Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Systems biology

RANKS: a flexible tool for node label ranking and classification in biological networks: Supplementary Information

Giorgio Valentini [1,*], Giuliano Armano [2], Marco Frasca [1], Jianyi Lin [1], Marco Mesiti [1] and Matteo Re [1]

[1] AnacletoLab, Department of Computer Science, University of Milan, 20139 Milan, Italy and
[2] Department of Electric and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy.

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

The Supplementary Information includes a detailed description of *RANKS – RAnking of Nodes with Kernelized Score functions*, and some examples of its application to the Gene Function Prediction, Drug repositioning and Human Phenotype Ontology prediction problems. All the examples include R code that shows how to use the *RANKS* R package in the context of node label ranking problems. A final section explains how to expand the functionalities of the package through user defined score functions and kernels.

1 The *RANKS* method

Most of the node label ranking algorithms proposed for the analysis of biomolecular networks exploit local or global learning strategies to properly rank nodes [25, 22, 19, 5, 6]. In this context nodes represent usually genes or proteins, and edges functional similarities between nodes, according to the biological property under investigation.

RANKS – RAnking of Nodes with Kernelized Score functions is a very fast semi-supervised network method that combines local and global learning strategies to exploit both "local" similarities between nodes and "global" similarities embedded in the topology of the biomolecular network. From this standpoint *RANKS* can be considered a generalization of both guilt-by-association methods [20], and kernel based algorithms for semi-supervised network analysis [11].

Indeed, the guilt-by-association (GBA) approach [20] is generalized through fast and efficient local learning strategies based on an extended notion of functional distance between nodes. Global learning strategies are introduced by using kernel functions able to exploit the relationships and the overall topology of the underlying biological network. This approach can be also seen as a general algorithmic scheme: by introducing different local score functions and by choosing different kernels to model the similarity between nodes, we can derive different network-based algorithms. For instance, by adopting graph kernels [11], both direct and indirect relationships between genes can be exploited, thus taking into account the overall topology of the network.

Let $G = (V, E)$ be an undirected graph with weighted adjacency matrix $\boldsymbol{W}$ representing a biomolecular network $\boldsymbol{W}$ (e.g. a gene or protein network), where V denotes the set of nodes (corresponding, for instance, to genes or proteins), and $V_C \subset V$ denotes a subset of nodes having a specific property C (C could be, for instance, a Gene Ontology term, or a genetic disease). For the sake of simplicity, we can represent the nodes of the graph with natural numbers $1, 2, \ldots, n$. Moreover a set of features $\boldsymbol{x}_i \in X$ can be associated to a node i. For instance, $\boldsymbol{x}_i$ could represent the expression or the phylogenetic profile of a node i, or whatever available data for a given gene/node i.

The *node label ranking problem* consists of finding a score function $S : V \longrightarrow \mathbb{R}^+$ by which we can directly rank nodes according to their likelihood to belong to a specific category C (the higher the score, the higher the likelihood that a node belongs to C). It is worth noting that *node label ranking* can be seen as a "one-class" semi-supervised learning problem on biomolecular networks G, since we can exploit the labeling of the known "positive" vertices $v \in V_C$ belonging to the category C, but also the similarity relationships between labeled or unlabeled vertices $v \in V$.

In *RANKS* score functions are based on distance measures defined in a suitable Hilbert space $\mathcal{H}$. More precisely, let $\phi : X \to \mathcal{H}$, be a mapping to a given Reproducing Kernel Hilbert Space $\mathcal{H}$, and $K : X \times X \to \mathbb{R}$ its associated kernel function, such that $< \phi(\cdot), \phi(\cdot) >_{\mathcal{H}} = K(\cdot, \cdot)$, where $< \cdot, \cdot >_{\mathcal{H}}$ represents an internal product in $\mathcal{H}$.

A distance measure $D(i, V_C)$ in the Hilbert space between a given node/gene i and the set of nodes V_C having a specific property C can be thus introduced by using a proper kernel function [32]. Indeed, we can embed any valid kernel into the distance measure itself, thus resulting in a modular approach by which existing graph kernels, or in perspective new graph kernels properly designed for node label ranking problems in biomolecular networks, can be applied to rank nodes according to the property C under study. By choosing different distance measures, diverse score functions can be derived. As examples we consider the following score functions, each one based on a different notion of distance between nodes:

1. *Nearest Neighbours* score
2. *K-Nearest Neighbours* score
3. *Average score*

1.1 Nearest-Neighbours Score

We can define a distance measure $D_{AV}(i, V_C)$ of a vertex $i \in V$ w.r.t. to a set of nodes V_C, as the minimum squared distance in the Hilbert space between i and V_C:

$$D_{NN}(i, V_C) = \min_{j \in V_C} \left|\left| \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j) \right|\right|^2 \quad (1)$$

By developing the square (1) we can obtain:

$$D_{NN}(i, V_C) = \min_{j \in V_C} [< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i) > + < \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_j) > -2 < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >] \quad (2)$$

where $< \cdot, \cdot >$ is the inner product in the feature space $\mathcal{H}$. We can achieve a similarity measure by simply changing the sign of equation 2 and recalling that $< \phi(\cdot), \phi(\cdot) >= K(\cdot, \cdot)$:

$$Sim_{NN}(i, V_C) = - \min_{j \in V_C} [K(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j) + K(\boldsymbol{x}_j, \boldsymbol{x}_j)] \quad (3)$$

If $K(\boldsymbol{x}_j, \boldsymbol{x}_j)$ is equal for all $j \in V$, we can disregard these terms, thus achieving the *nearest neighbours score* S_{NN}:

$$S_{NN}(i, V_C) = - \min_{j \in V_C} -2K(\boldsymbol{x}_i, \boldsymbol{x}_j) = 2 \max_{j \in V_C} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad (4)$$

1.2 K-Nearest-Neighbours Score

If we consider the k-nearest neighbours, i.e. $I_k(i) = \{j \in V_C | j$ is ranked among the first k nearest neighbors of $i\}$, then we can easily extend the S_{NN} score by introducing the *k-nearest neighbours distance*:

$$D_{kNN}(i, V_C) = \sum_{j \in I_k(i)} \left|\left| \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j) \right|\right|^2 \quad (5)$$

