HW1

September 12, 2021

1 HW1. Driver discovery

In this problem set, you will implement several iterations of a driver discovery algorithm and examine their performance on simulated and real data. Code snippets are provided as a jumping off point, but are not necessary to use (and if desired you may do this problem set in a language other than python). Code should be well documented and easy to follow, and should be runnable as a single notebook or script.

Included are the following files:

1.1 MEL_simulated_mutations_uniform_model.maf

Simulated data assuming passenger mutations accumulated at some fixed probability per base (which you will need to infer), in addition to spiked-in driver mutations in the cancer genes reported for Melanomas on http://tumorportal.org/ (MEL dataset). Each line corresponds to a single mutation, and the following information is provided:

- The gene in which the mutation occurred
- The patient in which the mutation was found
- What the protein coding effect of the mutation was. For simplicity all non-silent mutations are classified as missense.

1.2 MEL.maf

A list of mutations observed in melanomas sequenced in Berger et al. *Cell* (2012) as downloaded from http://tumorportal.org/. Additional information given about these mutations include:

- classification: Whether the mutation was a point mutation (SNP), deletion (DEL), insertion (INS), dinucleotide mutation (DNP), or trinucleotide mutation (TNP).
- chr: the chromosome on which the mutation occured.
- pos: the chromosomal position at which the mutation occurred.
- ref_allele: The reference genome base at the position of the mutation.
- newbase: The nucleotide to which the base was changed after the mutation.

1.3 exome.coverage.txt

This file gives the number of bases at risk for each gene for a given type of mutation resulting in a particular effect. For example, the first line tells us the gene A1BG has 239 A bases at which an A->C mutation would result in a nonsilent mutation. If you summed all the coverage values for a given gene you would get a value 3× the covered gene length (since each base can be mutated in

3 possible ways). A covered base in one in which we sequence enough reads covering it to be able to make a mutation call if one exists. This file is derived from the coverage file used by MutSigCV (Lawrence et al., *Nature* (2013)) https://software.broadinstitute.org/cancer/cga/mutsig).

1.4 gene.covariates.txt

Gene-level covariates of mutation rate, again sourced from MutSigCV, include expression data averaged across cancer cell lines from different tumor types, replication timing data, and a Hi-C derived measure of chromatin openness.

[ ]:

```python
# Some possibly useful imports (if using python)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom, poisson, beta, betabinom
```

1.5 Question 1. Exploring the dataset

First let's get a feel for the general scope of parameters in the genome and this Melanoma dataset. Calculate/plot the following:

- The number of genes with ≥ 1 mutation in the dataset
- The number of patients in the dataset
- A histogram of the number of mutations per patient across the dataset (in log10 scale). What is the median number of mutations for a patient?
- The ratio of non-silent:silent mutations in the dataset
- The 25 most highly mutated genes (in terms of absolute number of mutations)
- The 25 longest genes in the coverage file

[3]:

```python
## Load maf file
m = pd.read_csv('MEL.maf',sep='\t')
m.head()
```

[3]:

```
  ttype patient      gene classification               type  chr        pos  \
0   MEL   ME001  C1orf127            SNP  Missense_Mutation    1   11024199
1   MEL   ME001   TNFRSF4            SNP             Silent    1    1147121
2   MEL   ME001      NRAS            SNP  Missense_Mutation    1  115256530
3   MEL   ME001  C1orf158            SNP             Silent    1   12815748
4   MEL   ME001   PPIAL4G            SNP  Missense_Mutation    1  143767710

  ref_allele newbase  context65  cons46
0          G       C         42    54.0
1          C       T         22    51.0
2          G       T         48    99.0
3          G       A         41    39.0
4          G       A         37    59.0
```

[4]:

```python
C = pd.read_csv('exome.coverage.txt',sep='\t')
C.head()
```

[4]:

```
   gene mutation     effect  coverage
0  A1BG     A->C  nonsilent       239
1  A1BG     A->C     silent        26
2  A1BG     A->G  nonsilent       211
3  A1BG     A->G     silent        54
4  A1BG     A->T  nonsilent       236
```

1.6 Question 2. Driver discovery on simulated data

1.6.1 (a) Now let's start to develop a simple driver discovery model. Using the simulated data provided, implement and apply a model that assumes a uniform mutation rate across all patients and bases of the genome similar to the example discussed in class. First, infer the mutation rate parameter from silent mutations. Then test whether we see more non-silent mutations than expected by chance. You should return a dataframe with the following values for every gene:

- The expected number of non-silent mutations in the gene
- A p-value with the probability of seeing the observed number of nonsilent mutations (or more) by chance using your inferred background mutation rate
- A Bonferonni FWER corrected p-value
- A Benjamini-Hochberg FDR corrected p-value

[ ]:

```python
## Load simulated maf file
m_sim = pd.read_csv('MEL_simulated_mutations_uniform_model.maf',sep='\t')
m_sim.head()
```

[ ]:

```python
# Inputs:
   # m: maf file
   # C: coverage file
def uniform_significance_model(m,C):

    # Implement your driver discovery algorithm:
    ## - Infer the mutation rate from silent mutations
    ## - Calculate probability of seeing observed number of nonsilent mutations (or greater) per gene
    ## Return dataframe with the results per gene
    pass
```

[ ]:

```python
results = uniform_significance_model(m_sim,C)
```

1.6.2 (b) Make a Q-Q plot of the results. Color ground truth drivers

```
[ ]: # Here's the ground truth list of drivers spiked in, and their frequencies.
     # Do not use any part of this file in your significance model!!!!!
     drivers = pd.read_csv('MEL_mutsig_gene_frequencies.txt',sep='\t')
     drivers.head()
```

```
[ ]: def qq(results):
         # Implement
         pass
```

```
[ ]: qq(results)
```

1.6.3 (c) Calulate your false positive and negative rate with and without Bonferonni FWER correction. Do the same with Benjamini-Hochberg FDR correction.

1.6.4 (d) Using your inferred background mutation rate, calculate your power to detect driver mutations as a function of the number of patients observed. Draw such curves for drivers at 5%, 10%, 20%, and 50% frequencies. Assume the gene which you are testing is of length 1.5kb and that 2/3 of the possible mutations would cause a nonsilent effect. For simplicity you may assume we plan to use Bonferonni correction (at a threshold of 10%) rather than FDR.

1.7 Question 3. Driver discovery on real data

```
[2]: ## Load maf file
     m = pd.read_csv('MEL.maf',sep='\t')
     m.head()
```

```
[2]:   ttype patient      gene classification              type  chr        pos  \
     0   MEL   ME001  C1orf127            SNP  Missense_Mutation    1   11024199
     1   MEL   ME001   TNFRSF4            SNP             Silent    1    1147121
     2   MEL   ME001      NRAS            SNP  Missense_Mutation    1  115256530
     3   MEL   ME001  C1orf158            SNP             Silent    1   12815748
     4   MEL   ME001   PPIAL4G            SNP  Missense_Mutation    1  143767710

       ref_allele newbase  context65  cons46
     0          G       C         42    54.0
     1          C       T         22    51.0
     2          G       T         48    99.0
     3          G       A         41    39.0
     4          G       A         37    59.0
```

1.7.1 (a) Taking the real Melanoma data, again test significance with your already implemented model. Make a Q-Q plot as before. Comment on whether the results seem reasonable. If you have "new" drivers, explore several on tumorportal.org (include screenshots with your submission) and discuss whether you think they are genuine drivers or if not, what factors could be leading to false positives.

1.7.2 (b) Take the genomic covariates, and perform an exploratory analysis to examine their relationship with mutation rate in this dataset. Which are positively correlated? Which negatively? Hint: you may need to bin or smooth genes of similar covariate values for clearest vizualization.

1.7.3 (c) Implement a new model that uses the genomic covariates to better estimate the mutation rate. Feel free to use regression models implemented by packages such as sklearn or statsmodels (but do not use any genomics-specific packages). Make sure to do any appropriate pre-processing to covariates before using them. Perfect performance is not expected, but inclusion of covariates should yield an improvement over the baseline model.

[ ]:
```
X = pd.read_csv('gene.covariates.txt',sep='\t').set_index('gene')
X.head()
```

[ ]:
```
# Inputs:
   # m: maf file
   # C: coverage file
   # X: Genomic coveriates
def pergene_significance_model(m,C,X):

    # Implement your driver discovery algorithm:
    ## - Infer necessary mutation rate parameters from silent mutations
        ## - e.g. Fit a regression model that predicts the mutation rate for␣
 ↪each gene using the covariates
    ## - Calculate the probability of seeing observed number of nonsilent␣
 ↪mutations (or greater) per gene
    ## using your gene-specific mutation rate estimates
    ## Return dataframe with the results per gene
    pass
```

[ ]:
```
results_pergenemodel = pergene_significance_model(m,C,X)
```

1.7.4 (d) Again make Q-Q plots and examine the results. Is your test well calibrated? What is the overlap with previously reported MutSig genes? Investigate several disagreeing genes on tumorportal.org and discuss whether you think they are real drivers or false positives (Note, the MutSig list likely contains false positives as well!). Discuss (but you do not need to implement) improvements to the model that might help further refine your driver list.

[ ]: