netDx-manual Documentation

Release latest

Oct 30, 2017

Contents

The first version of the netDx user manual is in active development and not all sections are currently complete. Please contact _Shraddha Pai: mailto:shraddha.pai@utoronto.ca if these pages do not answer your questions.

Citation: Pai S., Hui S., Isserlin R., Shah M.A., Kaka H., and GD Bader. (2017) [netDx: Interpretable patient classification using integrated patient similarity networks](https://www.biorxiv.org/content/early/2017/09/26/084418). bioRXiV. (**doi:** https://doi.org/10.1101/084418)

Contents:

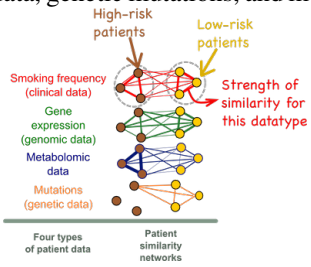
CHAPTER 1

Introduction

netDx is a **patient classifier** algorithm that can integrate several types of patient data into a single model. It specializes in the use of genomic data, which is distinct in the number and correlation structure of measures (e.g. 20,000 genes)It does this by converting each type of data into a view of patient similarity; i.e. by converting the data into a graph in which more similar patients are tightly linked, while less similar patients are not so tightly linked.

Motivation

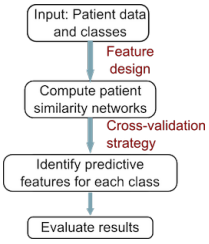
In this example, we try to predict which patients are at high-risk for lung cancer. We have four types of data: relevant clinical variables, including smoking frequency, gene expression data, genetic mutations, and metabolomic data. netDx converts the data into 4 views of patient similarity (edge strength



In the graphs above, the nodes are patients and the edges are weighted by similarity for that particular datatype. It is evident that the high-risk patients form a strongly interconnected cluster based on smoking frequency (red network) but that the clustering is less evident for gene expression data (green network).

How netDx works

The conceptual workflow for netDx is shown below. netDx starts with patient data as above. It allows users to define similarity for each of the input datatypes and creates the resulting patient similarity networks. It then uses machine learning to identify which of the input features were predictive for each class. Finally, it uses the predictive features to classify new patients of unknown type.



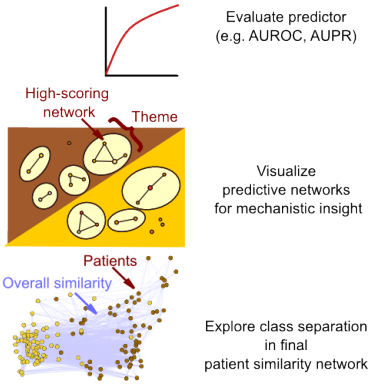
An important aspect of the predictor is the score associated with each input feature. This score indicates the frequency with which cross-validation identified a particular network as predictive for a patient label, and is a measure of predictive power. A threshold can be applied to this score, making passing networks "feature-selected".

Output

netDx broadly has two purposes. First, it serves as a classifier that can integrate heterogeneous datatypes. Second, it serves as a tool for clinical discovery and research, as identified features may provide mechanistic insight into the condition under study or identify new biomarkers.

netDx therefore provides several types of output that allow the user to examine the nature of the predictor:

- **Predicted labels** for test patients. If nested cross-validation is used, labels for all iterations are provided, along with individual-level classification accuracy.
- **Summary network scores:** Network-level scores for all cross-validation folds. Applying a cutoff for these results in "feature-selected" networks.
- Detailed output: All **intermediate results**, showing network rankings across cross-validation
- An **overall patient similarity network** created by integrating feature-selected networks
- Where applicable, a network visualization of selected features (also called an EnrichmentMap) is generated. This view shows the major themes present in feature-selected variables.



