CellSpectra: Advanced Parameters and Underlying Genes

Konstantin A. Klötzer

2025-01-19

CellSpectra parameters and advanced usage

We will now focus on some of the CellSpectra parameters and how they impact analysis results. We will first load our packages, define an output_folder_base, load a seurat object (subsetted version of our extended human kidney atlas), and define some columns for seurat_object preparation (see our Query_Reference_Tutorial if not familiar).

Cave: Developmental version using Seurat V4 objects!

Note: This object has condition information stored in "disease_grouped_qr" column with one query sample ("sample_of_interest") and a few healthy reference samples

```r
start.time <- Sys.time()

library(Matrix)
library(dplyr)
library(Seurat)
library(CellSpectra)
library(KEGGREST)

output_folder_base = "output/"

dir.create(output_folder_base)

seurat_object = readRDS("Human_PT_subset.rds")

# We assume the respective metadata is stored in the following columns
sample_id_col <- "orig_ident"
condition_col <- "disease_grouped_qr"
celltype_annotation_col <- "C_scANVI_modified1"

#we use the prepare function to get our seurat object in shape
seurat_object <- prepare_seurat_object(
  seurat_obj = seurat_object, #our object to prepare
  celltype_col = celltype_annotation_col, #column in which annotation is stored
  sample_id_col = sample_id_col, #sample identifier
  condition_col = condition_col, #column storing the condition information
  query_list = c("sample_of_interest"), #categories in the condition_col defining the query samples
  control_list = c("reference_samples") #categories in the condition_col defining the reference
)
```

Check the number of cells in our object

To keep things easy, our provided object only contains PT cells. We will now check the cell numbers per sample. Consider that sample 32-2 is our diseased patient.

```r
table(seurat_object$orig_ident)
```

```
##
## 18-162 18-312   32-2   3499
##    629   1531    377     74
```

As we can see, 2 of our reference samples have 1.5-4 times the cells of our query. Since scRNA-seq data is sparse, different cell numbers / capturing depth could introduce technical noise. CellSpectra controls these differences by matching the total counts to the counts of our query. To not lose too many cells we could create pseudo-replicates per reference sample.

Lets assume our query has 300 cells, and most reference samples have many more (e.g. 3000 cells) we could make approx. 10 reference pseudoreplicates from one reference sample. In other words, we would lose cells. Here 2 or 3 replicates would seem reasonable. However, we don't recommend using pseudo-replicates if 1. Cell numbers of the query are similar or even larger compared to the reference samples and 2. Total number of biological replicates in the reference is small. Using pseudoreplicates would result in false homogeneity in the LOO approach. In the manuscript, since we were usually working with many reference samples (50 in the human extended kidney), we used 3 pseudoreplicates.

Here, due to the small number of reference samples, we will not use pseudoreplicates for now.

Create on_the_fly references and create a KEGG gene set database

Lets create the reference (and a Kegg gene set database if not already provided)

```r
#Step 1
create_references(
  seurat_object = seurat_object,
  output_folder_base = output_folder_base,
  num_replicates = 1,
  cell_types = c("PT"),
  cell_number_threshold = 10,
  seed = 123
)

#this could take a while to run - we provide a computed version
#kegg_gene_matrix <- process_gene_sets_from_KEGG("hsa", seurat_object = seurat_object)
#saveRDS(kegg_gene_matrix, "output/go_sets_modified.rds")
```

We should now have the folder output/PT and the file samples_of_interest.rds. In output/PT/ we should find a subfolder for our query sample (storing the query and on-the-fly reference gene expression matrix) and a valid_samples file (this file would store information which samples had enough cells included to be considered).

