netDx: Interpretable patient classification using integrated patient similarity networks

Reproducing results with Docker

Author: Shraddha Pai <shraddha.pai@utoronto.ca>
Last updated: 28 November 2018

Table of Contents

Introduction

We have created a Docker image with an installation of **`netDx`**, its companion package **`netDx.examples`**, and all the code used to generate results in the manuscript Pai et al. (2018). "netDx: Interpretable patient classification using integrated patient similarity networks". In all instances, we have included results from previously run predictors, so these can be directly plotted*. Plotting functions will automatically use these pre-generated results.* If you are plotting with freshly-generated results, please alter the plotting code to pull data from the directory where your output was stored.

Note:

1) Running each predictor can take between 1 to 2 days, using 8 cores. Our tests were run on a 12-core hyperthreaded Intel Xeon 3GHz CPU with 62Gb RAM.

2) When copy-pasting commands from this manual, extra spaces can be introduced; these would need to be removed before commands are run.

Install Docker and Attach netDx Container

1. Install Docker for your operating system: https://www.docker.com/products/docker-desktop
2. Get the Docker image containing an installation of the R packages netDx, netDx.examples, and the examples run in the manuscript.
3. Load the Docker image. If your system doesn't directly load from the .gz file you may need to run "**`gunzip netdx_dock.tar.gz`**" and then load "**`netdx_dock.tar`**".

```
$ docker load --input netdx_dock.tar.gz
```

If the container correctly loads, you will get a message with the container name (e.g. "**`Loaded image:netdx_dock:version11`**") or, on some systems, a string with the container ID (e.g. "**`Loaded image ID: sha256:b96b10e57b586a2cb0c521484ded8df3632484f94c9d459c4b17b22e7bd4c4cd`**").

Check that the container has been loaded correctly:

```
$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
netdx_dock          version11           f72426aa548f        5 weeks
ago          4.71 GB
```

A Docker container can be "run" by either its name and version number, (**`netdx_dock:version11`**) or its image ID (here, **`f72426aa548f`**).

4. Start a container (instance) of the image.

```
$ docker run -it netdx_dock:version11
```

If the container correctly starts, your terminal prompt will change to the ID of the container to reflect that you are in the container's filesystem:

```
root@735027f918de:/examples#
```

In a separate terminal (reflecting your host machine's file system), you can see the registered container by using the '**`docker ps`**' command (only first 4 columns of output shown).

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
f3fa9f9c577d        netdx_dock          "/bin/bash"         2 days ago
```

The container ID is in the first column (highlighted)**. For illustration purposes we have used this ID throughout the tutorial, but the ID on your machine will likely be different.** Copy-paste your container ID wherever **<containerID>** is indicated.

5. You may want to have more than one terminal window connected to the same Docker container. This setup is useful for editing code in one window and running it in the other. To attach a second terminal to the Docker container, get the container ID using '**docker ps**' and then run this command:

```
$ docker exec -it <containerID> /bin/bash
```

Visit the Docker documentation pages for more information on available Docker commands: https://docs.docker.com

6. Once the container is loaded, run a quick test to make sure the netDx package works. For this you need to be in the "/examples" directory; your container will likely default to this directory. Run the example R script to make sure it is working. The R function runs a simple version of a binary breast tumour classifier that uses a single feature, a single train/test split and small number of resamplings. The predictor should take ~2-3 min to run.

```
root@3796db0af1a9:/examples# R
> source("BRCA_simple.R")
```

PanCancer Survival

Several models were tested for each tumour type. The table below lists the location of the code for each of these.

Note:
1) Running each predictor can take between 1 to 2 days, using 8 cores. Our tests were run on a 12-core hyperthreaded Intel Xeon 3GHz CPU with 62Gb RAM.
2) When copy-pasting commands from this manual, extra spaces can be introduced; these would need to be removed before commands are run.

Location of Code for Predictor-building

PanCancer: GBM	PanCancer/noPrune/GBM_noPrune_pipeline.R
PanCancer: GBM	PanCancer/diff_kernels/eucscale/GBM_eucscale_impute.R
PanCancer: GBM	PanCancer/diff_kernels/pearscale/GBM_pearscale_impute.R
PanCancer: KIRC	PanCancer/noPrune/KIRC_noPrune_pipeline.R
PanCancer: OV	PanCancer/noPrune/OV_noPrune_sp1_pipeline.R
PanCancer: OV	PanCancer/diff_kernels/eucscale/OV_eucscale_pipeline.R
PanCancer: OV	PanCancer/noPrune/OV_noPrune_pipeline.R
PanCancer: LUSC	PanCancer/noPrune/LUSC_noPrune_pipeline.R
PanCancer: LUSC	PanCancer/diff_kernels/pearscale/LUSC_top_pearscale.R

Example: Running a Predictor

Let us consider the example of running the *eucimpute* model for glioblastoma (GBM). This model integrates five different datatypes for GBM (clinical, RNA, DNAm, miRNA, and CNV) , generating one feature (network) per datatype. The PSNs in this model are generated using Euclidean dissimilarity followed by exponential scaling of the network.

This predictor is run as follows (note: container ID is provided for example purposes, you will need to run '**`docker ps`**' to find the ID on your machine):

