regNet: An R package for network-based propagation of gene expression alterations

June 9, 2017

Package: regNet (https://github.com/seifemi/regNet)
Type: R package
Version: 1.0
Maintainer: Michael Seifert (michael.seifert@tu-dresden.de)
Description: regNet utilizes gene expression and copy number data to learn regulatory networks for the quantification of potential impacts of individual gene expression alterations on user-defined target genes via network propagation.
Depends: R (>= 2.15), glmnet, Matrix, lars, covTest

Data sets: http://doi.org/10.5281/zenodo.580600
Underlying algorithms: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-1058-1

Contents

1 regNet installation

To install the R package regNet, the R package 'devtools' is required. regNet further depends on the R packages 'glmnet', 'Matrix', 'lars', and 'covTest'. We assume that these packages are installed on your system. Download the R package regNet from https://github.com/seifemi/regNet and use the following R commands to install regNet globally or locally on your system.

```
library( devtools )
#Set path to the parent directory into which regNet was downloaded
setwd( regNetParentDir )

#Option 1: Global installation as root into the generally used R package system folder
install( "regNet" )

#Option 2: Local installation as standard user into a user-specific R package folder
.libPaths( c( .libPaths(), "/home/seifert/LocalRLibs/" ) )
install( pkg = "regNet", args = c( '-library="/home/seifert/LocalRLibs/"' ) )
```

regNet should now be installed on your system.

2 regNet architecture

regNet uses a standardized folder structure (Fig. 1) to simplify the general usage. regNet is further divided into four main modules: (i) the data loader module, (ii) the network inference module, (ii) the network prediction module, and (iv) the network propagation module. The most important R functions of each regNet module are listed in Tab. 1.

2.1 Standardized folder structure

regNet uses a fixed folder structure (Fig. 1) to enable an uncomplicated storage and loading of an inferred network, network predictions and network propagation results. A user-defined project name specifies the head folder that always contains the following subfolders.

- The subfolder `NetworkConnectivity` is used to save network connectivity tables representing links contained in a learned network.
- The subfolder `NetworkModel` is used to save a network. Since network inference is usually very time-consuming, regNet allows to split the network inference into sub-network inference tasks. The sub-networks are saved under `NetworkModel/SingleJobs` and the corresponding runtime of each sub-network inference job is saved under `NetworkModel/SingleJobs/Runtimes`. After all sub-network inference tasks have been finished, the individual sub-networks are combined to a global network that is saved under `NetworkModel/WholeNetwork`.
- The subfolder `NetworkPredictions` is used to save results of network-based predictions of gene expression levels.

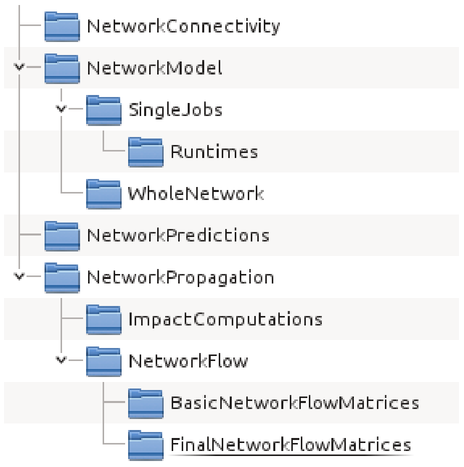

Figure 1: Basic folder structure. Basic folder structure created and used by regNet. See Section 2.1 for more details about the functions of individual folders.

- The subfolder `NetworkPropagation` is used to save the results of network propagation studies. The subfolder `NetworkPropagation/ImpactComputations` saves results of the network propagation algorithm (e.g. impact of tumor-specific gene copy number mutations on user-defined signature genes). Such impact computations require a basic network flow matrix saved under `NetworkPropagation/NetworkFlow/BasicNetworkFlowMatrices` and a resulting final network flow matrix stored under `NetworkPropagation/NetworkFlow/FinalNetworkFlowMatrices`.

Generally, regNet will always report which files have been saved into specific folders as long as individual function calls are made with `output = TRUE`, which is default. The standardized folder structure (Fig. 1) is created using the regNet R function `createBasicFolderStructure`.

2.2 Data loader

regNet expects gene expression profiles and corresponding gene copy number profiles as input data for network inference, network predictions and network propagation. regNet handles tab-delimited data sets with a fixed column-structure (Tab. 2): `Gene, Chromosome, Location, Sample_1, ..., Sample_N`.

- The `Gene` column contains the unique gene identifiers (e.g. gene names or gene identifiers).
- The `Chromosome` column contains the corresponding chromosome names.
- The `Location` column contains the corresponding chromosome-specific physical locations of genes (e.g. start position or mid point of genes).
- The measurement values of genes (expression levels or copy numbers) belonging to a specific sample `n` must be provided in the corresponding `Sample_n` column.

Generally, genes must be sorted according to their chromosomal locations. Separate tab-delimited tables for gene expression and gene copy number data are required. Both tables

Data loader
createBasicFolderStructure
getProjectPath
loadGeneExpressionAndCopyNumberDataSet
testDataSetCompatibility
makeTestDataSetCompatible
loadNetworkWithFilteringForSignificantPredictors
Network inference
learnNetwork_ParallelComputation
combineSingleJobs
getNetworkConnections
determineRandomNetworkWithFilteringForSignificantPredictors
Network prediction
predictGeneExpression
Network propagation: Cohort-specific absolute impacts
computeNetworkFlow_CohortSpecificAbsoluteImpacts
getAverageImpacts_CohortSpecificAbsoluteImpacts
getImpacts_CohortSpecificAbsoluteImpacts
getRawImpacts_CohortSpecificAbsoluteImpacts
Network propagation: Patient-specific absolute impacts
computeNetworkFlow_PatientSpecificAbsoluteImpacts
getAverageImpacts_PatientSpecificAbsoluteImpacts
getImpacts_PatientSpecificAbsoluteImpacts
getRawImpacts_PatientSpecificAbsoluteImpacts
Network propagation: Patient-specific relative impacts
computeNetworkFlow_PatientSpecificRelativeImpacts
getAverageImpacts_PatientSpecificRelativeImpacts
getImpacts_PatientSpecificRelativeImpacts
getRawImpacts_PatientSpecificRelativeImpacts

Table 1: regNet functions. Summary of most relevant regNet functions. See Sections 2 and 3 for more details to individual functions or utilize the R help of regNet.

Gene	Chromosome	Location	Sample_1	...	Sample_N
CCNL2	1	1321091	-1.73	...	-2.25
CHD5	1	6161853	-1.23	...	-1.23
ATAD2B	2	23971534	-0.98	...	-1.37
SATB2	2	200134223	3.79	...	1.87
THRB	3	24158651	1.19	...	1.11

Table 2: regNet data input format. Standard structure of an input data table of gene expression or gene copy number data processed by regNet.

must use the identical sample identifiers in the same order. Missing values are not allowed and should be replaced by zero beforehand, which ensures that affected entries do not make a contribution to network inference, network prediction or network propagation. The tabulator is used to separate the columns. A gene expression and its corresponding gene copy number data set can be loaded with the function `loadGeneExpressionAndCopyNumberDataSet`.

Additionally, data sets can be loaded and further transformed to fulfill constraints set by the other modules or learned networks to ensure correct analyses. This can be done via the functions `testDataSetCompatibility` and `makeTestDataSetCompatible`. This allows to make a test data set from a different source compatible with an initially used training data set by removing additional genes and by artificially adding missing genes with measurement values of zero (no contribution to analysis). This enables to quickly test a learned network on other data sets.

2.3 Network inference

regNet splits the global network inference problem into independent gene-specific sub-network inference tasks. We assume for each gene that its expression level can be predicted by a linear combination of its gene-specific copy number and the expression levels of other potential regulator genes. A detailed description of the underlying mathematical model and in-depth validation studies were done in [1]. Lasso (least absolute shrinkage and selection operator) regression [2] followed by a significance test for lasso [3] is used to learn for each gene those predictors (gene copy number and/or expression levels of other genes) that best predict the expression level of the gene in a given data set. This is implemented using the R packages glmnet [4], lars [5] and covTest [6]. For regNet, we modified the function covTest to avoid the implicit rounding of p-values to four decimals. We now further standardly provide FDR-adjusted p-values [7].

Sequential network inference can be very time-consuming. Thus, regNet can consider independent blocks of sub-network inference problems in parallel (or semi-sequentially) using a user-defined number of compute cores to account for the available hardware. Sub-networks are computed using the function `learnNetwork_ParallelComputation`. Created output files are saved by regNet into the standardized folder structure under `NetworkModel/SingleJobs`. Individual sub-networks are combined to a global network using the function `combineSingleJobs`. The resulting network file is saved by regNet into the folder `NetworkModel/WholeNetwork`.

After the computation of a global network is finished, a network connectivity table can be created using the function `getNetworkConnections`. A specific network instance can be loaded by the function `loadNetworkWithFilteringForSignificantPredictors`. This is usually not needed, because regNet loads required networks automatically. A random network instance can be created using the function `determineRandomNetworkWithFilteringForSignificantPredictors` computing a random degree-preserving network permutation of the original network. This provides basics for comparisons to random baseline models of identical complexity using the network prediction and propagation modules.

All generated files are saved into the standardized folder structure using pre-defined file-naming conventions. regNet reports the exact locations and names of created files by default.

2.4 Network prediction

regNet uses an inferred network to predict gene expression levels followed by the computation of correlations between predicted and originally measured expression levels of genes in a given data set. This is is done using the function `predictGeneExpression`. The created output file is reported by regNet and saved into the standardized folder `NetworkPredictions`. Different instances of a network can be analyzed by setting a global FDR-adjusted p-value cutoff to only consider the most relevant regulatory links and by defining a global threshold to remove local chromosomal regulators up- and downstream of each gene, which may only represent the copy number state of nearby genes instead of potential regulatory dependencies. The gene-specific correlations are further tested for their significance of being greater than zero enabling to further evaluate the predictive power of the network. This is important to test whether a network can be used to analyze independent data, to further analyze the prediction quality of individual genes, and to integrate the quality of the predictions of individual genes into the impact computations via the network propagation module. All these different regNet features also work for randomized networks enabling comparisons to baseline models.

2.5 Network propagation

regNet implements a recently proposed algorithm that quantifies for a given data set the impact of individual regulator genes on all other genes utilizing an inferred network. A detailed description of this network propagation algorithm and in-depth validation studies are provided in [1]. This algorithm quantifies for each gene pair (a, b) the direct and indirect contribution of gene a on the expression of gene b under consideration of all existing network paths from a to b, the prediction quality of individual genes along the paths, and possibly existing feedback loops.

Impact computations based on a given network are possible for a whole patient cohort or individually for each patient-specific tumor sample. Patient-specific impact computations enable the user to gain absolute and relative contributions of each gene pair. Absolute impacts allow to identify those regulator genes that have the greatest impact on specific genes of interest, whereas relative impacts further account for opposed contributions of genes acting as inhibitors or activators. Generally, such impact computations allow for example to quantify the impact of patient-specific gene copy number mutations on clinical relevant signature genes (e.g. patient survival, treatment response, signaling or metabolic pathways).

These different impact computations are realized by specific regNet functions. Initially, an impact matrix must be computed. A cohort-specific impact matrix is computed by the function `computeNetworkFlow_CohortSpecificAbsoluteImpacts`. Patient-specific impact matrices are computed by the functions `computeNetworkFlow_PatientSpecificAbsoluteImpacts` or `computeNetworkFlow_PatientSpecificRelativeImpacts`. Next, the computed impact matrix allows to determine the impacts of source on target genes. The average impact of each source gene on all target genes is determined by the approach-specific `getAverageImpacts_CohortSpecificAbsoluteImpacts`, `getAverageImpacts_PatientSpecificAbsoluteImpacts`, and `getAverageImpacts_PatientSpecificRelativeImpacts` functions. Additionally, individual impacts of each source gene on each target gene are determined by the `getImpacts_CohortSpecificAbsoluteImpacts`, `getImpacts_PatientSpecificAbsoluteImpacts`, and `getImpacts_PatientSpecificRelativeImpacts`, or the `getRawImpacts_CohortSpecificAbsoluteImpacts`, `getRawImpacts_PatientSpecificAbsoluteImpacts`, and `getRawImpacts_PatientSpecificRelativeImpacts` functions.

Standardly, regNet scales the impact entries of each column of an impact matrix into proportions. This scaling can be avoided by working with raw impacts. Further, regNet can use pre-computed network prediction qualities (correlations between predicted and measured expression levels) enabling impact computations for data sets that contain too few samples for a robust estimation of the prediction quality required for network propagation. All these different functions can also be applied to random networks. Again, all generated files follow pre-defined naming conventions. These files are saved into the standardized folder structure (`NetworkPropagation/...`) and the exact locations and names of these files are reported.

3 regNet code usage examples

To provide a detailed guideline of individual R function calls along with direct feedback from regNet, we compiled a small data set using transcription factors (TFs) with altered expression levels between pilocytic astrocytomas and diffuse astrocytomas [8]. This data set enables to study the potential activity and interplay of TFs distinguishing different astrocytoma grades. After establishing the project basics, we show how to use regNet (i) to learn a gene regulatory network, (ii) to make network-based predictions, and (iii) to perform network-based impact computations. We further discuss revealed major regulators in the context of the existing literature. Importantly, this small study requires less than ten minutes on a standard computer.

Generally, regNet has intensively been tested under Linux. regNet also runs under MS Windows, but the user must ensure that the absolute path lengths to files do not get longer than 255 characters, which can be reached by using a shorter project path and shorter file names. All utilized data sets are available online at http://doi.org/10.5281/zenodo.580600 (AstrocytomaGrades.zip) and the R code is available online at https://github.com/seifemi/regNet (basicCodeUsageExamples.R).

