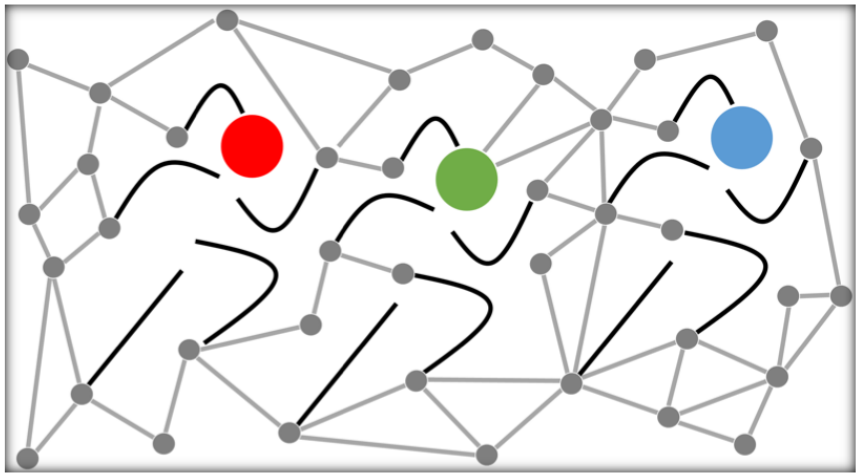


SAveRUNNER

Searching off-l**A**bel d**RU**g a**N**d **NE**two**R**k

BMC Bioinformatics 2021, **22**:150

PLoS Computational Biology 2021; **17**(2):e1008686

https://github.com/sportingCode/SAveRUNNER.git

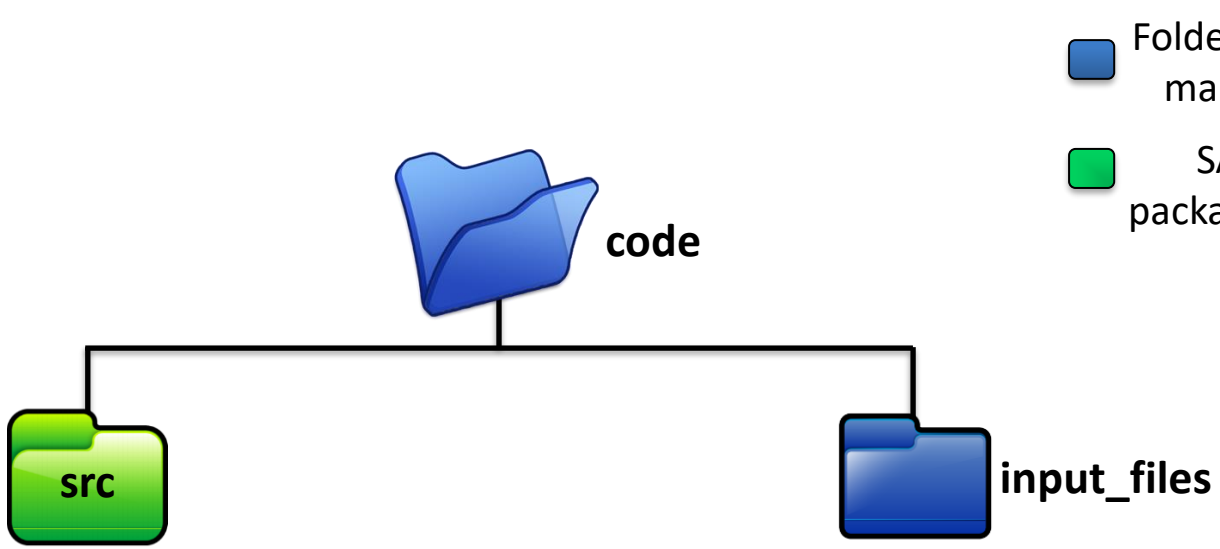
GETTING STARTED

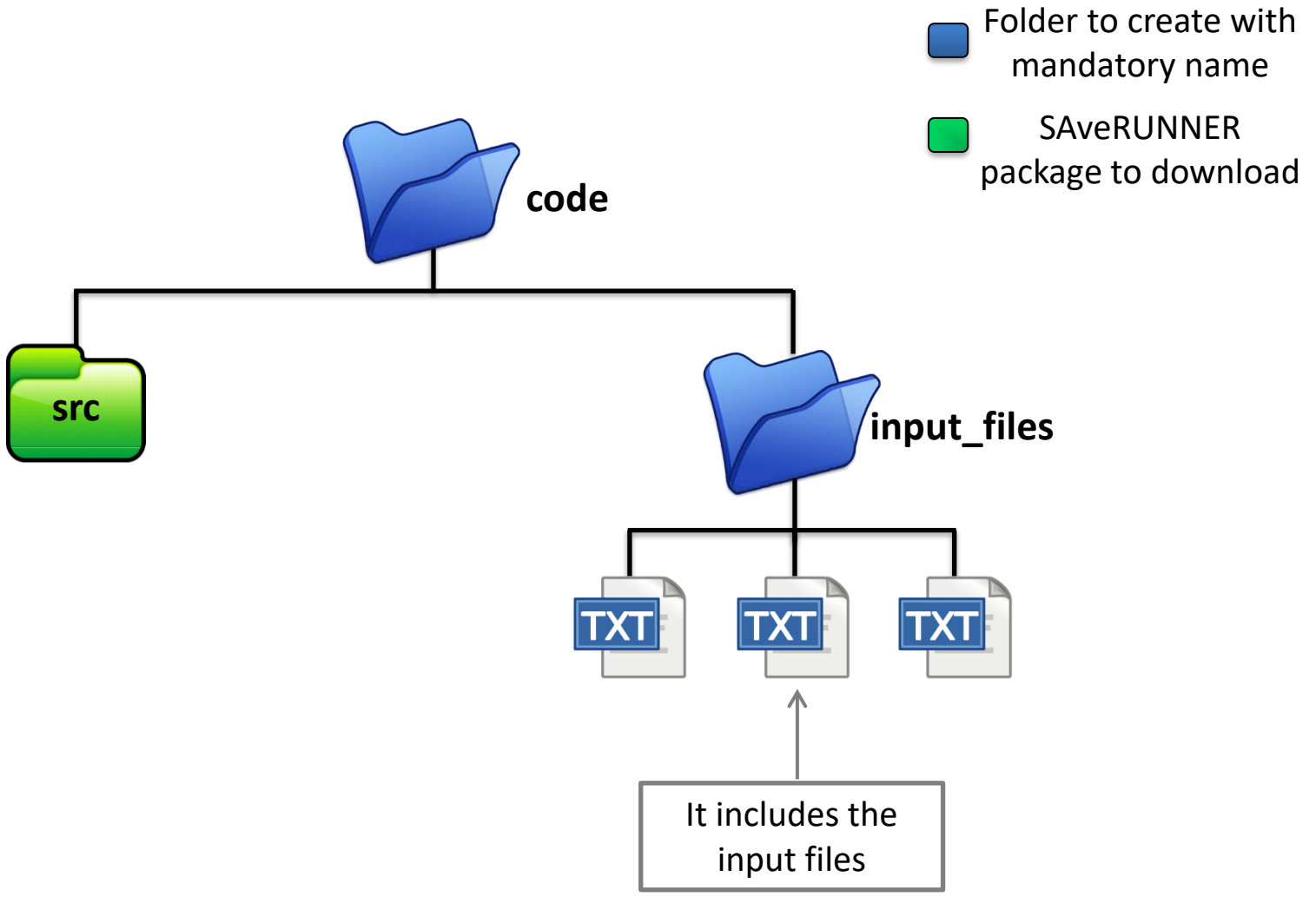
Software requirement

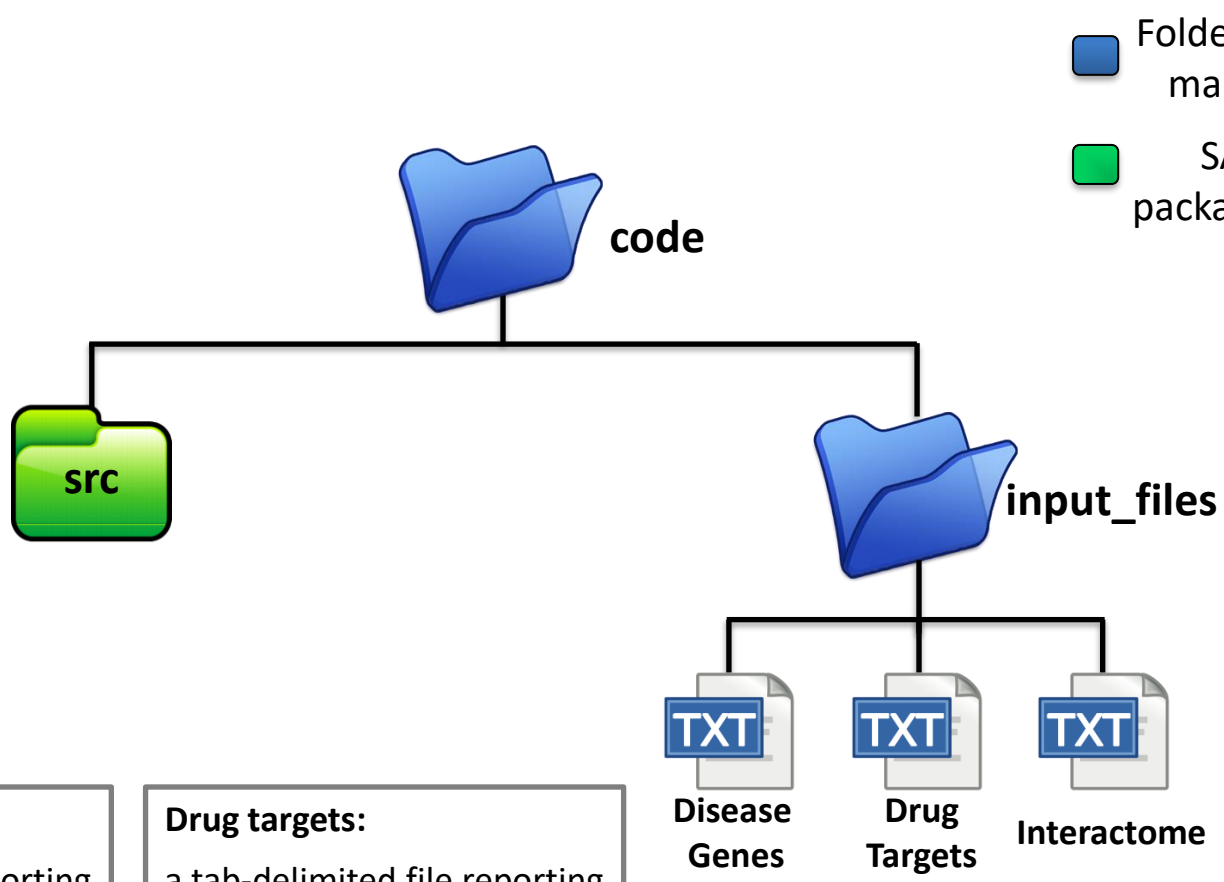
- SAveRUNNER has been developed in R (version 3.6.1) and tested on the following operative systems:
  - macOS High Sierra 10.13.6
  - Windows 10 Pro

Setting up

- Install R and R studio
- Download and unzip the SAveRUNNER software package (“*src*” folder)


Folder to create with mandatory name
SAveRUNNER package to download
code
src
input_files


Folder to create with mandatory name
SAveRUNNER package to download
code
src
input_files
TXT
TXT
TXT
It includes the input files



Disease genes:

a tab-delimited file reporting the disease-gene association as an edges list. It must include two columns: one named "**disease**" with the disease names and the other one named "**GeneID**" with the associated disease genes in Entrez Gene ID.

Drug targets:

a tab-delimited file reporting the drug-targets interactions as an edges list. It must include two columns: one named "**Drug**" with the drug names and the other one named "**GeneID**" with the interacting target genes in Entrez Gene ID.

Interactome:

a tab-delimited file reporting the human interactome. It must be provided as an edges list, i.e., it must include two columns reporting all the pairs of interacting genes in Entrez Gene ID.

Disease Genes - example

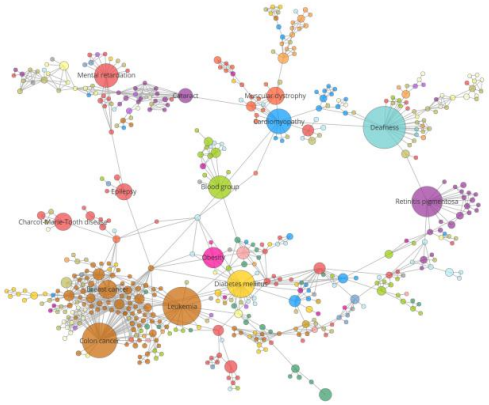
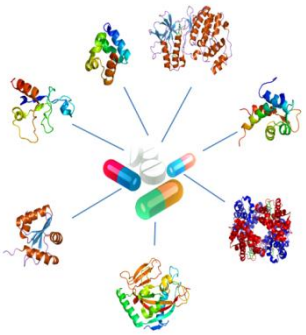
	A	B
1	disease	GeneID
2	22q11 Deletion Syndrome	1283
3	22q11 Deletion Syndrome	1312
4	22q11 Deletion Syndrome	221037
5	22q11 Deletion Syndrome	2246
6	22q11 Deletion Syndrome	2255
7	22q11 Deletion Syndrome	2260
8	22q11 Deletion Syndrome	29801
9	22q11 Deletion Syndrome	5288
10	22q11 Deletion Syndrome	5297
11	22q11 Deletion Syndrome	54487
12	22q11 Deletion Syndrome	54584
13	22q11 Deletion Syndrome	54716
14	22q11 Deletion Syndrome	5625
15	22q11 Deletion Syndrome	5660

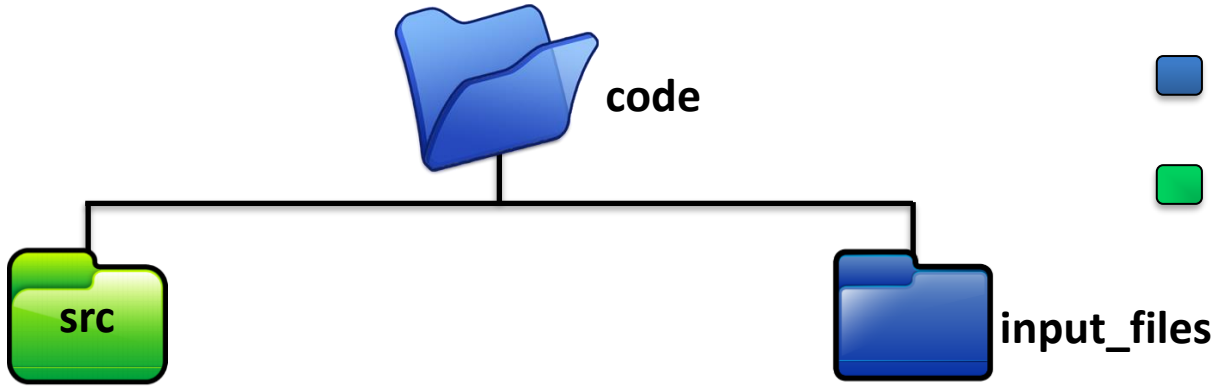
Drug Targets - example

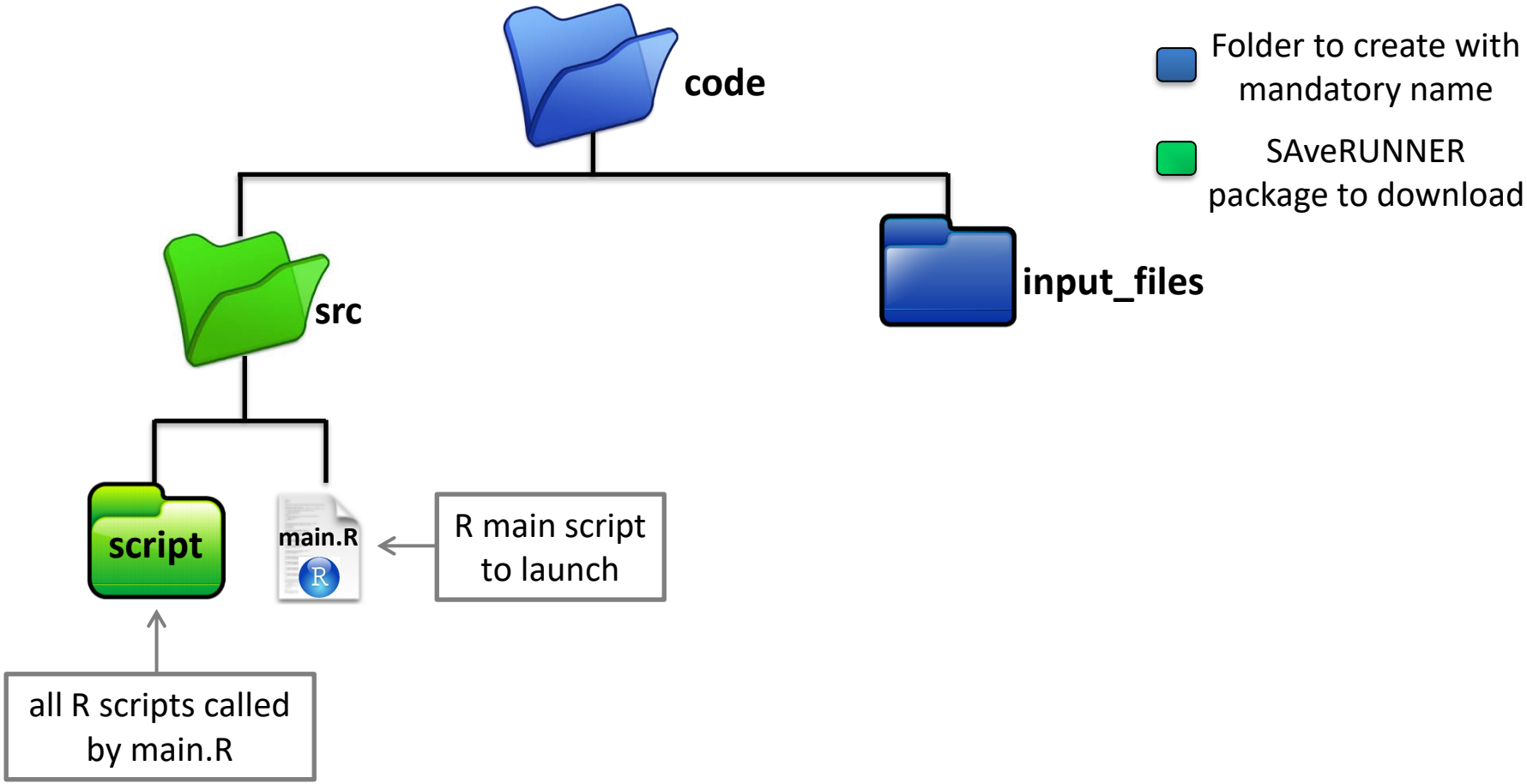
	A	B
1	Drug	GeneID
2	Zuclopenthixol	150
3	Zuclopenthixol	1813
4	Zuclopenthixol	1812
5	Zuclopenthixol	1816
6	Zuclopenthixol	3356
7	Zuclopenthixol	148
8	Zuclopenthixol	3269
9	Zucapsaicin	7442
10	Zotepine	1813
11	Zotepine	1812
12	Zotepine	1816
13	Zotepine	6530
14	Zotepine	3356
15	Zotepine	6532

