A lung Tgf-beta-signaling-mediated endothelial-interstitial macrophage axis prevents age-related abnormalities

6-NicheNet analysis

Rendered 2024-12-20 09:40:17 +0100

Abstract

Lung interstitial macrophages (IMs) are monocyte-derived parenchymal macrophages whose homeostatic and tissue-supportive functions remain unclear. While recent progress has been made about the diversity and transcriptional regulation of lung IMs, the microenvironmental signals responsible for their development from monocytes and for their functional specification remain unidentified. Here we found, in mice, that lung endothelial cell-derived Tgf-beta1 specifically triggered a core Tgf-beta receptor-dependent IM signature in bone marrow-derived monocytes and macrophages (Macs). In vivo, myeloid-specific ablation of Tgf-beta receptor signaling severely impaired monocyte-to-IM development, resulting in the accumulation of perivascular monocytes, decreased IM numbers and a loss of IM-intrinsic identity. Of note, monocyte-to-IM development was similarly impaired in the absence of endothelial-specific Tgf-beta1. Functionally, lungs from mice selectively lacking Tgf-beta receptor in IMs exhibited spatial changes in monocyte and IM niche occupancies, a severe disruption in their immunoregulatory environment, and prematurely developed fibrosis, hyperinflation, increased compliance and decreased elastance, changes classically associated with aging. Our work identifies a novel endothelial-IM axis involving Tgf-beta1 - Tgf-beta receptor interactions that shapes IM development and identity and thereby sustains lung tissue integrity, thus providing foundations for IM-targeted interventions in the context of lung aging and other chronic inflammatory disorders.

Contents

1 Description

In this study, we use published scRNAseq data *(1)* and apply NicheNet *(2)* to predict which ligands expressed by Endothelial cells and effected in Classical monocytes, MI_CD206, MI_MHCII are most likely to have induced the differential expression in the differentiation from Classical monocytes to MI_CD206, MI_MHCII in steady-state. The reference of the codes used in this analysis is here: https://github.com/saeyslab/nichenetr/blob/master/vignettes/ligand_activity_geneset.md.

For the genes implicated in the differentiation, we use the genes differentially expressed from Classical monocytes to MI_CD206, MI_MHCII. The method to calculate the differential expression is here the standard Seurat Wilcoxon test *(3)*.

2 Prepare NicheNet analysis

2.1 Load required packages

2.1.1 Load Packages and data:

```
suppressMessages(library(nichenetr))
suppressMessages(library(Seurat))
suppressMessages(library(tidyverse))

so <- readRDS("./all_ss.seuratObject.rds")
Idents(so) <- "cell.type2"
```

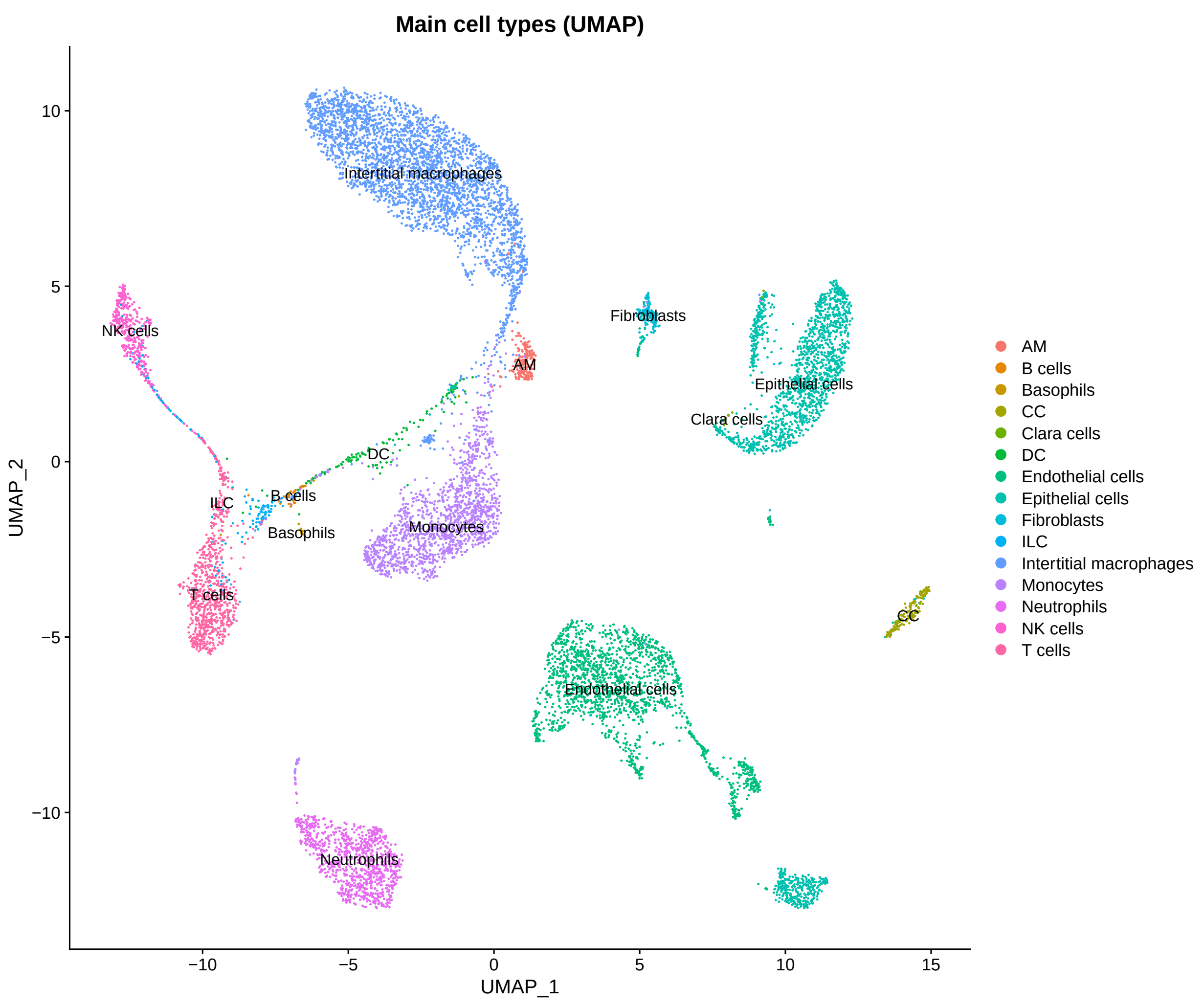
Visualize which cell populations are present:

```
so@meta.data$cell.type1 %>%
    table()
```

```
## .
##                     AM                   B cells                Basophils
##                    191                        27                       28
##                     CC               Clara cells                       DC
##                    169                        10                      128
##      Endothelial cells         Epithelial cells              Fibroblasts
##                   1804                      1826                      168
##                    ILC Intertitial macrophages                 Monocytes
##                    144                      3319                     1502
##            Neutrophils                  NK cells                  T cells
##                    968                       404                      829
```

Visualize the data to see to main cell types.

```
DimPlot(so, group.by = "cell.type1", label = TRUE) + ggtitle("Main␣cell␣
   types␣(UMAP)")
```



Load mouse matrix and tables:

```
ligand_target_matrix <- readRDS(file = "/mnt/Data/NicheNet_database/mouse_
   ligand_target_matrix.Rds")
weighted_networks_lr <- readRDS(file = "/mnt/Data/NicheNet_database/mouse_
   weighted_networks_lr.Rds")
lr_network <- readRDS(file = "/mnt/Data/NicheNet_database/mouse_lr_network
   .Rds")
```

3 Perform the NicheNet analysis

3.1 1. Define a "sender/niche" cell population and a "receiver/target" cell population present in your expression data and determine which genes are expressed in both populations

In this case study, the receiver cell population is Classical monocytes, MI_CD206, MI_MHCII, whereas the sender cell populations are Endothelial cells. We will consider a gene to be expressed when it is expressed in at least 10% of cells in one cluster.

```r
source("~/Desktop/velocyto/Script/get_expressed_genes.R")
## receiver
Idents(so) <- "cell.type3"
receiver = receiver.cells
expressed_genes_receiver = get_expressed_genes(receiver, so, pct = 0.1)

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

## sender
sender_celltypes = sender.cells

list_expressed_genes_sender = sender_celltypes %>%
    unique() %>%
    lapply(get_expressed_genes, so, 0.1)
expressed_genes_sender = list_expressed_genes_sender %>%
    unlist() %>%
    unique()
```

3.2 2. Define a gene set of interest: these are the genes in the "receiver/target" cell population that are potentially affected by ligands expressed by interacting cells (e.g. genes differentially expressed upon cell-cell interaction)

Here, the gene set of interest are the genes differentially expressed from Classical monocytes to MI_CD206, MI_MHCII. The method to calculate the differential expression is here the standard Seurat Wilcoxon test.

DE genes in reveiver cells Classical monocytes, MI_CD206, MI_MHCII:

```r
# seurat_obj_receiver= subset(so, idents = receiver) # we will not compare
# comditions within one receiver.
seurat_obj_receiver <- so
# seurat_obj_receiver = SetIdent(seurat_obj_receiver, value =
# seurat_obj_receiver[['treatment']])


condition_oi = celltype.to
condition_reference = celltype.from

DE_table_receiver = FindMarkers(object = seurat_obj_receiver, ident.1 =
   condition_oi,
    ident.2 = condition_reference, min.pct = 0.1) %>%
    rownames_to_column("gene")

geneset_oi = DE_table_receiver %>%
    filter(p_val_adj <= 0.05 & abs(avg_log2FC) >= 0.25) %>%
    pull(gene)
geneset_oi = geneset_oi %>%
    .[. %in% rownames(ligand_target_matrix)]
```

3.3 3. Define a set of potential ligands: these are ligands that are expressed by the "sender/niche" cell population and bind a (putative) receptor expressed by the "receiver/target" population

Top potential ligands:

```
ligands = lr_network %>%
    pull(from) %>%
    unique()
receptors = lr_network %>%
    pull(to) %>%
    unique()

expressed_ligands = intersect(ligands, expressed_genes_sender)
expressed_receptors = intersect(receptors, expressed_genes_receiver)

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

3.4 4. Perform NicheNet ligand activity analysis: rank the potential ligands based on the presence of their target genes in the gene set of interest (compared to the background set of genes)

The ligand activity table:

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

ligand_activities = ligand_activities %>%
    arrange(-pearson) %>%
    mutate(rank = rank(desc(pearson)))
```

The different ligand activity measures (auroc, aupr, pearson correlation coefficient) are a measure for how well a ligand can predict the observed differentially expressed genes compared to the background of expressed genes. In our validation study (the author of the md), we showed that the pearson correlation coefficient between a ligand's target predictions and the observed transcriptional response was the most informative measure to define ligand activity. Therefore, NicheNet ranks the ligands based on their pearson correlation coefficient.

The number of top-ranked ligands that are further used to predict active target genes and construct an active ligand-receptor network is here 20.

```
best_upstream_ligands = ligand_activities %>%
    top_n(20, pearson) %>%
    arrange(-pearson) %>%
    pull(test_ligand) %>%
    unique()
```

These ligands are expressed by one or more of the input sender cells. To see which cell population expresses which of these top-ranked ligands:

```
DotPlot(so, features = best_upstream_ligands %>%
    rev(), cols = "RdYlBu") + RotatedAxis() + ggtitle(paste("Top␣20",
        paste(sender_celltypes,
    collapse = "␣"), "ligands␣targeting", paste(receiver.cells, collapse =
        "␣")))
```

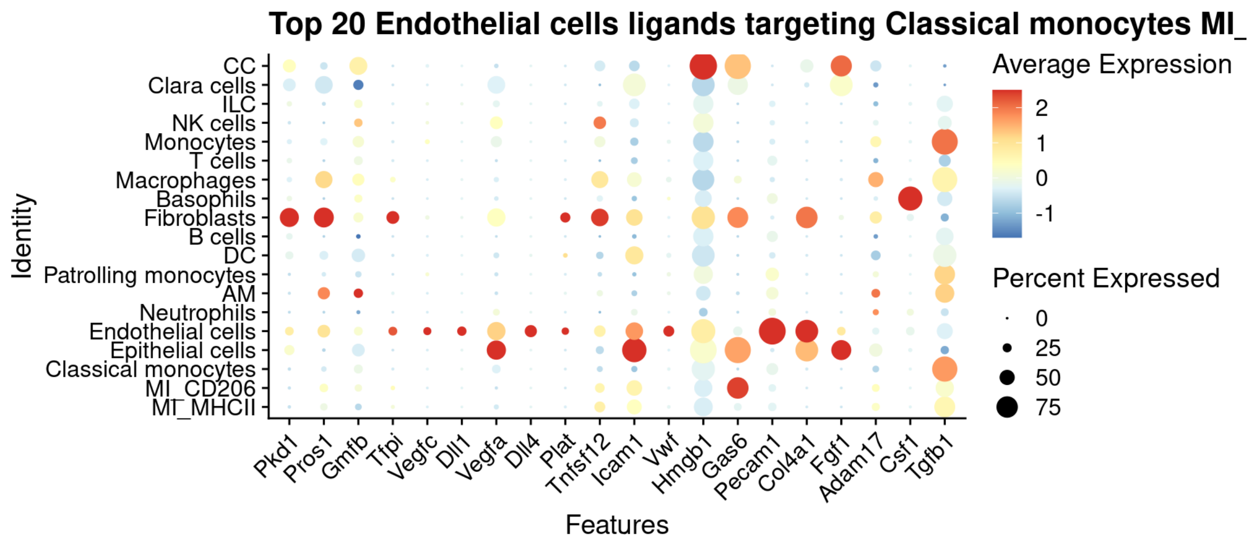


Figure 1: 1-top-ranked ligands

3.5 5. Infer receptors and top-predicted target genes of ligands that are top-ranked in the ligand activity analysis

3.5.1 Active target gene inference

```
active_ligand_target_links_df = best_upstream_ligands %>%
    lapply(get_weighted_ligand_target_links, geneset = geneset_oi, ligand_
        target_matrix = ligand_target_matrix,
         n = 200) %>%
    bind_rows() %>%
    drop_na()

active_ligand_target_links = prepare_ligand_target_visualization(ligand_
   target_df = active_ligand_target_links_df,
    ligand_target_matrix = ligand_target_matrix, cutoff = 0.33)

order_ligands = intersect(best_upstream_ligands, colnames(active_ligand_
   target_links)) %>%
    rev() %>%
    make.names()
order_targets = active_ligand_target_links_df$target %>%
    unique() %>%
    intersect(rownames(active_ligand_target_links)) %>%
    make.names()
```

```r
rownames(active_ligand_target_links) = rownames(active_ligand_target_links) %>%
    make.names()  # make.names() for heatmap visualization of genes like H2-T23
colnames(active_ligand_target_links) = colnames(active_ligand_target_links) %>%
    make.names()  # make.names() for heatmap visualization of genes like H2-T23

vis_ligand_target = active_ligand_target_links[order_targets, order_ligands] %>%
    t()
```

```r
p_ligand_target_network = vis_ligand_target %>%
    make_heatmap_ggplot("Prioritized ligands", "Predicted target genes", color = "purple",
      legend_position = "top", x_axis_position = "top", legend_title = "Regulatory potential") +
    theme(axis.text.x = element_text(face = "italic")) + scale_fill_gradient2(low = "whitesmoke",
    high = "purple", breaks = c(0, 0.006, 0.012))
p_ligand_target_network
```

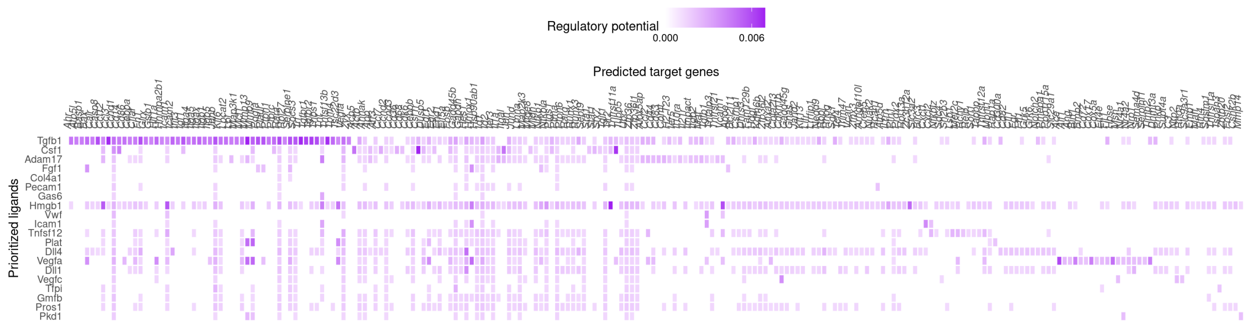


Figure 2: 2-Active target gene inference

3.5.2 Receptors of top-ranked ligands

```r
lr_network_top = lr_network %>%
    filter(from %in% best_upstream_ligands & to %in% expressed_receptors) %>%
    distinct(from, to)
best_upstream_receptors = lr_network_top %>%
    pull(to) %>%
    unique()

lr_network_top_df_large = weighted_networks_lr %>%
    filter(from %in% best_upstream_ligands & to %in% best_upstream_receptors)

lr_network_top_df = lr_network_top_df_large %>%
    spread("from", "weight", fill = 0)
lr_network_top_matrix = lr_network_top_df %>%
    select(-to) %>%
    as.matrix() %>%
    magrittr::set_rownames(lr_network_top_df$to)

dist_receptors = dist(lr_network_top_matrix, method = "binary")
hclust_receptors = hclust(dist_receptors, method = "ward.D2")
order_receptors = hclust_receptors$labels[hclust_receptors$order]

dist_ligands = dist(lr_network_top_matrix %>%
    t(), method = "binary")
hclust_ligands = hclust(dist_ligands, method = "ward.D2")
order_ligands_receptor = hclust_ligands$labels[hclust_ligands$order]

order_receptors = order_receptors %>%
    intersect(rownames(lr_network_top_matrix))
order_ligands_receptor = order_ligands_receptor %>%
    intersect(colnames(lr_network_top_matrix))

vis_ligand_receptor_network = lr_network_top_matrix[order_receptors, order
   _ligands_receptor]
rownames(vis_ligand_receptor_network) = order_receptors %>%
    make.names()
colnames(vis_ligand_receptor_network) = order_ligands_receptor %>%
    make.names()

p_ligand_receptor_network = vis_ligand_receptor_network %>%
    t() %>%
    make_heatmap_ggplot("Ligands", "Receptors", color = "mediumvioletred",
        x_axis_position = "top",
        legend_title = "Prior␣interaction␣potential")
p_ligand_receptor_network
```

3.5.3 Receptors of top-ranked ligands, but after considering only bona fide ligand-receptor interactions documented in literature and publicly available databases

```
lr_network_strict = lr_network %>%
    filter(database != "ppi_prediction_go" & database != "ppi_prediction")
          # remove these 2 predictions
ligands_bona_fide = lr_network_strict %>%
    pull(from) %>%
    unique()
receptors_bona_fide = lr_network_strict %>%
    pull(to) %>%
    unique()

lr_network_top_df_large_strict = lr_network_top_df_large %>%
    distinct(from, to) %>%
    inner_join(lr_network_strict, by = c("from", "to")) %>%
    distinct(from, to)
lr_network_top_df_large_strict = lr_network_top_df_large_strict %>%
```

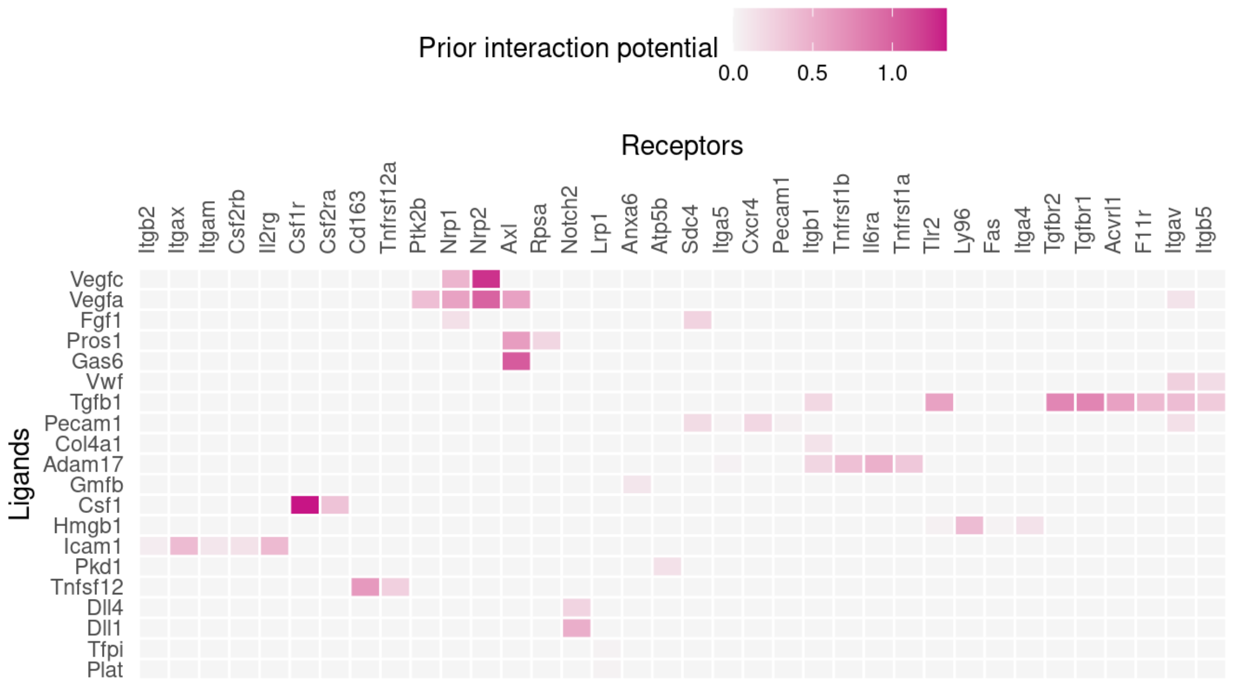


Figure 3: 3-Receptors of top-ranked ligands

```
    inner_join(lr_network_top_df_large, by = c("from", "to"))

lr_network_top_df_strict = lr_network_top_df_large_strict %>%
    spread("from", "weight", fill = 0)
lr_network_top_matrix_strict = lr_network_top_df_strict %>%
    select(-to) %>%
    as.matrix() %>%
    magrittr::set_rownames(lr_network_top_df_strict$to)

dist_receptors = dist(lr_network_top_matrix_strict, method = "binary")
hclust_receptors = hclust(dist_receptors, method = "ward.D2")
order_receptors = hclust_receptors$labels[hclust_receptors$order]

dist_ligands = dist(lr_network_top_matrix_strict %>%
    t(), method = "binary")
hclust_ligands = hclust(dist_ligands, method = "ward.D2")
order_ligands_receptor = hclust_ligands$labels[hclust_ligands$order]

order_receptors = order_receptors %>%
    intersect(rownames(lr_network_top_matrix_strict))
order_ligands_receptor = order_ligands_receptor %>%
    intersect(colnames(lr_network_top_matrix_strict))

vis_ligand_receptor_network_strict = lr_network_top_matrix_strict[order_
   receptors,
    order_ligands_receptor]
rownames(vis_ligand_receptor_network_strict) = order_receptors %>%
    make.names()
```

```
colnames(vis_ligand_receptor_network_strict) = order_ligands_receptor %>%
    make.names()

p_ligand_receptor_network_strict = vis_ligand_receptor_network_strict %>%
    t() %>%
    make_heatmap_ggplot("Ligands", "Receptors", color = "mediumvioletred",
        x_axis_position = "top",
        legend_title = "Prior␣interaction␣potential\n(bona␣fide)")
p_ligand_receptor_network_strict
```

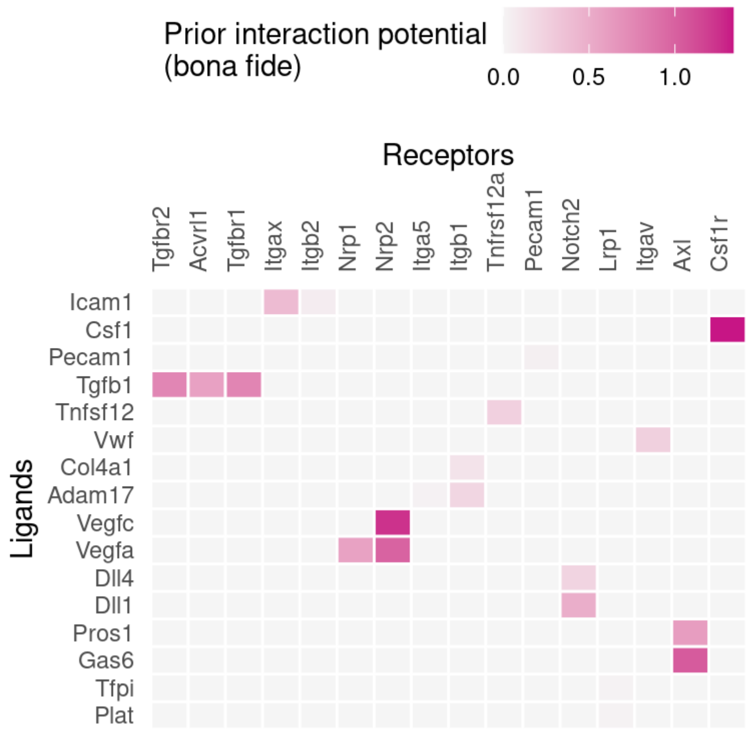


Figure 4: 4-bona fide ligand-receptor interactions

3.6 6) Summary visualizations of the NicheNet analysis

```
# combined heatmap: overlay ligand activities with target genesRemarks

ligand_pearson_matrix = ligand_activities %>%
    select(pearson) %>%
    as.matrix() %>%
    magrittr::set_rownames(ligand_activities$test_ligand)

rownames(ligand_pearson_matrix) = rownames(ligand_pearson_matrix) %>%
    make.names()
colnames(ligand_pearson_matrix) = colnames(ligand_pearson_matrix) %>%
    make.names()

vis_ligand_pearson = ligand_pearson_matrix[order_ligands, ] %>%
    as.matrix(ncol = 1) %>%
    magrittr::set_colnames("Pearson")
p_ligand_pearson = vis_ligand_pearson %>%
    make_heatmap_ggplot("Prioritized␣ligands", "Ligand␣activity", color = "darkorange",
     legend_position = "top", x_axis_position = "top", legend_title = "Pearson␣correlation␣coefficient\ntarget␣gene␣prediction␣ability)") +
    theme(legend.text = element_text(size = 9))

figures_without_legend = cowplot::plot_grid(p_ligand_pearson + theme(legend.position = "none",
    axis.ticks = element_blank()) + theme(axis.title.x = element_text()), p_ligand_target_network +
    theme(legend.position = "none", axis.ticks = element_blank()) + ylab(""), align = "hv",
    nrow = 1, rel_widths = c(ncol(vis_ligand_pearson) + 10, ncol(vis_ligand_target)))

legends = cowplot::plot_grid(ggpubr::as_ggplot(ggpubr::get_legend(p_ligand_pearson)),
    ggpubr::as_ggplot(ggpubr::get_legend(p_ligand_target_network)), nrow = 1, align = "h")

combined_plot = cowplot::plot_grid(figures_without_legend, legends, rel_heights = c(10,
    2), nrow = 2, align = "hv")
combined_plot
```

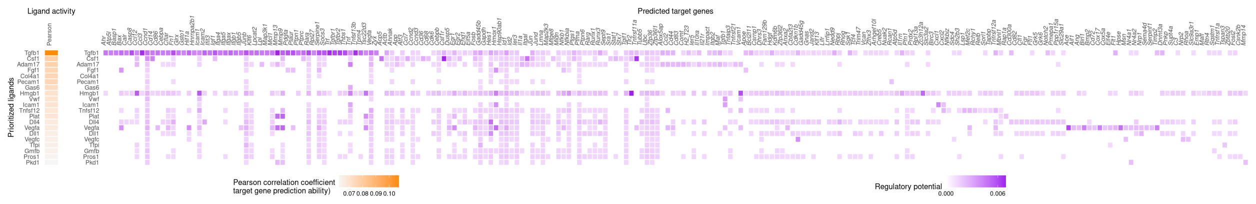


Figure 5: 5-summary visualisation

4 Session information

R sesssion:

