Interaction sources comparison: NicheNet VS Omnipath

Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 08/01/2020

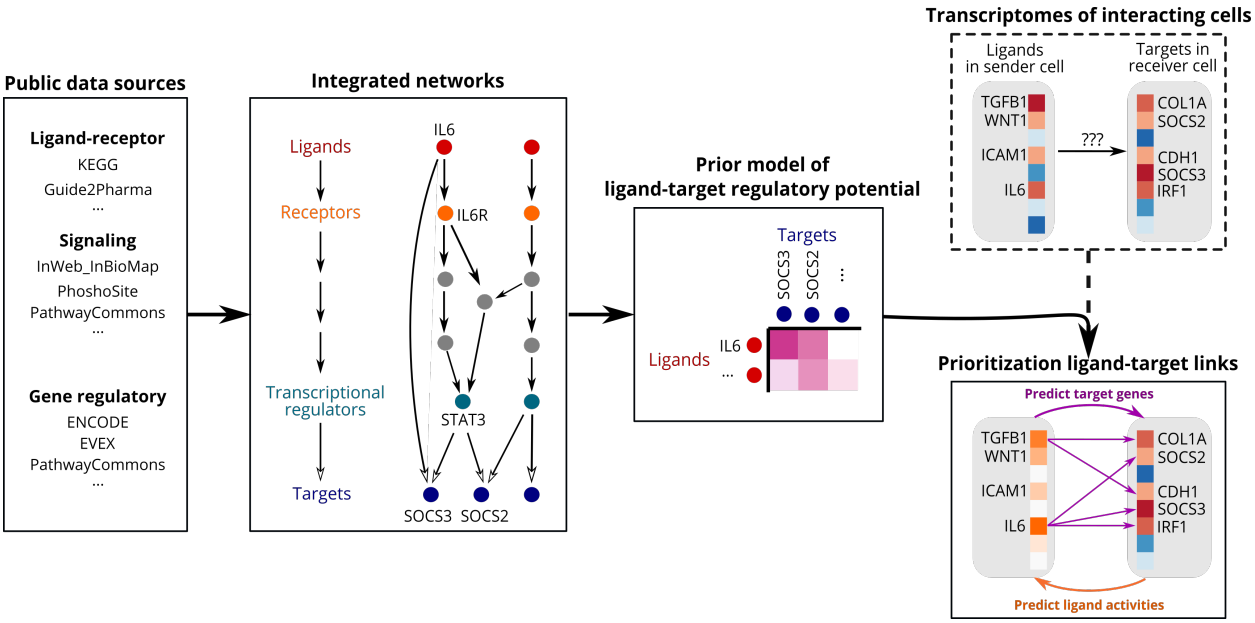
Abstract

This vignette shows a comparison between the protein interaction sources used in the NicheNet method and the ones available on Omnipath.

The NicheNet Method

NicheNet (https://github.com/saeyslab/nichenetr) is a method to predict ligand-target links between interacting cells by combining their data with prior knowledge on signaling and gene regulatory networks (Browaeys et al 2019). **NicheNet** has already been applied to predict upstream niche signals driving Kupffer cell differentiation (Bonnardel et al. 2019).

The figure below shows a graphical representation of the NicheNet workflow. Interactions inferred from several complementary ligand-receptor, signaling and gene regulatory data sources were aggregated in respective integrated networks from which ligand-target regulatory potential scores were calculated. This model of prior information on potential ligand-target links can then be used to infer active ligand-target links between interacting cells. NicheNet prioritizes ligands according to their activity (i.e., how well they predict observed changes in gene expression in the receiver cell) and looks for affected targets with high potential to be regulated by these prioritized ligands(Browaeys et al 2019).



You can find below the list of public resources used to generate the prior model of ligand-target regulatory potential.

Database	Source	Reference
cpdb	cpdb_interaction	Kamburov et al. (2013)
cpdb	cpdb_complex	Kamburov et al. (2013)
evex_expression	lr_evex_regulation_expression	Van Landeghem et al. (2012)
evex_expression	evex_regulation_expression	Van Landeghem et al. (2012)

Database	Source	Reference
evex_signaling	evex_catalysis	Van Landeghem et al. (2012)
evex_signaling	evex_regulation_other	Van Landeghem et al. (2012)
evex_signaling	evex_phosphorylation	Van Landeghem et al. (2012)
evex_signaling	evex_regulation_binding	Van Landeghem et al. (2012)
evex_signaling	evex_binding	Van Landeghem et al. (2012)
guide2pharmacology	pharmacology	Pawson et al. (2014)
harmonizome	harmonizome_KEA	Lachmann and Ma'ayan (2009)
harmonizome	harmonizome_PhosphoSite	Hornbeck et al. (2015)
harmonizome	harmonizome_kinase_substrate_predictions	Rouillard et al. (2016)
harmonizome	harmonizome_DEPOD	Duan et al. (2015)
harmonizome_gr	harmonizome_CHEA	Lachmann et al. (2010)
harmonizome_gr	harmonizome_ENCODE	Consortium (2004)
harmonizome_gr	harmonizome_JASPAR	Mathelier et al. (2014)
harmonizome_gr	harmonizome_TRANSFAC_CUR	Matys et al. (2006)
harmonizome_gr	harmonizome_TRANSFAC	Matys et al. (2006)
harmonizome_gr	harmonizome_MOTIFMAP	Xie et al. (2009)
harmonizome_gr	harmonizome_GEO_TF	Edgar et al. (2002)
harmonizome_gr	harmonizome_GEO_KINASE	Edgar et al. (2002)
harmonizome_gr	harmonizome_GEO_GENE	Edgar et al. (2002)
harmonizome_gr	harmonizome_MSIGDB_GENE	Subramanian et al. (2005)
HTRIDB	HTRIDB	Bovolenta et al. (2012)
inweb_inbiomap	inweb_inbio_interaction	Li et al. (2017)
inweb_inbiomap	inweb_inbio_interaction_pathway	Li et al. (2017)
inweb_inbiomap	inweb_inbio_pathway	Li et al. (2017)
kegg	kegg_cytokines	Kanehisa et al. (2016)
kegg	kegg_cams	Kanehisa et al. (2016)
kegg	kegg_neuroactive	Kanehisa et al. (2016)
kegg	kegg_ecm	Kanehisa et al. (2016)
omnipath	omnipath_directional	Türei et al. (2016)
omnipath	omnipath_undirectional	Türei et al. (2016)
ontogenet	ontogenet_coarse	Jojic et al. (2013)
pathwaycommons_expression	lr_pathwaycommons_controls_expression_of	Cerami et al. (2011)
pathwaycommons_expression	pathwaycommons_controls_expression_of	Cerami et al. (2011)
pathwaycommons_signaling	pathwaycommons_controls_phosphorylation_of	Cerami et al. (2011)
pathwaycommons_signaling	pathwaycommons_controls_state_change_of	Cerami et al. (2011)
pathwaycommons_signaling	pathwaycommons_catalysis_precedes	Cerami et al. (2011)
pathwaycommons_signaling	pathwaycommons_controls_transport_of	Cerami et al. (2011)
pathwaycommons_signaling	pathwaycommons_interacts_with	Cerami et al. (2011)
pathwaycommons_signaling	pathwaycommons_in_complex_with	Cerami et al. (2011)
ppi_prediction	ppi_lr	
ppi_prediction	ppi_l_bidir	
ppi_prediction	ppi_bidir_r	
ppi_prediction	ppi_bidir_bidir	
ppi_prediction_go	ppi_lr_go	
ppi_prediction_go	ppi_l_bidir_go	
ppi_prediction_go	ppi_bidir_r_go	

Database	Source	Reference
ppi_prediction_go	ppi_bidir_bidir_go	
ramilowski	ramilowski_known	Ramilowski et al. (2015)
regnetwork	regnetwork_source	Liu et al. (2015)
regnetwork	regnetwork_encode	Liu et al. (2015)
Remap	Remap_5	Griffon et al. (2015)
trrust	trrust	Han et al. (2015)
vinayagam	vinayagam_ppi	Vinayagam et al. (2011)

Omnipath Resources

As we can see in the previous figure, **NicheNet** used many different publically available resources to build a prior knowledge network. Their final integrated network is composed of three individual networks:

- A network of ligand-receptor interactions (Inter-cellular)
- A network of signaling interactions (Intra-cellular)
- A network of gene regulation (Intra-cellular)

The new versin of the **Omnipath** (http://omnipathdb.org/) database contains curated interactions belonging to these three categories. One can therefore build an integrated network equivalent to the one used in **NicheNet** by only fetching the **Omnipath** webserver. This can significantly ease the integration of different databases, each one of them storing data in distint formats and whose interaction show different levels of reliability.

We therefore here compare the interactions used in the **NicheNet** method with those freely available in the **Omnipath** database. You can find below three sections comparing each one of three different interaction categories.

Ligand-Receptor Interaction Network

We first load the ligand-receptor interactions used to build the **NicheNet** model.

```
## We first load the required libraries to run this script
library(OmnipathR)
library(dplyr)
library(VennDiagram)
library(readr)
```

```
## We read the data which are freely available via Zenodo and we display
## how they look like.
lr_Network_Nichenet <-
    readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_Network_Nichenet
## # A tibble: 12,651 x 4
##    from  to    source         database
##    <chr> <chr> <chr>          <chr>
##  1 CXCL1 CXCR2 kegg_cytokines kegg
##  2 CXCL2 CXCR2 kegg_cytokines kegg
##  3 CXCL3 CXCR2 kegg_cytokines kegg
##  4 CXCL5 CXCR2 kegg_cytokines kegg
##  5 PPBP  CXCR2 kegg_cytokines kegg
##  6 CXCL6 CXCR2 kegg_cytokines kegg
##  7 CXCL8 CXCR2 kegg_cytokines kegg
##  8 CXCL6 CXCR1 kegg_cytokines kegg
##  9 CXCL8 CXCR1 kegg_cytokines kegg
```

```
## 10 CXCL9 CXCR3 kegg_cytokines kegg
## # ... with 12,641 more rows
```

We show the total number of ligand-receptor interactions after removing duplicates:

```
lr_Network_Nichenet_Unique <- lr_Network_Nichenet %>%
    dplyr::distinct(from, to)
nrow(lr_Network_Nichenet_Unique)
## [1] 12019
```

Omnipath possess a dedicated dataset storing these type of interactions (*LigrecExtra*). We now fetch the **Omnipath** web service to get these interactions.

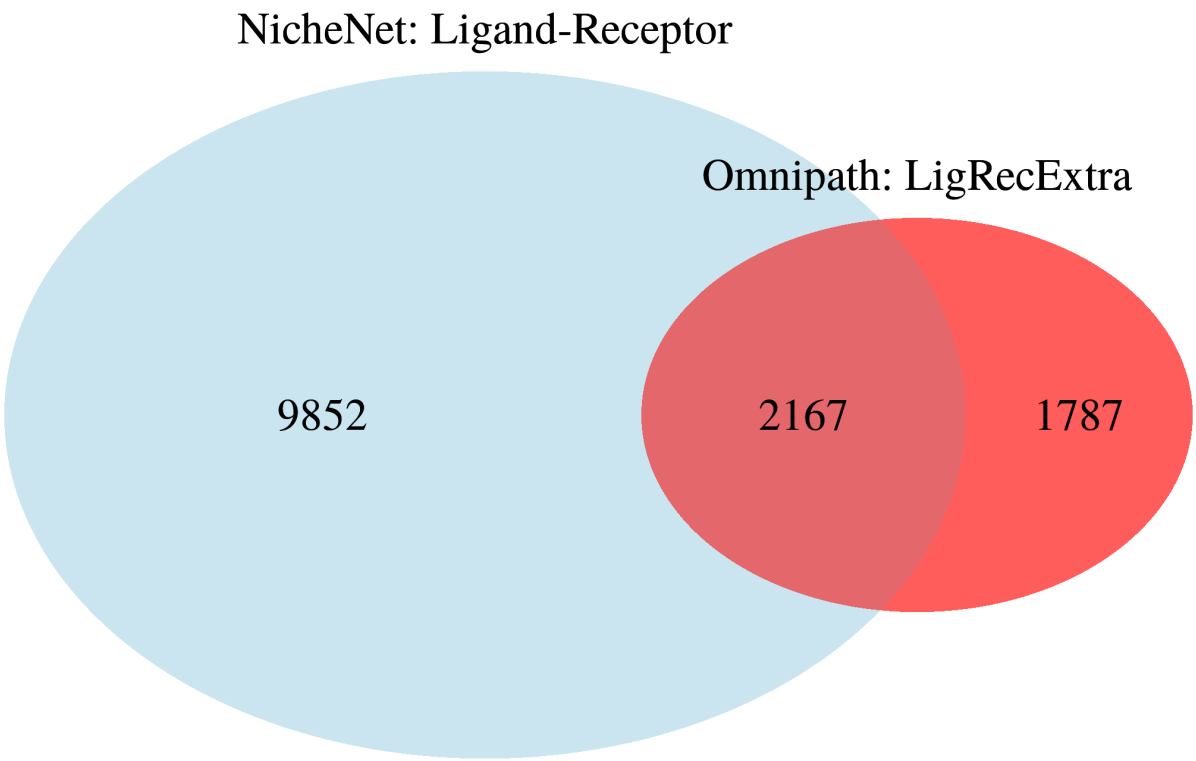
```
## We access to the Omnipath webservice usign the OmnipathR package and we
## display how the interactions look like.
lr_Network_Omnipath <- import_ligrecextra_interactions()
lr_Network_Omnipath[1:6,c(3,4,5,6,7,12)]
##   source_genesymbol target_genesymbol is_directed is_stimulation is_inhibition
## 1             CALM1             TRPC3           1              0             1
## 2            NOTCH1              JAG2           1              0             1
## 3              JAG2            NOTCH1           1              1             1
## 4              DLL1            NOTCH1           1              1             0
## 5            NOTCH1              DLL1           1              1             1
## 6              IGF1             IGF1R           1              1             0
##
## 1
## 2
## 3                                                 Baccin2019;CellPhoneDB;EMBRACE;Fantom5;HPMR;HPRD;IC
## 4                                                       Baccin2019;CellPhoneDB;EMBRACE;Fantom5;HPMR;
## 5
## 6 Baccin2019;CA1;CellPhoneDB;DIP;EMBRACE;Fantom5;Guide2Pharma;HPMR;HPRD;KEGG;Kirouac2010;LRdb;ProtMa
```

We show the total number of ligand-receptor interactions available in the **Omnipath** *LigrecExtra* dataset after removing duplicates:

```
## We also remove self-interactions in case they exist
lr_Network_Omnipath_Unique <- lr_Network_Omnipath %>%
    dplyr::distinct(source_genesymbol,target_genesymbol) %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(from != to)
nrow(lr_Network_Omnipath_Unique)
## [1] 3954
```

We display the number of matching interactions between different sources with a Venn diagram:

```
Venn_plot <- draw.pairwise.venn(nrow(lr_Network_Nichenet_Unique),
    nrow(lr_Network_Omnipath_Unique),
    nrow(dplyr::intersect(lr_Network_Nichenet_Unique,
    lr_Network_Omnipath_Unique)),
    category = c("NicheNet: Ligand-Receptor", "Omnipath: LigRecExtra"),
    lty = rep("blank", 2), fill = c("light blue", "red"), alpha = rep(0.4, 2),
    cat.pos = c(0, 0), cat.dist = rep(0.025, 2), cex= 1.5, cat.cex=1.5,
    verbose = FALSE)
grid.draw(Venn_plot)
```



In **NicheNet**, the authors predicted ligand–receptor interactions by searching in protein–protein interaction databases for interactions between genes annotated as ligands and receptors (Browaeys et al 2019).

We can also do something similar using **Omnipath**. The new version of **Omnipath** also contains protein annotations describing roles in inter-cellular signaling, e.g. if a protein is a ligand, a receptor, an extracellular matrix (ECM) component, etc... Thus, we selected proteins annotated as ligand or receptors and we searched for interactions between them (with ligands as sources of interactions and receptors as sources). The process is described in the following code chunks:

```
## We import Omnipath Inter cellular annotations
InterCell_Annotations <- import_omnipath_intercell()

## We filter those proteins which are mainly annotated as receptor or ligand
Ligands_Receptors <- InterCell_Annotations %>%
    dplyr::filter(category %in% c("receptor","ligand"))

## There are also some complexes. We are going to deal with them by including
## each of its individual proteins in our list
Ligand_Receptors_class <- character()
Ligand_Receptors_name <- character()
for (i in seq(nrow(Ligands_Receptors))){
    if (Ligands_Receptors$entity_type[i] == "complex"){
        Genescomplex <-unlist(strsplit(gsub("COMPLEX:", "",
            Ligands_Receptors$genesymbol[i]),"_"))
        class <- rep(Ligands_Receptors$category[i],length(Genescomplex))
        Ligand_Receptors_name <- c(Ligand_Receptors_name,Genescomplex)
```

```
        Ligand_Receptors_class <- c(Ligand_Receptors_class,class)

    } else {
        Ligand_Receptors_name <-
            c(Ligand_Receptors_name, Ligands_Receptors$genesymbol[i])
        Ligand_Receptors_class <-
            c(Ligand_Receptors_class, Ligands_Receptors$category[i])
    }
}
```

We remove duplicates and we display the number of proteins that we have annotated as ligand or receptors.

```
Ligand_Receptors_df <- data.frame(GeneSymbol = Ligand_Receptors_name,
    Class = Ligand_Receptors_class, stringsAsFactors = FALSE) %>%
    dplyr::distinct()
AllLigands_vec <-
    dplyr::filter(Ligand_Receptors_df, Class == "ligand") %>%
    dplyr::pull(GeneSymbol)
AllReceptors_vec <-
    dplyr::filter(Ligand_Receptors_df, Class == "receptor") %>%
    dplyr::pull(GeneSymbol)
table(Ligand_Receptors_df$Class)
##
##   ligand receptor
##     1049     2328
```

We next get protein-protein interactions from the different datasets availabe in **Omnipath**

```
AllInteractions <-
    import_post_translational_interactions(exclude = "ligrecextra") %>%
    dplyr::select(source_genesymbol, target_genesymbol) %>%
    dplyr::distinct() %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol)
```

```
nrow(AllInteractions)
## [1] 77799
```

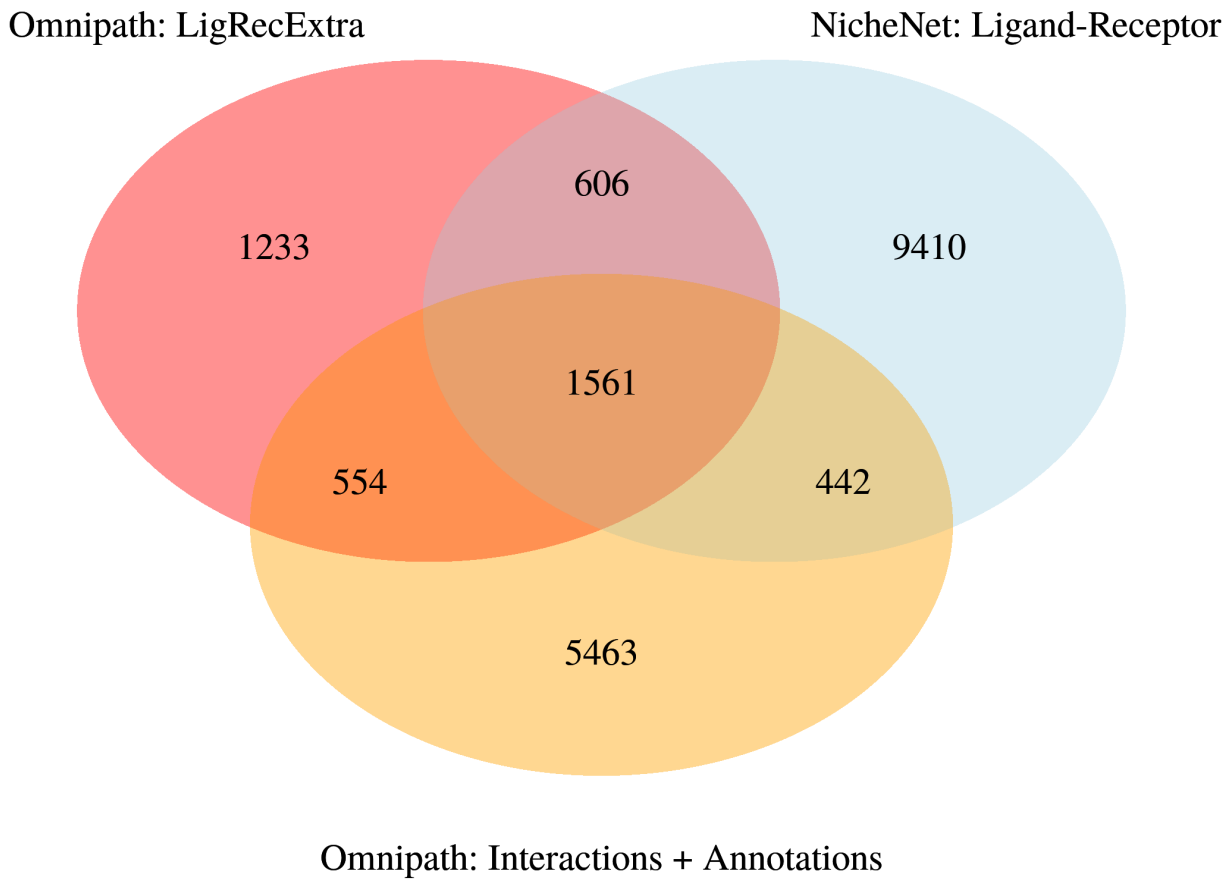
Now, we search for pairs of proteins annotated as ligand and receptor with an interaction between them.

```
## Do the other way around? I only used from=ligands and to=receptors
Matching_Interactions_Annotations <- AllInteractions %>%
    dplyr::filter(from %in% AllLigands_vec) %>%
    dplyr::filter(to %in% AllReceptors_vec) %>%
    dplyr::distinct()
nrow(Matching_Interactions_Annotations)
## [1] 8020
```