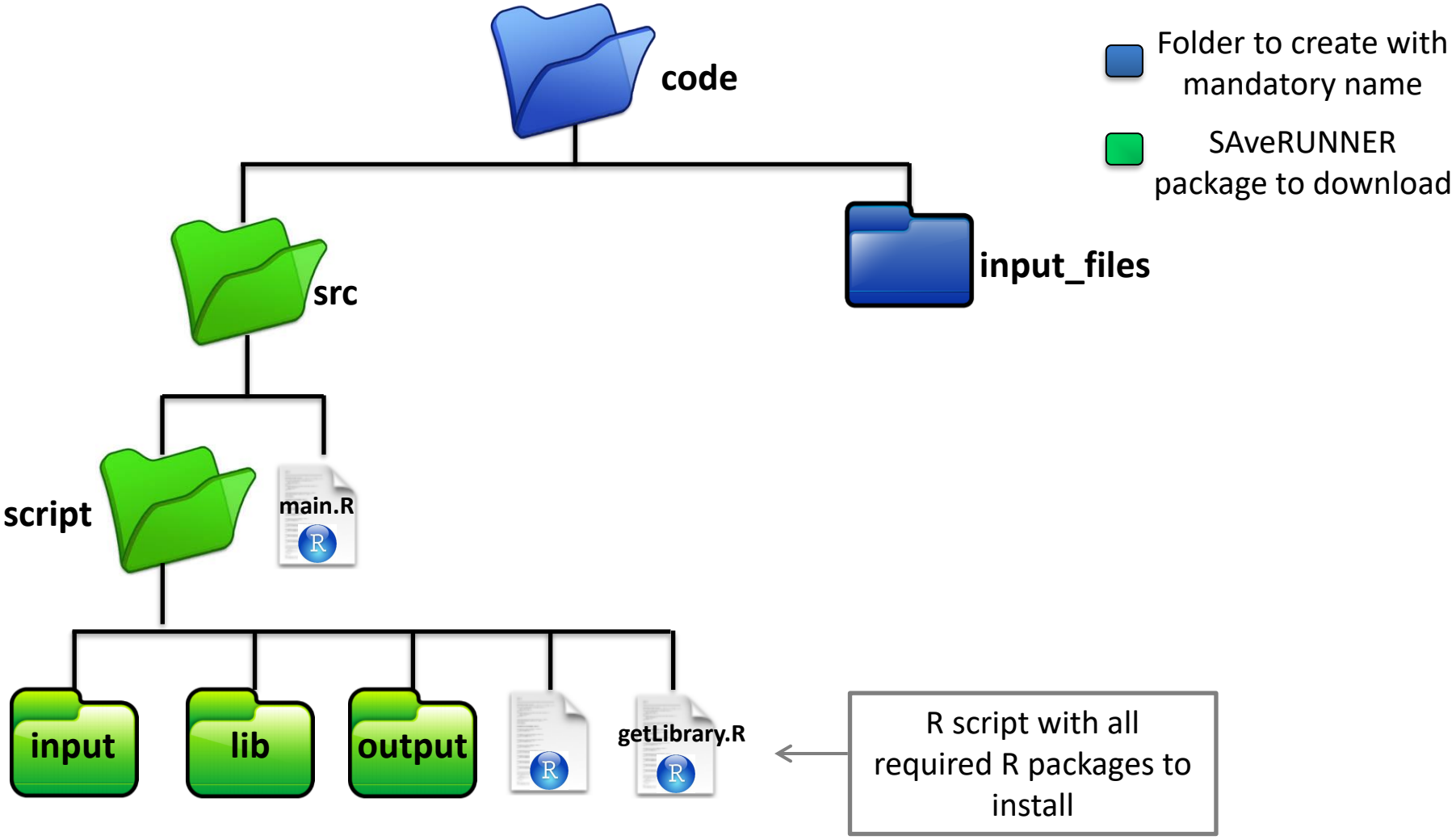
Interactome - example

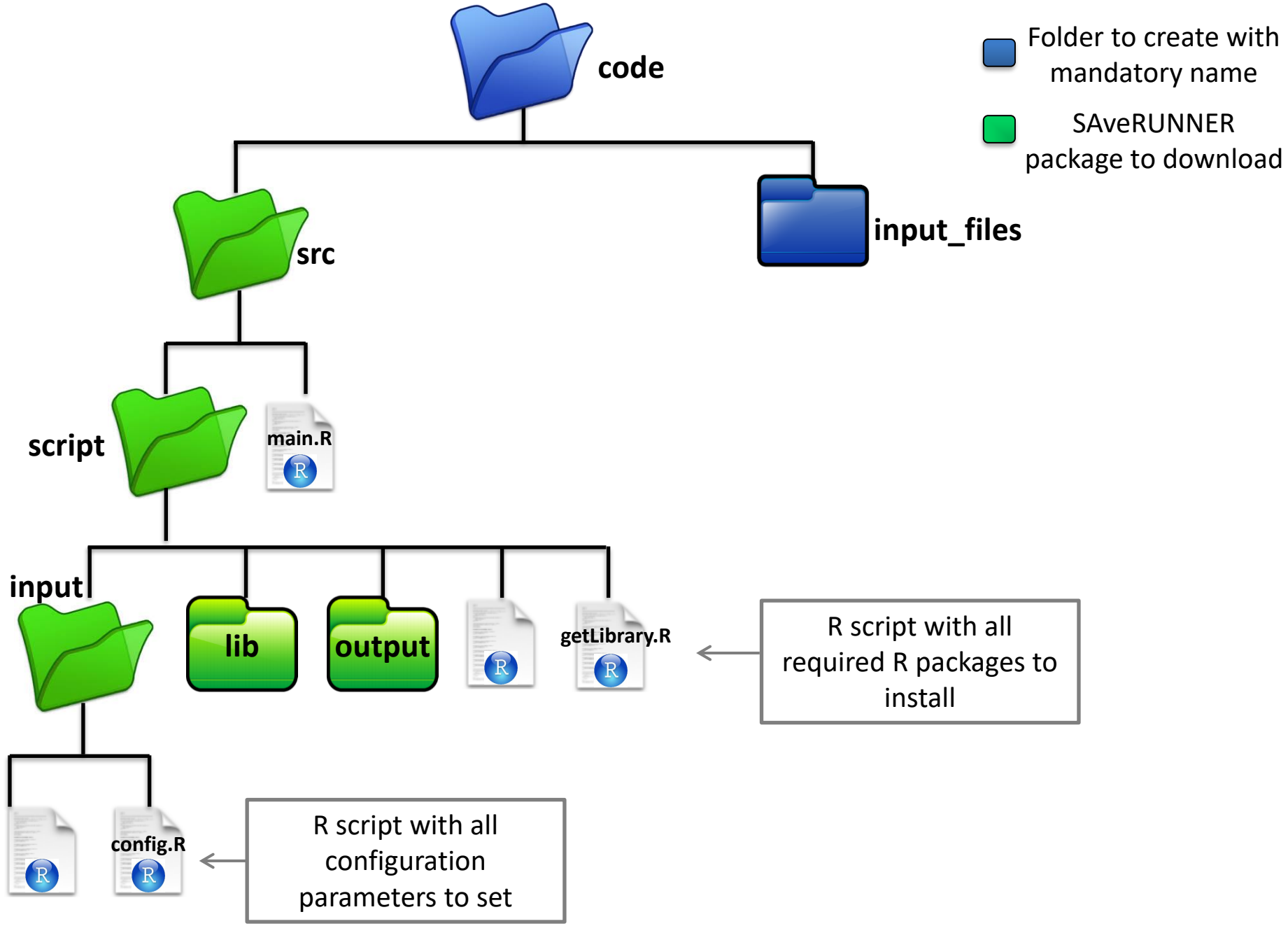
	A	B	C	D
1	Gene_A_Entrez ID	GeneB_Entrez_ID	data_source(s)	
2	1	310	IntAct,PINA	
3	1	368	IntAct,PINA	
4	1	1026	IntAct,PINA	
5	1	2886	IntAct,PINA	
6	1	3958	InnateDB	
7	1	4899	PINA	
8	1	6606	IntAct,PINA	
9	1	6622	IntAct,PINA	
10	1	7083	IntAct,PINA	
11	1	10321	HPRD,PINA	
12	1	10549	IntAct,PINA	
13	1	80854	IntAct,PINA	
14	2	60	IntAct,PINA	
15	2	250	IntAct	
16	2	259	BioGRID,HPRD,PINA	
17	2	292	IntAct	
18	2	293	IntAct	
19	2	309	HPRD	
20	2	310	IntAct,PINA	

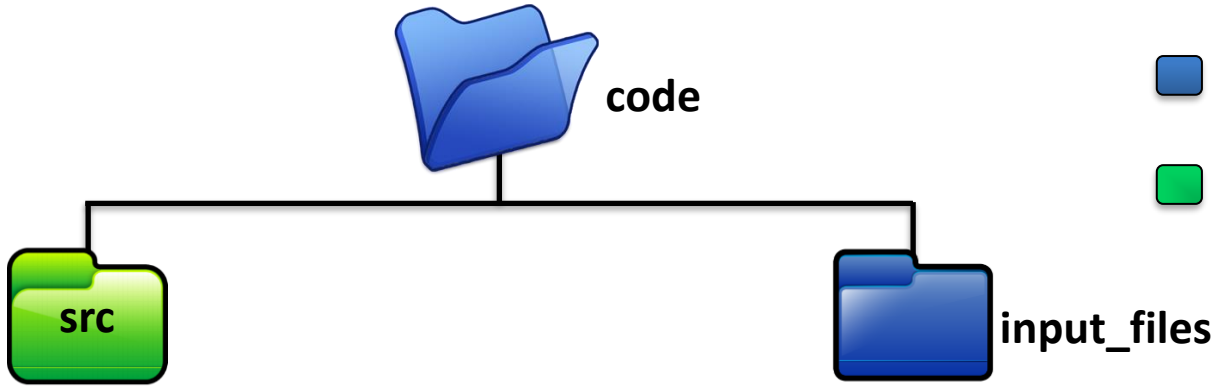
Input Data	Network	Reference
Human interactome Nodes = 15,970 Links = 217,160		Cheng et al. Nature Commun 2018
Disease Genes 14 diseases 874 disease genes SARS-CoV-2 targets 332 human proteins		Phenopedia https://phgkb.cdc.gov/PHGKB/startPagePhenoPedia.action Gordon et. al 2020
Drug Targets 1873 drugs 2166 drug targets		Drug Bank https://www.drugbank.ca/

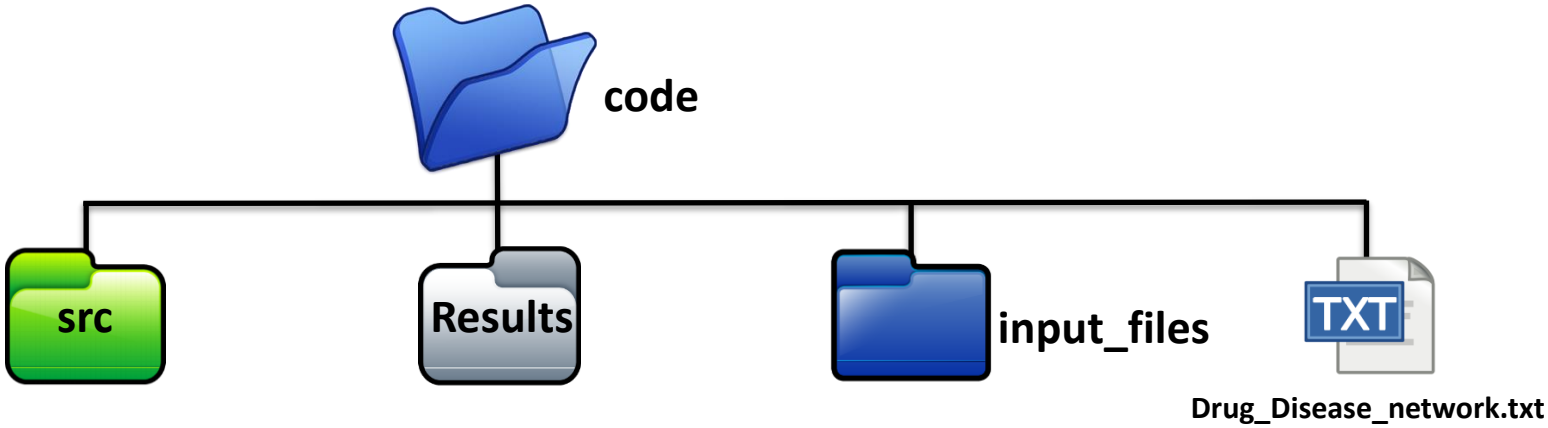

code
src
input_files
Folder to create with mandatory name
SAveRUNNER package to download


code
src
input_files
script
main.R
R main script to launch
all R scripts called by main.R
Folder to create with mandatory name
SAveRUNNER package to download


code
Folder to create with mandatory name
SAveRUNNER package to download
src
input_files
script
main.R
input
lib
output
getLibrary.R
R script with all required R packages to install


code
Folder to create with mandatory name
SAveRUNNER package to download
src
input_files
script
main.R
input
lib
output
getLibrary.R
R script with all required R packages to install
config.R
R script with all configuration parameters to set


code
src
input_files
Folder to create with mandatory name
SAveRUNNER package to download



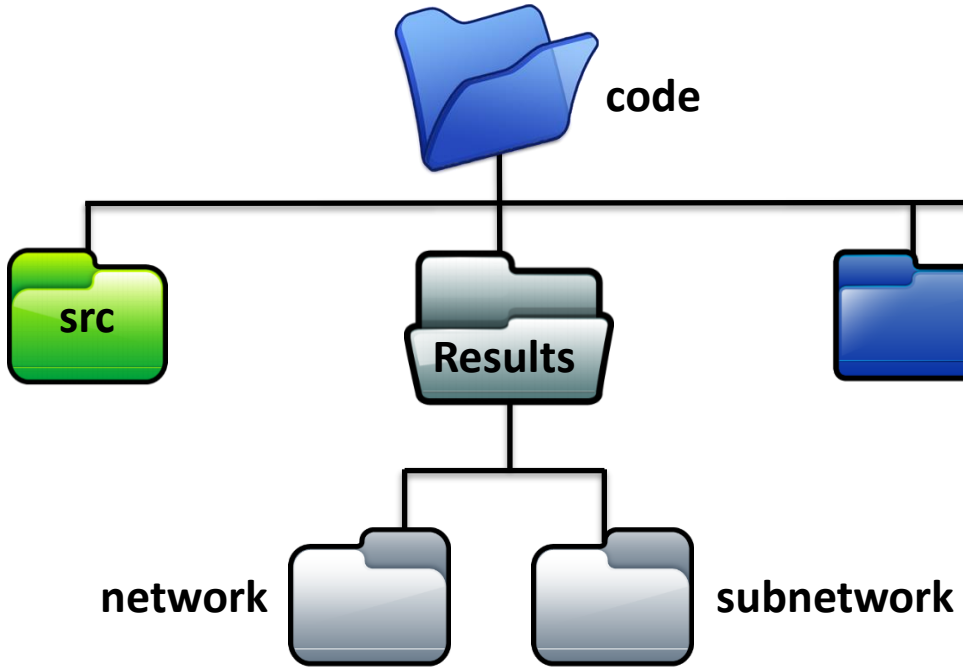
Drug_Disease_network:

a tab-delimited file created as output by SAveRUNNER reporting **all** the drug-disease associations with the **proximity** measure and corresponding **p-values**

Folder to create with mandatory name

SAveRUNNER package to download

Output folder



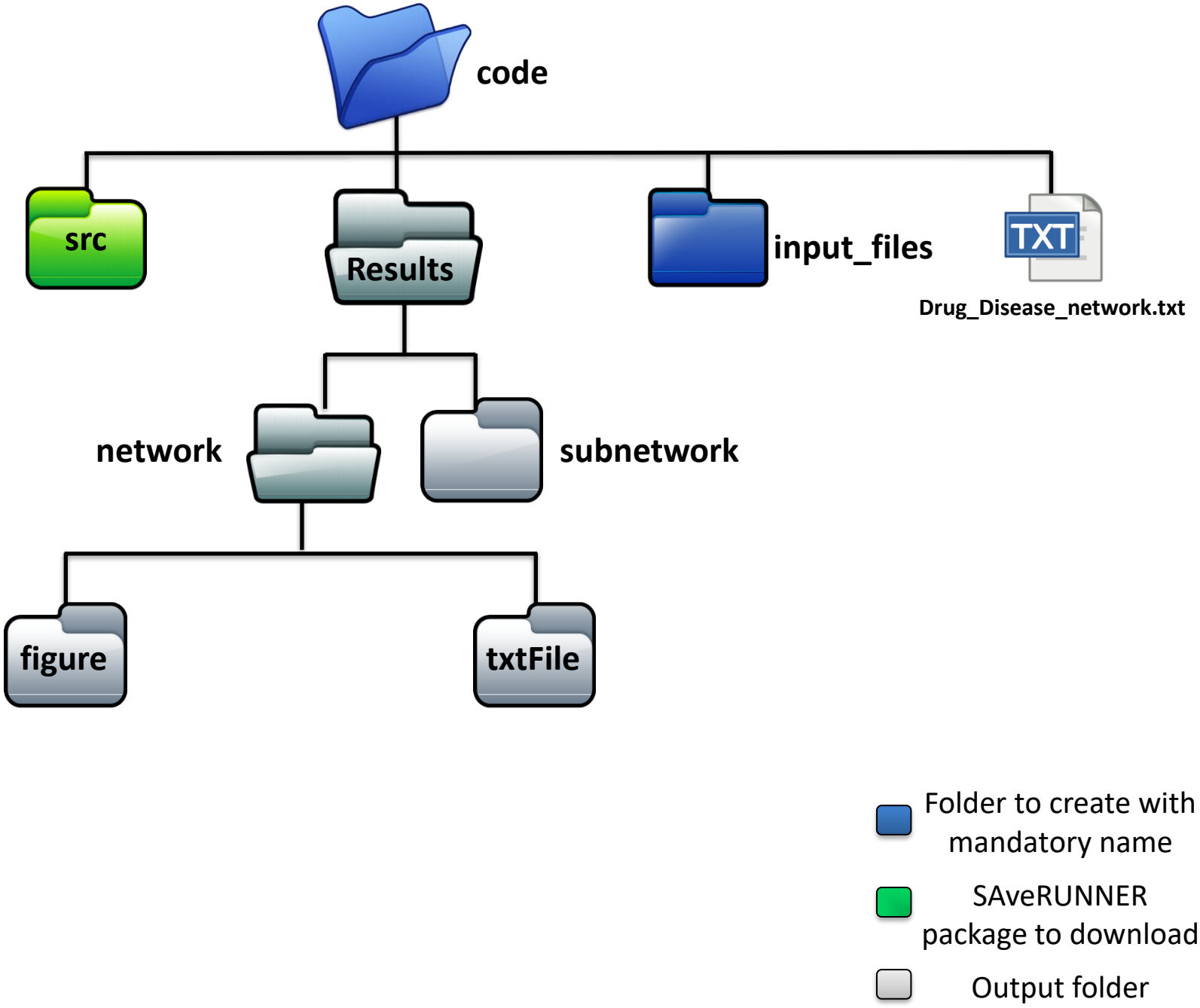
Drug_Disease_network:

