Random Walk with Restart on Multiplex and Heterogeneous Network

Alberto Valdeolivas Urbelz[*1, 2]

[1]MMG, Marseille Medical Genetics U 1251, Faculte de Medecine, France
[2]ProGeLife, 8 Rue Sainte Barbe 13001, Marseille, France

[*]alvaldeolivas@gmail.com

May 19, 2021

Abstract

This vignette describes how to use the *RandomWalkRestartMH* package to run Random Walk with Restart algorithms on monoplex, multiplex, heterogeneous and multiplex-heterogeneous networks. It is based on the work we presented on the following article:

https://www.biorxiv.org/content/early/2017/08/30/134734

Package

RandomWalkRestartMH 1.12.0

Report issues to alvaldeolivas@gmail.com

Contents

1 Introduction 3

2 Installation of the *RandomWalkRestartMH* package 3

3 A Detailed Workflow 4

3.1 Random Walk with Restart on a Monoplex Network 4

3.2 Random Walk with Restart on a Heterogeneous Network 6

3.3 Random Walk with Restart on a Multiplex Network 9

3.4 Random Walk with Restart on a Multiplex-Heterogeneous Network . 11

A Session info 17

1 Introduction

RandomWalkRestartMH (Random Walk with Restart on Multiplex and Heterogeneous Networks) is an R package built to provide easy access to the Random Walk with Restart (RWR) algorithm on different types of networks: i) Monoplex networks, ii) Multiplex networks, iii) Heterogeneous networks and iv) Multiplex-Heterogeneous networks. It is based on the work we presented in the article: https://www.biorxiv.org/content/early/2017/08/30/134734.

RWR simulates an imaginary particle that starts on a seed(s) node(s) and follows randomly the edges of a network. At each step, there is a restart probability, r, meaning that the particle can come back to the seed(s) [1]. This imaginary particle can explore the following types of networks:

- A monoplex or single network, which contains solely nodes of the same nature. In addition, all the edges belong to the same category.
- A multiplex network, defined as a collection of monoplex networks considered as layers of the multiplex network. In a multiplex network, the different layers share the same set of nodes, but the edges represent relationships of different nature [2]. In this case, the RWR particle can jump from one node to its counterparts on different layers.
- A heterogeneous network, which is composed of two monoplex networks containing nodes of different nature. These different kind of nodes can be connected thanks to bipartite edges, allowing the RWR particle to jump between the two networks.
- A multiplex and heterogeneous network, which is built by linking the nodes in every layer of a multiplex network to nodes of different nature thanks to bipartite edges. The RWR particle can now explore the full multiplex-heterogeneous network.

The user can introduce up to six single networks (monoplex networks) to create a multiplex network. The multiplex network can also be integrated, thanks to bipartite relationships, with a network containing nodes of different nature. Proceeding this way, a network both multiplex and heterogeneous will be generated.

Please note that this first version of the package deals only with undirected and unweighted networks. New functionalities will be included in future updated versions of *RandomWalkRestartMH*.

2 Installation of the *RandomWalkRestartMH* package

First of all, you need a current version of *R* (www.r-project.org). *RandomWalkRestartMH* is a freely available package deposited on http://bioconductor.org/. You can install it by running the following commands on an *R* console:

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install("RandomWalkRestartMH")
```

3 A Detailed Workflow

In the following paragraphs, we describe how to use the *RandomWalkRestartMH* package to perform RWR on different types of biological networks. Concretely, we use a protein-protein interaction (PPI) network, a pathway network, a disease-disease similarity network and combinations thereof. These networks are obtained as detailed in [3]. The PPI and the Pathway network were reduced by only considering genes/proteins expressed in the adipose tissue, in order to reduce the computation time of this vignette.

The goal in the example presented here is, as described in [3], to find candidate genes potentially associated with diseases by a guilt-by-association approach. This is based on the fact that genes/proteins with similar functions or similar phenotypes tend to lie closer in biological networks. Therefore, the larger the RWR score of a gene, the more likely it is to be functionally related with the seeds.

We focus on a real biological example: the SHORT syndrome (MIM code: 269880) and its causative gene *PIK3R1* as described in [3]. We will see throughout the following paragraphs how the RWR results evolve due to the the integration and exploration of additional networks.

3.1 Random Walk with Restart on a Monoplex Network

RWR has usually been applied within the framework of single PPI networks in bioinformatics[4]. A gene or a set of genes, so-called seed(s), known to be implicated in a concrete function or in a specific disease, are chosen as the starting point(s) of the algorithm. The RWR particle explores the neighbourhood of the seeds and the algorithm computes a score for all the nodes of the network. The larger it is the score of a node, the closer it is to the seed(s).

Let us generate an object of the class *Multiplex*, even if it is a monoplex network, with our PPI network.