We assume that regNet is installed on your system (see Section 1 for installation details). To test the following R code examples, we assume that the R package regNet is loaded into the R console. We further assume that the path `/home/[user]/regNet/AstrocytomaGrades/` exists. Note that you have to define the variable `user` in R (e.g. `user = "seifert"`).

3.1 Create standardized folder structure

regNet uses a fixed folder structure (Fig. 1) to ease storage and loading of files. We call our project `MyFirstNetwork` and create the corresponding standardized folder structure under this path using the following R code.

```
output = TRUE
projectName = "MyFirstNetwork"
path = paste0( "/home/", user, "/regNet/AstrocytomaGrades/" )

#Create basic folder structure
projectPath = createBasicFolderStructure( projectName = projectName, path = path,
   output = output )
```

The function `createBasicFolderStructure` creates the standardized folder structure under `/home/seifert/AstrocytomaGrades/MyFirstNetwork/` and returns the corresponding project path. Further, all created subfolders are shown as standard output in the R console.

3.2 Load data set

regNet requires gene expression and corresponding gene copy number profiles of samples as input data for network inference, network-based prediction of gene expression levels and for network-based propagation of impacts of gene copy number mutations. The basic structure of individual data files that can be handled by regNet is shown in Tab. 2. Considering transcription factors with altered gene expression levels comparing astrocytomas in children to those in adults, the corresponding gene expression and gene copy number data can be loaded using the following R code.

```
geneExpressionFile = "AS_SignatureTFs_ExpressionLevels.txt"
geneCopyNumberFile = "AS_SignatureTFs_CopyNumbers.txt"

loadPath = paste0( "/home/", user, "regNet/AstrocytomaGrades/Data/" )


#Load data set
data = loadGeneExpressionAndCopyNumberDataSet( geneExpressionFile =
   geneExpressionFile, geneCopyNumberFile = geneCopyNumberFile, path = loadPath )
```

The function `loadGeneExpressionAndCopyNumberDataSet` loads both data sets. The returned data object is a list with gene copy number profiles `U`, gene expression profiles `Y`, gene identifiers `genes`, corresponding chromosomes `chr` and gene location information `loc`. This data object can now be used as input data for regNet to perform network inference.

3.3 Network inference

3.3.1 Learning of a gene regulatory network

regNet is used to learn a regulatory network from gene expression and gene copy number data of 124 different astrocytoma samples containing measurements for 171 transcription factors (TFs). The global network inference task is divided into four sub-network inference tasks. Each of these sub-networks is computed separately. regNet automatically divides the 171 genes into four nearly equally-sized gene blocks. Each of these blocks represents a sub-network inference task. Since the network inference for this case study is fast (e.g. less than ten minutes on a standard computer), we perform it sequentially for simplicity. Network inference is done using the following R code.

```
networkName = "AS_SignatureTFs"
totalNumberOfJobs = 4

#Learn sub-networks
for( i in 1:totalNumberOfJobs )
{
   learnNetwork_ParallelComputation( data = data, networkName = networkName,
     cores = totalNumberOfJobs, job = i, path = projectPath, nfolds = 10,
     cvReplicates = 10 )
}
```

The sub-networks computed by the function `learnNetwork_ParallelComputation` are saved into the folder `NetworkModel/SingleJobs` along with the runtimes of individual jobs under `NetworkModel/SingleJobs/Runtimes`. It is also possible to modify the standard settings `nfold=10` (ten-fold cross-validation) with `cvReplicates=10` with ten repeats per gene used for lasso to obtain gene-specific predictors. Additionally, FDR-adjusted p-values of gene-specific predictors are computed automatically and can later be used to analyze different network instances by focusing on the most relevant links.

Importantly, when more genes and samples are considered, network inference cannot be done without using parallel computations on a high performance compute server. Thus, one typically has to replace the for-loop with a function call that enables the parallel execution of individual jobs. This can for example be done by writing a bash script that starts individual sub-network inference tasks on a separate compute core or by writing a wrapper function to use the R function `mclapply` from the R core package 'parallel'.

Next, after the sub-network inference tasks have been computed, the four individual sub-networks have to be combined to obtain a global network. This is done using the following R code.

```
#Combine sub-networks
combineSingleJobs( networkName = networkName, cores = totalNumberOfJobs,
   path = projectPath )
```

The function `combineSingleJobs` realizes the integration of the sub-networks and saves the resulting global network file and the corresponding network statistics under `Network-Model/WholeNetwork`. regNet always automatically loads a specific network if required, but one can also load a network using the function `loadNetworkWithFilteringForSignifi-cantPredictors`.