By expanding the square in (5) and inverting the sign we can obtain a similarity measure:

$$Sim_{kNN}(i, V_C) = = - \sum_{j \in I_k(v)} (K(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j) + K(\boldsymbol{x}_j, \boldsymbol{x}_j)) \quad (6)$$

This similarity measure can be directly used as a *k-nearest neighbours score* S_{kNN}, but in the case that $K(\boldsymbol{x}_j, \boldsymbol{x}_j)$ is equal for all $j \in V$, we can disregard constant terms and hence obtain the simplification:

$$S_{kNN}(i, V_C) = 2 \sum_{j \in I_k(i)} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad (7)$$

1.3 Average Score

Another distance can be simply defined as the average distance in the Hilbert space $\mathcal{H}$ between i and the set of nodes V_C:

$$D_{AV}(i, V_C) = \left|\left| \phi(\boldsymbol{x}_i) - \frac{1}{|V_C|} \sum_{j \in V_C} \phi(\boldsymbol{x}_j) \right|\right|^2 \quad (8)$$

By expanding the square (8) we obtain:

$$D_{AV}(i, V_C) =< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i) > - \frac{2}{|V_C|} \sum_{j \in V_C} < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) > + \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} < \phi(\boldsymbol{x}_k), \phi(\boldsymbol{x}_j) > \quad (9)$$

Also in this case a similarity measure can be obtained by changing the sign:

$$Sim_{AV}(i, V_C) = -K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \frac{2}{|V_C|} \sum_{j \in V_C} K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} K(\boldsymbol{x}_k, \boldsymbol{x}_j) \quad (10)$$

We can finally obtain the *average score* S_{AV}, by observing that the third term of (10) is equal for all $i \in V$:

$$S_{AV}(i, V_C) = -K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \frac{2}{|V_C|} \sum_{j \in V_C} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad (11)$$

By using the proposed kernelized score functions we can rank nodes with respect to their likelihood to belong to a given category C simply by evaluating the selected kernel function. If the kernel matrix is computed in advance, the time complexity of *RANKS* is $\mathcal{O}(|V_C||V|)$, that is approximately linear with respect to the number of nodes when $|V_C| << |V|$.

1.4 Random Walk kernels

We can obtain different node ranking algorithms by embedding different kernels in eq. 4, 7 and 11. In principle, any valid kernel can be used (e.g. linear, polynomial, gaussian, Laplacian, Cauchy and inverse multiquadric kernels), but in the context of biomolecular networks it is often meaningful to use a *random walk kernel* [34] constructed from the weighted adjacency matrix $\boldsymbol{W}$ of the graph under study. Indeed it can capture not only relationships coming from direct neighborhoods between nodes, similarly to *guilt by association* methods [20], but also relationships coming from shared and more in general indirect neighbours between nodes. For instance, in the context of network-based Gene Function Prediction (GFP), while it is quite obvious that functional relationships are coded into direct neighbours, important functional relationships between genes can also be coded through indirect neighbours [3]. For instance, enzymes belonging to the same biological process may not share the same links, since their catalyzed reactions can be linked through other intermediate reactions belonging to the same pathway, but of course the involved enzymes do belong to the same biological process.

Random walk kernels represent the kernelized version of *Markov Random Walks*, by which random trajectories across graphs can be exploited to investigate the relationships between nodes and to score or label each node with respect to a specific property of the nodes[18].

The one-step random walk kernel matrix $\boldsymbol{K}$ can be obtained from the adjacency matrix $\boldsymbol{W}$ of the graph in the following way:

$$\boldsymbol{K} = (a-1)\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}^{-\frac{1}{2}} \quad (12)$$

where $\boldsymbol{D}$ is the "degree" diagonal matrix with elements $d_{ii} = \sum_j w_{ij}$, $\boldsymbol{I}$ is the identity matrix and a is a value larger than 2. The *q-step random*

walk kernel can be directly obtained from (12):

$$\boldsymbol{K}^q = [(a-1)\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}}]^q \tag{13}$$

where $q \geq 2$ is an integer representing the number of steps of the random walk across the graph and can be easily computed by a recursive chain of matrix multiplications:

$$\boldsymbol{K}^q = \boldsymbol{K}^{q-1}\boldsymbol{K} \tag{14}$$

Fig. 1 provides a derivation of the random walk kernel.

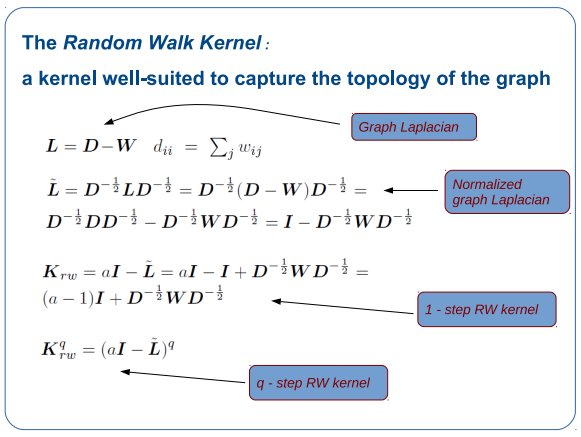

Fig. 1. Derivation of the random walk kernel.

By setting $q = 2$, the random walks consider indirect neighbours, that is two nodes are similar if either they are directly connected or they share common nodes in their neighborhood. It is worth noting that if $q = 1$ we obtain the *one-step random walk kernel*, by which only the direct neighbours of each node are visited. More in general, by setting $q > 2$ two vertices are considered similar if they are directly connected or if they are indirectly connected through a path including from 1 to $q - 1$ intermediate vertices. Large values of q may introduce remote similarities between nodes, in a way similar to diffusion kernels [14]. It can be shown that (13) is up to scaling terms equivalent to a q-step random walk on the graph with random restarts, a well-known algorithm used for scoring web pages in the Google search engine [2].

2 Application examples

RANKS has been successfully applied to gene function prediction, gene disease prioritization and drug repositioning ([24]) problems, comparing favourably with state-of-the-art network based methods [27, 28, 29, 38].