```
> library(RandomWalkRestartMH)
> library(igraph)
> data(PPI_Network) # We load the PPI_Network
> ## We create a Multiplex object composed of 1 layer (It's a Monoplex Network)
> ## and we display how it looks like
> PPI_MultiplexObject <- create.multiplex(PPI_Network,Layers_Name=c("PPI"))
> PPI_MultiplexObject

Number of Layers:
[1] 1

Number of Nodes:
[1] 4317

IGRAPH b72da1b UN-- 4317 18062 --
+ attr: name (v/c), type (e/c)
+ edges from b72da1b (vertex names):
 [1] AAMP   --VPS52     AAMP   --BHLHE40   AAMP   --GABARAPL2 AAMP   --MAP1LC3B
 [5] VPS52  --TXN2      VPS52  --DDX6      VPS52  --MFAP1     VPS52  --PRKAA1
 [9] VPS52  --LMO4      VPS52  --STX11     VPS52  --KANK2     VPS52  --PPP1R18
[13] VPS52  --TXLNA     VPS52  --KIAA1217 VPS52  --VPS28     VPS52  --ATP6V1D
[17] VPS52  --TPM3      VPS52  --KIF5B     VPS52  --NOP2      VPS52  --RNF41
[21] VPS52  --WTAP      VPS52  --MAPK3     VPS52  --ZMAT2     VPS52  --VPS51
```

```
[25] BHLHE40--AES       BHLHE40--PRKAA1    BHLHE40--CCNK      BHLHE40--RBPMS
[29] BHLHE40--COX5B     BHLHE40--UBE2I     BHLHE40--MAGED1    BHLHE40--PLEKHB2
+ ... omitted several edges
```

To apply the RWR on a monoplex network, we need to compute the adjacency matrix of the network and normalize it by column [4], as follows:

```
> AdjMatrix_PPI <- compute.adjacency.matrix(PPI_MultiplexObject)
> AdjMatrixNorm_PPI <- normalize.multiplex.adjacency(AdjMatrix_PPI)
```

Then, we need to define the seed(s) before running the RWR algorithm on this PPI network. As commented above, we are focusing on the example of the SHORT syndrome. Therefore, we take the *PIK3R1* gene as seed, and we execute RWR.

```
> SeedGene <- c("PIK3R1")
> ## We launch the algorithm with the default parameters (See details on manual)
> RWR_PPI_Results <- Random.Walk.Restart.Multiplex(AdjMatrixNorm_PPI,
+                         PPI_MultiplexObject,SeedGene)
> # We display the results
> RWR_PPI_Results

Top 10 ranked Nodes:
   NodeNames       Score
1       GRB2 0.006845881
2       EGFR 0.006169129
3        CRK 0.005674261
4       ABL1 0.005617041
5        FYN 0.005611086
6      CDC42 0.005594680
7       SHC1 0.005577900
8       CRKL 0.005509182
9    KHDRBS1 0.005443541
10     TYRO3 0.005441887

Seed Nodes used:
[1] "PIK3R1"
```

Finally, we can create a network (an *igraph* object) with the top scored genes. Visualize the top results within their interaction network is always a good idea in order to prioritize genes, since we can have a global view of all the potential candidates. The results are presented in Figure 1.

```
> ## In this case we selected to induce a network with the Top 15 genes.
> TopResults_PPI <-
+     create.multiplexNetwork.topResults(RWR_PPI_Results,PPI_MultiplexObject,
+                                        k=15)
```

```
> par(mar=c(0.1,0.1,0.1,0.1))
> plot(TopResults_PPI, vertex.label.color="black",vertex.frame.color="#ffffff",
+     vertex.size= 20, edge.curved=.2,
+     vertex.color = ifelse(igraph::V(TopResults_PPI)$name == "PIK3R1","yellow",
+     "#00CCFF"), edge.color="blue",edge.width=0.8)
```

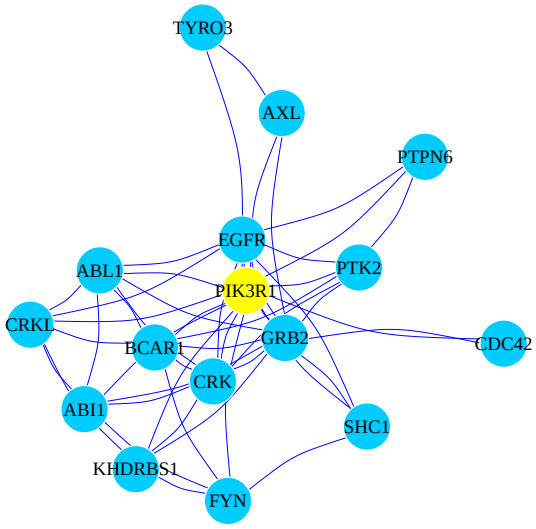

Figure 1: RWR on a monoplex PPI Network
Network representation of the top 15 ranked genes when the RWR algorithm is executed using the *PIK3R1* gene as seed (yellow node). Blue edges represent PPI interactions.

3.2 Random Walk with Restart on a Heterogeneous Network

A RWR on a heterogeneous (RWR-H) biological network was described by [5]. They connected a PPI network with a disease-disease similarity network using known gene-disease associations. In this case, genes and/or diseases can be used as seed nodes for the algorithm. In the following example, we also use a heterogeneous network integrating a PPI and a disease-disease similarity network. However, the procedure to obtain these networks is different to the one proposed in [5], and the details are described in our article [3].

To generate a PPI-disease heterogeneous network object, we load the disease-disease network, and combine it with our previously defined *Multiplex* object containing the PPI network, thanks to the gene-diseases associations obtained from OMIM [6]. A *MultiplexHet* object will be created, even if we are dealing with a monoplex-heterogeneous network.