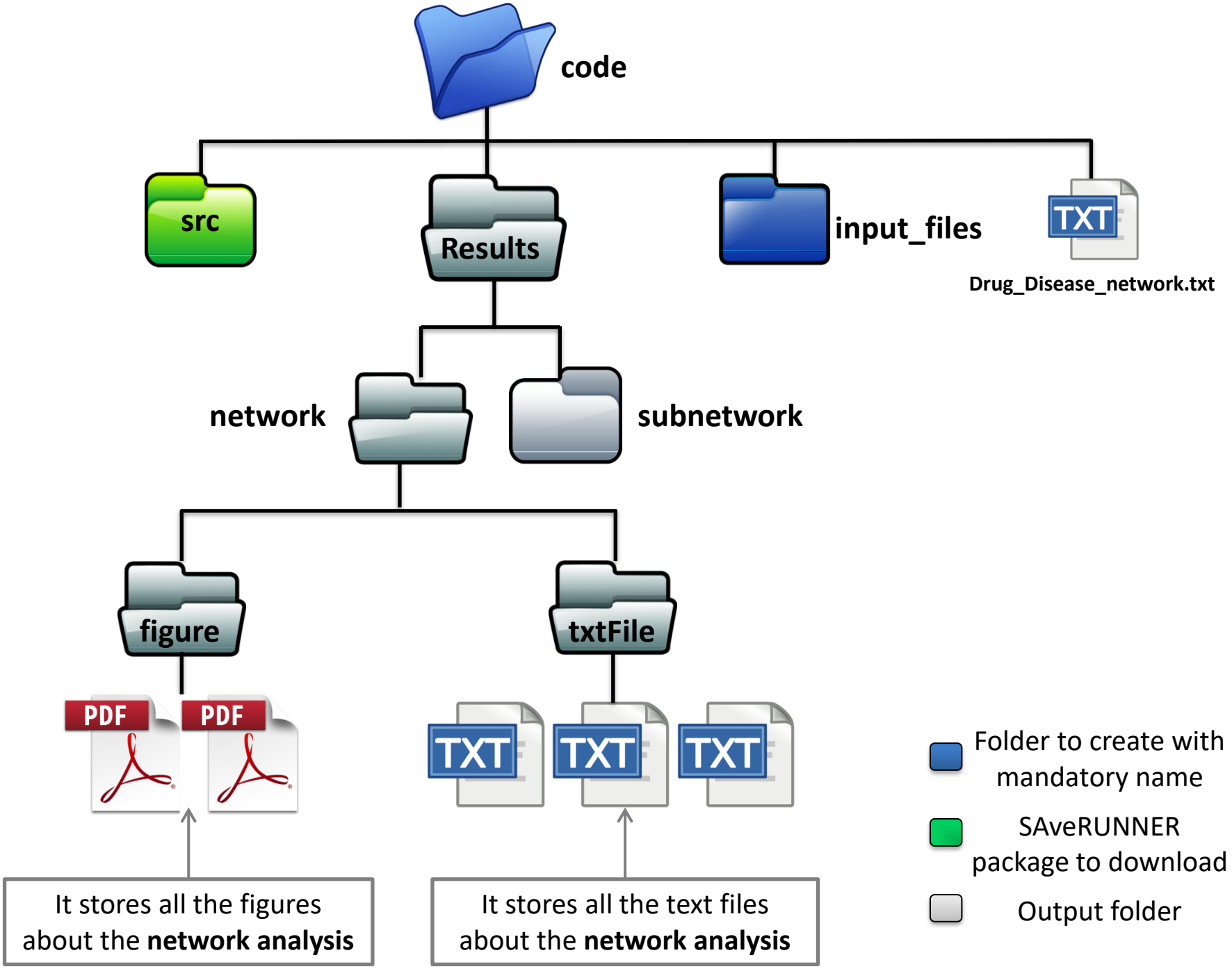
a tab-delimited file created as output by SAveRUNNER reporting **all** the drug-disease associations with the **proximity** measure and corresponding **p-values**

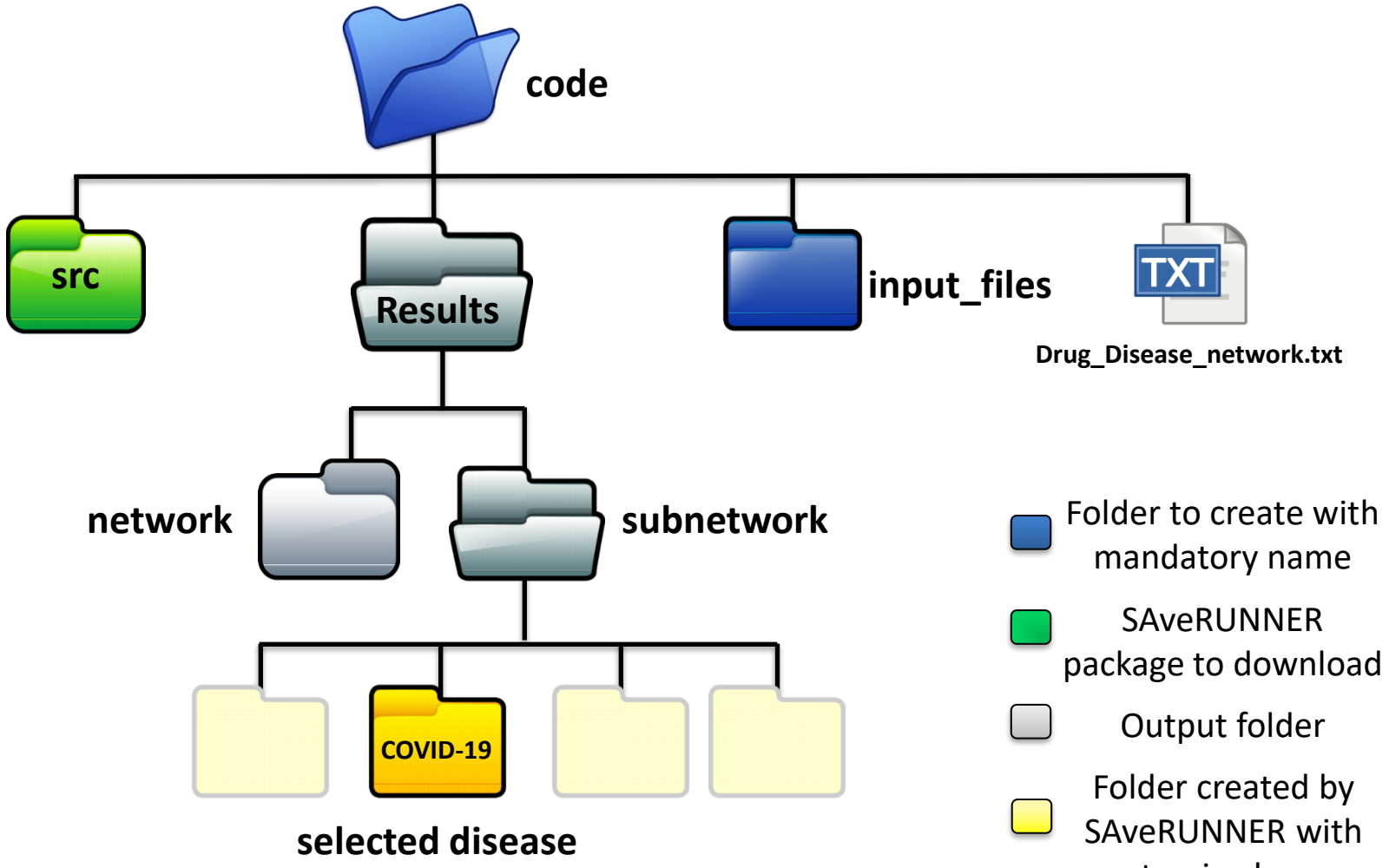
Folder to create with mandatory name

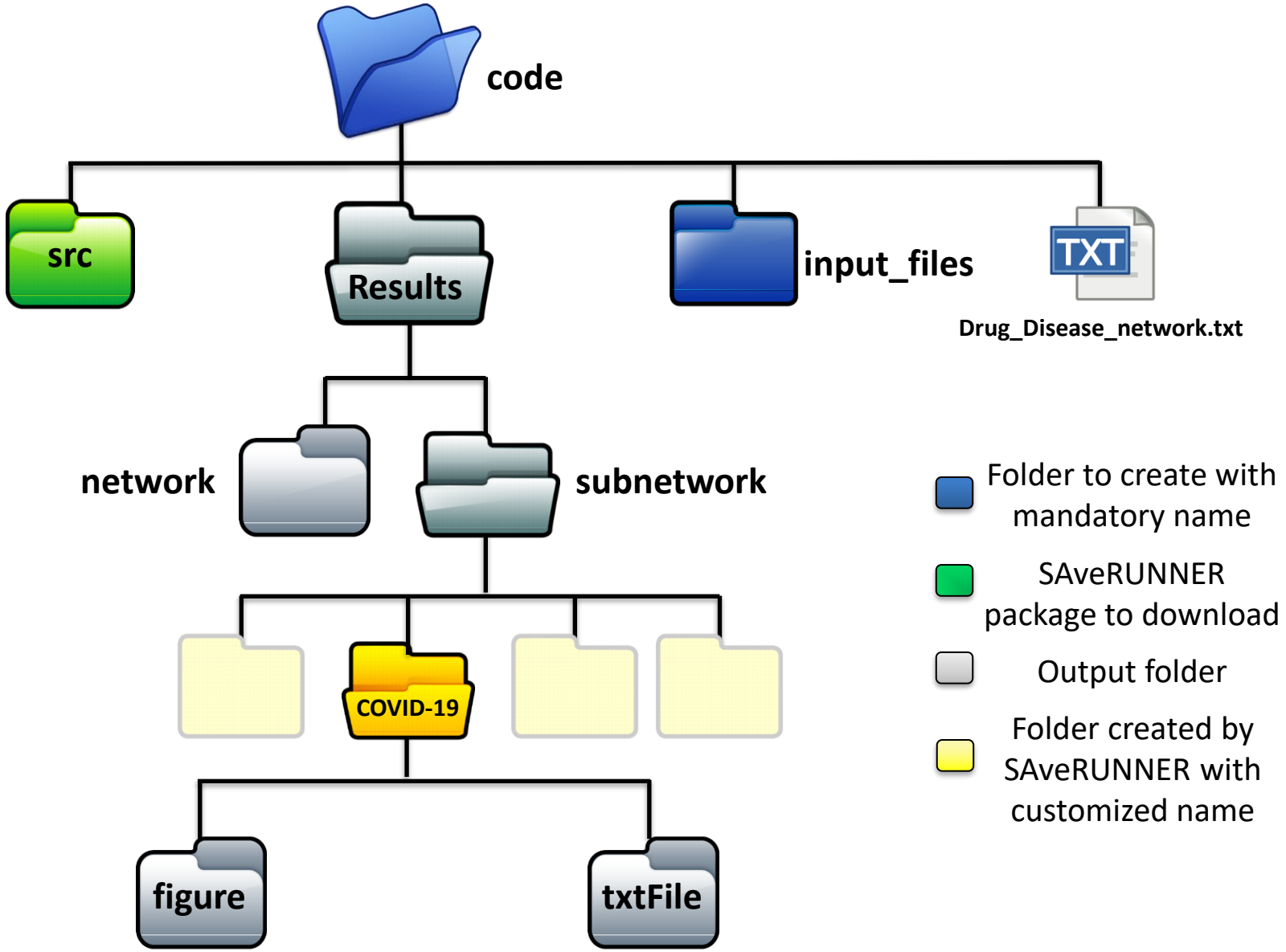
SAveRUNNER package to download

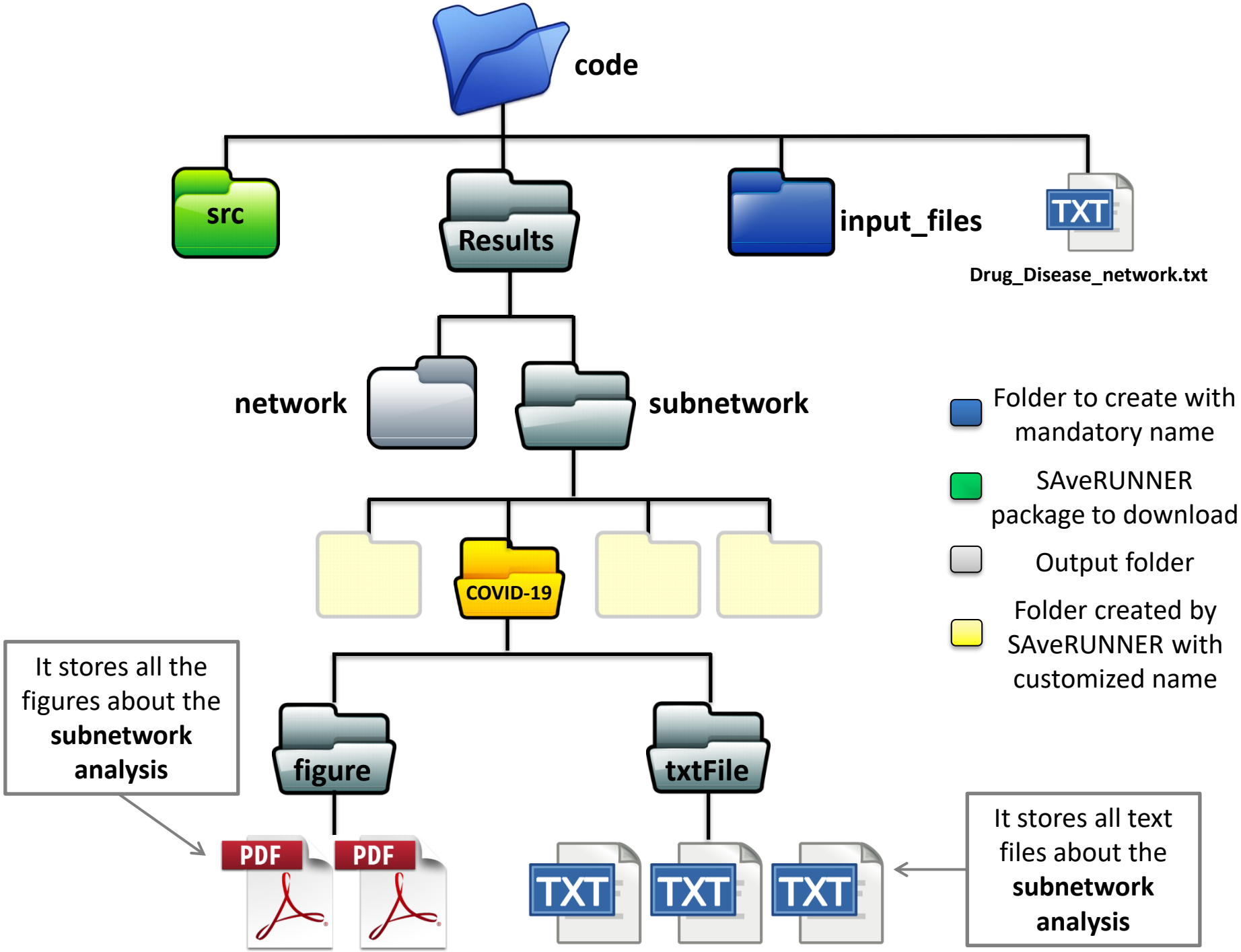
Output folder


code
src
Results
input_files
TXT
Drug_Disease_network.txt
network
subnetwork
figure
txtFile
Folder to create with mandatory name
SAveRUNNER package to download
Output folder

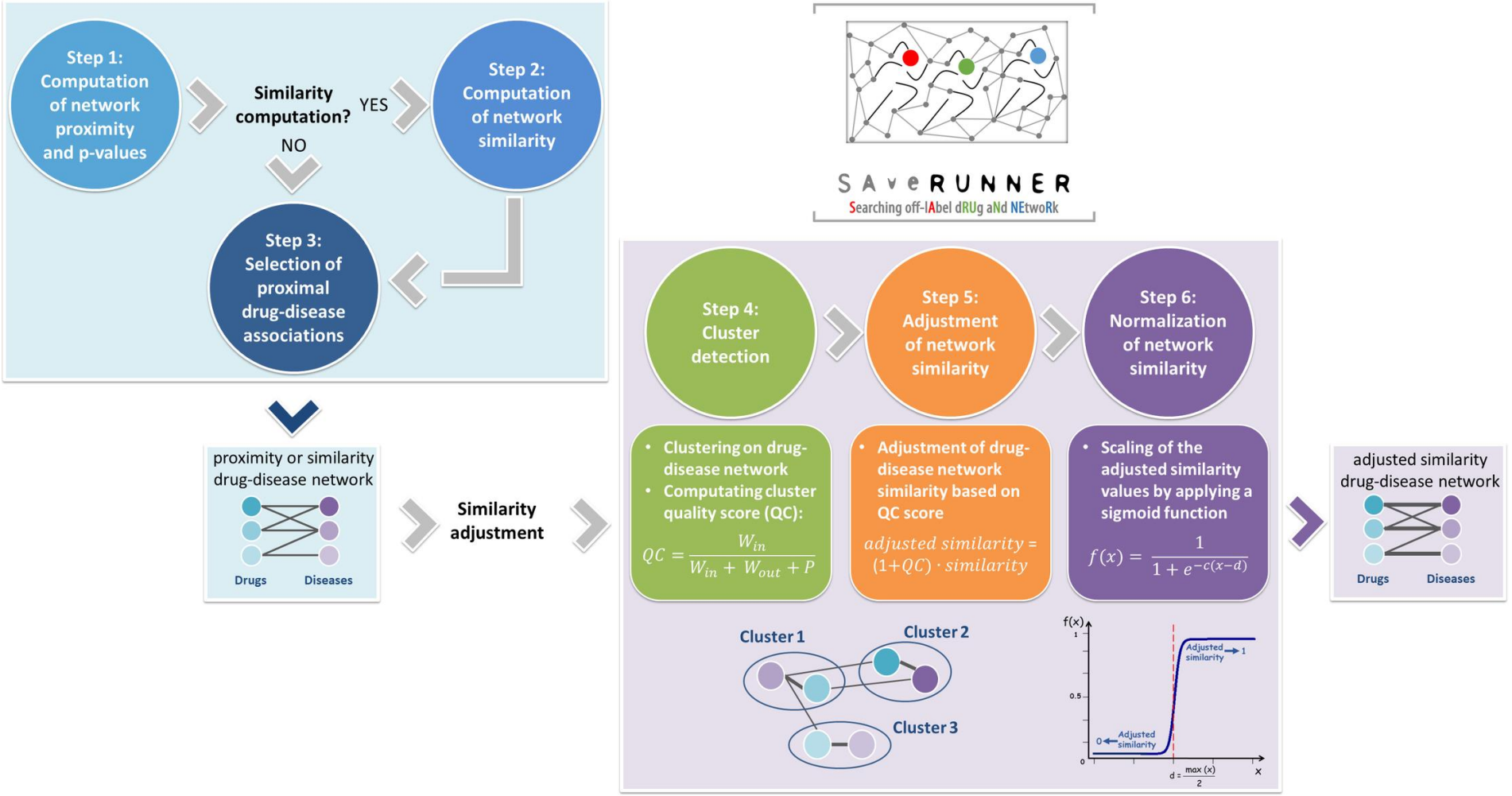

code
src
Results
input_files
TXT
Drug_Disease_network.txt
network
subnetwork
figure
txtFile
PDF
PDF
TXT
TXT
TXT
It stores all the figures about the **network analysis**
It stores all the text files about the **network analysis**
Folder to create with mandatory name
SAveRUNNER package to download
Output folder


code
src
Results
input_files
TXT
Drug_Disease_network.txt
network
subnetwork
COVID-19
selected disease
Folder to create with mandatory name
SAveRUNNER package to download
Output folder
Folder created by SAveRUNNER with customized name


code
src
Results
input_files
TXT
Drug_Disease_network.txt
network
subnetwork
COVID-19
figure
txtFile
Folder to create with mandatory name
SAveRUNNER package to download
Output folder
Folder created by SAveRUNNER with customized name


code
src
Results
input_files
TXT
Drug_Disease_network.txt
network
subnetwork
COVID-19
Folder to create with mandatory name
SAveRUNNER package to download
Output folder
Folder created by SAveRUNNER with customized name
It stores all the figures about the subnetwork analysis
figure
txtFile
PDF
PDF
TXT
TXT
TXT
It stores all text files about the subnetwork analysis

Ready to launch!


