mitoClone2: Supplementary Tutorial

Benjamin Story, Ahrmad Annan

12 June 2024

Contents

References

Corces, M., Buenrostro, J., Wu, B. *et al.* Lineage-specific and single-cell chromatin accessibility charts human hematopoiesis and leukemia evolution. *Nat Genet* **48**, 1193–1203 (2016). https://doi.org/10.1038/ng.3646. PMID: 27526324

1 Demonstration of mitochondrial variant discovery from RNA-sequencing data

As a proof of concept, we demonstrate the ability of the mitoClone2 package to extract *bona fide* private mutations from paired bulk ATAC-seq and RNA-seq samples. The goal here was to verify that variants in the mtDNA, which are specific to individual populations of cells, can be readily detected using RNA-seq data alone. We downloaded public bulk ATAC-seq and RNA-seq experimental data which were performed in parallel on cancer patients.

The data used for this analysis, from Corces *et al.* (cited in **References**), can be downloaded from the NCBI Gene Expression Omnibus (GEO) which is accessible at: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE74912

The RNA-seq reads were aligned to the *Homo sapiens* hg38 genome (Ensembl release 95) using `STAR v2.7.2b` with default parameters. The ATAC-seq reads were trimmed and aligned to the *Homo sapiens* hg38 genome using `bowtie2 v2.3.0` with the following parameters: `-X 2000`.

Count tables for each sample were tabulated using the `baseCountsFromBamList` a function included in the mitoClone2 package. The sample name for each putative patient was inferred from the GEO metadata. Only the following patients were included: `6792, 1022, 2596, 5483, 6926, 7256, SU048, SU070, SU209, SU351, SU353, SU444, SU484, SU496, SU501, SU575, SU583, and SU654` as these patients had both DNA (i.e. ATAC-seq) and RNA data available.

We note here that although this is bulk data, we can still treat each sample as if it were a single cell.

We note at this point that there are two possible filtering methods that can be used for downstream analysis.

1.0.1 *Cohort* Method:

- Preferred for controlling wet-lab variability by using multiple independent single-cell datasets, generated using the same sequencing method, to exclude shared variants and identify patient/time-point specific variants.
- The excluded variants could be biological (common polymorphisms) or technical (artifacts from specific reagents or bioinformatics tools).
- This method helps filter out both technical and biological noise, by determining what signal can be attributed to background noise.

1.0.2 *Exclusionlist* Method:

- Utilizes established lists of known artifactual or challenging genomic regions, encompassing three sets:
  - Known RNA-editing sites in mitochondria from REDIportal (PMID: 27587585)
  - Regions flanking homopolymer stretches of 3 or more nucleotides
  - Shared variants specific to a given sequencing method
- The latter set of shared variants, included in the package, is derived from datasets generated using a modified Smart-seq2.
- This method is ideal for users who lack access to multiple datasets or require a fast, predefined filtering option.

The following sections provide a detailed walkthrough of both filtering methods, starting with the *Cohort* method.

2 Complete workflow

2.1 Load packages

```
library(mitoClone2)
library(reshape2)
library(viridis)
library(ComplexHeatmap)
library(ggplot2)
```

2.2 Prepare some helper functions

A variety of extra functions are provided to minimize code redundancy.

```
#-- Function that converts variant string format from 'X13370.G' to '1337 0>G'
fixvarName <- function(x) {
    gsub("\\.", ">", gsub("^X", "", x))
}

#-- Function that takes a dataset and name and gets mean allele freq. per patient
getptMean <- function(Data, name) {
    df <- data.frame(t(Data))
    df.l <- lapply(split(df, gsub("\\-.*", "", row.names(df))), colMeans)
    df.l <- melt(do.call(cbind, df.l))
    colnames(df.l) <- c("var", "Patient", name)
    return(df.l)
}
```

This vignette uses data that is large (~30 MB) and thus not included with the package, but it can be downloaded separately via GitHub at: https://github.com/benstory/mitoClone2_supplemental.

The **mitoClone2** package provides some sample datasets. However, to emphasize the ability of mitoClone2 to detect real mitochondrial variants in RNA-seq data alone, we opted to use an additional more appropriate dataset where orthogonal variant confirmation via ATAC-seq was possible. These allele count tables, imported below, were generated by the `baseCountsFromBamList` function - for ATAC-seq the `bam2R` quality threshold was set to 30 instead of the default.

```
#-- Here is an example of how we would generate our allele count files locally - SKIPPED
## vars.rna <- baseCountsFromBamList(bamfiles =
## list.files('corces_RNA','\\.bam$',full = TRUE), sites =
## 'chrM:1-15000') the files need to be located in R's working
## directory!
#-- Download the external data from the mitoClone2 Supplemental GitHub Page
SuppGitURL <- "https://github.com/benstory/mitoClone2_supplemental/raw/main/"
## system(paste0('wget -nv ',
## SuppGitURL,'corces_nt_counts_per_position_rna.RDS'))
## system(paste0('wget -nv ',
## SuppGitURL,'corces_nt_counts_per_position_quality30_atac.RDS'))
#-- Loading the external data from GSE74246 (rna)
vars.rna <- readRDS("corces_nt_counts_per_position_rna.RDS")
#-- Loading the external data from GSE74912 (atac)
vars.atac <- readRDS("corces_nt_counts_per_position_quality30_atac.RDS")
```

We run mutationCallsFromCohort on the RNA-seq allele count tables for each of these patient samples using default parameters excepting for `MINFRAC.OTHER=0.2, MINREADS=500, MINCELL=1, MINCELLS.PATIENT=1`.These parameters are modified from the default values because we are not dealing with single-cell data. The `MINCELL` parameter now refers to the total number of samples from each individual patient. Given that there are two or more samples per patient, we require that the variants be present in at least a single sample.

The resulting 153 variants from this command were further filtered to remove 8 universal variants with an allele frequency over 5% in more than 6 RNA-seq samples (i.e. non-patient exclusive mutations) leaving 145 putative variants of interest.