3.3.2 Computation of a network connectivity table

Further, a corresponding network connectivity table representing learned links between genes and genes and their underlying copy number can be created. This is done using the following

R code.

```
#Create network connectivity table
getNetworkConnections( networkName = networkName, pValCutoff = 0.01,
  localGeneCutoff = 0, path = projectPath )
```

The function `getNetworkConnections` creates a connectivity table and saves it into the folder `NetworkConnectivity`. The parameter `pValCutoff` enables to select the most relevant links. All learned links are listed using `pValCutoff = 1`. Since this is not a genome-wide study, no spurious local gene-specific regulators need to be removed (`localGeneCutoff = 0`).

Considering the resulting connectivity table, we find that the learned network contains more putative activator than repressor links between TFs (Fig. 2a). Only few TFs have more than four outgoing links to other TFs (Fig. 2b). Some of these major regulators are known to be involved in the development of the central nervous system (PAX6, THRB, TBR1) [9]. Other major regulators are known to act on cell proliferation (MEOX2), apoptosis (CCNL2), and histone acetylation (RBBP4) [9].

3.4.1 Computation of a random network instance

Finally, a random network instance of the learned network can be computed using the following R code.

```
#Create random network instance
determineRandomNetworkWithFilteringForSignificantPredictors( networkName =
networkName, pValCutoff = 0.01, randomNetworkInstance = 1, path = projectPath )
```

The function `determineRandomNetworkWithFilteringForSignificantPredictors` loads a learned network and creates a corresponding degree-preserving random network permutation of same complexity as the loaded network. The obtained random network instance is saved under `NetworkModel/WholeNetwork` using a pre-defined file-naming convention with random network name prefix `paste0( "RandomNetwork_", randomNetworkInstance, "_PValueCutoff_", pValCutoff, "_BasedOn_", networkName )`. Generally, random network instances are compatible with all other regNet modules and can therefore be used to generate baseline results for network predictions and network propagation.

3.5 Network-based predictions

regNet can use a learned network to predict expression levels of genes followed by the computation of correlations between predicted and originally measured gene expression levels. Network-based prediction is done using the function `predictGeneExpression`. This requires that the test data set is compatible with the data set that was used to learn the network. Thus, all genes that were present in the training data set must be also be present in the test data set. regNet offers the function `testDataSetCompatibility` to test for compatibility.

Further, regNet can make a test data set compatible with the training data set by removing additional genes and by artificially adding missing genes with measurement values of zero (no contribution to prediction) to the test data set using the function `makeTestDataSetCompatible`.

