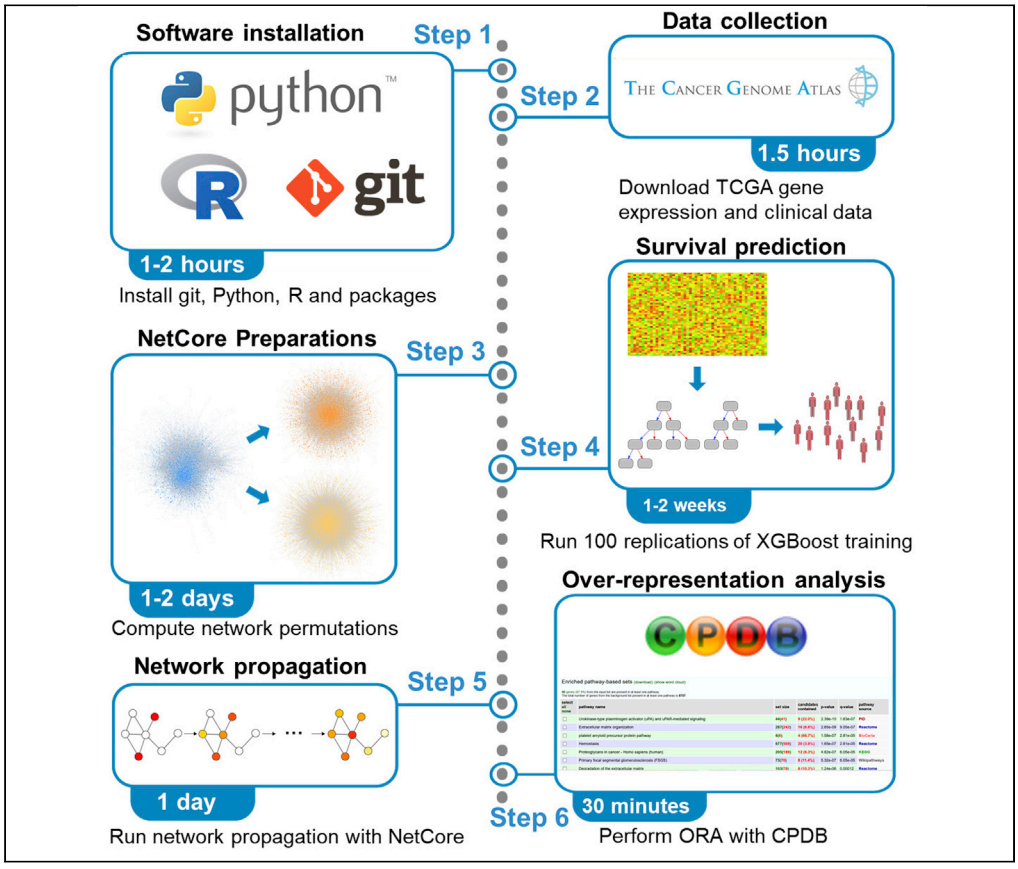
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Gradient tree boosting and network propagation for the identification of pan-cancer survival networks



Cancer survival prediction is typically done with uninterpretable machine learning techniques, e.g., gradient tree boosting. Therefore, additional steps are needed to infer biological plausibility of the predictions. Here, we describe a protocol that combines pan-cancer survival prediction with XGBoost tree-ensemble learning and subsequent propagation of the learned feature weights on protein interaction networks. This protocol is based on TCGA transcriptome data of 8,024 patients from 25 cancer types but can easily be adapted to cancer patient data from other sources.

Kristina Thedinga, Ralf Herwig

thedinga@molgen.mpg.de (K.T.)
herwig@molgen.mpg.de (R.H.)

Highlights

Efficient pan-cancer survival prediction with XGBoost

Network propagation with NetCore improves biological plausibility of features

Combined approach identifies pan-cancer survival networks

Thedinga & Herwig, STAR Protocols 3, 101353
June 17, 2022
https://doi.org/10.1016/j.xpro.2022.101353

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Gradient tree boosting and network propagation for the identification of pan-cancer survival networks

Kristina Thedinga[1,2,*] and Ralf Herwig[1,3,*]

[1]Department of Computational Molecular Biology, Max-Planck-Institute for Molecular Genetics, Ihnestrasse 63-73, 14195 Berlin, Germany

[2]Technical contact

[3]Lead contact

*Correspondence: thedinga@molgen.mpg.de (K.T.), herwig@molgen.mpg.de (R.H.)
https://doi.org/10.1016/j.xpro.2022.101353

SUMMARY

Cancer survival prediction is typically done with uninterpretable machine learning techniques, e.g., gradient tree boosting. Therefore, additional steps are needed to infer biological plausibility of the predictions. Here, we describe a protocol that combines pan-cancer survival prediction with XGBoost tree-ensemble learning and subsequent propagation of the learned feature weights on protein interaction networks. This protocol is based on TCGA transcriptome data of 8,024 patients from 25 cancer types but can easily be adapted to cancer patient data from other sources.

For complete details on the use and execution of this protocol, please refer to Thedinga and Herwig (2022).

BEFORE YOU BEGIN

This protocol describes how to predict cancer patient survival and compute a pan-cancer survival network using TCGA data. However, in principle it is also possible to predict survival and derive disease-specific networks for other datasets. If this is desired, the respective data needs to be pre-processed such that its structure resembles the structure of the pre-processed TCGA data. In particular, metrics should be used that allow reasonable comparison across different samples. The code described in this protocol was tested under the Linux operating system only. If you are using another operating system, please check for compatibility.

Installing software

Timing: 1–2 h

Several software packages are used in our protocol for performing computational tasks such as handling of data, machine learning as well as statistical analysis and representation of the results.

1. Install Python and related packages.
   a. Install Python (version 3.7). Download and documentation is available from https://www.python.org/.
   b. Install NumPy (version 1.18.5) (Harris et al., 2020). Documentation is available from https://numpy.org/. To install NumPy, you can type:

```
>pip install numpy==1.18.5
```

c. Install Pandas (version 1.1.5) (McKinney, 2010). Documentation is available from https://pandas.pydata.org/. To install Pandas, you can type:

```
>pip install pandas==1.1.5
```

d. Install tqdm (version 4.38.0) (da Costa-Luis et al., 2021). Documentation is available from https://tqdm.github.io/. To install tqdm, you can type:

```
>pip install tqdm==4.38.0
```

e. Install SciPy (version 1.2.1) (Virtanen et al., 2020). Documentation is available from https://scipy.org/. To install SciPy, you can type:

```
>pip install scipy=1.2.1
```

f. Install Matplotlib (version 3.1.1) (Hunter, 2007). Documentation is available from https://matplotlib.org/. To install Matplotlib, you can type:

```
>pip install matplotlib=3.1.1
```

g. Install scikit-learn (version 0.22.2.post1) (Pedregosa et al., 2011). Documentation is available from https://scikit-learn.org/. To install scikit-learn, you can type:

```
>pip install scikit-learn==0.22.2.post1
```

h. Install Seaborn (version 0.9.0) (Waskom, 2021). Documentation is available from https://seaborn.pydata.org/. To install Seaborn, you can type:

```
>pip install seaborn==0.9.0
```

i. Install NetworkX (version 2.3) (Hagberg et al., 2008). Documentation is available from https://networkx.org/. To install NetworkX, you can type:

```
>pip install networkx==2.3
```

j. Install XGBoost (version 0.90) (Chen and Guestrin, 2016). Documentation is available from https://xgboost.readthedocs.io. To install XGBoost, you can type:

```
>pip install xgboost==0.90
```

k. Install MyGene (version 3.1.0) (Wu et al., 2013). Documentation is available from https://docs.mygene.info/projects/mygene-py. To install MyGene, you can type:

```
>pip install mygene==3.1.0
```