2.3 Call variants from the RNA-seq data

```
vars.rna.call <- mutationCallsFromCohort(vars.rna, MINFRAC = 0.1, MINFRAC.PATIENT = 0.01,
    MINFRAC.OTHER = 0.2, MINREADS = 500, MINCELL = 1, MINCELLS.PATIENT = 1,
    patient = gsub("\\-.*", "", names(vars.rna)), genome = "hg38", sites = "chrM:1-16569",
    USE.REFERENCE = TRUE)
```

2.4 Examine the excluded universally shared variants

```
#-- We extract the list of excluded sites and save for downstream analysis
vars.exclusions <- vars.rna.call[["exclusionlist"]]
head(vars.exclusions)
```

```
## [1] "73 A>G"  "146 T>C" "152 T>C" "263 A>G" "646 C>T" "709 G>A"
```

```
vars.rna.call[["exclusionlist"]] <- NULL
```

2.5 Extract putative private variants

```
#-- Pull all the identified mutations that are likely exclusive to patients
candidates <- unlist(lapply(vars.rna.call, function(x) names(x@cluster)))

#-- Remove variants that appear in clusters separated by less than 3bp for each patient
candidates <- removeWindow(fixvarName(sort(unique(candidates))), window = 3)

#-- Extract counts of putative candidates from the RNA-seq
rna.counts <- pullcountsVars(vars.rna, candidates)
plotData <- rna.counts$M/(rna.counts$M + rna.counts$N)

#-- Remove variants that seem to be universal (i.e. non-patient exclusive)
#-- Our cutoff is an allele frequency over 5% in more than 6 samples
plotData <- plotData[rowSums(plotData > 0.05) <= 6, ]
print(NROW(plotData))
```

```
## [1] 145
```

2.6 Prepare metadata for the downstream plotting

Extract patient identifiers and set colors.

```r
#-- Extract patient IDs
pts <- gsub("\\-.*", "", colnames(plotData))
#-- Set unique colors for each patient
pts.colors <- c("#FF7F00", "brown", "palegreen2", "khaki2", "#FB9A99",
    "darkorange4", "green1", "gold1", "steelblue4", "dodgerblue2", "darkturquoise",
    "orchid1", "black", "#6A3D9A", "blue1", "#CAB2D6", "gray70", "yellow3")
names(pts.colors) <- levels(factor(pts))
```

Prepare metadata for samples and heatmaps.

```r
#-- Preparing a metadata data.frame that includes:
#-- the sample name
#-- the patient associated to the sample
#-- the technology used to generate the sample
meta.data <- melt(c(names(vars.rna), names(vars.atac)))
row.names(meta.data) <- meta.data$value
meta.data$Patient <- gsub("\\-.*", "", meta.data$value)
meta.data$Method <- paste0(gsub(".*\\_", "", meta.data$value), "-seq")
meta.data$value <- NULL
```

2.7 Investigate allele frequencies of putative private variants across sequencing technologies

We then extracted these same variants from both the ATAC-seq and RNA-seq (above) samples using the `pullcountsVars` function.

Prepare a heatmap of the RNA-seq data.

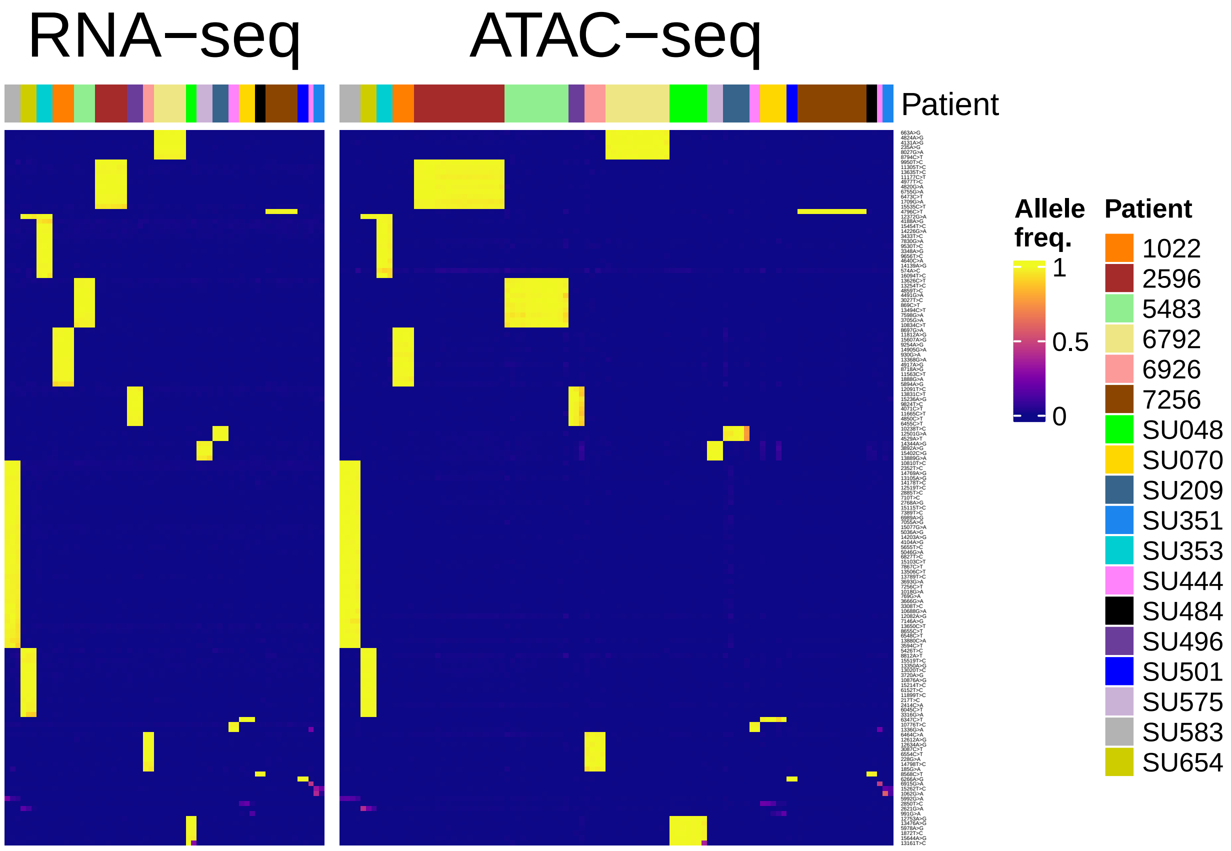
```r
#-- Setup heatmap annotation - plotData
ha_column = HeatmapAnnotation(df = meta.data[colnames(plotData), "Patient",
    drop = FALSE], col = list(Patient = pts.colors), show_annotation_name = FALSE)
#-- Build a heatmap of the RNA-seq allele frequencies for each variant
hrna <- Heatmap(plotData, col = viridis(100, option = "plasma"), show_row_names = TRUE,
    show_column_names = FALSE, top_annotation = ha_column, column_title = "RNA-seq",
    show_column_dend = FALSE, show_row_dend = FALSE, column_title_gp = gpar(fontsize = 24),
    row_names_gp = gpar(fontsize = 2), name = "Allele\nfreq.")
```