In this section we provide several application examples of *RANKS*, including the R code that shows how to use the *RANKS* R package in the context of node label ranking problems. More precisely, we at first consider an example of application to two classical prediction problems, Gene Function Prediction [26] and Drug repositioning [33]. Then we describe an application to the Human Phenotype Ontology prediction problem [13], a complex prediction task proposed in the recent Critical Assessment of Functional Annotation 2 (CAFA2) international challenge, where *RANKS* was one of the top ranked method in this specific prediction task (`http://biofunctionprediction.org/node/8`) [9].

The *RANKS* package is freely downloadable from CRAN (`https://cran.r-project.org/` for Unix, Windows and Mac operating systems. The data used in these examples are downloadable from `http://homes.di.unimi.it/valentini/DATA/RANKS`. Detailed explanations about the syntax and the semantic of each function and method implemented in the package are available in the *RANKS* reference manual:
(`https://cran.r-project.org/web/packages/RANKS/RANKS.pdf`).

2.1 Gene Function Prediction

Gene Function Prediction (GFP) can be formalized as a node label ranking problem, where nodes represent genes and edges relationships weighted according to the evidence of co-functionality implied by data sources [40]. By exploiting the labeling of a subset of genes annotated with a specific function, and the topology of the network, the prediction task consists of ranking unlabeled nodes with respect to the function under study [23, 4, 21].

In this section we present a simple application for ranking genes with respect to FunCat (Functional Catalogue) classes with the yeast model organism [31]. Here we limited the experiments to a relatively low number of genes and FunCat classes to allow to easily experiment with *RANKS*. We combined 6 bio-molecular data sets previously used for the related task of gene classification [35]. The data sets include pairwise sequence similarity data, protein-protein interaction, protein domain and gene expression data. We considered only yeast genes common to all data sets. Moreover, in order to get a not too small set of positive examples for training, for each data set we selected only the FunCat-annotated genes[1], and classes with at least 20 positive examples, using the *HCgene* R package [36]. This selection process yielded 1901 yeast genes annotated to 168 FunCat classes (see [27] for more details about the construction of the integrated network).

Let us now illustrate, step-by-step, how to use the package in an R script to predict gene functions in the yeast using 5-fold cross-validation, repeated 10 times, in order to evaluate the scores for each of the 1901 yeast genes relative to all the considered FunCat classes. At first we need to load the necessary packages:

```
library(RANKS);
library(PerfMeas);
```

PerfMeas is another package available from CRAN to measure the performance of supervised and semi-supervised learning methods (i.e. accuracy, AUC, etc). Then network data (in the form of a weighted adjacency matrix) and annotation data (another matrix T, where $T[i,j] = 1$ if gene i is annotated to the FunCat category j, otherwise $T[i,j] = 0$) are loaded:

```
#loading network data
data.file <- "data/yeast.data.matrix.rda";
load(data.file); # avgKF.matrix loaded
W <- avgKF.matrix;  # 1901 X 1901
rm(avgKF.matrix);
# filtering negative values
W[W<0]<-0;
# read labels matrix with gene annotations
labels.file <- "data/yeast.label.matrix.rda";
load(labels.file);
T <- Yeast.Funcat.Table.intersection.filtered20ormoreGenes;
rm(Yeast.Funcat.Table.intersection.filtered20ormoreGenes);
T <- as.matrix(T[,-1]); # root category deleted
nclasses <- ncol(T);
ngenes <- nrow(T);
```

[1] Our experiments build on annotations coded in the FunCat-2.1 scheme, and FunCat-2.1_data_20070316 data, available from the MIPS web site (http://mips.gsf.de/projects/funcat).

```
classnames <- colnames(T);
```

Then the score matrix to collect the final scores and the random walk kernel are constructed using the $rw.kernel$ *RANKS* method:

```
# construction of the matrix of scores.
S1  <- matrix(numeric(nclasses * ngenes), nrow=ngenes);
rownames(S1) <- rownames(T);
colnames(S1) <- classnames;
# Construction of a 1-step RW kernel
RW <- rw.kernel(W);
```

Then the scores are computed by 5 fold cross-validation repeated 10 times by calling the $multiple.ker.score.cv$ *RANKS* method. Note that the *KNN* score function is used, considering $k = 19$ nearest neighbours:

```
for (i in 1:nclasses)  {
  ind.pos <- which(T[,i]==1);
  # 10-fold CV with 1 step RW kernel
  res<-multiple.ker.score.cv(RW, ind.pos, m=5, p=10,
                  init.seed=1, fun=KNN.score, k=19);
  S1[,i] <- res$av.scores;
  cat("Class ", i, " : ", classnames[i], "\n");
}
```

By simply changing the argument fun you can experiment with other scores, e.g. by setting $fun = eav.score$ we can use the *average score*. Finally we can estimate the performance by computing the Area Under the ROC curve and precision at different levels of recall by calling the corresponding functions of the *PerfMeas* package:

```
# saving scores
save(S1, file="Results/Scores.RANKS.GFP.rda");
# Computing precision at different levels of recall
recall.levels <- seq(from=0.1, to=0.9, by=0.1);
res <- precision.at.multiple.recall.level.over.classes
                  (T, S1, rec.levels = recall.levels);
# Computing AUC
auc <- AUC.single.over.classes(T, S1);
# saving results
save(res, auc, file="Results/AUC.PXR.RANKS.GFP.rda");
```

RANKS is very fast: on a notebook with an Intel i7 2.20 GHz with 4 GB RAM the entire 5 fold cross validation procedure repeated 10 times for all the 168 considered FunCat classes required less than 50 seconds, including also the I/O and the computation of the performance measures.

Note that *RANKS* is basically a ranker, since it provides scores for each gene/node, but can be also used as a classifier through the *RANKS* functions $ker.score.classifier.cv$, $ker.score.classifier.holdout$ and $multiple.ker.score.thresh.cv$, by which an "optimal" threshold is applied to the score to obtain the label associated to each node. The optimal threshold is obtained by internal cross-validation on training data (see the Reference Manual for more details).