2. Install R and related packages.

a. Install R (version 3.6.3 or current version). Download and documentation is available from https://www.r-project.org/.
b. Install Bioconductor (version 3.10 or current version) (Huber et al., 2015). Installation instructions and documentation are available from https://www.bioconductor.org/. To install Bioconductor, start R and type:

```
>if (!require("BiocManager", quietly = TRUE))

>  install.packages("BiocManager")

>BiocManager::install(version = "3.10")
```

c. Install Bioconductor package TCGAbiolinks (version 2.12.6 or current version) (Colaprico et al., 2016; Mounir et al., 2019; Silva et al., 2016). Installation instructions and documentation are available from https://bioconductor.org/packages/release/bioc/html/TCGAbiolinks.html. To install TCGAbiolinks, start R and type:

```
>BiocManager::install("TCGAbiolinks")
```

d. Install optparse (version 1.6.6 or current version) . Installation instructions and documentation are available from https://github.com/trevorld/r-optparse. To install optparse, start R and type:

```
>install.packages(``optparse'')
```

e. Install dplyr (version 1.0.0 or current version) (Wickham et al., 2021). Installation instructions and documentation are available from https://dplyr.tidyverse.org/. To install dplyr, start R and type:

```
>install.packages(``dplyr'')
```

f. Install reshape2 (version 1.4.4) (Wickham, 2007). Installation instructions and documentation are available from https://rdocumentation.org/packages/reshape2. To install reshape2, start R and type:

```
>install.packages(``reshape2'')
```

g. Install rjson (version 0.2.20 or current version) (Couture-Beil, 2022). Download and documentation are available from https://cran.r-project.org/web/packages/rjson/. To install rjson, start R and type:

```
>install.packages(``rjson'')
```

h. Install ggplot2 (version 3.3.1 or current version) (Wickham, 2009). Installation instructions and documentation are available from https://www.rdocumentation.org/packages/ggplot2. To install ggplot2, start R and type:

```
>install.packages(``ggplot2'')
```

i. Install ggpubr (version 0.2.5 or current version) (Kassambara, 2020). Installation instructions and documentation are available from https://rdocumentation.org/packages/ggpubr. To install ggpubr, start R and type:

```
>install.packages(''ggpubr'')
```

3. Install Git version control system. Download and documentation is available from https://git-scm.com/.

⚠ CRITICAL: The indicated software and package versions were used in (Thedinga and Herwig, 2022). Other versions than the ones indicated here were not tested. If you intend to use other versions of software or packages, please check for compatibility and be aware that steps described in this protocol might not work as expected.

Downloading XGBoost survival network implementation

Timing: 5 min

The code used in this protocol, including data download and pre-processing, training XGBoost pan-cancer survival prediction models, preparing the results for network propagation and over-representation analysis, and visualization of results, is included in the following GitHub repository.

4. Download or clone XGBoost Survival Network GitHub repository by executing the following command in your Unix/Linux shell:

```
>git clone https://github.molgen.mpg.de/thedinga/xgb_survival_network
```

Downloading and preprocessing TCGA data

Timing: 1.5 h

For survival prediction with XGBoost (Chen and Guestrin, 2016) and the identification of a survival network with network propagation, FPKM-normalized gene expression and supporting clinical data from 25 different TCGA cancer cohorts is used. This data can be downloaded from the Genomic Data Commons (GDC) in one of two ways. The first option is to download manifest files and sample sheets for the 25 cancer cohorts from https://portal.gdc.cancer.gov/ and then using the GDC Data Transfer Tool (available at https://gdc.cancer.gov/access-data/gdc-data-transfer-tool) to download the actual data based on these manifest files. The second option is to use the TCGAbiolinks Bioconductor/R package (Colaprico et al., 2016; Mounir et al., 2019; Silva et al., 2016) to download the data programmatically. In this protocol, we will focus on the second option of downloading the data with TCGAbiolinks because it allows downloading all necessary data with only one program call and avoids having to download manifest files and sample sheets for each cohort separately.

5. Download TCGA gene expression data and supporting clinical data.
   a. Navigate to the directory you have downloaded the XGBoost Survival Network repository into.
   b. Run "dowloadTCGAData.R" script to download TCGA FPKM-normalized gene expression data and clinical data. When using default options, data from the 25 TCGA cohorts used in this protocol will be downloaded to a subdirectory "TCGA_data" in your current working

directory. To download and prepare gene expression and clinical data for the survival prediction step, type: troubleshooting 1.

```
>Rscript downloadTCGAData.R
```

Note: The downloaded gene expression data contains some patients with multiple samples corresponding to the same case ID. For these patients, only the sample with the lexicographically highest TCGA barcode is kept.

Note: For the analyses in (Thedinga and Herwig, 2022) and shown in the Figures of this protocol, TCGA data was downloaded using manifest files and the GDC Data Transfer Tool. This can lead to a different ordering of samples as compared to downloading the data with TCGAbiolinks, which results in different training and test data splits when training the XGBoost survival prediction model. In case of multiple samples corresponding to the same case ID, the first sample was kept. In case you want to reproduce the exact results from (Thedinga and Herwig, 2022) and shown in the Figures of this protocol, use the GDC Data Transfer Tool to download the TCGA data based on the manifest files provided in the "data" directory at https://github.molgen.mpg.de/thedinga/xgb_survival_network. Keep only samples with sample type annotation "Primary Tumor" and "Primary Blood Derived Cancer – Peripheral Blood" and for multiple samples corresponding to the same case ID, keep the first sample. Additionally, you should change the IDs of the samples to their corresponding case IDs so that gene expression measurements can be matched with clinical patient data during training of the survival prediction model.

Note: It is necessary to apply a reasonable metric that allows direct comparison of features across the different samples. In our study, we used FPKM (fragments per kilobase million) as a metric because it had been used before (Dereli et al., 2019) and our original study performed a comparison of XGBoost with these methods (Thedinga and Herwig, 2022). Another choice of gene expression data normalization would be TPM (transcripts per kilobase million). Here, the sum of the normalized features in each sample is the same and, thus, the feature expression is better comparable across samples.

Installing NetCore and preparing network propagation

Timing: 1–2 days

NetCore (Barel and Herwig, 2020) is a network propagation approach based on node coreness. In this protocol, network propagation is applied to genes identified as survival prediction features during XGBoost pan-cancer model training. NetCore uses a permutation test to compute p-values for the network nodes. Thus, permutations of the input protein-protein interaction (PPI) network are required as an additional input and need to be pre-computed before performing network propagation.