```
> data(Disease_Network) # We load our disease Network
> ## We load a data frame containing the gene-disease associations.
> ## See ?create.multiplexHet for details about its format
> data(GeneDiseaseRelations)
> ## We keep gene-diseases associations where genes are present in the PPI
> ## network
> GeneDiseaseRelations_PPI <-
+     GeneDiseaseRelations[which(GeneDiseaseRelations$hgnc_symbol %in%
+     PPI_MultiplexObject$Pool_of_Nodes),]
> ## We create the MultiplexHet object.
> PPI_Disease_Net <- create.multiplexHet(PPI_MultiplexObject,
+     Disease_Network, GeneDiseaseRelations_PPI, c("Disease"))
> ## The results look like that
> PPI_Disease_Net

Number of Layers:
```

```
[1] 1

Number of Nodes Multiplex:
[1] 4317

IGRAPH b72da1b UN-- 4317 18062 --
+ attr: name (v/c), type (e/c)
+ edges from b72da1b (vertex names):
 [1] AAMP   --VPS52     AAMP   --BHLHE40   AAMP   --GABARAPL2 AAMP   --MAP1LC3B
 [5] VPS52  --TXN2      VPS52  --DDX6      VPS52  --MFAP1     VPS52  --PRKAA1
 [9] VPS52  --LMO4      VPS52  --STX11     VPS52  --KANK2     VPS52  --PPP1R18
[13] VPS52  --TXLNA     VPS52  --KIAA1217  VPS52  --VPS28     VPS52  --ATP6V1D
[17] VPS52  --TPM3      VPS52  --KIF5B     VPS52  --NOP2      VPS52  --RNF41
[21] VPS52  --WTAP      VPS52  --MAPK3     VPS52  --ZMAT2     VPS52  --VPS51
[25] BHLHE40--AES       BHLHE40--PRKAA1    BHLHE40--CCNK      BHLHE40--RBPMS
[29] BHLHE40--COX5B     BHLHE40--UBE2I     BHLHE40--MAGED1    BHLHE40--PLEKHB2
+ ... omitted several edges

Number of Nodes of the second network:
[1] 6947

Second Network
IGRAPH 8f3386a UN-- 6947 28246 --
+ attr: name (v/c), type (e/c)
+ edges from 8f3386a (vertex names):
 [1] 100050--122470 100050--227330 100050--259200 100050--305400 100050--601803
 [6] 100070--105800 100070--105805 100070--107550 100070--120000 100070--130090
[11] 100070--132900 100070--154750 100070--180300 100070--192310 100070--208060
[16] 100070--210050 100070--219100 100070--252350 100070--277175 100070--300537
[21] 100070--309520 100070--600459 100070--604308 100070--606519 100070--608967
[26] 100070--609192 100070--610168 100070--610380 100070--611788 100070--613780
[31] 100070--613834 100070--614042 100070--614437 100070--614980 100070--610655
[36] 100070--615436 100070--616166 100100--192350 100100--236700 100100--236730
+ ... omitted several edges
```

To apply the RWR-H on a heterogeneous network, we need to compute a matrix that accounts for all the possible transitions of the RWR particle within that network [5].

```
> PPIHetTranMatrix <- compute.transition.matrix(PPI_Disease_Net)
```

Before running RWR-H on this PPI-disease heterogeneous network, we need to define the seed(s). As in the previous paragraph, we take *PIK3R1* as a seed gene. In addition, we can now set the SHORT syndrome itself as a seed disease.