```
root@f3fa9f9c577d:/ cd /examples/MSB-18-
8497/PanCancer/diff_kernels/eucscale

root@f3fa9f9c577d:/examples/MSB-18-
8497/PanCancer/diff_kernels/eucscale R

> source("GBM_eucscale_impute.R")
```

Results will be stored in: /examples/MSB-18-8497/PanCancer/output/GBM/eucclean_impute_maxEdge3000_top50_*yymmdd*

Compiling netDx Results and Comparing to Other Methods

This is done within the Docker container and assumes you have already attached the Docker container.

```
$ /examples/MSB-18-8497/PanCancer/plotResults
$ R
> source("compileRes.R"); # compiles results for PanCancer
> source("compare_Yuan.R") # boxplots comparing netDx to Yuan et al.
```

Results are in /examples/MSB-18-8497/PanCancer/output/results (.pdf and .txt files)

Breast Cancer binary classification

Building the Model and Evaluating Performance

This run takes 2.5 days at 4 cores and 13Gb RAM per core. Results from a previous run have been included at: /examples/MSB-18-8497/BRCA/output/BRCA_180818
(Code at: /examples/MSB-18-8497/BRCA/ BRCA_example.R).
To plot results for this run, see "Plotting Performance and Input for Enrichment Map" section.

Plotting Performance and Input for Enrichment Map

1. Run code (this has been already done)

```
$ cd /examples/MSB-18-8497/BRCA
$ R
> source("BRCA_plotResults.R")  ## make sure to change output dir
                              ## to your own if rerunning
```

2. Retrieve results:

```
$ docker ps # get container id
```

```
$ docker cp <containerID>:/examples/MSB-18-
8497/BRCA/output/BRCA_180818/plot .
```

3. Create Enrichment Map in Cytoscape (requires Cytoscape, EnrichmentMap app and AutoAnnotate app)

- Create a "generic" enrichment map. For the GMT file use **`LumA.gmt`**
- bottom right "number of edges" > select "Advanced" > Jaccard = 0.05

    - Click OK.
- Import node attributes to colour nodes by feature score:
- File > Import Table > **LumA_nodeAttrs.txt**
    - Set node fill by score.
- Use AutoAnnotate to assign node clusters and cluster labels ("pathway themes")

Correlating Features with Outcome

This code will create the table of feature-level correlation of the first 3 principal component projections with outcome. It is run inside the container:

```
root@f3fa9f9c577d: cd /examples/MSB-18-8497/BRCA/

root@f3fa9f9c577d:/examples/MSB-18-8497/BRCA# R

> source("BRCA_corrFeat.R")
```

To copy results to your current directory, from your host machine type the following (substitute *yymmdd* with the date on which your result files were generated):

```
$ docker cp <containerID>:/examples/MSB-18-
8497/BRCA/output/BRCA_180818/plot/ LumA_top10_PCtable_<yymmdd.pdf>  #
coloured matrix of correlations

$ docker cp <containerID>:/examples/MSB-18-
8497/BRCA/output/BRCA_180818/plot/ LumA_top10_PCview_<yymmdd>.pdf #
scatterplot of PC1 vs 2, 2 vs 3, and 1 vs 3 for each of the top
features listed in the table.
```

Generating Integrated PSN from Selected Features

In the docker container, run the following:

```
root@f3fa9f9c577d:# cd /examples/MSB-18-8497/BRCA

root@f3fa9f9c577d:/examples/MSB-18-8497/BRCA# R

> source("BRCA_plotPSN.R")
```

Then retrieve results on host machine:

```
$ docker cp <containerID>:/examples/MSB-18-
8497/BRCA/output/BRCA_180818/plot/BRCA_prunedNet_top0.40.txt .