3.4 Figure 2: Astrocytoma network characteristics

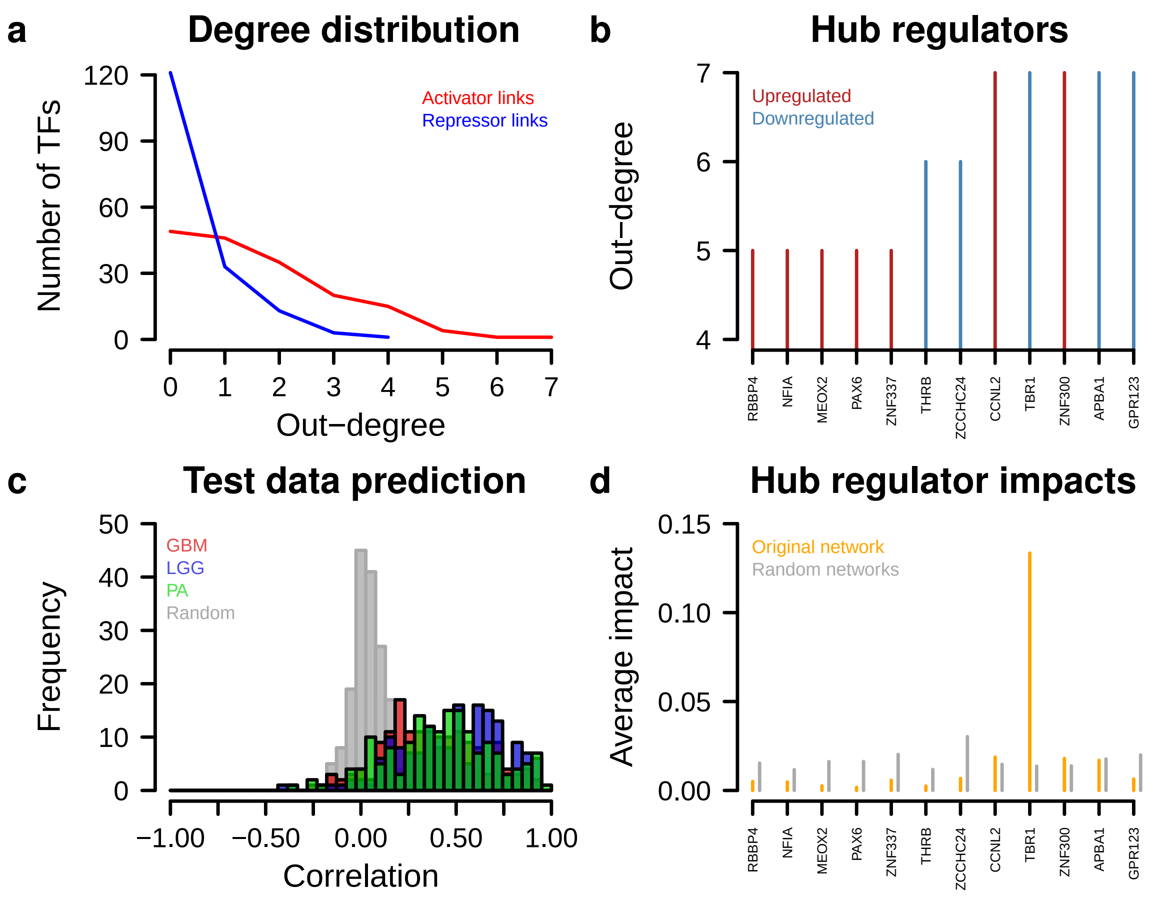

Figure 2: Astrocytoma network characteristics. Characteristics of the network distinguishing pilocytic from diffuse astrocytomas. Only network links with FDR-adjusted p-values equal or less than 0.01 were considered. **a,** Out-degree distribution of the learned network considering activator (red) and repressor links (blue). **b,** Major regulators of the learned network. Genes with higher average expression levels in diffuse astrocytomas than in pilocytic astrocytomas are shown in red. Genes with lower average expression are shown in blue. **c,** Predictive power of the learned network on three independent astrocytoma test cohorts (GBM [10], LGG [11], PA [12]) in comparison to corresponding average predictions obtained by ten random networks of the same complexity (Random). The learned network reaches significantly better predictions than random networks ($P < 6.75 \times 10^{-35}$, Mann-Withney U-Test). **d,** Average impacts of major regulators on other network genes comparing the learned network (orange) to 100 random networks of the same complexity (grey). CCNL2, TBR1 and ZNF300 reach impacts that are significantly greater than expected by chance (Bonferroni-adjusted p-values $P < 1.47 \times 10^{-4}$, Wilcoxon signed-rank tests).

3.5.1 Network-based predictions on independent test data

In the following, we demonstrate the usage of the different functions by using the learned astrocytoma network to evaluate its predictive power for an independent astrocytoma test data set. We therefore use glioblastoma (GBM) data from [10]. GBMs represent the most aggressive type of astrocytomas [13]. Corresponding network-based predictions are done using the following R code.

```
#Load test data set
geneExFile = "TCGA_GBM_ExpressionLevels.txt"
geneCnFile = "TCGA_GBM_CopyNumbers.txt"
loadPath = paste0( "/home/", user, "regNet/AstrocytomaGrades/Data/" )

gbmData = loadGeneExpressionAndCopyNumberDataSet( geneExpressionFile = geneExFile,
   geneCopyNumberFile = geneCnFile, path = loadPath )

#Make test and training data set compatible
print( testDataSetCompatibility( testDataSet = gbmData, trainDataSet = data ) )

gbmTFData = makeTestDataSetCompatible( testDataSet = gbmData, trainDataSet = data )

print( testDataSetCompatibility( testDataSet = gbmTFData, trainDataSet = data ) )


#Network prediction
networkName = "AS_SignatureTFs"
dataSetName = "TCGA_GBM_SignatureTFs"

predictGeneExpression( data = gbmTFData, dataSetName = dataSetName, networkName =
   networkName, pValCutoff = 0.01, localGeneCutoff = 0, path = projectPath )
```

First, a genome-wide test data set of GBM gene expression and copy number data is loaded. Next, regNet is used to make this test data set compatible with the training data set that we used before to learn the network. This is done using the function `makeTestDataSetCompatible` that returns a transformed data set. This data set contains the gene expression levels and gene copy numbers of the different glioblastoma samples for transcription factors that were present in the training data set. Finally, the function `predictGeneExpression` is used to compute the correlations between predicted and originally measured gene expression levels for transcription factors in the GBM test data set using the previously learned network. The resulting gene-specific correlations and corresponding p-values that quantify if specific correlations are significantly greater than zero are saved into the folder `NetworkPredictions`. One can further change the p-value cutoff (`pValCutoff`) to increase or reduce the number of considered links. Since this is not a genome-wide study, it would not make sense to modify the local gene cutoff (`localGeneCutoff`) here.

