1.4.2
## [147] survival_3.2-11           glue_1.4.2
## [149] zip_2.2.0                 png_0.1-7
## [151] bit_4.0.4                 HDF5Array_1.20.0
## [153] stringi_1.6.2             sass_0.4.0
## [155] blob_1.2.1                BiocSingular_1.8.1
## [157] memoise_2.0.0             irlba_2.3.3
## [159] future.apply_1.7.0
```

```
writeLines(capture.output(sessionInfo()), "210727Bir_sessionInfo.txt")
```