Table 1 reports cross-validated results on the FunCat-yeast prediction task, stratified across the 5 levels of the FunCat taxonomy. In particular, level 1 is the highest level, which includes the most general terms, level 2 is less general, till to level 5, which includes the most specific FunCat categories. Results show that *RANKS* methods (the first 3 columns) are competitive with other state of the art network-based algorithm such as GeneMania [23], or the classical random walk and random walk with restart algorithm, as well as with inductive supervised algorithms such as Support Vector Machines.

2.2 Drug repositioning

Drug repositioning consists of predicting novel therapeutic indications for existing drugs. In this context nodes represent drugs and edges their structural or functional similarities. The task consists of scoring and ranking unlabeled nodes/drugs with respect to a given therapeutic indication starting from a small set of labeled drugs/nodes [8, 7, 33].

Here we show how to predict the therapeutic category of drugs according to the annotations provided by DrugBank 3.0 [12] using *RANKS*. We considered 51 Therapeutic Categories (TC) from DrugBank (Table 2), that is the TC having more that 15 drugs annotated. We constructed three drug networks involving 1253 drugs: the first one is based on the similarity of chemical structures according to their SMILES molecular fingerprint; the second is obtained through network projections between drugs and their molecular targets and the third one has been constructed by network projections of the chemical-chemical interactions in the STITCH database [15]. Then we constructed three networks $U1$, $U2$ and $U3$, by progressively integrating the first, the second and the third networks described above (see [29] for more details about the construction of the drug networks).

In the rest of this section we show how to process data through cross-validation techniques using a single call to the high-level function $do.RANKS$ of the *RANKS* package.

After loading *RANKS* and *PerfMeas* packages, we call $do.RANKS$ to perform a 5-fold cross validation ($kk = 5$) using the average score ($score = eav.score$) and a 1-step random walk kernel ($kernel = rw.kernel, p = 1$) to score the drugs with respect to 51 TCs. The network data are in the directory $data$ ($data.dir = "data/"$) and the name of the file of the network data is $U1.rda$ ($data = "U1"$), corresponding to a network constructed by structural similarity between each pair of drugs. In the same way the directory where the labels of the drugs are stored is $data$ ($labels.dir = "data/"$) and the file name of the drug labels is "T" ($labels = "T"$), where $T[i, j] = 1$ if drug i is annotated with the Therapeutic Category j, otherwise $T[i, j] = 0$. The corresponding results (computed scores, as well as AUC results and precision at different recall levels) are stored in the $Results$ directory ($output.dir = "Results/"$):

```
library(RANKS);
library(PerfMeas);
do.RANKS(score=eav.score, kernel=rw.kernel, a=2,  p=1,
         sparsify=TRUE, kk=5, rep=1, seed=0,
         data.dir="data/", labels.dir="data/",
         output.dir="Results/", data="U1", labels="T");
```

Note that this interface is conceived for batch experiments, where both input data and the results are automatically stored in R compressed .rda files. In this way we can experiment with different score functions or kernels by only changing the arguments of the high-level function $do.RANKS$. In a similar way experiments with different data can be easily performed by only changing the input data theirselves, just using only 1 line of code.

The following lines of code show an example of cross-validations with different score functions and kernels, and different input networks:

```
# The same task as above, but using a different network
# U2, constructed by integrating molecular fingerprints
# of  the drugs and drug-target interactions:
do.RANKS(score=eav.score, kernel=rw.kernel, a=2,  p=1,
         sparsify=TRUE, kk=5, rep=1, seed=0,
         data.dir="data/", labels.dir="data/",
         output.dir="Results/", data="U2", labels="T");
# Here also chemical-chemical interaction form STITCH
# have been integrated (data="U3"):
do.RANKS(score=eav.score, kernel=rw.kernel, a=2,  p=1,
         sparsify=TRUE, kk=5, rep=1, seed=0,
         data.dir="data/", labels.dir="data/",
         output.dir="Results/", data="U3", labels="T");
# A different score function, i.e.
# Weighted Sum Linear Decay is used with the same data:
do.RANKS(score=WSLD.score, kernel=rw.kernel, a=2,  p=1,
         sparsify=TRUE, kk=5, rep=1, seed=0,
```

Table 1. Average AUC at the five levels of the FunCat taxonomy. Numbers in boldface refer to the best results for a given level. S_{kNN} stands for K-nearest-neighbours score, S_{AV} stands for average score, S_{NN} nearest-neighbours score, GM GeneMania, RWR random walk with restart, RW random walk till to convergence, $RW2st$ random walk limited to 2 steps, GBA guilt-by-association, and SVM is a linear Support Vector Machine.

level	S_{kNN}	S_{AV}	S_{NN}	GM	RWR	RW	RW 2 st	GBA	SVM	n. class
1	0.8276	0.8069	0.7754	0.7920	0.7466	0.5194	0.7608	0.7006	**0.8752**	16
2	**0.8620**	0.8535	0.8424	0.8370	0.8071	0.4902	0.7389	0.7649	0.7849	49
3	**0.9006**	0.8983	0.8874	0.8807	0.8614	0.5146	0.7712	0.8279	0.8058	65
4	**0.9052**	0.9030	0.8880	0.8860	0.8717	0.5216	0.7888	0.8419	0.7248	30
5	**0.9206**	**0.9206**	0.9164	0.9067	0.9064	0.5733	0.8173	0.8682	0.4954	8

```
        data.dir="data/", labels.dir="data/",
        output.dir="Results/", data="U3", labels="T");
# Here the KNN score with k=19 nearest neighbours
# is used instead:
do.RANKS(score=KNN.score, kernel=rw.kernel, a=2,  p=1,
        sparsify=TRUE, kk=5, rep=1, seed=0,
        data.dir="data/", labels.dir="data/",
        output.dir="Results/", data="U3", labels="T",
        k=19);
# KNN score with k=19 and 2 steps random walk kernel:
do.RANKS(score=KNN.score, kernel=rw.kernel, a=5,  p=2,
        sparsify=TRUE, kk=5, rep=1, seed=0,
        data.dir="data/", labels.dir="data/",
        output.dir="Results/", data="U3", labels="T",
        k=19);
# NN score and 5 step random walk kernel
# and with 10 fold CV:
do.RANKS(score=NN.score, kernel=rw.kernel, a=10,  p=5,
        sparsify=TRUE, kk=10, rep=1, seed=0,
        data.dir="data/", labels.dir="data/",
        output.dir="Results/", data="U3", labels="T");
```