Prepare a heatmap of the ATAC-seq data.

```r
#-- Extract counts of putative candidates from the ATAC-seq
atac.counts <- pullcountsVars(vars.atac, row.names(plotData))
plotData.atac <- atac.counts$M/(atac.counts$M + atac.counts$N)

#-- Setup heatmap annotation for plotData.atac
ha_column = HeatmapAnnotation(df = meta.data[colnames(plotData.atac), "Patient",
    drop = FALSE], col = list(Patient = pts.colors))
#-- Build a heatmap of the ATAC-seq allele frequencies for each variant
hatac <- Heatmap(plotData.atac, col = viridis(100, option = "plasma"),
    show_row_names = TRUE, show_column_names = FALSE, top_annotation = ha_column,
    column_title = "ATAC-seq", show_column_dend = FALSE, show_row_dend = FALSE,
    column_title_gp = gpar(fontsize = 24), row_names_gp = gpar(fontsize = 2),
    name = "Allele\nfreq.")
```

The variants identified seem to allow clear differentiation of the individual patients across technologies.

```
#-- Plot heatmaps for both technologies side-by-side - S3
hrna + hatac
```



Supplementary Figure S3. Allele frequencies of mitochondrial variants identified as patient-specific from RNA-seq.

2.8 Verifying consistency between ATAC-seq and RNA-seq for putative private variants

Extract the mean allele frequencies per patient for each variant.

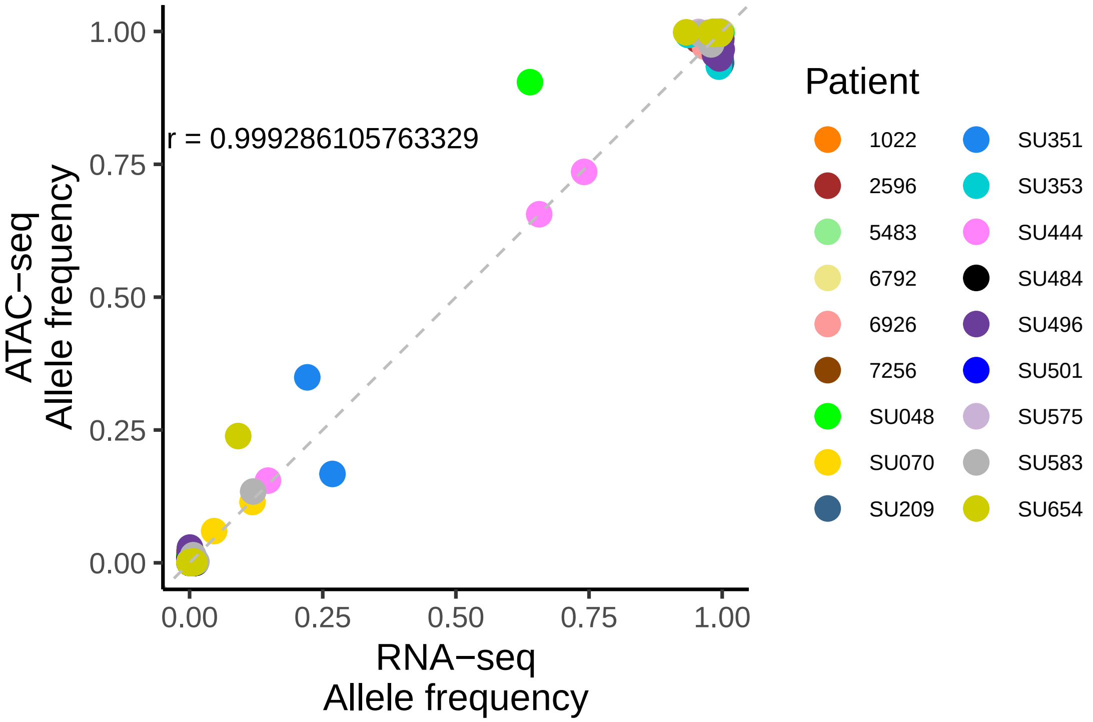
```
#-- Extract mean allele freq per patient per technology and merge
df.rna <- getptMean(plotData, "RNA-seq")
df.atac <- getptMean(plotData.atac, "ATAC-seq")

#-- Sanity check all data.frames should be similar
all(c(identical(df.rna$Patient, df.atac$Patient)), identical(df.rna$var,
    df.atac$var))
```

```
## [1] TRUE
```

```
#-- merge into a new data.frame
df.both <- df.rna
df.both$"ATAC-seq" <- df.atac$"ATAC-seq"

#-- Perform a correlation test on the samples with Pearson
cor_both <- cor.test(df.both$ATAC, df.both$RNA)
colnames(df.rna)[2] <- "Patient"
```

Plot the correlation between the RNA-seq and the ATAC-seq.

```
#-- Plot the correlation and allele frequency comparison - S4
ggplot(df.both, aes(x = `RNA-seq`, y = `ATAC-seq`, color = Patient)) +
    geom_point(size = 4) + theme_classic(base_size = 14) + geom_abline(slope = 1,
    color = "grey", linetype = "dashed") + scale_colour_manual(values = pts.colors) +
    annotate("text", x = 0.25, y = 0.8, label = paste0("r = ", cor_both$estimate)) +
    ylab("ATAC-seq\nAllele frequency") + xlab("RNA-seq\nAllele frequency") +
    theme(plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.text = element_text(size = 8)) +
    guides(color = guide_legend(ncol = 2))
```



Supplementary Figure S4. Allele frequencies of RNA-seq compared to ATAC-seq with Pearson correlation coefficient.

Merge your two large allele frequency data.frames.

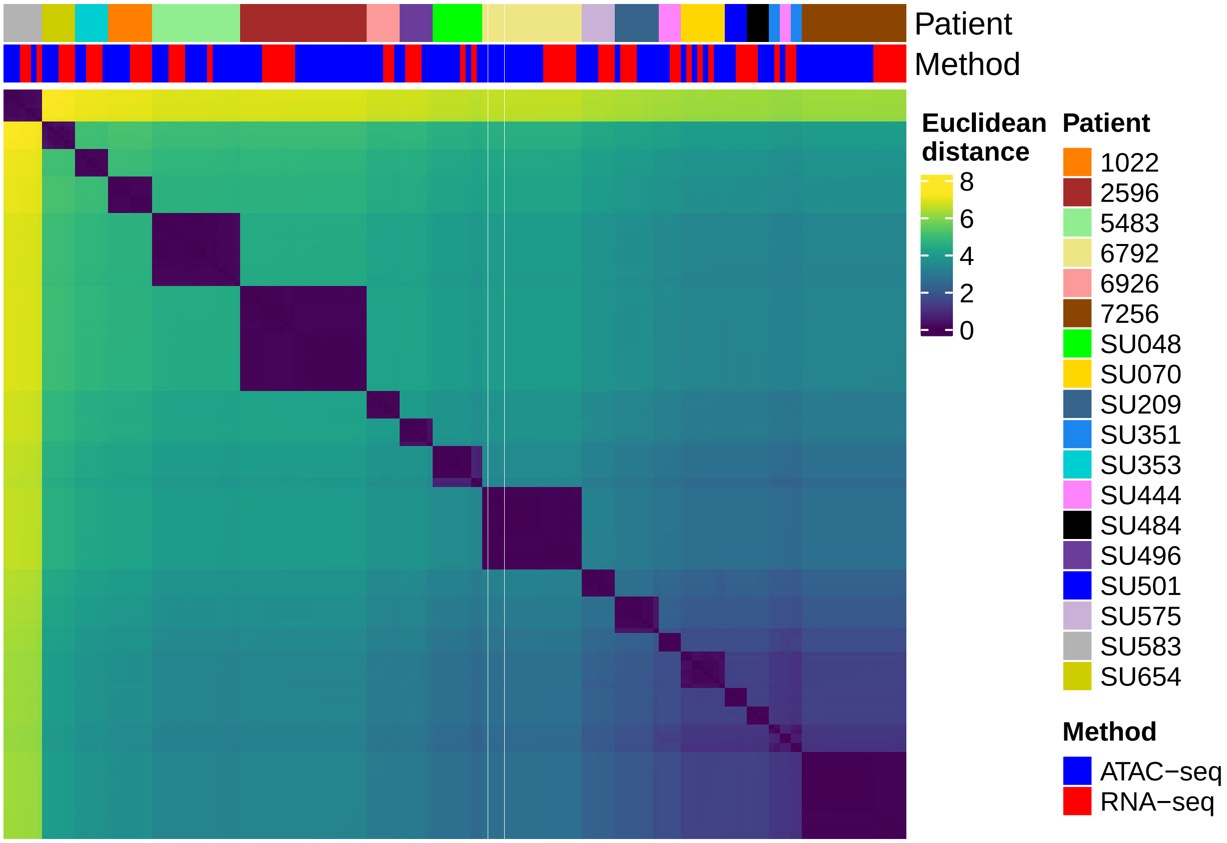
```
#-- Merge the RNA-seq and ATAC-seq allele frequency data.frames
merged.vars <- merge(plotData.atac, plotData, by = "row.names", all = T)
row.names(merged.vars) <- merged.vars$Row.names
merged.vars$Row.names <- NULL
```

Calculate the Euclidean distance between samples and prepare metadata.

```
#-- Create distance matrix from merged method data.frame
#-- The default method is set to 'euclidean'
distmat <- data.frame(as.matrix(dist(t(merged.vars))))
colnames(distmat) <- row.names(distmat)
distance.colors <- list(Method = c(`RNA-seq` = "red", `ATAC-seq` = "blue"),
    Patient = pts.colors)
ha_column <- HeatmapAnnotation(df = meta.data[colnames(distmat), ], col = distance.colors)
```

Plot the distance-matrix heatmap.

```
#-- S5
Heatmap(distmat, col = viridis(100), show_row_names = FALSE, show_column_names = FALSE,
    top_annotation = ha_column, show_column_dend = FALSE, show_row_dend = FALSE,
    name = "Euclidean\ndistance")
```



Supplementary Figure S5. Clustered distance matrix of combined variant allele frequencies from ATAC-seq and RNA-seq.

2.9 Identify variants resulting from cancer evolution

Even in this bulk sequencing experiment, we see evidence of variants that differ between the cancerous and pre-cancerous states.

Let's home in on an example, patient SU444, and extract their unique variants.

```
su444 <- fixvarName(names(vars.rna.call[["SU444"]]@cluster))
print(su444)
```

```
## [1] "1336G>A"  "6915G>A"  "10776T>C"
```

Pull the corresponding variant allele frequencies from both the ATAC-seq and RNA-seq datasets and prepare to plot.

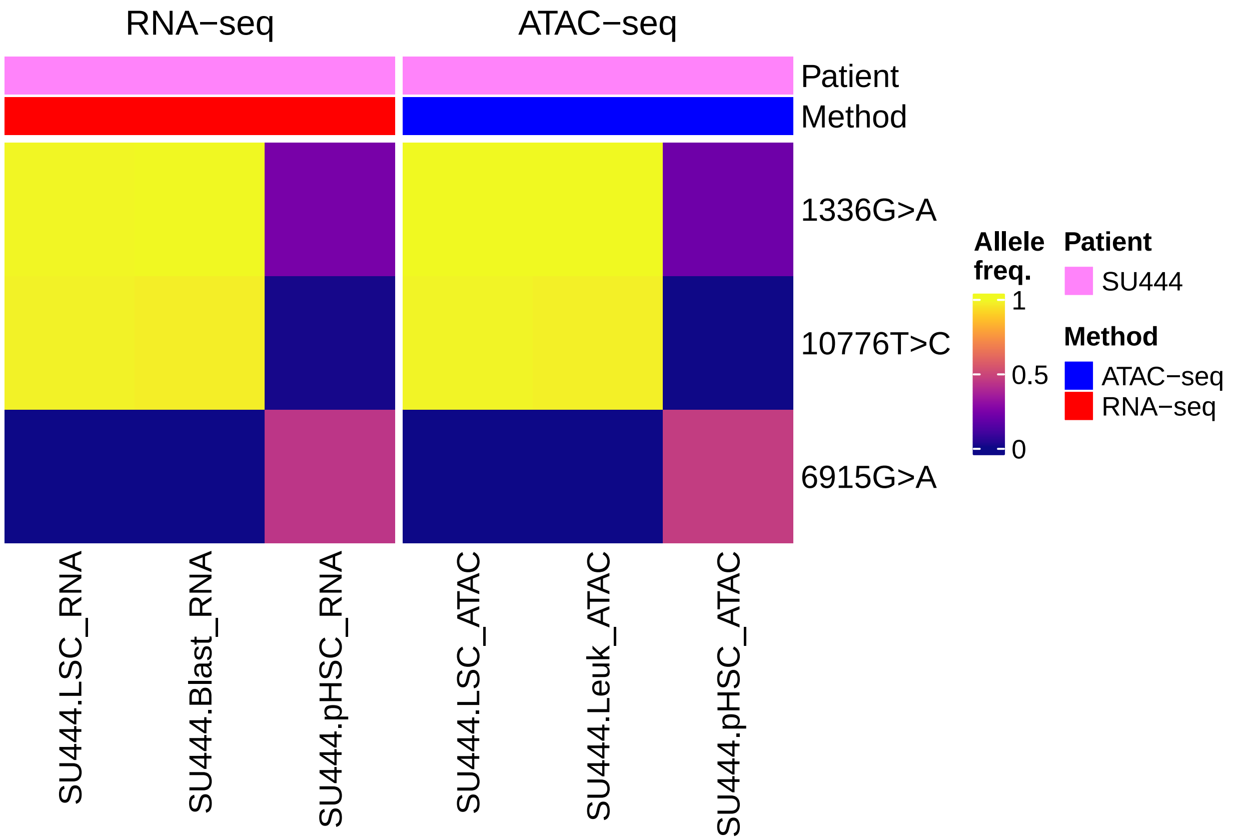
```
#-- Extract raw AFs from the datasets
su444rna <- data.frame(plotData[su444, grep("SU444", colnames(plotData))])
su444atac <- data.frame(plotData.atac[su444, grep("SU444", colnames(plotData.atac))])

#-- Merge into a single data.frame and print
su444 <- cbind(su444rna, su444atac)
print(su444)
```

```
##          SU444.Blast_RNA SU444.LSC_RNA SU444.pHSC_RNA SU444.Leuk_ATAC
## 1336G>A     0.9960236461  0.9920427389     0.23440704    0.9984565830
## 6915G>A     0.0003128715  0.0003790751     0.44083000    0.0001405185
## 10776T>C    0.9745313053  0.9832127352     0.01153846    0.9782385060
##          SU444.LSC_ATAC SU444.pHSC_ATAC
## 1336G>A       0.9987066     0.209903122
## 6915G>A       0.0000000     0.464139344
## 10776T>C      0.9876380     0.001962709
```

```
su444_ann <- HeatmapAnnotation(df = subset(meta.data, Patient == "SU444"),
    col = list(Method = c(`RNA-seq` = "red", `ATAC-seq` = "blue"), Patient = pts.colors))
```

Plot the variant allele frequencies for this patient.

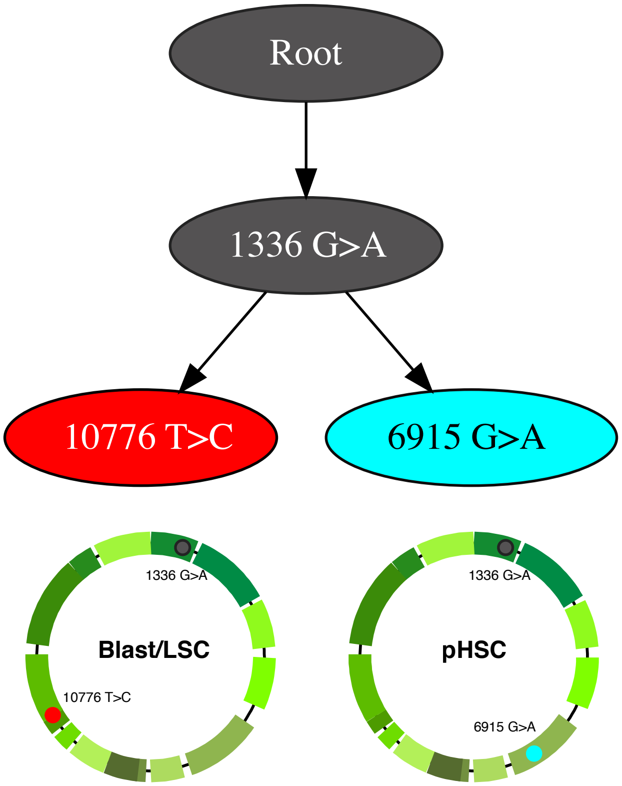
```
#-- S6
Heatmap(as.matrix(su444), col = viridis(100, option = "plasma"), show_row_names = TRUE,
    show_column_names = TRUE, top_annotation = su444_ann, column_split = rep(c("RNA-seq",
        "ATAC-seq"), each = 3), show_column_dend = FALSE, show_row_dend = FALSE,
    name = "Allele\nfreq.")
```



Supplementary Figure S6. A zoomed-in version of (**Supplementary Figure S3**), focused on variants specific to patient SU444.

We can also attempt to infer a cancer phylogenetic tree (CPT) relating the different mutations and illustrate their locations using the built-in `mitoPlot` function.

```
#-- If you want to reproduce the final figure (S7) you need to run this step
#-- However the figure here (below) has undergone significant post-processing
#-- Use SCITE to infer a CPT for patient SU444
clust444 <- varCluster(vars.rna.call[["SU444"]], method = "SCITE")

#-- S7 - bottom
mitoPlot(fixvarName(names(vars.rna.call[["SU444"]]@cluster))[c(1, 3, 1,
    2)], patient = rep(rev(c("pHSC", "Blast/LSC")), each = 2), showLegend = TRUE)
```



Supplementary Figure S7. The results of running `SCITE` (top) and the `mitoPlot` function (bottom) on the RNA-seq variant counts from patient SU444. The tree illustrates a potential phylogenetic relationship underlying cancerous clonal evolution differentiating the pre-leukemic (pHSC) sample from the blast/leukemic stem cell (LSC) samples.

3 Alternate workflow with the *Exclusionlist* filtering method

To complete our proof of concept, we apply a different universal variant filtering method based on existing exclusion lists that are shipped with the mitoClone2 package. This filtering method should only be used when the dataset includes only a single sample from one individual/time-point.

We re-import the allele count tables.

```
#-- Here is an example of how we would generate our allele count files locally - SKIPPED
## vars.rna <- baseCountsFromBamList(bamfiles =
## list.files('corces_RNA','\\.bam$',full = TRUE), sites =
## 'chrM:1-15000') the files need to be located in R's working
## directory!
#-- Loading the external data from GSE74246 (rna)
vars.rna <- readRDS("corces_nt_counts_per_position_rna.RDS")
#-- Loading the external data from GSE74912 (atac)
vars.atac <- readRDS("corces_nt_counts_per_position_quality30_atac.RDS")
```

We run mutationCallsFromExclusionlist on the RNA-seq allele count tables for each of these patient samples using default parameters to call mutations using the exclusion list included in the package.

3.1 Call variants from the RNA-seq data using provided *Exclusionlists*

```
# calling mutations using our exclusionlist
vars.rna.call <- mutationCallsFromExclusionlist(vars.rna)
```

```
## Subsetting for specific cells...
```

3.2 Downstream analysis similar to 2.5-2.9

From there, we will use the same downstream analysis as with the variant calling using the cohort filtering method.

```
#-- Pull all the identified mutations that are likely exclusive to patients
candidates <- sub(">", "", fixvarName(sort(unique(names(vars.rna.call@cluster)))),
    fixed = TRUE)

#-- Remove variants that appear in clusters separated by less than 3bp for each patient
candidates <- removeWindow(candidates, window = 3)

#-- Extract counts of putative candidates from the RNA-seq
rna.counts <- pullcountsVars(vars.rna, candidates)
plotData <- rna.counts$M/(rna.counts$M + rna.counts$N)

#-- Remove variants that seem to be universal (i.e. non-patient exclusive)
#-- Our cutoff is an allele frequency over 5% in more than 6 samples
plotData <- plotData[rowSums(plotData > 0.05) <= 6, ]

#-- Extract patient IDs
pts <- gsub("\\-.*", "", colnames(plotData))
#-- Set unique colors for each patient
pts.colors <- c("#FF7F00", "brown", "palegreen2", "khaki2", "#FB9A99",
    "darkorange4", "green1", "gold1", "steelblue4", "dodgerblue2", "darkturquoise",
    "orchid1", "black", "#6A3D9A", "blue1", "#CAB2D6", "gray70", "yellow3")
names(pts.colors) <- levels(factor(pts))

#-- Preparing a metadata data.frame that includes:
#-- the sample name
#-- the patient associated to the sample
#-- the technology used to generate the sample
meta.data <- melt(c(names(vars.rna), names(vars.atac)))
row.names(meta.data) <- meta.data$value
meta.data$Patient <- gsub("\\-.*", "", meta.data$value)
meta.data$Method <- paste0(gsub(".*\\_", "", meta.data$value), "-seq")
meta.data$value <- NULL

#-- Setup heatmap annotation - plotData
ha_column = HeatmapAnnotation(df = meta.data[colnames(plotData), "Patient",
    drop = FALSE], col = list(Patient = pts.colors), show_annotation_name = FALSE)
#-- Build a heatmap of the RNA-seq allele frequencies for each variant
hrna <- Heatmap(plotData, col = viridis(100, option = "plasma"), show_row_names = TRUE,
    show_column_names = FALSE, top_annotation = ha_column, column_title = "RNA-seq",
    show_column_dend = FALSE, show_row_dend = FALSE, column_title_gp = gpar(fontsize = 24),
```

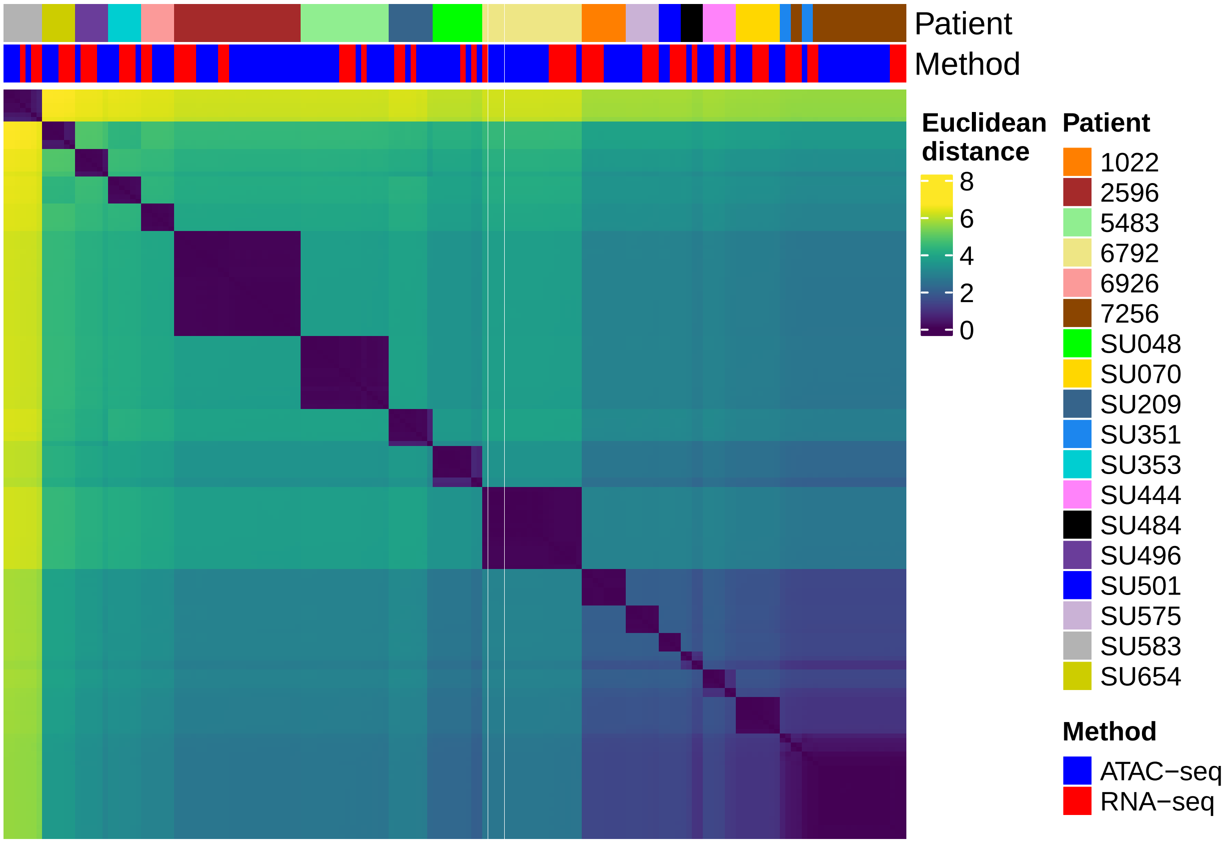
```r
    row_names_gp = gpar(fontsize = 2), name = "Allele\nfreq.")

#-- Extract counts of putative candidates from the ATAC-seq
atac.counts <- pullcountsVars(vars.atac, row.names(plotData))
plotData.atac <- atac.counts$M/(atac.counts$M + atac.counts$N)

#-- Setup heatmap annotation for plotData.atac
ha_column = HeatmapAnnotation(df = meta.data[colnames(plotData.atac), "Patient",
    drop = FALSE], col = list(Patient = pts.colors))
#-- Build a heatmap of the ATAC-seq allele frequencies for each variant
hatac <- Heatmap(plotData.atac, col = viridis(100, option = "plasma"),
    show_row_names = TRUE, show_column_names = FALSE, top_annotation = ha_column,
    column_title = "ATAC-seq", show_column_dend = FALSE, show_row_dend = FALSE,
    column_title_gp = gpar(fontsize = 24), row_names_gp = gpar(fontsize = 2),
    name = "Allele\nfreq.")

#-- Plot heatmaps for both technologies side-by-side
hrna + hatac
```

```r
#-- Extract mean allele freq per patient per technology and merge
df.rna <- getptMean(plotData, "RNA-seq")
df.atac <- getptMean(plotData.atac, "ATAC-seq")

#-- Sanity check all data.frames should be similar
all(c(identical(df.rna$Patient, df.atac$Patient)), identical(df.rna$var,
    df.atac$var))

#-- merge into a new data.frame
df.both <- df.rna
df.both$"ATAC-seq" <- df.atac$"ATAC-seq"

#-- Perform a correlation test on the samples with Pearson
cor_both <- cor.test(df.both$ATAC, df.both$RNA)
colnames(df.rna)[2] <- "Patient"
```

```r
#-- Plot the correlation and allele frequency comparison
ggplot(df.both, aes(x = `RNA-seq`, y = `ATAC-seq`, color = Patient)) +
    geom_point(size = 4) + theme_classic(base_size = 14) + geom_abline(slope = 1,
    color = "grey", linetype = "dashed") + scale_colour_manual(values = pts.colors) +
    annotate("text", x = 0.25, y = 0.8, label = paste0("r = ", cor_both$estimate)) +
    ylab("ATAC-seq\nAllele frequency") + xlab("RNA-seq\nAllele frequency") +
    theme(plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.text = element_text(size = 8)) +
    guides(color = guide_legend(ncol = 2))
```

```r
#-- Merge the RNA-seq and ATAC-seq allele frequency data.frames
merged.vars <- merge(plotData.atac, plotData, by = "row.names", all = T)
row.names(merged.vars) <- merged.vars$Row.names
merged.vars$Row.names <- NULL

#-- Create distance matrix from merged method data.frame
#-- The default method is set to 'euclidean'
distmat <- data.frame(as.matrix(dist(t(merged.vars))))
colnames(distmat) <- row.names(distmat)
```

```
distance.colors <- list(Method = c(`RNA-seq` = "red", `ATAC-seq` = "blue"),
    Patient = pts.colors)
ha_column <- HeatmapAnnotation(df = meta.data[colnames(distmat), ], col = distance.colors)

#-- S8
Heatmap(distmat, col = viridis(100), show_row_names = FALSE, show_column_names = FALSE,
    top_annotation = ha_column, show_column_dend = FALSE, show_row_dend = FALSE,
    name = "Euclidean\ndistance")
```



