LErNet Introduction

Vincenzo Bonnici, Simone Caligola, Giulia Fiorini, Luca Giudice, Andrea Furlani, Rosalba Giugno

April 4, 2020

LErNet Introduction

Long non-coding RNAs (lncRNA) have recently acquired a boost of interest for their implication in several biological conditions. However, many of these elements are not yet annotate. LErNet is focused on a new network analysis method for annotating lncRNAs and guide the enrichment analysis. The core is a network expansion algorithm that aims to enrich the context of lncRNAs. The context is constructed by integrating the genes encoding proteins that are found next to the non-coding elements at both the genome and the system level. The pipeline is particularly useful in situations where the functions of discovered lncRNAs are not yet known.

Citation

If you have used the package LErNet in your project, please cite the following paper:

Bonnici V., Caligola S., Fiorini G., Giudice L., Giugno R.
LErNet: characterization of lncRNAs via context-aware network expansion and enrichment analysis.
In 2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) (pp. 1-8). IEEE.

License

LErNet is distributed under the MIT license. This means that it is free for both academic and commercial use. Note however that some third party components in LErNet require that you reference certain works in scientific publications. You are free to link or use LErNet inside source code of your own program. If do so, please reference (cite) LErNet and this website.

We appreciate bug fixes and would be happy to collaborate for improvements.

Installation

If you have not installed the "devtools" package, you will need to install and load it in order to install LErNet:

```r
if (!require(devtools)) {
    install.packages("devtools", repos = "http://cran.us.r-project.org")
    require(devtools)
}
library(devtools)
```

Then, to install LErNet:

```r
if (!require(LErNet)) {
    install_github("InfOmics/LErNet")
    require(LErNet)
}
library(LErNet)
```

Running Example

We report a step-by-step example to execute LErNet on published data (Zhao et al., Scientific reports, 2018). The dataset is composed by a list of differentially expressed genes and long non-coding RNA (lncRNA). Original excel files are provided with the LErNet package in order to correctly execute the analysis. Further, a GTF file (from GENCODE database) is provided to retrieve genomic context of genes and lncRNAs.

It's necessary to install and load the following libraries to run the example:

```r
library(R.utils)
library(xlsx)
library(biomaRt)
```

The first step of the analysis is to retrieve a set of genes and lncRNAs of interest and the information of the genomic context. In the folowing lines of code DE genes and DE lncRNAs are obtained directly from the excel files provided within the LErNet package and loaded after several preprocess operations.

```r
lncrna_file <- system.file("extdata",
                           "41598_2018_30359_MOESM2_ESM.xlsx", package = "LErNet")
pcrna_file <- system.file("extdata",
                          "41598_2018_30359_MOESM3_ESM.xlsx", package = "LErNet")
gtf_file <- system.file("extdata",
                        "gencode.vM20.chr_patch_hapl_scaff.annotation.gtf.gz",
                        package = "LErNet")

pcgenes<-read.xlsx(pcrna_file,sheetIndex = 1)
pcgenes<-as.character(pcgenes$gene_id)

lncrnaInfo<-read.xlsx(lncrna_file, sheetIndex = 1)
lncrnaInfo <- data.frame(lapply(lncrnaInfo, as.character), stringsAsFactors=FALSE)
last<-which(lncrnaInfo$significant == 'FALSE')[1]
lncrnaInfo<-lncrnaInfo[1:last-1,]
lncrnaAll<-as.character(lncrnaInfo$gene_id)
```

LErNet provides the function load_gtf to load into a dataframe the necessary information from a GTF file.

```r
complete_positions <- LErNet::load_gtf(gtf_file)
```

It is necessary that the dataframe contains the information for all genes and lncRNAs in input. In this example data come with information about novel lncRNAs. These information must be added to the dataframe complete_positions:

```r
# Extract the novel lncRNAs from the dataframe "lncrnaInfo"

novel<-lncrnaInfo
novel<-novel[novel$isoform_status == "lncRNA_Novel", ]

# Some elaboration to extract the necessary information about lncRNAs

chrs <- paste0("chr",sapply(strsplit
                             (sapply(strsplit( novel$locus, "-"), `[`, 1), ":"), `[`, 1))
starts <- sapply(strsplit(sapply(strsplit(novel$locus, "-"), `[`, 1), ":"), `[`, 2)
ends <- sapply(strsplit(novel$locus, "-"), `[`, 2)
novel_gtf <- data.frame( "id" = novel$gene_id, "type" = rep('novel lncRNA',
                          times = nrow(novel)), "seqname" = chrs,
                         "start" = starts, "end" = ends )

# Add information of novel lncRNAs into the dataframe containing
# information about known genes/lncRNAs

complete_positions <- rbind(complete_positions, novel_gtf)
rownames(complete_positions) <- seq(1:nrow(complete_positions))
```

LErNet exploits PPI network to expand a set of protein coding genes associated with lncRNAs. In this example the database STRING is exploited to build the PPI network, however LErNet can take as input a dataframe with 2 columns containing the edges of the network. Each element of the netwoek must be identified with its ENSEMBL id. To build the network with STRING is necessary to specfy a threshold of significance for protein interactions, the taxonomy id of the organism of interest:

```r
mart <- useMart(biomart = "ensembl", dataset = "mmusculus_gene_ensembl")
# WARNING: on R <= 3.4 this may cause mutiple errors.
# Please, run it until no errors are arised.
# or, artenatively, use
# mart <- useEnsembl(biomart = "ensembl", dataset = "mmusculus_gene_ensembl",
#                    mirror = "useast")

stringdb_tax = 10090
stringdb_thr = 900

# alteratively, for human
# mart <- useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
# WARNING: on R <= 3.4 this may cause mutiple errors.
# Please, run it until no errors are arised.
# stringdb_tax = 9606
```

After, the function get_stringdb must be executed to map ENSEMBL protein ids into ENSEMBL gene ids. To perform the mapping phase a connection to BioMart is needed.

```r
ret <- LErNet::get_stringdb( stringdb_tax = stringdb_tax,
                             stringdb_thr = stringdb_thr, mart = mart)
```

The function get_stringdb returns a list with 2 dataframe, one named ppi_network containing the PPI network and the second named ensp_to_ensg containing the mapping table of ENSEMBL ids. The first dataframe can be provided to LErNEt without the use of STRING.

The second step is to generate the genomic context, i.e. to find the genomic seeds necessary to run the expansion phase through th PPI network. The function to perform accomplish this task is get_genomic_context. The function takes in input the information retrieved from the GTF file (complete_positions), the list of protein coding genes and lncRNAs and a window in which to search for genomic neighbors. The function returns a dataframe containing for each lncRNA one or more partner coding genes.

```r
library(GenomicRanges)
genomic_context <- LErNet::get_genomic_context(positions = complete_positions,
                                               lncgenes = lncrnaAll, pcgenes = pcgenes,
                                               max_window = 100000, strict_genomics = TRUE)
```

It can be useful to show some basic statistics on the generated seeds:

```r
# Number of genomic seeds
length(unique(genomic_context$partner_coding_gene))

# Mean number of genomic seeds for each lncRNA
mean(table(genomic_context$partner_coding_gene))
```

The following lines of codes are necessary to match the seeds proteins with the input coding genes to obtain a set of strict starting proteins.

```r
annot<-getBM(attributes = c("ensembl_gene_id", "ensembl_transcript_id",
                            "ensembl_peptide_id"),
             filters = "ensembl_gene_id", values = unique(pcgenes), mart = mart)
strict_proteins<-annot$ensembl_peptide_id
strict_proteins<-strict_proteins[ strict_proteins != ""  ]
```

The next step is the core phase of LErNEet, i.e. the expansion phase with the function expand_seeds. Expansion takes innput the genomic context, the PPI network, the id mapping table, the list of starting proteins. The parameter strict_connectors (TRUE as default) specify that the connector proteins must be in the list of strict starting proteins.