The same experiment with the S_{NN} score but using a leave-one-out technique to assess the generalization performances could be in principle attained by using the same function $do.RANKS$ and by setting $kk = 1253$ (i.e. the total number of nodes/drugs), but a more efficient version is implemented in the *RANKS* package through the function $do.loo.RANKS$:

```
# the same task with loo
do.loo.RANKS(score=NN.score, compute.kernel=TRUE,
       kernel=rw.kernel,  a=10, p=5, sparsify=TRUE,
       norm=FALSE, data="U3", labels="T",
       output.dir="Results/", output.name="drug_rep");
```

RANKS implement also high level functions for cross-validation to experiment with other popular network-based algorithms, such as guilt-by-association, random walk, random walk with restart and label propagation algorithms:

```
# High level function for 5 fold CV with GBA
do.GBA(fun=GBAmax, k=5, filter=TRUE, seed=0,
       data="U3", labels="T");
# High level function for 5 fold CV with random walk
do.RW(tmax=5, eps=1e-10, k=5, filter=TRUE, seed=0,
       data="U3", labels="T");
# 5 fold CV with random walk with restart
do.RWR(gamma=0.2, tmax=20, k=5, filter=TRUE, seed=0,
       data="U3", labels="T");
```

Note that in all cases the cross-validated computed scores, the AUC and precision at fixed recall levels results for each single class and averaged across classes are available as R compressed .rda files in the directory $Results$ (or whatever directory indicated in the argument $output.dir$ of the high-level function $do.RANKS$). For instance the previous call:

```
# KNN score with k=19 and 2 steps random walk kernel:
do.RANKS(score=KNN.score, kernel=rw.kernel, a=5,  p=2,
        sparsify=TRUE, kk=5, rep=1, seed=0,
        data.dir="data/", labels.dir="data/",
        output.dir="Results/", data="U3", labels="T",
        k=19);
```

automatically generates in the directory $Results$ the files:

```
Scores.RW.5step.fTRUE.U3.T.rda   # Score results
AUC.KNN.score19.p2.a5.fTRUE.U3.T.rda # AUC results
PXR.KNN.score19.p2.a5.fTRUE.U3.T.rda # Precision/recall
```

For instance by looking at AUC results:

```
load("AUC.KNN.score19.p2.a5.fTRUE.U3.T.rda");
AUC$av  # AUC averaged across the 51 TC categories
[1] 0.9276313
AUC$per.class  # per class results
# only the first 10 visualized:
     Adrenergic_Agents
     0.9353646
    Adrenergic_alpha.Agonists
     0.9750442
       Analgesics
     0.8287407
      Anti.Allergic_Agents
     0.9627023
     Anti.Bacterial_Agents
     0.9604981
 Adrenergic_Uptake_Inhibitors
0.9887672
  Adrenergic_beta.Antagonists
0.9639088
   Analgesics._Opioid
0.9977624
   Anti.Arrhythmia_Agents
0.9317467
  Anti.HIV_Agents
0.9351045
. . .
```

2.3 Human Phenotype Ontology prediction

The Human Phenotype Ontology (HPO) project [30] provides a comprehensive and well-structured set of more than 10000 terms (classes) that represent human phenotypic abnormalities annotated to more than 7000 hereditary syndromes listed in OMIM, Orphanet and DECIPHER databases [1]. An important computational task is represented by the prediction or ranking of genes with respect to HPO terms [10, 37]. HPO prediction has been recently proposed in the international CAFA2 challenge, and *RANKS* was one of the top-scoring methods participating to the challenge (`http://biofunctionprediction.org/node/8`).

Here we show an application of *RANKS* to HPO term ranking using integrated networks of human genes obtained from two previous

Table 2. DrugBank Therapeutic Categories (TC) with more than 15 drugs considered in the experiments. The first column reports the abbreviated name, the second the full DrugBank name and the third the cardinality of the TC.

Therapeutic categories with more than 15 drugs		
Abbreviated name	**Full DrugBank name name**	**Card.**
Adren.A.	Adrenergic_Agents	26
Adren.In.	Adrenergic_Uptake_Inhibitors	20
Adren.a.	Strategic_alpha.Agonists	23
Adren.b.	Adrenergic_beta.Antagonists	25
Analges.	Analgesics	40
Analg.Op.	Analgesics._Opioid	24
Anti.Aller.	Anti.Allergic_Agents	35
Anti.Arrh.	Anti.Arrhythmia_Agents	42
Anti.Bact.	Anti.Bacterial_Agents	103
Anti.HIV	Anti.HIV_Agents	22
Anti.Inf.A.	Anti.Infective_Agents	29
Anti.Inf.	Anti.Infectives	19
Anti.Ulcer	Anti.Ulcer_Agents	19
Anti.anx.	Anti.anxiety_Agents	35
Anti.infl.	Anti.inflammatory_Agents	49
Antiarr.A.	Antiarrhythmic_Agents	29
Anticonv.	Anticonvulsants	46
Antidysk.	Antidyskinetics	23
Antiemetics	Antiemetics	34
Antifungal	Antifungal_Agents	22
Antihist.	Antihistamines	24
Antihypert.	Antihypertensive_Agents	105
Antimetab.	Antimetabolites	26
Antineopl.	Antineoplastic_Agents	86
Antineopl.H.	Antineoplastic_Agents._Hormonal	18
Antipark.	Antiparkinson_Agents	27
Antipsyc.A.	Antipsychotic_Agents	39
Antipsyc.	Antipsychotics	27
Antiviral	Antiviral_Agents	25
Bronchodil.	Bronchodilator_Agents	33
Ca.Ch.Block.	Calcium_Channel_Blockers	22
Cephalosp.	Cephalosporins	30
Cycloox.Inh.	Cyclooxygenase_Inhibitors	24
Dietary.sup.	Dietary_supplement	47
Diuretics	Diuretics	23
Dopam..Ant.	Dopamine_Antagonists	29
Enzyme.Inh.	Enzyme_Inhibitors	64
GABA.Mod.	GABA_Modulators	26
Glucocort.	Glucocorticoids	21
Hist.H1.Ant.	Histamine_H1_Antagonists	28
Hypnot.Sed.	Hypnotics_and_Sedatives	41
Hypoglyc.	Hypoglycemic_Agents	22
Immunosup.	Immunosuppressive_Agents	20
Micronutr.	Micronutrient	45
Musc.Ant.	Muscarinic_Antagonists	23
Narcotics	Narcotics	22
Penicillins	Penicillins	20
Sympathol.	Sympatholytics	24
Sympathomim.	Sympathomimetics	32
Vasoconstr.	Vasoconstrictor_Agents	25
Vasodilator	Vasodilator_Agents	55

studies [41, 17]. The Functional Interaction (*FI*) network [41] has been constructed by using functional interactions predicted by a Naive Bayes classifier trained on pairwise relationships extracted from Reactome [39] and other curated pathways databases, and from uncurated pairwise relationships derived from physical protein-protein interactions (PPI) in human and other species, from gene co-expression data, proteins domain-domain interactions, protein interactions obtained via biomedical text mining, and Gene Ontology annotations. The network presented in [17] integrates diverse lines of evidence in order to produce a functional human gene network (*HumanNet*) that has then been used in several tests to predict causal genes for human diseases and to increase the power of genome-wide association studies. The most significant difference between *HumanNet* and *FI* networks consists of including in the former functional interactions borrowed from other species (yeast, fly and worm) through comparative genomics techniques.

As an example of application of *RANKS* we integrated the two networks through a simple unweighted integration method, thus obtaining a net with 16505 nodes (genes) and about one million and two hundred thousands edges. We randomly selected 100 HPO terms having more than 20 annotated genes and applied *RANKS* to HPO term ranking and prediction. Please, note that this task requires a computer with 16 GB RAM to be comfortably executed (other machines with less available memory can be used, but in this case it is likely that a certain computational burden will occur, due to memory swapping problems).

After loading the *RANKS* and *PerfMeas* packages, we load the HPO annotations, randomly selecting 100 HPO terms having more than 20 annotated genes:

```
# the network data (net) and the corresponding labels
# (both files in .rda format) are assumed to be in
# the directory data:
net = "hnnet-finet.UA.net";
labels <- "hpo.hnnet.finet.ann.20";
load("data/hpo.hnnet.finet.ann.rda");
# elimination of terms having less than 20 annotations
x <- apply(hpo.hnnet.finet.ann,2,sum);
y <- which (x<=20);
# 5066 terms removed, 1464 terms remained
hpo.hnnet.finet.ann.20 <- hpo.hnnet.finet.ann[,-y];
# random selection of 100 HPO terms
n <- ncol(hpo.hnnet.finet.ann.20);
selected <- sample(1:n, 100);
hpo.hnnet.finet.ann.20.selected <-
                    hpo.hnnet.finet.ann.20[,selected];
save(hpo.hnnet.finet.ann.20.selected,
     file="data/hpo.selected.labels.rda");
labels.sel <- "hpo.selected.labels";
```

With a single call to the high level function $do.loo.RANKS$ we can perform a leave-one-out run on the entire network using the average score and a 1-step random walk kernel; scores, AUC and precision/recall results are automatically stored in .rda files in the current directory:

```
# RW kernel with 1 step
output.dir <- "./";
labels.sel <- "hpo.selected.labels";
do.loo.RANKS(score=eav.score, compute.kernel=TRUE,
         kernel=rw.kernel,  a=2, p=1, sparsify=TRUE,
         data=net, labels=labels.sel,
         output.dir=output.dir, output.name="hpo.sel");
```

The same task, using different combination of kernels, can be easily performed with the following lines of code:

```
# RW kernel with 3 steps
do.loo.RANKS(score=eav.score, compute.kernel=TRUE,
        kernel=rw.kernel,  a=10, p=3, sparsify=TRUE,
        data=net, labels=labels.sel, output.dir=output.dir,
        output.name="hpo.sel");
# linear kernel
do.loo.RANKS(score=eav.score, compute.kernel=TRUE,
    kernel=linear.kernel,  a=2, p=1, sparsify=TRUE,
    data=net, labels=labels.sel, output.dir=output.dir,
```

```
    output.name="hpo.sel.linear");
# usage of the network "as it is" (identity kernel)
do.loo.RANKS(score=eav.score, compute.kernel=TRUE,
    kernel=identity.kernel, p=1, sparsify=TRUE,
    data=net, labels=labels.sel, output.dir=output.dir,
    output.name="hpo.sel.identity");
```

The distribution of the AUC results across the 100 considered HPO terms, for the identity (Id), linear (Lin) and random walk 1-step (RW1) and random walk 3 steps (RW3) kernels are summarized in Fig. 2.

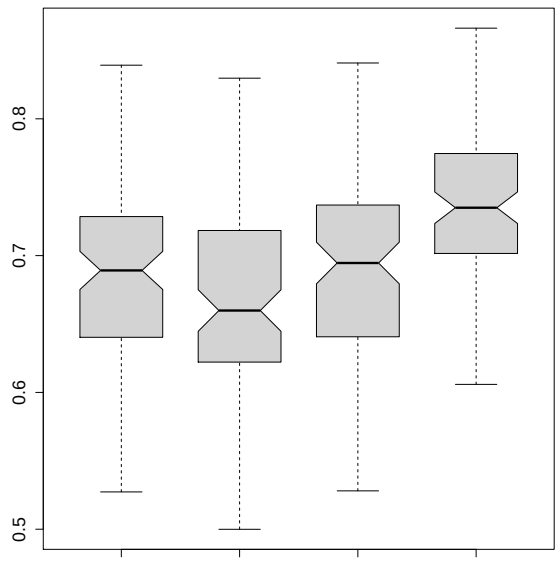

Fig. 2. Boxplots of the AUC results across the randomly selected 100 HPO terms. Id stands for Identity kernel, Lin for linear, RW1 and RW3 for 1 and 3 steps random walk kernel.