Supplementary Figure S8. Clustered distance matrix of combined variant allele frequencies from ATAC-seq and RNA-seq, using the *Exclusionlist* filtering method.

3.3 The *Exclusionlist* filtering method compared to the *Cohort* method

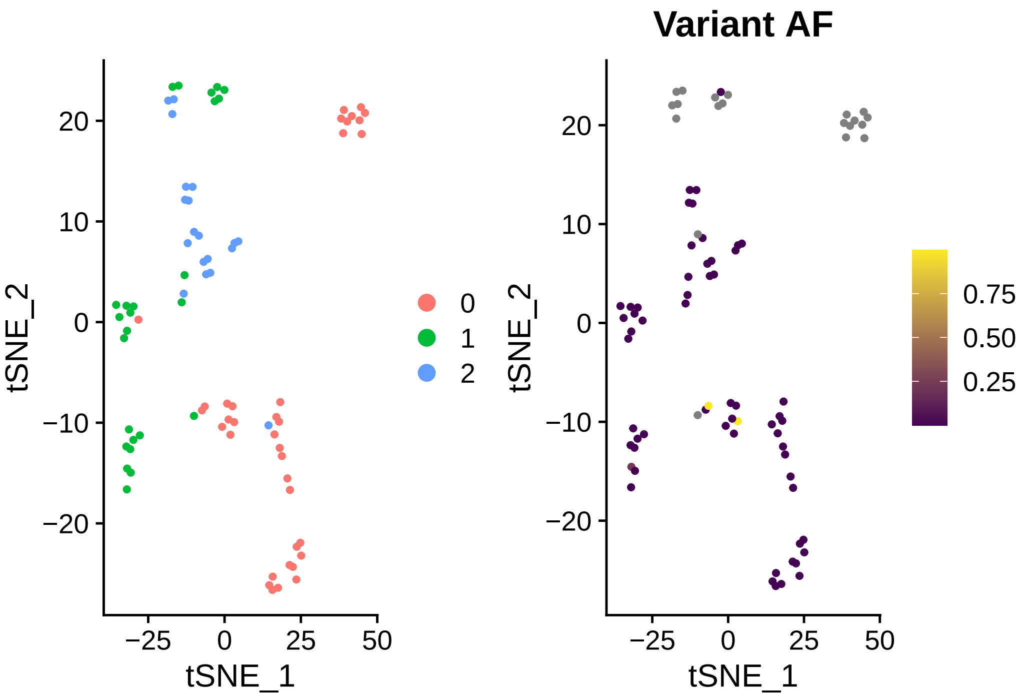
The Exclusionlist variant filtering approach increased false negative hits and slightly decreased the correlation between RNA-seq and ATAC-seq allele frequencies. It does not work as well as the cohort method for filtering shared variants but still can be used with confidence to identify clones when no other option is available.

In conclusion, we suggest that users always use the cohort method whenever possible (i.e. access to multiple similar datasets).