Step 1: Computation of network proximity and p-values
Similarity computation?
YES
NO
Step 2: Computation of network similarity
Step 3: Selection of proximal drug-disease associations
SAveRUNNER
Searching off-lAbel dRUg aNd NEtwoRk
proximity or similarity drug-disease network
Drugs
Diseases
Similarity adjustment
Step 4: Cluster detection
Step 5: Adjustment of network similarity
Step 6: Normalization of network similarity
• Clustering on drug-disease network
• Computating cluster quality score (QC):
$QC = \frac{W_{in}}{W_{in} + W_{out} + P}$
• Adjustment of drug-disease network similarity based on QC score
$adjusted\ similarity = (1+QC) \cdot similarity$
• Scaling of the adjusted similarity values by applying a sigmoid function
$f(x) = \frac{1}{1 + e^{-c(x-d)}}$
Cluster 1
Cluster 2
Cluster 3
f(x)
Adjusted similarity → 1
Adjusted similarity → 0
d = max (x) / 2
x
adjusted similarity drug-disease network
Drugs
Diseases

Main file

Initial settings

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ####################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ####################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ####################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ####################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ####################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ####################################
30
```

Set the working directory

Environment is empty

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

Main file

Initial settings

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ####################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ####################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ####################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ####################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ####################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ####################################
30
```

30:1 (Untitled) R Script

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

```
>
```

Environment History Connections

Import Dataset List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Delete Rename More

ıe › Documents › Didattica › class › BioinformaticsII › lessons › 6_SaveRUNNER › Code › src

Name	Size	Modified
..		
main.R	868 B	Sep 19, 2020, 11:20 AM
script		

Main file

Initial settings

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ###################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ###################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ###################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ###################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ###################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ###################################
30
```

30:1 (Untitled) R Script

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

```
>
```

Environment History Connections

Import Dataset List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Delete Rename More

uments > Didattica > class > BioinformaticsII > lessons > 6_SaveRUNNER > Code > src > script

Name	Size	Modified
..		
getLibrary.R	193 B	Apr 19, 2020, 11:12 AM
getSource.R	2.5 KB	Sep 19, 2020, 12:31 AM
input		
lib		
output		

Main file

Initial settings

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ####################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ####################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ####################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ####################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ####################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ####################################
30
```

30:1 (Untitled) R Script

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

```
>
```

Environment History Connections

Import Dataset List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Delete Rename More

Didattica > class > BioinformaticsII > lessons > 6_SaveRUNNER > Code > src > script > input

Name	Size	Modified
..		
config.R	2.1 KB	Sep 19, 2020, 11:20 AM
inputFiles.R	1 KB	Sep 18, 2020, 7:28 PM

Configuration file

Initial settings

main.R | config.R

Source on Save | Run | Source

```
config <- function(){

  # for executing SAveRUNER launch:
  # source('~/SaveRUNNER/Code/src/main.R')

  ################################
  # parameters for computing start network with edge-weight =

  diseases <- c("COVID-19", "Severe Acute Respiratory Syndrome",
                "Arthritis, Rheumatoid","Diabetes Mellitus",
                "Respiratory Distress Syndrome, Adult", "Arrhythmia","Atherosclerosis",
                "Cardiomyopathies", "Heart Arrest", "Hypertension",
                "HIV Infections", "Multiple Sclerosis", "Hemorrhagic Fever, Ebola",
                "Influenza, Human", "Malaria")
  ################################
  # parameters for computing end network

  dirRes <- "Results/"

  interaction = "similarity"  # edge-weight = similarity or proximity
  pval_thr = 0.05             # select significative drug-disease association
  adjust_link = T            # adjust similarity or not
  new_link = F                # add new drug-disease association or not (without compute pval)
  ################################
  # parameters for making figure

  if( (interaction == "proximity") ) distance = "proximity"
  if( (interaction == "similarity") & (adjust_link == F) ) distance = "similarity"
  if( (interaction == "similarity") & (adjust_link == T) ) distance = "adjusted_similarity"
  ################################
```

17:3 (Untitled) R Script

Short command line to launch SAveRUNNER

Environment | History | Connections

Import Dataset | List

Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

Didattica > class > BioinformaticsII > lessons > 6_SaveRUNNER > Code > src > script > input

Name	Size	Modified
..		
config.R	2 KB	Sep 19, 2020, 10:50 AM
inputFiles.R	1 KB	Sep 18, 2020, 7:28 PM

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

>

Configuration file

Initial settings

main.R | config.R

Source on Save | Run | Source

```
1  config <- function(){
2
3    # for executing SAveRUNER launch:
4    # source('~/SaveRUNNER/Code/src/main.R')
5
6    ###################################
7    # parameters for computing start network with edge-weight = proximity
8
9    diseases <- c("COVID-19", "Severe Acute Respiratory Syndrome",
10                 "Arthritis, Rheumatoid","Diabetes Mellitus",
11                 "Respiratory Distress Syndrome, Adult", "Arrhythmia","Athero
12                 "Cardiomyopathies", "Heart Arrest", "Hypertension",
13                 "HIV Infections", "Multiple Sclerosis", "Hemorrhagic Fever, Ebola",
14                 "Influenza, Human", "Malaria")
15   ###################################
16   # parameters for computing end net
17
18   dirRes <- "Results/"
19
20   interaction = "similarity"  # edge
21   pval_thr = 0.05             # select significative drug-disease association
22   adjust_link = T            # adjust similarity or not
23   new_link = F                # add new drug-disease association or not (without compute pval)
24   ###################################
25   # parameters for making figure
26
27   if( (interaction == "proximity") ) distance = "proximity"
28   if( (interaction == "similarity") & (adjust_link == F) ) distance = "similarity"
29   if( (interaction == "similarity") & (adjust_link == T) ) distance = "adjusted_similarity"
30   ###################################
```

17:3 (Untitled) R Script

Insert names of *diseases* to be tested

Insert name of *Output* folder

⚠ **Caveat:** The disease names should be exactly the same as reported in the input files of *Disease Genes* (e.g., Phenopedia)

Environment | History | Connections

Import Dataset | List

Global Environment

Environment is empty

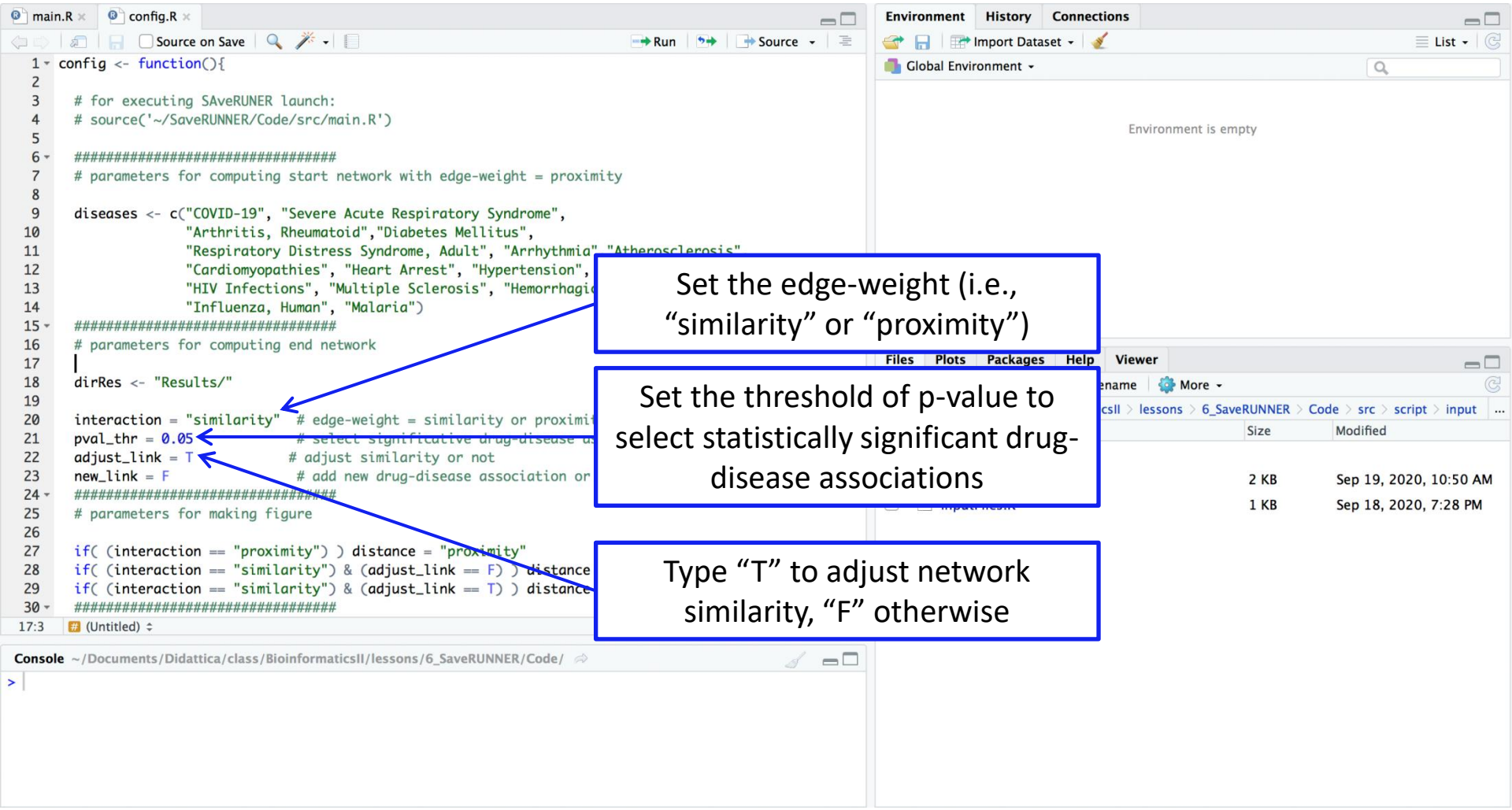
Name	Size	Modified
..		
config.R	2 KB	Sep 19, 2020, 10:50 AM
inputFiles.R	1 KB	Sep 18, 2020, 7:28 PM

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

>

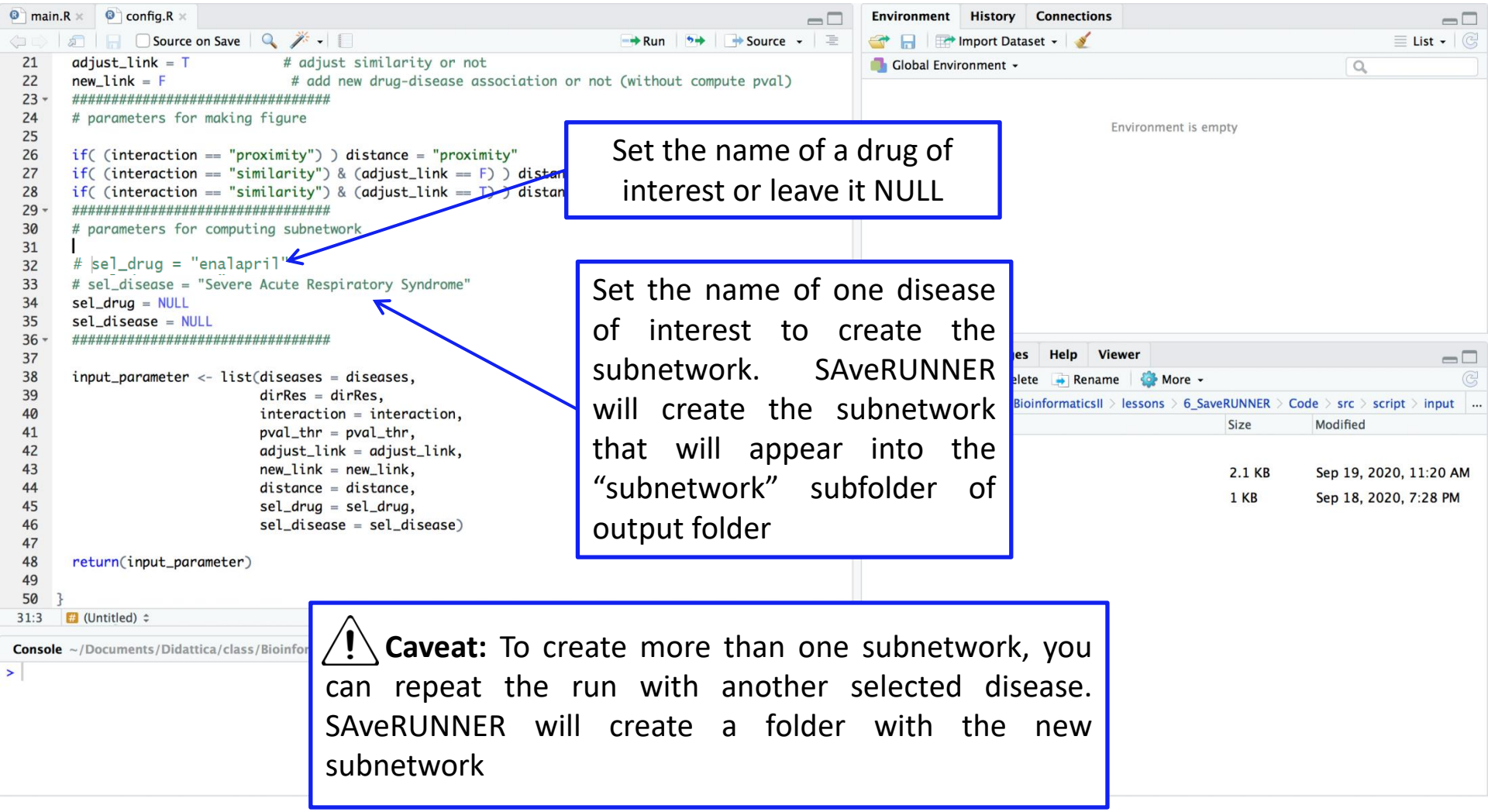
Configuration file

Initial settings



Configuration file

Initial settings



Set the name of a drug of interest or leave it NULL

Set the name of one disease of interest to create the subnetwork. SAveRUNNER will create the subnetwork that will appear into the "subnetwork" subfolder of output folder

⚠ **Caveat:** To create more than one subnetwork, you can repeat the run with another selected disease. SAveRUNNER will create a folder with the new subnetwork

Main file

Input files

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ####################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ####################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ####################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ####################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ####################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ####################################
30
```

30:1 (Untitled) R Script

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

```
>
```

Environment History Connections

Import Dataset List

Global Environment

Environment is empty

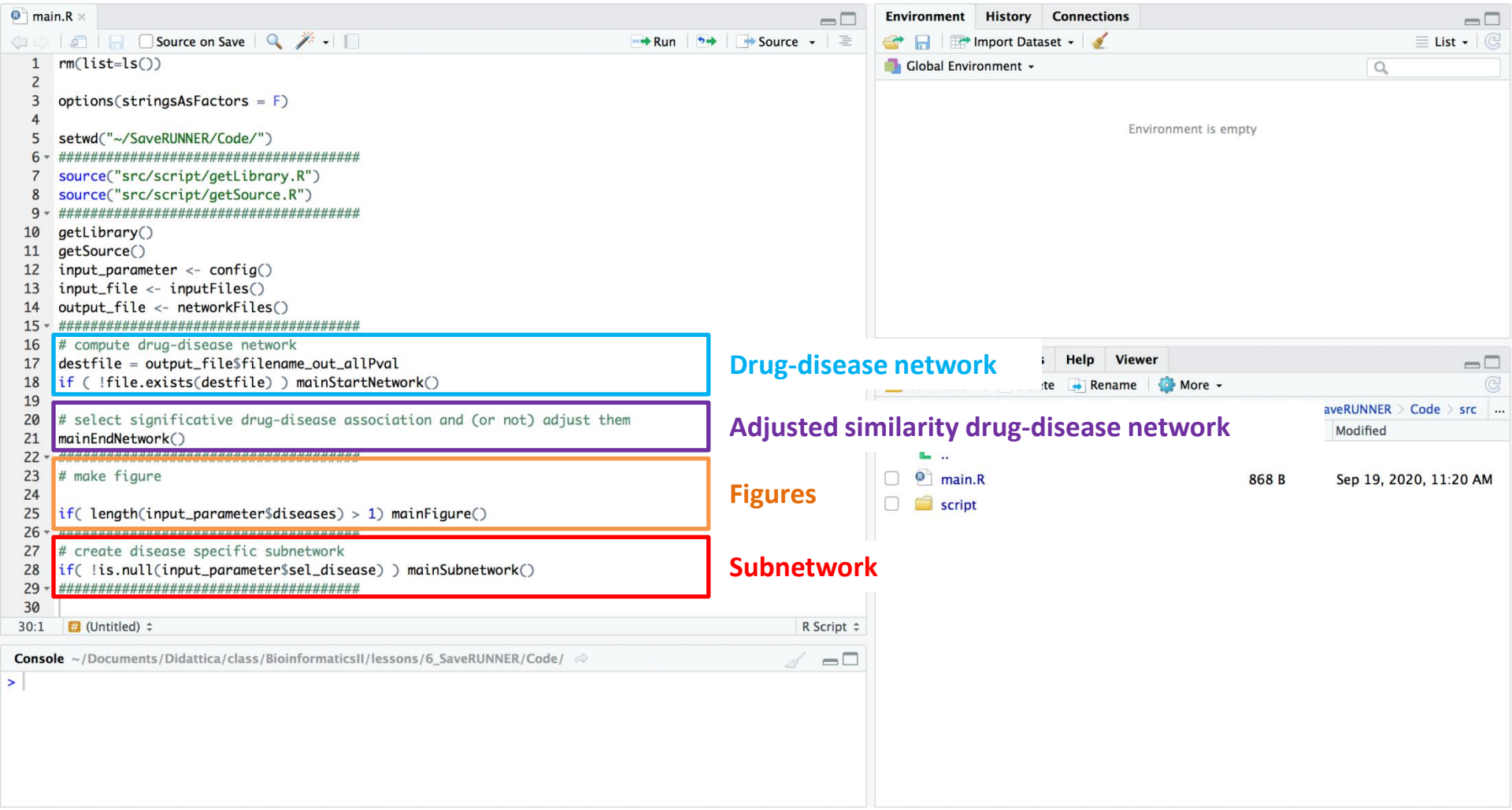
Files Plots Packages Help Viewer

New Folder Delete Rename More

ıe › Documents › Didattica › class › BioinformaticsII › lessons › 6_SaveRUNNER › Code › src

Name	Size	Modified
..		
main.R	868 B	Sep 19, 2020, 11:20 AM
script		

Main file


main.R
Source on Save
Run
Source
rm(list=ls())
options(stringsAsFactors = F)
setwd("~/SaveRUNNER/Code/")
source("src/script/getLibrary.R")
source("src/script/getSource.R")
getLibrary()
getSource()
input_parameter <- config()
input_file <- inputFiles()
output_file <- networkFiles()
# compute drug-disease network
destfile = output_file$filename_out_allPval
if (!file.exists(destfile)) mainStartNetwork()
# select significative drug-disease association and (or not) adjust them
mainEndNetwork()
# make figure
if(length(input_parameter$diseases) > 1) mainFigure()
# create disease specific subnetwork
if(!is.null(input_parameter$sel_disease)) mainSubnetwork()
30:1
(Untitled)
R Script
Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Help
Viewer
Rename
More
aveRUNNER
Code
src
Modified
main.R
868 B
Sep 19, 2020, 11:20 AM
script
Drug-disease network
Adjusted similarity drug-disease network
Figures
Subnetwork

Main file

Drug-disease network

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ####################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ####################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ####################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ####################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ####################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ####################################
30
```

Environment | History | Connections

Import Dataset | List

Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

ıe > Documents > Didattica > class > BioinformaticsII > lessons > 6_SaveRUNNER > Code > src

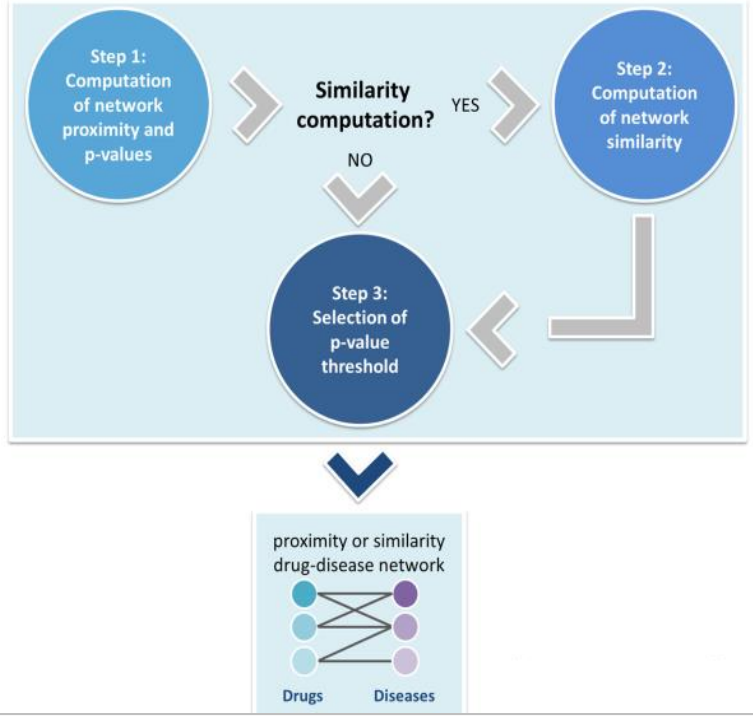
Name	Size	Modified
..		
main.R	868 B	Sep 19, 2020, 11:20 AM
script		

⚠ **Caveat:** This step could be very time consuming. However, once the drug-disease network is created, SAveRUNNER can repeat the analysis without re-computing this network.

mainStartNetwork.R

```
mainStartNetwork <- function(){

  ######################################
  # input parameters
  diseases <- input_parameter$diseases

  # input files
  interactome <- input_file$interactome
  disease_gene <- input_file$disease_gene
  drug_target <- input_file$drug_target

  # output files
  filename_out_allPval <- output_file$filename_out_allPval
  ######################################

  graph_info <- getGraph(interactome)

  list_disease <- split(disease_gene,disease_gene$disease)
  list_disease <- list_disease[diseases]

  list_drug <- split(drug_target,drug_target$Drug)

  drug_disease_net <- computeDrugDiseaseNetwork(list_disease,list_drug,graph_info)

  drug_disease_net <- drug_disease_net[is.finite(drug_disease_net$proximity),]

  ind <- which(is.na(drug_disease_net$pval))
  if(length(ind)>0) drug_disease_net <- drug_disease_net[-ind,]

  write.table(drug_disease_net, filename_out_allPval, sep = "\t", row.names = F, col.names = T,
quote = F)

}
```

