20/20+ Documentation

Release 1.2.3

Collin Tokheim

Sep 07, 2020

Contents

1 Download **3**
- 1.1 20/20+ releases . . . 3
- 1.2 Necessary data files . . . 3
- 1.3 Pre-trained classifier . . . 4
- 1.4 Pan-cancer mutation data . . . 4
- 1.5 Example data . . . 4

2 Installation **5**
- 2.1 Releases . . . 5
- 2.2 Package requirements . . . 5
- 2.3 Check your PATH variable . . . 6

3 Quick Start **7**
- 3.1 Creating Features . . . 7
- 3.2 Prediction . . . 7

4 Tutorial **9**
- 4.1 Technical background . . . 9
- 4.2 20/20+ pipeline . . . 9
  - 4.2.1 Cancer type specific analysis . . . 10
  - 4.2.2 Pan-cancer analysis . . . 11
  - 4.2.3 Train a 20/20+ classifier . . . 13

5 FAQ **15**

6 Releases **17**

7 Citation **19**

Author Collin Tokheim

Contact ctokhei1 AT alumni.jh.edu

Source code GitHub

Q&A Biostars (tag: 2020+)

Next-generation DNA sequencing of the exome has detected hundreds of thousands of small somatic variants (SSV) in cancer. However, distinguishing genes containing driving mutations rather than simply passenger SSVs from a cohort sequenced cancer samples requires sophisticated computational approaches. 20/20+ integrates many features indicative of positive selection to predict oncogenes and tumor suppressor genes from small somatic variants. The features capture mutational clustering, conservation, mutation *in silico* pathogenicity scores, mutation consequence types, protein interaction network connectivity, and other covariates (e.g. replication timing). Contrary to methods based on mutation rate, 20/20+ uses ratiometric features of mutations by normalizing for the total number of mutations in a gene. This decouples the genes from gene-level differences in background mutation rate.

Contents:

CHAPTER 1

Download

1.1 20/20+ releases

- 2020plus v1.2.3 - 4/6/2019 - Minor change for installation procedure
- 2020plus v1.2.2 - 9/10/2018 - Added option to handle mutational data sets where silent mutations are not reported
- 2020plus v1.2.1 - 8/2/2018 - Fixed bug where configuration file would not load
- 2020plus v1.2.0 - 3/21/2018 - Change to null distribution simulation
- 2020plus v1.1.3 - 8/17/2017 - Bug fixes for different versions of rpy2
- 2020plus v1.1.2 - 7/3/2017 - Further bug fixes for latest versions of 20/20+ dependencies
- 2020plus v1.1.1 - 5/22/2017 - Bug fixes to work with newest versions of pandas
- 2020plus v1.1.0 - 11/21/2016 - Improved training procedure and added p-value diagnostic plots
- 2020plus v1.0.3 - 10/12/2016 - Fixed error in logging
- 2020plus v1.0.2 - 10/03/2016 - Fixed python3 conversion bug
- 2020plus v1.0.1 - 6/26/2016 - Added ability to run 20/20+ as a pipeline
- 2020plus v1.0.0 - 5/1/2016 - Initial release

1.2 Necessary data files

- Pre-computed scores data set
- Reference SNVBox transcripts in BED format

1.3 Pre-trained classifier

We have trained a 20/20+ classifier on pan-cancer data. This can be used to predict on cancer type specific mutations.

Current trained classifier (>= v1.1.0):

- 2020plus_10k.Rdata (NUMSIMULATIONS=10,000, default)
- 2020plus_100k.Rdata (NUMSIMULATIONS=100,000)

Trained classifier for versions 1.0.0-1.0.3 (old):

- 2020plus.Rdata

1.4 Pan-cancer mutation data

- full pan-cancer data set from:

Collin J. Tokheim, Nickolas Papadopoulos, Kenneth W. Kinzler, Bert Vogelstein, and Rachel Karchin. Evaluating the evaluation of cancer driver genes. PNAS 2016 ; published ahead of print November 22, 2016, doi:10.1073/pnas.1616440113

Details about how the mutations were filtered is available here.

1.5 Example data

- Example pan-cancer data set

CHAPTER 2

Installation

20/20+ is designed to run on *linux* operating systems.

2.1 Releases

20/20+ can be downloaded on github.

2.2 Package requirements

Once you have downloaded the source code for 20/20+, please add the directory to your `PATH` variable.

We recommend that you install the dependencies for 20/20+ through conda. Once conda is installed, setting up the environment is done as follows:

```
$ conda env create -f environment_python.yml  # install dependencies for python
$ source activate 2020plus  # activate the 20/20+ conda environment
$ conda install r r-randomForest rpy2  # install the R related dependencies
```

Every time you wish to run 20/20+, you will then need to activate the "2020plus" conda environment.

```
$ source activate 2020plus
```

The 20/20+ conda environment can also be deactivated.

```
$ source deactivate 2020plus
```

2.3 Check your PATH variable

Make sure that you have add the 20/20+ directory to your *PATH* variable. We recommend you add this line to your bashrc file.

```
export PATH=$PATH:/path/to/2020plus
```

Where "/path/to/2020plus" represents the path where **you** placed 20/20+. If you have done this correctly, the following command should print the location of the 2020plus.py script.

```
$ which 2020plus.py
```

CHAPTER 3

Quick Start

This quick start is only meant as a test to check whether 20/20+ has been **properly installed**. Please see the *Tutorial* for a more detailed example on the full pipeline that you can modify to use for your own data.

3.1 Creating Features

First, download the intermediate data files since the output from the probabilistic2020 package has already been prepared for you.

```
$ wget http://karchinlab.org/data/2020+/pancan_example.tar.gz
$ tar xvzf pancan_example.tar.gz
$ cd pancan_example
```

You do not have to concern yourself with how these files were generated for the purpose of this quick start. You should see, however, oncogene.txt for oncogene related features, tsg.txt for tumor suppressor related features, and the summary file named summary_pancan.txt.

To create the features, use the **features** sub-command for the 2020plus.py script.

```
$ python `which 2020plus.py` features \
     -og-test oncogene.txt \
     -tsg-test tsg.txt \
     --summary summary_pancan.txt \
     -o features_pancan.txt
```

3.2 Prediction

The command for prediction in this quick start example is for pan-cancer data encompassing many cancer types and samples. The **classify** sub-command performs the 20/20+ predictions of driver genes. This step needs the 20/20+ features file already created (features_pancan.txt), and the emprirical null distribution file to additionally report p-values/FDR. If an empirical null distribution file is not provided then only the random forest scores for prediction.

The **empricial null distribution** relates the classifier score to a p-value. An example null distribution **specific** to this pan-cancer dataset is simulated_null_dist.txt.

```
$ python `which 2020plus.py` --out-dir=result_compare classify \
    -f features_pancan.txt \
    -nd simulated_null_dist.txt
```

You should see the results in the result_compare/result/r_random_forest_prediction.txt file. It should be the same as the output already provided in result/ directory. Particularly, you should shoulde see 106 TSG scores, 64 oncogene scores, and 197 driver scores (208 unique genes) as significant at a Benjamini-Hochberg FDR of .1.

CHAPTER 4

Tutorial

4.1 Technical background

20/20+ uses the random forest algorithm to predict cancer driver genes. Because 20/20+ uses supervised machine learning, cancer driver gene prediction depends on a list of defined oncogenes and tumor suppressor genes used for training. The list is already pre-defined from the Cancer Genome Landscapes paper. Included in the results are a score specifically for oncogene or tumor suppressor gene, and a cumulative driver gene score. The score represents the fraction of decision trees in the random forest which voted for the particular class (oncogene, TSG, driver (either oncogene or TSG), passenger). Scores nearer one indicate stronger evidence for the gene as a cancer driver. There are two ways to use 20/20+ scores. One is to just obtain cancer driver scores for genes, which allows ranking of genes in a prioritized manner. The second way evaluates the statistical significance of the cancer driver score. A p-value and associated Benjamini-Hochberg false discovery rate will be reported, but this requires establishing the null distribution for random forest scores. 20/20+ uses an empirical null distribution by scoring simulated mutations in genes. This extra step requires additional work and computational resources. 20/20+ can be applied to pan-cancer and tumor type specific data. 10-fold cross-validation is performed internally within 20/20+ to avoid overfitting.

4.2 20/20+ pipeline

The easiest way to run the entire 20/20+ pipeline from somatic mutations to cancer driver gene prediction is to use snakemake. We have created a **Snakefile** that will run the multiple steps needed to get the final results. You first will need to install snakemake (requires python 3.X), so please see the snakemake installation instructions. You should be able to see the snakemake help, if you are in the 2020plus directory and it is installed correctly.

```
$ snakemake --help  # help using snakemake
$ snakemake -q help  # help for 2020plus commands
```

There are two ways to perform predictions with 20/20+. Either in a pan-cancer setting where mutations from several cancer types are aggregated together, or predicting cancer type specific driver genes by using a 20/20+ model previously trained on pan-cancer data.