6. Download and install NetCore.
   a. Download or clone NetCore as follows:

```
>git clone https://github.molgen.mpg.de/barel/NetCore
```

   b. To install NetCore, navigate to the directory where NetCore has been downloaded to and type:

```
>python setup.py install
```

7. Prepare network propagation by computing random permutations of the ConsensusPathDB high-confidence PPI:
   a. Navigate to the directory where NetCore has been downloaded to and launch Python.
   b. In Python, compute network permutations with NetCore as follows: troubleshooting 2.

```
>import sys

>sys.path.append(''netcore/'')

>import netcore

>import pandas as pd

>from permutations_test import make_network_permutations

>make_network_permutations(net_file=''data/CPDB_high_confidence.txt'',
                          net_name=''CPDB_high_confidence'',
                          output_path=''data/'',
                          num_perm=100,
                          swap_factor=100,
                          num_cores=64)
```

⚠ CRITICAL: This code will be executed assuming it can use 64 cores. If you are running this code on a machine with less cores, you need to change the `num_cores` argument accordingly before execution.

Note: The network module identification step of NetCore is based on significance value calculation of the re-ranked node weights derived from randomly permuted graphs. Thus, the output of the method depends on the structure of the protein-protein interaction network that was used. For our analysis we used the high confidence PPI network as provided by the previous ConsensusPathDB version 34 (Herwig et al., 2016), which is stored in the file "CPDB_high_confidence.txt" in the NetCore GitHub repository. The use of other PPI networks might lead to different results.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
HTSeq-FPKM gene expression and clinical files	The Cancer Genome Atlas (TCGA); available through GDC data portal or TCGAbiolinks Bioconductor/R package (Colaprico et al., 2016; Mounir et al., 2019; Silva et al., 2016)	GDC data portal: https://portal.gdc.cancer.gov/ TCGAbiolinks: https://bioconductor.org/packages/release/bioc/html/TCGAbiolinks.html
ConsensusPathDB (CPDB) protein-protein interaction network (version 34)	(Herwig et al., 2016)	https://github.molgen.mpg.de/barel/NetCore/blob/6860331ae9ff8725e666d936e9b853ef28893a00/data/CPDB_high_confidence.txt

(Continued on next page)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
Python 3.7	Python Software Foundation	https://www.python.org/
NumPy (version 1.18.5) Python package	(Harris et al., 2020)	https://numpy.org/
Pandas (version 1.1.5) Python package	(McKinney, 2010)	https://pandas.pydata.org/
tqdm (version 4.38.0) Python package	(da Costa-Luis et al., 2021)	https://tqdm.github.io/
SciPy (version 1.2.1) Python package	(Virtanen et al., 2020)	https://scipy.org/
Matplotlib (version 3.1.1) Python package	(Hunter, 2007)	https://matplotlib.org/
scikit-learn (version 0.22.2.post1) Python package	(Pedregosa et al., 2011)	https://scikit-learn.org/
Seaborn (version 0.9.0) Python package	(Waskom, 2021)	https://seaborn.pydata.org/
NetworkX (version 2.3) Python package	(Hagberg et al., 2008)	https://networkx.org/
XGBoost (version 0.90) Python package	(Chen and Guestrin, 2016)	https://xgboost.readthedocs.io
MyGene (version 3.1.0) Python package	(Wu et al., 2013)	https://docs.mygene.info/projects/mygene-py
R 3.6.3	The R Foundation	https://www.r-project.org/
Bioconductor (version 3.10) R package	(Huber et al., 2015)	https://www.bioconductor.org/
TCGAbiolinks (version 2.12.6) Bioconductor/R package	(Colaprico et al., 2016; Mounir et al., 2019; Silva et al., 2016)	https://bioconductor.org/packages/release/bioc/html/TCGAbiolinks.html
optparse (version 1.6.6) R package	The Comprehensive R Archive Network (CRAN)	https://github.com/trevorld/r-optparse
dplyr (version 1.0.0) R package	(Wickham et al., 2021)	https://dplyr.tidyverse.org/
reshape2 (version 1.4.4) R package	(Wickham, 2007)	https://rdocumentation.org/packages/reshape2
rjson (version 0.2.20) R package	(Couture-Beil, 2022)	https://cran.r-project.org/web/packages/rjson/
ggplot2 (version 3.3.1) R package	(Wickham, 2009)	https://www.rdocumentation.org/packages/ggplot2
ggpubr (version 0.2.5) R package	(Kassambara, 2020)	https://rdocumentation.org/packages/ggpubr
Git version control system	Software Freedom Conservancy	https://git-scm.com/
Pan-cancer XGBoost survival prediction	(Thedinga and Herwig, 2022)	https://github.molgen.mpg.de/thedinga/xgb_survival_network
Code for processing intermediate results	This Paper; (Thedinga and Herwig, 2022)	https://github.molgen.mpg.de/thedinga/xgb_survival_network
NetCore	(Barel and Herwig, 2020)	https://github.molgen.mpg.de/barel/NetCore
CPDB over-representation analysis	(Herwig et al., 2016; Kamburov and Herwig, 2022)	http://cpdb.molgen.mpg.de/

STEP-BY-STEP METHOD DETAILS

Here we describe step-by-step how to train the XGBoost pan-cancer survival prediction model, infer a pan-cancer survival network by performing network propagation on the important genes identified during model training, and find significantly enriched biological pathways based on the network propagation results. To illustrate these steps, we show as an example the results for 25 different TCGA cohorts and 100 replications of model training from (Thedinga and Herwig, 2022).

Survival prediction with XGBoost

Timing: 1–2 weeks (~7 h per replication)

TCGA patients from 25 different cancer cohorts are randomly split into 80% training and 20% test patients and a survival prediction model is trained on the gene expression data corresponding to the training patients. Model training includes a feature selection step, where the number of genes used for survival prediction is reduced to 500 in each replication, and a hyperparameter optimization step, where model hyperparameters such as tree depth are tuned. We refer the reader to (Thedinga and Herwig, 2022) for a more detailed description. After training is completed, the test data is then used to evaluate the trained model. This procedure is repeated 100 times for different splits of the

patients into training and test data. Users might also run a smaller number of replications (e.g., 10) to reduce runtime of this step. However, results of the network propagation and over-representation analysis following the survival prediction step can vary depending on the number of replications.

1. Navigate to the directory you have downloaded the XGBoost Survival Network repository into.
2. Run the model replications of XGBoost model training as follows:

```
>python run_xgb_survival_replications.py
   -result results/
   -features features/
   -replication_start 1
   -replication_end 100
   -threads 64
```

to run 100 replications of model training. If you want to run a single model replication only or distribute model training (e.g., to multiple servers), you can also execute each model replication separately by setting the `-s` and `-e` flags to the respective model replication. E.g., for running model replication 3 only, type: troubleshooting 2, 3, and 4.

```
>python run_xgb_survival_replications.py
   -result results/
   -features features/
   -replication_start 3
   -replication_end 3
   -threads 64
```

⚠ CRITICAL: The code for running the model replications of XGBoost training will use 64 threads. You should change the `threads` argument to the number of threads you want XGBoost to use according to the machine you are using.

Note: The random seed used for splitting the data into training and test sets is computed based on the model replication (i.e., `seed*num_replication`). If you want to reproduce the results from (Thedinga and Herwig, 2022), you should use the default seed of 135, otherwise you can change the seed via the `-seed` argument in the program call.

Note: The machine learning step is the most time-consuming step of the protocol and is heavily dependent on the hardware that is used. Our time estimations rely on the use of a Supermicro 2023US-TR4 Linux server with dual AMD EPYC 7601 CPU and 64 cores.

Note: Although the XGBoost framework generally offers GPU support, the objective function and metric (`survival:cox` and `cox-nloglik`, respectively) used in this protocol for survival prediction are currently not supported on GPU.