- This function releases the **drug-disease network** (Drug_Disease_network.txt) that stores the **proximity** measure computed between each drug-disease pair with the corresponding p-values (Step 1).



computeDrugDiseaseNetwork.R

Step 1

```
computeDrugDiseaseNetwork <- function(list_disease,list_drug,graph_info){

  tmp <- lapply(list_disease, function(x){

    disease_genes <- x[,"GeneID"]
    disease <- unique(x[,"disease"])

    DG_degree_distribution <- computeDegreeDistribution(disease_genes,graph_info)

    if( nrow(DG_degree_distribution) > 0 ){

      tmp2 <-lapply(list_drug, function(y){

        drug_targets <- y[,"GeneID"]
        drug <- unique(y[,"Drug"])

        DT_degree_distribution <- computeDegreeDistribution(drug_targets,graph_info)

        if( nrow(DT_degree_distribution) > 0 ){

          proximity <- computeProximity(drug_targets,disease_genes,graph_info)

          random_distribution_proximity <- computeRandomProximity(DT_degree_distribution,DG_degree_distribution,graph_info)

          stat <- computeStatistics(random_distribution_proximity,proximity)

          list <- list(drug = drug, proximity = proximity, zscore = stat$z, pval = stat$pval)

          return(list)

        }

      })

      res <- rbindlist(tmp2)

      df <- data.frame(disease = disease, res)

    }

    })

  drug_disease_net <- rbindlist(tmp)

  return(drug_disease_net)

}
```

- This function builds the **drug-disease network**

drug_disease_net | computeDrugDiseaseNetwork.R

Filter

	disease	drug	proximity	pval
1	COVID-19	2-mercaptobenzothiazole	2.0000000	0.4718848020
2	COVID-19	5-o-phosphono-alpha-d-ribofuranosyl diphosphate	2.0000000	0.9500349030
3	COVID-19	5cocktail	1.4166667	0.1798218970
4	COVID-19	abaloparatide	1.0000000	0.1151683790
5	COVID-19	abarelix	2.0000000	0.5000000000
6	COVID-19	abatacept	2.0000000	0.7070198020
7	COVID-19	abciximab	1.9230769	0.9910414400
8	COVID-19	abiraterone	1.0000000	0.1160145670
9	COVID-19	acalabrutinib	1.0000000	0.6679458630
10	COVID-19	acamprosate	1.8095238	0.6135531050
11	COVID-19	acarbose	1.7500000	0.1471950820
12	COVID-19	acebutolol	1.0000000	0.1450095480
13	COVID-19	aceclofenac	1.5000000	0.3774438720
14	COVID-19	acemetacin	1.5000000	0.3702033890
15	COVID-19	acenocoumarol	1.0000000	0.4007604840

computeDrugDiseaseNetwork.R

Step 1

```
computeDrugDiseaseNetwork <- function(list_disease,list_drug,graph_info){

 tmp <- lapply(list_disease, function(x){

    disease_genes <- x[,"GeneID"]
    disease <- unique(x[,"disease"])

    DG_degree_distribution <- computeDegreeDistribution(disease_genes,graph_info)

    if( nrow(DG_degree_distribution) > 0 ){

     tmp2 <-lapply(list_drug, function(y){

       drug_targets <- y[,"GeneID"]
       drug <- unique(y[,"Drug"])

       DT_degree_distribution <- computeDegreeDistribution(drug_targets,graph_info)

       if( nrow(DT_degree_distribution) > 0 ){

         proximity <- computeProximity(drug_targets,disease_genes,graph_info)

         random_distribution_proximity <- computeRandomProximity(DT_degree_distribution,DG_degree_distribution,graph_info)

         stat <- computeStatistics(random_distribution_proximity,proximity)

         list <- list(drug = drug, proximity = proximity, zscore = stat$z, pval = stat$pval)

         return(list)

       }

     })

     res <- rbindlist(tmp2)

     df <- data.frame(disease = disease, res)

    }

 })

 drug_disease_net <- rbindlist(tmp)

 return(drug_disease_net)

}
```

- This function builds the **drug-disease network**

drug_disease_net | computeDrugDiseaseNetwork.R

Filter

	disease	drug	proximity	pval
1	COVID-19	2-mercaptobenzothiazole	2.0000000	0.4718848020
2	COVID-19	5-o-phosphono-alpha-d-ribofuranosyl diphosphate	2.0000000	0.9500349030
3	COVID-19	5cocktail	1.4166667	0.1798218970
4	COVID-19	abaloparatide	1.0000000	0.1151683790
5	COVID-19	abarelix	2.0000000	0.5000000000
6	COVID-19	abatacept	2.0000000	0.7070198020
7	COVID-19	abciximab	1.9230769	0.9910414400
8	COVID-19	abiraterone	1.0000000	0.1160145670
9	COVID-19	acalabrutinib	1.0000000	0.6679458630
10	COVID-19	acamprosate	1.8095238	0.6135531050
11	COVID-19	acarbose	1.7500000	0.1471950820
12	COVID-19	acebutolol	1.0000000	0.1450095480
13	COVID-19	aceclofenac	1.5000000	0.3774438720
14	COVID-19	acemetacin	1.5000000	0.3702033890
15	COVID-19	acenocoumarol	1.0000000	0.4007604840

computeDegreeDistribution.R

Step 1

```
computeDegreeDistribution <- function(list){

  from <- which(node %in% list)
  d <- degree(graph, v = V(graph)[from])

  t <- table(d)

  degree_sorted <- as.numeric(names(t))
  freq <- as.numeric(t)

  df <- data.frame(degree = degree_sorted, frequency = freq)

  return(df)

}
```

- This function computes the **degree distribution** in the human interactome for the list of **disease genes** and **drug targets**

Filter

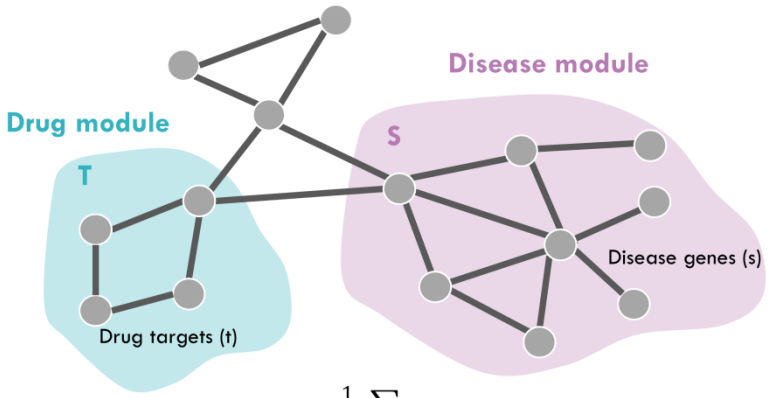
	degree	frequency
1	0	2
2	1	22
3	2	20
4	3	26
5	4	11
6	5	20
7	6	15
8	7	10
9	8	11
10	9	14
11	10	13
12	11	7
13	12	7
14	13	5
15	14	4

computeProximity.R

Step 1

```
computeProximity <- function(list1,list2,graph_info){

  graph <- graph_info$graph
  node <- graph_info$node

  from <- which(node %in% list1)
  to <- which(node %in% list2)

  distance_matrix <- distances(graph, v = V(graph)[from], to = V(graph)[to])

  minimum <- computeMinimum(distance_matrix)

  proximity <- mean(minimum, na.rm = T)

  return(proximity)

}
```

- This function computes the **proximity** between the lists of drug targets and disease genes



$$p(T,S) = \frac{1}{\|T\|}\sum_{t \epsilon T} \min_{s \epsilon S} d(t,s)$$

T = set of drug targets
S = set of disease genes
p = average shortest path length between drug targets t in the drug module T and the nearest disease genes s in the disease module S

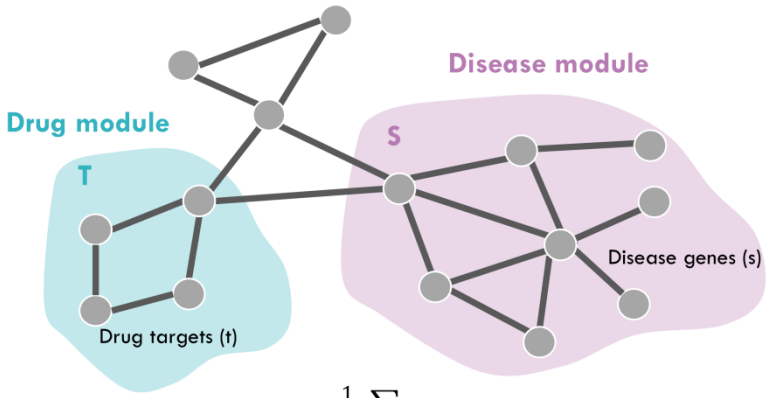
computeProximity.R

Step 1

```
computeProximity <- function(list1,list2,graph_info){

  graph <- graph_info$graph
  node <- graph_info$node

  from <- which(node %in% list1)
  to <- which(node %in% list2)

  distance_matrix <- distances(graph, v = V(graph)[from], to = V(graph)[to])

  minimum <- computeMinimum(distance_matrix)

  proximity <- mean(minimum, na.rm = T)

  return(proximity)

}
```

```
computeMinimum <- function(distance_matrix,dim=1,perc_thr=5){

  # dim = 1 -> row mean (ie, from DrugTarget to DiseaseGene),
  # dim = 2 -> col mean (ie, from DiseaseGene to DrugTarget)

  minimum <- apply(distance_matrix,dim,min,na.rm = T)

  count <- length(minimum[is.infinite(minimum)])

  perc <- ( count / length(minimum) ) * 100

  if( perc > perc_thr ){

    minimum <- minimum

  }else{

    minimum <- minimum[is.finite(minimum)]

  }

}
```

- This function computes the **proximity** between the lists of drug targets and disease genes



$$p(T,S) = \frac{1}{\|T\|}\sum_{t\epsilon T} \min_{s\epsilon S} d(t,s)$$

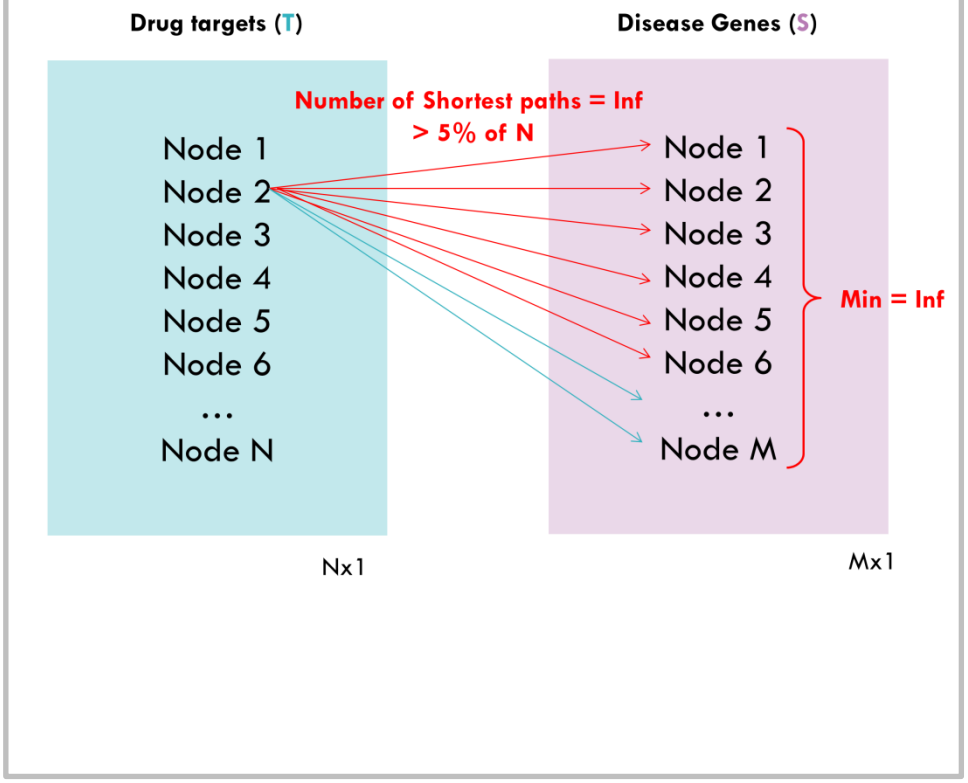
T = set of drug targets
S = set of disease genes
p = average shortest path length between drug targets t in the drug module T and the nearest disease genes s in the disease module S

computeProximity.R

Step 1

```
computeProximity <- function(list1,list2,graph_info){

  graph <- graph_info$graph
  node <- graph_info$node

  from <- which(node %in% list1)
  to <- which(node %in% list2)

  distance_matrix <- distances(graph, v = V(graph)[from], to = V(graph)[to])

  minimum <- computeMinimum(distance_matrix)

  proximity <- mean(minimum, na.rm = T)

  return(proximity)

}
```

```
computeMinimum <- function(distance_matrix,dim=1,perc_thr=5){

  # dim = 1 -> row mean (ie, from DrugTarget to DiseaseGene),
  # dim = 2 -> col mean (ie, from DiseaseGene to DrugTarget)

  minimum <- apply(distance_matrix,dim,min,na.rm = T)

  count <- length(minimum[is.infinite(minimum)])

  perc <- ( count / length(minimum) ) * 100

  if( perc > perc_thr ){

    minimum <- minimum

  }else{

    minimum <- minimum[is.finite(minimum)]

  }

}
```



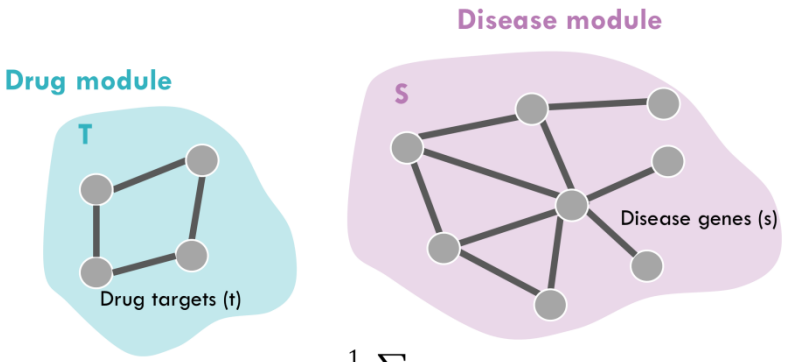
computeProximity.R

Step 1

```
computeProximity <- function(list1,list2,graph_info){

  graph <- graph_info$graph
  node <- graph_info$node

  from <- which(node %in% list1)
  to <- which(node %in% list2)

  distance_matrix <- distances(graph, v = V(graph)[from], to = V(graph)[to])

  minimum <- computeMinimum(distance_matrix)

  proximity <- mean(minimum, na.rm = T)

  return(proximity)

}
```

```
computeMinimum <- function(distance_matrix,dim=1,perc_thr=5){

  # dim = 1 -> row mean (ie, from DrugTarget to DiseaseGene),
  # dim = 2 -> col mean (ie, from DiseaseGene to DrugTarget)

  minimum <- apply(distance_matrix,dim,min,na.rm = T)

  count <- length(minimum[is.infinite(minimum)])

  perc <- ( count / length(minimum) ) * 100

  if( perc > perc_thr ){

    minimum <- minimum

  }else{

    minimum <- minimum[is.finite(minimum)]

  }

}
```

- If **Min = inf** means that the list of drug targets and disease genes are disconnected in the human interactome, i.e. there is no path connecting them



$$p(T,S) = \frac{1}{\|T\|}\sum_{t \epsilon T} \min_{s \epsilon S} d(t,s)$$

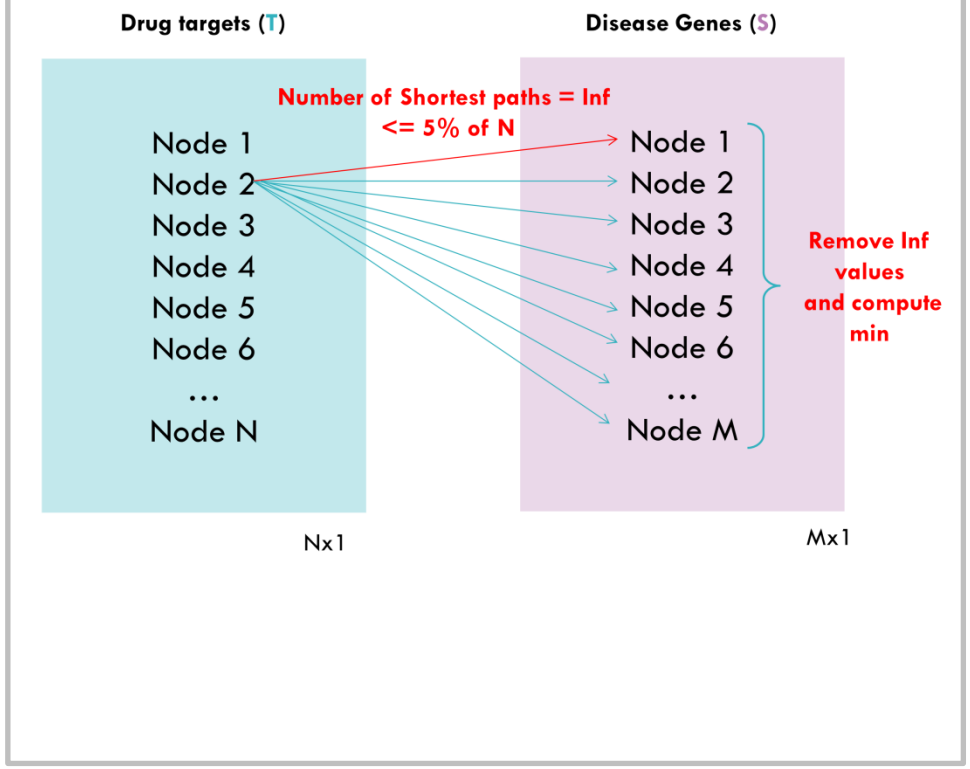
T = set of drug targets
S = set of disease genes
p = average shortest path length between drug targets t in the drug module T and the nearest disease genes s in the disease module S

computeProximity.R

Step 1

```
computeProximity <- function(list1,list2,graph_info){

  graph <- graph_info$graph
  node <- graph_info$node

  from <- which(node %in% list1)
  to <- which(node %in% list2)

  distance_matrix <- distances(graph, v = V(graph)[from], to = V(graph)[to])

  minimum <- computeMinimum(distance_matrix)

  proximity <- mean(minimum, na.rm = T)

  return(proximity)

}
```

```
computeMinimum <- function(distance_matrix,dim=1,perc_thr=5){

  # dim = 1 -> row mean (ie, from DrugTarget to DiseaseGene),
  # dim = 2 -> col mean (ie, from DiseaseGene to DrugTarget)

  minimum <- apply(distance_matrix,dim,min,na.rm = T)

  count <- length(minimum[is.infinite(minimum)])

  perc <- ( count / length(minimum) ) * 100

  if( perc > perc_thr ){

    minimum <- minimum

  }else{

    minimum <- minimum[is.finite(minimum)]

  }

}
```



computeRandomProximity.R

Step 1

```
computeRandomProximity <- function(targets_degree_distribution,genes_degree_distribution,graph_info,iter=100){

  random_proximity <- NULL

  for (i in 1:iter){

    random_node1 <- selectRandomNodes(targets_degree_distribution,graph_info)
    random_node2 <- selectRandomNodes(genes_degree_distribution,graph_info)

    tmp <- computeProximity(random_node1,random_node2,graph_info)

    random_proximity <- c(random_proximity,tmp)

  }

  random_proximity[is.infinite(random_proximity)] <- NaN

  return(random_proximity)

}
```

```
selectRandomNodes <- function(degree_distribution){

  d <- degree_distribution$degree
  freq <- degree_distribution$freq

  random_node <- unlist(lapply(d, function(x){
    sample(node[degree==x],freq[d==x])
  }))

  return(random_node)
}
```