CHAPTER 2

Concepts used in netDx

In progress: 170905

Patient similarity network

Enrichment map

Feature

Feature selection

GeneMANIA

An algorithm that integrates similarity networks and ranks patients by similarity to a query (e.g. "rank patients by similarity to those with high-risk for lung cancer.") **Cite original papers**

Nested cross-validation

CHAPTER 3

Installation

Instructions for installing netDx software and running examples are provided in the netDx repo, which is currently a private github repo. The repo will be publicly released upon publication of the netDx methods paper (currently in preparation).

In the interim, please contact Gary Bader to use netDx in your project.

Update:5 Sep 2017

CHAPTER 4

Create Patient Similarity Networks (or Feature Design)

Overview

Defining meaningful patient similarity networks (or features) is the key to building a high-performing classifier. It is also crucial to using netDx for clinical discovery. When designing patient similarity networks, two considerations are the level at which individual variables should be grouped, and the measure used to define patient similarity.

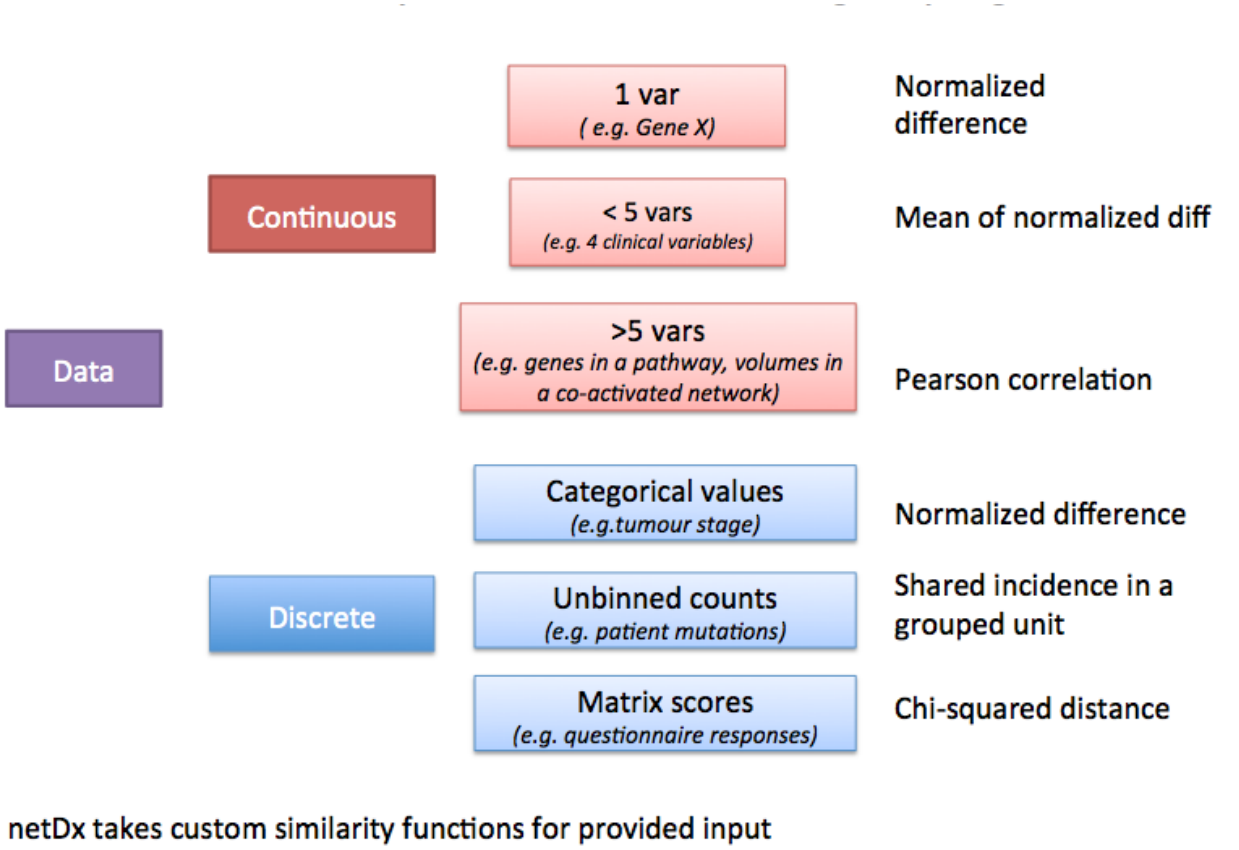
Grouping Variables: What makes an input network?

