HW1

September 10, 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 accumulate at some fixed probability per base (which you don't know), and spiking in driver gene mutations in the genes observed in MEL on tumorportal.org. 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 TCGA as downloaded from 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. This file is derived from the covarage file used by MutSigCV (https://software.broadinstitute.org/cancer/cga/mutsig).

1.4 gene.covariates.txt

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

openness.

[1]:
```
# 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. Driver discovery on simulated data

1.5.1 (a) Using the simulated data provided, implement a simple driver discovery model that assumes a uniform mutation rate across all patients and bases of the genome. This mutation rate parameter should be inferred from silent mutations, and the model should return a dataframe summarizing the results of the test. Your tool should create a dataframe with the following values for every gene:

- A p-value testing the null hypothesis of seeing the number of nonsilent mutations (or more) by your inferred background mutation rate
- A bonferonni FWER corrected p-value
- A benjamini-hochberg FDR corrected p-value

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

[2]:

	gene	patient	type
0	AADACL3	Patient0	Missense_Mutation
1	AAK1	Patient0	Missense_Mutation
2	ABCA1	Patient0	Silent
3	ABCA12	Patient0	Missense_Mutation
4	ABCA13	Patient0	Silent

[3]:
```
C = pd.read_csv('exome.coverage.txt',sep='\t')
C.head()
```

[3]:

	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

[4]:
```
# Inputs:
   # m: maf file
   # C: coverage file
def uniform_significance_model(m,C):

```

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

[5]:
```
results = uniform_significance_model(m,C)
```

1.5.2 (b) Make a qq plot of the results. Color ground truth drivers

[11]:
```
# 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()
```

[11]:

	gene	freq
0	BRAF	0.63
1	NRAS	0.23
2	BCLAF1	0.18
3	TP53	0.17
4	CDKN2A	0.15

[6]:
```
def qq(results):
    # Implement
    pass
```

[7]:
```
qq(results)
```

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

1.5.4 (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 is of length 1.5kb and that 3/4 of the possible mutations would cause a nonsilent effect. For simplicity you may assume we plan to use bonferonni correction rather than FDR.

[ ]:
```

```

1.6 Question 2. Real data

1.6.1 (a) Taking the real MEL data, again test significance with your already implemented model. Make diagnostic plots 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.

1.6.2 (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?

1.6.3 (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 (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.

[8]:
```
X = pd.read_csv('gene.covariates.txt',sep='\t').set_index('gene')
X.head()
```

[8]:

gene	expr	reptime	hic
A1BG	1621097.0	406.0	-25.0
A1CF	113129.0	613.0	19.0
A2BP1	2474.0	1138.0	-47.0
A2M	348389.0	364.0	17.0
A2ML1	300254.0	407.0	12.0

[9]:
```
# 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
    ## - Calculate probability of seeing observed number of nonsilent mutations
 ↪(or greater) per gene
    ## Return dataframe with the results per gene
    pass
```

[10]:
```
results_pergenemodel = uniform_significance_model(m,C)
```

1.6.4 (d) Again make diagnostic plots and 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. If the later, discuss (but you do not need to implement) improvements to the model that might help further refine your driver list.

[ ]: