

11-1-Nichenet-Control

Joan Abinet

2025-06-26 11:51:12 +0200

Contents

Introduction	2
Loading packages	2
Loading Annotated gCap	2
Loading Neutro	2
Define expressed genes in Gcap and neutro	2
Define expressed ligands and receptors	3
Load the ligand-target model	3
Load the gene set of interest and background of genes	3
Define potential ligand	4
Perform NicheNet's ligand activity analysis on the gene set of interest	4
Infer target genes of top-ranked ligands and visualize in a circos plot	5
Filter low score link	5
Prepare the circos visualization	7

Introduction

The NicheNet package was used to investigate ligand–target interactions. gCap endothelial cells (ECs) were defined as sender cells, and neutrophils as receiver cells. Two gene sets of interest were used for the analysis: one consisting of genes significantly upregulated in the PBS control condition, and the other consisting of genes upregulated in the MarNeu-depleted condition. Ligand prioritization was performed separately for each gene set to identify ligands in sender cells most likely to regulate the observed gene expression in receiver cells. In both cases, the top predicted ligand–target interactions were visualized using circos plots.

The following code was performed based on the vignette available in <https://github.com/saeyslab/nichennetr/blob/master/vignettes/circos.md>

```
library(lr_network)
readRDS(url("https://zenodo.org/record/7074291/files/lr_network_human_21122021.rds"))
ligand_target_matrix <- readRDS(url("https://zenodo.org/record/7074291/files/ligand_target_matrix_nsga2r_final.rds"))
```

Loading packages

```
suppressMessages({

  library(dplyr)
  library(Seurat)
  library(patchwork)
  library(ggplot2)
  library(nichennetr)
  library(tidyverse)
  library(circlize)

})
```

Loading Annotated gCap

```
endothelial_cells <- readRDS("../02-Clustering_Endo/endothelial_cells_annotated.rds")
Capillary_cells <- subset(endothelial_cells, CellType %in% "1 general capillaries")
Capillary_cells <- subset(Capillary_cells, Condition %in% "PBS")
```

Loading Neutro

```
neutro_depletion <- readRDS("../06-Neutrophils_Processing/neutro_depleted_Part1.rds")
```

Define expressed genes in Gcap and neutro

```
expression_endo = Capillary_cells@assays$RNA@counts
sample_info_endo = Capillary_cells@meta.data
expression_endo <- as.matrix(expression_endo)
expression_endo <- t(expression_endo)

expression_neutro = neutro_depletion@assays$RNA@counts
sample_info_neutro = neutro_depletion@meta.data
```

```
expression_neutro <- as.matrix(expression_neutro)
expression_neutro <- t(expression_neutro)
```

Define expressed ligands and receptors

```
Endo_ids <- CellsByIdentities(object = Capillary_cells)$`1 general capillaries`

neutro_ids <- CellsByIdentities(object = neutro_depletion)
neutro_ids <- unlist(neutro_ids)

expressed_genes_Endo = expression_endo[Endo_ids, ] %>%
  apply(2, function(x) {
    10 * (2^x - 1)
  }) %>%
  apply(2, function(x) {
    log2(mean(x) + 1)
  }) %>%
  .[. >= 4] %>%
  names()

expressed_genes_neutro = expression_neutro[neutro_ids, ] %>%
  apply(2, function(x) {
    10 * (2^x - 1)
  }) %>%
  apply(2, function(x) {
    log2(mean(x) + 1)
  }) %>%
  .[. >= 4] %>%
  names()
```

Load the ligand-target model

```
# adapt to the file path of your own system
ligand_target_matrix <- readRDS(file = "/mnt/Data/NicheNet_database/mouse_ligand_target_matrix.Rds")
```

Load the gene set of interest and background of genes

```
DE_PBS_treated = FindMarkers(object = neutro_depletion, ident.1 = "PBS",
  ident.2 = c("acxcr2", "aLy6G"), min.pct = 0.1, group.by = "orig.ident") %>%
  rownames_to_column("gene")
DE_PBS_treated <- DE_PBS_treated[order(abs(DE_PBS_treated$avg_log2FC),
  decreasing = T), ]
DE_PBS_treated <- DE_PBS_treated[DE_PBS_treated$avg_log2FC >= 0, ]

geneset <- DE_PBS_treated$gene
geneset <- unique(geneset)

background_expressed_genes = expressed_genes_neutro %>%
  .[. %in% rownames(ligand_target_matrix)]
```

Define potential ligand

```
# adapt to the file path of your own system
lr_network <- readRDS(file = "/mnt/Data/NicheNet_database/mouse_lr_network.Rds")

ligands = lr_network %>%
  pull(from) %>%
  unique()
expressed_ligands_endo = intersect(ligands, expressed_genes_Endo)

receptors = lr_network %>%
  pull(to) %>%
  unique()
expressed_receptors = intersect(receptors, expressed_genes_neutro)

potential_ligands = lr_network %>%
  filter(from %in% expressed_ligands_endo & to %in% expressed_receptors) %>%
  pull(from) %>%
  unique()
head(potential_ligands)
```

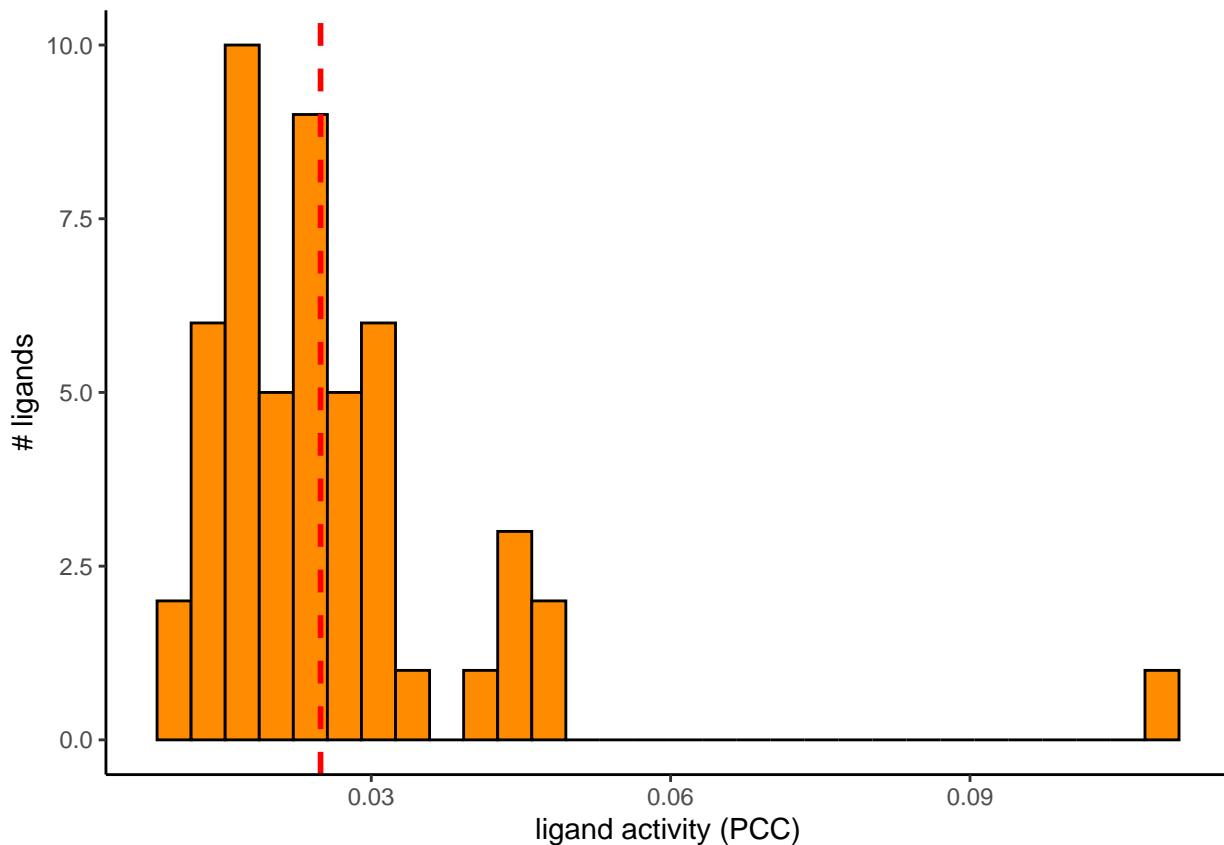
Perform NicheNet's ligand activity analysis on the gene set of interest

```
ligand_activities = predict_ligand_activities(geneset = geneset, background_expressed_genes = background
                                             ligand_target_matrix = ligand_target_matrix, potential_ligands = potential_ligands)

ligand_activities %>%
  arrange(-aupr_corrected)

best_upstream_ligands = ligand_activities %>%
  top_n(20, aupr_corrected) %>%
  arrange(-aupr_corrected) %>%
  pull(test_ligand)
head(best_upstream_ligands)

## [1] "Cxcl9"   "Cxcl12"  "Icam1"   "Cx3cl1"  "Adm"      "Sele"
ggplot(ligand_activities, aes(x = aupr_corrected)) + geom_histogram(color = "black",
  fill = "darkorange") + geom_vline(aes(xintercept = min(ligand_activities %>%
  top_n(20, aupr_corrected)) %>%
  pull(aupr_corrected))), color = "red", linetype = "dashed", size = 1) +
  labs(x = "ligand activity (PCC)", y = "# ligands") + theme_classic()
```



```
ligand_type_indication_df = tibble(ligand_type = c(rep("Endothelial-specific",
  times = best_upstream_ligands %>%
    length())),
  ligand = c(best_upstream_ligands))
```