4.2.1 Cancer type specific analysis

Preparing data

To set up this tutorial example you will first need to download the data. This step is needed regardless of whether your doing predictions on pan-cancer or cancer type specific data.

```
$ mkdir data
$ cd data
$ wget http://karchinlab.org/data/Protocol/bladder.txt.gz  # download mutations
$ wget http://karchinlab.org/data/2020+/snvboxGenes.bed  # download transcript␣
↪annotation
$ wget http://karchinlab.org/data/2020+/scores.tar.gz  # download pre-computed scores
$ wget http://karchinlab.org/data/2020+/2020plus_10k.Rdata  # download pre-computed␣
↪scores
$ gunzip bladder.txt.gz
$ tar xvzf scores.tar.gz
$ cd ..
```

The somatic mutations in the mutations.txt file is a MAF-like format described here. snvboxGenes.bed contains reference transcripts for each gene (described here), and the scores directory contains pre-computed scores (described here). You will also need to create a FASTA file of gene sequences by the following these instructions.

Running 20/20+

When performing predictions on cancer type specific mutations, a pre-trained 20/20+ classifier based on pan-cancer data is used to make predictions. Available pre-trained 20/20+ classifiers are shown on the *Download* page. Associated data should be collected like in the above *Preparing data* section. When using a pre-trained classifier, the **snakemake -s Snakefile pretrained_predict** command should be used. In the below example command, we use the command for a local machine, but it can be adopted to run on a cluster.

```
$ snakemake -s Snakefile pretrained_predict -p --cores 1 \
    --config mutations="data/bladder.txt" output_dir="output_bladder" trained_
↪classifier="data/2020plus_10k.Rdata"
```

The *–cores* argument specifies the number of computer cores that are allowable to be used at a given time. In this example, the output will be saved in the "output_bladder" directory as specified by the output_dir parameter (also changeable in config.yaml).

It is generally recommended to run 20/20+ on a cluster to parallelize calculations. The below command will execute the 20/20+ pipeline on an SGE computer cluster using qsub. The cluster submission command can be changed to fit your particular cluster scheduler.

```
$ snakemake -s Snakefile pretrained_predict -p -j 999 -w 10 --max-jobs-per-second 1 \
    --config mutations="data/bladder.txt" output_dir="output_bladder" trained_
↪classifier="data/2020plus_10k.Rdata" \
    --cluster-config cluster.yaml \
    --cluster "qsub -cwd -pe smp {threads} -l mem_free={cluster.mem},h_vmem={cluster.
↪vmem} -v PATH=$PATH"
```

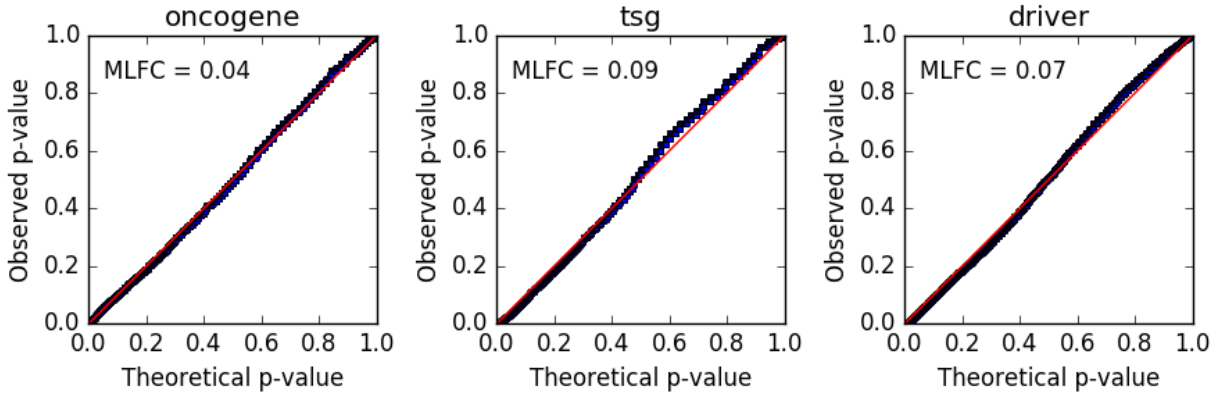
The **–cluster** argument specifies the command prefix for submitting to your cluster job scheduler. In the above example, **qsub** is used for the SGE scheduler, but this obviously is cluster specific and therefore you should look up the manual for your cluster. Of importance, though, is that certain template values can be inserted in to the job submission. Templated values are denoted by curly braces, and are used to set the number of threads ("{threads}") and memory ("{cluster.mem}" and "{cluster.vmem}"). Templated values with "cluster." are specified in the cluster config file (cluster.yaml; **–cluster-config** argument). It is also recommended that your PATH environmental variable is passed

into the cluster job submission so that you do not receive a command not found error. The "-j" argument can restrict the number of concurrent jobs submitted to the cluster,but in our case we use 999 to let the cluster job scheduler to identify which jobs get executed. The "-w 10 –max-jobs-per-second 1" parameters are issued to avoid overly quick job submissions to the cluster. The difference with the next pan-cancer tutorial is that the mutations ("data/bladder.txt") are from a single cancer type, and the pre-trained classifier is specified with the **trained_classifier** option. In this case the pre-trained 20/20+ classifier was assumed to be placed into the data directory.