- This function randomly selects two groups of **proteins of the same size** and **degree distribution** as the original lists of disease genes and drug targets in the human interactome
- Then, it computes the **proximity** between the two random lists of nodes
- This procedure is repeated **"iter" times** (default = 100 iterations) to obtain reference random distribution

computeStatistics.R

Step 1

```r
computeStatistics <- function(distribution,observation,plot=F,filename=NULL){

  m <- mean(distribution, na.rm = T)
  sd <- sd(distribution, na.rm = T)

  z <- (observation - m) / sd

  pval = pnorm(z, lower.tail = T) # H1 = be less than expected by chance

  # ######################################

  if(plot){

    par(xpd=F)

    xmin <- min((m-3*sd),observation)
    xmax = max((m+3*sd),observation)

    fun <- function(x) dnorm(x,m,sd)

    pdf(file = filename)

    plot <- curve(fun, xlim = c(xmin,xmax),
                  xlab = "Network proximity",
                  ylab = "Probability density",
                  n = 1000)

    polygon(plot, col = "grey")

    abline(v = observation, lty = 2, lwd = 2, col = "red")

    legend("topright",
           paste("Observation =", format(observation, digits = 3), "\n",
                 "p-value =", format(pval, digits=1)),
           text.col=2,
           text.font = 2,
           bty="o",
           xjust = 0.5,
           yjust = 0.5)

    dev.off()

  }

  return(pval)

}
```

- This function **z-score-normalizes the proximity** calculated between the two original lists of disease genes and drug targets with respect to the reference random distribution
- Subsequently, the **p-value** for the given z statistic is calculated

⚠ **Caveat:** the plot variable is set to **F**, thus the distribution plots are not shown

If the proximity measure is statistically significant (p-value < 0.05), the analyzed list of disease genes and drug targets are **closer** than expected by chance in the human interactome

Main file

Adjusted similarity drug-disease network

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ###################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ###################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ###################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ###################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ###################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ###################################
30
```

30:1 (Untitled) R Script

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

```
>
```

Environment History

Import Dataset List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Delete Rename More

ıe > Documents > Didattica > class > BioinformaticsII > lessons > 6_SaveRUNNER > Code > src

Name	Size	Modified
..		
main.R	868 B	Sep 19, 2020, 11:20 AM
script		

mainEndNetwork.R

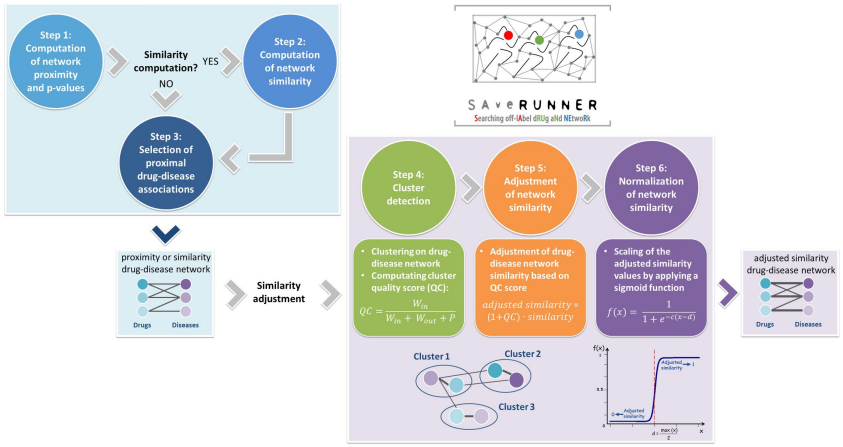
```
mainEndNetwork <- function(){

  ######################################
  # input parameter
  pval_thr <- input_parameter$pval_thr
  interaction <- input_parameter$interaction
  adjust_link <- input_parameter$adjust_link
  new_link <- input_parameter$new_link

  # input files
  filename_out_allPval <- output_file$filename_out_allPval

  # output files
  filename_out_cluster <- output_file$filename_out_cluster
  filename_out_clusterInfo <- output_file$filename_out_clusterInfo
  filename_out <- output_file$filename_out
  ######################################

  drug_disease_net <- read.table(filename_out_allPval, header = T, sep = '\t', check.names = F,
quote = "")
```

similarity drug-disease network

```
  if(interaction == "similarity") drug_disease_net <- computeSimilarity(drug_disease_net)

  drug_disease_net <- selectPval(drug_disease_net,pval_thr,interaction,adjust_link,filename_out)

  if( (adjust_link == T) & interaction == "similarity"){

    adj <- createAdjMatrix(drug_disease_net,interaction)
    node <- row.names(adj)

    cluster <- computeClustering(drug_disease_net,interaction,filename_out_cluster)

    clusterInfo <- computeClusterInfo(node,adj,cluster,filename_out_clusterInfo)

    new_adj <- createNewAdjMatrix(node,adj,clusterInfo,new_link)

    new_adj <- computeNormalization(new_adj)

    drug_disease_net <- createEdgeList(drug_disease_net,node,new_adj,interaction,filename_out)
  }
}
```

adjusted similarity drug-disease network

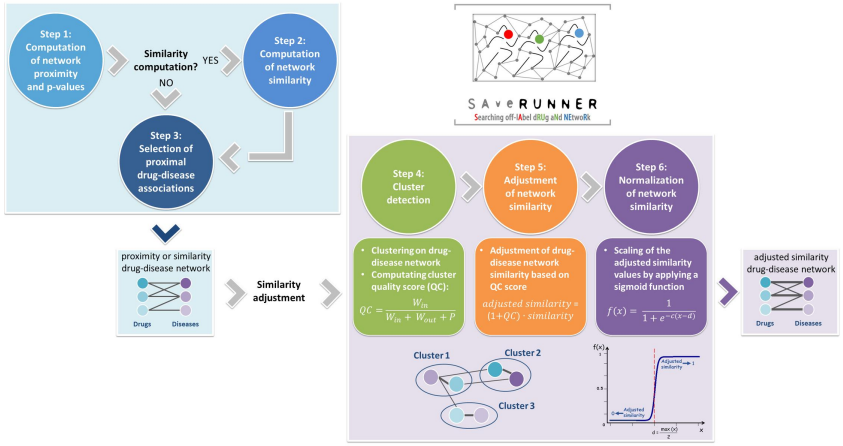
- This function computes the **similarity** and the **adjusted similarity** drug-disease network (Step 2-6) and releases the final network of statistically significant drug-disease predicted associations



mainEndNetwork.R

```
mainEndNetwork <- function(){

  ######################################
  # input parameter
  pval_thr <- input_parameter$pval_thr
  interaction <- input_parameter$interaction
  adjust_link <- input_parameter$adjust_link
  new_link <- input_parameter$new_link

  # input files
  filename_out_allPval <- output_file$filename_out_allPval

  # output files
  filename_out_cluster <- output_file$filename_out_cluster
  filename_out_clusterInfo <- output_file$filename_out_clusterInfo
  filename_out <- output_file$filename_out
  ######################################

  drug_disease_net <- read.table(filename_out_allPval, header = T, sep = '\t', check.names = F,
quote = "")



  drug_disease_net <- selectPval(drug_disease_net,pval_thr,interaction,adjust_link,filename_out)
                                                                                              )

}
```

- In this case this function does NOT compute the **similarity** and the **adjusted similarity** drug-disease network but releases the final network of statistically significant drug-disease predicted associations reporting only the proximity measure.



ComputeSimilarity.R

Step 2

```r
computeSimilarity <- function(drug_disease_net) {
  max_proximity <- max(drug_disease_net$proximity, na.rm = T)
  similarity <- ( max_proximity - drug_disease_net$proximity ) / max_proximity
  drug_disease_net <- data.frame(drug_disease_net, similarity = similarity)
  return(drug_disease_net)
}
```

- This function computes the **similarity measure** between drug module and disease module defined as follows:

$$similarity = \frac{\max(p) - p}{\max(p)}$$

- where p is the network proximity measure
- It assumes values ranging in [0,1]

Null similarity means that the corresponding disease and drug modules are **very distal** in the human interactome (i.e., *p* is maximum); whereas **maximum similarity** means that the corresponding disease and drug modules are **very proximal** in the human interactome (i.e., *p* equal to zero).

selectPval.R

Step 3

```
selectPval <- function(drug_disease_net,pval_thr,interaction,adjust_link
,filename_out){

  ind <- which(drug_disease_net$pval > pval_thr)

  if(length(ind)>0) drug_disease_net <- drug_disease_net[-ind,]

  condition <- (adjust_link == T) & (interaction == "similarity")

  if(!condition){

    write.table(drug_disease_net, filename_out, sep = "\t", row.names = F, col
.names = T, quote = F)

  }

  return(drug_disease_net)

}
```

- This function selects the **statistically significant drug-disease associations** according to the significance level for the p-values (default: p-value ≤ 0.05)
- It returns the **proximity and/or similarity** drug-disease network

disease	drug	proximity	pval	similarity
COVID-19	acrivastine	1.00000000	4.220080e-02	0.8000000
COVID-19	alcaftadine	1.00000000	4.161242e-02	0.8000000
COVID-19	alimemazine	1.00000000	4.484301e-02	0.8000000
COVID-19	alverine	1.00000000	4.578700e-02	0.8000000
COVID-19	amifampridine	1.00000000	3.433878e-02	0.8000000
COVID-19	aminocaproic acid	0.66666667	3.054591e-03	0.8666667
COVID-19	amrinone	1.00000000	4.953049e-03	0.8000000
COVID-19	anisindione	0.00000000	4.947910e-04	1.0000000
COVID-19	antazoline	1.00000000	2.615329e-02	0.8000000
COVID-19	antihemophilic factor human	1.00000000	3.700048e-02	0.8000000
COVID-19	asparagine	1.16666667	1.088828e-02	0.7666667
COVID-19	astemizole	1.00000000	2.194849e-02	0.8000000
COVID-19	avanafil	1.00000000	5.180000e-06	0.8000000
COVID-19	azacitidine	0.00000000	6.440000e-08	1.0000000
COVID-19	azatadine	1.00000000	3.414587e-02	0.8000000
COVID-19	bacitracin	0.50000000	3.304639e-03	0.9000000
COVID-19	bepotastine	1.00000000	1.670887e-02	0.8000000

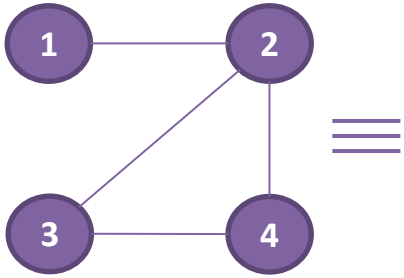
Given a disease A and a drug b, if the **p-value** associated to their distance in the human interactome is smaller of the chosen significance level (default 0.05), the probability that the off-label drug b would be effective for this disease A is greater than expected by chance

createAdjMatrix.R

Step 4

```
createAdjMatrix <- function(network,interaction){

  source <- network$disease
  target <- network$drug
  interaction <- network[,interaction]

  nodi <- unique(c(source,target))

  N <- length(nodi)

  adj <- matrix(0,N,N)
  colnames(adj) <- nodi
  row.names(adj) <- nodi

  for(i in 1:N){

    ind <- which(source %in% nodi[i])

    M <- length(ind)
    if ( M > 0 ){
      for (k in 1:M){
        j <- which(nodi %in% target[ind[k]])
        adj[i,j] = interaction[ind[k]]
        adj[j,i] =  adj[i,j]
      }
    }

  }

  return(adj)

}
```

- Taking as input the similarity drug-disease network, this function builds a **weighted adjacency matrix,** where the weights are the similarity values.



	1	2	3	4
1	0	0.7	0	0
2	0.7	0	0.9	0.5
3	0	0.9	0	0.6
4	0	0.5	0.6	0

$adj_{N\times N}$

computeClustering.R

Step 4

```
computeClustering <- function(network,interaction,filename_out){

  ind <- which(colnames(network) == interaction)
  colnames(network)[ind] <- "weight"

  graph <- graph_from_data_frame(network, directed = F)

  res <- cluster_fast_greedy(graph, merges = TRUE,
                             modularity = TRUE,
                             membership = TRUE,
                             weights = E(graph)$weight)

  cluster <- data.frame(node = res$names, cluster = res$membership)
  cluster <- cluster[order(cluster$cluster),]

  write.table(cluster, filename_out, sep ="\t", row.names = F, col.names = T, quote = F)

  return(cluster)

}
```

- This function performs a **cluster analysis** on the drug-disease network to detect groups of drugs and diseases in such a way that members in the same group (cluster) are more similar to each other than to those in other groups (clusters)
- SAveRUNNER exploits a cluster detection algorithm based on the **greedy optimization of the network modularity**.
- **Modularity** measures the strength of network division into clusters, i.e. networks with high modularity have dense connections between the nodes within clusters but sparse connections between nodes in different clusters.

computeClustering.R

Step 4

```
computeClustering <- function(network,interaction,filename_out){

  ind <- which(colnames(network) == interaction)
  colnames(network)[ind] <- "weight"

  graph <- graph_from_data_frame(network, directed = F)

  res <- cluster_fast_greedy(graph, merges = TRUE,
                             modularity = TRUE,
                             membership = TRUE,
                             weights = E(graph)$weight)

  cluster <- data.frame(node = res$names, cluster = res$membership)
  cluster <- cluster[order(cluster$cluster),]

  write.table(cluster, filename_out, sep ="\t", row.names = F, col.names = T, quote = F)

  return(cluster)

}
```

node	cluster
Severe Acute Respiratory Syndrome	1
5-o-phosphono-alpha-d-ribofuranosyl diphosphate	1
abciximab	1
acenocoumarol	1
adalimumab	1
ademetionine	1
albutrepenonacog alfa	1
aldesleukin	1
alefacept	1
alemtuzumab	1
alogliptin	1
alpha-1-proteinase inhibitor	1
alteplase	1
ambrisentan	1
amlexanox	1
amsacrine	1
anakinra	1
ancrod	1
angiotensin ii	1
anistreplase	1
anti-inhibitor coagulant complex	1
antihemophilic factor, human recombinant	1
antithrombin alfa	1

computeClusterInfo.R

Step 4

```
computeClusterInfo <- function(node,adj,cluster,filename_out){

  SimMat <- matrix(0, nrow = nrow(adj), ncol = ncol(adj))
  SimMat[upper.tri(SimMat)] <- adj[upper.tri(adj)]
  rownames(SimMat) <- node
  colnames(SimMat) <- node

  list <- split(cluster,cluster$cluster)

  cluster_quality <- lapply(names(list), function(x){

    member <- list[[x]]$node

    Win <- SimMat[member,member]
    Wtot <- SimMat[member,]

    class(Win) <- "numeric"
    class(Wtot) <- "numeric"

    w_in <- sum(Win)
    w_out <- sum(Wtot)

    P <- length(member) / length(node)

    QC <- w_in / (w_out + P)

    df <- data.frame(cluster = as.numeric(x),
                     size = length(member),
                     node_density = P,
                     internal_weight = w_in,
                     total_weight = w_out,
                     quality = QC,
                     members = paste(member, collapse = "/"))
  })

  clusterInfo <- rbindlist(cluster_quality)

  write.table(clusterInfo, filename_out, sep ="\t", quote=F, row.names = F, col.names = T)

  return(clusterInfo)

}
```

- This function evaluates the quality of each cluster by computing the following **quality cluster (QC) score**:

$$QC = \frac{W_{in}}{W_{in} + W_{out} + P}$$

where:

- W_{in} is the total weight of edges within the cluster
- W_{out} is the total weight of edges connecting this cluster to the rest of network
- P is a penalty term which considers the node density within the cluster

createNewAdjMatrix.R

Step 5

```
createNewAdjMatrix <- function(node,adj,clusterInfo,new_link){

  list <- lapply(node, function(x){

    ind <- grep(x,clusterInfo$members)

    if(length(ind)>0) df <- data.frame(name=x, cluster=ind, quality=clusterInfo$quality[ind])
  })

  df <- rbindlist(list)
  df <- df[order(df$quality),]

  QC <- matrix(0, nrow(adj), ncol(adj))
  rownames(QC) <- node
  colnames(QC) <- node

  N <- nrow(df)

  for ( i in 1:N){

    ind <- which(df$cluster == df$cluster[i])
    nn <- df$name[ind]

    QC[df$name[i],nn] <- df$quality[i]

  }
  new_adj <- (1 + QC) * adj

  ##################################
  # assign similarity value = QC if two nodes with adj=0 are in the same cluster

  if(new_link){

    list <- lapply(rownames(new_adj), function(i){
      data.frame( t( pmax(new_adj[i,],QC[i,]) ) )
    })

    new_adj <- data.frame(rbindlist(list))
    rownames(new_adj) <- node
    colnames(new_adj) <- node

  }
  ##################################

  return(new_adj)

}
```

- This function computes the **adjusted similarity measure**
- The quality cluster score serves to reward associations between drugs and diseases belonging to the same cluster, based on the assumption that if a drug and a disease group together is more likely that the drug can be effectively repurposed for that disease.
- In this sense, we say that drug and the disease that are members of the same cluster tend to be “more similar” and this translates into the following **adjustment** for the similarity:

$$similarity = (1 + QC) \cdot similarity$$

$$similarity = similarity + QC \cdot similarity$$

⚠ **Caveat:** the adjusted similarity values could be greater than 1

If two nodes fall in **the same cluster their similarity value increases** by a factor proportional to the QC score of the cluster which they belong; **otherwise** if two nodes do not fall in the same cluster QC is set to zero and their **similarity value does not change**.

createNewAdjMatrix.R

Step 5

```
createNewAdjMatrix <- function(node,adj,clusterInfo,new_link){

  list <- lapply(node, function(x){

    ind <- grep(x,clusterInfo$members)

    if(length(ind)>0) df <- data.frame(name=x, cluster=ind, quality=clusterInfo$quality[ind])
  })

  df <- rbindlist(list)
  df <- df[order(df$quality),]

  QC <- matrix(0, nrow(adj), ncol(adj))
  rownames(QC) <- node
  colnames(QC) <- node

  N <- nrow(df)

  for ( i in 1:N){

    ind <- which(df$cluster == df$cluster[i])
    nn <- df$name[ind]

    QC[df$name[i],nn] <- df$quality[i]

  }

  new_adj <- (1 + QC) * adj

  ##################################
  # assign similarity value = QC if two nodes with adj=0 are in the same cluste

  if(new_link){

    list <- lapply(rownames(new_adj), function(i){
      data.frame( t( pmax(new_adj[i,],QC[i,]) ) )
    })

    new_adj <- data.frame(rbindlist(list))
    rownames(new_adj) <- node
    colnames(new_adj) <- node

  }
  ##################################

  return(new_adj)

}
```

- This function computes the **adjusted similarity measure**
- The quality cluster score serves to reward associations between drugs and diseases belonging to the same cluster, based on the assumption that if a drug and a disease group together is more likely that the drug can be effectively repurposed for that disease.
- In this sense, we say that drug and the disease that are members of the same cluster tend to be “more similar” and this translates into the following **adjustment** for the similarity:

$$similarity = (1 + QC) \cdot similarity$$

- If the parameter **new_link** is set to «T», this function assigns also a similarity value equal to QC if two nodes with an adjacency matrix value = 0 (i.e., not connected in the drug-disease network) are in the same cluster

computeNormalization.R

Step 6

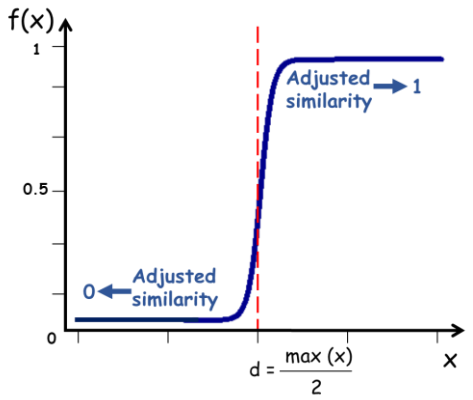
```
computeNormalization <- function(M){

  c = 10
  d = max(M) / 2

  ind <- which(M > 0, arr.ind = T)

  M[ind] <- 1 / ( 1+exp(-c*(M[ind]-d)) )

  return(M)

}
```

- This function implements a **normalization** procedure by applying the following sigmoid function to bound the similarity measure to values that monotonically increase from 0 to 1:

$$f(x) = \frac{1}{1 + e^{-c(x-d)}}$$

where:

- x is the adjusted similarity measure
- d is the sigmoid midpoint (i.e., the value at which the function approaches to 0.5)
- c is the sigmoid steepness.
- SAveRUNNER set $d = \frac{\max(x)}{2}$ and $c = 10$



createEdgeList.R

Step 6