3.5.2 Comparison of predictions to random network instances

In analogy, the previously created random network instance can be utilized to predict the gene expression levels of GBMs using the following R code.

```
#Random network prediction
randomNetworkName = "RandomNetwork_1_PValueCutoff_0.01_BasedOn_AS_SignatureTFs"
dataSetName = "TCGA_GBM_SignatureTFs"


predictGeneExpression( data = gbmTFData, dataSetName = dataSetName, networkName =
   randomNetworkName, pValCutoff = 0.01, localGeneCutoff = 0, path = projectPath )
```

Here, `randomNetworkName` specifies the prefix of the name of the previously created random network instance. Since the random network instance was computed based on the original network by degree-preserving network permutations, all links in the random network instance have an artificial p-value of 0.01 as indicated by `randomNetworkName`. Thus, the attribute `pValCutoff` must be set to 0.01, whereas values of `localGeneCutoff` greater than zero would still enable to further filter the random network instance.

Slightly extending these code usage examples, correlations between network-based predicted and original measured gene expression levels are shown in Fig. 2c for GBMs [10], low grade gliomas (LGG) [11], and pilocytic astrocytomas (PA) [12]. For each of these independent test cohorts, the learned network reached significantly improved predictions in comparison to corresponding random network instances ($P < 6.75 \times 10^{-35}$, Mann-Withney U-Test).

3.6 Network-based impact computations

regNet can use a learned network to compute for a given data set or an individual sample the potential impact that one gene may have on the expression level of another gene.

3.6.1 Computation of impacts of major regulators on other transcription factors

For example, considering our learned TF-TF interaction network, we next use regNet to determine which of the revealed major regulators (Fig. 2b) tend to have the greatest impact on the expression levels of all other TFs. This is done using the following R code.

```
#Compute cohort-specific impact matrix
dataSetName = "AS_SignatureTFs"
networkName = "AS_SignatureTFs"

computeNetworkFlowMatrix_CohortSpecificAbsoluteImpacts( data = data, dataSetName =
   dataSetName, networkName = networkName, pValCutoff = 0.01, localGeneCutoff = 0,
   colSumsThreshold = 1e-3, path = projectPath )
```

```
#Determine average impacts of major regulators
sourceGenes = c( "RBBP4", "NFIA", "MEOX2", "PAX6", "ZNF337", "THRB", "ZCCHC24",
   "CCNL2", "TBR1", "ZNF300", "APBA1", "GPR123" )
targetGenes = data$genes
outputFile = "AvgImpactsOfMajorRegulators.txt"


getAverageImpacts_CohortSpecificAbsoluteImpacts( sourceGenes = sourceGenes,
   targetGenes = targetGenes, dataSetName = dataSetName, networkName = networkName,
   pValCutoff = 0.01, localGeneCutoff = 0, colSumsThreshold = 1e-3, path =
   projectPath, outputFile = outputFile )
```

First, the impact matrix is computed iteratively using the function `computeNetworkFlowMatrix_CohortSpecificAbsoluteImpacts` stopping if the sum of the column sums of the current impact matrix increases less than the value of the attribute `colSumsThreshold` in comparison to the previous impact matrix. The resulting impact matrix is automatically saved into the folder `NetworkPropagation/NetworkFlow/FinalNetworkFlowMatrices` using a pre-defined file name. This matrix contains for each TF-pair (a, b) the potential impact that TF a has on the expression level of TF b. Next, all previously revealed major regulators (`sourceGenes`) are considered to determine their average impacts on each individual gene in the network (`targetGenes`). This is done using the function `getAverageImpacts_CohortSpecificAbsoluteImpacts` with parameters (`dataSetName`, `networkName`, `pValCutoff`, `localGeneCutoff`, `colSumsThreshold`) identical to those used for the computation of the impact matrix to ensure that regNet is able to load the previously saved impact matrix. The resulting average impacts of each major regulator are contained in the file `outputFile` saved under `NetworkPropagation/ImpactComputations`. Again, regNet reports details about the individual computations and the exact location of all created files.