Note: The run time of 20/20+ depends on the number of simulations. By default, the NUMSIMULATIONS is set at 10000, which is lower than used in the original 20/20+ paper. This can be increased via the NUMSIMULATIONS variable (e.g. from 10000 to 100000) in the *config.yaml* file or specification in the command line of snakemake via *–config NUMSIMULATIONS=100000*. This might result in a slight increase in prediction performance but may be too time consuming for large data. Make sure you use the correct trained classifier based on your NUMSIMULATIONS option, by using 2020plus_100k.Rdata for NUMSIMULATIONS=100000 and 2020plus_10k.Rdata for NUMSIMULATIONS=10000.

20/20+ output

Like in the quick start, you will find the result in output_bladder/results/r_random_forest_prediction.txt. There will be a p-value/q-value for the oncogene, tumor suppressor gene, and driver score. At a false discovery rate of 0.1, you should get 9 significant oncogene scores, 34 significant TSG scores, and 46 significant driver scores. The file will also contain all of the features used for prediction. Examine the QQ plot of p-values as a diagnostic check on the reported p-values (output_bladder/plots/qq_plot.png). You will need the matplotlib python package installed for the plot to be created (see installation instructions). The observed p-values (blue line) should be close to the theoretically expected p-values (red line). In this case, the mean absolute log2 fold change (MLFC) indicates that the p-values are in good agreement with expectations. Please see our paper for more discussion on the MLFC.



A mean absolute log2 fold change (MLFC) of greater than 0.3 may indicate problems with the null distribution. One cause of high MLFC are problems in the provided mutations. For example, if mutations of both a primary tumor and metastasis were provided or data arising from low quality mutation calls (e.g. caused by read mapability problems, etc.). Further quality control of the mutation data could fix the problem. In scenarios where mutation quality does not appear to be the cause, a more stringent false discovery rate threshold may be needed, or just rely on the random forest score without placing emphasis on the reported p-value.

4.2.2 Pan-cancer analysis

Note: The pan-cancer tutorial is more computationally intensive and the run time will take a while even on a computer cluster. However, it does demonstrate the correct usage of the 20/20+ pipeline, which is quicker for cancer type specific

data sets.

The **snakemake -s Snakefile predict** command will perform predictions on pan-cancer data. Here, it is assumed you are in the 2020plus directory where the Snakefile is located.

Preparing data

To set up this tutorial example you will first need to download the data. This step is needed regardless of whether your doing predictions on pan-cancer or cancer type specific data.

```
$ mkdir data
$ cd data
$ wget http://karchinlab.org/data/Protocol/pancan-mutation-set-from-Tokheim-2016.txt.
↪gz  # download mutations
$ wget http://karchinlab.org/data/2020+/snvboxGenes.bed  # download transcript␣
↪annotation
$ wget http://karchinlab.org/data/2020+/scores.tar.gz  # download pre-computed scores
$ gunzip pancan-mutation-set-from-Tokheim-2016.txt.gz
$ mv pancan-mutation-set-from-Tokheim-2016.txt mutations.txt  # rename file
$ tar xvzf scores.tar.gz
$ cd ..
```

The somatic mutations in the mutations.txt file is a MAF-like format described here. snvboxGenes.bed contains reference transcripts for each gene (described here), and the scores directory contains pre-computed scores (described here). You will also need to create a FASTA file of gene sequences by the following these instructions.

Running 20/20+

By default, the data is assumed to be located in the "data/" directory and mutations are "data/mutations.txt". You can change the default by editing the config.yaml file. However you can also override the default from the command line by specifying variables with the **–config** argument. The following command executes the 20/20+ on a local machine.

```
$ snakemake -s Snakefile predict -p --cores 1 \
    --config mutations="data/mutations.txt" output_dir="output_pancan"
```

The **–cores** argument specifies the number of computer cores that are allowable to be used at a given time. In this example, the output will be saved in the "output_pancan" directory as specified by the output_dir parameter (also changeable in config.yaml).