```
> SeedDisease <- c("269880")
> ## We launch the algorithm with the default parameters (See details on manual)
> RWRH_PPI_Disease_Results <-
+     Random.Walk.Restart.MultiplexHet(PPIHetTranMatrix,
+     PPI_Disease_Net,SeedGene,SeedDisease)
> # We display the results
> RWRH_PPI_Disease_Results
```

```
Top 10 ranked Multiplex nodes:
     NodeNames       Score
1479      GRB2 0.001965500
1081      EGFR 0.001754048
797        CRK 0.001636329
603      CDC42 0.001630575
19        ABL1 0.001623304
798       CRKL 0.001605543
1360       FYN 0.001598567
3405      SHC1 0.001597720
2583    PDGFRB 0.001596830
4027     TYRO3 0.001589911

Multiplex Seed Nodes used:
[1] "PIK3R1"
Top 10 ranked Second Network Nodes:
     SecondNet_node       Score
6352         615214 0.020817435
6705         616005 0.020785895
1699         194050 0.005868407
3625         309000 0.005687206
2901         262500 0.005681162
686          138920 0.005655559
2150         223370 0.005655117
4770         608612 0.005649291
4464         606176 0.005641777
1411         180500 0.005639919

Second Network Seed Nodes used:
[1] "269880"
```

Finally, we can create a heterogeneous network (an *igraph* object) with the top scored genes and the top scored diseases. The results are presented in Figure 2.

```
> ## In this case we select to induce a network with the Top 10 genes
> ## and the Top 10 diseases.
> TopResults_PPI_Disease <-
+     create.multiplexHetNetwork.topResults(RWRH_PPI_Disease_Results,
+     PPI_Disease_Net, GeneDiseaseRelations_PPI, k=10)
```

```
> par(mar=c(0.1,0.1,0.1,0.1))
> plot(TopResults_PPI_Disease, vertex.label.color="black",
+     vertex.frame.color="#ffffff",
+     vertex.size= 20, edge.curved=.2,
+     vertex.color = ifelse(V(TopResults_PPI_Disease)$name == "PIK3R1"
+     | V(TopResults_PPI_Disease)$name == "269880","yellow",
+     ifelse(V(TopResults_PPI_Disease)$name %in% PPI_Disease_Net$Pool_of_Nodes,
+     "#00CCFF","Grey75")),
+     edge.color=ifelse(E(TopResults_PPI_Disease)$type == "PPI","blue",
+         ifelse(E(TopResults_PPI_Disease)$type == "Disease","black","grey50")),
+     edge.width=0.8,
```

```
+     edge.lty=ifelse(E(TopResults_PPI_Disease)$type == "bipartiteRelations",
+         2,1),
+     vertex.shape= ifelse(V(TopResults_PPI_Disease)$name %in%
+         PPI_Disease_Net$Pool_of_Nodes,"circle","rectangle"))
```

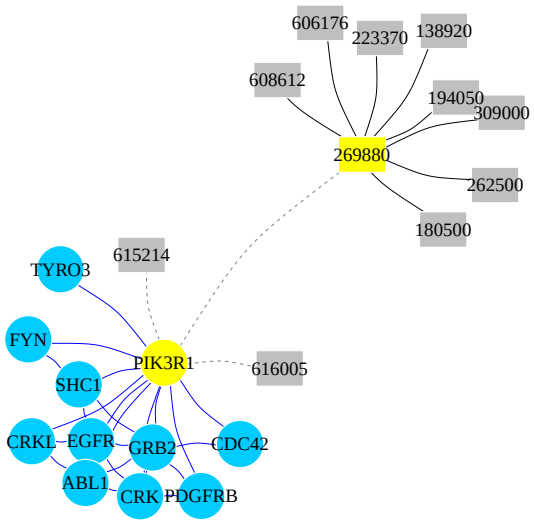

Figure 2: RWR-H on a heterogeneous PPI-Disease Network
Network representation of the top 10 ranked genes and the top 10 ranked diseases when the RWR-H algorithm is executed using the *PIK3R1* gene and the SHORT syndrome disease (MIM code: 269880) as seeds (yellow nodes). Circular nodes represent genes and rectangular nodes show diseases. Blue edges are PPI interactions and black edges are similarity links between diseases. Dashed edges are the bipartite gene-disease associations.

3.3 Random Walk with Restart on a Multiplex Network

Some limitations can arise when single networks are used to represent and describe systems whose entities can interact through more than one type of connections [2]. This is the case of social interactions, transportation networks or biological systems, among others. The Multiplex framework provides an appealing approach to describe these systems, since they are able to integrate this diversity of data while keeping track of the original features and topologies of the different sources.

Consequently, algorithms able to exploit the information stored on multiplex networks should improve the results provided by methods operating on single networks. In this context, we extended the random walk with restart algorithm to multiplex networks (RWR-M) [3].

In the following example, we create a multiplex network integrated by our PPI network and a network derived from pathway databases [3].