We finally display the number of matching interactions between the different sources with a Venn diagram:

```
Venn_plot <- draw.triple.venn(nrow(lr_Network_Nichenet_Unique),
    nrow(lr_Network_Omnipath_Unique),
    nrow(Matching_Interactions_Annotations),
    n12 = nrow(dplyr::intersect(lr_Network_Nichenet_Unique,
        lr_Network_Omnipath_Unique)),
    n23 = nrow(dplyr::intersect(lr_Network_Omnipath_Unique,
        Matching_Interactions_Annotations)),
    n13 = nrow(dplyr::intersect(lr_Network_Nichenet_Unique,
```

```
        Matching_Interactions_Annotations)),
    n123 = nrow(dplyr::intersect(dplyr::intersect(lr_Network_Nichenet_Unique,
        lr_Network_Omnipath_Unique),
        Matching_Interactions_Annotations)),
    category = c("NicheNet: Ligand-Receptor", "Omnipath: LigRecExtra",
        "Omnipath: Interactions + Annotations"),
    lty = rep("blank", 3), fill = c("light blue", "red","orange"),
    alpha = rep(0.25, 3), euler.d = TRUE, scaled=TRUE,
    rotation.degree = 0, reverse=TRUE, cex=1.25, cat.pos = c(330, 30 , 180),
    cat.dist = rep(0.075, 3), cat.cex = 1.25)
grid.draw(Venn_plot)
```



Signaling Network

In this section, we compare the singaling network used to build **NicheNet** with the one available through **Omnipath**. We first load the signaling interactions used to build the **NicheNet** model.

```
## We read the data which are freely available via Zenodo and we display
## how they look like.
sig_Network_Nichenet <-
    readRDS(url("https://zenodo.org/record/3260758/files/signaling_network.rds"))
sig_Network_Nichenet
## # A tibble: 3,621,987 x 4
##    from    to    source                                    database
##    <chr>   <chr> <chr>                                     <chr>
```

```
##  1 BTRC    PDCD4 omnipath_directional                    omnipath
##  2 RPS6KB1 PDCD4 omnipath_directional                    omnipath
##  3 RPS6KB1 PDCD4 pathwaycommons_controls_phosphorylation... pathwaycommons_signal...
##  4 RPS6KB1 PDCD4 pathwaycommons_controls_state_change_of  pathwaycommons_signal...
##  5 RPS6KB1 PDCD4 harmonizome_KEA                         harmonizome
##  6 RPS6KB1 PDCD4 harmonizome_PhosphoSite                 harmonizome
##  7 AKT1    PDCD4 omnipath_directional                    omnipath
##  8 AKT1    PDCD4 pathwaycommons_controls_phosphorylation... pathwaycommons_signal...
##  9 AKT1    PDCD4 pathwaycommons_controls_state_change_of  pathwaycommons_signal...
## 10 AKT1    PDCD4 harmonizome_KEA                         harmonizome
## # ... with 3,621,977 more rows
```

We show the total number of ligand-receptor interactions after removing duplicates:

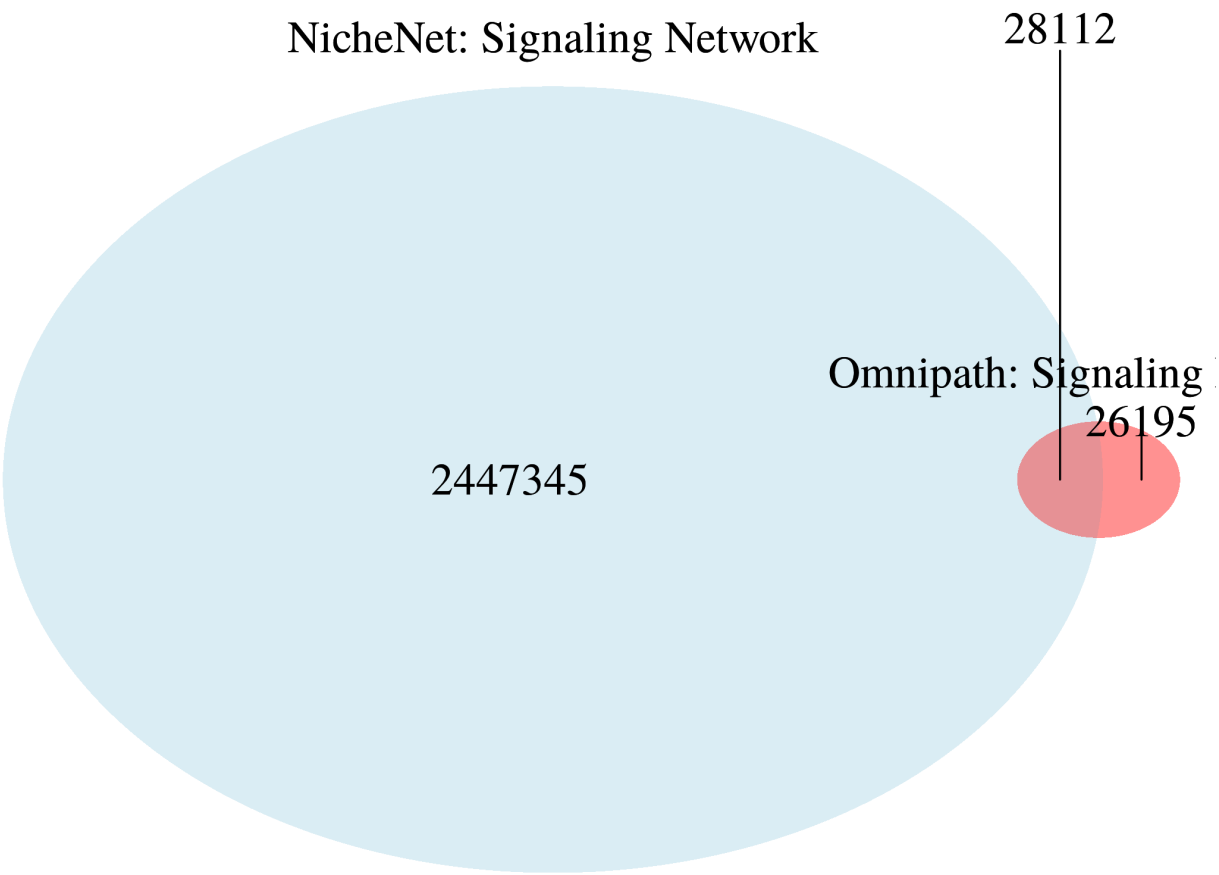
```
sig_Network_Nichenet_Unique <- sig_Network_Nichenet %>%
    dplyr::distinct(from, to)
nrow(sig_Network_Nichenet_Unique)
## [1] 2475457
```

We generate a signaling network using **Omnipath** resources:

```
sig_Interactions_Omnipath <-
    import_post_translational_interactions(exclude = "ligrecextra") %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(consensus_direction == "1") %>%
    dplyr::distinct(from, to, .keep_all = TRUE)  %>%
    dplyr::select(from, to)

sig_Network_Omnipath <- sig_Interactions_Omnipath %>%
    dplyr::distinct()
nrow(sig_Network_Omnipath)
## [1] 54307
```

We finally display the number of matching interactions between the different sources with a Venn diagram:

```
Venn_plot <- draw.pairwise.venn(nrow(sig_Network_Nichenet_Unique),
    nrow(sig_Network_Omnipath),
    nrow(dplyr::intersect(sig_Network_Nichenet_Unique,
    sig_Network_Omnipath)),
    category = c("NicheNet: Signaling Network",
        "Omnipath: Signaling Network"),
    lty = rep("blank", 2), fill = c("light blue", "red"), alpha = rep(0.25, 2),
    cat.pos = c(0, 0), cat.dist = rep(0.025, 2), cex= 1.5, cat.cex=1.5)
grid.draw(Venn_plot)
```



Gene Regulatory Network

In this section, we compare the GRN network used to build **NicheNet** with the **DoRothEA** regulons available through **Omnipath**. We first load the GNR interactions used to build the **NicheNet** model.

```
## We read the data which are freely available via Zenodo and we display
## how they look like.
gr_Network_Nichenet <-
    readRDS(url("https://zenodo.org/record/3260758/files/gr_network.rds"))
gr_Network_Nichenet
## # A tibble: 3,592,299 x 4
##    from  to      source           database
##    <chr> <chr>   <chr>            <chr>
##  1 KLF2  DLGAP1  harmonizome_CHEA harmonizome_gr
##  2 KLF2  DTNB    harmonizome_CHEA harmonizome_gr
##  3 KLF2  BHLHE40 harmonizome_CHEA harmonizome_gr
##  4 KLF2  RPS6KA1 harmonizome_CHEA harmonizome_gr
##  5 KLF2  PXN     harmonizome_CHEA harmonizome_gr
##  6 KLF2  UBE2V1  harmonizome_CHEA harmonizome_gr
##  7 KLF2  MSRA    harmonizome_CHEA harmonizome_gr
##  8 KLF2  TEX14   harmonizome_CHEA harmonizome_gr
##  9 KLF2  CYLD    harmonizome_CHEA harmonizome_gr
## 10 KLF2  RYBP    harmonizome_CHEA harmonizome_gr
## # ... with 3,592,289 more rows
```

We show the total number of ligand-receptor interactions after removing duplicates:

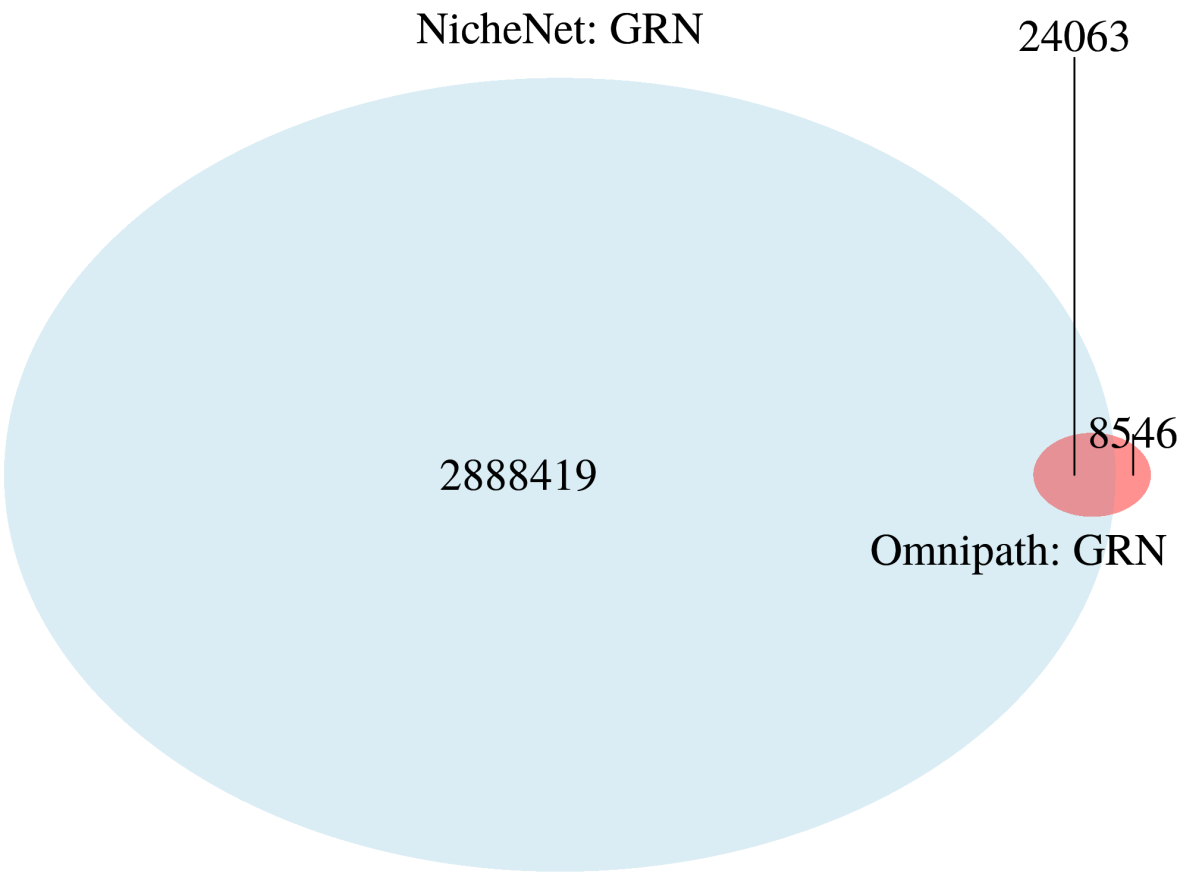
```
gr_Network_Nichenet_unique <- gr_Network_Nichenet %>%
    dplyr::distinct(from, to)
nrow(gr_Network_Nichenet_unique)
## [1] 2912482
```

We generate a GRN network using **Omnipath** resources:

```
gr_Interactions_Omnipath <-
    import_dorothea_interactions(dorothea_levels = c('A','B','C')) %>%
    dplyr::select(source_genesymbol, target_genesymbol) %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::distinct(from, to)
nrow(gr_Interactions_Omnipath)
## [1] 32609
```

We finally display the number of matching interactions between the different sources with a Venn diagram:

```
Venn_plot <-
    draw.pairwise.venn(nrow(gr_Network_Nichenet_unique),
    nrow(gr_Interactions_Omnipath),
    nrow(dplyr::intersect(gr_Network_Nichenet_unique,gr_Interactions_Omnipath)),
    category = c("NicheNet: GRN", "Omnipath: GRN"),
    lty = rep("blank", 2), fill = c("light blue", "red"), alpha = rep(0.25, 2),
    cat.pos = c(0, 215), cat.dist = rep(0.025, 2), cex= 1.5, cat.cex=1.5)
grid.draw(Venn_plot)
```

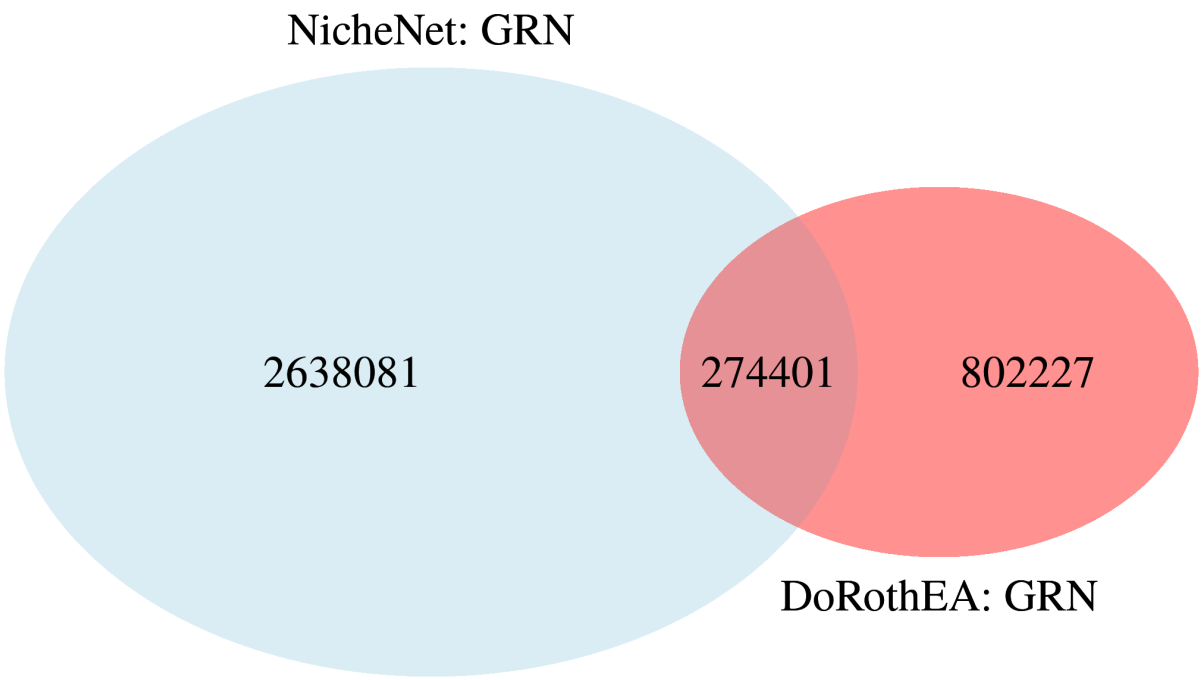


it is to note that **DoRothEA** contains some additional interactions which are not available in the **Omnipath**

web server. We now compare **NicheNet** gene regulatory interaction with the all **DoRothEA** regulons.

```
## To download from:
## https://github.com/saezlab/DoRothEA/tree/master/data/TFregulons/consensus/table
full_dorothea <- read_csv("DoRothEA/database.csv")

full_dorothea_unique <- full_dorothea %>%
    dplyr::select(TF,target) %>%
    dplyr::rename(from=TF, to=target) %>%
    dplyr::distinct(from,to)
nrow(full_dorothea_unique)
## [1] 1076628
```

```
Venn_plot <- draw.pairwise.venn(nrow(gr_Network_Nichenet_unique),
    nrow(full_dorothea_unique),
    nrow(dplyr::intersect(gr_Network_Nichenet_unique,
    full_dorothea_unique)),
    category = c("NicheNet: GRN", "DoRothEA: GRN"),
    lty = rep("blank", 2), fill = c("light blue", "red"), alpha = rep(0.25, 2),
    cat.pos = c(0, 180), cat.dist = rep(0.025, 2), cex= 1.5, cat.cex=1.5)
grid.draw(Venn_plot)
```



References

Bonnardel et al. Stellate Cells, Hepatocytes, and Endothelial Cells Imprint the Kupffer Cell Identity on Monocytes Colonizing the Liver Macrophage Niche. Immunity (2019) doi:10.1016/j.immuni.2019.08.017

Browaeys, R., Saelens, W. & Saeys, Y. NicheNet: modeling intercellular communication by linking ligands to target genes. Nat Methods (2019) doi:10.1038/s41592-019-0667-5

Parameter optimization via mlrMBO of the NicheNet method using Omnipath Interactions ==========
Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 29/02/2020

Abstract

This vignette shows how to optimice the parameters of the Nichenet method, i.e. PageRank parameter and source weights, using sets of interactions availabe in the **Omnipath** database.

The NicheNet Method

NicheNet (https://github.com/saeyslab/nichenetr) is a method to predict ligand-target links between interacting cells by combining their data with prior knowledge on signaling and gene regulatory networks (Browaeys et al 2019). **NicheNet** has already been applied to predict upstream niche signals driving Kupffer cell differentiation (Bonnardel et al. 2019).

NicheNet uses many different publically available resources to build a prior knowledge network. Their final integrated network is composed of three individual networks:

- A network of ligand-receptor interactions (Inter-cellular)
- A network of signaling interactions (Intra-cellular)
- A network of gene regulation (Intra-cellular)

The new version of the **Omnipath** (http://omnipathdb.org/) database contains curated interactions belonging to these three categories. One can therefore build an integrated network equivalent to the one used in **NicheNet** by only fetching the **Omnipath** webserver. This can significantly ease the integration of different databases, each one of them storing data in distint formats and whose interaction show different levels of reliability.

We therefore here optimice the parameters used in the **NicheNet's** algorithm, i.e. PageRank parameters and source weights, using interactions available in the **Omnipath** database.

```
library(OmnipathR)
library(nichenetr)
library(tidyverse)
library(mlrMBO)
library(parallelMap)
```

Fetching Omnipath interactions and transfoming their format

We first define a function to transform the format of interactions in the **Omnipath** to the one used in the **NicheNet** method:

```
interactionFormatTransf <- function(InputDf, InteractionType){

    OutputInt <- tibble(from = character(), to = character(),
        source = character(), database = character())

    n <- nrow(InputDf)
    sources <- dplyr::pull(InputDf, sources)
    sourceNodes <- dplyr::pull(InputDf, from)
    targetNodes <- dplyr::pull(InputDf, to)

    for (i in seq(n)){
        currentSources <- unlist(strsplit(sources[i],";"))
        for (j in seq(length(currentSources))){
            OutputInt <- add_row(OutputInt,
                from = sourceNodes[i] ,
```

```
                to = targetNodes[i],
                # source = paste(currentSources[j], InteractionType, sep="_"),
                source = currentSources[j],
                database = currentSources[j])
        }
    }

    return(OutputInt)
}
```

Generating the Omnipath ligand-receptor network

Omnipath possess a dedicated dataset storing these type of interactions (*LigrecExtra*). Therefore, we get these interactions:

```
## We remove self-interacitons and duplicated records
lr_Interactions_Omnipath <- import_ligrecextra_interactions() %>%
    dplyr::select(source_genesymbol,target_genesymbol,sources) %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(from != to) %>%
    dplyr::distinct()
```

In **NicheNet**, the authors predicted ligand–receptor interactions by searching in protein–protein interaction databases for interactions between genes annotated as ligands and receptors (Browaeys et al 2019).

We can also do something similar using **Omnipath**. The new version of **Omnipath** also contains protein annotations describing roles in inter-cellular signaling, e.g. if a protein is a ligand, a receptor, an extracellular matrix (ECM) component, etc... Thus, we selected proteins annotated as ligand or receptors and we searched for interactions between them (with ligands as sources of interactions and receptors as sources). The process is described in the following code chunks:

```
## We import Omnipath Inter cellular annotations
InterCell_Annotations <- import_omnipath_intercell()

## We filter those proteins which are mainly annotated as receptor or ligand
Ligands_Receptors <- InterCell_Annotations %>%
    dplyr::filter(category %in% c("receptor","ligand"))

## There are also some complexes. We are going to deal with them by including
## each of its individual proteins in our list
Ligand_Receptors_class <- character()
Ligand_Receptors_name <- character()
for (i in seq(nrow(Ligands_Receptors))){
    if (Ligands_Receptors$entity_type[i] == "complex"){
        Genescomplex <-unlist(strsplit(gsub("COMPLEX:", "",
            Ligands_Receptors$genesymbol[i]),"_"))
        class <- rep(Ligands_Receptors$category[i],length(Genescomplex))
        Ligand_Receptors_name <- c(Ligand_Receptors_name,Genescomplex)
        Ligand_Receptors_class <- c(Ligand_Receptors_class,class)

    } else {
        Ligand_Receptors_name <-
            c(Ligand_Receptors_name, Ligands_Receptors$genesymbol[i])
        Ligand_Receptors_class <-
            c(Ligand_Receptors_class, Ligands_Receptors$category[i])
```

```
    }
}

## We create a vector with all the ligands and another with all the receptors.
Ligand_Receptors_df <- data.frame(GeneSymbol = Ligand_Receptors_name,
    Class = Ligand_Receptors_class, stringsAsFactors = FALSE) %>%
    dplyr::distinct()
AllLigands_vec <-
    dplyr::filter(Ligand_Receptors_df, Class == "ligand") %>%
    dplyr::pull(GeneSymbol)
AllReceptors_vec <-
    dplyr::filter(Ligand_Receptors_df, Class == "receptor") %>%
    dplyr::pull(GeneSymbol)

## We next get protein-protein interactions from the different datasets availabe
## in Omnipath
AllInteractions <-
    import_post_translational_interactions(exclude = "ligrecextra") %>%
    dplyr::select(source_genesymbol, target_genesymbol, sources) %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(from != to) %>%
    dplyr::distinct()

## I finally match interactions and annotations.
Matching_Interactions_Annotations <- AllInteractions %>%
    dplyr::filter(from %in% AllLigands_vec) %>%
    dplyr::filter(to %in% AllReceptors_vec) %>%
    dplyr::distinct()
```

We now combine these two soruces of ligand receptor interactions and then transform to the network format required by the **NicheNet** method:

```
## We access to the Omnipath webservice usign the OmnipathR package and we
## set the number of sources reporting an interactions as its weight
lr_Network_Omnipath <-
    bind_rows(lr_Interactions_Omnipath, Matching_Interactions_Annotations) %>%
    dplyr::distinct() %>%
    interactionFormatTransf(InteractionType="LigrecExtra") %>%
    dplyr::distinct()

## I have to remove self-interactions
lr_Network_Omnipath <- lr_Network_Omnipath %>%
    dplyr::filter(from != to)

## I also have to remove interactions going to ligands. See Methods Nichenet
## paper
ligands <- unique(dplyr::pull(lr_Network_Omnipath, from))
lr_Network_Omnipath <- lr_Network_Omnipath %>%
    dplyr::filter(!(to %in% ligands))

## There are in addition some records containing not input gene, we remove them
## since they are giving problems with running the model.
lr_Network_Omnipath <- lr_Network_Omnipath %>%
    dplyr::filter(from != "") %>%
```

```
    dplyr::filter(to != "")
nrow(lr_Network_Omnipath)
## [1] 20460
```

```
saveRDS(lr_Network_Omnipath,
        "OmniNetworks_NNformat/lr_Network_Omnipath.rds")
```

Generating the Omnipath signalling network

We generate a signaling network using **Omnipath** resources. In **Omnipath**, we can find different datasets describing protein interactions (https://github.com/saezlab/pypath). We will merge these datasts to generate a signaling network.

```
## Original Omnipath interactions
sig_Network_Omnipath <-
    interactionFormatTransf(AllInteractions, InteractionType="Signalling") %>%
    dplyr::distinct()

## I have to remove self-interactions in the signaling network
sig_Network_Omnipath <- sig_Network_Omnipath %>%
    dplyr::filter(from != to)

## I also have to remove interactions going to ligands. See Methods Nichenet
## paper
sig_Network_Omnipath <- sig_Network_Omnipath %>%
    dplyr::filter(!(to %in% ligands))

## There are in addition some records containing not input gene, we remove them
## since they are giving problems with running the model.
sig_Network_Omnipath <- sig_Network_Omnipath %>%
    dplyr::filter(from != "") %>%
    dplyr::filter(to != "")


## We also remove signaling interactions that are already in the lig-receptor
## network.
sig_Network_Omnipath <- dplyr::anti_join(
  sig_Network_Omnipath,
  lr_Network_Omnipath,
  by = c("from" = "from", "to" = "to"))

nrow(sig_Network_Omnipath)
## [1] 137910
```

```
saveRDS(sig_Network_Omnipath,
    "OmniNetworks_NNformat/sig_Network_Omnipath.rds")
```

Generating the Omnipath gene regulatory network

In this section, we generate a GRN network using the **DoRothEA** regulons which are available through **Omnipath**:

```
gr_Interactions_Omnipath <-
    import_dorothea_interactions(dorothea_levels = c("A","B","C")) %>%
    dplyr::select(source_genesymbol, target_genesymbol, sources) %>%
```

```
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(from != to) %>%
    dplyr::distinct()

gr_Network_Omnipath <-
    interactionFormatTransf(
        gr_Interactions_Omnipath,
        InteractionType="Dorothea") %>%
    dplyr::distinct()
nrow(gr_Network_Omnipath)
## [1] 113897
```

```
saveRDS(gr_Network_Omnipath,
    "OmniNetworks_NNformat/gr_Network_Omnipath.rds")
```

Parameter optimization via mlrMBO

We are going to optimice Nichenet method, i.e. PageRank parameters and source weights, basedon a collection of experiments where the effect of a ligand on gene expression was measured. We have to remove the experiments involving the ligand IFNA1 because it is not present in our network.

```
expression_settings_validation <-
    readRDS(url("https://zenodo.org/record/3260758/files/expression_settings.rds"))

index <- which(!unlist(lapply(expression_settings_validation,
    function(x) any(x$from != "IFNA1"))))

expression_settings_validation <- expression_settings_validation[-index]
```

Running the optimisation can take quite long depending on the number of interactions in the network, the number of iterations of the optimisation process and the number cores of your machine). We finally save the results that would be used in the forthcoming scripts.

```
All_sources <- unique(c(lr_Network_Omnipath$source,
    sig_Network_Omnipath$source, gr_Network_Omnipath$source))

my_source_weights_df <-
     tibble(source = All_sources, weight = rep(1,length(All_sources)))

additional_arguments_topology_correction <-
    list(source_names = my_source_weights_df$source %>% unique(),
        algorithm = "PPR",
        correct_topology = FALSE,
        lr_network = lr_Network_Omnipath,
        sig_network = sig_Network_Omnipath,
        gr_network = gr_Network_Omnipath,
        settings = lapply(expression_settings_validation,
            convert_expression_settings_evaluation),
        secondary_targets = FALSE,
        remove_direct_links = "no",
        cutoff_method = "quantile")

nr_datasources <- additional_arguments_topology_correction$source_names %>%
    length()
```

```
obj_fun_multi_topology_correction = makeMultiObjectiveFunction(name = "nichenet_optimization",
    description = "data source weight and hyperparameter optimization: expensive black-box function",
    fn = model_evaluation_optimization,
    par.set = makeParamSet(
        makeNumericVectorParam("source_weights", len = nr_datasources,
            lower = 0, upper = 1, tunable = FALSE),
        makeNumericVectorParam("lr_sig_hub", len = 1, lower = 0, upper = 1,
            tunable = TRUE),
        makeNumericVectorParam("gr_hub", len = 1, lower = 0, upper = 1,
            tunable = TRUE),
        makeNumericVectorParam("ltf_cutoff", len = 1, lower = 0.9,
            upper = 0.999, tunable = TRUE),
        makeNumericVectorParam("damping_factor", len = 1, lower = 0.01,
            upper = 0.99, tunable =TRUE)),
    has.simple.signature = FALSE,
    n.objectives = 4,
    noisy = FALSE,
    minimize = c(FALSE,FALSE,FALSE,FALSE))

optimization_results =
    lapply(1,mlrmbo_optimization, obj_fun = obj_fun_multi_topology_correction,
           niter = 8, ncores = 8, nstart = 160,
           additional_arguments = additional_arguments_topology_correction)

saveRDS(optimization_results, "Results/Optimization_results.rds")
```

References

Bonnardel et al. Stellate Cells, Hepatocytes, and Endothelial Cells Imprint the Kupffer Cell Identity on Monocytes Colonizing the Liver Macrophage Niche. Immunity (2019) doi:10.1016/j.immuni.2019.08.017

Browaeys, R., Saelens, W. & Saeys, Y. NicheNet: modeling intercellular communication by linking ligands to target genes. Nat Methods (2019) doi:10.1038/s41592-019-0667-5

Construction of NicheNet's ligand-target model using Omnipath Interactions ==========
Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 29/02/2020

Abstract

This vignette shows how to build the ligand-target prior regulatory potential scores based on the NicheNet method but using Omnipath interactions.

Construct a ligand-target model from Omnipath ligand-receptor, signaling and gene regulatory networks

We first load the required packages and the networks generated on the previous scripts: Omnipath ligand-receptor, Omnipath signaling and Omnipath (Dorothea) gene regulatory network. These networks are used to construct the ligand-target model.

```
library(nichenetr)
library(tidyverse)

## We load the networks generated in the ParameterOptimization script
lr_network_Omnipath = readRDS("OmniNetworks_NNformat/lr_Network_Omnipath.rds")
sig_network_Omnipath = readRDS("OmniNetworks_NNformat/sig_Network_Omnipath.rds")
gr_network_Omnipath = readRDS("OmniNetworks_NNformat/gr_Network_Omnipath.rds")
```

Construct NicheNet's ligand-target model from unoptimized data source weights

The interactions available in Omnipath are well curated, and we therefore first create a model considering the same weight for all the databases. On the other hand, we use the optimized parameters for the PageRank algorithm.

```
## The interactions are weighted only based in the number of data sources that
## report them
All_sources <- unique(c(lr_network_Omnipath$source,
    sig_network_Omnipath$source, gr_network_Omnipath$source))

my_source_weights_df <-
     tibble(source = All_sources, weight = rep(1,length(All_sources)))

weighted_networks <- construct_weighted_networks(
    lr_network = lr_network_Omnipath,
    sig_network = sig_network_Omnipath,
    gr_network = gr_network_Omnipath,
    source_weights_df = my_source_weights_df)

## We read the results of the optimization
resultsOptimization <- readRDS("Results/Optimization_results.rds")

optimized_parameters = resultsOptimization %>%
    process_mlrmbo_nichenet_optimization(
        source_names = my_source_weights_df$source %>% unique())

weighted_networks <- apply_hub_corrections(
    weighted_networks = weighted_networks,
    lr_sig_hub = optimized_parameters$lr_sig_hub,
    gr_hub = optimized_parameters$gr_hub)
saveRDS(weighted_networks,"Results/weighted_networksNonSourceWeights.rds")
```

We now generate the matrix containing the ligand-target regulatory potential scores based on the weighted integrated networks.

```
ligands <- as.list(unique(lr_network_Omnipath$from))

ligand_target_matrix <- construct_ligand_target_matrix(
    weighted_networks = weighted_networks,
    ligands = ligands,
    algorithm = "PPR",
    damping_factor = optimized_parameters$damping_factor,
    ltf_cutoff = optimized_parameters$ltf_cutoff)
saveRDS(ligand_target_matrix,"Results/ligand_target_matrixNoweights.rds")
```

Construct NicheNet's ligand-target model from optimized data source weights

Now, we create an alternative model using the optimized weights for the different sources of data.

```
## Here we also take into account the optimized source weights
weighted_networks <- construct_weighted_networks(
    lr_network = lr_network_Omnipath,
    sig_network = sig_network_Omnipath,
    gr_network = gr_network_Omnipath,
    source_weights_df = optimized_parameters$source_weight_df)

weighted_networks <- apply_hub_corrections(
    weighted_networks = weighted_networks,
    lr_sig_hub = optimized_parameters$lr_sig_hub,
    gr_hub = optimized_parameters$gr_hub)

ligand_target_matrix = construct_ligand_target_matrix(
    weighted_networks = weighted_networks,
    ligands = ligands,
    algorithm = "PPR",
    damping_factor = optimized_parameters$damping_factor,
    ltf_cutoff = optimized_parameters$ltf_cutoff)
saveRDS(ligand_target_matrix,"Results/ligand_target_matrixWithweights.rds")
saveRDS(weighted_networks,"Results/weighted_networksWithSourceWeights.rds")
```

SARS-CoV-2 dataset: Differential expression analysis

Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 14/04/2020

Introduction

The present script takes the RNAseq data from the study *"SARS-CoV-2 launches a unique transcriptional signature from in vitro, ex vivo, and in vivo systems"*

https://www.biorxiv.org/content/10.1101/2020.03.24.004655v1

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE147507

It performs a differential expression analysis to compare:

- Human lung epithelial cells (NHBE): mock treated vs infected with SARS-CoV-2.
- A549 alveolar cancer cell line: mock treated vs infected with SARS-CoV-2.
- Calu-3 human lung epithelial cancer cell line: mock treated vs infected with SARS-CoV-2.

We used the *DESeq2* R package.

Getting Started

We first load the required libraries.

```
library(dplyr)
library(DESeq2)
library(tibble)
library(VennDiagram)
library(fgsea)
```

We also read the raw counts from the original experiment (GSE147507)

```
## Raw counts table
GSE147507_raw_counts <-
    read.csv("GSE147507_RawReadCounts_Human.tsv", sep = "\t")
```

NHBE mock treated vs infected with SARS-CoV-2

We first select the Series 1, which corresponds to independent biological triplicates of primary human lung epithelium (NHBE) that were either mock treated or infected with SARS-CoV-2.

```
## We select series 1 as described above.
count_NHBEvsCOV2_df <- GSE147507_raw_counts[,c(2:7)]
row.names(count_NHBEvsCOV2_df) <- GSE147507_raw_counts$X
```

We create a dataframe describing the experimental design.

```
targets_NHBEvsCOV2 <-
    as.data.frame(matrix(NA,length(names(count_NHBEvsCOV2_df)),1))
names(targets_NHBEvsCOV2) <- c("condition")
row.names(targets_NHBEvsCOV2) <- names(count_NHBEvsCOV2_df)
targets_NHBEvsCOV2$condition <-
    gsub("Series1_", "", row.names(targets_NHBEvsCOV2))
targets_NHBEvsCOV2$condition <-
    factor(gsub("_[1-3]$", "", targets_NHBEvsCOV2$condition))
targets_NHBEvsCOV2
```

```
##                                 condition
## Series1_NHBE_Mock_1             NHBE_Mock
## Series1_NHBE_Mock_2             NHBE_Mock
## Series1_NHBE_Mock_3             NHBE_Mock
## Series1_NHBE_SARS.CoV.2_1 NHBE_SARS.CoV.2
## Series1_NHBE_SARS.CoV.2_2 NHBE_SARS.CoV.2
## Series1_NHBE_SARS.CoV.2_3 NHBE_SARS.CoV.2
```

Then, we perform the differential expression analysis with *DESeq2*

```
## Create deseq2 object
dds_NHBEvsCOV2 <-
    DESeqDataSetFromMatrix(countData = as.matrix(count_NHBEvsCOV2_df),
    colData = targets_NHBEvsCOV2, design = ~ condition)

## Set control
dds_NHBEvsCOV2$condition <- relevel(dds_NHBEvsCOV2$condition,
    ref = levels(targets_NHBEvsCOV2$condition)[1])

## Carry out diff exp
dds_NHBEvsCOV2 <- DESeq(dds_NHBEvsCOV2)
```

We finally save the table with the results of the analysis and the normalised counts

```
## See the comparisons carried out
comparison_NHBEvsCOV2 <- resultsNames(dds_NHBEvsCOV2)

## Get results table
results_NHBEvsCOV2 <-
    results(dds_NHBEvsCOV2, name=comparison_NHBEvsCOV2[2])

## Save the object
saveRDS(results_NHBEvsCOV2, file="Results/dds_results_NHBEvsCOV2.rds")

## Extract normalised counts data
dds_NHBEvsCOV2 <- estimateSizeFactors(dds_NHBEvsCOV2)
norm_counts_NHBEvsCOV2 <- counts(dds_NHBEvsCOV2, normalized = TRUE)
saveRDS(norm_counts_NHBEvsCOV2,
    file="Results/counts_norm_NHBEvsCOV2.rds")
```

A549 mock treated vs infected with SARS-CoV-2

We first select the Series 5, which corresponds to independent biological triplicates of alveolar cancer cell line (A549) that were either mock treated or infected with SARS-CoV-2.

```
## We select series 5 as described above.
count_A549vsCOV2_df <- GSE147507_raw_counts[,c(22:27)]
row.names(count_A549vsCOV2_df) <- GSE147507_raw_counts$X
```

We create a dataframe describing the experimental design.

```
targets_A549vsCOV2 <-
    as.data.frame(matrix(NA,length(names(count_A549vsCOV2_df)),1))
names(targets_A549vsCOV2) <- c("condition")
row.names(targets_A549vsCOV2) <- names(count_A549vsCOV2_df)
targets_A549vsCOV2$condition <-
    gsub("Series5_", "", row.names(targets_A549vsCOV2))
targets_A549vsCOV2$condition <-
    factor(gsub("_[1-3]$", "", targets_A549vsCOV2$condition))
targets_A549vsCOV2
```

```
##                                 condition
## Series5_A549_Mock_1             A549_Mock
## Series5_A549_Mock_2             A549_Mock
## Series5_A549_Mock_3             A549_Mock
## Series5_A549_SARS.CoV.2_1 A549_SARS.CoV.2
## Series5_A549_SARS.CoV.2_2 A549_SARS.CoV.2
## Series5_A549_SARS.CoV.2_3 A549_SARS.CoV.2
```

Then, we perform the differential expression analysis with *DESeq2*

```
## Create deseq2 object
dds_A549vsCOV2 <-
    DESeqDataSetFromMatrix(countData = as.matrix(count_A549vsCOV2_df),
    colData = targets_A549vsCOV2, design = ~ condition)

## Set control
dds_A549vsCOV2$condition <- relevel(dds_A549vsCOV2$condition,
    ref = levels(targets_A549vsCOV2$condition)[1])

## Carry out diff exp
dds_A549vsCOV2 <- DESeq(dds_A549vsCOV2)
```

We finally save the table with the results of the analysis and the normalised counts

```
## See the comparisons carried out
comparison_A549vsCOV2 <- resultsNames(dds_A549vsCOV2)

## Get results table
results_A549vsCOV2 <-
    results(dds_A549vsCOV2, name=comparison_A549vsCOV2[2])

## Save the object
saveRDS(results_A549vsCOV2,
    file="Results/dds_results_A549vsCOV2.rds")

## Extract normalised counts data
dds_A549vsCOV2 <- estimateSizeFactors(dds_A549vsCOV2)
norm_counts_A549vsCOV2 <- counts(dds_A549vsCOV2, normalized = TRUE)
saveRDS(norm_counts_A549vsCOV2,
    file="Results/counts_norm_A549vsCOV2.rds")
```

CALU-3 mock treated vs infected with SARS-CoV-2

We then select the Series 7, which corresponds to independent biological triplicates of a human lung epithelium cancer cell line named Calu-3, that were either mock treated or infected with SARS-CoV-2.

```
## We select series 7 as described above.
count_CALU3vsCOV2_df <- GSE147507_raw_counts[,c(34:39)]
row.names(count_CALU3vsCOV2_df) <- GSE147507_raw_counts$X
```

We create a dataframe describing the experimental design.

```
targets_CALU3vsCOV2 <-
    as.data.frame(matrix(NA,length(names(count_CALU3vsCOV2_df)),1))
names(targets_CALU3vsCOV2) <- c("condition")
row.names(targets_CALU3vsCOV2) <- names(count_CALU3vsCOV2_df)
targets_CALU3vsCOV2$condition <-
    gsub("Series1_", "", row.names(targets_CALU3vsCOV2))
targets_CALU3vsCOV2$condition <-
    factor(gsub("_[1-3]$", "", targets_CALU3vsCOV2$condition))
targets_CALU3vsCOV2
```

```
##                                         condition
## Series7_Calu3_Mock_1             Series7_Calu3_Mock
## Series7_Calu3_Mock_2             Series7_Calu3_Mock
## Series7_Calu3_Mock_3             Series7_Calu3_Mock
## Series7_Calu3_SARS.CoV.2_1 Series7_Calu3_SARS.CoV.2
## Series7_Calu3_SARS.CoV.2_2 Series7_Calu3_SARS.CoV.2
## Series7_Calu3_SARS.CoV.2_3 Series7_Calu3_SARS.CoV.2
```

Then, we perform the differential expression analysis with *DESeq2*

```
## Create deseq2 object
dds_CALU3vsCOV2 <-
    DESeqDataSetFromMatrix(countData = as.matrix(count_CALU3vsCOV2_df),
    colData = targets_CALU3vsCOV2, design = ~ condition)

## Set control
dds_CALU3vsCOV2$condition <- relevel(dds_CALU3vsCOV2$condition,
    ref = levels(targets_CALU3vsCOV2$condition)[1])

## Carry out diff exp
dds_CALU3vsCOV2 <- DESeq(dds_CALU3vsCOV2)
```

We finally save the table with the results of the analysis and the normalised counts

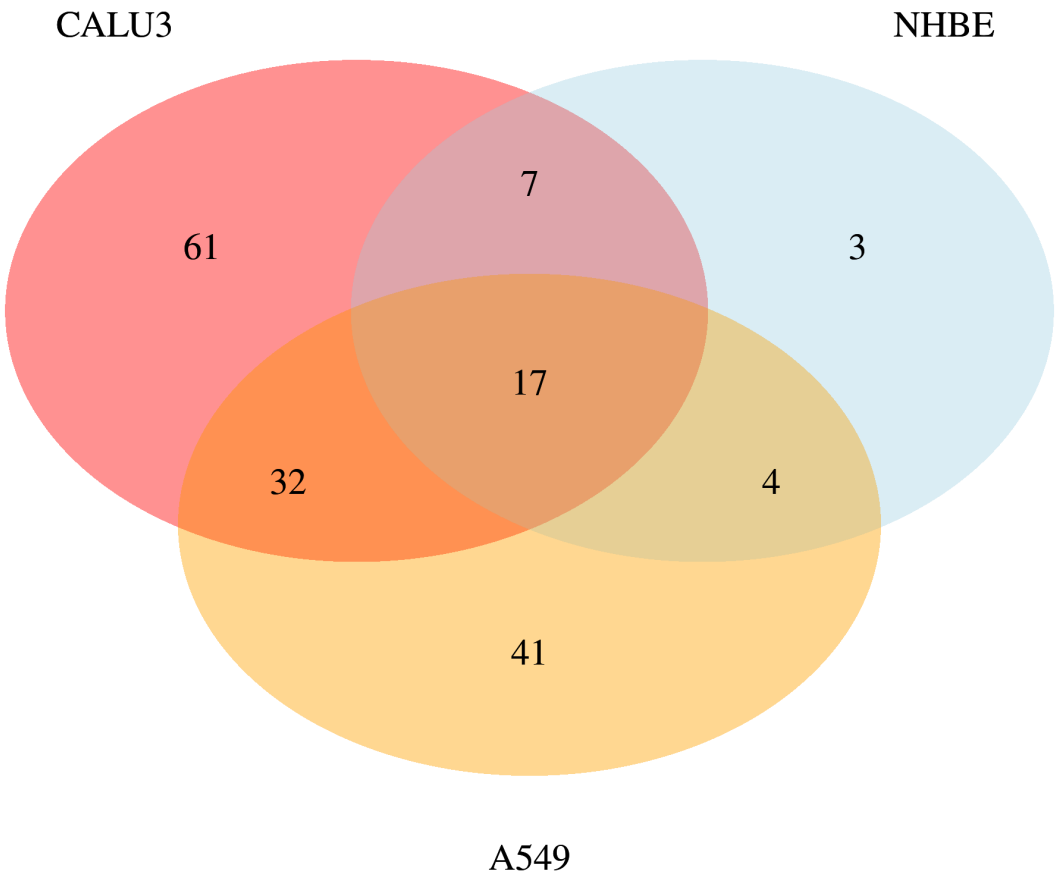
```
## See the comparisons carried out
comparison_CALU3vsCOV2 <- resultsNames(dds_CALU3vsCOV2)

## Get results table
results_CALU3vsCOV2 <-
    results(dds_CALU3vsCOV2, name=comparison_CALU3vsCOV2[2])

## Save the object
saveRDS(results_CALU3vsCOV2, file="Results/dds_results_CALU3vsCOV2.rds")

## Extract normalised counts data
dds_CALU3vsCOV2 <- estimateSizeFactors(dds_CALU3vsCOV2)
norm_counts_CALU3vsCOV2 <- counts(dds_CALU3vsCOV2, normalized = TRUE)
```

```
saveRDS(norm_counts_CALU3vsCOV2,
    file="Results/counts_norm_CALU3vsCOV2.rds")
```

We check the number of ligands over-expressed in every cell line.

```
padj_tres <- 0.1
log2FoldChange_tres <- 1

ligands <-
    readRDS("OmniNetworks_NNformat/lr_Network_Omnipath.rds") %>%
    dplyr::pull(from) %>%
    unique()

DDS_NHBE_ligands <-
    results_NHBEvsCOV2 %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(padj < padj_tres,
                  log2FoldChange > log2FoldChange_tres,
                  Gene %in% ligands) %>%
    dplyr::pull(Gene)

DDS_CALU3_ligands <-
    results_CALU3vsCOV2 %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(padj < padj_tres,
                  log2FoldChange > log2FoldChange_tres,
                  Gene %in% ligands) %>%
    dplyr::pull(Gene)

DDS_A549_ligands <-
    results_A549vsCOV2 %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(padj < padj_tres,
                  log2FoldChange > log2FoldChange_tres,
                  Gene %in% ligands)  %>%
    dplyr::pull(Gene)

Venn_plot <- draw.triple.venn(length(DDS_NHBE_ligands),
    length(DDS_CALU3_ligands),
    length(DDS_A549_ligands),
    n12 = length(intersect(DDS_NHBE_ligands,
        DDS_CALU3_ligands)),
    n23 = length(intersect(DDS_CALU3_ligands,
        DDS_A549_ligands)),
    n13 = length(intersect(DDS_NHBE_ligands,
        DDS_A549_ligands)),
    n123 = length(intersect(intersect(DDS_NHBE_ligands,
        DDS_CALU3_ligands),
        DDS_A549_ligands)),
    category = c("NHBE", "CALU3","A549"),
```

```
    lty = rep("blank", 3), fill = c("light blue", "red","orange"),
    alpha = rep(0.25, 3), euler.d = TRUE, scaled=TRUE,
    rotation.degree = 0, reverse=TRUE, cex=1.25, cat.pos = c(330, 30 , 180),
    cat.dist = rep(0.075, 3), cat.cex = 1.25)
grid.draw(Venn_plot)
```



Session Info Details

```
## R version 4.0.1 (2020-06-06)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 19.10
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/libopenblasp-r0.3.7.so
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_GB.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
```

```
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] fgsea_1.14.0                VennDiagram_1.6.20
##  [3] futile.logger_1.4.3         tibble_3.0.1
##  [5] DESeq2_1.28.1               SummarizedExperiment_1.18.1
##  [7] DelayedArray_0.14.0         matrixStats_0.56.0
##  [9] Biobase_2.48.0              GenomicRanges_1.40.0
## [11] GenomeInfoDb_1.24.0         IRanges_2.22.2
## [13] S4Vectors_0.26.1            BiocGenerics_0.34.0
## [15] dplyr_1.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.4.6           locfit_1.5-9.4         lattice_0.20-41
##  [4] digest_0.6.25          R6_2.4.1               futile.options_1.0.1
##  [7] RSQLite_2.2.0          evaluate_0.14          ggplot2_3.3.1
## [10] pillar_1.4.4           zlibbioc_1.34.0        rlang_0.4.6
## [13] data.table_1.12.8      annotate_1.66.0        blob_1.2.1
## [16] Matrix_1.2-18          rmarkdown_2.2          splines_4.0.1
## [19] BiocParallel_1.22.0    geneplotter_1.66.0     stringr_1.4.0
## [22] RCurl_1.98-1.2         bit_1.1-15.2           munsell_0.5.0
## [25] compiler_4.0.1         xfun_0.14              pkgconfig_2.0.3
## [28] htmltools_0.4.0        tidyselect_1.1.0       gridExtra_2.3
## [31] GenomeInfoDbData_1.2.3 XML_3.99-0.3           crayon_1.3.4
## [34] bitops_1.0-6           xtable_1.8-4           gtable_0.3.0
## [37] lifecycle_0.2.0        DBI_1.1.0              formatR_1.7
## [40] magrittr_1.5           scales_1.1.1           stringi_1.4.6
## [43] XVector_0.28.0         genefilter_1.70.0      ellipsis_0.3.1
## [46] generics_0.0.2         vctrs_0.3.1            fastmatch_1.1-0
## [49] lambda.r_1.2.4         RColorBrewer_1.1-2     tools_4.0.1
## [52] bit64_0.9-7            glue_1.4.1             purrr_0.3.4
## [55] survival_3.1-12        yaml_2.2.1             AnnotationDbi_1.50.0
## [58] colorspace_1.4-1       memoise_1.1.0          knitr_1.28
```

NicheNet's ligand activity analysis on a gene set of interest: predict active ligands and their target genes ========== Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 22/06/2020

Abstract

This vignette follows the steps of the original vignette, available in the NicheNetr repository:

https://github.com/saeyslab/nichenetr/blob/master/vignettes/ligand_activity_geneset.md

In our particular case, we use sets of interactions availabe in the **Omnipath** database. We will study potential ligand-targets influence upon SARS-CoV-2 infection.

Introduction

A NicheNet analysis can help one to generate hypotheses about an intercellular communication process of interest for which you have bulk or single-cell gene expression data. Specifically, NicheNet can predict 1) which ligands from one cell population ("sender/niche") are most likely to affect target gene expression in an interacting cell population ("receiver/target") and 2) which specific target genes are affected by which of these predicted ligands.

Because NicheNet studies how ligands affect gene expression in neighboring cells, you need to have data about this effect in gene expression you want to study. So, you need to have a clear set of genes that are putatively affected by ligands from one of more interacting cells.

The pipeline of a basic NicheNet analysis consist mainly of the following steps:

- 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
- 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)
- 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
- 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)
- 5) Infer top-predicted target genes of ligands that are top-ranked in the ligand activity analysis

This vignette guides you in detail through all these steps. We are going to use expression data after SARS-CoV-2 infection to try to dissect which ligands
expressed by infected cells can have an influence on the expression of target genes in the same cell lines (Autocrine view). In particular, we will focus on the inflamatory response potentially induced by this ligands.

Step 0: NicheNet's ligand-target prior model and expression data of interacting cells

We first loaded the required packages

```
library(nichenetr)
library(tidyverse)
library(VennDiagram)
library(fgsea)
```

Then, we read the prior ligand-target model. This model denotes the prior potential that a particular ligand might regulate the expression of a specific target gene.

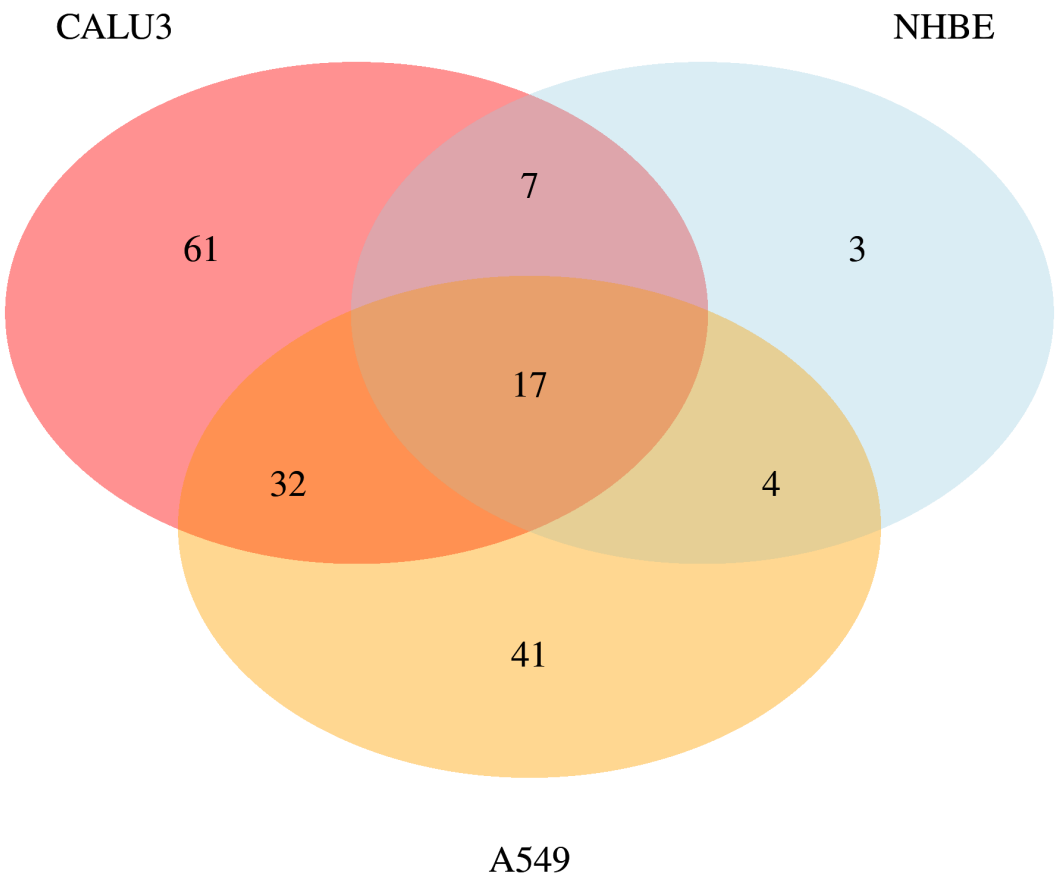
```
ligand_target_matrix = readRDS("Results/ligand_target_matrixWithweights.rds")
# target genes in rows, ligands in columns
dim(ligand_target_matrix)
## [1] 12547   840
ligand_target_matrix[1:5,1:5]
##               CALM1        WNT5A       CXCL16       CCL3L3      TNFSF10
## A1BG   0.0000000000 0.0000000000 0.000000e+00 0.000000e+00 0.0000000000
## A1CF   0.0000000000 0.0000000000 0.000000e+00 0.000000e+00 0.0000000000
## A2M    0.0011027517 0.0004845514 2.936421e-03 5.441192e-03 0.0017391820
## A2ML1  0.0000000000 0.0000000000 0.000000e+00 0.000000e+00 0.0000000000
## A4GALT 0.0002105736 0.0001070804 5.825834e-05 9.488076e-05 0.0001410451
```

We read the differential expression analysis results from several cell lines upon SARS-CoV-2 infection. We are going to explore which ligands are overexpressed after infection in different cell lines belonging to the following dataset: GSE147507 (https://www.biorxiv.org/content/10.1101/2020.03.24.004655v1)

```
padj_tres <- 0.1
log2FoldChange_tres <- 1

## We take our ligands in the network
ligands <-
    readRDS("OmniNetworks_NNformat/lr_Network_Omnipath.rds") %>%
    dplyr::pull(from) %>%
    unique()

DDS_NHBE_ligands <-
    readRDS("Results/dds_results_NHBEvsCOV2.rds") %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(padj < padj_tres,
                  log2FoldChange > log2FoldChange_tres,
                  Gene %in% ligands) %>%
    dplyr::pull(Gene)

DDS_CALU3_ligands <-
    readRDS("Results/dds_results_CALU3vsCOV2.rds") %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(padj < padj_tres,
                  log2FoldChange > log2FoldChange_tres,
                  Gene %in% ligands) %>%
    dplyr::pull(Gene)

DDS_A549_ligands <-
    readRDS("Results/dds_results_A549vsCOV2.rds") %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(padj < padj_tres,
                  log2FoldChange > log2FoldChange_tres,
                  Gene %in% ligands)  %>%
    dplyr::pull(Gene)
```

After checking the overlap between over-expressed ligands in the different cell lines, we decided to continue with the analysis using CALU3, since it has the larger number of over-expressed ligands.

```
Venn_plot <- draw.triple.venn(length(DDS_NHBE_ligands),
    length(DDS_CALU3_ligands),
    length(DDS_A549_ligands),
    n12 = length(intersect(DDS_NHBE_ligands,
        DDS_CALU3_ligands)),
    n23 = length(intersect(DDS_CALU3_ligands,
        DDS_A549_ligands)),
    n13 = length(intersect(DDS_NHBE_ligands,
        DDS_A549_ligands)),
    n123 = length(intersect(intersect(DDS_NHBE_ligands,
        DDS_CALU3_ligands),
        DDS_A549_ligands)),
    category = c("NHBE", "CALU3","A549"),
    lty = rep("blank", 3), fill = c("light blue", "red","orange"),
    alpha = rep(0.25, 3), euler.d = TRUE, scaled=TRUE,
    rotation.degree = 0, reverse=TRUE, cex=1.25, cat.pos = c(330, 30 , 180),
    cat.dist = rep(0.075, 3), cat.cex = 1.25)
grid.draw(Venn_plot)
```



Step 1: Define expressed genes in sender and receiver cell populations

Our research question is to prioritize which ligands overexpressed upon SARS-CoV-2 in the CALU-3 cell line have an effect in the inflamatory response in this very same cell line. This can be considered as an example of autocrine signaling.

Now, we will take again the overexpressed ligands after infection and we will define as a background all the genes expressed by the CALU3 cells.

```
expressed_genes_receiver <-
    readRDS("Results/dds_results_CALU3vsCOV2.rds") %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(!is.na(stat)) %>%
    dplyr::pull(Gene)

## Check the number of ligands and background genes
length(ligands)
## [1] 840
length(expressed_genes_receiver)
## [1] 16818
```

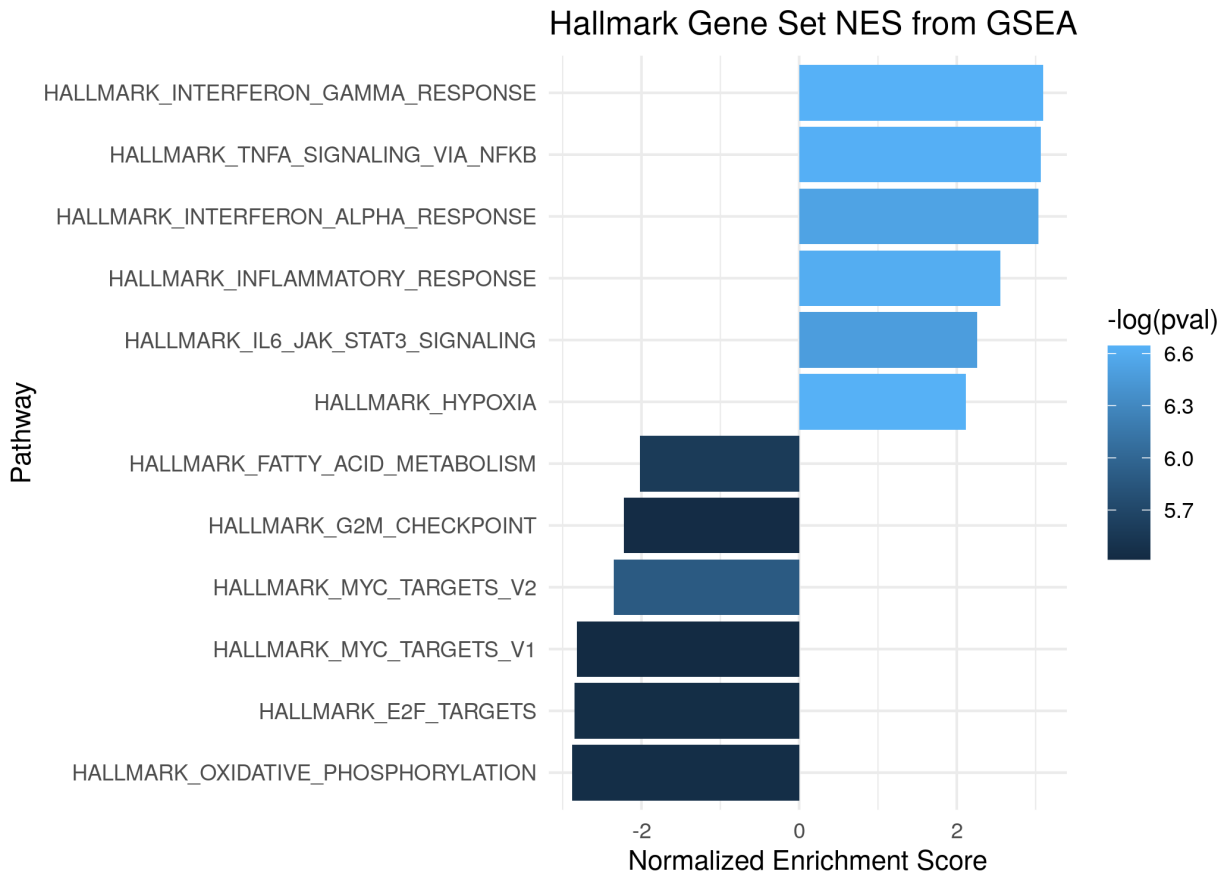
Step 2: Define the gene set of interest and a background of genes