Note: The number of replications determines both the runtime and the accuracy of the results. We strongly recommend using as much as 100 replications since this allows the XGBoost

method to sufficiently exhaust the large amount of features. However, runtime could be reduced by reducing the number of replications.

Optional: Visualize model performances of the survival prediction models trained in the different model replications by plotting each cohort against the C-Indices obtained in the different model replications using boxplots. You can create such a visualization by running:

```
>Rscript plotPredictionPerformance.R
   -output_file ''model_performance_xgb_pancancer.pdf''
   -result_path results/
   -num_replications 100
```

where `num_replications` should be set to the number of model training replications you have performed in the previous step. An example visualizing the survival prediction performances for the 100 replications of model training from (Thedinga and Herwig, 2022) is shown in Figure 1.

Network propagation with NetCore

Timing: 1 day

Gene weights are derived from the feature importance scores (measured as gain, see https://xgboost.readthedocs.io/en/latest/python/python_api.html) that were computed by the XGBoost algorithm in each replication of model training. To compute the weight of a gene, the sum of feature importance scores corresponding to this gene over all XGBoost model replications is calculated. All gene weights are then fed into NetCore as initial weights for network propagation. NetCore (Barel and Herwig, 2020) is a network propagation method based on node coreness and also implements a module identification step. The module identification step returns subnetworks connecting the most highly weighted input genes to genes that received a significant weight in the network propagation step.

3. Navigate to the directory you have downloaded the XGBoost Survival Network repository into.
4. Prepare XGBoost pan-cancer survival prediction results for network propagation.
   a. Compute gene weights from the feature importance scores calculated during the different replications of pan-cancer XGBoost training. To compute the gene weights for network propagation from the survival prediction results, type:

```
>python prepare_XGBoost_results_for_NetCore.py
      -result_path results/
      -num_replications 100
      -output_path survival_network/
```

where `num_replications` should be set to the number of replications you have performed for XGBoost pan-cancer model training.

Note: Gene weights are calculated as the sum of feature importance scores for each gene over all model replications. Additionally, gene identifiers are converted from Ensembl IDs as used in XGBoost model training to Hugo Symbols to be compatible with the protein-protein

interaction (PPI) network used in network propagation with NetCore. Genes that do not map to a Hugo Symbol are discarded as they cannot be used in network propagation.

5. Perform network propagation with NetCore as follows:

```
>python <path_to_netcore>/netcore/netcore.py
        -e <path_to_netcore>/data/CPDB_high_confidence.txt
        -w survival_network/pancancer_gene_weights.txt
        -pd <path_to_netcore>/data/CPDB_high_confidence_edge_permutations/
        -o survival_network/
```

where <path_to_netcore> should be set the path where NetCore has been downloaded to. Troubleshooting 2 and 5. As an example, Figure 2 shows the largest network module identified by NetCore based on the gene weights from (Thedinga and Herwig, 2022), which were computed from 100 replications of XGBoost pan-cancer training.

Overrepresentation analysis of the survival sub-network

Timing: 30 min

Genes contained in the network modules identified by NetCore are further analyzed by over-representation analysis (ORA) to find significantly enriched biological pathways. In (Thedinga and Herwig, 2022) ORA is performed with QIAGEN Ingenuity Pathway Analysis (IPA) (Krämer et al., 2014). However, since QIAGEN IPA is a commercial application and thus not freely available, we demonstrate here how to perform ORA with the ORA application implemented in ConsensusPathDB (Herwig et al., 2016; Kamburov and Herwig, 2022).

6. Extract genes from the network modules identified by NetCore. The following script reads the file "core_norm_subnetworks.txt", which is generated by NetCore during the module identification step and extracts all genes that appear in any of the identified network modules.

```
>python extract_network_module_genes.py
       -result_path survival_network/
       -output_file survival_network/network_module_genes.txt
```

7. Extract genes contained in the high-confidence ConsensusPathDB protein-protein interaction network used in network propagation for use as a background list of genes in the over-representation analysis. To extract the genes from the high-confidence protein-protein interaction network, run:

```
>python extract_ppi_network_genes.py
       -ppi_path <path_to_netcore>/data/CPDB_high_confidence.txt
       -output_file survival_network/CPDB_ppi_network_genes.txt
```

where <path_to_netcore> should be set the path where NetCore has been downloaded to.

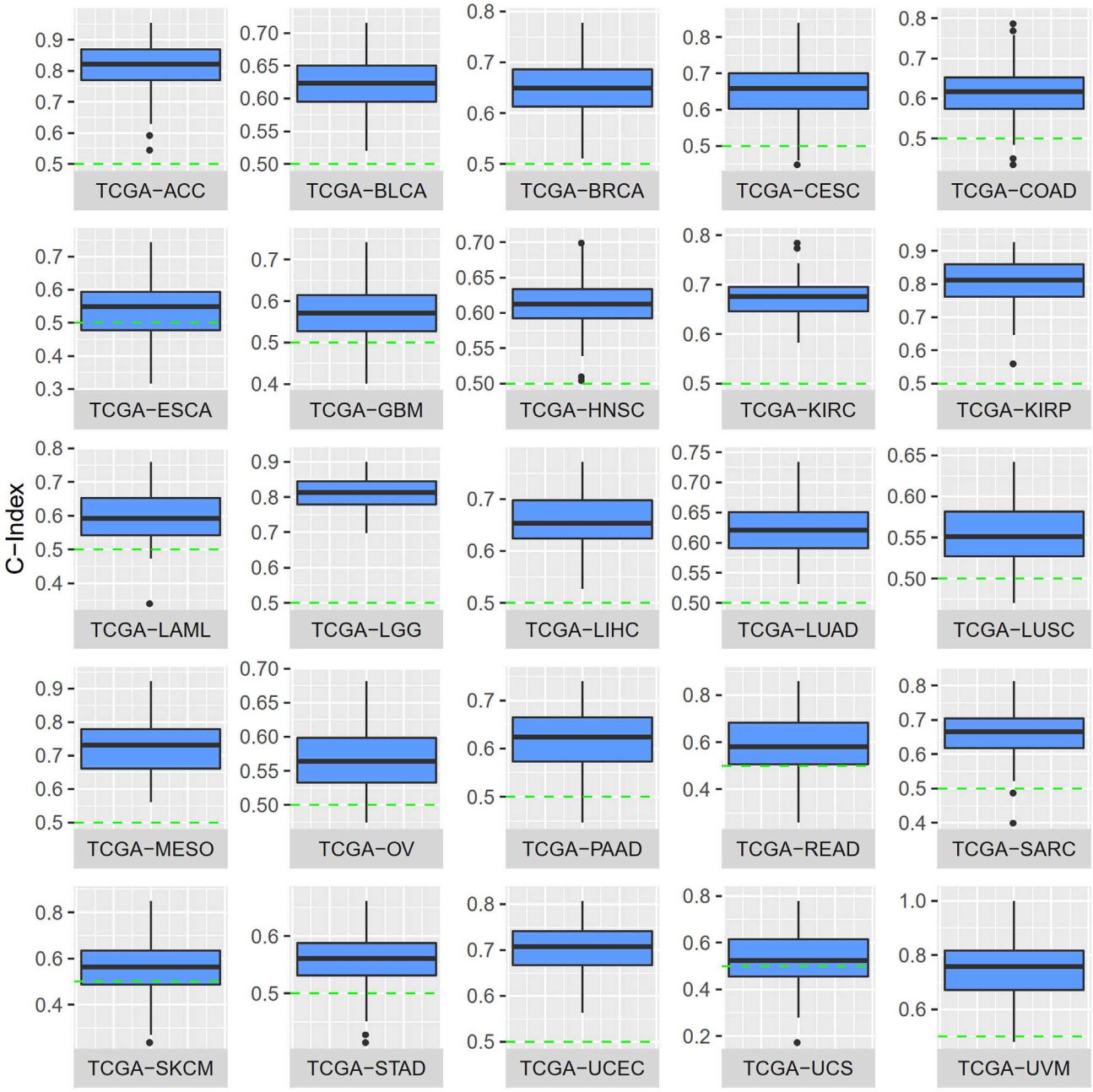


Figure 1. Survival prediction performances

The pan-cancer XGBoost survival prediction performance (depicted as C-Index boxplots) from (Thedinga and Herwig, 2022) for 100 replications of model training on 25 TCGA cancer cohorts.