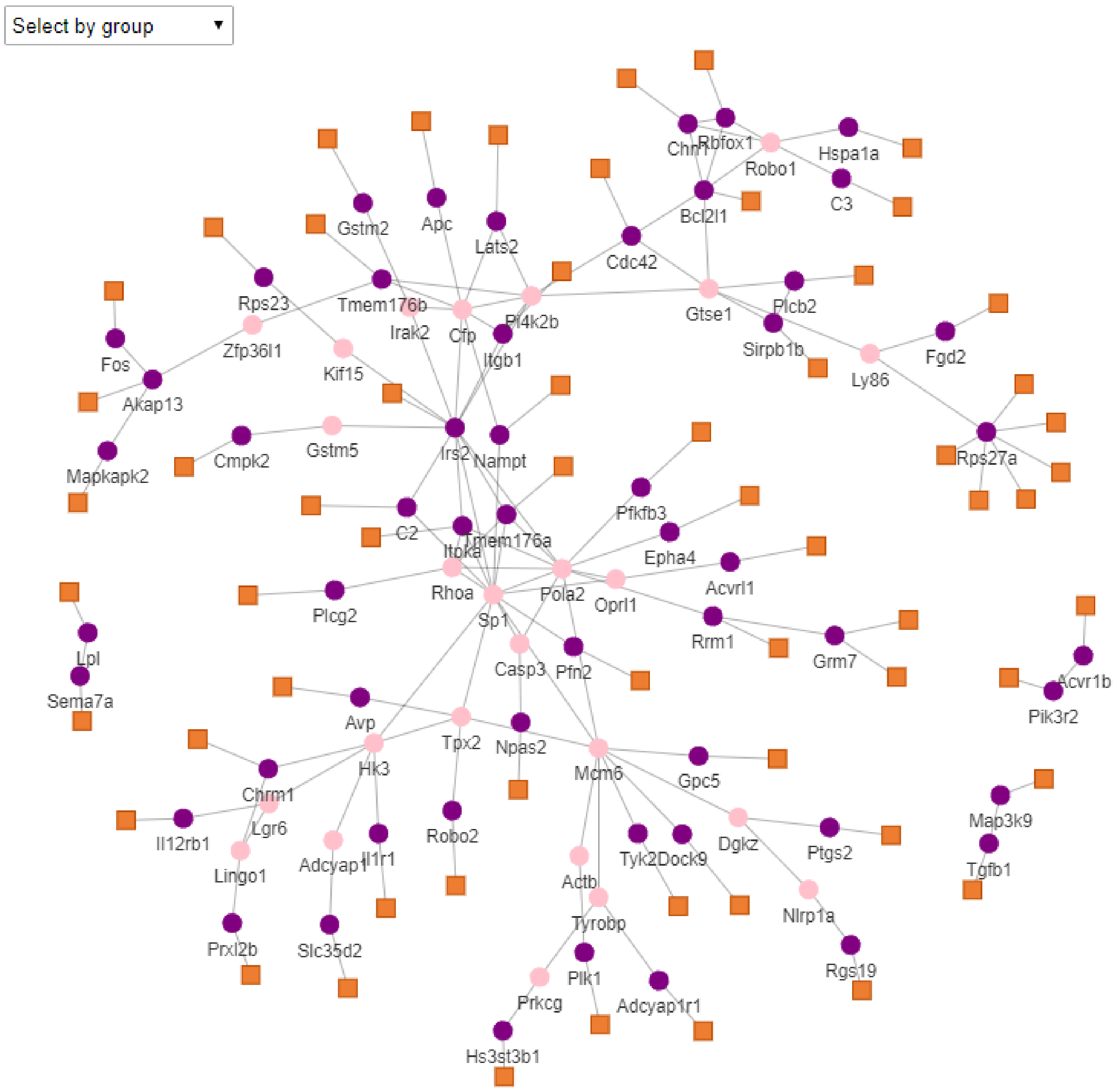
```r
ret <- LErNet::expand_seeds(
  genomic_context = genomic_context,
  ppi_network = ppi_network,
  ensp_to_ensg = ensp_to_ensg,
  strict_proteins = strict_proteins,
  strict_connectors = TRUE)
```

The function expand_seeds returns a list containing a dataframe with the network components, a vector with the proteins in input and a vector with the network seeds:

```r
network_components <- ret[["network_components"]]
input_proteins <- ret[["input_proteins"]]
network_seeds <- ret[["network_seeds"]]
```

LErNet allows to visualize the results of the analysis through the use of the package visNetwork. The function visualize takes in input the list of lncRNAs, the genomic context, the mapping table of ENSEMBL ids, the list of strict starting proteins, the network seeds, the PPI network, one or more network components extracted by LErNet, the connection to Biomart database and the Biomart identifier to show gene SYMBOLs.

```r
LErNet::visualize(
  lncgenes = lncrnaAll,
  genomic_context = genomic_context,
  ensp_to_ensg = ensp_to_ensg,
  input_proteins = input_proteins,
  network_seeds = network_seeds,
  ppi_network = ppi_network,
  expanded_elements = unlist(network_components) ,
  mart = mart,
  mart_symbol_column = "mgi_symbol"  # "hgnc_symbol" for human
)
```