To establish a gene set of interest, we perform a Gene set Enrichment analysis (GSEA) and we check among the most appealing overrepresanted signatures upon SARS-CoV-2 infection. We remove the differentially expressed ligands from this comparison.

```
ranks <- readRDS("Results/dds_results_CALU3vsCOV2.rds") %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(!(Gene %in% DDS_CALU3_ligands)) %>%
    dplyr::filter(!is.na(stat)) %>%
    dplyr::pull(stat, name=Gene)

# immunologic_signatures <- gmtPathways("NicheNet_Omnipath/c7.all.v7.1.symbols.gmt")
hallmarlk_signatures <- gmtPathways("h.all.v7.1.symbols.gmt")
# go_signatures <- gmtPathways("NicheNet_Omnipath/c5.bp.v7.1.symbols.gmt")

fgseaRes <- fgsea(hallmarlk_signatures, ranks,  nperm=1000)
# Testing only one pathway is implemented in a more efficient manner

SignificantResults <- fgseaRes %>%
    dplyr::filter(padj < 0.01) %>%
    dplyr::arrange(desc(NES)) %>%
    dplyr::top_n(12, abs(NES))
SignificantResults
##                              pathway        pval        padj         ES
##  1: HALLMARK_INTERFERON_GAMMA_RESPONSE 0.001302083 0.005199667  0.8627654
##  2:   HALLMARK_TNFA_SIGNALING_VIA_NFKB 0.001319261 0.005199667  0.8608318
##  3: HALLMARK_INTERFERON_ALPHA_RESPONSE 0.001461988 0.005199667  0.9172465
##  4:     HALLMARK_INFLAMMATORY_RESPONSE 0.001347709 0.005199667  0.7274370
##  5:   HALLMARK_IL6_JAK_STAT3_SIGNALING 0.001533742 0.005199667  0.7126008
##  6:                   HALLMARK_HYPOXIA 0.001307190 0.005199667  0.5893036
##  7:     HALLMARK_FATTY_ACID_METABOLISM 0.003787879 0.007407407 -0.5010564
##  8:            HALLMARK_G2M_CHECKPOINT 0.004385965 0.007407407 -0.5370578
##  9:            HALLMARK_MYC_TARGETS_V2 0.002793296 0.007407407 -0.6827875
## 10:            HALLMARK_MYC_TARGETS_V1 0.004444444 0.007407407 -0.6785459
## 11:               HALLMARK_E2F_TARGETS 0.004329004 0.007407407 -0.6829123
## 12: HALLMARK_OXIDATIVE_PHOSPHORYLATION 0.004310345 0.007407407 -0.6946604
##           NES nMoreExtreme size                                leadingEdge
```

```
##  1:  3.097636           0  174     OAS2,IFIT1,RSAD2,IFIT2,IFIT3,TNFAIP3,...
##  2:  3.062342           0  161 IFIT2,TNFAIP3,ATF3,PPP1R15A,NFKBIA,IFIH1,...
##  3:  3.037484           0   89        RSAD2,IFIT2,IFIT3,MX1,IFIH1,TXNIP,...
##  4:  2.555097           0  148       NFKBIA,IRF1,LAMP3,IFITM1,KLF6,RTP4,...
##  5:  2.256719           0   67         IRF1,STAT2,MAP3K8,STAT1,JUN,PIM1,...
##  6:  2.114591           0  173  TNFAIP3,ATF3,PPP1R15A,TIPARP,DUSP1,STC2,...
##  7: -2.019464           0  146        ACAT2,DHCR24,NSDHL,FASN,NTHL1,MIF,...
##  8: -2.225125           0  190         KPNA2,MCM5,SQLE,HSPA8,MCM6,LMNB1,...
##  9: -2.352049           0   58         TMEM97,MCM5,PHB,DCTPP1,PLK1,MCM4,...
## 10: -2.821150           0  193           KPNA2,MCM5,PHB,MCM6,SRSF2,NME1,...
## 11: -2.847626           0  195       KPNA2,MCM5,MXD3,SPAG5,NCAPD2,POLD1,...
## 12: -2.876772           0  184      MAOB,POLR2F,COX8A,LDHB,VDAC3,NDUFB2,...
```

```
plot_enrichment <- ggplot(SignificantResults, aes(reorder(pathway, NES), NES)) +
  geom_col(aes(fill=-log(pval))) +
  coord_flip() +
  labs(x="Pathway", y="Normalized Enrichment Score",
       title="Hallmark Gene Set NES from GSEA") +
  theme_minimal()
plot_enrichment
```




```
saveRDS(SignificantResults,file = "Results/Enrichment_Significant_Results.rds")
```

One of the most interesting results is inflamatory response. So, we define the leading edge genes involved in the inflamatory response as the target genes, i.e. we want to see how likely is that the secreted ligands have an effect in this inflamatory response.

```
## I am going to check with Inflamatory genes
InflamatoryGenes <- SignificantResults %>%
    dplyr::filter(pathway == "HALLMARK_INFLAMMATORY_RESPONSE") %>%
    dplyr::pull(leadingEdge) %>% unlist()

## We chech that there are no upregulated ligands here.
intersect(DDS_CALU3_ligands,InflamatoryGenes)
## character(0)

geneset_oi <- InflamatoryGenes[InflamatoryGenes %in% rownames(ligand_target_matrix)]

head(geneset_oi)
## [1] "NFKBIA" "IRF1"   "IFITM1" "KLF6"   "RTP4"   "IRAK2"
background_expressed_genes <- expressed_genes_receiver %>%
    .[. %in% rownames(ligand_target_matrix)]
head(background_expressed_genes)
## [1] "SAMD11"   "NOC2L"    "ISG15"    "AGRN"     "TNFRSF18" "SDF4"
```

Step 3: Define a set of potential ligands

As potentially active ligands, we will use ligands that are 1) Over-expressed in CALU3 after SARS-CoV-2 infection and 2) can bind a (putative) receptor expressed by malignant cells. Putative ligand-receptor links were gathered from Omnipath ligand-receptor data sources.

```
receptors <- unique(lr_network$to)
expressed_receptors <- intersect(receptors,expressed_genes_receiver)

lr_network_expressed <- lr_network %>%
    filter(from %in% DDS_CALU3_ligands & to %in% expressed_receptors)
head(lr_network_expressed)
## # A tibble: 6 x 4
##   from  to    source         database
##   <chr> <chr> <chr>          <chr>
## 1 CXCL1 CXCR2 kegg_cytokines kegg
## 2 CXCL2 CXCR2 kegg_cytokines kegg
## 3 CXCL3 CXCR2 kegg_cytokines kegg
## 4 CXCL5 CXCR2 kegg_cytokines kegg
## 5 CCL20 CCR6  kegg_cytokines kegg
## 6 CCL17 CCR4  kegg_cytokines kegg
```

This ligand-receptor network contains the expressed ligand-receptor interactions. As potentially active ligands for the NicheNet analysis, we will consider the ligands from this network.

```
potential_ligands <- lr_network_expressed %>% pull(from) %>% unique()
head(potential_ligands)
## [1] "CXCL1" "CXCL2" "CXCL3" "CXCL5" "CCL20" "CCL17"
```

Step 4: Perform NicheNet's ligand activity analysis on the gene set of interest

In this section, we calculate the ligand activity of each ligand, or in other words, we will assess how well each over-expressed ligand after viral infection can predict the inflatmatory response gene set compared to the background of expressed genes (predict whether a gene belongs to the inflatmatory response program or not).

```
ligand_activities <- predict_ligand_activities(
    geneset = geneset_oi,
```

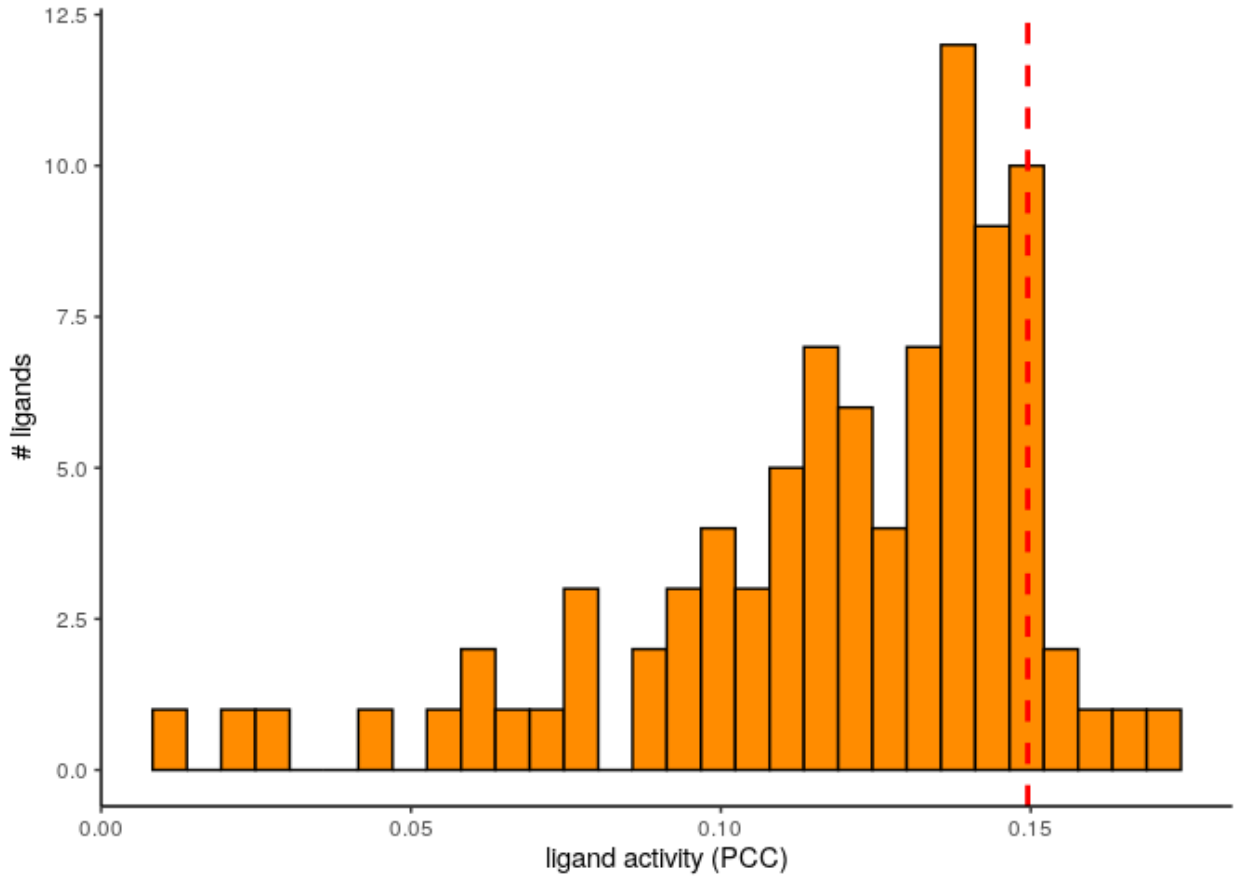
```
    background_expressed_genes = background_expressed_genes,
    ligand_target_matrix = ligand_target_matrix,
    potential_ligands = potential_ligands)
```

We will rank the ligands based on their pearson correlation coefficient. This allows us to prioritize inflamory response-regulating ligands.

```
ligand_activities %>% arrange(-pearson)
## # A tibble: 89 x 4
##    test_ligand auroc   aupr pearson
##    <chr>       <dbl>  <dbl>   <dbl>
##  1 IL23A       0.742 0.0693   0.173
##  2 TNF         0.753 0.0604   0.165
##  3 TNFSF13B    0.732 0.0568   0.159
##  4 IL1A        0.712 0.0532   0.155
##  5 LAMA2       0.740 0.0597   0.152
##  6 ICAM4       0.731 0.0645   0.151
##  7 L1CAM       0.735 0.0645   0.151
##  8 CXCL9       0.742 0.0771   0.151
##  9 NPPB        0.724 0.0721   0.151
## 10 INHBA       0.677 0.0591   0.150
## # ... with 79 more rows
best_upstream_ligands <- ligand_activities %>%
    top_n(12, pearson) %>%
    arrange(-pearson) %>%
    pull(test_ligand)
head(best_upstream_ligands)
## [1] "IL23A"    "TNF"      "TNFSF13B" "IL1A"     "LAMA2"    "ICAM4"
```

We see here that the performance metrics indicate that the 12 top-ranked ligands can predict the inflamatory genes reasonably, this implies that ranking of the ligands might be accurate as shown in our study. However, it is possible that for some gene sets, the target gene prediction performance of the top-ranked ligands would not be much better than random prediction. In that case, prioritization of ligands will be less trustworthy.

```
# show histogram of ligand activity scores
p_hist_lig_activity = ggplot(ligand_activities, aes(x=pearson)) +
  geom_histogram(color="black", fill="darkorange")  +
  # geom_density(alpha=.1, fill="orange") +
  geom_vline(aes(xintercept=min(ligand_activities %>% top_n(12, pearson) %>%
    pull(pearson))), color="red", linetype="dashed", size=1) +
  labs(x="ligand activity (PCC)", y = "# ligands") +
  theme_classic()
p_hist_lig_activity
```




```
saveRDS(ligand_activities,file = "Results/LigandActivityScoreDistribution.rds")
```

Step 5: Infer target genes of top-ranked ligands and visualize in a heatmap

Now we will show how you can look at the regulatory potential scores between ligands and target genes of interest. In this case, we will look at links between top-ranked ligands regulating inflamatory response genes. In the ligand-target heatmaps, we show here regulatory potential scores for interactions between the 12 top-ranked ligands and following target genes: genes that belong to the gene set of interest and to the 250 most strongly predicted targets of at least one of the 12 top-ranked ligands (the top 250 targets according to the general prior model, so not the top 250 targets for this dataset). Consequently, genes of your gene set that are not a top target gene of one of the prioritized ligands, will not be shown on the heatmap.

```
active_ligand_target_links_df <- best_upstream_ligands %>%
    lapply(get_weighted_ligand_target_links,
           geneset = geneset_oi,
           ligand_target_matrix = ligand_target_matrix,
           n = 250) %>%
    bind_rows()
nrow(active_ligand_target_links_df)
## [1] 179
head(active_ligand_target_links_df)
## # A tibble: 6 x 3
##   ligand target weight
##   <chr>  <chr>   <dbl>
## 1 IL23A  CD69   0.0239
## 2 IL23A  CDKN1A 0.0549
```

```
## 3 IL23A  F3     0.0314
## 4 IL23A  IFITM1 0.0185
## 5 IL23A  IL18   0.0232
## 6 IL23A  IL4R   0.0197
```

For visualization purposes, we adapted the ligand-target regulatory potential matrix as follows. Regulatory potential scores were set as 0 if their score was below a predefined threshold, which was here the 0.10 quantile of scores of interactions between the 10 top-ranked ligands and each of their respective top targets (see the ligand-target network defined in the data frame).

```
active_ligand_target_links <- prepare_ligand_target_visualization(
    ligand_target_df = active_ligand_target_links_df,
    ligand_target_matrix = ligand_target_matrix,
    cutoff = 0.10)
nrow(active_ligand_target_links_df)
## [1] 179
head(active_ligand_target_links_df)
## # A tibble: 6 x 3
##   ligand target weight
##   <chr>  <chr>   <dbl>
## 1 IL23A  CD69   0.0239
## 2 IL23A  CDKN1A 0.0549
## 3 IL23A  F3     0.0314
## 4 IL23A  IFITM1 0.0185
## 5 IL23A  IL18   0.0232
## 6 IL23A  IL4R   0.0197
```

The putatively active ligand-target links will now be visualized in a heatmap. The order of the ligands accord to the ranking according to the ligand activity prediction.

```
order_ligands <-
    intersect(best_upstream_ligands, colnames(active_ligand_target_links)) %>%
    rev()
order_targets <- active_ligand_target_links_df$target %>%
    unique()
vis_ligand_target <- active_ligand_target_links[order_targets,order_ligands] %>%
    t()
p_ligand_target_network <- vis_ligand_target %>%
    make_heatmap_ggplot("Prioritized ligands","Inflamatory Related genes",
        color = "blue",legend_position = "top", x_axis_position = "top",
        legend_title = "Regulatory potential") +
        scale_fill_gradient2() +
            # ) +
    theme(axis.text.x = element_text(face = "italic"))
p_ligand_target_network
```


```
saveRDS(vis_ligand_target,file = "Results/Ligand_Target_Matrix.rds")
```

Note that the choice of these cutoffs for visualization is quite arbitrary. We recommend users to test several cutoff values.

If you would consider more than the top 250 targets based on prior information, you will infer more, but less confident, ligand-target links; by considering less than 250 targets, you will be more stringent.

If you would change the quantile cutoff that is used to set scores to 0 (for visualization purposes), lowering this cutoff will result in a more dense heatmap, whereas highering this cutoff will result in a more sparse heatmap.

Follow-up analysis 1: Ligand-receptor network inference for top-ranked ligands

One type of follow-up analysis is looking at which receptors can potentially bind to the prioritized ligands.

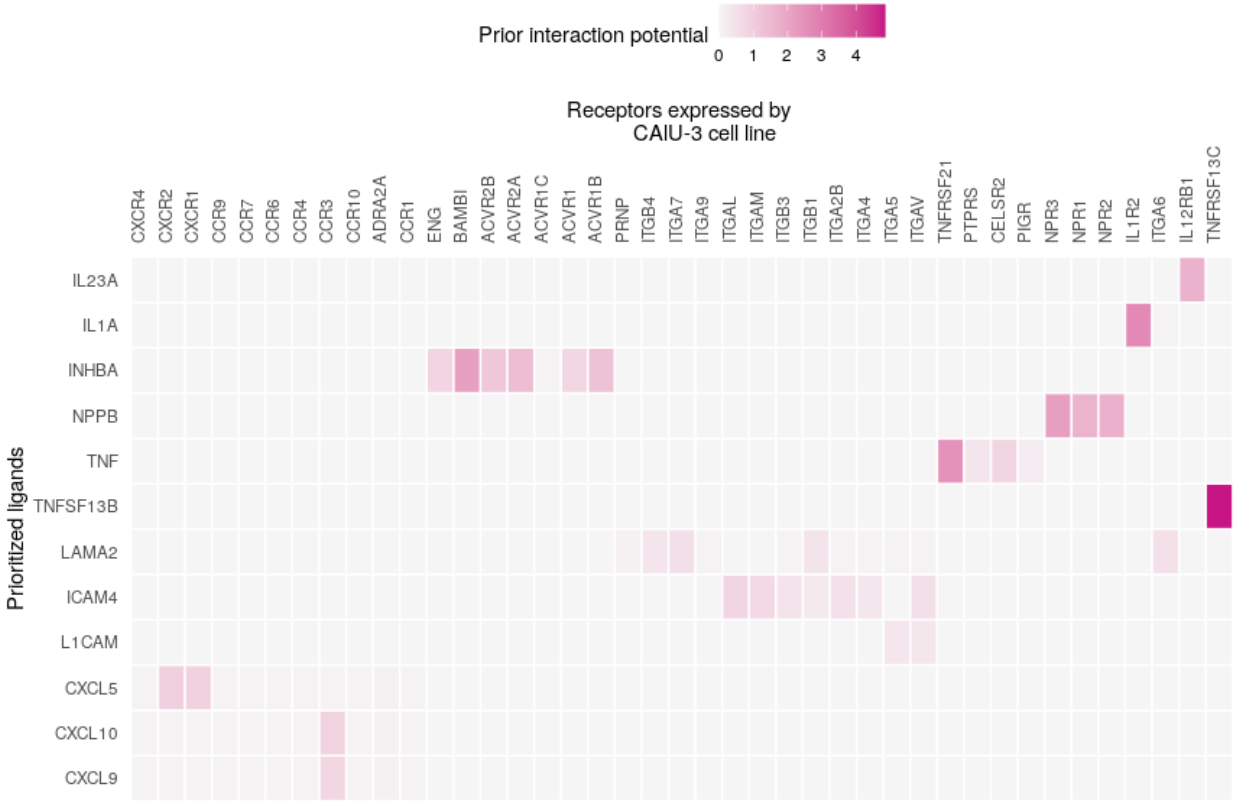
So, we will now infer the predicted ligand-receptor interactions of the top-ranked ligands and visualize these in a heatmap.

```
## get the ligand-receptor network of the top-ranked ligands
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()
```

```
## get the weights of the ligand-receptor interactions as used in the NicheNet model
weighted_networks <- readRDS("Results/weighted_networksWithSourceWeights.rds")

lr_network_top_df <- weighted_networks$lr_sig %>%
    filter(from %in% best_upstream_ligands & to %in% best_upstream_receptors)

## convert to a matrix
lr_network_top_df <- lr_network_top_df %>%
    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)

## perform hierarchical clustering to order the ligands and receptors
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]
```

Show a heatmap of the ligand-receptor interactions

```
vis_ligand_receptor_network <-
    lr_network_top_matrix[order_receptors, order_ligands_receptor]
p_ligand_receptor_network <- vis_ligand_receptor_network %>%
    t() %>%
    make_heatmap_ggplot("Prioritized ligands","Receptors expressed by
        CAlU-3 cell line", color = "mediumvioletred", x_axis_position = "top",
        legend_title = "Prior interaction potential")
p_ligand_receptor_network
```




```
saveRDS(vis_ligand_receptor_network,file = "Results/Ligand_Receptor_Matrix.rds")
```

