annotation_tutorial

June 30, 2020

0.1 Example for creating biological annotations for BANNs

BANNs framework requires constructing an annotation mask in order to guide the model structure based on biological annotations. We create SNP-sets based on the genomic annotations of SNPs used in the model. SNPs that fall into the neighborhood of the same gene are grouped together into a SNP-set that corresponds to this gene. If we consider intergenic regions (based on user input), SNPs that fall into unannotated regions of the genome are grouped together into intergenic regions. For example, two SNPs that fall into the intergenic region between *gene A* and *gene b* would be grouped together into a intergenic region called *Intergenic_gene A_gene B*.

In this example, we show how to carry out the annotation and create mask matrices (and annotation dataframes that store information on which SNPsets contain which SNPs) using the annotation.py script provided.

We need two inputs for carrying out biological annotations:
1) A SNP list file of .map or .bim format
2) A gene range file

SNP list file: We expect .map formatted SNP lists to be tab-delimited files have four fields/columns (with no header): Chromosome, Variant ID, Morgans, Position(bp) . Our annotation script works with position information. If morgan information is missing, you can replace this column with 0s (zeros) or leave it empty. We expect a similar format from .bim files but with six fields/columns: Chromosome, Variant ID, Morgans, Position, Minor Allele, Major Allele. If the files contain a different number of columns, annotation will not carry out and you will receive a warning about the error.
To get more information on file formats, you can visit:
https://www.cog-genomics.org/plink2/formats#map
https://www.cog-genomics.org/plink2/formats#bim

Important Note: In order to obtain accurate results, please make sure the order of SNPs in the SNP List file correspond to the order of SNPs in the genotype matrix fed into the model because indices of variables matter for the model and annotations. This means if variant rs7412 is the first SNP in the genotype matrix (i.e. the first column of the genotype matrix corresponds to variant rs7412), then the first row of the SNP list should have information on rs7412.

Gene range file: We expect gene range file to be a tab-delimited file with four fields/columns (with no header): Chromosome, Start, End, GeneID. We expect that the Start and End coordinates will be in basepaires. To get more information on this file format, you can visit the "Gene range lists" section in :

https://www.cog-genomics.org/plink/1.9/resources
and download gene range files for human genome.

Example Files In this tutorial, we will work with the very small SNP list and gene range lists provided in /sampleData/TestSNPList.txt and /sampleData/TestGeneList.txt files for the sake of demonstration.
Here is what these files look like:

[1]:
```
SNPList_file = open("sampleData/TestSNPList.txt", "r")
GeneList_file = open("sampleData/TestGeneList.txt", "r")

print("Printing SNP list file contents (.map formatted): \n")
print(SNPList_file.read())
print("\n\n Printing Gene range list file contents: \n")
print(GeneList_file.read())
```

```
Printing SNP list file contents (.map formatted):

2       rs2     0       11
2       rs1     0       2
18      rs3     0       13
18      rs4     0       1021
18      rs5     0       1800
19      rs5     0       609
19      rs6     0       5227
19      rs7     0       10187
19      rs8     0       12148
X       rs9     0       1
X       rs10    0       392
X       rs11    0       1107
X       rs12    0       4331


 Printing Gene range list file contents:

18      1023    1803    Gene4
7       1267    9569    Gene2
19      9532    10187   Gene7
X       1       501     Gene8
19      587     791     Gene6
18      240     391     Gene3
X       2675    5092    Gene9
X       4061    9582    Gene10
18      21200   29080   Gene5
7       241     905     Gene1
```

0.1.1 Carrying out annotations

In this tutorial, we will create two files:
1) an annotation dataframe that contains information on which SNP-sets contain which SNPs and what the genomic coordinate of these SNP-sets are, saved in a tab-delimited .txt file.
This dataframe is sorted in ascending order based on the chromosomal location of the SNP-sets. Carried out by the *annotate()* function

2) an annotation mask matrix, which is a sparse matrix of 0s and 1s used to guide the model architecture, saved in a tab-delimited .txt file.
This is a matrix of size: (number of SNPs by number of SNPsets). An entry in the matrix at location (row i, column j) tells us whether SNP i is contained in SNP-set j (yes if 1, no if 0).
The rows of the mask matrix (which correspond to SNPs) are in the same order as the SNP list file, meaning the first row contains annotation information of the first SNP in the SNP list. The columns of the mask matrix (which correspond to SNP-set) are in the same order as the annotation dataframe, so columns correspond to SNP-sets in an ascending order based on chromosomal location. Carried out by the *getMaskMatrix()* function

Parameters There are a few parameters we require from the user to specify how to carry out the annotations and save the results, apart from giving the path to the input SNP list file and gene range list file:

output file: Both *annotate()* and *getMaskMatrix()* functions require us to specify this. It is supposed to be path to the .txt file where we want to save the results (either annotation dataframe or mask matrix).

intergenic: For *annotate()* function. This is a boolean parameter, that expects either of the True or False values. The default value is True.
If *True*, the annotation script creates intergenic SNP-sets and considers them when annotating the SNPs. If *False*, it simply groups SNPs into genes defined in the gene range list and all SNPs that are unannotated (in intergenic regions) are grouped into one SNP-set called "Unannotated". This "Unannotated" SNP-set is the last entry of the annotation dataframe and would be the last column of the mask matrix to be created from the annotation dataframe. If the gene range list provided by the user has all chromosomes, there will be no "Unannotated" SNP-set when this parameter is set to *True*. However, if there are chromosomes missing (as in the case of this example, just for demonstrations), SNPs in the missing chromosomes will still be grouped into the "Unannotated" SNP-set.

buffer: For *annotate()* function. This is an integer and the default value is 0.
It tells the annotation script how many basepairs to allow for as a buffer when considering a SNP-set neighborhood. If set to, let's say, 50000, then we would allow a 50kb window around SNP-sets when considering whether a SNP is in the SNP-set or not.

dropSingletons: For *annotate()* function. This is a boolean parameter, that expects either of the True or False values. The default value is False.
When annotating, it is possible that some SNP-sets will only contain one SNP. We call these SNP-sets "singletons". In this case, whether they should still be considered a "SNP-set" is debatable. If you set this parameter value to *True*, then these SNP-sets will be dropped from the annotation, and SNPs will be re-annotated with the remaining SNP-sets. Otherwise (if set at *False*), singleton SNP-sets will be kept in the model.

Creating the annotation dataframe:

[2]:
```
# Importing annotation.py from the src folder:
import sys
sys.path.insert(0, '../src/') #Need to do this for this notebook example since␣
 ↪the code lives in a different directory

from annotation import * # Import annotation code

#Changing back to the original directory:
sys.path.insert(0, '../examples_docs/')

#Specifying the path to the SNP list file and gene range list file:
path_to_SNPList="sampleData/TestSNPList.txt"
path_to_geneList="sampleData/TestGeneList.txt"

#Specifying the path to the file we want the save annotation dataframe into:
file_toSave_annotationDF="sampleData/TestAnnotationDF.txt"

#Carrying out the annotation:
annotationDF=annotate(path_to_SNPList, path_to_geneList,␣
 ↪outputFile=file_toSave_annotationDF,
                      intergenic=True, buffer=0, dropSingletons=False)

print(annotationDF)
```

```
100%|      | 10/10 [00:00<00:00, 241.81it/s]
100%|      | 13/13 [00:00<00:00, 689.29it/s]

You have chosen to annotate SNP-sets with intergenic regions and with a buffer
of 0bp
Creating Intergenic SNP-sets
Annotating SNP-sets with the corresponding SNPs
Saving annotation results to file sampleData/TestAnnotationDF.txt
                    GeneID Chromosome    Start      End   SNPindex  \
0   Intergenic_Gene3_Gene4         18    392.0   1022.0        [3]
1                    Gene4         18   1023.0   1803.0        [4]
2                    Gene6         19    587.0    791.0        [5]
3   Intergenic_Gene6_Gene7         19    792.0   9531.0        [6]
4                    Gene7         19   9532.0  10187.0        [7]
5         Downstream_Gene7         19  10188.0  12148.0        [8]
6                    Gene8          X      1.0    501.0    [9, 10]
7   Intergenic_Gene8_Gene9          X    502.0   2674.0       [11]
8                    Gene9          X   2675.0   5092.0       [12]
9                   Gene10          X   4061.0   9582.0       [12]
10             UnAnnotated        NaN      NaN      NaN  [1, 0, 2]

          VariantID
0             [rs4]
```

```
1             [rs5]
2             [rs5]
3             [rs6]
4             [rs7]
5             [rs8]
6       [rs9, rs10]
7            [rs11]
8            [rs12]
9            [rs12]
10  [rs1, rs2, rs3]
```

Showing annotation results with different parameters:

[3]:
```
#Carrying out the annotation:
annotationDF=annotate(path_to_SNPList, path_to_geneList,␣
 ↪outputFile=file_toSave_annotationDF,
                      intergenic=False, buffer=500, dropSingletons=True)

print(annotationDF)
```

```
100%|      | 13/13 [00:00<00:00, 859.38it/s]
100%|      | 13/13 [00:00<00:00, 838.38it/s]

You have chosen to annotate SNP-sets without intergenic regions and with a
buffer of 500bp
Annotating SNP-sets with the corresponding SNPs
Dropping SNP-sets that are singletons (containing only one SNP) and re-
annotating SNPs without them
Annotating SNP-sets with the corresponding SNPs
Saving annotation results to file sampleData/TestAnnotationDF.txt
        GeneID Chromosome   Start      End          SNPindex  \
0        Gene3         18   240.0    391.0              [2]
1        Gene4         18  1023.0   1803.0            [3, 4]
2        Gene6         19   587.0    791.0               [5]
3        Gene7         19  9532.0  10187.0               [7]
4        Gene8          X     1.0    501.0           [9, 10]
5        Gene9          X  2675.0   5092.0              [12]
6       Gene10          X  4061.0   9582.0              [12]
7  UnAnnotated        NaN     NaN      NaN  [1, 0, 6, 8, 11]

                    VariantID
0                       [rs3]
1                  [rs4, rs5]
2                       [rs5]
3                       [rs7]
4                 [rs9, rs10]
5                      [rs12]
```

```
6                       [rs12]
7  [rs1, rs2, rs6, rs8, rs11]
```

Creating the mask matrix from the annotation dataframe:

[4]:
```
mask_outputFile="sampleData/TestMask.txt"
mask = getMaskMatrix(path_to_SNPList, annotationDF, mask_outputFile)
print(mask)
```

```
100%|      | 8/8 [00:00<00:00, 3834.79it/s]

creating mask
Saving annotation mask to file sampleData/TestMask.txt in tab-delimited format
[[0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 1. 0.]]
```