In a given datatype, not all measured variables are equally predictive. Moreover some groupings of variables may be more informative about mechanism, perhaps because they use prior knowledge. Such variables are more interpretable, as they reflect some process of clinical or biological relevance. For each datatype, the user needs to decide which level of variable-grouping to apply.

This table provides some examples for different types of data. Consider the lung cancer example from the Introduction, and assume we measure: 34 clinical variables; 20,000 genes; 16 metabolites; and genetic mutations from whole-genome sequencing.

Choosing a similarity metric

Patient similarity can be measured in different ways for different types of input data. Here is a set of recommended metrics to start with, depending on what the type of data is and how many variables were grouped to create a given net.



netDx takes custom similarity functions for provided input

Summary table

Defining custom similarity functions

netDx is agnostic to the choice of similarity metric. Set the `simMetric` argument of `makePSN_NamedMatrix` to `custom` to provide a user-defined similarity metric. Be aware that the choice of similarity metric could increase false positives or false negatives during feature selection. Build controls to guard against these.

Expression data

This situation applies to a table of >5 values with continuous-valued measures. An example is a table of gene expression data, with ~20,000 measures per patient. Another is proteomic data with ~20 measures per patient.

Suggested metric: *Pearson correlation*. This is the default similarity metric for `makePSN_NamedMatrix()` so no special specification is required.

Note: Be sure to set `writeProfiles=TRUE`.

Fewer than 5 datapoints

Pearson correlation is not a stable measure of similarity when there are fewer than 5 variables per patient. An alternate similarity measure in such a situation is the **average normalized difference for each variable**.

$$S(a,b,G) = \frac{\sum_{i=1}^{k} \frac{abs(a_i - b_i)}{max(g_i) - min(g_i)}}{k}$$

```
#' Similarity by average of normalized difference
#'
#' @details Similarity measure for network with 2-5 variables.
#' Defined as the average of normalized difference for each of the
#' member variables
#' @param x (matrix) rows are patients, columns are values for component
#' variables
normDiff_avg <- function(x) {

    # normalized difference for a single variable
    normDiff <- function(x) {
        nm <- colnames(x)
        x <- as.numeric(x)
        n <- length(x)
        rngX  <- max(x,na.rm=T)-min(x,na.rm=T)

        out <- matrix(NA,nrow=n,ncol=n);
        # weight between i and j is
        # wt(i,j) = 1 - (abs(x[i]-x[j])/(max(x)-min(x)))
        for (j in 1:n) out[,j] <- 1-(abs((x-x[j])/rngX))
        rownames(out) <- nm; colnames(out)<- nm
        out
    }

    sim <- matrix(0,nrow=ncol(x),ncol=ncol(x))
    for (k in 1:nrow(x)) {
        tmp <- normDiff(x[k,,drop=FALSE])
        sim <- sim + tmp
        rownames(sim) <- rownames(tmp)
        colnames(sim) <- colnames(tmp)
    }
    sim <- sim/nrow(x)
    sim
}
```

Use in `netDx`: Given

- `dat`matrix with 5 variables (5xN matrix, where N is number of patients)
- `dat_names` vector with variable names
- `myGroup` list with the group name as key and members as variables,

then the call to create networks would be:

```
makePSN_NamedMatrix(dat,dat_names,myGroup,
            outDir,simMetric="custom",customFunc=normDiff_avg,
            sparsify=TRUE)
```

Note:

- `simMetric="custom"`

- `customFunc` points to the custom function definition
- `writeProfiles=FALSE`
- `sparsify=TRUE` *keep only the strongest edges for efficient memory use*

Range-based data (genetic mutations)

Creating patient data from genomic events such as genetic mutations or DNA copy number polymorphisms, requires a different design for creating PSN.

This section coming soon - 170927

Setting up to get an enrichment map

CHAPTER 5

Design the Predictor

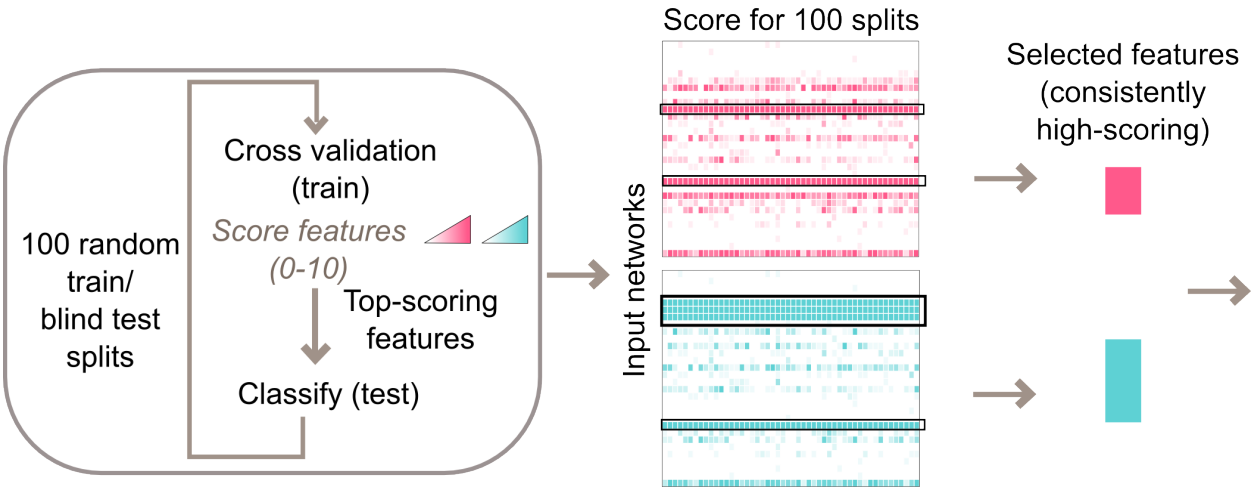
Nested cross-validation design

We recommend a nested cross-validation design for predictor building. **Inner loop:** Patient samples are split (e.g. 80:20) into a training and a blind test set. Using only the training samples, 10-fold cross validation is performed for each class, generating for each network a score between 0 and 10. Networks that scored 9 or 10 out of 10 are used to classify blind test samples. Note that this loop also generates a performance score for this particular blind test sample.

Outer loop: The inner loop is repeated (e.g. 100 times), which each loop using a new random split of train and blind test (100 trials).

The average of blind test classification across the 100 splits is used to measure overall predictor performance.

Feature-selected networks are those that score consistently well across all the outer loops. For example, a strict criterion would require networks to score score 10 out of 10 in all loops.

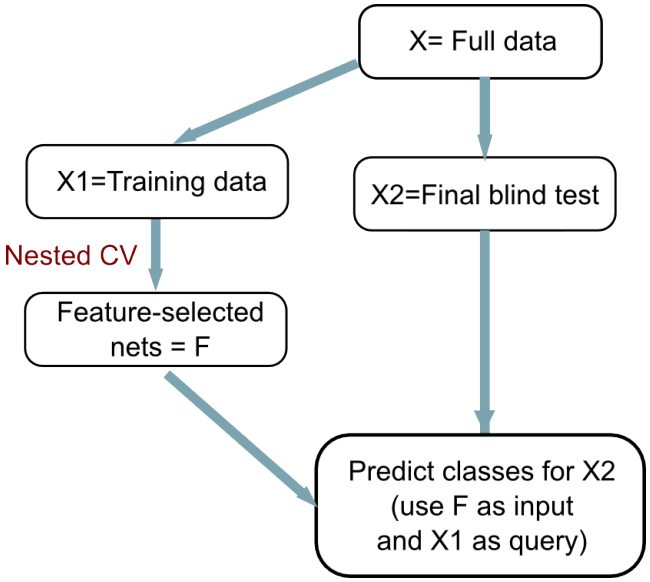


Example of nested cross-validation design in binary classification. Here an inner loop of 10-fold cross validation is run 100 times over different splits of data into train and blind test. At the end of the process, each class has a **network**