Note: When performing over-representation analysis, the background list of genes is important because it influences the resulting p-value computations. As default, the ConsensusPathDB uses all annotated genes as background, but this can be modified by the user. A reasonable choice of background genes for analyzing functional information of the network modules could be, for example, the set of genes that are covered by the underlying protein-protein interaction network.

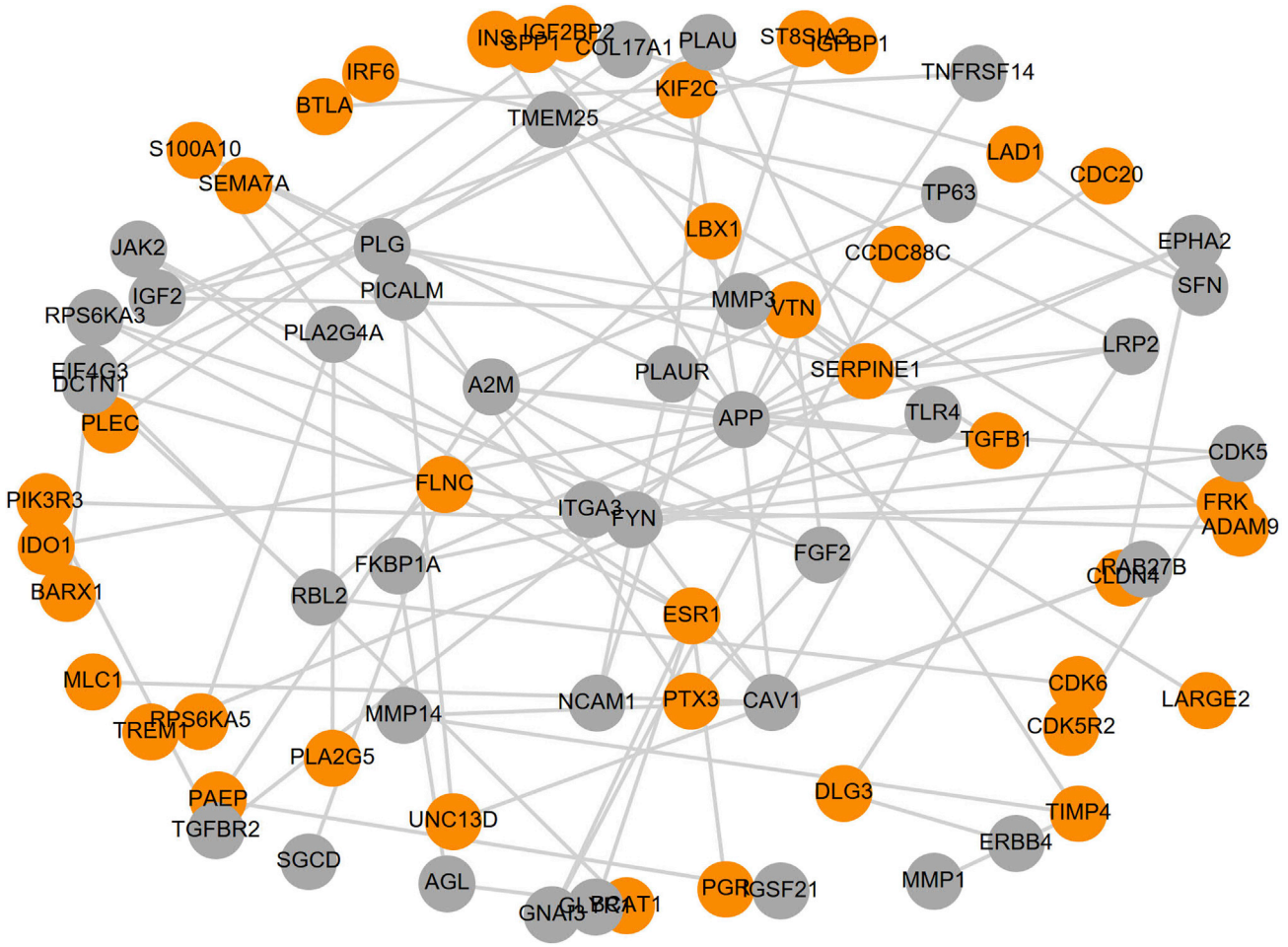
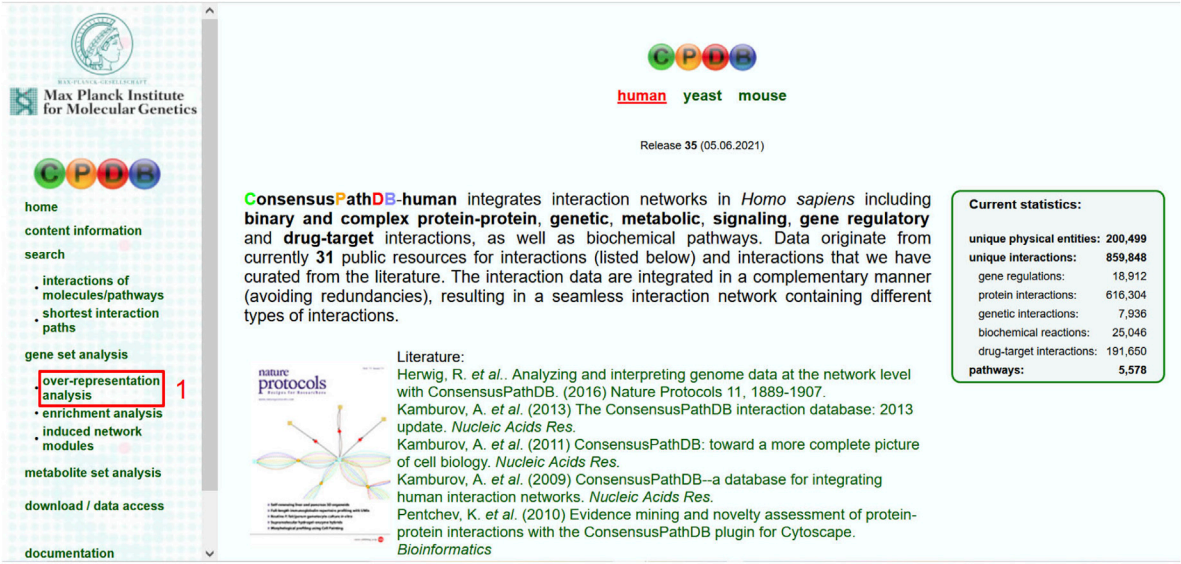


Figure 2. Pan-cancer survival network module

Largest network module identified by NetCore (Barel and Herwig, 2020) network propagation and module identification based on pan-cancer important features identified in (Thedinga and Herwig, 2022) from 100 replications of XGBoost model training. Orange nodes correspond to seed genes, while genes that were inferred during network propagation are colored in gray. Figure reprinted with permission from Thedinga and Herwig (2022).