In addition, the impact computation can also be done for the previously computed the random network instance of the TF-TF interaction network. First, one has to modify the parameter `networkName` to `RandomNetwork_1_PValueCutoff_0.01_BasedOn_AS_SignatureTFs` in the function call `computeNetworkFlowMatrix_CohortSpecificAbsoluteImpacts` above. Next, one has to specify a new output file `outputFile` for the function call `getAverageImpacts_CohortSpecificAbsoluteImpacts` to obtain the average impacts of the major regulators with respect to the random network instance. This is done using the previously computed random network instance (see Section 2.4.3) using the following R code.

```
#Compute cohort-specific impact matrix for a random network instance
dataSetName = "AS_SignatureTFs"
networkName = "RandomNetwork_1_PValueCutoff_0.01_BasedOn_AS_SignatureTFs"

computeNetworkFlowMatrix_CohortSpecificAbsoluteImpacts( data = data, dataSetName =
   dataSetName, networkName = networkName, pValCutoff = 0.01, localGeneCutoff = 0,
   colSumsThreshold = 1e-3, path = projectPath )
```

```
#Determine average impacts of major regulators under the random network instance
sourceGenes = c( "RBBP4", "NFIA", "MEOX2", "PAX6", "ZNF337", "THRB", "ZCCHC24",
   "CCNL2", "TBR1", "ZNF300", "APBA1", "GPR123" )
targetGenes = data$genes
outputFile = "AvgImpactsOfMajorRegulators_RandomNetwork_1.txt"

getAverageImpacts_CohortSpecificAbsoluteImpacts( sourceGenes = sourceGenes,
   targetGenes = targetGenes, dataSetName = dataSetName, networkName = networkName,
   pValCutoff = 0.01, localGeneCutoff = 0, colSumsThreshold = 1e-3, path =
   projectPath, outputFile = outputFile )
```

3.6.2 Comparison of impacts to random network instances

Marginally extending these code usage examples, we compared the average impact of each major regulator on other network genes to corresponding average impacts obtained under 100 random network instances. We find that the expression levels of three of the major regulators (CCNL2, TBR1, ZNF300) in the original TF-TF interaction network have impacts on other genes that are significantly greater than under random networks of the same complexity (Fig. 2d, Bonferroni-adjusted p-values $P < 1.47 \times 10^{-4}$). Especially, the underexpression of TBR1 in astrocytomas in comparison to normal brain tissue has a strong impact on the expression levels of all other TFs in the interaction network. TBR1 is only expressed in post-mitotic cells and known to be an important regulator of neural migration and axonal projection required for normal brain development [14]. Interestingly, TBR1 shows a clearly reduced expression in diffuse astrocytomas compared to pilocytic astrocytomas (Fig. 2b). Thus, in line with these findings, our results suggest that the downregulation of TBR1 plays an important role in astrocytoma development and may contribute to the substantially greater malignant behavior of diffuse astrocytomas in comparison to pilocytic astrocytomas that typically show a much better outcome.

References

[1] M. Seifert, B. Friedrich, and A. Beyer. Importance of rare gene copy number alterations for personalized tumor characterization and survival analysis. *Genome Biology*, 17:204, 2016.

[2] R. Tibshirani. Shrinkage and Selection via the Lasso. *J. R Stat. Soc B*, 58:267–288, 1996.

[3] R. Lockhart et al. A significance test for the lasso. *Ann. Stat.*, 42:413–468, 2014.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.*, 33(1):1–22, 2010.

[5] T. Hastie and B. Efron. lars: Least angle regression, lasso and forward stagewise, 2013.

[6] R. Lockhart, J. Taylor, R. J. Tibshirani, and R. Tibshirani. covTest: Computes covariance test for adaptive linear modelling, 2013.

[7] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J R Stat Soc Series B*, 57:289–300, 1995.

[8] M. Seifert, M. Garbe, B. Friedrich, M. Mittelbronn, and B. Klink. Comparative transcriptomics reveals similarities and differences between astrocytoma grades. *BMC Cancer*, 15:952, 2015.

[9] M. Safran et al. GeneCards Version 3: the human gene integrator. *Database (Oxford)*, 2010:baq20, 2010.

[10] TCGA. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455:1061–1068, 2008.

[11] TCGA. Comprehensive, integrative genomic analysis of diffuse lower-grade gliomas. *N. Eng. J. Med.*, 372:2481–2498, 2015.

[12] M. K. Sharma, D. B. Mansur, G. Reifenberger, A. Perry, J. R. Leonard, K. D. Aldape, et al. Distinct genetic signatures among pilocytic astrocytomas relate to their brain region origin. *Cancer Res.*, 67:890–900, 2007.

[13] H. Ohgaki and P. Kleihues. The definition of primary and secondary glioblastoma. *Clin Cancer Res.*, 19:764–772, 2013.

[14] A. Bulfone, S.M. Smiga, K. Shimamura, A. Peterson, L. Puelles, and J.L. Rubenstein. T-brain-1: a homolog of Brachyury whose expression defines molecularly distinct domains within the cerebral cortex. *Neuron*, 15:63–78, 1995.