This is everything we need to run CellSpectra. Consider, that we will not filter any low expression genes for now.

```r
#Step 2
run_spectra(output_folder_base = output_folder_base, cell_types = c("PT"),
            CHISQ.MAX = 4, expression_threshold = -1,
            gene_number_threshold = 10, QC_report = TRUE)
```

We should now have three additional subfolders of output/

1. R2: Here we store R2 values for all gene set of the query compared to the reference
2. Pval: Pvalus of our sample level Chi-Square-Statistic of residuals
3. Padj: Same Pvals after correcting for multiple testing (fdr) for several pathways

Lets open the R2 file:

```r
#load results
r2_df = read.csv("output/R2/R2_PT.csv", row.names = 1)
r2_df = t(r2_df)
```

In this case, the KEGG pathway "Staphylococcus.aureus.infection" had the lowest R2 of ~0.24. However, while this indicates that the query sample gene expression pattern could not be predicted based on the reference expression pattern, it doesn't tell us if this is expected inter-sample variance (= humans are not coordinated in this function) or a significant deviation from a usually coordinated gene expression pattern (query is "dyscoordinated"). It is also possible, that technical noise is responsible for the low degree of coordination. To better understand what is happening, we will now take a look at our QC report:

```r
#Step 2
qc_report_query = read.csv("output/PT/subfolder_sample_32-2/pathway_QC.csv", row.names = 1)
```

Our three reference samples showed a mean R2 0.55 with a SD of 0.28 and a V1 Variance explained of 0.89. This indicates, that within the reference the estimation of V1 was already poor and random. Reference samples did not show a convincing pattern of coordination. However, we will now further consider the pvalue of our statistical test to provide more confidence.

```r
#load results
pval_df = read.csv("output/Padj/Padj_PT.csv", row.names = 1)
pval_df = t(pval_df)
```

As we check the pvalues for all gene sets, we see that many pathways were significant after fdr correction. We generally assume that the real representation of biological inter-sample variance is poor, since we only are working with 3 different reference samples. Lets focus on the cAMP.signaling.pathway, which showed a high ranking based on the sample pvalue:

R2 of 0.91 with a LOO mean R2 of 0.94, SD of 0.02 and a V1VE of 0.98.

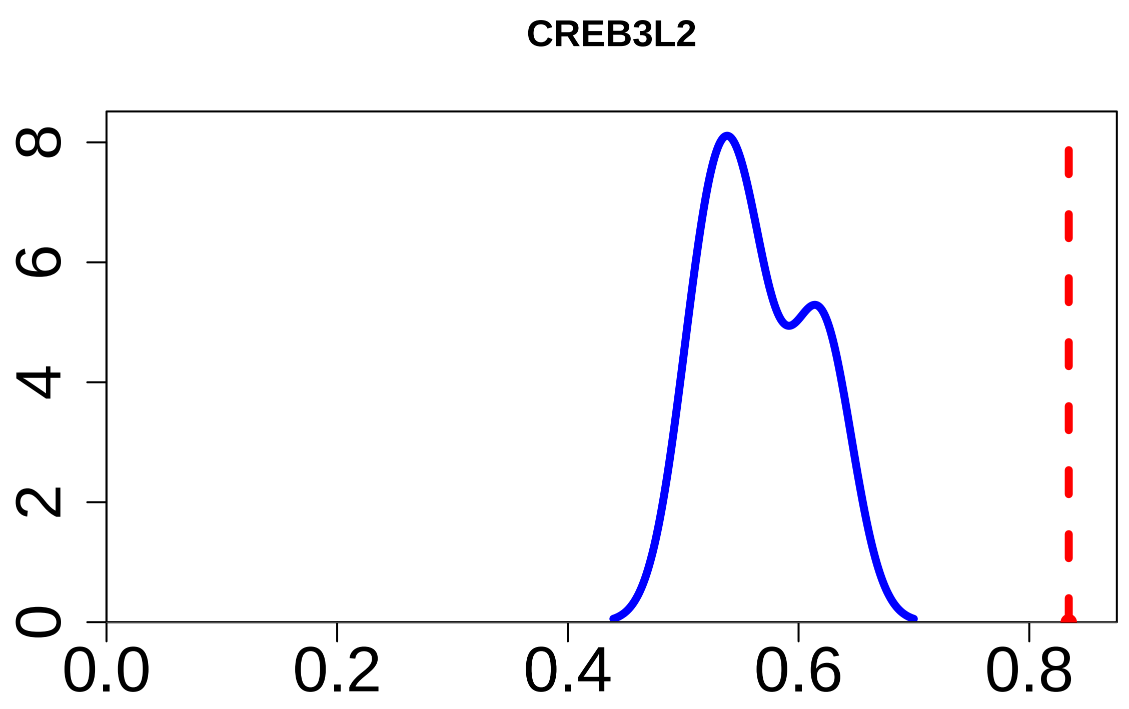
This is a highly coordinated pathway within our reference. While our query R2 of 0.91 is still high, our p value gives us confidence that our diseased sample shows significant changes within this pathway.

