annotate_brain_regulatory_variants

March 15, 2021

1 Introduction

The analysis provided in this notebook aims at selection and annotation of putative regulatory SNPs functional in human brain from whole-genome sequencing data (or targeted sequencing if putative regulatory regions were included as sequencing targets).

These are the main steps of the analysis:
1. Selection of SNPs located in promoters and non-promoter regulatory regions active in human brain.
2. Selection of rare SNPs, enriched in the analyzed group of samples.
3. Annotation with transcription factor motifs.
4. Assignment of target genes which expression could be affected by the selected SNPs.

2 Imports

[11]:
```
import pandas as pd
from scipy.stats import binom_test, spearmanr
from statsmodels.sandbox.stats.multicomp import multipletests
import pybedtools as pbt
```

[12]:
```
t1 = pd.to_datetime('today')
t1
```

[12]:
```
Timestamp('2021-03-15 09:52:00.718381')
```

3 Paths to input and output files and 3rd party software

Start by setting paths to input files, output folder and 3rd party software necessary to run this analysis.
Input files include your vcf file with variants and two interval_list files describing promoters and enhancers active in human brain. Common active promoters and common active enhancers identified and described in Stepniak et al are provided by default but you can replace them with your own regions files if you wish.
All intermediate output files will be saved to the output folder defined here.
If you use the VirtualBox Ubuntu image provided for this analysis the paths to software executables are already set.

```
[13]: ### Input files
      #INPUT_VCF = "data/test_variants_chr16.vcf.gz"
      INPUT_VCF = "data/Lib1-6.without113.norm.vcf.gz"
      PROMOTER_REGIONS = "data/brain_promoters_active.bed" #last column should␣
       ↪contain gene names, comma separated if promoters of several genes overlap
      ENHANCER_REGIONS = "data/brain_enhancers_active.bed" #last column should␣
       ↪contain gene names if enhancer is located inside gene, comma separated, "."␣
       ↪for intergenic enhancers = no gene overlaps

      ### Output folder and files
      OUTPUT = "output/"
      ANNOTATED_PROMOTER_SNPs = OUTPUT + "annotated_promoter_snps.csv"
      ANNOTATED_ENHANCER_SNPs = OUTPUT + "annotated_enhancer_snps.csv"

      ### Software and datasets paths
      GATK = "/home/researcher/Programs/gatk-4.1.9.0/gatk"
      ANNOVAR = "/home/researcher/Programs/annovar/"
      GTEX = 'data/GTEx_Analysis_2017-06-05_v8_RNASeQCv1.1.9_gene_median_tpm.gct'
      ENHANCER_ACTIVITY = 'data/h3k27ac_coverage_quantile_normalized.csv'
      GENE_EXPRESSION = 'data/transcrtpts_rnaseq_quantile_normalized.csv'
      CHROMATIN_CONTACTS = 'data/predicted_contacts.bed'
```

```
[14]: #Create output folder
      !mkdir $OUTPUT
```

```
mkdir: cannot create directory 'output/': File exists
```

4 Select biallelic SNPs located in promoters and enhancers

In the first step of the analysis biallelic SNPs located in promoter and enhancer regions are selected from the input .vcf files. Two vcf files are generated in this step: promoter_SNPs.vcf and enhancer_SNPs.vcf

```
[15]: count_before = !$GATK CountVariants -V $INPUT_VCF
      print("Number of variants in the input file:", count_before[-4])
```

```
Number of variants in the input file: 264271
```

```
[16]: select_logs = []
      count_logs = []
      for r, regions in [("promoter", PROMOTER_REGIONS), ("enhancer",␣
       ↪ENHANCER_REGIONS)]:
          command1 ="%s SelectVariants -V %s -L %s --select-type-to-include SNP␣
       ↪--restrict-alleles-to BIALLELIC -O %s%s_SNPs.vcf" % (GATK, INPUT_VCF,␣
       ↪regions, OUTPUT, r)
          print(command1)
          log1 = !$command1
```

```
    select_logs.append(log1)
    print("Done")

    command2 = "%s CountVariants -V %s%s_SNPs.vcf" % (GATK, OUTPUT, r)
    print(command2)
    log2 = !$command2
    count_logs.append(log2)
    print("Done")
    print("Number of biallelic SNPs in %s regions:" % r, log2[-4], "\n")
```

```
/home/researcher/Programs/gatk-4.1.9.0/gatk SelectVariants -V
data/Lib1-6.without113.norm.vcf.gz -L data/brain_promoters_active.bed --select-
type-to-include SNP --restrict-alleles-to BIALLELIC -O output/promoter_SNPs.vcf
Done
/home/researcher/Programs/gatk-4.1.9.0/gatk CountVariants -V
output/promoter_SNPs.vcf
Done
Number of biallelic SNPs in promoter regions: 83596

/home/researcher/Programs/gatk-4.1.9.0/gatk SelectVariants -V
data/Lib1-6.without113.norm.vcf.gz -L data/brain_enhancers_active.bed --select-
type-to-include SNP --restrict-alleles-to BIALLELIC -O output/enhancer_SNPs.vcf
Done
/home/researcher/Programs/gatk-4.1.9.0/gatk CountVariants -V
output/enhancer_SNPs.vcf
Done
Number of biallelic SNPs in enhancer regions: 71681
```

