Introduction

February 28, 2017

0.1 Cancer driver gene evaluation

0.1.1 About

A major goal of the huge public investment in large-scale cancer sequencing has been to find the majority of driver genes. Robust computational prediction of drivers from small somatic variants is critical to this mission, and it is essential that the best methods be identified. While many such methods have been proposed, it has been difficult to evaluate them because there is no gold standard to use as a benchmark. Here we developed an evaluation framework for driver gene prediction methods that does not soley rely on a gold standard. The framework includes a large set of small somatic mutations from a wide range of cancer types and five evaluation metrics. We propose it can be used to systematically evaluate new prediction methods and compare them to existing methods. The evaluations include (click links below to view):

1. Fraction overlap of predicted driver genes with the Cancer Gene Census (CGC)
2. Method concensus
3. P-value distribution
4. Prediction consistency
5. Number of predicted driver genes

The number of predicted driver genes can be informative along side other factors to understand whether there is an inflation of false positive driver genes or particularly low power. Due to both extremes being unfavorable, we do not include it as a measurement for ranking each method. The average ranking of a method on measures 1-4 can be used to assess relative performance.

0.1.2 Jupyter notebooks

We have prepared jupyter notebooks to help analysis of our cancer driver gene evaluation framework. You can use these notebooks and associated data to look at 8 methods which we have already analyzed, but also potentially compare additional new methods to these established results. For running your own methods, you will need to execute the method yourself on the same mutations. In the following sections, we go through setting up data for executing the evaluation protocol for cancer driver genes.

Download Be sure to be in the directory you would like to use for this protocol. The first step is to obtain the example data. You can use the `wget` command in bash to download the example data. On Mac, you may need to first install the `wget` command via HomeBrew by `brew install wget`. In the following command we assume you are in the top-level directory. On windows you

will need to download the file from the URL (without wget) and place it in the jupyter notebook directory.

```
$ wget http://karchinlab.org/data/Protocol/CGC-freeze-download-date-20160401.tsv
$ wget http://karchinlab.org/data/Protocol/example_data.tar.gz
$ tar xvzf example_data.tar.gz
```

Data format In the `example_data` folder, there are two folders which contain input data: `pancan` and `consistency_data`. The pancan folder contains the output files from each method. The first column should always be the gene name, and contain a column for p-values and Benjamini-Hochberg q-values. The results of several methods should be in the same directory named by the convention `METHODNAME.txt`. For example, `pancan/TUSON.txt` is the output from the TUSON method. Of the 5 evaluation criteria, 4 use the results on the full data set (here, the pancan folder). The last criterion compares the consistency of predicted driver genes when the data is split randomly in half. The `consistency_data` folder contains the prediction results from each method on these random splits. The consistency folder has a sub-directory for results in each repetition of the random split, in this case `consistency_data/Iteration_0` to `consistency_data/Iteration_9`. The results for the first half of the split is saved as `consistency_data/Iteration_0/METHODNAME_1.txt` while the second half is `consistency_data/Iteration_0/METHODNAME_2.txt`.

Configuration file Often methods will have different column names for the p-value/q-values. By default these jupyter notebooks assume the p-value column name is "pvalue", and the q-value column names is "qvalue". This can be changed through a YAML configuration file. In this tutorial, we have already provided you with a working configuration file (config.yaml). Although, this file can be tweaked and/or further extended for additional methods. The configuration file has the following format:

```
METHODNAME:
    qvalue:
        - qvalCol1
        - qvalCol2
    pvalue:
        - pvalCol1
        - pvalCol2
    consistency:
        - col1
        - col2
```

METHODNAME is the name of the method, which should match the file naming convention described in the data format above. In this example, the method reports two p-values (column names: "pvalCol1" and "pvalCol2") and two q-values (column names: "qvalCol1" and "qvalCol2"). In the common case that a method reports one p-value/q-value column, then only one bullet point for each attribute would be needed.