Focusing on underlying genes

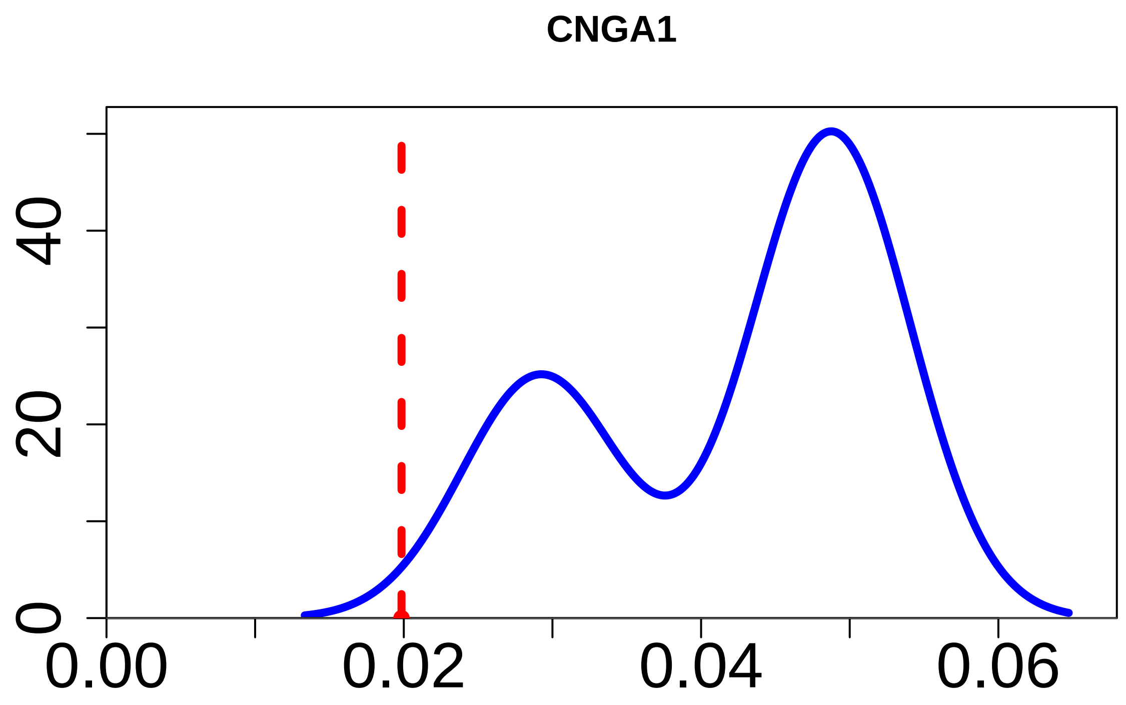
We next want to understand which gene expression changes are responsible for the observed "dyscoordination" in the cAMP.signaling.pathway. We can use our report_genes() function to do that:

```r
CHISQ.MAX = 4
gene_set_name = "hsa04024 cAMP signaling pathway - Homo sapiens (human)"

drivinggenes1 <- report_genes(output_folder_base, gene_set_name,
                              cell_type = "PT",
                              sample_of_interest = "32-2")
```

```r
#we can load our query and on-the-fly reference for plotting
datH = readRDS("output/PT/subfolder_sample_32-2/datH.rds")
datD = readRDS("output/PT/subfolder_sample_32-2/datD.rds")

#positive underlying gene example
gene_neg <- "CREB3L2"
expression_H_neg <- datH[, gene_neg]
expression_D_neg <- datD[, gene_neg]

# Calculate density
density_H_neg <- density(expression_H_neg)

# Set x-axis and y-axis to start from 0, and add a slight margin to max values
xlim_neg <- c(0, max(density_H_neg$x, expression_D_neg) * 1.05)
ylim_neg <- c(0, max(density_H_neg$y) * 1.05)

# Plot the density with thicker blue line, ensuring axes meet at 0 precisely
plot(density_H_neg, main = paste(gene_neg),
     xlab = "", ylab = "", col = "blue",
     xlim = xlim_neg, ylim = ylim_neg, lwd = 4, cex.axis = 2,
     xaxs = "i", yaxs = "i")

# Add red dashed line and points
abline(v = expression_D_neg, col = "red", lwd = 4, lty = 2)
points(expression_D_neg, 0, col = "red", pch = 19)
```



```r
#negative underlying gene example
gene_neg <- "CNGA1"
expression_H_neg <- datH[, gene_neg]
expression_D_neg <- datD[, gene_neg]

# Calculate density
density_H_neg <- density(expression_H_neg)

# Set x-axis and y-axis to start from 0, and add a slight margin to max values
xlim_neg <- c(0, max(density_H_neg$x, expression_D_neg) * 1.05)
ylim_neg <- c(0, max(density_H_neg$y) * 1.05)

# Plot the density with thicker blue line, ensuring axes meet at 0 precisely
plot(density_H_neg, main = paste(gene_neg),
     xlab = "", ylab = "", col = "blue",
     xlim = xlim_neg, ylim = ylim_neg, lwd = 4, cex.axis = 2,
     xaxs = "i", yaxs = "i")

# Add red dashed line and points
abline(v = expression_D_neg, col = "red", lwd = 4, lty = 2)
points(expression_D_neg, 0, col = "red", pch = 19)
```



The blue line indicates the reference gene expression distribution with the red line indicating the gene expression of the query. Since we only had a very small reference for demonstration purposes, the results are not very robust, but can show some trends. Plotting example underlying genes can help us to better understand the underlying gene expression changes. Next, we will repeat the analysis after adjusting some CellSpectra parameters.

Impact of gene filtering

Sparsity is one of the major issues in single-cell analysis. While analysis pipelines already filter genes of low expression, some genes might be low expressed in a specific cell type or in a specific set of samples. Therefore, we can directly remove genes of low expression in the query or reference gene expression matrix before running CellSpectra. The need for filtering genes depends on the cell type, cell numbers, the gene set database (number of genes per gene set) or tissu of investigation. For example, we applied gene filtering to the KEGG analysis in our manuscript to improve the robustness of our results. However, especially in rare cell types zero inflation is very common after pseudobulking, and filtering too stringent means that analysis cannot be performed for most pathways. We always recommend to apply gene filtering when focusing on underlying genes.

```r
#Step 2
run_spectra(output_folder_base = output_folder_base, cell_types = c("PT"),
            CHISQ.MAX = 4, expression_threshold = 0,
            gene_number_threshold = 10, QC_report = TRUE)

#load results
pval_df = read.csv("output/Padj/Padj_PT.csv", row.names = 1)
pval_df = t(pval_df)
```

```r
#load results
r2_df = read.csv("output/R2/R2_PT.csv", row.names = 1)
r2_df = t(r2_df)

qc_report_query = read.csv("output/PT/subfolder_sample_32-2/pathway_QC.csv",
                           row.names = 1)
```

Here we see that not filtering genes of zero expression increased the number of significant pathways in our query. Gene filtering is applied based on the normalized query gene expression matrix. If the average expression of a gene across the reference samples is below the threshold, genes are also removed from the analysis.

In summary: Users should think about the degree of sparsity and frequency of low expressed genes after pseudobulking in their data, since a high number of genes of zero / low expression, can inflate the statistics. If gene filtering is not applied but the data has a high count of zeros / very few cells, it might be advisable to check robustness of significant pathways for the filtering of low expressed genes. Does my identified pathway of interest stay significant after repeating the analysis with a more stringent filtering applied? Can I extract meaningful underlying genes with significant gene expression changes? Does my QC report indicate a good estimation of V1, with a small SD? If the answer to all these questions is no, a critical view on the result might be appropriate.