8. Perform over-representation analysis (ORA) on the extracted network module genes (Figure 3). Troubleshooting 6.
   a. Open a browser window and go to http://cpdb.molgen.mpg.de/.
   b. On the ConsensusPathDB website, select the "over-representation analysis" tab in the "gene set analysis" category.
   c. Upload gene data.
      i. Upload the file "network_module_genes.txt" from the folder "survival_network" as file containing gene identifiers.
      ii. Upload the file "CPDB_ppi_network_genes.txt" from the folder "survival_network" as background list of genes.
      iii. Select "gene symbol (HGNC symbol)" as gene/protein identifier type.
      iv. Click "Proceed".
   d. Select functional sets for ORA.
      i. In "Pathway-based sets", select "pathways as defined by pathway databases".
      ii. Click "Find enriched sets" at the bottom of the page.
   e. Download ORA results in tab-delimited format.


Max Planck Institute for Molecular Genetics
human yeast mouse
Release 35 (05.06.2021)
home
content information
search
interactions of molecules/pathways
shortest interaction paths
gene set analysis
over-representation analysis
enrichment analysis
induced network modules
metabolite set analysis
download / data access
documentation
1
ConsensusPathDB-human integrates interaction networks in Homo sapiens including binary and complex protein-protein, genetic, metabolic, signaling, gene regulatory and drug-target interactions, as well as biochemical pathways. Data originate from currently 31 public resources for interactions (listed below) and interactions that we have curated from the literature. The interaction data are integrated in a complementary manner (avoiding redundancies), resulting in a seamless interaction network containing different types of interactions.
Current statistics:
unique physical entities: 200,499
unique interactions: 859,848
gene regulations: 18,912
protein interactions: 616,304
genetic interactions: 7,936
biochemical reactions: 25,046
drug-target interactions: 191,650
pathways: 5,578
nature protocols
Literature:
Herwig, R. et al.. Analyzing and interpreting genome data at the network level with ConsensusPathDB. (2016) Nature Protocols 11, 1889-1907.
Kamburov, A. et al. (2013) The ConsensusPathDB interaction database: 2013 update. Nucleic Acids Res.
Kamburov, A. et al. (2011) ConsensusPathDB: toward a more complete picture of cell biology. Nucleic Acids Res.
Kamburov, A. et al. (2009) ConsensusPathDB--a database for integrating human interaction networks. Nucleic Acids Res.
Pentchev, K. et al. (2010) Evidence mining and novelty assessment of protein-protein interactions with the ConsensusPathDB plugin for Cytoscape. Bioinformatics

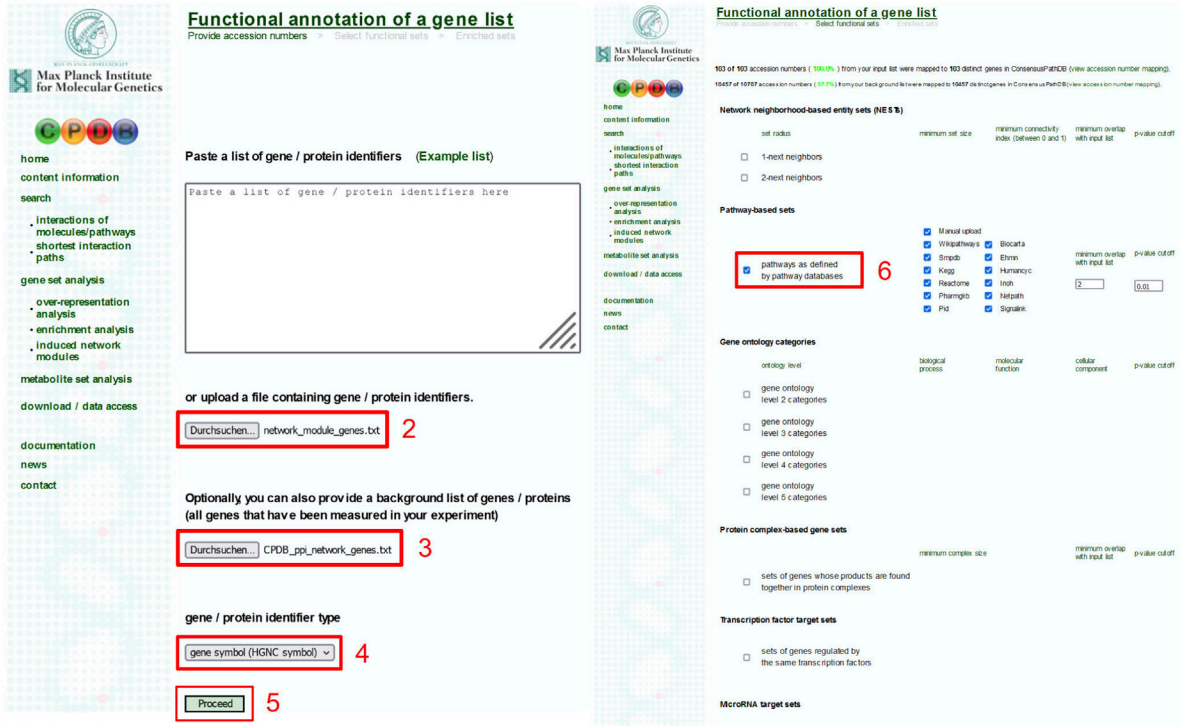

Functional annotation of a gene list
Provide accession numbers > Select functional sets > Enriched sets
Paste a list of gene / protein identifiers (Example list)
Paste a list of gene / protein identifiers here
or upload a file containing gene / protein identifiers.
Durchsuchen... network_module_genes.txt
2
Optionally you can also provide a background list of genes / proteins (all genes that have been measured in your experiment)
Durchsuchen... CPDB_ppi_network_genes.txt
3
gene / protein identifier type
gene symbol (HGNC symbol)
4
Proceed
5
Network neighborhood-based entity sets (NESTs)
Pathway-based sets
pathways as defined by pathway databases
6
Gene ontology categories
Protein complex-based gene sets
Transcription factor target sets
MicroRNA target sets
Putative enhancer target sets
Find enriched sets
7
Results summary
Enriched pathway-based sets
8

Figure 3. Over-representation analysis with ConsensusPathDB

Red numbers (1–8) illustrate the steps necessary to perform an over-representation analysis of the module genes identified during network propagation using the ConsensusPathDB (Herwig et al., 2016) ORA implementation.

Note: In (Thedinga and Herwig, 2022), QIAGEN IPA (Krämer et al., 2014) was used for over-representation analysis instead of ConsensusPathDB (Herwig et al., 2016; Kamburov and Herwig, 2022). For this reason results can deviate from the results shown in the paper.

Note: Over-representation analysis is typically dependent on annotation of gene sets, for example pathways, protein complexes, transcription factor target sets etc. Thus, when performing a Fisher test with these gene sets and the user's gene list, different pathways can be identified when using different pathway databases. If you do not have access to QIAGEN IPA, we suggest using ConsensusPathDB because it has collected such pathway-based gene sets from different source databases and over-representation analysis is done with all gene sets in parallel in order to gain a more comprehensive result.