4 Adding metadata to Seurat objects

Below is just an example where we add the variant allele frequencies calculated above to the minimal test dataset included with Seurat for illustrative purposes.

```
library(Seurat)

## randomly assign our labels to cells
set.seed(123)
colnames(pbmc_small)[seq(colnames(plotData))] <- sample(colnames(plotData))

## add the VAF information
pbmc_small <- AddMetaData(pbmc_small, col.name = "Variant AF", metadata = plotData["1336G>A",
    ])

## plot the default tSNE and the VAF colored one
DimPlot(pbmc_small) + FeaturePlot(pbmc_small, features = "Variant AF",
    cols = viridis(2))
```



Session information

```
## R version 4.3.3 (2024-02-29)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Sonoma 14.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  LAPACK ve
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Zurich
## tzcode source: internal
##
```

```
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] Seurat_5.0.0          SeuratObject_5.0.2    sp_2.1-4
##  [4] knitr_1.47            ComplexHeatmap_2.18.0 viridis_0.6.5
##  [7] viridisLite_0.4.2     reshape2_1.4.4        ggplot2_3.5.1
## [10] mitoClone2_1.8.1
##
## loaded via a namespace (and not attached):
##   [1] RcppAnnoy_0.0.22            splines_4.3.3
##   [3] later_1.3.2                 BiocIO_1.12.0
##   [5] bitops_1.0-7                filelock_1.0.3
##   [7] tibble_3.2.1                polyclip_1.10-6
##   [9] XML_3.99-0.16.1             fastDummies_1.7.3
##  [11] lifecycle_1.0.4             doParallel_1.0.17
##  [13] globals_0.16.3              deepSNV_1.48.0
##  [15] lattice_0.22-6              MASS_7.3-60.0.1
##  [17] magrittr_2.0.3              plotly_4.10.4
##  [19] rmarkdown_2.27              yaml_2.3.8
##  [21] httpuv_1.6.15               sctransform_0.4.1
##  [23] Rhtslib_2.4.1               spam_2.10-0
##  [25] spatstat.sparse_3.0-3       reticulate_1.37.0
##  [27] pbapply_1.7-2               cowplot_1.1.3
##  [29] DBI_1.2.3                   RColorBrewer_1.1-3
##  [31] abind_1.4-5                 zlibbioc_1.48.2
##  [33] Rtsne_0.17                  GenomicRanges_1.54.1
##  [35] purrr_1.0.2                 BiocGenerics_0.48.1
##  [37] RCurl_1.98-1.14             VariantAnnotation_1.48.1
##  [39] rappdirs_0.3.3              circlize_0.4.16
##  [41] GenomeInfoDbData_1.2.11     IRanges_2.36.0
##  [43] S4Vectors_0.40.2            ggrepel_0.9.5
##  [45] irlba_2.3.5.1               spatstat.utils_3.0-4
##  [47] listenv_0.9.1               goftest_1.2-3
##  [49] RSpectra_0.16-1             spatstat.random_3.2-3
##  [51] fitdistrplus_1.1-11         parallelly_1.37.1
##  [53] leiden_0.4.3.1              codetools_0.2-20
##  [55] DelayedArray_0.28.0         xml2_1.3.6
##  [57] tidyselect_1.2.1            shape_1.4.6.1
##  [59] farver_2.1.2                spatstat.explore_3.2-7
##  [61] matrixStats_1.3.0           stats4_4.3.3
##  [63] BiocFileCache_2.10.2        GenomicAlignments_1.38.2
##  [65] jsonlite_1.8.8              GetoptLong_1.0.5
##  [67] progressr_0.14.0            ggridges_0.5.6
##  [69] survival_3.7-0              iterators_1.0.14
##  [71] foreach_1.5.2               tools_4.3.3
##  [73] progress_1.2.3              ica_1.0-3
##  [75] Rcpp_1.0.12                 glue_1.7.0
##  [77] gridExtra_2.3               SparseArray_1.2.4
##  [79] xfun_0.44                   MatrixGenerics_1.14.0
##  [81] GenomeInfoDb_1.38.8         dplyr_1.1.4
##  [83] withr_3.0.0                 formatR_1.14
##  [85] fastmap_1.2.0               fansi_1.0.6
##  [87] digest_0.6.35               R6_2.5.1
##  [89] mime_0.12                   colorspace_2.1-0
##  [91] scattermore_1.2             tensor_1.5
##  [93] spatstat.data_3.0-4         biomaRt_2.58.2
##  [95] RSQLite_2.3.7               tidyr_1.3.1
##  [97] utf8_1.2.4                  generics_0.1.3
##  [99] data.table_1.15.4           rtracklayer_1.62.0
## [101] htmlwidgets_1.6.4           prettyunits_1.2.0
## [103] httr_1.4.7                  S4Arrays_1.2.1
## [105] uwot_0.1.16                 pkgconfig_2.0.3
## [107] gtable_0.3.5                blob_1.2.4
## [109] lmtest_0.9-40               XVector_0.42.0
## [111] htmltools_0.5.8.1           dotCall64_1.1-1
## [113] clue_0.3-65                 scales_1.3.0
## [115] Biobase_2.62.0              png_0.1-8
## [117] rjson_0.2.21                nlme_3.1-164
## [119] curl_5.2.1                  cachem_1.1.0
## [121] zoo_1.8-12                  GlobalOptions_0.1.2
## [123] stringr_1.5.1               KernSmooth_2.23-24
## [125] parallel_4.3.3              miniUI_0.1.1.1
## [127] AnnotationDbi_1.64.1        restfulr_0.0.15
## [129] pillar_1.9.0                vctrs_0.6.5
## [131] RANN_2.6.1                  VGAM_1.1-11
## [133] promises_1.3.0              dbplyr_2.5.0
## [135] xtable_1.8-4                cluster_2.1.6
## [137] evaluate_0.23               tinytex_0.51
## [139] GenomicFeatures_1.54.4      cli_3.6.2
## [141] compiler_4.3.3              Rsamtools_2.18.0
## [143] rlang_1.1.4                 crayon_1.5.2
## [145] future.apply_1.11.2         labeling_0.4.3
## [147] plyr_1.8.9                  stringi_1.8.4
## [149] deldir_2.0-4                BiocParallel_1.36.0
## [151] munsell_0.5.1               Biostrings_2.70.3
## [153] lazyeval_0.2.2              spatstat.geom_3.2-9
## [155] Matrix_1.6-5                RcppHNSW_0.6.0
## [157] BSgenome_1.70.2             patchwork_1.2.0
## [159] hms_1.1.3                   bit64_4.0.5
## [161] future_1.33.2               KEGGREST_1.42.0
## [163] shiny_1.8.1.1               SummarizedExperiment_1.32.0
## [165] highr_0.11                  ROCR_1.0-11
## [167] igraph_2.0.3                memoise_2.0.1
## [169] bit_4.0.5
```