Terminal output from 3rd party software is stored in log variables. You can check them if you suspect that something could have gone wrong during the calculations:

```
[17]: select_logs[0]
```

```
[17]: ['09:52:11.596 INFO  NativeLibraryLoader - Loading libgkl_compression.so from
jar:file:/home/researcher/Programs/gatk-4.1.9.0/gatk-
package-4.1.9.0-local.jar!/com/intel/gkl/native/libgkl_compression.so',
 'Mar 15, 2021 9:52:11 AM
shaded.cloud_nio.com.google.auth.oauth2.ComputeEngineCredentials
runningOnComputeEngine',
 'INFO: Failed to detect whether we are running on Google Compute Engine.',
 '09:52:11.749 INFO  SelectVariants -
------------------------------------------------------------',
 '09:52:11.750 INFO  SelectVariants - The Genome Analysis Toolkit (GATK)
v4.1.9.0',
 '09:52:11.750 INFO  SelectVariants - For support and documentation go to
https://software.broadinstitute.org/gatk/',
 '09:52:11.750 INFO  SelectVariants - Executing as researcher@brain-reg-var on
```

```
Linux v5.8.0-44-generic amd64',
 '09:52:11.750 INFO  SelectVariants - Java runtime: OpenJDK 64-Bit Server VM
v11.0.10+9-Ubuntu-0ubuntu1.20.04',
 '09:52:11.750 INFO  SelectVariants - Start Date/Time: March 15, 2021 at 9:52:11
AM CET',
 '09:52:11.751 INFO  SelectVariants -
------------------------------------------------------------',
 '09:52:11.751 INFO  SelectVariants -
------------------------------------------------------------',
 '09:52:11.751 INFO  SelectVariants - HTSJDK Version: 2.23.0',
 '09:52:11.752 INFO  SelectVariants - Picard Version: 2.23.3',
 '09:52:11.752 INFO  SelectVariants - HTSJDK Defaults.COMPRESSION_LEVEL : 2',
 '09:52:11.752 INFO  SelectVariants - HTSJDK
Defaults.USE_ASYNC_IO_READ_FOR_SAMTOOLS : false',
 '09:52:11.752 INFO  SelectVariants - HTSJDK
Defaults.USE_ASYNC_IO_WRITE_FOR_SAMTOOLS : true',
 '09:52:11.752 INFO  SelectVariants - HTSJDK
Defaults.USE_ASYNC_IO_WRITE_FOR_TRIBBLE : false',
 '09:52:11.752 INFO  SelectVariants - Deflater: IntelDeflater',
 '09:52:11.752 INFO  SelectVariants - Inflater: IntelInflater',
 '09:52:11.752 INFO  SelectVariants - GCS max retries/reopens: 20',
 '09:52:11.752 INFO  SelectVariants - Requester pays: disabled',
 '09:52:11.753 INFO  SelectVariants - Initializing engine',
 '09:52:11.873 INFO  FeatureManager - Using codec VCFCodec to read file file:///
home/researcher/brain_regulatory_variants_tool/brain_reg_var/data/Lib1-6.without
113.norm.vcf.gz',
 '09:52:12.255 INFO  FeatureManager - Using codec BEDCodec to read file file:///
home/researcher/brain_regulatory_variants_tool/brain_reg_var/data/brain_promoter
s_active.bed',
 '09:52:12.454 INFO  IntervalArgumentCollection - Processing 25573839 bp from
intervals',
 '09:52:12.487 INFO  SelectVariants - Done initializing engine',
 '09:52:12.631 INFO  ProgressMeter - Starting traversal',
 '09:52:12.632 INFO  ProgressMeter -        Current Locus  Elapsed Minutes
Variants Processed  Variants/Minute',
 '09:52:22.718 INFO  ProgressMeter -       chr15:89751516              0.2
61000         362915.2',
 '09:52:25.726 INFO  ProgressMeter -       chrX:118345852              0.2
85470         391645.0',
 '09:52:25.726 INFO  ProgressMeter - Traversal complete. Processed 85470 total
variants in 0.2 minutes.',
 '09:52:26.128 INFO  SelectVariants - Shutting down engine',
 '[March 15, 2021 at 9:52:26 AM CET]
org.broadinstitute.hellbender.tools.walkers.variantutils.SelectVariants done.
Elapsed time: 0.24 minutes.',
 'Runtime.totalMemory()=713031680',
 'Using GATK jar /home/researcher/Programs/gatk-4.1.9.0/gatk-
```

```
package-4.1.9.0-local.jar',
 'Running:',
 '    java -Dsamjdk.use_async_io_read_samtools=false
-Dsamjdk.use_async_io_write_samtools=true
-Dsamjdk.use_async_io_write_tribble=false -Dsamjdk.compression_level=2 -jar
/home/researcher/Programs/gatk-4.1.9.0/gatk-package-4.1.9.0-local.jar
SelectVariants -V data/Lib1-6.without113.norm.vcf.gz -L
data/brain_promoters_active.bed --select-type-to-include SNP --restrict-alleles-
to BIALLELIC -O output/promoter_SNPs.vcf']
```

5 Annotate with allele frequencies from gnomAD genome

We will use ANNOVAR to annotate promoter and enhancer SNPs with population frequencies from the gnomAD genome resource.