```
createEdgeList <- function(network,node,adj,interaction,filename_out){

  SimMat <- matrix(0, nrow = nrow(adj), ncol = ncol(adj))
  SimMat[upper.tri(SimMat)] <- adj[upper.tri(adj)]

  rownames(SimMat) <- node
  colnames(SimMat) <- node

  ind <- which(SimMat>0, arr.ind = TRUE)

  source <- rownames(SimMat)[ind[,1]]
  target <- colnames(SimMat)[ind[,2]]
  weight <- SimMat[ind]

  df <- data.frame(source,target,weight)
  colnames(df)[3] <- paste0("adjusted_",interaction)

  network_adjusted <- merge(network,df,by.x = c("disease","drug"),by.y = c("source","target"),all=T)

  write.table(network_adjusted, filename_out, sep = "\t", row.names = F, col.names = T, quote = F)

  return(network_adjusted)

}
```

- This function builds the final **adjusted similarity drug-disease network**

disease	drug	proximity	pval	similarity	adjusted_similarity
COVID-19	acrivastine	1.0000000	4.220080e-02	0.8000000	0.9916570
COVID-19	alcaftadine	1.0000000	4.161242e-02	0.8000000	0.9916570
COVID-19	alimemazine	1.0000000	4.484301e-02	0.8000000	0.9916570
COVID-19	alverine	1.0000000	4.578700e-02	0.8000000	0.9916570
COVID-19	amifampridine	1.0000000	3.433878e-02	0.8000000	0.5091856
COVID-19	aminocaproic acid	0.6666667	3.054591e-03	0.8666667	0.9970988
COVID-19	amrinone	1.0000000	4.953049e-03	0.8000000	0.9916570
COVID-19	anisindione	0.0000000	4.947910e-04	1.0000000	0.9996521
COVID-19	antazoline	1.0000000	2.615329e-02	0.8000000	0.9916570
COVID-19	antihemophilic factor human	1.0000000	3.700048e-02	0.8000000	0.9916570
COVID-19	asparagine	1.1666667	1.088828e-02	0.7666667	0.9858957
COVID-19	astemizole	1.0000000	2.194849e-02	0.8000000	0.5091856
COVID-19	avanafil	1.0000000	5.180000e-06	0.8000000	0.5091856
COVID-19	azacitidine	0.0000000	6.440000e-08	1.0000000	0.9996521
COVID-19	azatadine	1.0000000	3.414587e-02	0.8000000	0.9916570
COVID-19	bacitracin	0.5000000	3.304639e-03	0.9000000	0.7382218
COVID-19	bepotastine	1.0000000	1.670887e-02	0.8000000	0.9916570

At the end of this step, SAveRUNNER offers a list of predicted/prioritized associations between drugs and diseases as a **weighted bipartite drug-disease network**, in which one set of nodes corresponds to drugs and the other one corresponds to diseases. A link between a drug and a disease occurs if the corresponding drug targets and disease genes are nearby in the interactome more than expected by chance (p-value $\leq$ 0.05) and the weight of their interaction corresponds to the **adjusted and normalized similarity value.**

Main file

Figure

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ###################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ###################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ###################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ###################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ###################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ###################################
30
```

30:1 (Untitled) R Script

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

```
>
```

Environment History Connections

Import Dataset List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Delete Rename More

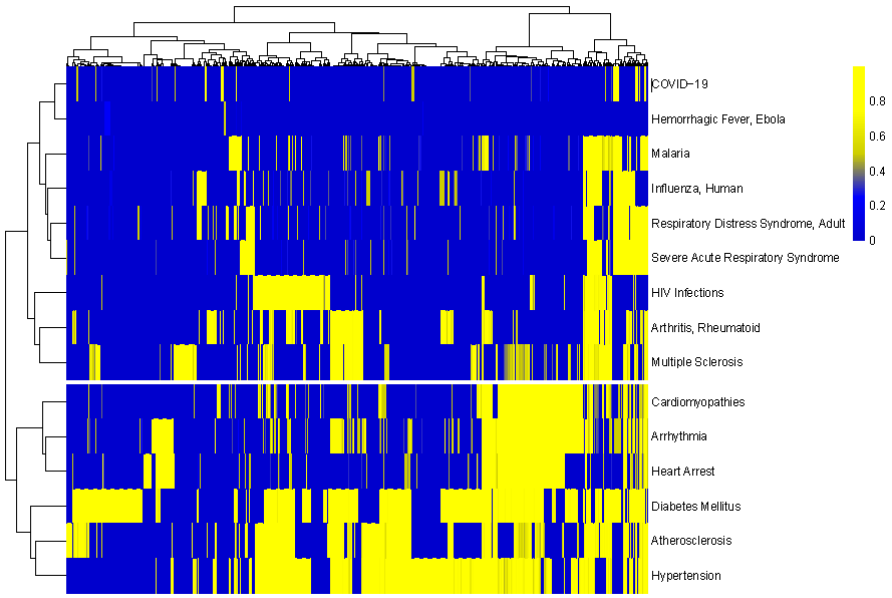
ıe › Documents › Didattica › class › BioinformaticsII › lessons › 6_SaveRUNNER › Code › src

Name	Size	Modified
..		
main.R	868 B	Sep 19, 2020, 11:20 AM
script		

mainFigure.R

```r
mainFigure <- function(){

  ########################################
  # input parameters
  distance <- input_parameter$distance
  pval_thr <- input_parameter$pval_thr

  # input files
  filename_out <- output_file$filename_out

  # output files
  filename_heatmap_DrugDis <- output_file$filename_heatmap_DrugDis

  filename_heatmap_DisDis <- output_file$filename_heatmap_DisDis

  filename_corrplot <- output_file$filename_corrplot
  ########################################

  buildHeatmap(filename_out,filename_heatmap_DrugDis,distance)

  buildHamming(filename_out,filename_heatmap_DisDis,display_numbers = T)

  buildCorrPlot(filename_out,pval_thr,filename_corrplot,distance)

}
```

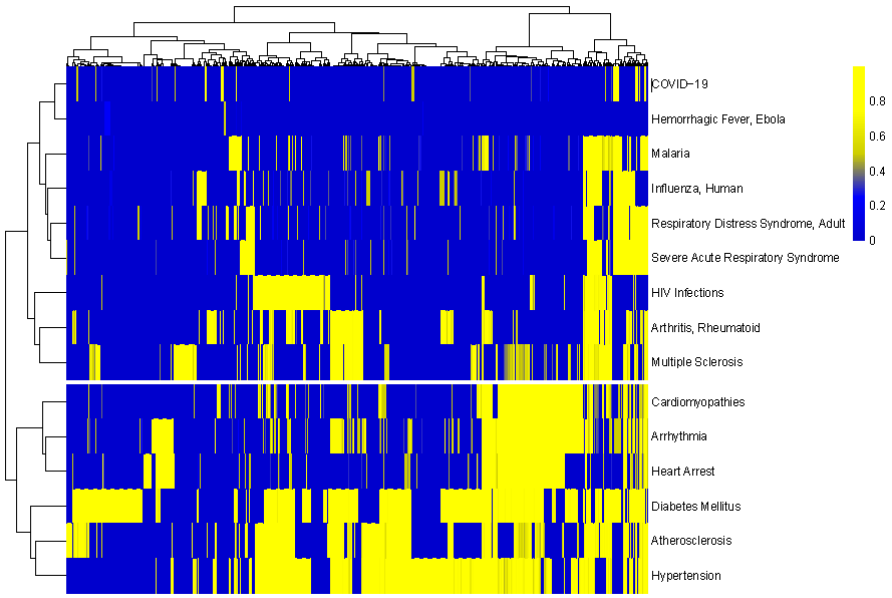
- This function computes the figures of drug-disease network



buildHeatmap.R

```
buildHeatmap <- function(filename_in,filename_out,distance){

  drug_disease_net <- read.table(filename_in, header = T, sep = '\t', check.names = F, quote = "")

  res <- createMatrix(drug_disease_net,"disease","drug",distance)
  mat <- res$matrix

  if( distance == "proximity" ){

    mat[is.na(mat)] <- max(mat,na.rm=T)

    colorBar <- colorRampPalette(colors = c("yellow","yellow1","yellow3","blue","blue3"))(100)

  }else{

    mat[is.na(mat)] <- 0

    colorBar <- colorRampPalette(colors = c("blue3","blue","yellow3","yellow1","yellow"))(100)

  }

  getHeatmapPlot(mat,colorBar,filename_out = filename_out)

}
```

- This function builds the dendrogram and heatmap of the drug-disease network



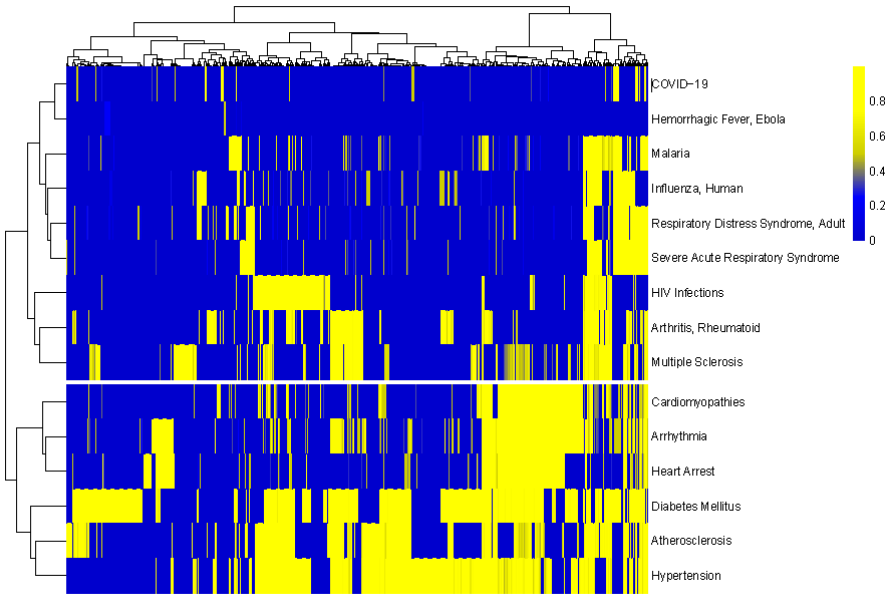
- Rows (diseases) and columns (drugs) are clustered by **Ward's linkage method** as hierarchical clustering algorithm and by using the **Euclidean distance** as distance metric.
- Color refers to adjusted and normalized network similarity between drug targets and disease genes, increasing from blue (less similar) to yellow (more similar)

GetHeatmapPlot.R

```r
getHeatmapPlot <- function(data,colorBar,annotation=NA,annotation_color=NA,filename_out){

  pheatmap(data, scale = "none",
           border_color = NA,
           clustering_distance_rows = "euclidean",
           clustering_distance_cols = "euclidean",
           clustering_method = "ward.D2",
           cluster_cols = T,
           cluster_rows = T,
           cutree_rows = 2,
           annotation_col = annotation,
           annotation_colors = annotation_color,
           color = colorBar,
           show_rownames = T,
           show_colnames = F,
           width = 15, height = 10,
           filename = filename_out,
           fontsize = 10, #treeheight_row = 30,
           cellwidth = 0.3, cellheight = 30
  )

}
```

⚠ **Caveat:** **Ward's method** is a procedure where the criterion for choosing the pair of clusters to merge at each step is based on the optimal value of the Error Sum of Squares (SSE).

- This function builds the dendrogram and heatmap of the drug-disease network

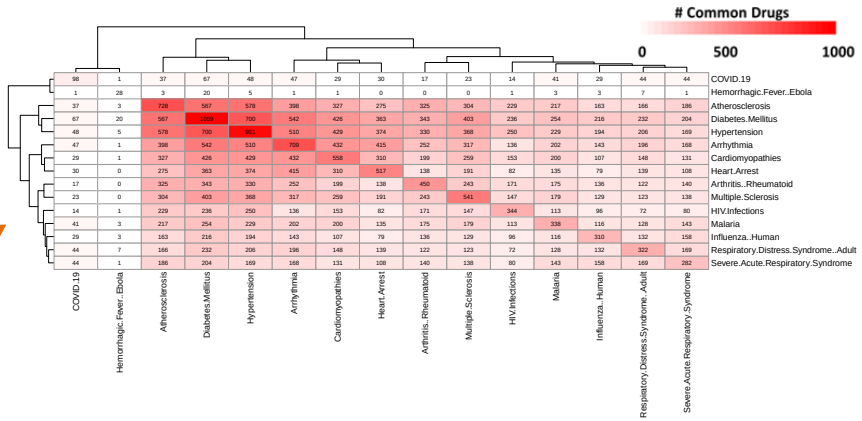


- Rows (diseases) and columns (drugs) are clustered by **Ward's linkage method** as hierarchical clustering algorithm and by using the **Euclidean distance** as distance metric.
- Color refers to adjusted and normalized network similarity between drug targets and disease genes, increasing from blue (less similar) to yellow (more similar)

mainFigure.R

```
mainFigure <- function(){

  ########################################
  # input parameters
  distance <- input_parameter$distance
  pval_thr <- input_parameter$pval_thr

  # input files
  filename_out <- output_file$filename_out

  # output files
  filename_heatmap_DrugDis <- output_file$filename_heatmap_DrugDis

  filename_heatmap_DisDis <- output_file$filename_heatmap_DisDis

  filename_corrplot <- output_file$filename_corrplot
  ########################################

  buildHeatmap(filename_out,filename_heatmap_DrugDis,distance)

  buildHamming(filename_out,filename_heatmap_DisDis,display_numbers = T)

  buildCorrPlot(filename_out,pval_thr,filename_corrplot,distance)

}
```

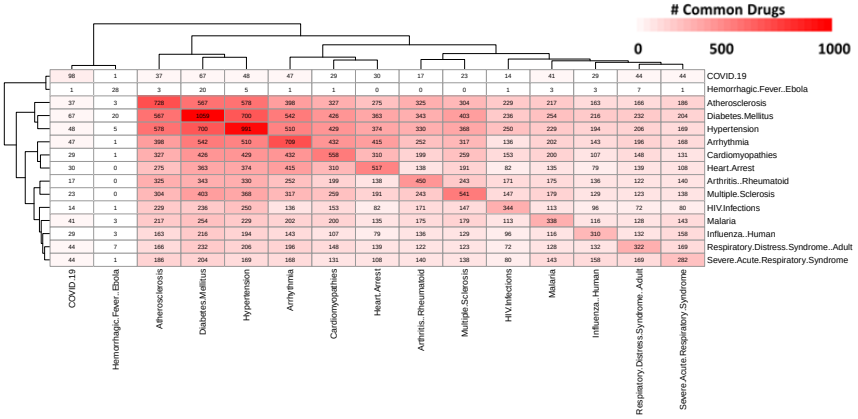
- This function computes the **Hamming distance** among the analyzed diseases pairs



buildHamming.R

```
buildHamming <- function(filename_in,filename_out,display_numbers){

  drug_disease_net <- read.table(filename_in, header = T, sep = '\t', check
.names = F, quote = "")

  f <- drug_disease_net[,"disease"]
  list <- split(drug_disease_net[,"drug"],f)

  output <- lapply(names(list),function(x){

    drug <- list[[x]]
    L <- length(drug)

    mat <- matrix(rep(1,L), L, 1)
    colnames(mat) <- x

    df <- data.frame(drug, mat)

  })

  DataMatrix <- Reduce(function(...) merge(..., by = "drug", all=TRUE),
output)
  DataMatrix[is.na(DataMatrix)] <- 0

  DataMatrix <- DataMatrix[,-1]

  getHammingPlot(DataMatrix,col,filename_out,display_numbers)

}
```

- This function computes the distance matrix based on **Hamming distance** of binary-encoded drugs annotation across diseases (1 = present; 0 = absent)
- This matrix is rendered in a symmetrical heatmap, where the number of common drugs between disease pairs increases from white to red



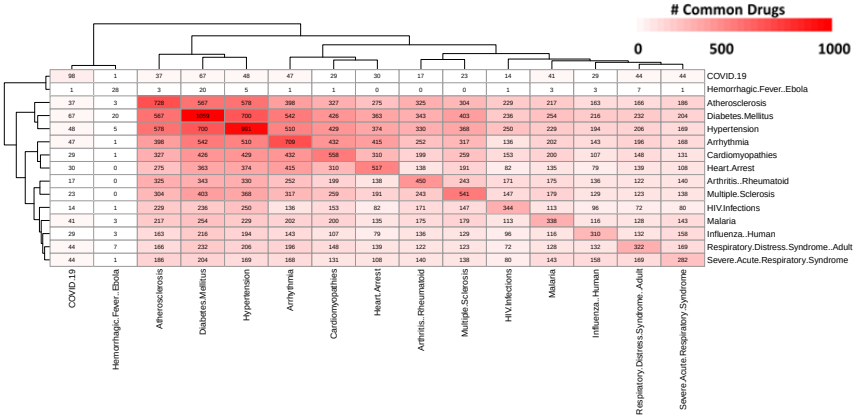
- Each cell reports the number of **shared drugs** among disease pairs in the drug-disease network.
- Rows (disease) and columns (disease) are clustered by a **complete linkage hierarchical clustering** algorithm and by using **Euclidean distance** as distance metrics

Rows and columns matrix refers to the analyzed diseases of the drug-disease network and on the matrix diagonal you will find the number of predicted drugs of each disease.

getHammingPlot.R

```
getHammingPlot <- function(data,col,filename_out,display_numbers){

  rho <- apply(data, 2, function(x){

    apply(data,2,function(y){

      ind<-which(x!=0 | y!=0)
      x<-x[ind]
      y<-y[ind]

      d <- hamming.distance(x,y)/length(x)
      rho <- (1 - d)*length(x)

    })
  })

  res <- pheatmap(rho,
                  clustering_distance_rows = "euclidean",
                  clustering_distance_cols = "euclidean",
                  clustering_method = "complete",
                  cluster_cols = TRUE,
                  cluster_rows = TRUE,
                  display_numbers = display_numbers,
                  number_format = "%.0f",
                  number_color = "black",
                  fontsize_number = 7,
                  #cutree_rows = 2,
                  #cutree_cols = 2,
                  show_colnames = TRUE,
                  color = colorRampPalette(c("white", "red"))(30),
                  filename = filename_out,
                  cellwidth=50, cellheight = 15, angle_col = 90)

}
```

- This function computes the distance matrix based on **Hamming distance** of binary-encoded drugs annotation across diseases (1 = present; 0 = absent)
- This matrix is rendered in a symmetrical heatmap, where the number of common drugs between disease pairs increases from white to red



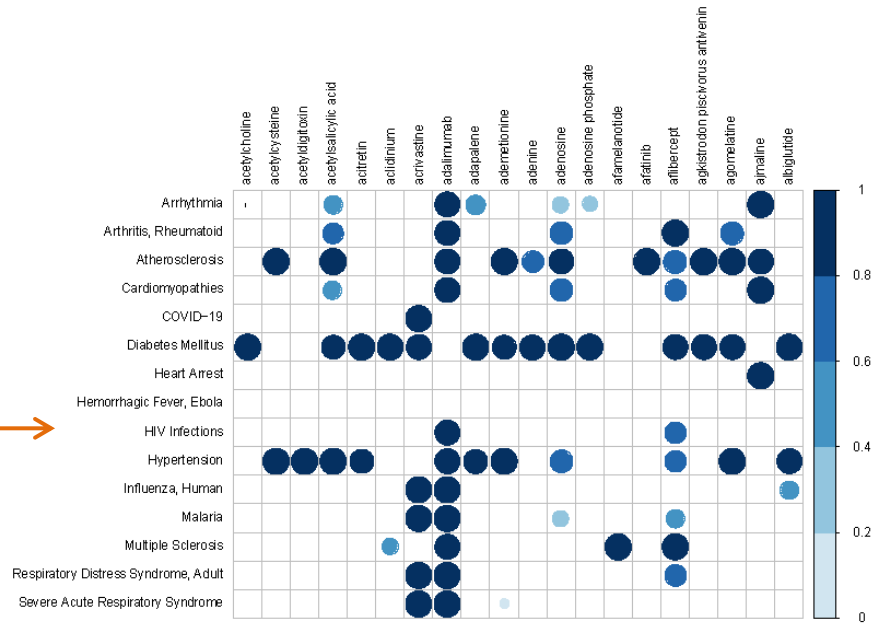
- Each cell reports the number of **shared drugs** among disease pairs in the drug-disease network.
- Rows (disease) and columns (disease) are clustered by a **complete linkage** hierarchical clustering algorithm and by using **Euclidean distance** as distance metrics