score table. This matrix (call it `S`) has K rows (where K is the number of networks) and 100 columns. `S[i,j]` has the cross-validation score for network `i` in loop `j`. The user then applies a cutoff to call **feature-selected networks**.

Overall workflow

Ideally a dataset would have sufficient samples to be set aside at the outset, before nested CV (figure below). Comparing predictor performance from the nested CV to that with the final blind test serves as a validation test.



CHAPTER 6

Output Files

This file describes netDx output files and formats

- *Directory tree of results after running netDx predictor*
  - *Example 1: Luminal A prediction from gene expresion and CNV: No resampling*
  - *Example 2: Predictor with three resamplings*
  - *Example 3: Predictor with nested cross-validation*
- *Structure of input database for cross-validation*
- **File formats:** [*_cont.txt][*.NRANK][*.PRANK][pathway_CV_score.txt or pathway_cumTally.txt][*.profile][*.query]

Overall directory tree

Assume the base directory for the dataset is `outRoot`.

Example 1: Luminal A predictfrom gene expression and CNV: No resampling

Here we see the output in the case where the predictor is run once, with a single round of 10-fold cross validation. An example is the vignette building a predictor for Luminal A type of breast cancer from TCGA data.

See code snippet 1 below for the full set of output files generated in this example. Let `N` be the total number of patients in the dataset, and let `N'` be number of training samples. GM = GeneMANIA, the algorithm used to integrate similarity networks and rank patients.

```
outRoot
    - profiles/  ### patient nets using training samples
        - *.profile : # input profiles to be converted to interaction nets
        - *_cont.txt: # interaction nets directly created by netDx
    - tmp/  # temporary files for GeneMANIA database (do not edit by hand)
        - GENES.txt: # list of patients in GM database
```

```
        - INTERACTIONS/*txt: # interaction nets as GM will see them
        - ... # (other files the user will not need)
    - dataset/ # location of indexed GeneMANIA database, not human-readable
    - LumA/ # results for class 1
        - GM_results
            - CV_*.query # GM query file for a fold of CV
            - CV_*.query-results.report.txt.PRANK # GM results file, patient ranking
            - CV_*.query-results.report.txt.NRANK # GM results file, network table
            - LumA_pathway_CV_score.txt # cumulative network score for 'LumA'␣
↪predictive
                            # value over entire CV
    - other/ # results for class 2
            - CV_*.query
            - CV_*.query-results.report.txt.PRANK
            - CV_*.query-results.report.txt.NRANK
            - other_pathway_CV_score.txt # cumulative network score for 'other'␣
↪predictive
                            # value over entire CV
```

Example 2: Predictor with 3 resamplings

The directory structure is similar to the previous example, with two notable differences:

1. results for each resampling are stored in their own subdirectory (e.g. `part1/`, `part2/`, etc.,)
2. there are different main directories for training vs test data. Training results are stored in `feature_select/` while test are stored in `networks_test/`.

In this example, we have a binary classifier and feature selection is only run for the class in question (rather than for A and not-A).

```
outRoot
    - feature_select/ # training results
        - networks_orig/     # similarity nets directly made by netDx
            - *_cont.txt # input similarity nets
        - part1/  # results for first round of resampling
            - GM_results/ # GeneMANIA input/output files
                - CV_*.query  # query file for a fold of cross-validation
                - CV_*.query-results.report.txt.PRANK # GM output: patient ranking
                - CV_*.query-results.report.txt.NRANK # GM output: network weights
                - pathwayScore.txt # cumulative net score for current resampling
            - networks/   # interaction nets
            - networksCliqueFilt/ # subset of networks that pass clique filtering
            - cliqueFilterNets.stats.txt # net score for clique-filtering
            - tmp/ # input files to create GM database
            - dataset/ # indexed GM database on which queries are run (not human-
↪readable)
        - part2/ # same structure as part1
        - part3/ # same structure as part1
        - pathway_cumTally.txt # cumulative net score for predictive class
```

Example 3: Predictor with nested cross-validation

In this design, the predictor has two loops for feature selection #. An outer loop where the data are split into several (say K1) train/blind test groups #. An inner loop where 10-fold cross validation is performed on each split. The result

is that there are K1 sets of network scores, each ranging from 0 to 10. There are also K1 sets of blind test predictions from which the mean and variation of predictor performance can be measured.

The output format recommended by netDx, and expected by downstream analysis code is as follows:

```
dataset_yymmdd/
  + rng1/
    + tmp/       # directory created by netDx, containing input data for GeneMANIA␣
↪database
    + networks/  # PSN created by calls to makePSN_NamedMatrix()
    +-- Class1
       + tmp/
       + networks/                              # networks for test classification␣
↪for this split
       + GM_results/                            # results of inner loop (10-fold CV)
       + Class1_pathway_CV_score.txt            # network scores for inner CV fold
       +--- CV_1.query                          # query for CV fold
       +--- CV_1.query-results.report.txt.NRANK  # network weights for CV fold
       +--- CV_1.query-results.report.txt.PRANK  # patient sim ranking for CV fold␣
↪(not used)
       ...
       +--- CV_10.query                          # query for CV fold
       +--- CV_10.query-results.report.txt.NRANK  # network weights for CV fold
       +--- CV_10.query-results.report.txt.PRANK  # patient sim ranking for CV fold␣
↪(not used)
    +-- Class2
    + predictionResults.txt  # test predictions for this train/test split
  + rng2/
  ...
  + rngK1/
```

Structure of input database for cross-validation

- Created by call to GM_createDB(). Serves as the input to create a generic GeneMANIA database.

Complete this section

```
tmp/
    - GENES.txt
    -   NETWORKS.txt
    - *.profile # tables from which interaction nets will be created
    + INTERACTIONS/
        - *.cont        # binary interaction nets
```

File formats

cont.txt

An individual patient similarity network, one per input feature. For example, if features are pathways, each net is named after a pathway (e.g. WNT_SIGNALING_NETWORK_cont.txt, XENOBIOTICS_cont.txt). Tab-delimited file without a header row; contains three columns: source patient, target patient, similarity weight.

Here is an example of TCGA patients that share CNVs in the same pathway. Patients without co-occurring CNVs are excluded.

```
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A2.A0CL.01A.11R.A115.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A2.A0D1.01A.11R.A034.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A2.A0YF.01A.21R.A109.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A7.A0D9.01A.31R.A056.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A8.A07O.01A.11R.A00Z.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A8.A081.01A.11R.A00Z.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A8.A083.01A.21R.A00Z.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A8.A08A.01A.11R.A00Z.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.A8.A08H.01A.21R.A00Z.07    1
TCGA.A1.A0SO.01A.22R.A084.07    TCGA.AN.A0AL.01A.11R.A00Z.07    1
```

Here is an example of a continuously-weighted network (e.g. based on gene expression correlation):

```
10      2       0.20394
15      2       0.23392
16      2       0.26524
27      2       0.2171
28      2       0.21943
54      2       0.18568
63      2       0.18642
74      2       0.15964
76      2       0.16618
78      2       0.17062
```