[18]:
```
annovar_logs = []
for r in ["promoter", "enhancer"]:
    command = "perl %stable_annovar.pl %s%s_SNPs.vcf %shumandb/ -buildver hg38␣
 ↪-remove -protocol gnomad_genome -operation f -nastring . -vcfinput -out␣
 ↪%s%s_SNPs" % (ANNOVAR, OUTPUT, r, ANNOVAR, OUTPUT, r)
    print(command)
    log = !$command
    annovar_logs.append(log)
    print("Done")
```

```
perl /home/researcher/Programs/annovar/table_annovar.pl output/promoter_SNPs.vcf
/home/researcher/Programs/annovar/humandb/ -buildver hg38 -remove -protocol
gnomad_genome -operation f -nastring . -vcfinput -out output/promoter_SNPs
Done
perl /home/researcher/Programs/annovar/table_annovar.pl output/enhancer_SNPs.vcf
/home/researcher/Programs/annovar/humandb/ -buildver hg38 -remove -protocol
gnomad_genome -operation f -nastring . -vcfinput -out output/enhancer_SNPs
Done
```

[19]:
```
!ls -lrth $OUTPUT/
```

```
total 798M
-rw-rw-r-- 1 researcher researcher  471 Feb  4 13:55
promoter_SNPs.hg38_multianno.csv
-rw-rw-r-- 1 researcher researcher  288 Feb  4 13:56 promoter_SNPs.csv
-rw-rw-r-- 1 researcher researcher  92M Mar 12 10:38
promoter_SNPs.hg38_multianno.nomissing.vcf
-rw-rw-r-- 1 researcher researcher  80M Mar 12 10:38
enhancer_SNPs.hg38_multianno.nomissing.vcf
-rw-rw-r-- 1 researcher researcher 9.4M Mar 12 10:39
promoter_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf
-rw-rw-r-- 1 researcher researcher 136K Mar 12 10:39
```

```
promoter_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf.idx
-rw-rw-r-- 1 researcher researcher 7.1M Mar 12 10:39
enhancer_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf
-rw-rw-r-- 1 researcher researcher 136K Mar 12 10:39
enhancer_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf.idx
-rw-rw-r-- 1 researcher researcher 327K Mar 12 10:39
promoter_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.csv
-rw-rw-r-- 1 researcher researcher 250K Mar 12 10:39
enhancer_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.csv
-rw-rw-r-- 1 researcher researcher  60K Mar 12 10:39
promoter_rare_enriched_SNPs.bed
-rw-rw-r-- 1 researcher researcher  36K Mar 12 10:39
enhancer_rare_enriched_SNPs.bed
-rw-rw-r-- 1 researcher researcher 2.8M Mar 12 12:33
promoter_rare_enriched_SNPs_motifbreakR-scores.csv
-rw-rw-r-- 1 researcher researcher 807K Mar 12 13:49
enhancer_rare_enriched_SNPs_motifbreakR-scores.csv
-rw-rw-r-- 1 researcher researcher 165K Mar 12 13:52 annotated_promoter_snps.csv
-rw-rw-r-- 1 researcher researcher 143K Mar 12 13:52 annotated_enhancer_snps.csv
-rw-rw-r-- 1 researcher researcher  74M Mar 15 09:52 promoter_SNPs.vcf
-rw-rw-r-- 1 researcher researcher 1.5M Mar 15 09:52 promoter_SNPs.vcf.idx
-rw-rw-r-- 1 researcher researcher  64M Mar 15 09:52 enhancer_SNPs.vcf
-rw-rw-r-- 1 researcher researcher 1.9M Mar 15 09:52 enhancer_SNPs.vcf.idx
-rw-rw-r-- 1 researcher researcher  77M Mar 15 09:53 promoter_SNPs.avinput
-rw-rw-r-- 1 researcher researcher  81M Mar 15 09:55
promoter_SNPs.hg38_multianno.txt
-rw-rw-r-- 1 researcher researcher  92M Mar 15 09:55
promoter_SNPs.hg38_multianno.vcf
-rw-rw-r-- 1 researcher researcher  67M Mar 15 09:55 enhancer_SNPs.avinput
-rw-rw-r-- 1 researcher researcher  71M Mar 15 09:56
enhancer_SNPs.hg38_multianno.txt
-rw-rw-r-- 1 researcher researcher  80M Mar 15 09:56
enhancer_SNPs.hg38_multianno.vcf
```

Annovar has generated two *.hg38_multianno.vcf files which contain frequency annotations.