```
> data(Pathway_Network) # We load the Pathway Network
> ## We create a 2-layers Multiplex object
> PPI_PATH_Multiplex <- create.multiplex(PPI_Network,Pathway_Network,
+                       Layers_Name=c("PPI","PATH"))
> PPI_PATH_Multiplex
```

```
Number of Layers:
[1] 2

Number of Nodes:
[1] 4899

IGRAPH 9f0a289 UN-- 4899 18062 --
+ attr: name (v/c), type (e/c)
+ edges from 9f0a289 (vertex names):
 [1] AAMP   --VPS52     AAMP   --BHLHE40   AAMP   --GABARAPL2 AAMP   --MAP1LC3B
 [5] VPS52  --TXN2      VPS52  --DDX6      VPS52  --MFAP1     VPS52  --PRKAA1
 [9] VPS52  --LMO4      VPS52  --STX11     VPS52  --KANK2     VPS52  --PPP1R18
[13] VPS52  --TXLNA     VPS52  --KIAA1217  VPS52  --VPS28     VPS52  --ATP6V1D
[17] VPS52  --TPM3      VPS52  --KIF5B     VPS52  --NOP2      VPS52  --RNF41
[21] VPS52  --WTAP      VPS52  --MAPK3     VPS52  --ZMAT2     VPS52  --VPS51
[25] BHLHE40--AES       BHLHE40--PRKAA1    BHLHE40--CCNK      BHLHE40--RBPMS
[29] BHLHE40--COX5B     BHLHE40--UBE2I     BHLHE40--MAGED1    BHLHE40--PLEKHB2
+ ... omitted several edges

IGRAPH 6e0d7bb UN-- 4899 62602 --
+ attr: name (v/c), type (e/c)
+ edges from 6e0d7bb (vertex names):
 [1] BANF1--PSIP1   BANF1--HMGA1   BANF1--PPP2R1A BANF1--PPP2CA  BANF1--KPNA1
 [6] BANF1--TPR     BANF1--NUP62   BANF1--NUP153  BANF1--RANBP2  BANF1--NUP54
[11] BANF1--POM121  BANF1--NUP85   BANF1--PPP2R2A BANF1--EMD     BANF1--LEMD2
[16] BANF1--ANKLE2  PSIP1--HMGA1   PSIP1--KPNA1   PSIP1--TPR     PSIP1--NUP62
[21] PSIP1--NUP153  PSIP1--RANBP2  PSIP1--NUP54   PSIP1--POM121  PSIP1--NUP85
[26] HMGA1--TP53    HMGA1--KPNA1   HMGA1--TPR     HMGA1--NUP62   HMGA1--NUP153
[31] HMGA1--RANBP2  HMGA1--NUP54   HMGA1--POM121  HMGA1--NUP85   HMGA1--MYC
[36] HMGA1--RB1     HMGA1--MAX     HMGA1--RPS6KB1 XRCC6--XRCC5   XRCC6--IRF3
+ ... omitted several edges
```

Afterwards, as in the monoplex case, we have to compute and normalize the adjacency matrix of the multiplex network.

```
> AdjMatrix_PPI_PATH <- compute.adjacency.matrix(PPI_PATH_Multiplex)
> AdjMatrixNorm_PPI_PATH <- normalize.multiplex.adjacency(AdjMatrix_PPI_PATH)
```

Then, we set again as seed the *PIK3R1* gene and we perform RWR-M on this new multiplex network.

```
> ## We launch the algorithm with the default parameters (See details on manual)
> RWR_PPI_PATH_Results <- Random.Walk.Restart.Multiplex(AdjMatrixNorm_PPI_PATH,
+                         PPI_PATH_Multiplex,SeedGene)
> # We display the results
> RWR_PPI_PATH_Results

Top 10 ranked Nodes:
   NodeNames       Score
1       GRB2 0.001662893
2        FYN 0.001517786
3       HCST 0.001506594
```

```
4       EGFR 0.001494459
5       SHC1 0.001492456
6       PTK2 0.001430282
7       JAK2 0.001401867
8       HRAS 0.001396854
9       CRKL 0.001389391
10    PDGFRB 0.001369533

Seed Nodes used:
[1] "PIK3R1"
```

Finally, we can create a multiplex network (an *igraph* object) with the top scored genes. The results are presented in Figure 3.

```
> ## In this case we select to induce a multiplex network with the Top 15 genes.
> TopResults_PPI_PATH <-
+     create.multiplexNetwork.topResults(RWR_PPI_PATH_Results,
+                                        PPI_PATH_Multiplex, k=15)
```

```
> par(mar=c(0.1,0.1,0.1,0.1))
> plot(TopResults_PPI_PATH, vertex.label.color="black",
+     vertex.frame.color="#ffffff", vertex.size= 20,
+     edge.curved= ifelse(E(TopResults_PPI_PATH)$type == "PPI",
+                     0.4,0),
+     vertex.color = ifelse(igraph::V(TopResults_PPI_PATH)$name == "PIK3R1",
+                     "yellow","#00CCFF"),edge.width=0.8,
+     edge.color=ifelse(E(TopResults_PPI_PATH)$type == "PPI",
+                       "blue","red"))
```

3.4 Random Walk with Restart on a Multiplex-Heterogeneous Network

RWR-H and RWR-M remarkably improve the results obtained by classical RWR on monoplex networks, as we demonstrated in the particular case of retrieving known gene-disease associations [3]. Therefore, an algorithm able to execute a random walk with restart on both, multiplex and heterogeneous networks, is expected to achieve an even better performance. We extended our RWR-M approach to heterogeneous networks, defining a random walk with restart on multiplex-heterogeneous networks (RWR-MH) [3].

Let us integrate all the networks described previously (PPI, Pathways and disease-disease similarity) into a multiplex and heterogeneous network. To do so, we connect genes in both multiplex layers (PPI and Pathways) to the disease network, if a bipartite gene-disease relation exists.