.NRANK

The part of a GeneMANIA query output that contains predictive weights of input networks. Tab-delimited file with a header row. The main columns of interest are 'Network' and 'Weight'.

```
Network Group   Network Weight  Title   Authors Year    Publication     PMID    URL
↪  Processing Method       Interactions    Source  Source URL     Tags
dummy_group             100.00
        RHO_GTPASES_ACTIVATE_CIT.profile        3.46
↪                       0
        NICOTINE_PHARMACODYNAMICS_PATHWAY.profile       3.02
↪                               0
        BIOCARTA_RANMS_PATHWAY.profile  2.80
↪                   0
        FOXM1_TRANSCRIPTION_FACTOR_NETWORK.profile      2.63
↪                               0
        KINESINS.profile        2.56
↪           0
        ASSOCIATION_OF_LICENSING_FACTORS_WITH_THE_PRE-REPLICATIVE_COMPLEX.profile
↪  2.44                                                         0

        DE_NOVO_PYRIMIDINE_DEOXYRIBONUCLEOTIDE_BIOSYNTHESIS.profile     2.36
↪                                               0
        AURORA_B_SIGNALING.profile      2.17
```

.PRANK

The part of a GeneMANIA query output that contains a table of patient rankings. Tab-delimited file with a header row. Contains three columns:

1. Gene: patient ID

2. Score: blank for patients submitted as the query. In decreasing order for patients. Higher scoring patients are more similar to the query.
3. Description: (ignore)

```
$  head CV_3.query-results.report.txt.PRANK
Gene    Score   Description
TCGA.A1.A0SH.01A.11R.A084.07            TCGA.A1.A0SH.01A.11R.A084.07
TCGA.B6.A0RN.01A.12R.A084.07            TCGA.B6.A0RN.01A.12R.A084.07
TCGA.BH.A0DP.01A.21R.A056.07            TCGA.BH.A0DP.01A.21R.A056.07
TCGA.AN.A0FS.01A.11R.A034.07            TCGA.AN.A0FS.01A.11R.A034.07
TCGA.E2.A1B4.01A.11R.A12P.07            TCGA.E2.A1B4.01A.11R.A12P.07
TCGA.A2.A0EO.01A.11R.A034.07            TCGA.A2.A0EO.01A.11R.A034.07
TCGA.BH.A0HA.01A.11R.A12P.07            TCGA.BH.A0HA.01A.11R.A12P.07
TCGA.A2.A0YI.01A.31R.A10J.07            TCGA.A2.A0YI.01A.31R.A10J.07
TCGA.A2.A0EX.01A.21R.A034.07            TCGA.A2.A0EX.01A.21R.A034.07

$ tail CV_3.query-results.report.txt.PRANK
TCGA.BH.A0B9.01A.11R.A056.07    7.20    TCGA.BH.A0B9.01A.11R.A056.07
TCGA.AR.A1AQ.01A.11R.A12P.07    7.19    TCGA.AR.A1AQ.01A.11R.A12P.07
TCGA.B6.A0RT.01A.21R.A084.07    7.17    TCGA.B6.A0RT.01A.21R.A084.07
TCGA.A2.A0D0.01A.11R.A00Z.07    7.17    TCGA.A2.A0D0.01A.11R.A00Z.07
TCGA.A7.A13D.01A.13R.A12P.07    7.16    TCGA.A7.A13D.01A.13R.A12P.07
TCGA.C8.A12K.01A.21R.A115.07    6.93    TCGA.C8.A12K.01A.21R.A115.07
TCGA.A2.A0YM.01A.11R.A109.07    6.87    TCGA.A2.A0YM.01A.11R.A109.07
TCGA.AR.A0U4.01A.11R.A109.07    6.86    TCGA.AR.A0U4.01A.11R.A109.07
TCGA.A8.A07O.01A.11R.A00Z.07    6.84    TCGA.A8.A07O.01A.11R.A00Z.07
TCGA.AN.A0XU.01A.11R.A109.07    6.65    TCGA.AN.A0XU.01A.11R.A109.07
```