5.1 Select SNPs with MAF < 0.01

In the next step we will choose only rare SNPs - those with minor allele frequency (MAF) below 0.01 in all populations included in gnomAD genome.

First we will replace ".", which marks missing MAF values, with "100.". As a result all variants with missing frequency data will be filtered out in the next step. You can motify this behaviour by changing the value which is inserted instead of "." but it must be a float value for the filtering to work properly. For example if you would like to treat missing data as equal to very low frequency you may replace "100." with "0.0".

[20]:
```
annotations = ['gnomAD_genome_ALL',
               'gnomAD_genome_AFR',
               'gnomAD_genome_AMR',
               'gnomAD_genome_ASJ',
               'gnomAD_genome_EAS',
               'gnomAD_genome_FIN',
               'gnomAD_genome_NFE',
               'gnomAD_genome_OTH']
  for r in ["promoter", "enhancer"]:
      with open('%s%s_SNPs.hg38_multianno.nomissing.vcf' % (OUTPUT, r), 'w') as o:
          for line in open('%s%s_SNPs.hg38_multianno.vcf' % (OUTPUT, r)).
  ↪readlines():
              for el in annotations:
                  if el + '=.' in line:
                      line = line.replace(el + '=.', el + '=100.0')
              o.write(line)
```

The *hg38_multianno.nomissing.vcf files contain "." frequency values replaced by 100.0.

Now we select rare variants and save them in *.hg38_multianno.nomissing.gnomad_below_0.01.vcf files.

[21]:
```
select_rare_logs = []
for r in ["promoter", "enhancer"]:
    command = "%s SelectVariants -V %s%s_SNPs.hg38_multianno.nomissing.vcf" \
            " -select 'gnomAD_genome_ALL < 0.01'" \
            " -select 'gnomAD_genome_AFR < 0.01'" \
            " -select 'gnomAD_genome_AMR < 0.01'" \
            " -select 'gnomAD_genome_ASJ < 0.01'" \
            " -select 'gnomAD_genome_EAS < 0.01'" \
            " -select 'gnomAD_genome_FIN < 0.01'" \
            " -select 'gnomAD_genome_NFE < 0.01'" \
            " -select 'gnomAD_genome_OTH < 0.01'" \
            " -O %s%s_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf" %␣
↪(GATK, OUTPUT, r, OUTPUT, r)
    log = !$command
    select_rare_logs.append(log)
    print("Done")
```

```
Done
Done
```

Let's check how many variants have been selected.

[22]:
```
count_rare_logs = []
for r in ["promoter", "enhancer"]:
    command = "%s CountVariants -V %s%s_SNPs.hg38_multianno.nomissing.
↪gnomad_below_0.01.vcf" % (GATK, OUTPUT, r)
    print(command)
    log = !$command
    count_rare_logs.append(log)
    print("Done")
    print("Number of rare SNPs in %s regions:" % r, log[-4])
```

```
/home/researcher/Programs/gatk-4.1.9.0/gatk CountVariants -V
output/promoter_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf
Done
Number of rare SNPs in promoter regions: 8762
/home/researcher/Programs/gatk-4.1.9.0/gatk CountVariants -V
output/enhancer_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf
Done
Number of rare SNPs in enhancer regions: 6489
```

6 Choose SNPs enriched in analyzed cohort compared to chosen population

We will now use binomial test to choose SNPs enriched in out analyzed cohort compared to population. Since the variants anayzed in the test example are from Polish population gnomad_NFE (non-Finnish European) population is chosen but you can modify this option according to your needs.

First we need to reformat vcf files to csv to be able to read them with pandas. For this we will use the VariantsToTable tool from the GATK package. You can specify fields from the vcf which will be present in the csv. Here I choose information about SNP position, REF and ALT alleles, allele counts and frequencies in the analyzed cohort and gnomAD genome frequencies in global population (gnomAD_genome_ALL) and non-Finnish Europeans (gnomAD_genome_NFE).

```
[23]: totable_logs = []
for r in ["promoter", "enhancer"]:
    command = "%s VariantsToTable  -V %s%s_SNPs.hg38_multianno.nomissing.
 ↪gnomad_below_0.01.vcf " \
    "-F CHROM -F POS -F REF -F ALT -F AC -F AF -F AN " \
    "-F gnomAD_genome_ALL -F gnomAD_genome_NFE " \
    "-O %s%s_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.csv" % (GATK,␣
 ↪OUTPUT, r, OUTPUT, r)
    print(command)
    log = !$command
    totable_logs.append(log)
    print("Done")
```

```
/home/researcher/Programs/gatk-4.1.9.0/gatk VariantsToTable  -V
output/promoter_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf -F CHROM -F
POS -F REF -F ALT -F AC -F AF -F AN -F gnomAD_genome_ALL -F gnomAD_genome_NFE -O
output/promoter_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.csv
Done
/home/researcher/Programs/gatk-4.1.9.0/gatk VariantsToTable  -V
output/enhancer_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.vcf -F CHROM -F
```

```
POS -F REF -F ALT -F AC -F AF -F AN -F gnomAD_genome_ALL -F gnomAD_genome_NFE -O
output/enhancer_SNPs.hg38_multianno.nomissing.gnomad_below_0.01.csv
Done
```

Read generated csv files with pandas and inspect their contents.