```
> ## We keep gene-diseases associations where genes are present in the PPI
> ## or in the pathway network
> GeneDiseaseRelations_PPI_PATH <-
+     GeneDiseaseRelations[which(GeneDiseaseRelations$hgnc_symbol %in%
+     PPI_PATH_Multiplex$Pool_of_Nodes),]
> ## We create the MultiplexHet object.
> PPI_PATH_Disease_Net <- create.multiplexHet(PPI_PATH_Multiplex,
```

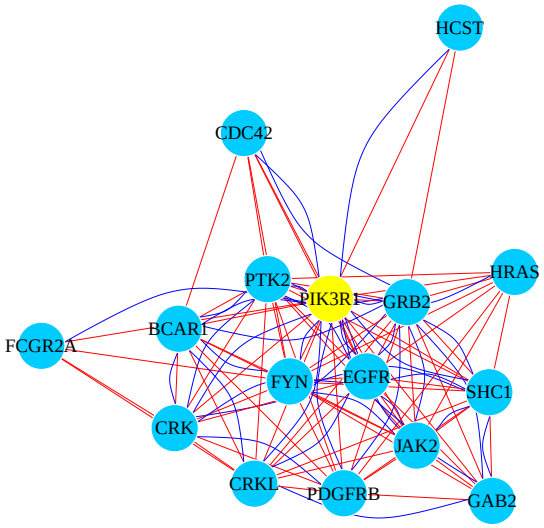

Figure 3: **RWR-M on a multiplex PPI-Pathway Network**
Network representation of the top 15 ranked genes when the RWR-M algorithm is executed using the *PIK3R1* gene (yellow node). Blue curved edges are PPI interactions and red straight edges are Pathways links. All the interactions are aggregated into a monoplex network only for visualization purposes.

```
+     Disease_Network, GeneDiseaseRelations_PPI_PATH, c("Disease"))
> ## The results look like that
> PPI_PATH_Disease_Net

Number of Layers:
[1] 2

Number of Nodes Multiplex:
[1] 4899

IGRAPH 9f0a289 UN-- 4899 18062 --
+ attr: name (v/c), type (e/c)
+ edges from 9f0a289 (vertex names):
 [1] AAMP   --VPS52     AAMP   --BHLHE40   AAMP   --GABARAPL2 AAMP   --MAP1LC3B
 [5] VPS52  --TXN2      VPS52  --DDX6      VPS52  --MFAP1     VPS52  --PRKAA1
 [9] VPS52  --LMO4      VPS52  --STX11     VPS52  --KANK2     VPS52  --PPP1R18
[13] VPS52  --TXLNA     VPS52  --KIAA1217  VPS52  --VPS28     VPS52  --ATP6V1D
[17] VPS52  --TPM3      VPS52  --KIF5B     VPS52  --NOP2      VPS52  --RNF41
[21] VPS52  --WTAP      VPS52  --MAPK3     VPS52  --ZMAT2     VPS52  --VPS51
[25] BHLHE40--AES       BHLHE40--PRKAA1    BHLHE40--CCNK      BHLHE40--RBPMS
[29] BHLHE40--COX5B     BHLHE40--UBE2I     BHLHE40--MAGED1    BHLHE40--PLEKHB2
+ ... omitted several edges

IGRAPH 6e0d7bb UN-- 4899 62602 --
+ attr: name (v/c), type (e/c)
+ edges from 6e0d7bb (vertex names):
 [1] BANF1--PSIP1   BANF1--HMGA1   BANF1--PPP2R1A BANF1--PPP2CA  BANF1--KPNA1
```

```
 [6] BANF1--TPR     BANF1--NUP62   BANF1--NUP153  BANF1--RANBP2  BANF1--NUP54
[11] BANF1--POM121  BANF1--NUP85   BANF1--PPP2R2A BANF1--EMD     BANF1--LEMD2
[16] BANF1--ANKLE2  PSIP1--HMGA1   PSIP1--KPNA1   PSIP1--TPR     PSIP1--NUP62
[21] PSIP1--NUP153  PSIP1--RANBP2  PSIP1--NUP54   PSIP1--POM121  PSIP1--NUP85
[26] HMGA1--TP53    HMGA1--KPNA1   HMGA1--TPR     HMGA1--NUP62   HMGA1--NUP153
[31] HMGA1--RANBP2  HMGA1--NUP54   HMGA1--POM121  HMGA1--NUP85   HMGA1--MYC
[36] HMGA1--RB1     HMGA1--MAX     HMGA1--RPS6KB1 XRCC6--XRCC5   XRCC6--IRF3
+ ... omitted several edges

Number of Nodes of the second network:
[1] 6947

Second Network
IGRAPH 0f94bd1 UN-- 6947 28246 --
+ attr: name (v/c), type (e/c)
+ edges from 0f94bd1 (vertex names):
 [1] 100050--122470 100050--227330 100050--259200 100050--305400 100050--601803
 [6] 100070--105800 100070--105805 100070--107550 100070--120000 100070--130090
[11] 100070--132900 100070--154750 100070--180300 100070--192310 100070--208060
[16] 100070--210050 100070--219100 100070--252350 100070--277175 100070--300537
[21] 100070--309520 100070--600459 100070--604308 100070--606519 100070--608967
[26] 100070--609192 100070--610168 100070--610380 100070--611788 100070--613780
[31] 100070--613834 100070--614042 100070--614437 100070--614980 100070--610655
[36] 100070--615436 100070--616166 100100--192350 100100--236700 100100--236730
+ ... omitted several edges
```

To apply the RWR-MH on a multiplex and heterogeneous network, we need to compute a matrix that accounts for all the possible transitions of the RWR particle within this network [3].

```
> PPI_PATH_HetTranMatrix <- compute.transition.matrix(PPI_PATH_Disease_Net)
```

As in the RWR-H situation, we can take as seeds both, the *PIK3R1* gene and the the SHORT syndrome disease.

```
> ## We launch the algorithm with the default parameters (See details on manual)
> RWRH_PPI_PATH_Disease_Results <-
+     Random.Walk.Restart.MultiplexHet(PPI_PATH_HetTranMatrix,
+     PPI_PATH_Disease_Net,SeedGene,SeedDisease)
> # We display the results
> RWRH_PPI_PATH_Disease_Results

Top 10 ranked Multiplex nodes:
     NodeNames        Score
1616       GHR 0.0005153889
4867  ZMPSTE24 0.0004781314
1706      GRB2 0.0004642946
1625      GJA1 0.0004457777
1781      HCST 0.0004408101
1249      EGFR 0.0004239882
1564       FYN 0.0004193109
3883      SHC1 0.0004161564
```

```
2990    PDGFRB 0.0004079884
3361      PTK2 0.0004069179

Multiplex Seed Nodes used:
[1] "PIK3R1"
Top 10 ranked Second Network Nodes:
     SecondNet_node       Score
6352         615214 0.020797695
6705         616005 0.020775842
1699         194050 0.005868202
4770         608612 0.005691321
2901         262500 0.005691027
3625         309000 0.005683228
686          138920 0.005653962
2150         223370 0.005653358
4464         606176 0.005639004
1411         180500 0.005631733

Second Network Seed Nodes used:
[1] "269880"
```

Finally, we can create a multiplex and heterogeneous network (an *igraph* object) with the top scored genes and the top scored diseases. The results are presented in Figure 4.

```
> ## In this case we select to induce a network with the Top 10 genes.
> ## and the Top 10 diseases.
> TopResults_PPI_PATH_Disease <-
+     create.multiplexHetNetwork.topResults(RWRH_PPI_PATH_Disease_Results,
+     PPI_PATH_Disease_Net, GeneDiseaseRelations_PPI_PATH, k=10)
```

```
> par(mar=c(0.1,0.1,0.1,0.1))
> plot(TopResults_PPI_PATH_Disease, vertex.label.color="black",
+     vertex.frame.color="#ffffff",
+     vertex.size= 20,
+     edge.curved=ifelse(E(TopResults_PPI_PATH_Disease)$type == "PATH",
+                     0,0.3),
+     vertex.color = ifelse(V(TopResults_PPI_PATH_Disease)$name == "PIK3R1"
+     | V(TopResults_PPI_Disease)$name == "269880","yellow",
+     ifelse(V(TopResults_PPI_PATH_Disease)$name %in%
+                PPI_PATH_Disease_Net$Pool_of_Nodes,
+     "#00CCFF","Grey75")),
+     edge.color=ifelse(E(TopResults_PPI_PATH_Disease)$type == "PPI","blue",
+     ifelse(E(TopResults_PPI_PATH_Disease)$type == "PATH","red",
+     ifelse(E(TopResults_PPI_PATH_Disease)$type == "Disease","black","grey50"))),
+     edge.width=0.8,
+     edge.lty=ifelse(E(TopResults_PPI_PATH_Disease)$type ==
+         "bipartiteRelations", 2,1),
+     vertex.shape= ifelse(V(TopResults_PPI_PATH_Disease)$name %in%
+         PPI_PATH_Disease_Net$Pool_of_Nodes,"circle","rectangle"))
```

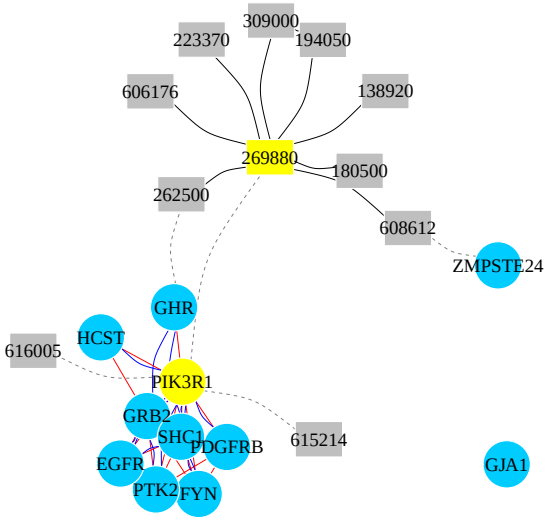

Figure 4: **RWR-MH on a multiplex and heterogeneous network (PPI-Pathway-Disease)** Network representation of the top 10 ranked genes and the top 10 ranked diseases when the RWR-H algorithm is executed using the *PIK3R1* gene and the SHORT syndrome disease (MIM code: 269880) as seeds (yellow nodes). Circular nodes represent genes and rectangular nodes show diseases. Blue curved edges are PPI interactions and red straight edges are Pathways links. Black edges are similarity links between diseases. Dashed edges are the bipartite gene-disease associations. Multiplex interactions are aggregated into a monoplex network only for visualization purposes.

References

[1] Jia-yu Pan, Hyung Yang, Pinar Duygulu, and Christos Faloutsos. Automatic Multimedia Cross-modal Correlation Discovery. pages 653–658, 2004.

[2] Federico Battiston, Vincenzo Nicosia, and Vito Latora. Structural measures for multiplex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 89(3):1–16, 2014. arXiv:1308.3182, doi:10.1103/PhysRevE.89.032804.

[3] Alberto Valdeolivas, Laurent Tichit, Claire Navarro, Sophie Perrin, Gaelle Odelin, Nicolas Levy, Pierre Cau, Elisabeth Remy, and Anais Baudot. Random Walk With Restart On Multiplex And Heterogeneous Biological Networks. *bioRxiv*, pages 1–31, 2017. URL: http://biorxiv.org/content/early/2017/05/05/134734?rss=1, doi:10.1101/134734.

[4] Sebastian Kohler, Sebastian Bauer, Denise Horn, and Peter N Robinson. AJHG - Walking the Interactome for Prioritization of Candidate Disease Genes. (April):949–958, 2008. URL: http://www.cell.com/AJHG/abstract/S0002-9297(08)00172-9, doi:10.1016/j.ajhg.2008.02.013.

[5] Yongjin Li and Jagdish C. Patra. Genome-wide inferring gene-phenotype relationship by walking on the heterogeneous network. *Bioinformatics*, 26(9):1219–1224, 2010. doi:10.1093/bioinformatics/btq108.

[6] Ada Hamosh, Alan F. Scott, Joanna S. Amberger, Carol A. Bocchini, and Victor A. McKusick. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research*, 33(DATABASE ISS.):514–517, 2005. doi:10.1093/nar/gki033.

A Session info

- R version 4.1.0 (2021-05-18), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04.2 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.13-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.13-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: RandomWalkRestartMH 1.12.0, igraph 1.2.6
- Loaded via a namespace (and not attached): BiocGenerics 0.38.0, BiocManager 1.30.15, BiocStyle 2.20.0, DBI 1.1.1, MASS 7.3-54, Matrix 1.3-3, R6 2.5.0, Rcpp 1.0.6, Rgraphviz 2.36.0, ape 5.5, assertthat 0.2.1, compiler 4.1.0, crayon 1.4.1, digest 0.6.27, dnet 1.1.7, dplyr 1.0.6, ellipsis 0.3.2, evaluate 0.14, fansi 0.4.2, generics 0.1.0, glue 1.4.2, graph 1.70.0, grid 4.1.0, hexbin 1.28.2, hms 1.1.0, htmltools 0.5.1.1, knitr 1.33, lattice 0.20-44, lifecycle 1.0.0, magrittr 2.0.1, nlme 3.1-152, parallel 4.1.0, pillar 1.6.1, pkgconfig 2.0.3, purrr 0.3.4, readr 1.4.0, rlang 0.4.11, rmarkdown 2.8, stats4 4.1.0, stringi 1.6.2, stringr 1.4.0, supraHex 1.30.0, tibble 3.1.2, tidyr 1.1.3, tidyselect 1.1.1, tools 4.1.0, utf8 1.2.1, vctrs 0.3.8, xfun 0.23, yaml 2.2.1