.profile

Feature-level patient data that gets converted into a single similarity network. For example, the dataset contains gene expression and networks are at the level of a pathway, then each pathway (network) would have an input profile of gene expression limited to just the genes for that pathway. These are temporarily created by netDx and converts to similarity networks by calling the GeneMANIA function `ProfileToNetworkDriver`. Rows are patients and columns are the units grouped together for a network (e.g. genes in a pathway); values are the corresponding patient data. Tab-delimited file without a header. First column contains ID name for patient. In this example, the profile contains information for 3 units (columns) and 3 patients (rows).

```
PatientA    0.25    0.53   -1.224
PatientB     -1.540042        -1.092875        -1.249
PatientK     -0.290125        -0.212625        -0.825
```

.query

GeneMANIA query file as required by QueryRunner. File format (as described in the QueryRunner man page):

Note: the first line is hard-coded by netDx to be 'predictor'. Do not change this unless you understand what you are doing.

```
predictor
query-patient-1 [ \t query-patient-2 ... ]
networks
related-patient-limit
[ combining-method ]
```

Example:

```
predictor
TCGA.AO.A12G.01A.11R.A10J.07    TCGA.E2.A156.01A.11R.A12D.07    TCGA.C8.A132.01A.31R.
↪A115.07    TCGA.A8.A083.01A.21R.A00Z.07    TCGA.AO.A12E.01A.11R.A10J.07    TCGA.BH.
↪A18S.01A.11R.A12D.07    TCGA.E2.A14Q.01A.11R.A12D.07    TCGA.A2.A0EX.01A.21R.A034.
↪07    TCGA.AO.A12A.01A.21R.A115.07
all
232
automatic
```

pathway_CV_score.txt or pathway_cumTally.txt

This file contains feature-level scores following feature selection. *Note:* The pathway-centric terminology is outdated and will be replaced in future versions of netDx. The results will appear the same no matter what the features are, and they are not limited to pathways. Tab-delimited file with a header row. 'PATHWAY_NAME' and 'SCORE'.

```
PATHWAY_NAME    SCORE
CELLULAR_RESPONSES_TO_STRESS_cont.txt   19
NABA_ECM_AFFILIATED_cont.txt    16
COPII__COAT_PROTEIN_2__MEDIATED_VESICLE_TRANSPORT_cont.txt      10
SIGNALING_BY_WNT_cont.txt       13
SIGNALING_BY_TGF-BETA_RECEPTOR_COMPLEX_cont.txt 10
TRANSCRIPTIONAL_ACTIVITY_OF_SMAD2_SMAD3:SMAD4_HETEROTRIMER_cont.txt     10
L1CAM_INTERACTIONS_cont.txt     21
GAP_JUNCTION_ASSEMBLY_cont.txt  10
GAP_JUNCTION_TRAFFICKING_AND_REGULATION_cont.txt        20
```

CHAPTER 7

Interpreting Output

This page will tell you how to:

- look at enrichment map data.
- How to set cutoffs for consistently high-scoring networks.
  - how to read the integrated psn. computing dijkstra for evaluating group separation.

CHAPTER 8

Clique filtering

Clique filtering is a technique used to remove "random-like" networks when working with binary similarity and very sparse data. Data types include copy number variations, which are few per patient.

Motivation section TBA

Output format of `cliqueFilterNets.R`

A data.frame with per-network stats on clique-filtering:

- NETWORK: network name
- orig_pp: Num (+,+) interactions.
- orig_rest: Num (+,-) and (-,-) interactions
- ENR: Enrichment or bias of (+,+) interactions relative to other interactions. Specifically defined as `(orig_pp-orig_rest)/(orig_pp+orig_rest)`. Ranges between -1 (all non-(+,+)) to +1 (all (+,+)).
- TOTAL_INT: orig_pp+orig_rest
- numPerm: num permutations done in clique filtering
- shuf_mu: mean ENR of permuted nets (i.e. mean null ENR)
- shuf_sigma: standard deviation of ENR of permuted nets (i.e. s.d. of null ENR)
- Z: Z-score of ENR in real network, relative to null distribution
- pctl: Percentile of ENR in real network, relative to null distribution. Also the p-value
- Q: Benjamini-Hochberg corrected pvalue.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search