[24]:
```
rare_promoter_snps = pd.read_csv("%s/promoter_SNPs.hg38_multianno.nomissing.
 ↪gnomad_below_0.01.csv" % OUTPUT, sep = '\t')
rare_promoter_snps.head()
```

[24]:
```
  CHROM      POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  gnomAD_genome_NFE  \
0  chr1    30570   C   T   2  0.067  30           0.000000             0.0000
1  chr1   939354   C   T   1  0.022  46           0.001200             0.0014
2  chr1   939561   G   A   2  0.043  46           0.000052             0.0000
3  chr1  1013784   C   T   1  0.022  46           0.000400             0.0006
4  chr1  1033651   A   C   1  0.026  38           0.000000             0.0000

   Unnamed: 9
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
```

[25]:
```
rare_enhancer_snps = pd.read_csv("%s/enhancer_SNPs.hg38_multianno.nomissing.
 ↪gnomad_below_0.01.csv" % OUTPUT, sep = '\t')
rare_enhancer_snps.head()
```

[25]:
```
  CHROM      POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  gnomAD_genome_NFE  \
0  chr1    20184   A   G   1  0.023  44           0.004300             0.0065
1  chr1    20254   G   A   1  0.023  44           0.004400             0.0029
2  chr1  1157576   G   C   1  0.022  46           0.000065             0.0001
3  chr1  1158496   C   T   1  0.022  46           0.002500             0.0040
4  chr1  1159286   C   G   1  0.022  46           0.004700             0.0074

   Unnamed: 9
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
```

Both csv files contain empty "Unnamed:9" column - remove it.

[26]:
```
for df in [rare_enhancer_snps, rare_promoter_snps]:
    for col in df.columns:
        if "Unnamed:" in col:
            df.drop(labels = col, axis=1, inplace = True)
```

```
rare_enhancer_snps.head()
```

```
[26]:   CHROM      POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  gnomAD_genome_NFE
      0  chr1    20184   A   G   1  0.023  44           0.004300             0.0065
      1  chr1    20254   G   A   1  0.023  44           0.004400             0.0029
      2  chr1  1157576   G   C   1  0.022  46           0.000065             0.0001
      3  chr1  1158496   C   T   1  0.022  46           0.002500             0.0040
      4  chr1  1159286   C   G   1  0.022  46           0.004700             0.0074
```

```
[27]: rare_promoter_snps.head()
```

```
[27]:   CHROM      POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  gnomAD_genome_NFE
      0  chr1    30570   C   T   2  0.067  30           0.000000             0.0000
      1  chr1   939354   C   T   1  0.022  46           0.001200             0.0014
      2  chr1   939561   G   A   2  0.043  46           0.000052             0.0000
      3  chr1  1013784   C   T   1  0.022  46           0.000400             0.0006
      4  chr1  1033651   A   C   1  0.026  38           0.000000             0.0000
```

Calculate p-values for one-sided binomial test in which the number of successes is equal to the number of ALT alleles in the cohort (AC), the number of trials is equal to the total number of identified alleles (AN) and probability of success is equal to population frequency of ALT allele (gnomAD_genome_NFE). The alternative hypothesis is that observed frequency is greater than expected.

```
[28]: def calc_binom_pval(row):
              x = row['AC']
              n = row['AN']
              p = float(row['gnomAD_genome_NFE'])

              return binom_test(x,n,p, alternative = 'greater')
```

```
[29]: for df in [rare_enhancer_snps, rare_promoter_snps]:
          df['binom_pval']=df.apply(calc_binom_pval, axis=1)

          #apply correction for multiple hypothesis testing with the␣
       ↪Benjamini-Hochberg procedure, use FDR = 0.01
          multipletests_correction = multipletests(df['binom_pval'], alpha=0.01,
                      method='fdr_bh', is_sorted=False, returnsorted=False)
          df['B-H_reject_H0'] = multipletests_correction[0]
          df['corrected_binom_pval'] = multipletests_correction[1]
```

```
[30]: rare_promoter_snps.head()
```

```
[30]:   CHROM      POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  gnomAD_genome_NFE  \
      0  chr1    30570   C   T   2  0.067  30           0.000000             0.0000
      1  chr1   939354   C   T   1  0.022  46           0.001200             0.0014
      2  chr1   939561   G   A   2  0.043  46           0.000052             0.0000
```

```
3  chr1  1013784   C   T   1  0.022  46           0.000400              0.0006
4  chr1  1033651   A   C   1  0.026  38           0.000000              0.0000

   binom_pval  B-H_reject_H0  corrected_binom_pval
0    0.000000           True              0.000000
1    0.062412          False              0.103377
2    0.000000           True              0.000000
3    0.027231          False              0.057433
4    0.000000           True              0.000000
```