Infer target genes of top-ranked ligands and visualize in a circos plot

```
active_ligand_target_links_df = best_upstream_ligands %>%
  lapply(get_weighted_ligand_target_links, geneset = geneset, ligand_target_matrix = ligand_target_ma
  n = 250) %>%
  bind_rows()

active_ligand_target_links_df = active_ligand_target_links_df %>%
  mutate(target_type = "neutro-specific") %>%
  inner_join(ligand_type_indication_df)
```

Filter low score link

```
cutoff_include_all_ligands = active_ligand_target_links_df$weight %>%
  quantile(0.66)

active_ligand_target_links_df_circos = active_ligand_target_links_df %>%
  filter(weight > cutoff_include_all_ligands)
```

```

ligands_to_remove = setdiff(active_ligand_target_links_df$ligand %>%
    unique(), active_ligand_target_links_df_circos$ligand %>%
    unique())
targets_to_remove = setdiff(active_ligand_target_links_df$target %>%
    unique(), active_ligand_target_links_df_circos$target %>%
    unique())

circos_links = active_ligand_target_links_df %>%
    filter(!target %in% targets_to_remove & !ligand %in% ligands_to_remove)

circos_links <- active_ligand_target_links_df_circos

grid_col_ligand = c(`Endothelial-specific` = "orange")

grid_col_target = c(`neutro-specific` = "blue")

grid_col_tbl_ligand = tibble(ligand_type = grid_col_ligand %>%
    names(), color_ligand_type = grid_col_ligand)
grid_col_tbl_target = tibble(target_type = grid_col_target %>%
    names(), color_target_type = grid_col_target)

circos_links = circos_links %>%
    mutate(ligand = paste(ligand, " "))
circos_links = circos_links %>%
    inner_join(grid_col_tbl_ligand) %>%
    inner_join(grid_col_tbl_target)
links_circle = circos_links %>%
    select(ligand, target, weight)

ligand_color = circos_links %>%
    distinct(ligand, color_ligand_type)
grid_ligand_color = ligand_color$color_ligand_type %>%
    set_names(ligand_color$ligand)
target_color = circos_links %>%
    distinct(target, color_target_type)
grid_target_color = target_color$color_target_type %>%
    set_names(target_color$target)

grid_col = c(grid_ligand_color, grid_target_color)

# give the option that links in the circos plot will be
# transparant ~ ligand-target potential score
transparency = circos_links %>%
    mutate(weight = (weight - min(weight))/(max(weight) - min(weight))) %>%
    mutate(transparency = 1 - weight) %>%
    .$transparency

target_order = circos_links$target %>%
    unique()
ligand_order = best_upstream_ligands %>%
    c(paste(., " ")) %>%
    intersect(circos_links$ligand)
order = c(ligand_order, target_order)