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS 15.1.1
##
## 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 version 3.11.0
##
## 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/Paris
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.9.4    forcats_1.0.0      stringr_1.5.1      dplyr_1
   .1.4
##  [5] purrr_1.0.2        readr_2.1.5        tidyr_1.3.1        tibble_3
   .2.1
##  [9] ggplot2_3.5.1      tidyverse_2.0.0    Seurat_5.1.0
   SeuratObject_5.0.2
## [13] sp_2.1-4           nichenetr_2.1.0
##
## loaded via a namespace (and not attached):
##   [1] RcppAnnoy_0.0.22       splines_4.3.3          later_1.4.1
##   [4] bitops_1.0-9           polyclip_1.10-7        hardhat_1.4.0
##   [7] pROC_1.18.5            rpart_4.1.23           fastDummies_1.7.4
##  [10] lifecycle_1.0.4        doParallel_1.0.17      globals_0.16.3
##  [13] lattice_0.22-6         MASS_7.3-60.0.1        backports_1.5.0
##  [16] magrittr_2.0.3         limma_3.58.1           Hmisc_5.2-1
##  [19] plotly_4.10.4          rmarkdown_2.29         yaml_2.3.10
##  [22] httpuv_1.6.15          sctransform_0.4.1      spam_2.11-0
##  [25] spatstat.sparse_3.1-0  reticulate_1.40.0      cowplot_1.1.3
##  [28] pbapply_1.7-2          RColorBrewer_1.1-3     abind_1.4-8
##  [31] Rtsne_0.17             BiocGenerics_0.48.1    nnet_7.3-19
##  [34] tweenr_2.0.3           ipred_0.9-15           circlize_0.4.16
##  [37] lava_1.8.0             IRanges_2.36.0         S4Vectors_0.40.2
##  [40] ggrepel_0.9.6          irlba_2.3.5.1          listenv_0.9.1
##  [43] spatstat.utils_3.1-1   goftest_1.2-3          RSpectra_0.16-2
##  [46] spatstat.random_3.3-2  fitdistrplus_1.2-1     parallelly_1.40.1
##  [49] leiden_0.4.3.1         codetools_0.2-20       ggforce_0.4.2
##  [52] tidyselect_1.2.1       shape_1.4.6.1          farver_2.1.2
##  [55] base64enc_0.1-3        matrixStats_1.4.1      stats4_4.3.3
##  [58] spatstat.explore_3.3-3 jsonlite_1.8.9         caret_7.0-1
##  [61] GetoptLong_1.0.5       e1071_1.7-16           Formula_1.2-5
##  [64] progressr_0.15.1       ggridges_0.5.6         survival_3.7-0
##  [67] iterators_1.0.14       foreach_1.5.2          tools_4.3.3
##  [70] ggnewscale_0.5.0       ica_1.0-3              Rcpp_1.0.13-1
##  [73] glue_1.8.0             prodlim_2024.06.25     gridExtra_2.3
##  [76] xfun_0.49              withr_3.0.2            formatR_1.14
##  [79] fastmap_1.2.0          fansi_1.0.6            caTools_1.18.3
##  [82] digest_0.6.37          timechange_0.3.0       R6_2.5.1
##  [85] mime_0.12              colorspace_2.1-1       scattermore_1.2
##  [88] tensor_1.5             spatstat.data_3.1-4    DiagrammeR_1.0.11
##  [91] utf8_1.2.4             generics_0.1.3         data.table_1.16.4
##  [94] recipes_1.1.0          class_7.3-22           httr_1.4.7
##  [97] htmlwidgets_1.6.4      uwot_0.1.16            ModelMetrics_1
    .2.2.2
## [100] pkgconfig_2.0.3        gtable_0.3.6           timeDate_4041.110
## [103] ComplexHeatmap_2.18.0  lmtest_0.9-40          shadowtext_0.1.4
## [106] htmltools_0.5.8.1      dotCall64_1.2          clue_0.3-66
## [109] scales_1.3.0           png_0.1-8              gower_1.0.1
## [112] spatstat.univar_3.1-1  knitr_1.49             rstudioapi_0.17.1
## [115] tzdb_0.4.0             reshape2_1.4.4         rjson_0.2.23
## [118] checkmate_2.3.2        visNetwork_2.1.2       nlme_3.1-166
## [121] proxy_0.4-27           zoo_1.8-12             GlobalOptions_0.1.2
## [124] KernSmooth_2.23-24     parallel_4.3.3         miniUI_0.1.1.1
## [127] foreign_0.8-87         pillar_1.9.0           grid_4.3.3
## [130] vctrs_0.6.5            RANN_2.6.2             randomForest_4
    .7-1.2
## [133] promises_1.3.2         xtable_1.8-4           cluster_2.1.7
## [136] htmlTable_2.4.3        evaluate_1.0.1         cli_3.6.3
## [139] compiler_4.3.3         rlang_1.1.4            crayon_1.5.3
## [142] future.apply_1.11.3    labeling_0.4.3         fdrtool_1.2.18
## [145] plyr_1.8.9             stringi_1.8.4          viridisLite_0.4.2
## [148] deldir_2.0-4           munsell_0.5.1          lazyeval_0.2.2
## [151] spatstat.geom_3.3-4    Matrix_1.6-5           RcppHNSW_0.6.0
## [154] hms_1.1.3              patchwork_1.3.0        future_1.34.0
## [157] statmod_1.5.0          shiny_1.9.1            ROCR_1.0-11
## [160] igraph_2.1.2
```

References

1. J. Schyns, Q. Bai, C. Ruscitti, C. Radermecker, S. D. Schepper, S. Chakarov, F. Farnir, D. Pirottin, F. Ginhoux, G. Boeckxstaens, F. Bureau, T. Marichal, Non-classical tissue monocytes and two functionally distinct populations of interstitial macrophages populate the mouse lung. *Nature Communications* **10**, 3964 (2019).

2. R. Browaeys, W. Saelens, Y. Saeys, NicheNet: Modeling intercellular communication by linking ligands to target genes. *Nature Methods* (2019) (available at https://www.nature.com/articles/s41592-019-0667-5).

3. Y. Hao, T. Stuart, M. H. Kowalski, S. Choudhary, P. Hoffman, A. Hartman, A. Srivastava, G. Molla, S. Madad, C. Fernandez-Granda, R. Satija, Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nature Biotechnology* (2023), doi:10.1038/s41587-023-01767-y.