[31]:
```python
rare_enhancer_snps.head()
```

[31]:
```
  CHROM      POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  gnomAD_genome_NFE  \
0  chr1    20184   A   G   1  0.023  44           0.004300             0.0065
1  chr1    20254   G   A   1  0.023  44           0.004400             0.0029
2  chr1  1157576   G   C   1  0.022  46           0.000065             0.0001
3  chr1  1158496   C   T   1  0.022  46           0.002500             0.0040
4  chr1  1159286   C   G   1  0.022  46           0.004700             0.0074

   binom_pval  B-H_reject_H0  corrected_binom_pval
0    0.249438          False              0.273069
1    0.119958          False              0.165396
2    0.004590          False              0.017124
3    0.168371          False              0.208415
4    0.289414          False              0.305077
```

Select SNPs significantly enriched in analyzed cohort at FDR = 0.01.

[32]:
```python
rare_enriched_promoter_snps =␣
 ↪rare_promoter_snps[rare_promoter_snps["B-H_reject_H0"]]
rare_enriched_enhancer_snps =␣
 ↪rare_enhancer_snps[rare_enhancer_snps["B-H_reject_H0"]]
print(len(rare_enriched_promoter_snps), "SNPs in promoters are enriched in␣
 ↪analyzed cohort.")
print(len(rare_enriched_enhancer_snps), "SNPs in enhancers are enriched in␣
 ↪analyzed cohort.")
```

```
1313 SNPs in promoters are enriched in analyzed cohort.
784 SNPs in enhancers are enriched in analyzed cohort.
```

7 Annotate with predicted TF binding sites

Use motifbreakR package and Hocomoco v11 full database of TF models to identify SNPs which may destroy or create a TF binding site.

We will first save SNPs to a bed file which will serve as input to motifbreakR.

[33]:
```
snps_bed_files = []
for snps_df, r in [(rare_enriched_promoter_snps, "promoter"),␣
↪(rare_enriched_enhancer_snps, "enhancer")]:
    snps_bed = pd.DataFrame()
    snps_bed["chromosome"] = snps_df["CHROM"]
    snps_bed["start"] = snps_df["POS"] - 1
    snps_bed["end"] = snps_df["POS"]
    snps_bed["name"] = snps_df["CHROM"] + ":" + snps_df["POS"].astype(str) + ":
↪" + snps_df["REF"] + ":" + snps_df["ALT"]
    snps_bed["score"] = 0
    snps_bed["strand"] = "+"
    output_bed_path = "%s%s_rare_enriched_SNPs.bed" % (OUTPUT, r)
    snps_bed.to_csv(output_bed_path, sep="\t", index=False, header=False)
    snps_bed_files.append(output_bed_path)
```

[34]:
```
!ls -lrth $output
```

```
total 780K
-rw-rw-r-- 1 researcher researcher 526K Feb 24 13:51
annotate_brain_regulatory_variants.html
-rw-rw-r-- 1 researcher researcher 168K Feb 24 16:49
annotate_brain_regulatory_variants.pdf
drwxrwxr-x 2 researcher researcher 4.0K Mar 12 10:33 data
-rw-rw-r-- 1 researcher researcher 1.4K Mar 15 09:51 install_dependencies.sh
-rw-rw-r-- 1 researcher researcher 6.9K Mar 15 09:51 README.md
drwxrwxr-x 2 researcher researcher 4.0K Mar 15 09:56 output
-rw-rw-r-- 1 researcher researcher  61K Mar 15 09:57
annotate_brain_regulatory_variants.ipynb
```

Quick look at one of the bed files:

[35]:
```
!head "$OUTPUT"promoter_rare_enriched_SNPs.bed
```

```
chr1    30569   30570   chr1:30570:C:T  0       +
chr1    939560  939561  chr1:939561:G:A 0       +
chr1    1033650 1033651 chr1:1033651:A:C        0       +
chr1    1033651 1033652 chr1:1033652:G:C        0       +
chr1    1033652 1033653 chr1:1033653:T:C        0       +
chr1    1033662 1033663 chr1:1033663:G:C        0       +
chr1    1033663 1033664 chr1:1033664:G:C        0       +
chr1    1034901 1034902 chr1:1034902:A:T        0       +
chr1    1117137 1117138 chr1:1117138:T:C        0       +
chr1    1470770 1470771 chr1:1470771:T:G        0       +
```

[36]:
```
%load_ext rpy2.ipython
```

To analyze our SNPs with motifbreakR we load hg38 as a reference genome and we choose all human TF models from HOCOMOCO v11.

[37]:

```
%%R

#load libraries and select TF motifs
library(motifbreakR)
library(BSgenome.Hsapiens.UCSC.hg38)
library(MotifDb)

motifs <- query(MotifDb, andStrings=c("hocomocov11", "hsapiens"))
length(motifs)
```

```
R[write to console]: Loading required package: grid

R[write to console]: Loading required package: MotifDb

R[write to console]: Loading required package: BiocGenerics

R[write to console]: Loading required package: parallel

R[write to console]:
Attaching package: ‘BiocGenerics’


R[write to console]: The following objects are masked from ‘package:parallel’:

    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
    clusterExport, clusterMap, parApply, parCapply, parLapply,
    parLapplyLB, parRapply, parSapply, parSapplyLB


R[write to console]: The following objects are masked from ‘package:stats’:

    IQR, mad, sd, var, xtabs


R[write to console]: The following objects are masked from ‘package:base’:

    Filter, Find, Map, Position, Reduce, anyDuplicated, append,
    as.data.frame, basename, cbind, colnames, dirname, do.call,
    duplicated, eval, evalq, get, grep, grepl, intersect, is.unsorted,
    lapply, mapply, match, mget, order, paste, pmax, pmax.int, pmin,
    pmin.int, rank, rbind, rownames, sapply, setdiff, sort, table,
    tapply, union, unique, unsplit, which.max, which.min


R[write to console]: Loading required package: S4Vectors

R[write to console]: Loading required package: stats4
```

```
R[write to console]:
Attaching package: ‘S4Vectors’


R[write to console]: The following object is masked from ‘package:base’:

    expand.grid


R[write to console]: Loading required package: IRanges

R[write to console]: Loading required package: GenomicRanges

R[write to console]: Loading required package: GenomeInfoDb

R[write to console]: Loading required package: Biostrings

R[write to console]: Loading required package: XVector

R[write to console]:
Attaching package: ‘Biostrings’


R[write to console]: The following object is masked from ‘package:base’:

    strsplit


R[write to console]: See system.file("LICENSE", package="MotifDb") for use
restrictions.

R[write to console]: Loading required package: BSgenome

R[write to console]: Loading required package: rtracklayer


[1] 768
```

motifbreakR implements three methods for calculation of motif match scores: “log”, “default” and “ic”. We will use the “log” method with uniform background and filter results with p-value threshold = 1e-4.

[38]:
```
%%R

score_snps <- function(snps_file, out_file) {
    #read SNPs from input bed file
    snps.mb.frombed <- snps.from.file(file = snps_file, search.genome =␣
 ↪BSgenome.Hsapiens.UCSC.hg38, format = "bed")
```

```
    #calculate scores
    results_log <- motifbreakR(snpList = snps.mb.frombed, filterp = TRUE,
                           pwmList = motifs,
                           threshold = 1e-5,
                           method = "log",
                           bkg = c(A=0.25, C=0.25, G=0.25, T=0.25),
                           BPPARAM = BiocParallel::bpparam())

    #reformat results to dataframe and save to file
    results_log_df <- data.frame(results_log)
    write.table(results_log_df, out_file, quote=F, sep="\t", row.names=F)
}
```

[39]:
```
for snp_bed in snps_bed_files:
    snp_scores_csv = snp_bed.replace(".bed", "_motifbreakR-scores.csv")
    print("Calculate scores for input: %s, save output to: %s" % (snp_bed,␣
 ↪snp_scores_csv))
    %Rpush snp_bed
    %Rpush snp_scores_csv
    %R snp_scores = score_snps(snp_bed, snp_scores_csv)
    print("Done")
```

```
Calculate scores for input: output/promoter_rare_enriched_SNPs.bed, save output
to: output/promoter_rare_enriched_SNPs_motifbreakR-scores.csv
Done
Calculate scores for input: output/enhancer_rare_enriched_SNPs.bed, save output
to: output/enhancer_rare_enriched_SNPs_motifbreakR-scores.csv
Done
```

Inspect results.

[40]:
```
promoter_SNPs_motifbreakr = pd.
 ↪read_csv("%spromoter_rare_enriched_SNPs_motifbreakR-scores.csv" % OUTPUT,␣
 ↪sep = "\t")
promoter_SNPs_motifbreakr.head()
```