It is generally recommended to run 20/20+ on a cluster to parallelize calculations. The below command will execute the 20/20+ pipeline on an SGE computer cluster using qsub, like in the previous cancer type specific analysis. The cluster submission command can be changed to fit your particular cluster scheduler.

```
$ snakemake -s Snakefile predict -p -j 999 -w 10 --max-jobs-per-second 1 \
    --config mutations="data/mutations.txt" output_dir="output_pancan" \
    --cluster-config cluster.yaml \
    --cluster "qsub -cwd -pe smp {threads} -l mem_free={cluster.mem},h_vmem={cluster.
↪vmem} -v PATH=$PATH"
```

20/20+ output

Like in the quick start, you will find the result in output/results/r_random_forest_prediction.txt. There will be a p-value/q-value for the oncogene, tumor suppressor gene, and driver score. The file will also contain all of the features used for prediction.

4.2.3 Train a 20/20+ classifier

You can also train your own 20/20+ model to predict on new data (e.g. new cancer type specific data) using the **train** command. Training should be performed on a pan-cancer collection of mutations. This either could be those mutations used in our evaluation or a new collected set. Note, the provided pre-trained classifier on the downloads page is already trained on the mutations linked in the previous sentence. The file format for mutations is described here. Like above, the command can be easily modified to run on a cluster.

```
$ snakemake -s Snakefile train -p --cores 1 \
    --config mutations="data/my_pancancer_mutations.txt" output_dir="output"
```

where "data/my_pancancer_mutations.txt" is the file containing small somatic mutations and the trained 20/20+ model will be saved as "output/2020plus.Rdata".

CHAPTER 5

FAQ

Who should I contact if I encounter a problem?

If you believe your problem may be encountered by other users, please post the question on biostars. Check to make sure your question has not been already answered by looking at posts with the tag 2020+. Otherwise, create a new post with the 2020+ tag. We will be checking biostars for questions. You may also contact me directly at ctokhei1 AT alumni dot jh dot edu.

Can I use my own custom list of oncogenes/tumor suppressor genes for training?

Yes, you can change which genes are used for the training list of well-supported oncogenes and tumor suppressor genes. All you need is to change the gene names found in **data/gene_lists/oncogenes.txt** and **data/gene_lists/tsgs.txt**. This should only be done for advanced users as the training list of oncogenes and tumor suppressor genes were established from cancer experts.

How can I speed up the run time of 20/20+?

You can substantially speed up run time by reducing the number of simulations. This can be done by reducing the NUMSIMULATIONS variable (e.g. from 100000 to 10000) in the *config.yaml* file or specification in the command line of snakemake via *–config NUMSIMULATIONS=10000*. This might result in a slight decrease in prediction performance but may be waranted for large data.

What happens if silent mutations were not recorded in my data set?

Ocassionally, in the literature, studies may only report non-silent mutations from a sequencing study. If not accounted for, this may bias estimates of statistical significance. To make an adjustment for this problem, provide the *drop_silent* option via the command line: *–config drop_silent="yes"*.

CHAPTER 6

Releases

- 2020plus v1.2.3 - 4/6/2019 - Minor change for installation procedure
- 2020plus v1.2.2 - 9/10/2018 - Added option to handle mutational data sets where silent mutations are not reported
- 2020plus v1.2.1 - 8/2/2018 - Fixed bug where configuration file would not load
- 2020plus v1.2.0 - 3/21/2018 - Change to null distribution simulation
- 2020plus v1.1.3 - 8/17/2017 - Bug fixes for different versions of rpy2
- 2020plus v1.1.2 - 7/3/2017 - Further bug fixes for latest versions of 20/20+ dependencies
- 2020plus v1.1.1 - 5/22/2017 - Bug fixes to work with newest version of pandas
- 2020plus v1.1.0 - 11/21/2016 - Improved training procedure and added p-value diagnostic plots
- 2020plus v1.0.3 - 10/12/2016 - Fixed error in logging
- 2020plus v1.0.2 - 10/03/2016 - Fixed python3 conversion bug
- 2020plus v1.0.1 - 6/26/2016 - Added ability to run 20/20+ as a pipeline
- 2020plus v1.0.0 - 5/1/2016 - Initial release

CHAPTER 7

Citation

Collin J. Tokheim, Nickolas Papadopoulos, Kenneth W. Kinzler, Bert Vogelstein, and Rachel Karchin. Evaluating the evaluation of cancer driver genes. PNAS 2016 ; published ahead of print November 22, 2016, doi:10.1073/pnas.1616440113