Alternatives: It is also possible to use other publicly available tools such as PANTHER (Mi et al., 2021), Enrichr (Kuleshov et al., 2016), or DAVID (Huang et al., 2009a; 2009b) for analyzing over-representation of the gene list obtained after network propagation and module identification.

EXPECTED OUTCOMES

The protocol steps described above each yield output files and/or visualizations of the respective results. In the following subsections, the expected outputs and results of every step are described.

Survival prediction with XGBoost

XGBoost pan-cancer survival prediction produces two .json files for each replication of model training. The first file with the filename "important_genes_replication_<n>.json", with <n> being replaced by the respective model replication, contains the genes selected during the feature selection step of the pan-cancer XGBoost method and their respective feature importance scores assigned during feature selection. The second file named "xgb_measure_CI_replication_<n>_result.json" contains the results of the n^{th} replication of model training. Besides the model's predictions for the test patients, their TCGA case identifiers and their true survival times and censoring status, it also contains C-Indices evaluating the model's prediction performance and feature importance scores for the genes used for survival prediction by the trained model. Optionally, the prediction performances of the models trained in the different replications can be visualized in a boxplot. This boxplot is subdivided by TCGA cohorts and for each cohort depicts the C-Indices over all replications. Figure 1 displays an example boxplot visualizing the survival prediction performances of the 100 replications of model training from (Thedinga and Herwig, 2022).

Network propagation with NetCore

Network propagation with NetCore produces several output files, including gene weights after network propagation, the identified network modules and visualizations of these modules. The file "random_walk_weights.txt" contains the weights of all nodes in the protein-protein interaction network after network propagation as well as their p-values, which NetCore computes by permutation tests. The network modules identified in NetCore's module identification step are contained in a file with the name "core_norm_subnetworks.txt", where each row corresponds to one module and includes the genes comprising the module, the size of the module (i.e., the number of genes contained in the module), and the sum of node weights of the module. Visualizations of the network modules are stored in the folder "modules" of the output directory. Figure 2 shows the largest network module identified by NetCore based on the gene weights from (Thedinga and Herwig, 2022), which were computed from 100 replications of XGBoost pan-cancer training. In addition to the individual network modules, NetCore also outputs a seed subnetwork containing all identified

network modules. The file "extended_seed_subnet.pdf" contains a visualization of this seed subnetwork and "extended_seed_subnet.adjlist" contains the corresponding edge list. The module genes used in the over-representation analysis are extracted from "core_norm_subnetworks.txt".

Over-representation analysis of the survival sub-network

Over-representation analysis with ConsensusPathDB (Herwig et al., 2016; Kamburov and Herwig, 2022) produces an output file named "ORA_results.tab". This file contains the biological pathways that have been found to be significantly enriched in the over-representation analysis. For each of the significantly enriched pathways, this file lists information about this pathway including the enrichment p- and q-value of the pathway, its name, its source database (e.g., KEGG) as well as pathway size and the input genes contained in the pathway.

LIMITATIONS

The identification of the cancer survival sub-network is based on the XGBoost pan-cancer survival prediction results from 100 replications of model training. Executing 100 replications is computationally intensive in terms of the required runtime since each replication takes approximately seven hours on a server with 64 cores and running all 100 replications can thus take several weeks. However, runtime can potentially be reduced by parallelizing replications since one execution of XGBoost model training can most likely not effectively utilize all 64 cores and the server might be utilized more efficiently by running multiple instances of training in parallel. An alternative for executing 100 training replications is performing the network propagation and over-representation analysis on the features identified in a smaller number of model replications (e.g., just 10 instead of 100 replications). However, analysis results can deviate from the results shown in (Thedinga and Herwig, 2022) and become unstable when only a small number of replications is performed because the genes identified as important features can vary over replications.

Our method is based on cancer patient gene expression data provided by the TCGA consortium. Although TCGA comprises a variety of cancer patients from diverse cancer types, and we have implemented rigorous train-test splits and also tested our method on cancer types withheld in model training, we have not evaluated the method on any cancer patient data completely unrelated to the TCGA database. Applying the model to expression data from different cancer cohorts can raise several issues with respect to proper cross-cohort normalization methods, cancer-type classifications as well as comparable sample preparation steps. These issues are not covered by this protocol.

Furthermore, we considered only one molecular data modality, namely RNA-seq gene expression data, for model training and identification of the pan-cancer survival network. Incorporation of additional molecular data modalities like methylation and mutations could add more information to the model and complement the gene expression data. However, these additional data types would have to be brought into context with each other and require careful preprocessing.

TROUBLESHOOTING

Problem 1

Connection timed out or program interrupted during download and preprocessing of TCGA data (see "before you begin" step 5).

Potential solution

Restart the script for TCGA data download and preprocessing. The TCGAbiolinks package will automatically recognize which TCGA files have already been downloaded and only download the missing files. Additionally, the script will check which output files have already been created and skip download and preprocessing of existing files unless the *force_recomputation* flag is set.

Problem 2

Provided GitHub code from https://github.molgen.mpg.de/thedinga/xgb_survival_network or NetCore implementation from https://github.molgen.mpg.de/barel/NetCore produces error(s) (see "before you begin" step 7, and steps 2 and 5).

Potential solution

Make sure you have installed all required software and packages in the specified version (see "before you begin"). In case you have installed a required package, but in a different version than indicated, it might be necessary to up- or downgrade the respective package to the indicated version for the code to work properly.

Problem 3

When executing run_xgb_survival_replications.py, no patients are found for any/some of the cohorts (see step 2).

Potential solution

Make sure the clinical data files include a "submitter_id" or "case_submitter_id" column and the case identifiers listed in this column include the case identifiers from the header row of the corresponding gene expression file. It is important that the header row of the gene expression files contains case identifiers and not sample identifiers, so that these case identifiers can be matched with patient information from the clinical data files. In addition to "submitter_id" or "case_submitter_id", the clinical data files should include at least the following columns: "age_at_index", "days_to_death", "gender", "vital_status", and "days_to_last_follow_up".

Problem 4

XGBoost model training fails because of duplicate gene or patient identifiers (see step 2).

Potential solution

Use unique gene identifiers, for instance Ensembl IDs, and for patients with multiple measured samples, keep only one sample and discard the other ones.

Problem 5

NetCore does not find any network modules in module identification step (see step 5).

Potential solution

A possible cause for NetCore not finding any network modules might be that there are no genes with significant p-values or no genes with a weight above the weight threshold after network propagation. Potential solutions for this are increasing the p-value threshold (via the -pt argument) or decreasing the weight threshold (via the -wt argument) when executing NetCore.

Problem 6

When performing over-representation analysis (ORA), the module genes identified by NetCore cannot be mapped to ConsensusPathDB physical entities or no enriched pathways are found (see step 8).

Potential solution

Make sure you have selected the correct gene identifier type (e.g., "gene symbol (HGNC symbol)") for the genes you want to analyze when uploading your data to ConsensusPathDB and your provided background list of genes is also converted to this gene identifier type.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Ralf Herwig (herwig@molgen.mpg.de).

Materials availability

This study does not use any materials.

