timeseries_clustering

Aileen

2025-11-17

In this analysis, I am clustering a time series dataset of microRNA expression in plant floral tissue throughout development in the Brassica rapa genotype, R-o-18.

The goal of this analysis is to cluster miRNAs based on similar time series expression profiles, using fuzzy c-means clustering with the R package Mfuzz.

DATA DESCRIPTION

This dataset consists of:

Genotype: R-o-18 (late flowering) Tissue types: Apex, leaf Timepoints: 15 timepoints, from day 9 until day 37

```
## Rows: 19097 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): sample, mirna, genotype, timepoint, tissue, stages
## dbl (4): cpm, total_reads, count, rep
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

sample	mirna	cpm	total_reads	count	genotype	timepoint	tissue	rep	stages	batch
RO_leaf_d9_B3	Bra-miR156a	542.9528308	25941480	14085	R-o-18	9	leaf	3	vegetative	run1
RO_leaf_d9_B3	Bra-miR156d	4.5872479	25941480	119	R-o-18	9	leaf	3	vegetative	run1
RO_leaf_d9_B3	Bra-miR156h	1.6961253	25941480	44	R-o-18	9	leaf	3	vegetative	run1
RO_leaf_d9_B3	Bra-miR156i	7.7096604	25941480	200	R-o-18	9	leaf	3	vegetative	run1
RO_leaf_d9_B3	Bra-miR156n	0.0770966	25941480	2	R-o-18	9	leaf	3	vegetative	run1
RO_leaf_d9_B3	Bra-miR157a	281.6338929	25941480	7306	R-o-18	9	leaf	3	vegetative	run1

VST & BATCH CORRECTION

For clustering analysis, variance stabilizing transformation is performed on the dataset to prevent extremely highly expressed genes to skew the clustering. This transformation allows both highly expressed and lowly expressed genes to contribute equally to the clustering. This is done with DESeq2's varianceStabilizingTransformation() function.

More on VST: https://bioconductor.org/packages//release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html#data-transformations-and-visualization

The data from this project was sequenced in two separate batches, therefore I performed batch correction prior to time series clustering. This can be done using limma's removeBatchEffect() function.

```
# prep data
counts <- data_complete %>%
  filter(tissue == "apex", genotype == "R-o-18", stages != "bbch51") %>%
  dplyr::mutate(timepoint = as.numeric(str_extract(timepoint, "[0-9]+$"))) %>%
  select(sample, mirna, count) %>%
  pivot_wider(values_from = count, names_from = sample) %>%
  tibble::column_to_rownames(var = "mirna")

coldata <- data_complete %>%
  filter(tissue == "apex", genotype == "R-o-18", stages !="bbch51") %>%
  select(sample, genotype, tissue, timepoint, rep, stages, batch) %>%
  unique() %>%
  dplyr::mutate(timepoint = as.numeric(str_extract(timepoint, "[0-9]+$"))) %>%
  arrange(sample) %>%
  tibble::column_to_rownames(var = "sample")

counts <- counts[, rownames(coldata)]

# get VST data
# 1. filter very low-count features (reduce noise)
keep <- rowSums(counts) >= 10
counts_f <- counts[keep, ]

# 2. construct DESeqDataSet
dds <- DESeqDataSetFromMatrix(countData = counts_f,
                              colData = coldata,
                              design = ~ 1)
```

```
## converting counts to integer mode
```

```
# 3. VST (variance stabilizing transformation)
vsd <- varianceStabilizingTransformation(dds, blind = TRUE)

# 4. convert into a matrix
mat <- assay(vsd)                  # rows=miRNAs, cols=samples

coldata <- coldata %>%
  tibble::rownames_to_column("sample")

## Batch correction

n_cold <- coldata[match(colnames(mat), coldata$sample),]
design <- model.matrix(~ timepoint, data = n_cold)

vst_mat_batch_corrected <- removeBatchEffect(mat, batch = n_cold$batch, design = design)
```

```
vst_long <- as.data.frame(vst_mat_batch_corrected) %>%
  tibble::rownames_to_column("mirna") %>%
  pivot_longer(-mirna, names_to = "sample", values_to = "vst") %>%
  left_join(n_cold, by = "sample")

mat_mean_by_time <- vst_long %>%
  group_by(mirna, genotype, timepoint) %>%
  summarise(vst_mean = mean(vst), .groups = "drop")

mat_wide <- mat_mean_by_time %>%
  select(mirna, timepoint, vst_mean) %>%
  pivot_wider(names_from = timepoint, values_from = vst_mean) %>%
  tibble::column_to_rownames("mirna")

mat_wide %>%
  head() %>%
  kable()
```

	9	11	13	15	17	19	21	23
Bra-miR1140	12.536339	12.631770	12.368350	12.335923	12.427369	12.613775	12.670444	12.624018
Bra-miR1511	8.514086	8.812541	8.484136	8.044546	8.393019	8.664611	8.638006	8.691169
Bra-miR156a	13.403803	12.853364	12.884060	12.629172	12.165180	12.312639	11.180482	11.306630
Bra-miR156d	6.012180	5.529457	4.859305	4.318566	3.921534	4.238256	4.887688	4.192876
Bra-miR156h	4.882890	4.696341	4.961811	4.921100	4.554366	4.612558	3.644095	4.151982
Bra-miR156i	6.310326	6.252142	6.276494	6.569275	6.638218	6.455633	5.863155	5.858613

CLUSTERING

Fuzzy c-means clustering is a soft clustering method which assigns membership values to clusters based on how well it corresponds with the gene it is assigned to. Some genes may have shared membership between two clusters, such as transition genes which change expression dynamics throughout development. This makes fuzzy c-means clustering an ideal method for gene expression time series clustering during development.

More on Mfuzz and fuzzy c-means clustering: https://bioconductor.statistik.tu-dortmund.de/packages/2.3/bioc/vignettes/Mfuzz/inst/doc/Mfuzz.pdf

```
# Create ExpressionSet
eset <- ExpressionSet(as.matrix(mat_wide))

# Standardize ExpressionSet
eset_std_ro_bc <- standardise(eset)

# Estimate number of clusters
m_est_ro_bc <- mestimate(eset_std_ro_bc)
```

```r
# Perform clustering
cl_ro_bc <- mfuzz(eset_std_ro_bc, c = 5, m = m_est_ro_bc)

# Plot clusters of miRNA expression
# (I commented this out since it plots on XQuartz)

#mfuzz.plot(eset_std_ro_bc,
#           cl = cl_ro_bc,
#           mfrow = c(5,1),
#           time.labels = c(9,11,13,15,17,19,21,23),
#           new.window = F
#           )

# Get membership values per cluster
ro_membership_raw <- cl_ro_bc$membership %>%
   as.data.frame() %>%
   tibble::rownames_to_column("mirna")

# Get hard assigned clusters (cluster with highest membership value per gene)
ro_hardcluster_raw <- cl_ro_bc$cluster %>%
   as.data.frame() %>%
   tibble::rownames_to_column("mirna") %>%
  dplyr::rename(cluster = ".")
```

Check members

The clustering produces two outputs:

1) $membership : Membership values of all clusters in each gene

mirna	1	2	3	4	5
Bra-miR1140	0.0522666	0.1741949	0.0783201	0.2929608	0.4022577
Bra-miR1511	0.0816215	0.1304497	0.0865218	0.3117855	0.3896215
Bra-miR156a	0.8058065	0.0146697	0.0622960	0.0774513	0.0397764
Bra-miR156d	0.0848744	0.0178874	0.0300323	0.7956230	0.0715829
Bra-miR156h	0.7973842	0.0182039	0.0956480	0.0468394	0.0419244
Bra-miR156i	0.2490965	0.0413565	0.5962142	0.0543433	0.0589895

2) $cluster : The cluster with the highest membership value for each gene, which would be the cluster which best corresponds with the gene's expression profile

mirna	cluster
Bra-miR1140	5
Bra-miR1511	5
Bra-miR156a	1
Bra-miR156d	4
Bra-miR156h	1
Bra-miR156i	3

For downstream analysis, we could analyse genes with strong membership in a specific cluster, thus having shared expression profiles and possibly sharing similar biological function (in this case, miRNAs which promote or repress flowering development). Thus, I wrote a data wrangling function to filter out genes with low membership (< 60%) and presented it in a reader-friendly format.

```r
get_members <- function(df){
  cl_members <- df$membership %>%
  as.data.frame() %>%
  tibble::rownames_to_column("mirna") %>%
  pivot_longer(-mirna, names_to = "cluster", values_to = "membership") %>%
  filter(membership > 0.6) %>%
  group_by(cluster) %>%
  mutate(miRNA_idx = row_number()) %>%   # index within each cluster
  ungroup() %>%
  pivot_wider(names_from = cluster, values_from = mirna, names_prefix = "cluster_", id_cols = miRNA_idx
  select(-c(miRNA_idx))
  return(cl_members)
}

high_mem_cluster_ro <- get_members(cl_ro_bc)

high_mem_cluster_ro %>%
  kable()
```

cluster_1	cluster_4	cluster_2	cluster_3
Bra-miR156a	Bra-miR156d	Bra-miR159a	Bra-miR169g
Bra-miR156h	Bra-miR398a	Bra-miR159c	Bra-miR169i
Bra-miR157a	Bra-miR398c	Bra-miR165a	Bra-miR169q
Bra-miR164a	Bra-miR408	Bra-miR168b	Bra-miR2111a
Bra-miR169a	Bra-miR9558	Bra-miR172a	Bra-miR395a
Bra-miR393a	Bra-miRN334	Bra-miR172c	Bra-miR6032
Bra-miR396a	Bra-miRN344	Bra-miR172d	Bra-miRN378
Bra-miR5718	Bra-miRN381	Bra-miR319a	NA
Bra-miR827	NA	Bra-miR319c	NA
Bra-miRN340	NA	Bra-miR319e	NA
Bra-miRN350	NA	Bra-miR319f	NA
Bra-miRN366	NA	Bra-miR394a	NA
Bra-miRN367	NA	Bra-miR5711	NA
Bra-miRN369	NA	Bra-miR5723	NA
NA	NA	Bra-miR9554	NA
NA	NA	Bra-miR9560a	NA
NA	NA	Bra-miRN271	NA
NA	NA	Bra-miRN349	NA
NA	NA	Bra-miRN370	NA
NA	NA	Bra-miRN375	NA