$ docker cp <containerID>:/examples/MSB-18-
8497/BRCA/output/BRCA_180818/plot/pheno.txt .  # colour node by status
```

On a computer with Cytoscape installed, open Cytoscape. Import the network and view:

1. File > Import Network > **BRCA_prunedNet_top0.40.txt**
2. Layout > Settings …
    a. Layout Algorithm : Edge-Weighted Spring embedded layout
    b. Spring strength: 5
    c. Apply layout
3. Colour nodes by patient class:
    a. File > Import Table > pheno.txt
    b. Style > Node Fill > Discrete (by STATUS)
4. Clean up by Style > Edge Transparency > 10.

Asthma

This run takes 2.5 days using 4 cores and 13Gb RAM per core. Results from a previous run have been included at: /examples/MSB-18-8497Asthma_PBMC/output/basic_180820. To plot results for this run, see "Plotting Results and Generating Input for Enrichment Map" section.

Building the Model and Evaluating Performance

In Docker container:

```
root@f3fa9f9c577d:# cd /examples/MSB-18-8497/Asthma_PBMC
root@f3fa9f9c577d:/examples/MSB-18-8497/Asthma_PBMC# R
> source("netDx.R")
```

Plotting Results and Generating Input for Enrichment Map

In Docker container:

```
root@f3fa9f9c577d:# cd /examples/MSB-18-8497/Asthma_PBMC
root@f3fa9f9c577d:/examples/MSB-18-8497/Asthma_PBMC# R
> source("Asthma_plotResults.R")
```

Copy results to the host machine (shown for pre-generated results, showing example container ID):

```
$ docker cp  f3fa9f9c577d:/examples/MSB-18-
8497/Asthma_PBMC/output/basic_180820/plot .
```

Use Cytoscape to generate Enrichment Map of top-scoring pathways, using a method similar to that for the breast cancer pathway analysis.

GSEA LumA vs. other

Use expression data to rank genes by t-statistic (input for GSEA). From within the docker image, go to the directory with the R script that generates this input and run it.

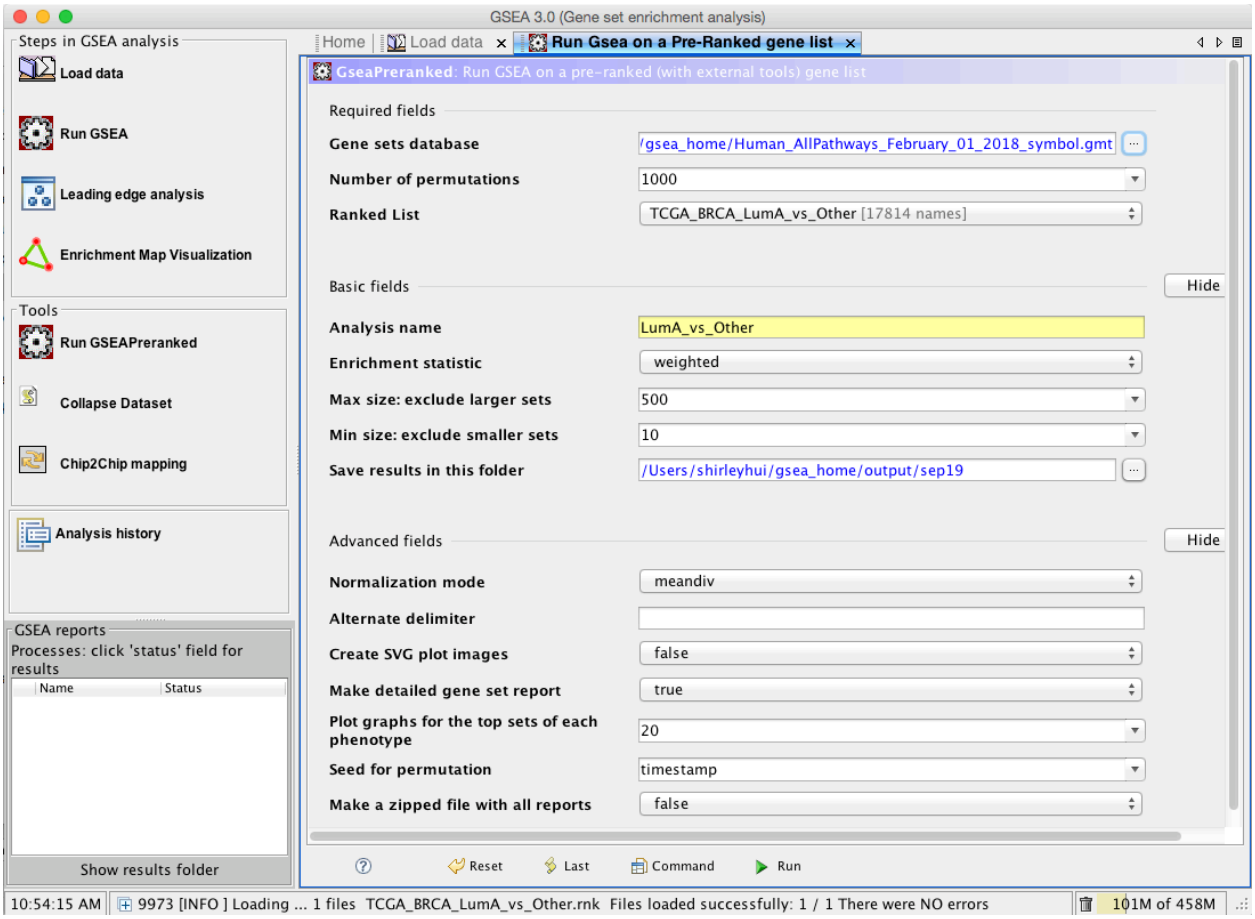
```
root@bcbd74a7503e:/# cd /examples/MSB-18-8497/BRCA/GSEA/Limma_files/
root@bcbd74a7503e:/examples/MSB-18-8497/BRCA/GSEA/Limma_files# R
> source("MakeRankFile.R")
```

If you don't have GSEA installed, download and install from
http://software.broadinstitute.org/gsea/downloads.jsp

Run GSEA in pre-ranked mode. For input you will need two files:
1) Gene-level statistics (**.rnk** file, generated by the call above), and
2) Pathway definitions (.gmt file).
On your computer, copy these input files from the Docker image to your working directory:

```
$ docker cp <containerID>:/examples/MSB-18-8497/BRCA/GSEA/Limma_files/
TCGA_BRCA_LumA_vs_Other.rnk .
$ docker cp <containerID>:/examples/MSB-18-8497/BRCA/data/
Human_AllPathways_February_01_2018_symbol.gmt .
```

Ensure the GSEA settings look like as follows before running:



Click "Run".

After GSEA results are generated, on a computer running Cytoscape, create the EnrichmentMap in Cytoscape using GSEA output. See tutorial for details: https://enrichmentmap.readthedocs.io/en/docs-2.2/Tutorial_GSEA.html

Comparison with DIABLO

Code to run LumA/other classification with DIABLO

NOTE: This code and associated data are provided for completeness, but have not installed *mixOmics* to limit the size of the Docker container. The user will need to install the R *mixOmics* package for this example to work (http://mixomics.org).

```
root@71d57173e17a:/# cd examples/MSB-18-8497/BRCA/DIABLO/
root@71d57173e17a:/examples/MSB-18-8497/BRCA/DIABLO# R
```

```
> source("BRCA_DIABLO.R")
```

netDx with pathway-level RNA and miRNA

For the DIABLO comparison, we designed a netDx predictor that uses pathway-level features for mRNA and miRNA.

This code maps miRNA to pathways.

```
root@71d57173e17a:/# cd examples/MSB-18-8497/BRCA/DIABLO/
root@71d57173e17a:/examples/MSB-18-8497/BRCA/DIABLO# R
> source("map_mir2path.R")
```

Running this code runs the netDx predictor for this design.

```
root@71d57173e17a:/# cd examples/MSB-18-8497/BRCA/DIABLO/
root@71d57173e17a:/examples/MSB-18-8497/BRCA/DIABLO# R
> source("BRCA_netDx_RNAmir.R")
```

Output is generated in: /examples/MSB-18-8497/BRCA/output/BRCA_DIABLO_2_*yymmdd* where *yymmdd* represent the date on which your script is run.

Pregenerated results can be found at: /examples/MSB-18-8497/BRCA/output/BRCA_DIABLO_2_180919 .

To plot results from pregenerated data,

```
root@71d57173e17a:/# cd examples/MSB-18-8497/BRCA/DIABLO/
root@71d57173e17a:/examples/MSB-18-8497/BRCA/DIABLO# R
> source("netDx_plotResults.R")
```

Modify the input directory in the above script to plot results for your run.

Output can be found in: /examples/MSB-18-8497/BRCA/output/BRCA_DIABLO_2_180919/plot.