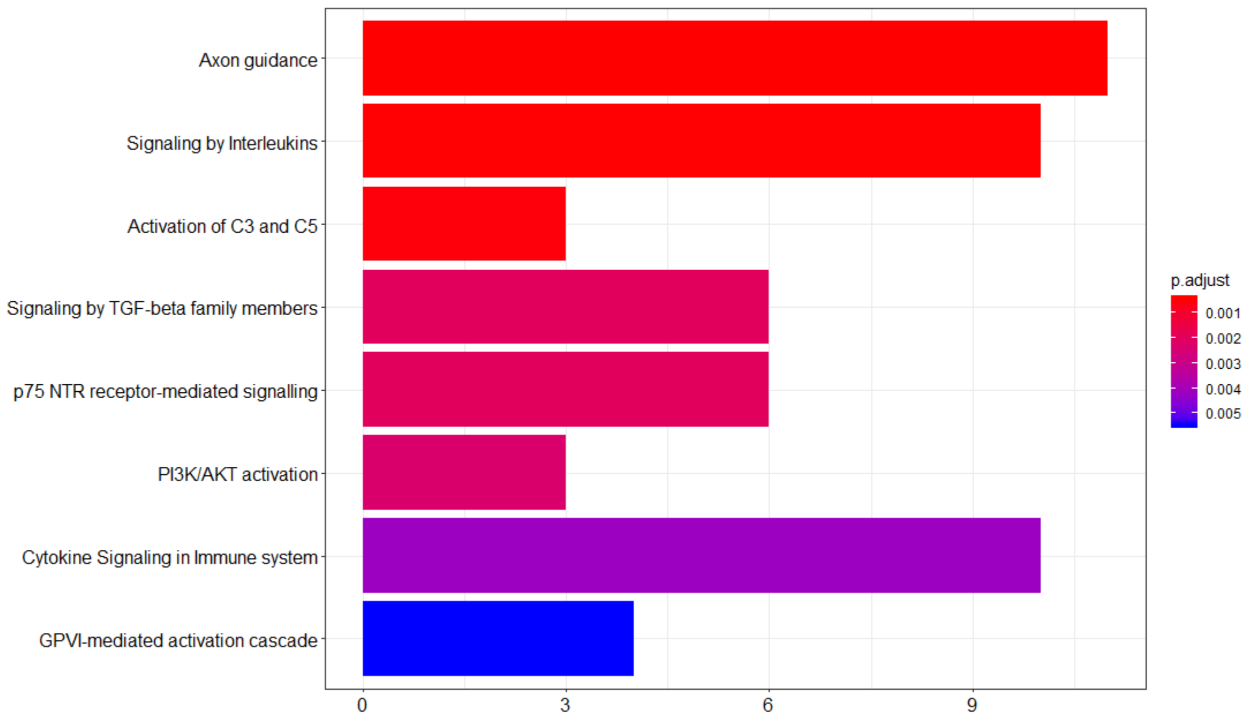
This is the image returned by the function. In the left upper box it is possible to select only a group to be viewed in the plot (lncRNA, Seed Connector and Seed Protein).

The last step is the functional enrichment of the results. Basically LErNet exploits the package ReactomePA to retrieve significant pathways through the function enrich:

```
if (!require("org.Mm.eg.db")) {
    BiocManager::install("org.Mm.eg.db")
    require("org.Mm.eg.db")
}

enrichment <- LErNet::enrich(  ens_proteins = unlist(network_components),
                               organism = "mouse",   mart = mart)
# LErNet::enrich(  ens_proteins = unlist(network_components),
#                  organism = "mouse",   mart = mart, max_to_show =2)
# for human: organism = "human"
```

```r
barplot(enrichment)
```



These are the most significant pathways retrieved by LErNet for the example with mouse genes. However, the user can use the preferred tool to make functional enrichment.

Documentation

Function Index

- `enrich`
- `enps_to_entrez`
- `expand_seeds`
- `get_genomic_context`
- `get_stringdb`
- `load_gtf`
- `visualize`

– enrich

```
enrich(ens_proteins, organism, mart, max_to_show = NULL)
```

Description

Computes functional enrichment of a given set of proteins via the ReactomePA package. A bar plot reporting the enriched pathway and their p-values is shown.

Inputs

- *ens_proteins*: list of proteins, in Ensembl format, to compute the enrichment
- *organism*: organism name(see `enrichPathway`)
- *mart*: a biomaRt object of the given species neeed for the conversion from Ensembl to Entrez IDs
- *max_to_show*:

Output

A ReactomePA result object.

– enps_to_entrez

```
enps_to_entrez(ens_proteins, mart)
```

Description

Maps a list of protein IDs in the Ensmbl format to the Entrez naming system.

Inputs

- *ens_proteins*: list of Ensembl IDs
- *mart*: a biomaRt object of the given species neeed for the conversion from Ensembl to Entrez IDs

Output

A data.frame representing the mapping

– expand__seeds

```r
expand_seeds(genomic_context, ppi_network, ensp_to_ensg,
             strict_proteins, strict_connectors = TRUE)
```

Description

Expands with connectors the network formed by seed proteins, that are the producs fo the genes int he genomic context, by the expasion algorithm. Connectors are neighbors of selected proteins in the input PPI network.

Inputs

- *genomic__context*: a two column data.fram produced by `get_genomic_context`
- *ppi__network*: a two column data.frame representing PPI network edges (see also `get_stringdb`)
- *ensp__to__ensg*: a two column data.frame for mapping proteins to their producer genes (see also `get_stringdb`)
- *strict__proteins*: a list of proteins
- *strict__connectors*: if `TRUE` connectors can onyl be choosen from the `strict_proteins` list

Output

A list

- network__components

  a list of connected components of the resultant expanded network. Each compoent is a list of proteins.

- network__seeds

  list of seed proteins that have succefully been mapped to the PPI network.

– get__genomic__context

```r
get_genomic_context(positions, lncgenes, pcgenes, max_window = 1e+05,
                    strict_genomics = TRUE)
```

Description

Retrieves the genomic context of input lncRNAs. The genomic context is defined as the set of protin coding genes that resides within a given range.

Inputs

- *positions*: a data.frame reporting gemoic positions. Columns are `id type seqname start end`. It may contain features nto lited in `lncgenes` and `pcgenes`
- *lncgenes*: a list of lncRNA genes
- *pcgenes*: a list of protein-coding genes that are of interest for the study
- *max__window*: the maximum size of the genomic range
- *strict__genomics*: if `FALSE`, it allows the genomic context to be formed by p.c. genes in the `pcgenes` list

Output

A two column data.frame reporting neighborhood information. The first column gives lncRNAs and the second column gives their associated neighbors.

– get_stringdb

```
get_stringdb(stringdb_tax = 9606, stringdb_thr = 900, mart)
```

Description

Retrieves the PPI network formt he STRING database via the STRINGdb. STRINGdb often onyl associates a primary product to a gene, thus other products are not reported. The function also returns the proteins associated to each gene within the STRING database.

Inputs

- *stringdb_tax*: taxa of the species
- *stringdb_thr*: threshold to be applied to the score on the edges of the PPI
- *mart*: a biomaRt object for mapping proteins to producer genes (Ensembl IDs)

Output

A list

- ppi_network

  a two columns data.frame representing the PPI network by listing its edges.

- ensp_to_ensg

  a two columns data.frame reporting for each protein the corresponding gene (Ensembl IDs)

– load_gtf

```
load_gtf(gtf_file)
```

Description

Creates the choordinates data.frame by reading the data from a GTF file having 9 columns (whihc is the typical format of GTF files from GENCODE).

Inputs

- *gtf_file*: the path to the GTF file

Output

A data.frame with columns: `id type seqname start end`

– visualize

```
visualize(lncgenes, genomic_context, ensp_to_ensg, input_proteins, network_seeds,
          ppi_network, expanded_elements, mart, mart_symbol_column = NULL)
```

Description

Visualizes the expanded network, composed by seed proteins and connectors. LncRNA are added together with extra edges in order to report the genomic context. LncRNAs are linked to the proteins that are products of their genomic context.

Inputs

- *lncgenes*: a list of lncRNA genes
- *genomic_context*: the genomc context of the lncRNAs (see `get_genomic_context`)
- *ensp_to_ensg*: a two column data.frame for mapping proteins to their producer genes (see also `get_stringdb`)
- *input_proteins*: the complete list of input proteins,that are of interest for the study
- *network_seeds*: the lsit of seed protieins
- *ppi_network*: a two column data.frame representing PPI network edges (see also `get_stringdb`)
- *expanded_elements*: the list of proteins that must be visualized
- *mart*: a biomaRt ojbect used to visualize symbols instead of Ensembl IDs
- *mart_symbol_column*: column of the biomaRt ojbect fomr wich symbols are retrieved

Output

Visualizes the network in the Viewer window.