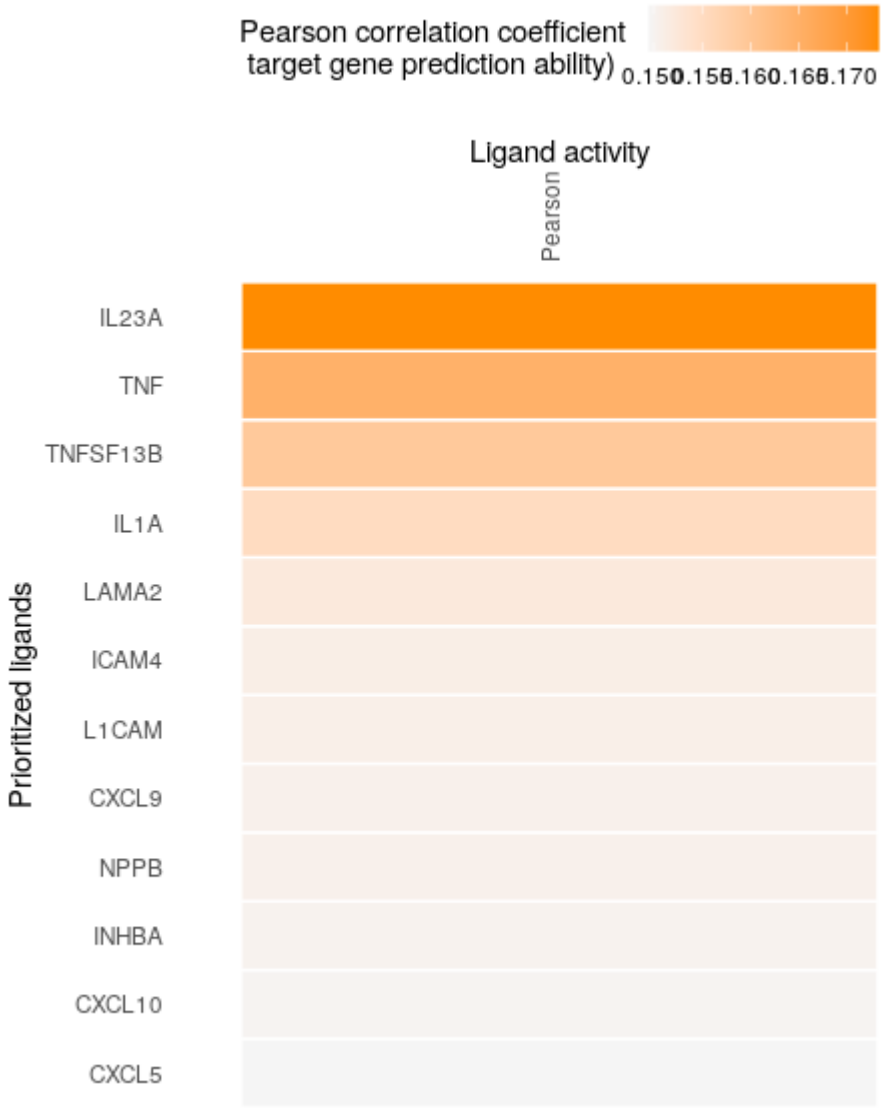
Follow-up analysis 2: Visualize expression of top-predicted ligands and their target genes in a combined heatmap

NicheNet only considers expressed ligands of sender cells, but does not take into account their expression for ranking the ligands. The ranking is purely based on the potential that a ligand might regulate the gene set of interest, given prior knowledge. Because it is also useful to further look into expression of ligands and their target genes, we demonstrate here how you could make a combined figure showing ligand activity, ligand expression, target gene expression and ligand-target regulatory potential.

```
library(RColorBrewer)
library(cowplot)
library(ggpubr)
```

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

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 \n target gene prediction ability)")
p_ligand_pearson
```



Prepare the ligand activity matrix

```
saveRDS(vis_ligand_pearson, file = "Results/ligand_Pearson.rds")
```

References

Browaeys, R., Saelens, W. & Saeys, Y. NicheNet: modeling intercellular communication by linking ligands to target genes. Nat Methods (2019) doi:10.1038/s41592-019-0667-5

Puram, Sidharth V., Itay Tirosh, Anuraag S. Parikh, Anoop P. Patel, Keren Yizhak, Shawn Gillespie, Christopher Rodman, et al. 2017. "Single-Cell Transcriptomic Analysis of Primary and Metastatic Tumor Ecosystems in Head and Neck Cancer." Cell 171 (7): 1611–1624.e24. https://doi.org/10.1016/j.cell.2017.10.044.

NicheNet Results Plot Arrangement

Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 15/06/2020

Results

This vignette contains the rearrangement of the plots obtained in the previous markdown (05_ligandActivity-Analysis.Rmd) for publication.

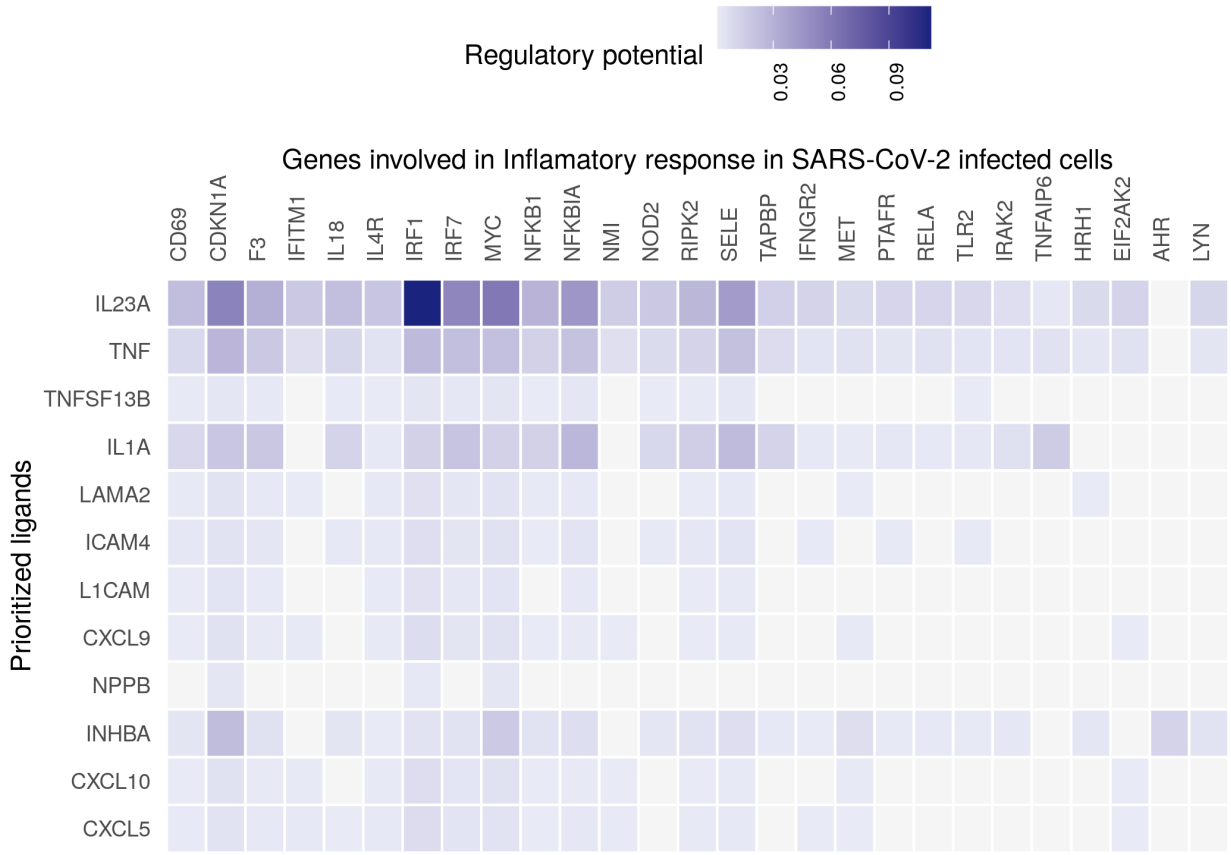
We first load the required libraries and read the previously generated results:

```
library(RColorBrewer)
library(cowplot)
library(ggpubr)
library(dplyr)
library(tibble)
library(tidyr)
library(gridExtra)

TargetExpression <- readRDS(file = "Results/target_expression.rds")
LigandTarget <- readRDS(file = "Results/Ligand_Target_Matrix.rds")
LigandReceptor <- readRDS(file = "Results/Ligand_Receptor_Matrix.rds")  %>%
    t()
LigandPearsonCor <- readRDS(file = "Results/ligand_Pearson.rds")
SignificantResults <- readRDS(file = "Results/Enrichment_Significant_Results.rds") %>%
  dplyr::mutate(pathway = gsub("HALLMARK_","", pathway))
ligand_activities <- readRDS(file = "Results/LigandActivityScoreDistribution.rds")
```

Ligand-Target Heatmap

```
LigandTarget_df = LigandTarget %>%
    data.frame() %>%
    rownames_to_column("y") %>%
    tbl_df() %>%
    gather(x, "score", -y) %>%
    mutate(y = factor(y, levels = rownames(LigandTarget), ordered = TRUE),
           x = factor(x, levels = colnames(LigandTarget), ordered = TRUE)) %>%
    mutate(score = ifelse(score == 0, NA, score))

plot_LigandTarget <- LigandTarget_df %>%
    ggplot(aes(x, y, fill = score)) +
    geom_tile(color = "white", size = 0.5) +
    scale_fill_gradient(low = "#E8EAF6", high = "#1A237E",
                        na.value = "whitesmoke")  +
    theme_minimal() +
    theme(panel.grid.minor = element_line(color = "transparent"),
          panel.grid.major = element_line(color = "transparent"),
          legend.position = "top",
          legend.text = element_text(size = 8, angle = 90, hjust = 1),
          axis.ticks = element_line(size = 0),
          axis.text.x.top = element_text(angle = 90, hjust = 0),
          axis.text.x = element_text(angle = 90, hjust = 1),
          axis.title = element_text(),
          axis.text.y = element_text()) +
          scale_x_discrete(position = "top") +
```

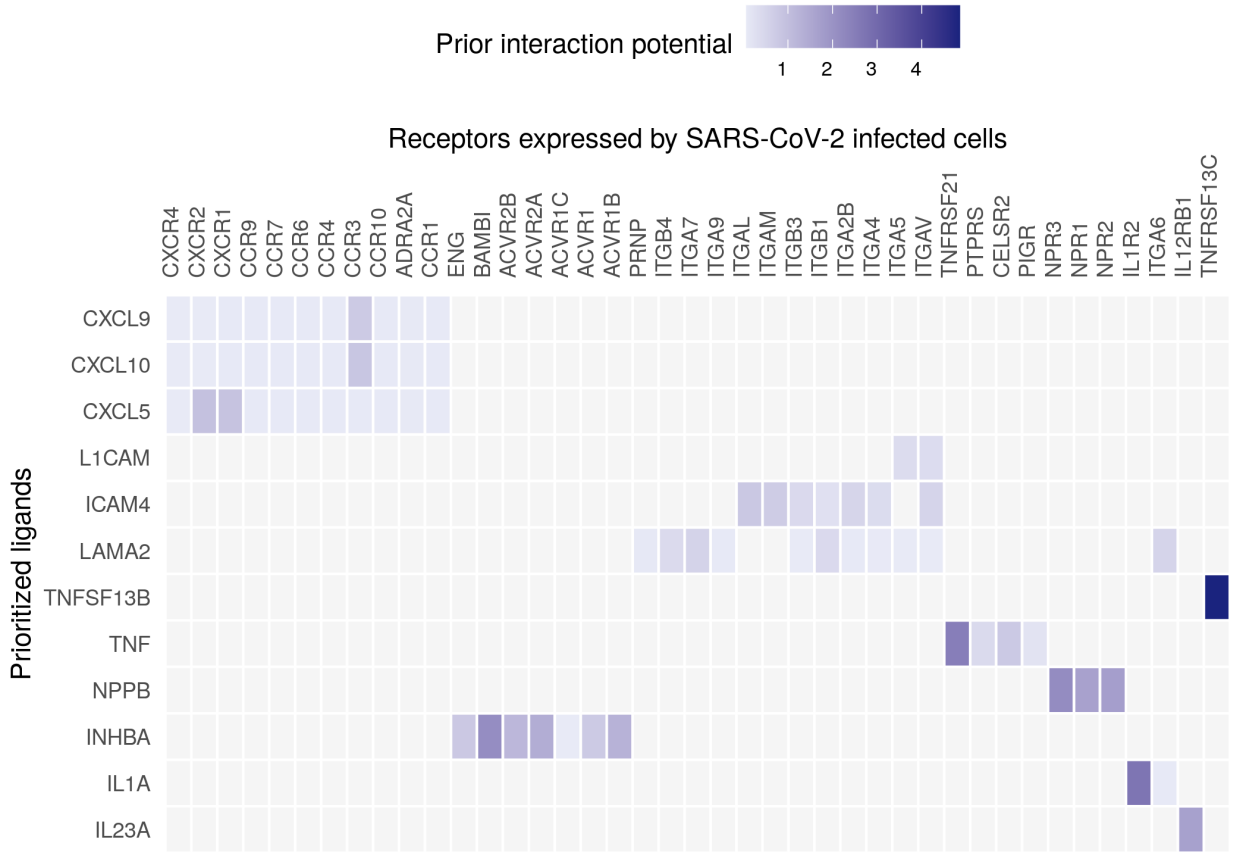
```
          xlab(paste0("Genes involved in Inflamatory response in SARS-CoV-2 infected cells")) +
          ylab(paste0("Prioritized ligands")) +
    labs(fill = "Regulatory potential")
plot_LigandTarget
```



Ligand-Receptor Heatmap

```
LigandReceptor_df = LigandReceptor %>%
    data.frame() %>%
    rownames_to_column("y") %>%
    tbl_df() %>%
    gather(x, "score", -y) %>%
    mutate(y = factor(y, levels = rev(rownames(LigandReceptor)), ordered = TRUE),
           x = factor(x, levels = colnames(LigandReceptor), ordered = TRUE)) %>%
    mutate(score = ifelse(score == 0, NA, score))

plot_LigandReceptor <- LigandReceptor_df %>%
    ggplot(aes(x, y, fill = score)) +
    geom_tile(color = "white", size = 0.5) +
    scale_fill_gradient(low = "#E8EAF6", high = "#1A237E",
                        na.value = "whitesmoke") +
    theme_minimal() +
    theme(panel.grid.minor = element_line(color = "transparent"),
          panel.grid.major = element_line(color = "transparent"),
```

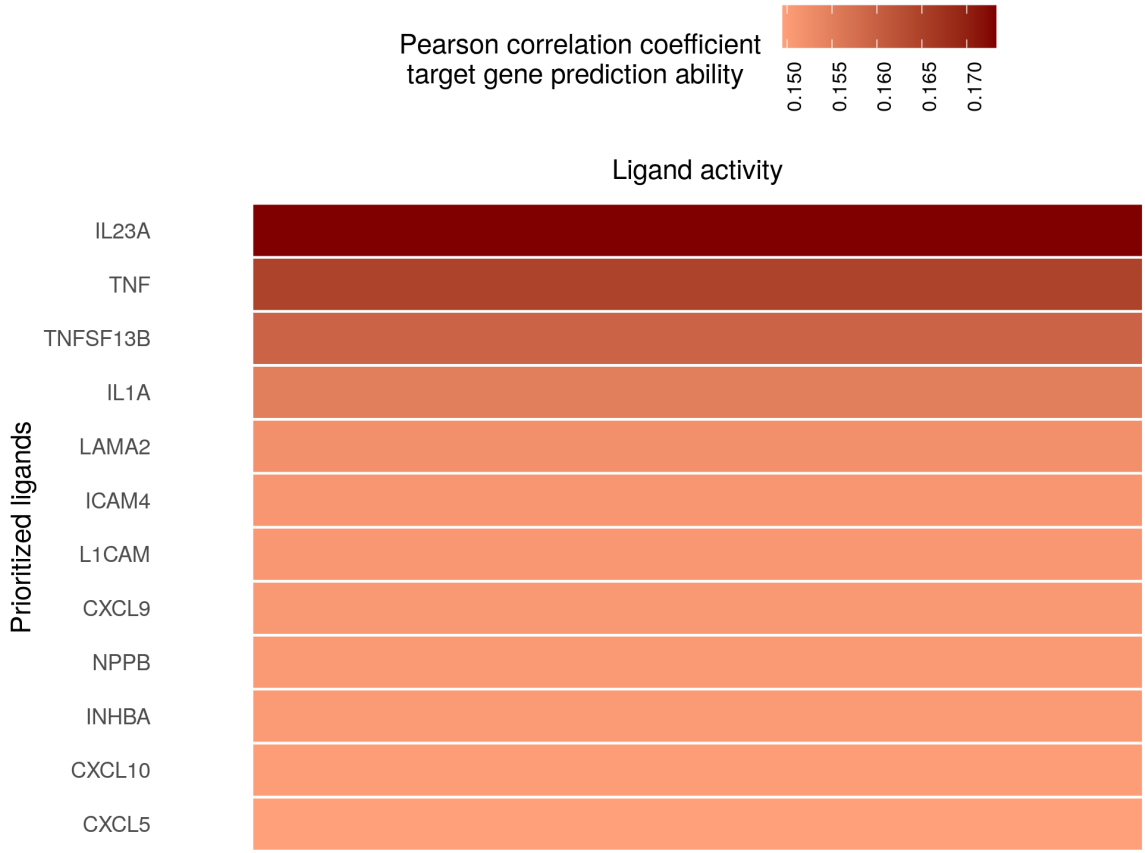
```
          legend.position = "top",
          legend.text = element_text(size = 8, hjust = 1),
          axis.ticks = element_line(size = 0),
          axis.text.x.top = element_text(angle = 90, hjust = 0),
          axis.text.x = element_text(angle = 90, hjust = 1),
          axis.title = element_text(),
          axis.text.y = element_text()) +
          scale_x_discrete(position = "top") +
          xlab(paste0("Receptors expressed by SARS-CoV-2 infected cells")) +
          ylab(paste0("Prioritized ligands")) +
    labs(fill = "Prior interaction potential")
plot_LigandReceptor
```



Ligand-Targets Pearson Correlation

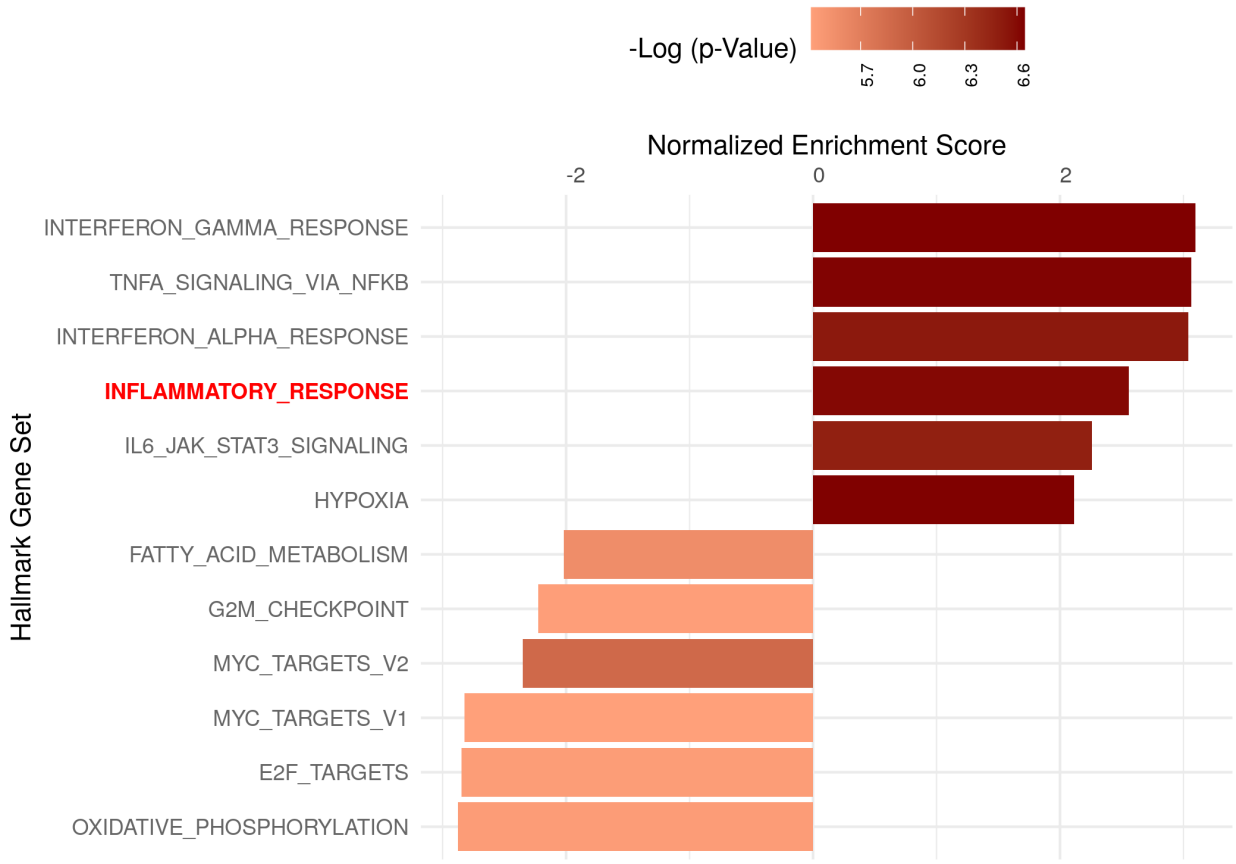
```
LigandPearsonCor_df = LigandPearsonCor %>%
    data.frame() %>%
    rownames_to_column("y") %>%
    tbl_df() %>%
    gather(x, "score", -y) %>%
    mutate(y = factor(y, levels = rownames(LigandPearsonCor), ordered = TRUE),
           x = factor(x, levels = colnames(LigandPearsonCor), ordered = TRUE)) %>%
    mutate(score = ifelse(score == 0, NA, score))
```