Data and code availability

In this protocol, data from The Cancer Genome Atlas (TCGA) consortium (https://www.cancer.gov/tcga) is used, which can be obtained through the Genomic Data Commons (GDC) data portal (https://portal.gdc.cancer.gov/) or through the TCGAbiolinks Biocoductor/R package (https://bioconductor.org/packages/release/bioc/html/TCGAbiolinks.html). The code generated during this study is available at https://github.molgen.mpg.de/thedinga/xgb_survival_network. A DOI can be found at https://doi.org/10.17617/1.6pxdnz96.

ACKNOWLEDGMENTS

The project was funded by the German Bundesministerium für Bildung und Forschung [01IS18044A (ML-Med)].

We thank Paul Prasse, Tobias Scheffer, and Martin Vingron for useful discussions on machine learning methodology. Atanas Kamburov has developed the ConsensusPathDB that is the source of the protein-protein interaction network, and Gal Barel has developed the NetCore propagation algorithm. Furthermore, discussions and feedback within the framework of the International Max Planck Research School for Biology and Computation were appreciated. The results of this study are in part based upon data generated by the TCGA Research Network: https://www.cancer.gov/tcga.

AUTHOR CONTRIBUTIONS

Conceptualization, K.T. and R.H.; Methodology, K.T.; Software, K.T.; Writing – Original Draft, K.T. and R.H.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Barel, G., and Herwig, R. (2020). NetCore: a network propagation approach using node coreness. Nucleic Acids Res. *48*, e98. https://doi.org/10.1093/nar/gkaa639.

Chen, T., and Guestrin, C. (2016). XGBoost: a scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Presented at the KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM), pp. 785–794. https://doi.org/10.1145/2939672.2939785.

Colaprico, A., Silva, T.C., Olsen, C., Garofano, L., Cava, C., Garolini, D., Sabedot, T.S., Malta, T.M., Pagnotta, S.M., Castiglioni, I., et al. (2016). TCGAbiolinks: an R/Bioconductor package for integrative analysis of TCGA data. Nucleic Acids Res. *44*, e71. https://doi.org/10.1093/nar/gkv1507.

Couture-Beil, A. (2022). Rjson: JSON for R (The Comprehensive R Archive Network (CRAN)).

da Costa-Luis, C., Larroque, S.K., Altendorf, K., Mary, H., Sheridan, R., Korobov, M., Yorav-Raphael, N., Ivanov, I., Bargull, M., Rodrigues, N., et al. (2021). tqdm: A fast, Extensible Progress Bar for Python and CLI (Zenodo). https://doi.org/10.5281/zenodo.5517697.

Dereli, O., Oğuz, C., and Gönen, M. (2019). Path2Surv: pathway/gene set-based survival analysis using multiple kernel learning. Bioinformatics *35*, 5137–5145. https://doi.org/10.1093/bioinformatics/btz446.

Hagberg, A.A., Schult, D.A., and Swart, P.J. (2008). Exploring Network Structure, Dynamics, and Function Using NetworkX (Proceedings of the 7th Python in Science Conference (SciPy2008)), pp. 11–15.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al. (2020). Array programming with NumPy. Nature *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2.

Herwig, R., Hardt, C., Lienhard, M., and Kamburov, A. (2016). Analyzing and interpreting genome data at the network level with ConsensusPathDB. Nat. Protoc. *11*, 1889–1907. https://doi.org/10.1038/nprot.2016.117.

Huang, D.W., Sherman, B.T., and Lempicki, R.A. (2009a). Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. Nat. Protoc. *4*, 44–57. https://doi.org/10.1038/nprot.2008.211.

Huang, D.W., Sherman, B.T., and Lempicki, R.A. (2009b). Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. Nucleic Acids Res. *37*, 1–13. https://doi.org/10.1093/nar/gkn923.

Huber, W., Carey, V.J., Gentleman, R., Anders, S., Carlson, M., Carvalho, B.S., Bravo, H.C., Davis, S., Gatto, L., Girke, T., et al. (2015). Orchestrating high-throughput genomic analysis with bioconductor. Nat. Methods *12*, 115–121. https://doi.org/10.1038/nmeth.3252.

Hunter, J.D. (2007). Matplotlib: a 2D graphics environment. Comput. Sci. Eng. *9*, 90–95. https://doi.org/10.1109/MCSE.2007.55.

Kamburov, A., and Herwig, R. (2022). ConsensusPathDB 2022: molecular interactions update as a resource for network biology. Nucleic Acids Res. *50*, D587–D595. https://doi.org/10.1093/nar/gkab1128.

Kassambara, A. (2020). ggpubr: "ggplot2" Based Publication Ready Plots (The Comprehensive R Archive Network (CRAN)).

Krämer, A., Green, J., Pollard, J., and Tugendreich, S. (2014). Causal analysis approaches in ingenuity pathway analysis. Bioinformatics *30*, 523–530. https://doi.org/10.1093/bioinformatics/btt703.

Kuleshov, M.V., Jones, M.R., Rouillard, A.D., Fernandez, N.F., Duan, Q., Wang, Z., Koplev, S., Jenkins, S.L., Jagodnik, K.M., Lachmann, A., et al. (2016). Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. Nucleic Acids Res. *44*, W90–W97. https://doi.org/10.1093/nar/gkw377.

McKinney, W. (2010). Data structures for statistical computing in python. In Presented at the Python in Science Conference, pp. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a.

Mi, H., Ebert, D., Muruganujan, A., Mills, C., Albou, L.-P., Mushayamaha, T., and Thomas, P.D. (2021). PANTHER version 16: a revised family classification, tree-based classification tool, enhancer regions and extensive API. Nucleic Acids Res. *49*, D394–D403. https://doi.org/10.1093/nar/gkaa1106.

Mounir, M., Lucchetta, M., Silva, T.C., Olsen, C., Bontempi, G., Chen, X., Noushmehr, H., Colaprico, A., and Papaleo, E. (2019). New functionalities in the TCGAbiolinks package for the study and integration of cancer data from GDC and GTEx. PLoS Comput. Biol. *15*, e1006701. https://doi.org/10.1371/journal.pcbi.1006701.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: machine learning in Python. J. Machine Learn. Res. *12*, 2825–2830.

Silva, T.C., Colaprico, A., Olsen, C., D'Angelo, F., Bontempi, G., Ceccarelli, M., and Noushmehr, H. (2016). TCGA Workflow: analyze cancer genomics and epigenomics data using Bioconductor packages. F1000Res. *5*, 1542. https://doi.org/10.12688/f1000research.8923.2.

Thedinga, K., and Herwig, R. (2022). A gradient tree boosting and network propagation derived pan-cancer survival network of the tumor microenvironment. iScience *25*. https://doi.org/10.1016/j.isci.2021.103617.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2.

Waskom, M. (2021). seaborn: statistical data visualization. JOSS *6*, 3021. https://doi.org/10.21105/joss.03021.

Wickham, H. (2009). ggplot2: Elegant Graphics for Data, Use R! (Springer International Publishing). https://doi.org/10.1007/978-0-387-98141-3.

Wickham, H. (2007). Reshaping data with the reshape package. J. Stat. Softw. *21*, 1–20. https://doi.org/10.18637/jss.v021.i12.

Wickham, H., François, R., Henry, L., and Müller, K. (2021). dplyr: A Grammar of Data Manipulation (The Comprehensive R Archive Network (CRAN)).

Wu, C., MacLeod, I., and Su, A.I. (2013). BioGPS and MyGene.info: organizing online, gene-centric information. Nucleic Acids Res. *41*, D561–D565. https://doi.org/10.1093/nar/gks1114.