3 User defined score functions and kernels

One of the strengths of the *RANKS* is its modularity: it offers an algorithmic scheme where the specific choice of a score function and a kernel leads to a different semi-supervised learning algorithm (Section 1).

RANKS offers a set of different score functions implemented in the package: Nearest-Neighbour (S_{NN}), K-Nearest-Neighbour (S_{KNN}) and average (S_{AV} score) – Section 1), as well as Weighted Sum with Linear Decay S_{AV} score (S_{WSLD}), which represents a generalization of the score introduced in [16]. Moreover several kernels are just implemented in the library, such as the Cauchy, the Laplacian, the Gaussian, the inverse multiquadric, the linear and Polynomial and the random walk kernels (Section 1.4). In addition, a user can easily devise and implement her/his own score functions and kernels, which can be used by any high-level function of the library as they were in fact built-in.

If you would like to implement your own score function (e.g. the $My.score$ function), you should implement two S4 methods:

```
single.My.score(K, x, x.pos)
My.score(K, x, x.pos, norm = TRUE)
```

whose arguments are:

```
K: a matrix. It must be a kernel matrix or a symmetric
matrix  expressing the similarity between nodes
x: integer. Index corresponding to the element of the K
matrix for which the score must be computed
x.pos: vector of integer. Indices of the positive
elements of the K matrix
```

The $single.My.Score$ is expected to return the score of a single node, while $My.score$ should return a vector embodying the scores of all nodes of the network. For the sake of clarity, let us provide the declaration of the method for $single.My.score$:

```
# generic function
setGeneric("single.My.score",
   function(RK, x, x.pos) standardGeneric("single.My.Score"));
# method for class matrix
setMethod("single.My.score", signature(K="matrix"),
  function(K, x, x.pos) { ... })
# method for class graph - package graph
setMethod("single.My.score", signature(K="graph"),
  function(K, x, x.pos) { ... })
```

Of course the body of the functions should be filled with the code implementing the method, and parameters of the score can be added as additional arguments. For more details, please see the reference manual of the package *RANKS*.

Adding novel kernel functions is even easier. For instance a user-defined kernel $My.kernel$ can be added by implementing a function as follows:

```
My.kernel <- function(W, a) { ... }
```

where

```
W: numeric matrix.
Rows are examples and columns are features
a: parameter for the kernel
(if necessary other kernel parameters can be added).
```

$My.kernel$ should return a square symmetric kernel matrix representing the similarities between the examples (rows of W), as specified in the $My.kernel$ function.

Once a new score function and/or a new kernel function have been defined as specified above, they can be used in any high-level *RANKS* functions where a generic score or kernel function can be used. For instance, if we consider the high level function $do.RANKS$, to perform a 5-fold cross-validation with the network "net.rda" and the labels "labels.rda", both stored in the directory "data", one should call $My.score$ and $My.kernel$ as follows:

```
do.RANKS(score = My.score, kernel = My.kernel, kk = 5,
data.dir = "data/", labels.dir = "data/",
output.dir = "Results/", data="net", labels="labels")
```

The following additional example considers a hold-out classification of a kernel matrix K using the user-defined $My.score$ score function:

```
ker.score.classifier.holdout(K, ind.pos, ind.test,
                             fun = My.score)
```

As already pointed out, custom score functions and kernels can be passed as arguments to any high level function of the software library, such as $do.RANKS$ or $do.loo.RANKS$.

References

[1]J. Amberger, C. Bocchini, and A. Amosh. A new face and new challenges for Online Mendelian inheritance in Man (OMIM). *Hum. Mutat.*, 32:564–7, 2011.

[2]S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide*

Web 7, pages 107–117, Amsterdam, The Netherlands, 1998. Elsevier Science Publishers.

[3]H.N. Chua, W. Sung, and L. Wong. An efficient strategy for extensive integration of diverse biological data for protein function prediction. *Bioinformatics*, 23(24):3364–3373, 2007.

[4]M. Frasca, A. Bertoni, M. Re, and G. Valentini. A neural network algorithm for semi-supervised node label learning from unbalanced data. *Neural Networks*, 43:84–98, 2013.

[5]M. Frasca, A. Bertoni, and G. Valentini. Unipred: Unbalance-aware network integration and prediction of protein functions. *Journal of Computational Biology*, 22(12):1057–1074, 2015.

[6]M. Frasca and G. Valentini. COSNet: an R package for label prediction in unbalanced biological networks *Neurocomputing*, doi:10.1016/j.neucom.2015.11.096, 2016.

[7]A. Gottlieb, G. Stein, E. Ruppin, and R. Sharan. PREDICT, a method for inferring novel drug indications with application to personalized medicine. *Molecular Systems Biology*, 7(496), 2011.

[8]F. Iorio, R. Bosotti, E. Scacheri, P. Mithbaokar, R. Ferriero, L. Murino, R. Tagliaferri, N. Brunetti-Pierri, A. Isacchi, and D. di Bernardo. Discovery of drug mode of action and drug repositioning from transcriptional responses. *PNAS*, 107(33):14621–14626, 2010.

[9]Jiang, Y. *et al.*. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *ArXiv e-prints*, article ID: 1601.00891, 2016.

[10]I Kahanda, C. Funk, K. Verspoor, and A. Ben-Hur. PHENOstruct: prediction of human phenotype ontology terms using heterogeneous data sources. *F1000Research*, 4(259), 2015.

[11]H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for graphs. In B. Scholkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 155–170. MIT Press, Cambridge, MA, 2004.

[12]C. Knox, V. Law, T. Jewison, P. Liu, S. Ly, A. Frolkis, A. Pon, K. Banco, C. Mak, V. Neveu, Y. Djoumbou, R. Eisner, A.C. Guo, and D.S. Wishart. DrugBank 3.0: a comprehensive resource for 'omics' research on drugs. *Nucleic Acids Res.*, 39(Jan):D1035–41, 2011.