Rows and columns matrix refer to the analyzed diseases of the drug-disease network and on the matrix diagonal you will find the number of predicted drugs of each disease.

mainFigure.R

```r
mainFigure <- function(){

  ########################################
  # input parameters
  distance <- input_parameter$distance
  pval_thr <- input_parameter$pval_thr

  # input files
  filename_out <- output_file$filename_out

  # output files
  filename_heatmap_DrugDis <- output_file$filename_heatmap_DrugDis

  filename_heatmap_DisDis <- output_file$filename_heatmap_DisDis

  filename_corrplot <- output_file$filename_corrplot
  ########################################

  buildHeatmap(filename_out,filename_heatmap_DrugDis,distance)

  buildHamming(filename_out,filename_heatmap_DisDis,display_numbers = T)

  buildCorrPlot(filename_out,pval_thr,filename_corrplot,distance)

}
```

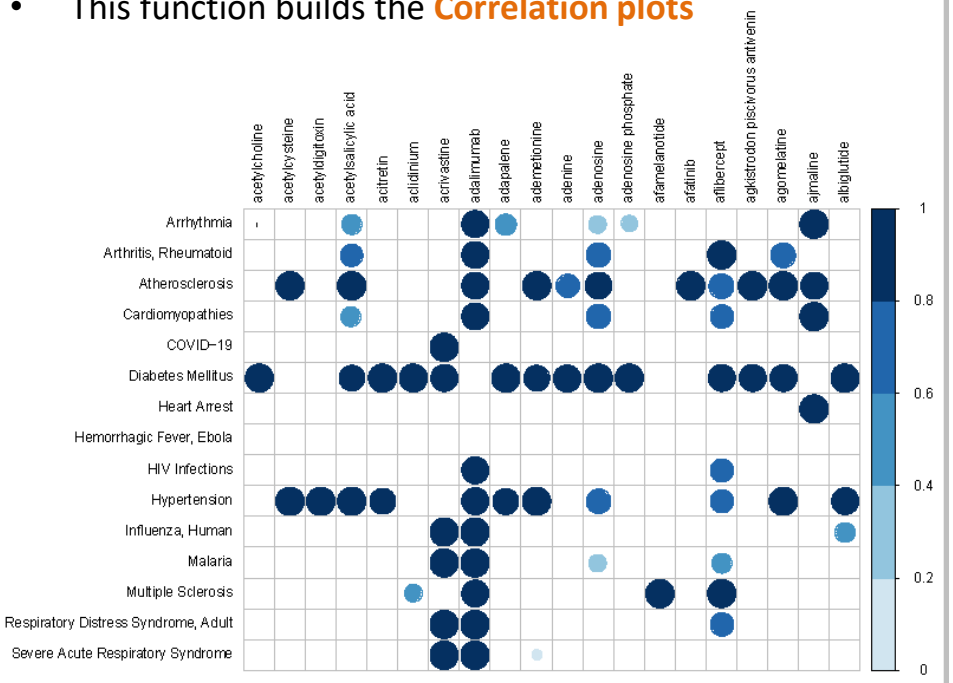
- This function computes the figures of drug-disease network



buildCorrPlot.R

```
buildCorrPlot <- function(filename_in,pval_thr,filename_out,distance){

  drug_disease_net <- read.table(filename_in, header = T, sep = '\t', check.names = F, quote = "")

  res <- createMatrix(drug_disease_net,"disease","drug",distance)
  mat <- res$matrix
  pval <- res$pval

  if( distance == "proximity" ){

    mat[is.na(mat)] <- max(mat,na.rm=T)

  }else{

    mat[is.na(mat)] <- 0

  }

  pval[is.na(pval)] <- 1

  mat <- mat[order(rownames(mat)),]
  pval <- pval[order(rownames(pval)),]

  list <- createList(drug_disease_net)

  getCorrPlot(mat,pval,list,pval_thr,filename_out)

}
```

- This function builds the **Correlation plots**

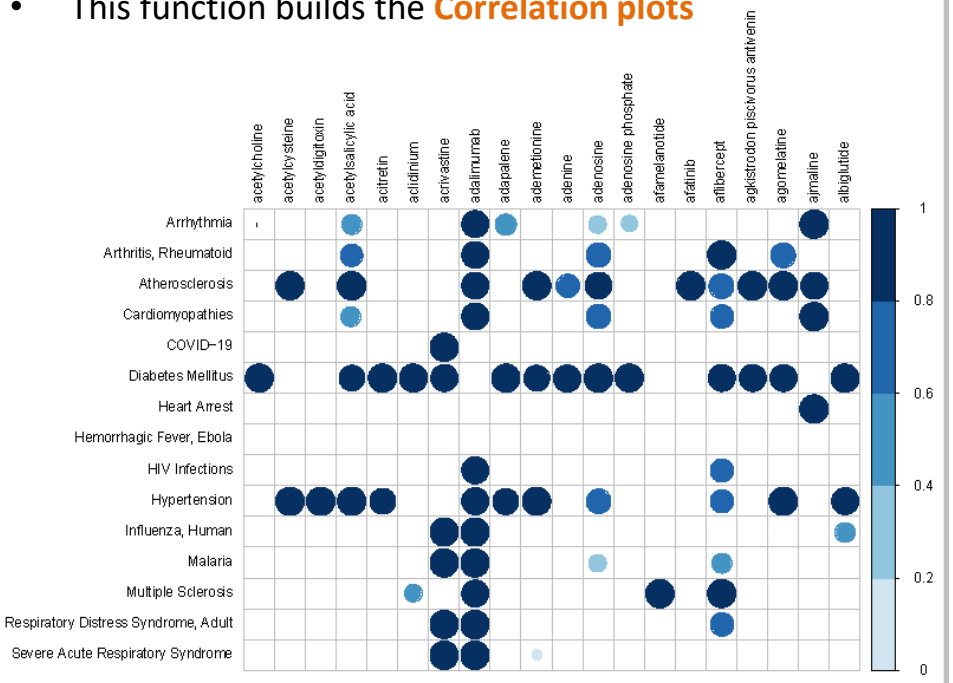


- Rows refer to the analyzed **diseases**
- Columns refer to predicted **drugs**
- Circles are scaled and colored according to the **adjusted similarity measure**, increasing from light blue (low similarity) to dark blue (high similarity)
- Plots are divided in N groups by drugs alphabetic order each one of at most 20 drugs

getCorrPlot.R

```
getCorrPlot <- function(data,pval,list,pval_thr,filename_out){

  lapply(names(list),function(x){

    y <- list[[x]]
    y <- y[order(y$drug),]

    tl.col = "black"

    y <- data.frame(y)
    colnames(y) = "drug"

    if(nrow(y)==1){

      data <- as.matrix(data[,y[,"drug"]])
      pval <- as.matrix(pval[,y[,"drug"]])

      colnames(data) <- y["drug"]
      colnames(pval) <- y["drug"]

    }else{

      data <- data[,y[,"drug"]]
      pval <- pval[,y[,"drug"]]

    }

   x <- gsub(pattern = " ", x = x, replacement = "_")

   pdf(file = paste0(filename_out,"_",x,".pdf"))

   corrplot(data, is.corr = F,
            method = "circle",
            p.mat = pval,
            sig.level = pval_thr,
            tl.cex = 0.6,
            tl.col = tl.col,
            insig = "blank",
            #insig = "pch", pch = "·", pch.col = "white", pch.cex = 1,
            na.label = "square", na.label.col = "white",
            cl.cex = 0.6,
            col = brewer.pal(n = 10, name = "RdBu"))

   dev.off()

  })
}
```

- This function builds the **Correlation plots**



- Rows refer to the analyzed **diseases**
- Columns refer to predicted **drugs**
- Circles are scaled and colored according to the **adjusted similarity measure**, increasing from light blue (low similarity) to dark blue (high similarity)
- Plots are divided in N groups by drugs alphabetic order each one of at most 20 drugs

Main file

Subnetwork

main.R

```
1  rm(list=ls())
2
3  options(stringsAsFactors = F)
4
5  setwd("~/SaveRUNNER/Code/")
6  ###################################
7  source("src/script/getLibrary.R")
8  source("src/script/getSource.R")
9  ###################################
10 getLibrary()
11 getSource()
12 input_parameter <- config()
13 input_file <- inputFiles()
14 output_file <- networkFiles()
15 ###################################
16 # compute drug-disease network
17 destfile = output_file$filename_out_allPval
18 if ( !file.exists(destfile) ) mainStartNetwork()
19
20 # select significative drug-disease association and (or not) adjust them
21 mainEndNetwork()
22 ###################################
23 # make figure
24
25 if( length(input_parameter$diseases) > 1) mainFigure()
26 ###################################
27 # create disease specific subnetwork
28 if( !is.null(input_parameter$sel_disease) ) mainSubnetwork()
29 ###################################
30
```

30:1 (Untitled) R Script

Console ~/Documents/Didattica/class/BioinformaticsII/lessons/6_SaveRUNNER/Code/

```
>
```

Environment History Connections

Import Dataset List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Delete Rename More

ıe › Documents › Didattica › class › BioinformaticsII › lessons › 6_SaveRUNNER › Code › src

Name	Size	Modified
..		
main.R	868 B	Sep 19, 2020, 11:20 AM
script		

mainSubNetwork.R

```
mainSubnetwork <- function(){

  ########################################
  # input parameters
  distance <- input_parameter$distance
  drug <- input_parameter$sel_drug
  disease <- input_parameter$sel_disease

  # input files
  filename_out <- output_file$filename_out

  # output files
  filename_out_DrugDis <- output_file$filename_out_DrugDis
  filename_out_DisDis <- output_file$filename_out_DisDis
  filename_selDisease_specDrug <- output_file$filename_selDisease_specDrug
  filename_DrugDistribution <- output_file$filename_DrugDistribution
  ########################################

  drug_disease_subnet <- getDrugDiseaseSubnet(filename_out,disease,filename_out_DrugDis)
  disease_disease_subnet <- getDiseaseDiseaseSubnet(filename_out,drug_disease_subnet,
                                                    disease,filename_out_DisDis)
  getSpecificDrug(disease_disease_subnet,disease,filename_selDisease_specDrug)

  if( !is.null(drug) ){
    plotDrugDistribution(filename_out,drug,disease,filename_DrugDistribution)
  }

}
```

- This function computes the **subnetwork** of the selected disease

getDrugDiseaseSubnet.R

```r
getDrugDiseaseSubnet <- function(filename_in,disease,filename_out){

  drug_disease_net <- read.table(filename_in, header = T, sep = '\t', check.names = F, quote = "")

  ind <- which(drug_disease_net$disease == disease)

  drug_disease_subnet <- unique(drug_disease_net[ind,])

  write.table(drug_disease_subnet, filename_out, sep = "\t", row.names = F, col.names = T, quote = F)

  return(drug_disease_subnet)

}
```

- This function computes the **drug-disease** subnetwork of the selected disease

Selected disease **Candidate repurposable drugs**

disease	drug	proximity	pval	similarity	adjusted_similarity
COVID-19	acrivastine	1.0000000	4.220080e-02	0.8000000	0.9916570
COVID-19	alcaftadine	1.0000000	4.161242e-02	0.8000000	0.9916570
COVID-19	alimemazine	1.0000000	4.484301e-02	0.8000000	0.9916570
COVID-19	alverine	1.0000000	4.578700e-02	0.8000000	0.9916570
COVID-19	amifampridine	1.0000000	3.433878e-02	0.8000000	0.5091856
COVID-19	aminocaproic acid	0.6666667	3.054591e-03	0.8666667	0.9970988
COVID-19	amrinone	1.0000000	4.953049e-03	0.8000000	0.9916570
COVID-19	anisindione	0.0000000	4.947910e-04	1.0000000	0.9996521
COVID-19	antazoline	1.0000000	2.615329e-02	0.8000000	0.9916570
COVID-19	antihemophilic factor human	1.0000000	3.700048e-02	0.8000000	0.9916570
COVID-19	asparagine	1.1666667	1.088828e-02	0.7666667	0.9858957
COVID-19	astemizole	1.0000000	2.194849e-02	0.8000000	0.5091856
COVID-19	avanafil	1.0000000	5.180000e-06	0.8000000	0.5091856
COVID-19	azacitidine	0.0000000	6.440000e-08	1.0000000	0.9996521
COVID-19	azatadine	1.0000000	3.414587e-02	0.8000000	0.9916570
COVID-19	bacitracin	0.5000000	3.304639e-03	0.9000000	0.7382218
COVID-19	bepotastine	1.0000000	1.670887e-02	0.8000000	0.9916570
COVID-19	bilastine	1.0000000	2.313395e-02	0.8000000	0.9916570
COVID-19	bromodiphenhydramine	1.0000000	2.325987e-03	0.8000000	0.9916570
COVID-19	caffeine	1.5000000	3.491870e-03	0.7000000	0.2762273
COVID-19	camphor	1.5000000	3.500855e-02	0.7000000	0.9602774
COVID-19	cetirizine	1.0000000	2.663977e-02	0.8000000	0.9916570
COVID-19	chlorcyclizine	1.0000000	2.900589e-02	0.8000000	0.9916570

getDiseaseDiseaseSubnet.R

```r
getDiseaseDiseaseSubnet <- function(filename_in,drug_disease_subnet,disease,filename_out){

  drug_disease_net <- read.table(filename_in, header = T, sep = '\t', check.names = F, quote = "")

  disease_disease_subnet <- merge(drug_disease_subnet, drug_disease_net, by ="drug")

  write.table(disease_disease_subnet, filename_out, sep = "\t", row.names = F, col.names = T, quote = F
)

  return(disease_disease_subnet)

}
```

- This function computes the **disease-disease** subnetwork of the selected disease
- This network shows the repurposable drugs for the selected disease (e.g., COVID 19) shared with other analysed diseases

Candidate repurposable drugs of selected disease

Selected disease **Other network diseases**

drug	disease.x	proximity.	pval.x	similarity.x	adjusted_similarity.x	disease.y	proximity.	pval.y	similarity.y	adjusted_similarity.y
acrivastine	COVID-19	1	0.042201	0.8	0.991656981	Influenza, H	1	9.41E-10	0.8	0.991657
acrivastine	COVID-19	1	0.042201	0.8	0.991656981	Severe Acute	1	0.00217	0.8	0.991657
acrivastine	COVID-19	1	0.042201	0.8	0.991656981	COVID-19	1	0.042201	0.8	0.991657
acrivastine	COVID-19	1	0.042201	0.8	0.991656981	Malaria	1	0.003467	0.8	0.991657
acrivastine	COVID-19	1	0.042201	0.8	0.991656981	Respiratory	1	0.000161	0.8	0.991657
acrivastine	COVID-19	1	0.042201	0.8	0.991656981	Diabetes Me	0	0.037547	1	0.884602
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Influenza, H	1	0.000161	0.8	0.991657
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Cardiomyop	1	0.0305	0.8	0.509186
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Arrhythmia	1	0.010876	0.8	0.509186
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Severe Acute	1	4.84E-05	0.8	0.991657
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Respiratory	1	4.48E-06	0.8	0.991657
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	COVID-19	1	0.041612	0.8	0.991657
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Malaria	1	0.016317	0.8	0.991657
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Diabetes Me	0	0.038102	1	0.884602
alimemazine	COVID-19	1	0.044843	0.8	0.991656981	Malaria	1	0.034176	0.8	0.991657
alimemazine	COVID-19	1	0.044843	0.8	0.991656981	Severe Acute	1	0.008659	0.8	0.991657
alimemazine	COVID-19	1	0.044843	0.8	0.991656981	COVID-19	1	0.044843	0.8	0.991657
alimemazine	COVID-19	1	0.044843	0.8	0.991656981	Arrhythmia	1	0.022677	0.8	0.509186
alimemazine	COVID-19	1	0.044843	0.8	0.991656981	Respiratory	1	0.000402	0.8	0.991657

getDiseaseDiseaseSubnet.R

```r
getDiseaseDiseaseSubnet <- function(filename_in,drug_disease_subnet,disease,filename_out){

  drug_disease_net <- read.table(filename_in, header = T, sep = '\t', check.names = F, quote = "")

  disease_disease_subnet <- merge(drug_disease_subnet, drug_disease_net, by ="drug")

  write.table(disease_disease_subnet, filename_out, sep = "\t", row.names = F, col.names = T, quote = F
)

  return(disease_disease_subnet)

}
```

Shared drugs between COVID-19 and SARS

drug	disease	proxim	pval.x	similarity.x	adjusted_similarity	disease.y
acrivastine	COVID-19	1	0.042201	0.8	0.991656981	Severe Acute Respiratory Syndrome
alcaftadine	COVID-19	1	0.041612	0.8	0.991656981	Severe Acute Respiratory Syndrome
alimemazine	COVID-19	1	0.044843	0.8	0.991656981	Severe Acute Respiratory Syndrome
aminocaproic acic	COVID-19	0.666667	0.003055	0.86666667	0.997098782	Severe Acute Respiratory Syndrome
amrinone	COVID-19	1	0.004953	0.8	0.991656981	Severe Acute Respiratory Syndrome
anisindione	COVID-19	0	0.000495	1	0.999652102	Severe Acute Respiratory Syndrome
antazoline	COVID-19	1	0.026153	0.8	0.991656981	Severe Acute Respiratory Syndrome
antihemophilic fa	COVID-19	1	0.037	0.8	0.991656981	Severe Acute Respiratory Syndrome
azatadine	COVID-19	1	0.034146	0.8	0.991656981	Severe Acute Respiratory Syndrome
bepotastine	COVID-19	1	0.016709	0.8	0.991656981	Severe Acute Respiratory Syndrome
bilastine	COVID-19	1	0.023134	0.8	0.991656981	Severe Acute Respiratory Syndrome
bromodiphenhyd	COVID-19	1	0.002326	0.8	0.991656981	Severe Acute Respiratory Syndrome
cetirizine	COVID-19	1	0.02664	0.8	0.991656981	Severe Acute Respiratory Syndrome
chlorcyclizine	COVID-19	1	0.029006	0.8	0.991656981	Severe Acute Respiratory Syndrome
clofedanol	COVID-19	1	0.026153	0.8	0.991656981	Severe Acute Respiratory Syndrome
coagulation factor	COVID-19	1.142857	0.036754	0.77142857	0.43807759	Severe Acute Respiratory Syndrome
coccidioides immi	COVID-19	1	0.010547	0.8	0.991656981	Severe Acute Respiratory Syndrome
desloratadine	COVID-19	1	0.046397	0.8	0.991656981	Severe Acute Respiratory Syndrome
dexbrompheniran	COVID-19	1	0.040702	0.8	0.991656981	Severe Acute Respiratory Syndrome

getSpecificDrug.R

```
getSpecificDrug <- function(network,disease,filename_out){

  l <- split(network$disease.y,network$drug)

  list <- suppressWarnings(lapply(l,function(x){

    if(length(x)==1 & x == disease) return(x)

  }))

  specific_drug <- names(unlist(list))

  if(length(specific_drug) > 0){

    write.table(specific_drug, filename_out, sep = "\t", row
.names = F, col.names = F, quote = F)

  }

}
```

- This function computes the drugs that are **specific** of the selected disease (i.e., those drugs that are not predicted by SAveRUNNER as repurposable for the other analyzed diseases)

COVID-19 specific drugs

↓

	A
1	camphor
2	dacarbazine
3	nelarabine
4	threonine

plotDrugDistribution.R

```
plotDrugDistribution <- function(filename_in,drug,disease,filename_out){

  drug_disease_net <- read.table(filename_in, header = T, sep = '\t',
check.names = F, quote = "")

  ind_drug <- which(drug_disease_net$drug == drug & drug_disease_net$dise
ase == disease)
  observation <- drug_disease_net[ind_drug,"proximity"]

  ind_disease <- which(drug_disease_net$disease == disease)
  distribution <- drug_disease_net[ind_disease,"proximity"]

  computeStatistics(distribution,observation,plot=T,filename_out)

  return()

}
```

- This function z-score-normalizes the proximity of the **selected drug** with respect to the proximity measure distribution of all the candidate repurposable drugs for the selected disease and plots the distribution.