```
plot_LigandPearsonCor <- LigandPearsonCor_df %>%
    ggplot(aes(x, y, fill = score)) +
    geom_tile(color = "white", size = 0.5) +
    scale_fill_gradient(low = "#FFA07A", high = "#800000",
                        na.value = "whitesmoke") +
    theme_minimal() +
    theme(panel.grid.minor = element_line(color = "transparent"),
          panel.grid.major = element_line(color = "transparent"),
          legend.position = "top",
          legend.text = element_text(size = 8, hjust = 1, angle = 90),
          axis.ticks = element_line(size = 0),
          axis.text.x.top = element_text(angle = 90, hjust = 0),
          axis.text.x = element_text(angle = 90, hjust = 1),
          axis.title = element_text(),
          axis.text.y = element_text()) +
          scale_x_discrete(position = "top" ,labels = "") +
          xlab(paste0("Ligand activity")) +
          ylab(paste0("Prioritized ligands")) +
    labs(fill = "Pearson correlation coefficient \n target gene prediction ability")
plot_LigandPearsonCor
```



Enrichment of SARS-CoV-2 Infected cells

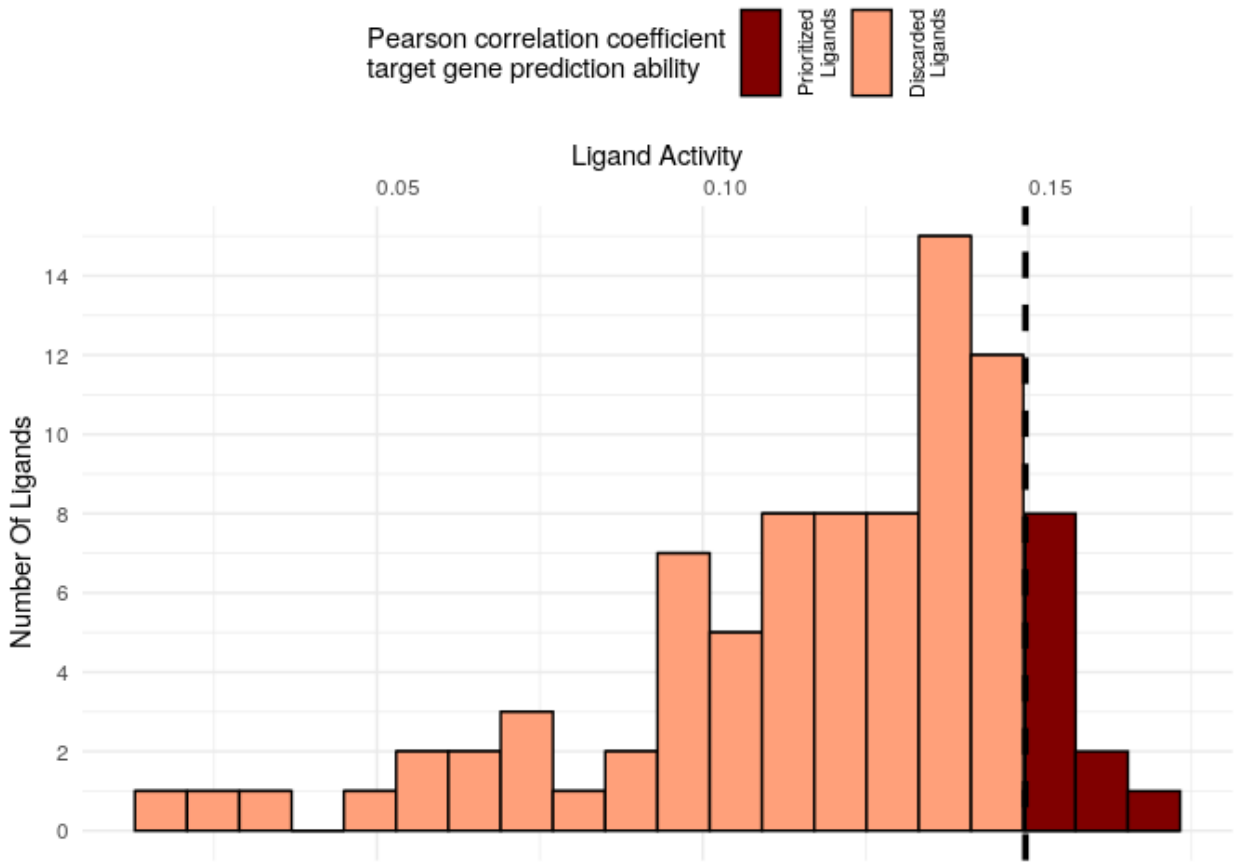
```
plot_enrichment <- ggplot(SignificantResults, aes(NES, reorder(pathway, NES))) +
  geom_col(aes(fill=-log(pval))) +
  # coord_flip() +
  # labs(x="Pathway", y="Normalized Enrichment Score") +
  scale_fill_gradient(low = "#FFA07A", high = "#800000",
                      na.value = "whitesmoke") +
  theme_minimal() +
  theme(legend.position = "top",
        legend.text = element_text(size = 7, hjust = 1, angle = 90),
        axis.text.x = element_text(angle = 0, hjust = 0),
        axis.text.y = element_text(angle = 0, hjust = 1,
          colour = ifelse(levels(reorder(SignificantResults$pathway,
                SignificantResults$NES)) == "INFLAMMATORY_RESPONSE" ,
                "red", "grey40"),
            face = ifelse(levels(reorder(SignificantResults$pathway,
                SignificantResults$NES)) == "INFLAMMATORY_RESPONSE",
                "bold", "plain")),
 ) +
        ylab(paste0("Hallmark Gene Set")) +
        xlab(paste0("Normalized Enrichment Score")) +
        labs(fill = "-Log (p-Value)") +
  scale_x_continuous(position = "top")

        #
        #axis.text.x = element_text(angle = 0, hjust = 1),
        #axis.title = element_text(),
        #axis.text.y = element_text(angle = 0, hjust = 1)) +
        #scale_y_discrete(position = "top") +
        #xlab(paste0("Hallmark Gene Signatures")) +
        #ylab(paste0("Normalized Enrichment Score")) +
        #labs(fill = "-log(p-value)")
plot_enrichment
```



Ligand Activity: Pearson Correlation coefficients distribution

```
cut_off_Value <- min(ligand_activities %>% top_n(12, pearson) %>% pull(pearson))
ligand_activities_color <- ligand_activities %>%
  mutate(pcc_color = ifelse(pearson <= cut_off_Value, "#FFA07A", "#800000")) %>%
  mutate(pcc_color = as.factor(pcc_color))

bins <- seq(min(ligand_activities_color$pearson),
            max(ligand_activities_color$pearson),
            length.out = 21)
bins <- c(bins, cut_off_Value) %>% sort()

p_hist_lig_activity = ggplot(
  ligand_activities_color,
  aes(x=pearson, fill=pcc_color)) +
  geom_histogram(color= "black", breaks = bins) +
  geom_vline(aes(xintercept=min(ligand_activities %>% top_n(12, pearson) %>%
    pull(pearson))), color="black", linetype="dashed", size=1.25) +
  theme_minimal() +
  ylab(paste0("Number Of Ligands")) +
        xlab(paste0("Ligand Activity")) +
  labs(fill = "Pearson correlation coefficient\ntarget gene prediction ability") +
  scale_x_continuous(position = "top") +
  scale_y_continuous(breaks = seq(0, 14, by = 2)) +
  scale_fill_manual(labels = c("Prioritized\nLigands", "Discarded\nLigands"),
```

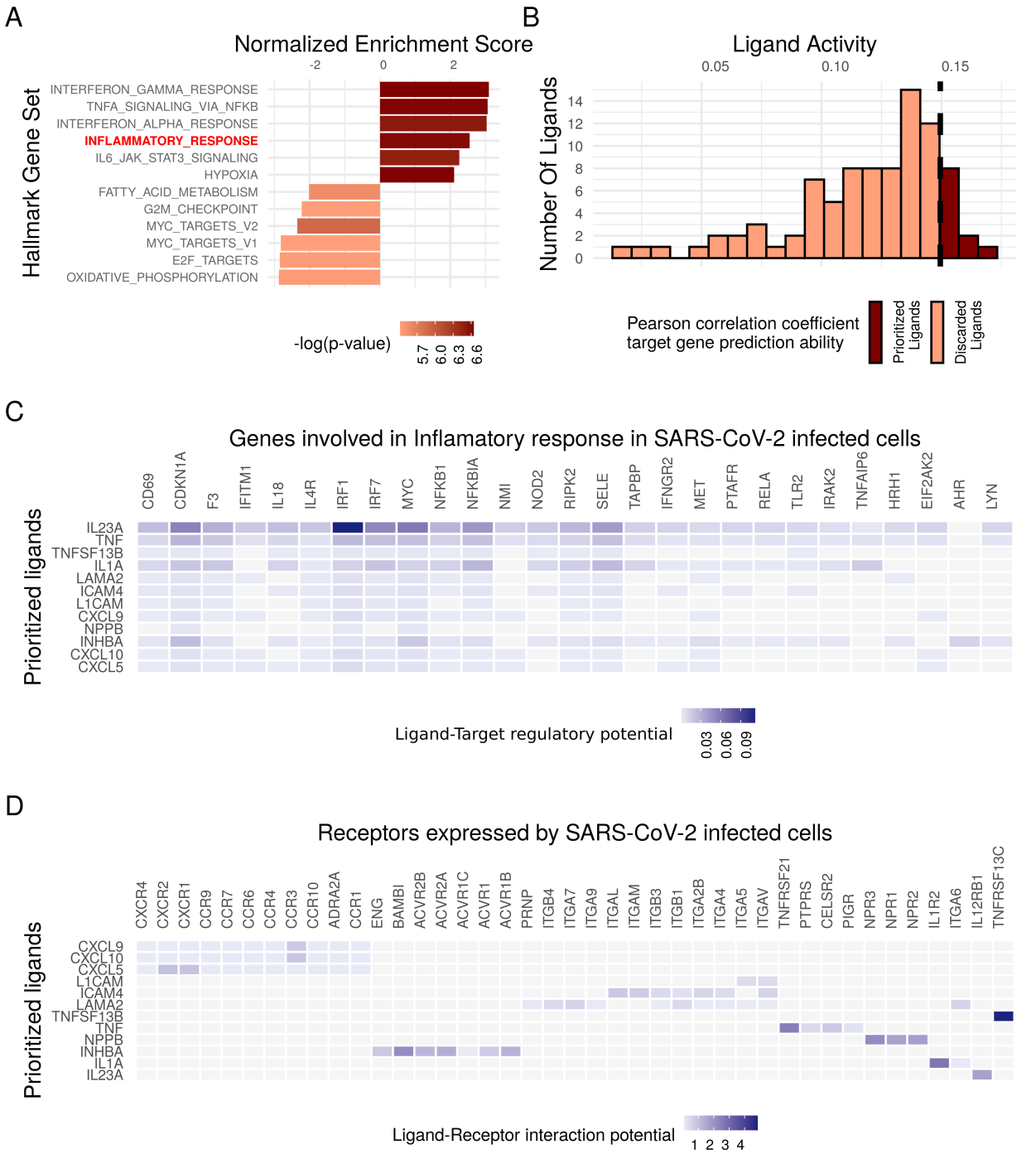
```
                    values=levels(ligand_activities_color$pcc_color)) +
   theme(legend.position = "top",
        legend.text = element_text(size = 8, hjust = 1, angle = 90),
        axis.text.x = element_text(angle = 0, hjust = 0),
        axis.text.y = element_text(angle = 0, hjust = 1))
p_hist_lig_activity
```



Combine plots in a single Figure

```
spaceLegend <- 0.5
lay <- rbind(c(1,1,2,2),
             c(3,3,3,3),
             c(4,4,4,4))

plot_enrichment_final <-
  plot_enrichment +
      theme(legend.position = "bottom", axis.ticks = element_blank()) +
        theme(axis.title.x = element_text(),
              axis.text = element_text(size = 6)) +
        theme(legend.title = element_text(size=8.5),
          legend.text = element_text(size=7, angle = 90),
          legend.key.size = unit(spaceLegend, "lines")) +
          labs(fill = "-log(p-value)", tag="A")
```

```
p_hist_lig_activity_final <-
  p_hist_lig_activity +
    theme(legend.position = "bottom", axis.ticks = element_blank()) +
    theme(axis.title.x = element_text(),
          axis.text = element_text(size = 7)) +
    theme(legend.title = element_text(size=8.5),
          legend.text = element_text(size=7, angle = 90),
          legend.key.size = unit(spaceLegend, "lines")) +
          labs(tag = "B")


plot_LigandTarget_final <-
    plot_LigandTarget +
        theme(legend.position = "bottom", axis.ticks = element_blank()) +
        theme(axis.title.x = element_text(),
              axis.text = element_text(size = 7)) + # ylab("") +
        theme(legend.title = element_text(size=8,5),
          legend.text = element_text(size=7, angle = 90),
          legend.key.size = unit(spaceLegend, "lines")) +
          labs(fill = "Ligand-Target regulatory potential", tag = "C")

plot_LigandReceptor_final <-
    plot_LigandReceptor +
        theme(legend.position = "bottom", axis.ticks = element_blank(),
              axis.text = element_text(size = 7)) +
        # ylab("") +
        theme(legend.title = element_text(size=8.5),
          legend.text = element_text(size=7, angle = 0),
          legend.key.size = unit(spaceLegend, "lines")) +
          labs(fill = "Ligand-Receptor interaction potential", tag = "D")

figures_with_legend <-
    grid.arrange(plot_enrichment_final, p_hist_lig_activity_final,
    plot_LigandTarget_final, plot_LigandReceptor_final,
    layout_matrix = lay)
```




```
figures_with_legend
## TableGrob (3 x 4) "arrange": 4 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-2) arrange gtable[layout]
## 2 2 (1-1,3-4) arrange gtable[layout]
## 3 3 (2-2,1-4) arrange gtable[layout]
## 4 4 (3-3,1-4) arrange gtable[layout]
```

```
ggsave(filename = "Results/MegeHeatmaps.eps", plot=figures_with_legend,
       device = "eps",  dpi = 600, limitsize = FALSE, width=6, height=8,
       units = c("in"))
ggsave(filename = "Results/MegeHeatmaps.png", plot=figures_with_legend,
       device = "png",  dpi = 600, limitsize = FALSE, width=6, height=8,
       units = c("in"))
ggsave(filename = "Results/MegaHeatmaps.svg", plot=figures_with_legend,
       device = "svg",  dpi = 600, limitsize = FALSE, width=6, height=8,
       units = c("in"))
```

NicheNet Results: Ligand-Targets interesting paths

Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 15/06/2020

Results

This vignette contains the code to generate the most interesting path going from the most relevant ligands to their targets going throught receptors, signaling intermediates and TFs.

We first load the required libraries and read the previously generated results:

```
library(nichenetr)
library(tidyverse)
library(igraph)
library(OmnipathR)

weighted_networks <- readRDS("Results/weighted_networksWithSourceWeights.rds")
ligand_tf_matrix <- readRDS("Results/ligand_target_matrixWithweights.rds")

lr_network <- readRDS("OmniNetworks_NNformat/lr_Network_Omnipath.rds")
sig_network <- readRDS("OmniNetworks_NNformat/sig_Network_Omnipath.rds")
gr_network <- readRDS("OmniNetworks_NNformat/gr_Network_Omnipath.rds")
```

We then select the most relevant ligands, receptors and targets.

```
ligands <- c("IL23A","IL1A","TNF","TNFSF13B","CXCL9","CXCL10","CXCL5")
receptors <- c("CXCR2","CXCR1","CCR3","IL1R2","IL12RB1","TNFRSF21",
  "TNFRSF13C")
targets <-
    c("CDKN11","F3","IRF1", "IRF7", "MYC", "NFKB1", "NFKBIA", "SELE", "RIPK2",
      "NOD2", "IL18", "CD69", "IL4R")
```

We then select the ligand receptor interactions and the Gene regulatory interactions involving the genes defined above.

```
LigRec_Interactions <- lr_network %>%
    dplyr::filter(from %in% ligands & to %in% receptors) %>%
    dplyr::distinct(from,to) %>%
    dplyr::mutate(type = "LigReceptor")

GeneRegulatory_Interactions <- weighted_networks$gr %>%
    dplyr::filter(!(from %in% c(ligands,receptors,targets))) %>%
    dplyr::filter(to %in% targets)  %>%
    dplyr::distinct(from,to,.keep_all = TRUE) %>%
    dplyr::mutate(type = "GeneRegulation")

## I am going to select the TFs with the best weights in the interactions
## related to the target genes. K defines the number of TFs to be selected.
k <- 20
TFs <- GeneRegulatory_Interactions  %>%
    group_by(from) %>%
    mutate(weight = sum(weight)) %>%
    ungroup() %>%
    arrange(desc(weight)) %>%
    pull(from) %>%
    unique %>%
    .[1:k]
```

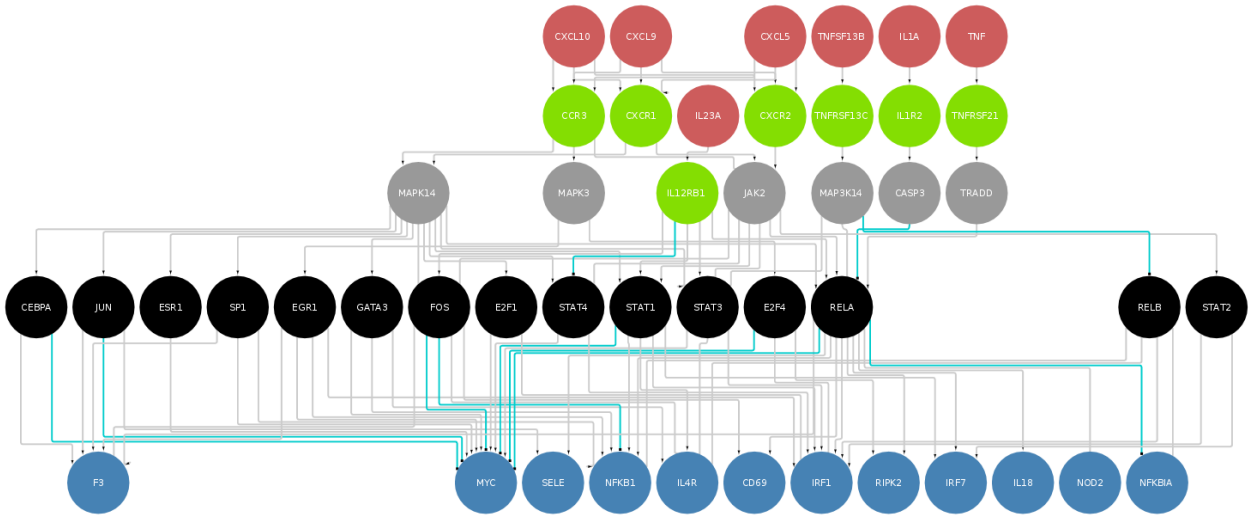
What are the most important TFs in this context? I now compute the distance between receptors and TFs in the signaling network.

```
sig_network_noDup <-sig_network %>%
    dplyr::distinct(from,to)

sig_network_igraph <-
    graph_from_data_frame(sig_network_noDup, directed = TRUE)

Distance_Receptor_TF <-
    distances(sig_network_igraph, receptors, TFs, mode = c("out"))

## For every row (Receptors) we select the closest column (TF)
receptorTO_TFs_graph <- make_empty_graph(n = 0, directed = TRUE)

for (i in seq(nrow(Distance_Receptor_TF))){
    receptor <- rownames(Distance_Receptor_TF)[i]
    closestTF <-
        names(which(Distance_Receptor_TF[i,] == min(Distance_Receptor_TF[i,])))
    sp <- shortest_paths(sig_network_igraph, receptor, closestTF,
                         output = c("both"))
    temporal_graph <- do.call(graph.union, lapply(sp$epath, function(x)
        subgraph.edges(sig_network_igraph,eids = x)))

    receptorTO_TFs_graph <- graph.union(receptorTO_TFs_graph,temporal_graph)
}


## For clarity also of the flow, we also remove signaling interactions to the
## receptors
SignalingInteractions <- igraph::as_data_frame(receptorTO_TFs_graph) %>%
    mutate(type="Signaling") %>%
    dplyr::filter(!(to %in% receptors))

## We remove those interactions from TFs that are connected to the ligands
## via the signaling network.
GeneRegulatory_Interactions <-
    dplyr::filter(GeneRegulatory_Interactions,from %in%
                unique(c(SignalingInteractions$from, SignalingInteractions$to)))
```

We generate the final graph and export it to a cytoscape compatible format.

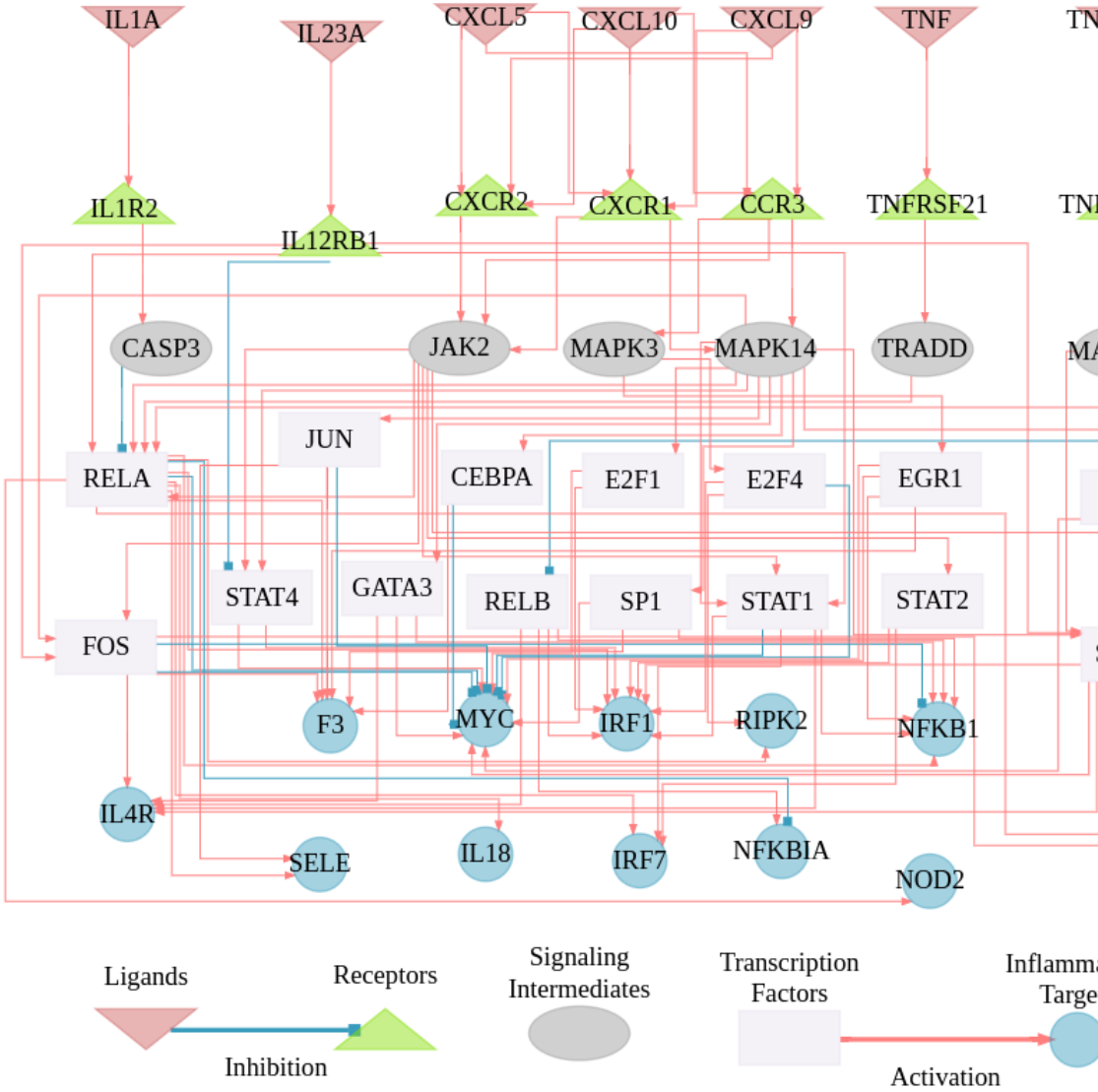
```
## We generate the final graph
Final_graph <- dplyr::bind_rows(LigRec_Interactions,
    SignalingInteractions,
    GeneRegulatory_Interactions) %>%
    dplyr::distinct(from, to, .keep_all = TRUE)

## We generate a table with the node information (attributes)
All_genes <- unique(c(Final_graph$from,Final_graph$to))

Table_attributes <-
    tibble(Gene = All_genes, Role = character(length = length(All_genes)))

Table_attributes <- dplyr::mutate(Table_attributes, Role =
```

```
    ifelse(Gene %in% ligands, "ligand",
    ifelse(Gene %in% receptors, "receptor",
    ifelse(Gene %in% targets, "target",
    ifelse(Gene %in% TFs, "transcriptionFactor", "SignalingMediator")))))

### Check with Omnipath the sign of the interactions!!

AllInteractionsOmnipath <- import_all_interactions() %>%
    dplyr::select(source_genesymbol, target_genesymbol, consensus_direction,
                  consensus_stimulation, consensus_inhibition)  %>%
    dplyr::distinct(source_genesymbol, target_genesymbol, .keep_all = TRUE)

Final_graph <- dplyr::left_join(Final_graph, AllInteractionsOmnipath,
    by = c("from" = "source_genesymbol", "to" = "target_genesymbol")) %>%
    dplyr::mutate(interaction_sign = ifelse(consensus_inhibition == 1, -1 ,1)) %>%
    dplyr::select(c(from,to,type, interaction_sign)) %>%
    dplyr::mutate_all(~replace(., is.na(.), 1))

### Export to cytoscape:

write_tsv(Final_graph, "Results/Cytoscape/Network_ToDraw.txt")
write_tsv(Table_attributes, "Results/Cytoscape/annotation_table.txt")
```




```
Dot_text <- NULL
Dot_text <- c(Dot_text, "digraph {")
Dot_text <- c(Dot_text, "")
Dot_text <- c(Dot_text, "node [fixedsize = \"true\" ]")


for (i in seq_len(length.out = nrow(Final_graph))) {
  ArrowType <-
    ifelse(Final_graph$interaction_sign[i] == 1, "\"vee\"", "\"tee\"")
```

```
  ArrowColor <-
    ifelse(Final_graph$interaction_sign[i] == 1, "\"#FFCCCB\"", "\"#ADD8E6\"")
  Dot_text <-
    c(Dot_text, paste0(Final_graph$from[i], "->", Final_graph$to[i],
      " [penwidth=1", ", color=", ArrowColor[1], ", arrowhead=", ArrowType[1],
              "];"))
}

for (i in seq_len(length.out = nrow(Table_attributes))){
  Shape <-
    ifelse(Table_attributes$Role[i]=="ligand", "invtriangle",
      ifelse(Table_attributes$Role[i]=="receptor", "triangle",
        ifelse(Table_attributes$Role[i]=="SignalingMediator", "ellipse",
          ifelse(Table_attributes$Role[i]=="transcriptionFactor", "box",
          "circle"))))
  # fillcolor <- "lavander"
    Dot_text <-
    c(Dot_text, paste0(Table_attributes$Gene[i], " [style=filled, ",
      "fillcolor=lavender, ", "color=lavender, ", "shape = ", Shape,"];"))


}
Dot_text <- c(Dot_text, "")
Dot_text <- c(Dot_text, "}")

# writeLines(Dot_text, "Results/Cytoscape/Network_Figure5.dot")
```



Export to .DOT files.

NicheNet: Assess how well top-ranked ligands can predict a gene set of interest ==========
Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 23/06/2020

Abstract

This vignette aims at reproducing the results of the following original vignette, available in the NicheNetr repository:

https://github.com/saeyslab/nichenetr/blob/master/vignettes/target_prediction_evaluation_geneset.md

In our particular case, we use sets of interactions availabe in the **Omnipath** database. We will study potential ligand-targets influence upon SARS-CoV-2 infection.

Introduction

This vignette shows how NicheNet can be used to to predict which ligands might regulate a given set of genes and how well they do this prediction. For this analysis, one needs to define:

- a set of genes of which expression in a "receiver cell" is possibly affected by extracellular protein signals (ligands) (e.g. genes differentially expressed upon cell-cell interaction)
- a set of potentially active ligands (e.g. ligands expressed by interacting "sender cells")

Therefore, you often first need to process expression data of interacting cells to define both.

In this example, we are going to use expression data after SARS-CoV-2 infection to try to dissect which ligands expressed by infected cells can have an i nfluence on the expression of target genes in the same cell lines (Autocrine view). In particular, we will focus on the inflammation response potentially induced by this ligands.

```
library(nichenetr)
library(tidyverse)
library(fgsea)
```

Read expression data of interacting cells

First, we read the results of the differentially expression analysis after infection with SARS-CoV-2 on the CALU-3 cell line.

```
expressed_genes_receiver <-
    readRDS("Results/dds_results_CALU3vsCOV2.rds") %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(!is.na(stat)) %>%
    dplyr::pull(Gene)
```

Secondly, we determine which ligands are over-expressed after SARS-CoV-2 infection.

```
padj_tres <- 0.1
log2FoldChange_tres <- 1

## We take our ligands in the network
ligands <-
    readRDS("OmniNetworks_NNformat/lr_Network_Omnipath.rds") %>%
    dplyr::pull(from) %>%
    unique()

DDS_CALU3_ligands <-
    readRDS("Results/dds_results_CALU3vsCOV2.rds") %>%
    as.data.frame() %>%
```

```
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(padj < padj_tres,
                  log2FoldChange > log2FoldChange_tres,
                  Gene %in% ligands) %>%
    dplyr::pull(Gene)
```

Load the ligand-target model we want to use

```
ligand_target_matrix <- readRDS("Results/ligand_target_matrixWithweights.rds")
ligand_target_matrix[1:5,1:5] # target genes in rows, ligands in columns
##               CALM1        WNT5A       CXCL16       CCL3L3      TNFSF10
## A1BG   0.0000000000 0.0000000000 0.000000e+00 0.000000e+00 0.0000000000
## A1CF   0.0000000000 0.0000000000 0.000000e+00 0.000000e+00 0.0000000000
## A2M    0.0011027517 0.0004845514 2.936421e-03 5.441192e-03 0.0017391820
## A2ML1  0.0000000000 0.0000000000 0.000000e+00 0.000000e+00 0.0000000000
## A4GALT 0.0002105736 0.0001070804 5.825834e-05 9.488076e-05 0.0001410451
```

Load the gene set of interest and background of genes

To establish a gene set of interest, we perform a Gene set Enrichment analysis (GSEA) and we check among the most appealing overrepresanted signatures upon SARS-CoV-2 infection. We remove the differentially expressed ligands from this comparison.

```
ranks <- readRDS("Results/dds_results_CALU3vsCOV2.rds") %>%
    as.data.frame() %>%
    tibble::rownames_to_column(var = "Gene") %>%
    dplyr::filter(!(Gene %in% DDS_CALU3_ligands)) %>%
    dplyr::filter(!is.na(stat)) %>%
    dplyr::pull(stat, name=Gene)

# immunologic_signatures <- gmtPathways("NicheNet_Omnipath/c7.all.v7.1.symbols.gmt")
hallmarlk_signatures <- gmtPathways("h.all.v7.1.symbols.gmt")
# go_signatures <- gmtPathways("NicheNet_Omnipath/c5.bp.v7.1.symbols.gmt")

fgseaRes <- fgsea(hallmarlk_signatures, ranks,  nperm=1000)
# Testing only one pathway is implemented in a more efficient manner

SignificantResults <- fgseaRes %>%
    dplyr::filter(padj < 0.01) %>%
    dplyr::arrange(desc(NES)) %>%
    dplyr::top_n(12, abs(NES))
SignificantResults
##                              pathway        pval        padj         ES
##  1: HALLMARK_INTERFERON_GAMMA_RESPONSE 0.001369863 0.005056634  0.8627654
##  2:   HALLMARK_TNFA_SIGNALING_VIA_NFKB 0.001386963 0.005056634  0.8608318
##  3: HALLMARK_INTERFERON_ALPHA_RESPONSE 0.001550388 0.005056634  0.9172465
##  4:     HALLMARK_INFLAMMATORY_RESPONSE 0.001440922 0.005056634  0.7274370
##  5:   HALLMARK_IL6_JAK_STAT3_SIGNALING 0.001579779 0.005056634  0.7126008
##  6:                   HALLMARK_HYPOXIA 0.001375516 0.005056634  0.5893036
##  7:                HALLMARK_APOPTOSIS 0.001438849 0.005056634  0.5743182
##  8:            HALLMARK_G2M_CHECKPOINT 0.003831418 0.006561680 -0.5370578
##  9:            HALLMARK_MYC_TARGETS_V2 0.002638522 0.006561680 -0.6827875
## 10:            HALLMARK_MYC_TARGETS_V1 0.003921569 0.006561680 -0.6785459
## 11:               HALLMARK_E2F_TARGETS 0.003937008 0.006561680 -0.6829123
```

```
## 12: HALLMARK_OXIDATIVE_PHOSPHORYLATION 0.003787879 0.006561680 -0.6946604
##           NES nMoreExtreme size                                leadingEdge
##  1:  3.143190            0  174     OAS2,IFIT1,RSAD2,IFIT2,IFIT3,TNFAIP3,...
##  2:  3.114605            0  161 IFIT2,TNFAIP3,ATF3,PPP1R15A,NFKBIA,IFIH1,...
##  3:  3.029033            0   89        RSAD2,IFIT2,IFIT3,MX1,IFIH1,TXNIP,...
##  4:  2.590435            0  148       NFKBIA,IRF1,LAMP3,IFITM1,KLF6,RTP4,...
##  5:  2.269832            0   67         IRF1,STAT2,MAP3K8,STAT1,JUN,PIM1,...
##  6:  2.144207            0  173  TNFAIP3,ATF3,PPP1R15A,TIPARP,DUSP1,STC2,...
##  7:  2.039701            0  140       ATF3,TXNIP,IRF1,TAP1,PMAIP1,ISG20,...
##  8: -2.237533            0  190        KPNA2,MCM5,SQLE,HSPA8,MCM6,LMNB1,...
##  9: -2.354863            0   58        TMEM97,MCM5,PHB,DCTPP1,PLK1,MCM4,...
## 10: -2.827153            0  193          KPNA2,MCM5,PHB,MCM6,SRSF2,NME1,...
## 11: -2.857211            0  195       KPNA2,MCM5,MXD3,SPAG5,NCAPD2,POLD1,...
## 12: -2.878447            0  184      MAOB,POLR2F,COX8A,LDHB,VDAC3,NDUFB2,...
```

I select inflammation related genes.

```
## I am going to check with inflammation genes
inflammationGenes <- SignificantResults %>%
    dplyr::filter(pathway == "HALLMARK_INFLAMMATORY_RESPONSE") %>%
    dplyr::pull(leadingEdge) %>% unlist()

## We chech that there are no upregulated ligands here.
intersect(DDS_CALU3_ligands,inflammationGenes )
## character(0)

geneset_oi <- inflammationGenes[inflammationGenes %in% rownames(ligand_target_matrix)]

head(geneset_oi)
## [1] "NFKBIA" "IRF1"   "IFITM1" "KLF6"   "RTP4"   "IRAK2"
background_expressed_genes <- expressed_genes_receiver %>%
    .[. %in% rownames(ligand_target_matrix)]
head(background_expressed_genes)
## [1] "SAMD11"   "NOC2L"    "ISG15"    "AGRN"     "TNFRSF18" "SDF4"
```

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

As potentially active ligands, we will use ligands that are 1) Over-expressed in CALU3 after SARS-CoV-2 infection and 2) can bind a (putative) receptor expressed by malignant cells. Putative ligand-receptor links were gathered from Omnipath ligand-receptor data sources.

```
expressed_ligands <- intersect(ligands,DDS_CALU3_ligands)

receptors <- unique(lr_network$to)
expressed_receptors <- intersect(receptors,expressed_genes_receiver)

potential_ligands <- lr_network %>%
    filter(from %in% expressed_ligands & to %in% expressed_receptors) %>%
    pull(from) %>%
    unique()
head(potential_ligands)
## [1] "CXCL1" "CXCL2" "CXCL3" "CXCL5" "CCL20" "CCL17"
```

we now calculate the ligand activity of each ligand, or in other words, we will assess how well each over-expressed ligand after viral infection can predict the inflammation gene set compared to the background of

expressed genes (predict whether a gene belongs to the inflammation program or not).

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

Now, we want to rank the ligands based on their ligand activity. In our validation study, we showed that the pearson correlation between a ligand's target predictions and the observed transcriptional response was the most informative measure to define ligand activity. Therefore, we will rank the ligands based on their pearson correlation coefficient.

```
ligand_activities %>%
    arrange(-pearson)
## # A tibble: 89 x 4
##    test_ligand auroc   aupr pearson
##    <chr>       <dbl>  <dbl>   <dbl>
##  1 IL23A       0.742 0.0693   0.173
##  2 TNF         0.753 0.0604   0.165
##  3 TNFSF13B    0.732 0.0568   0.159
##  4 IL1A        0.712 0.0532   0.155
##  5 LAMA2       0.740 0.0597   0.152
##  6 ICAM4       0.731 0.0645   0.151
##  7 L1CAM       0.735 0.0645   0.151
##  8 CXCL9       0.742 0.0771   0.151
##  9 NPPB        0.724 0.0721   0.151
## 10 INHBA       0.677 0.0591   0.150
## # ... with 79 more rows
best_upstream_ligands <- ligand_activities %>%
    top_n(12, pearson) %>%
    arrange(-pearson) %>%
    pull(test_ligand)
head(best_upstream_ligands)
## [1] "IL23A"    "TNF"      "TNFSF13B" "IL1A"     "LAMA2"    "ICAM4"
```

For the top 12 ligands, we will now build a multi-ligand model that uses all top-ranked ligands to predict whether a gene belongs to the inflammatory response program of not. This classification model will be trained via cross-validation and returns a probability for every gene.

```
## To increase these numbers.
k = 3 # 3-fold
n = 10 # 10 rounds
inflammation_gene_predictions_top12_list <- seq(n) %>%
    lapply(assess_rf_class_probabilities,
           folds = k,
           geneset = geneset_oi,
           background_expressed_genes = background_expressed_genes,
           ligands_oi = best_upstream_ligands,
           ligand_target_matrix = ligand_target_matrix)
```

Evaluate now how well the target gene probabilies agree with the gene set assignments

```
# get performance: auroc-aupr-pearson
target_prediction_performances_cv <-
    inflammation_gene_predictions_top12_list %>%
    lapply(classification_evaluation_continuous_pred_wrapper) %>%
```

```
    bind_rows() %>%
    mutate(round=seq(1:nrow(.)))
```

We display here the AUROC, AUPR and PCC of this model (averaged over cross-validation rounds)

```
target_prediction_performances_cv$auroc %>% mean()
## [1] 0.70906
target_prediction_performances_cv$aupr %>% mean()
## [1] 0.0306664
target_prediction_performances_cv$pearson %>% mean()
## [1] 0.09457835
```

Evaluate now whether genes belonging to the gene set are more likely to be top-predicted. We look at the top 5% of predicted targets here.

```
## get performance: how many inflammatory genes and inflammatory genes among
## top 5% predicted targets
target_prediction_performances_discrete_cv <-
    inflammation_gene_predictions_top12_list %>%
    lapply(calculate_fraction_top_predicted, quantile_cutoff = 0.95) %>%
    bind_rows() %>%
    ungroup() %>%
    mutate(round=rep(1:length(inflammation_gene_predictions_top12_list), each = 2))
```

What is the fraction of inflammation related genes that belongs to the top 5% predicted targets?

```
target_prediction_performances_discrete_cv %>%
    filter(true_target) %>%
    .$fraction_positive_predicted %>%
    mean()
## [1] 0.3016129
```

What is the fraction of non-inflammation related genes that belongs to the top 5% predicted targets?

```
target_prediction_performances_discrete_cv %>%
    filter(!true_target) %>%
    .$fraction_positive_predicted %>%
    mean()
## [1] 0.0496038
```

We see that the inflammation genes are enriched in the top-predicted target genes. To test this, we will now apply a Fisher's exact test for every cross-validation round and report the average p-value.

```
target_prediction_performances_discrete_fisher <-
    inflammation_gene_predictions_top12_list %>%
    lapply(calculate_fraction_top_predicted_fisher, quantile_cutoff = 0.95)

target_prediction_performances_discrete_fisher %>% unlist() %>% mean()
## [1] 3.250723e-08
```

Finally, we will look at which genes are well-predicted in every cross-validation round.

```
# get top predicted genes
top_predicted_genes <- seq(length(inflammation_gene_predictions_top12_list)) %>%
    lapply(get_top_predicted_genes,inflammation_gene_predictions_top12_list) %>%
    purrr::reduce(full_join, by = c("gene","true_target"))
top_predicted_genes %>% filter(true_target)
## # A tibble: 27 x 12
```

```
##    gene  true_target predicted_top_t... predicted_top_t... predicted_top_t...
##    <chr> <lgl>       <lgl>             <lgl>             <lgl>
##  1 NFKB... TRUE        TRUE              TRUE              TRUE
##  2 SELE  TRUE        TRUE              TRUE              TRUE
##  3 IRF1  TRUE        TRUE              TRUE              TRUE
##  4 NFKB1 TRUE        TRUE              TRUE              TRUE
##  5 EIF2... TRUE        TRUE              TRUE              TRUE
##  6 MYC   TRUE        TRUE              TRUE              TRUE
##  7 RIPK2 TRUE        TRUE              TRUE              NA
##  8 LYN   TRUE        TRUE              NA                TRUE
##  9 CDKN... TRUE        TRUE              TRUE              TRUE
## 10 PTAFR TRUE        TRUE              TRUE              TRUE
## # ... with 17 more rows, and 7 more variables: predicted_top_target_round4 <lgl>,
## #   predicted_top_target_round5 <lgl>, predicted_top_target_round6 <lgl>,
## #   predicted_top_target_round7 <lgl>, predicted_top_target_round8 <lgl>,
## #   predicted_top_target_round9 <lgl>, predicted_top_target_round10 <lgl>
```

References

Browaeys, R., Saelens, W. & Saeys, Y. NicheNet: modeling intercellular communication by linking ligands to target genes. Nat Methods (2019) doi:10.1038/s41592-019-0667-5

Puram, Sidharth V., Itay Tirosh, Anuraag S. Parikh, Anoop P. Patel, Keren Yizhak, Shawn Gillespie, Christopher Rodman, et al. 2017. "Single-Cell Transcriptomic Analysis of Primary and Metastatic Tumor Ecosystems in Head and Neck Cancer." Cell 171 (7): 1611–1624.e24. https://doi.org/10.1016/j.cell.2017.10.044.