[13]S Kohler, SC Doelken, CJ Mungall, S Bauer, HV Firth, I Bailleul-Forestier, GC Black, DL Brown, M Brudno, J Campbell, DR FitzPatrick, JT Eppig, AP Jackson, K Freson, M Girdea, I Helbig, JA Hurst, J Jahn, LG Jackson, AM Kelly, DH Ledbetter, S Mansour, CL Martin, C Moss, A Mumford, WH Ouwehand, SM Park, ER Riggs, RH Scott, S Sisodiya, S Van Vooren, RJ Wapner, AO Wilkie, CF Wright, AT Vulto-van Silfhout, N de Leeuw, BB de Vries, NL Washingthon, CL Smith, M Westerfield, P Schofield, BJ Ruef, GV Gkoutos, M Haendel, D Smedley, SE Lewis, and PN Robinson. The human phenotype ontology project: linking molecular biology and disease through phenotype data. *Nucleic Acids Research*, 42((Database issue)):D966–74, 2014.

[14]I.R. Kondor and J.D. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 315–322, 2002.

[15]M. Kuhn, D. Szklarczyk, A. Franceschini, M. Campillos, C. von Mering, L.J. Jensen, A. Beyer, and P. Bork. STITCH 2: an interaction network database for small molecules and proteins. *Nucleic Acids Res.*, 38(Jan):D552–6, 2010.

[16]I. Lee, B. Ambaru, P. Pranjali-Thakkar, E.M. Marcotte, and S.Y. Rhee. Rational association of genes with traits using a genome-scale gene network for arabidopsis thaliana. *Nat. Biotechnol.*, 28:149–156, 2010.

[17]I Lee, UM Blom, PI Wang, JE Shim, and EM Marcotte. Prioritizing candidate disease genes by network-based boosting of genome-wide association data. *Genome Res.*, 21:1109–1121, 2011.

[18]L. Lovasz. Random Walks on Graphs: a Survey. *Combinatorics, Paul Erdös is Eighty*, 2:1–46, 1993.

[19]A. Ma'ayan. Network integration and graph analysis in mammalian molecular systems biology. *IET Syst. Biol.*, 2(5):206–221, 2008.

[20]M.L. Mayer and P. Hieter. Protein networks - guilt by association. *Nature Biotechnology*, 18(12):1242–1243, 2000.

[21]M. Mesiti, M. Re, and G. Valentini. Think globally and solve locally: secondary memory-based network learning for automated multi-species function prediction. *GigaScience*, 3:5, 2014.

[22]A. Mitrofanova, V. Pavlovic, and B. Mishra. Prediction of protein functions with gene ontology and inter-species protein homology data. *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 8(3):775–784, 2011.

[23]S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, 9(S4), 2008.

[24]F. Napolitano, Y. Zhao, V.M. Moreira, R. Tagliaferri, J. Kere, M. D'Amato, and D. Greco. Drug repositioning: a machine-learning approach through data integration. *J Cheminform.*, Jun 22;5(1):30. doi: 10.1186/1758-2946-5-30, 2013.

[25]S. Oliver. Guilt-by-association goes global. *Nature*, 403:601–603, 2000.

[26]P. Radivojac et al. A large-scale evaluation of computational protein function prediction. *Nature Methods*, 10(3):221–227, 2013.

[27]M. Re, M. Mesiti, and G. Valentini. A Fast Ranking Algorithm for Predicting Gene Functions in Biomolecular Networks. *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 9(6):1812–1818, 2012.

[28]M. Re and G. Valentini. Cancer module genes ranking using kernelized score functions. *BMC Bioinformatics*, 13(Suppl 14/S3), 2012.

[29]M. Re and G. Valentini. Network-based Drug Ranking and Repositioning with respect to DrugBank Therapeutic Categories. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(6):1359–1371, 2013.

[30]P.N. Robinson, S. Kohler, S. Bauer, D. Seelow, D. Horn, and S. Mundlos. The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am. J. Hum. Genet.*, 83:610–615, 2008.

[31]A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Guldener, G. Mannhaupt, M. Munsterkotter, and H.W. Mewes. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545, 2004.

[32]B. Schölkopf, K. Tsuda, and J.P. Vert. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 2004.

[33]M. Sirota et al. Discovery and preclinical validation of drug indications using compendia of public gene expression data. *Sci. Transl. Med.*, 96(3):96–77, 2011.

[34]A.J. Smola and I.R. Kondor. Kernel and regularization on graphs. In B. Scholkopf and M.K. Warmuth, editors, *Proc. of the Annual Conf. on Computational Learning Theory*, Lecture Notes in Computer Science, pages 144–158. Springer, 2003.

[35]G. Valentini. True Path Rule hierarchical ensembles for genome-wide gene function prediction. *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 8(3):832–847, 2011.

[36]G. Valentini and N. Cesa-Bianchi. Hcgene: a software tool to support the hierarchical classification of genes. *Bioinformatics*, 24(5):729–731, 2008.

[37]G. Valentini, S. Kohler, M. Re, M. Notaro, and P.N. Robinson. Prediction of human gene - phenotype associations by exploiting the hierarchical structure of the human phenotype ontology. In *IWBBIO 2015 (3rd International Work-Conference on Bioinformatics and Biomedical Engineering)*, volume 9043 of *Lecture Notes in Bioinformatics*, pages 66–77. Springer, 2015.

[38]G. Valentini, A. Paccanaro, H. Caniza, A. Romero, and M. Re. An extensive analysis of disease-gene associations using network integration and fast kernel-based gene prioritization methods. *Artificial Intelligence in Medicine*, 61(2):63–78, 2014.

[39]I Vastrik, P D'Eustachio, E Schmidt, G Gopinath, D Croft, B de Bono, M Gillespie, B Jassal, S Lewis, L Matthews, G Wu, E Birney, and L Stein. Reactome: a knowledge base of biologic pathways and processes. *Genome Biol.*, 8:R39, 2007.

[40]A. Vazquez, A. Flammini, A Maritan, and A Vespignani. Global protein function prediction from protein-protein interaction networks. *Nature Biotechnology*, 21:697–700, 2003.

[41]G Wu, X Feng, and L Stein. A human functional protein interaction network and its application to cancer data analysis. *Genome Biology*, 11:R53, 2010.