```

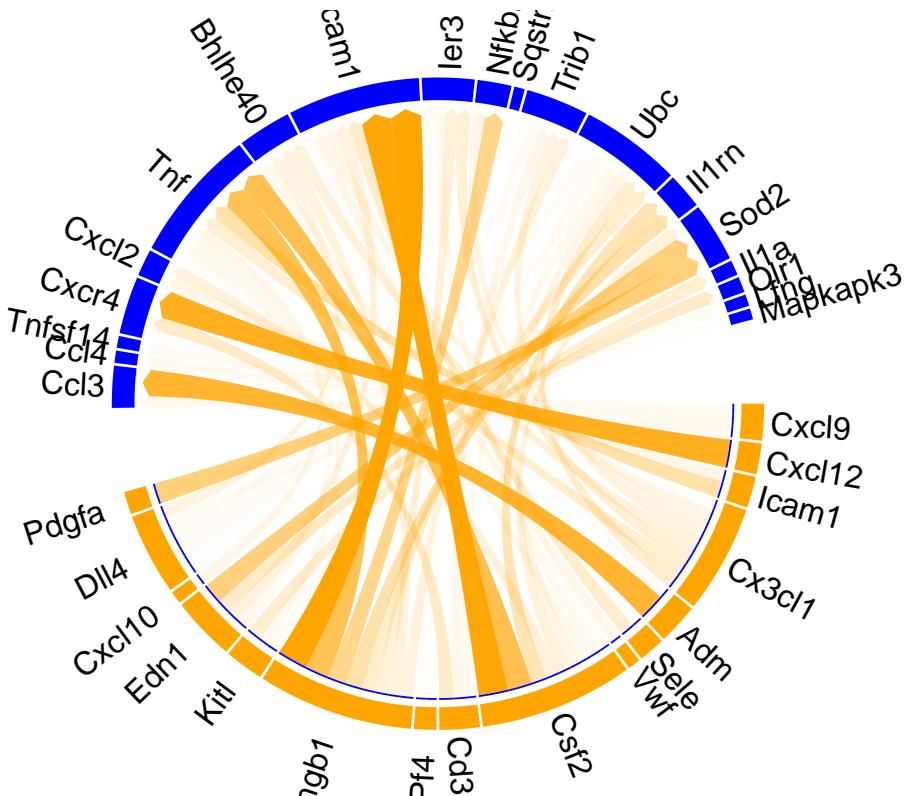
Prepare the circos visualization

```
width_same_cell_same_ligand_type = 0.5
width_different_cell = 6
width_ligand_target = 15
width_same_cell_same_target_type = 0.5

gaps = c(
  # width_ligand_target,
  rep(width_same_cell_same_ligand_type, times = (circos_links %>% filter(ligand_type == "Endothelial-specific") %>% nrow)),
  width_ligand_target,
  rep(width_same_cell_same_target_type, times = (circos_links %>% filter(target_type == "neutro-specific") %>% nrow)),
  width_ligand_target
)

# pdf('CP_Endo_Neutro_PBS.pdf', width = 8, height = 8) #
# measurement in inches

circos.par(gap.degree = gaps)
chordDiagram(links_circle, directional = 1, order = order, link.sort = TRUE,
  link.decreasing = FALSE, grid.col = grid_col, transparency = transparency,
  diffHeight = 0.005, direction.type = c("diffHeight", "arrows"),
  link.arr.type = "big.arrow", link.visible = links_circle$weight >=
    cutoff_include_all_ligands, annotationTrack = "grid", preAllocateTracks = list(track.height = 0))
# we go back to the first track and customize sector labels
circos.track(track.index = 1, panel.fun = function(x, y) {
  circos.text(CELL_META$xcenter, CELL_META$ylim[1], CELL_META$sector.index,
    facing = "clockwise", niceFacing = TRUE, adj = c(0, 0.55),
    cex = 1)
}, bg.border = NA)
```



```

# dev.off()

sessionInfo()

## R version 4.3.3 (2024-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 24.04.2 LTS
##
## Matrix products: default
## BLAS:    /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK:  /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblas-p0.3.26.so;  LAPACK version 3.12.0
##
## locale:
## [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
## [3] LC_TIME=fr_BE.UTF-8          LC_COLLATE=en_US.UTF-8
## [5] LC_MONETARY=fr_BE.UTF-8      LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=fr_BE.UTF-8          LC_NAME=C
## [9] LC_ADDRESS=C                  LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_BE.UTF-8   LC_IDENTIFICATION=C
##
## time zone: Europe/Brussels
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats      graphics   grDevices  utils      datasets   methods    base
##
## other attached packages:
## [1] circlize_0.4.15    lubridate_1.9.2    forcats_1.0.0     stringr_1.5.0
## [5] purrrr_1.0.1       readr_2.1.4       tidyverse_3.2.1

```

```

## [ 9] tidyverse_2.0.0      nichenetr_2.0.1      ggplot2_3.4.2       patchwork_1.1.2
## [13] SeuratObject_4.1.3   Seurat_4.3.0       dplyr_1.1.2
##
## loaded via a namespace (and not attached):
## [ 1] RcppAnnoy_0.0.21     splines_4.3.3      later_1.3.1
## [ 4] bitops_1.0-7         polyclip_1.10-4    hardhat_1.3.0
## [ 7] pROC_1.18.4          rpart_4.1.23      lifecycle_1.0.3
## [10] doParallel_1.0.17    globals_0.16.2     lattice_0.22-5
## [13] MASS_7.3-60.0.1      backports_1.4.1    magrittr_2.0.3
## [16] limma_3.56.2          Hmisc_5.1-0       plotly_4.10.2
## [19] rmarkdown_2.23        yaml_2.3.7       httpuv_1.6.11
## [22] sctransform_0.3.5    spam_2.9-1       sp_2.2-0
## [25] spatstat.sparse_3.0-2 reticulate_1.30    cowplot_1.1.1
## [28] pbapply_1.7-2        RColorBrewer_1.1-3 abind_1.4-5
## [31] Rtsne_0.16            BiocGenerics_0.46.0 nnet_7.3-19
## [34] tweenr_2.0.2          ipred_0.9-14      lava_1.7.2.1
## [37] IRanges_2.34.0        S4Vectors_0.38.1   ggrepel_0.9.3
## [40] irlba_2.3.5.1        listenv_0.9.0     spatstat.utils_3.0-3
## [43] goftest_1.2-3         spatstat.random_3.1-5 fitdistrplus_1.1-11
## [46] parallelly_1.36.0     leiden_0.4.3      codetools_0.2-19
## [49] ggforce_0.4.1         shape_1.4.6       tidyselect_1.2.0
## [52] farver_2.1.1          base64enc_0.1-3   matrixStats_1.0.0
## [55] stats4_4.3.3          spatstat.explore_3.2-1 jsonlite_1.8.7
## [58] caret_6.0-94           GetoptLong_1.0.5   e1071_1.7-13
## [61] Formula_1.2-5          ellipsis_0.3.2    progressr_0.13.0
## [64] ggridges_0.5.4         survival_3.5-8   iterators_1.0.14
## [67] foreach_1.5.2          tools_4.3.3      ggnewscale_0.4.9
## [70] ica_1.0-3              Rcpp_1.0.11      glue_1.6.2
## [73] prodlim_2023.03.31    gridExtra_2.3    xfun_0.39
## [76] withr_2.5.0            formatR_1.14     fastmap_1.1.1
## [79] fansi_1.0.4             caTools_1.18.2   digest_0.6.33
## [82] timechange_0.2.0        R6_2.5.1        mime_0.12
## [85] colorspace_2.1-0        scattermore_1.2  tensor_1.5
## [88] spatstat.data_3.0-1     DiagrammeR_1.0.10 utf8_1.2.3
## [91] generics_0.1.3          data.table_1.14.8 recipes_1.0.7
## [94] class_7.3-22            httr_1.4.6       htmlwidgets_1.6.2
## [97] uwot_0.1.16             ModelMetrics_1.2.2.2 pkgconfig_2.0.3
## [100] gtable_0.3.3          timeDate_4022.108 ComplexHeatmap_2.16.0
## [103] lmtest_0.9-40          shadowtext_0.1.2  htmltools_0.5.5
## [106] dotCall164_1.0-2       clue_0.3-64      scales_1.2.1
## [109] png_0.1-8              gower_1.0.1     knitr_1.43
## [112] rstudioapi_0.14        tzdb_0.4.0      reshape2_1.4.4
## [115] rjson_0.2.21            checkmate_2.2.0  visNetwork_2.1.2
## [118] nlme_3.1-164            proxy_0.4-27    zoo_1.8-12
## [121] GlobalOptions_0.1.2     KernSmooth_2.23-22 parallel_4.3.3
## [124] miniUI_0.1.1.1          foreign_0.8-86   pillar_1.9.0
## [127] grid_4.3.3               vctrs_0.6.3     RANN_2.6.1
## [130] randomForest_4.7-1.1    promises_1.2.0.1 xtable_1.8-4
## [133] cluster_2.1.6            htmlTable_2.4.1  evaluate_0.21
## [136] cli_3.6.1                compiler_4.3.3   rlang_1.1.1
## [139] crayon_1.5.2             future.apply_1.11.0 labeling_0.4.2
## [142] fdrtool_1.2.17            plyr_1.8.8      stringi_1.8.4
## [145] viridisLite_0.4.2        deldir_1.0-9     munsell_0.5.0
## [148] lazyeval_0.2.2            spatstat.geom_3.2-4 Matrix_1.6-1

```

```
## [151] hms_1.1.3          future_1.33.0        shiny_1.7.4.1
## [154] highr_0.10          ROCR_1.0-11          igraph_1.5.0.1
```