Impact of CHISQ.MAX

We already took a look at our underlying genes underlying the "dyscoordination" p value. One very easy and intuitive interpretation of underlying genes in this context is the question, if my gene in the query showed a much higher or much lower expression level compared to what was expected based on the reference gene expression pattern estimation (if our query could be perfectly predicted by the linear model). For the chi-square statistic we are applying, we are squaring the "normalized residuals" that can be extracted with the report_genes() function. However, we don't want individual genes to drive our p value. Therefore, we define the max. value that a gene can have after its "normalized residual" is squared (CHISQ.MAX). This parameter does not have an impact on the R2!

Lets check the pvalue after gene filtering for our cAMP Signaling Pathway again: padj = 1.365786e-05

Top pos. driver genes and normalized residuals: CREB3L2 4.3688108147

CALM3 3.8256698920

RAC1 3.7808482458

A CHISQ.MAX = 4 (CellSpectra's default) corresponds to a normalized residual of 2 or -2. (2^2 = 4). In other words, any underlying gene with a value greater than 2 is automatically set to 2. This would be the case for all of our top 3 pos underlying genes. These caveats ensure that the entire pathway is dyscoordinated, and not just very few individual genes.

We will now check what impact CHISQ.MAX = 4 will have on our pval:

```r
#Step 2
run_spectra(output_folder_base = output_folder_base, cell_types = c("PT"),
            CHISQ.MAX = 3, expression_threshold = 0,
            gene_number_threshold = 10, QC_report = TRUE)

#load results
pval_df = read.csv("output/Padj/Padj_PT.csv", row.names = 1)
pval_df = t(pval_df)
```

```r
#load results
r2_df = read.csv("output/R2/R2_PT.csv", row.names = 1)
r2_df = t(r2_df)

qc_report_query = read.csv("output/PT/subfolder_sample_32-2/pathway_QC.csv",
                           row.names = 1)
```

Our new results: cAMP.signaling.pathway padj = 4.761164e-03

The lower we set CHISQ.MAX, the more stringent we expect the entire pathway to be dyscoordinated. How much individual genes are allowed to drive the measured dyscoordination depends on the investigators research questions, the gene set database, the cell type and sparsity, etc. We recommend to start with the default setting and to tune the parameters if needed (see below).

Summary and General Recommendations:

We recommend to start with out default settings: CHISQ.MAX = 4, expression_threshold = 0

Some key questions we should ask after browsing our first results:

1.) Make sure that the results align with what you would expect from your biological system. For instance: Are you able to distinguish healthy and disease based on the dyscoordination quantity? Are expected pathways represented in the results?

2.) Check the QC report of you query: How well did the reference set of samples estimate V1? Are my reference samples too heterogeneous and noisy (trend towards low average R2 and high SD, low V1VE)? Is my reference too homogeneous (ratio of biosamples / pseudoreplicates too small? does my reference really represent the full spectrum of "normal" or is my reference sample size too small..)?

3.) Do I have enough statistical power to extract meaningful pathways from my biological system or do I see less significant features than what was expected? Is my gene set database used in the unsupervised analysis too large (correction for multiple testing)? Are too many pathways removed from the analysis due to the gene filtering?

4.) Underlying genes: Once my significant pathways were identified, can I describe the changes of the underlying genes and can I confirm these changes in other samples of the same condition?

Pro and Cons of tuning CellSpectra parameters:

I Gene filtering Pro: + Increases robustness of results + Highly recommended for the identification of underlying genes

Con: - Some pathways might be analyzed in one query but not the other (total pathway numbers cannot be compared) - For sparse cell types many pathways of potential interest might be removed from the analysis - Trends towards less significant features

II CHISQ.MAX Pro (= Decrease): + Increases robustness of results + Individual genes cannot drive the dyscoordination

Con: - Less power for detecting changes - Some changes might be driven by relatively few genes, but can still be biologically relevant

III Pseudoreplicates Pro: + Was designed for comparing biopsies of small cell numbers to references with larger cell numbers (considering more cells for the estimation of V1) + Adds the factor of sampling bias to the reference + Increases the power of the reference in cases of relatively few refsamples but relatively large cell numbers

Con: - Can introduce false homogeneity within the reference (inflates the number of significant results) - Especially in cases of relatively small reference samples compared to the query or a very small number of reference samples

```r
end.time <- Sys.time()

execution_time <- end.time - start.time

print(execution_time)
```

```
## Time difference of 17.23615 secs
```

```r
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS 15.1.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] KEGGREST_1.38.0    CellSpectra_0.1.0  Seurat_5.0.3       SeuratObject_5.0.1
## [5] sp_2.1-3           dplyr_1.1.4        Matrix_1.6-5
##
## loaded via a namespace (and not attached):
##   [1] Rtsne_0.17             colorspace_2.1-0       deldir_2.0-4
##   [4] ellipsis_0.3.2         ggridges_0.5.6         XVector_0.38.0
##   [7] RcppHNSW_0.6.0         rstudioapi_0.14        spatstat.data_3.0-4
##  [10] leiden_0.4.3.1         listenv_0.9.1          ggrepel_0.9.3
##  [13] RSpectra_0.16-1        fansi_1.0.6            codetools_0.2-19
##  [16] splines_4.2.2          knitr_1.45             polyclip_1.10-6
##  [19] spam_2.10-0            jsonlite_1.8.4         ica_1.0-3
##  [22] cluster_2.1.4          png_0.1-8              uwot_0.1.14
##  [25] shiny_1.8.0            sctransform_0.4.1      spatstat.sparse_3.0-3
##  [28] compiler_4.2.2         httr_1.4.5             fastmap_1.1.1
##  [31] lazyeval_0.2.2         cli_3.6.3              later_1.3.0
##  [34] htmltools_0.5.4        tools_4.2.2            igraph_1.3.5
##  [37] dotCall64_1.1-1        GenomeInfoDbData_1.2.9 gtable_0.3.1
##  [40] glue_1.8.0             RANN_2.6.1             reshape2_1.4.4
##  [43] Rcpp_1.0.10            scattermore_1.2        Biostrings_2.66.0
##  [46] vctrs_0.6.5            nlme_3.1-164           spatstat.explore_3.2-7
##  [49] progressr_0.14.0       lmtest_0.9-40          spatstat.random_3.2-3
##  [52] xfun_0.42              stringr_1.5.1          globals_0.16.3
##  [55] mime_0.12              miniUI_0.1.1.1         lifecycle_1.0.4
##  [58] irlba_2.3.5.1          goftest_1.2-3          future_1.33.2
##  [61] zlibbioc_1.44.0        MASS_7.3-58.2          zoo_1.8-12
##  [64] scales_1.3.0           promises_1.2.0.1       spatstat.utils_3.0-4
##  [67] parallel_4.2.2         RColorBrewer_1.1-3     yaml_2.3.7
##  [70] reticulate_1.34.0      pbapply_1.7-2          gridExtra_2.3
##  [73] ggplot2_3.5.0          stringi_1.8.4          highr_0.10
##  [76] S4Vectors_0.36.1       fastDummies_1.7.3      BiocGenerics_0.44.0
##  [79] GenomeInfoDb_1.34.9    bitops_1.0-7           rlang_1.1.4
##  [82] pkgconfig_2.0.3        matrixStats_0.63.0     evaluate_0.20
##  [85] lattice_0.20-45        ROCR_1.0-11            purrr_1.0.2
##  [88] tensor_1.5             patchwork_1.2.0        htmlwidgets_1.6.1
##  [91] cowplot_1.1.3          tidyselect_1.2.1       parallelly_1.37.1
##  [94] RcppAnnoy_0.0.20       plyr_1.8.8             magrittr_2.0.3
##  [97] R6_2.5.1               IRanges_2.32.0         generics_0.1.3
## [100] withr_3.0.2            pillar_1.9.0           fitdistrplus_1.1-11
## [103] RCurl_1.98-1.10        survival_3.5-0         abind_1.4-5
## [106] tibble_3.2.1           future.apply_1.11.1    crayon_1.5.2
## [109] KernSmooth_2.23-20     utf8_1.2.4             spatstat.geom_3.2-9
## [112] plotly_4.10.3          rmarkdown_2.20         grid_4.2.2
## [115] data.table_1.14.8      digest_0.6.31          xtable_1.8-4
## [118] tidyr_1.3.1            httpuv_1.6.14          stats4_4.2.2
## [121] munsell_0.5.0          viridisLite_0.4.1
```