[40]:
```
  seqnames      start        end  width strand              SNP_id REF ALT  \
0    chr19   45783029   45783029      1      +  chr19:45783029:T:G   T   G
1     chr2  222320241  222320241      1      -  chr2:222320241:T:C   T   C
2    chr17    6556629    6556629      1      -   chr17:6556629:G:C   G   C
3    chr14   92106635   92106635      1      -  chr14:92106635:G:C   G   C
4     chr2   26692643   26692643      1      +   chr2:26692643:C:G   C   G

  varType   motifPos  …                             seqMatch  \
0     SNV  c(-10, 7)  …      aggaggtggggaaGggggggtgaggacaggaccag
1     SNV  c(-11, 4)  …        gcgaccgcctcCccctcccgcctccccgtcc
2     SNV   c(-7, 3)  …              caccccCcgccccgccgggga
```

```
3     SNV  c(-4, 14)  …       ctctcctccgcccaCcccccccctccccggcccgccc
4     SNV  c(-3, 12)  …          gaggcggcctgCgggggggggcggggggcg

      pctRef    pctAlt   scoreRef   scoreAlt  Refpvalue  Altpvalue  altPos  \
0  0.857928  0.881182   8.465864   9.665273        NaN        NaN       1
1  0.923301  0.894667  10.236474   8.806286        NaN        NaN       1
2  0.820873  0.957512   2.275853   9.838534        NaN        NaN       1
3  0.907857  0.916432  11.693748  12.290103        NaN        NaN       1
4  0.900518  0.951450   9.098499  11.642407        NaN        NaN       1

   alleleDiff  effect
0    1.199409  strong
1   -1.430189  strong
2    7.562681  strong
3    0.596354    weak
4    2.543908  strong

[5 rows x 24 columns]
```

[41]:
```
enhancer_SNPs_motifbreakr = pd.
 ↪read_csv("%senhancer_rare_enriched_SNPs_motifbreakR-scores.csv" % OUTPUT,␣
 ↪sep = "\t")
enhancer_SNPs_motifbreakr.head()
```

[41]:
```
  seqnames      start        end  width strand               SNP_id REF ALT  \
0     chr2   90397298   90397298      1      +    chr2:90397298:T:G   T   G
1     chr2   90397732   90397732      1      -    chr2:90397732:G:A   G   A
2    chr12  116359966  116359966      1      -  chr12:116359966:T:C   T   C
3    chr13   26475911   26475911      1      +   chr13:26475911:A:C   A   C
4     chr4    4857069    4857069      1      -     chr4:4857069:C:A   C   A

  varType   motifPos  …                                 seqMatch  \
0     SNV  c(-10, 9)  …     atcatcttcgagtggAaccgaaaggaatcgccaaatgga
1     SNV  c(-10, 6)  …        tcgaatggaattGaatagaatcaacgaatggaa
2     SNV   c(-7, 8)  …         ccttccttcctTcttcctgtttcttttagac
3     SNV  c(-15, 4)  …     atggggcaagggggcAaggctgggaacaaggctgtggcc
4     SNV   c(-9, 2)  …               gccccccGcccccccccccccc

      pctRef    pctAlt   scoreRef   scoreAlt  Refpvalue  Altpvalue  altPos  \
0  0.858048  0.890445  10.935724  14.099791        NaN        NaN       1
1  0.903968  0.918037   9.979373  10.774472        NaN        NaN       1
2  0.869172  0.960194   6.350885  12.108658        NaN        NaN       1
3  0.847194  0.867085   8.574766   9.675056        NaN        NaN       1
4  0.958712  0.924593  10.942699   8.989977        NaN        NaN       1

   alleleDiff  effect
0    3.164068  strong
```

```
1    0.795099  strong
2    5.757773  strong
3    1.100290  strong
4   -1.952722  strong

[5 rows x 24 columns]
```

Extract SNPs which were predicted to have “strong” effect on TF binding. The strength is defined as absolute difference of proportional frequencies of REF and ALT alleles at position motifPos in the motif. If this difference is > 0.7 then the effect is classified as “strong”.

[42]:
```
#select records with "strong" effect
promoter_SNPs_motifbreakr_strong =␣
↪promoter_SNPs_motifbreakr[promoter_SNPs_motifbreakr["effect"] == "strong"]
enhancer_SNPs_motifbreakr_strong =␣
↪enhancer_SNPs_motifbreakr[enhancer_SNPs_motifbreakr["effect"] == "strong"]
```

Select only one record per SNP and keep information about motif with the highest pct score and biggest difference between alleles. These will be stored in motif_best_match and motif_highest_diff columns, in a format motif_id:score. I decide to keep both motifs because it is difficult to decide if good match is more or less important than high difference between alleles.

[43]:
```
#add columns with info about best matches: motif with highest pct score and␣
↪alleleDiff

def find_best_matching_motif(group):
    #find motif with highest pct score (either for REF or ALT)
    best_pctRef_score = max(group["pctRef"])
    best_pctAlt_score = max(group["pctAlt"])
    if best_pctRef_score > best_pctAlt_score:
        best_pct_score_motif = group[group["pctRef"] ==␣
↪best_pctRef_score]["providerId"].values[0]
    else:
        best_pct_score_motif = group[group["pctAlt"] ==␣
↪best_pctAlt_score]["providerId"].values[0]

    #find motif with highest abs(diff) between alleles
    best_alleleDiff = max(group["alleleDiff"].abs())
    best_alleleDiff_motif = group[group["alleleDiff"].abs() ==␣
↪best_alleleDiff]["providerId"].values[0]

    return best_pct_score_motif + ":" + "%.2f" % max(best_pctRef_score,␣
↪best_pctAlt_score), best_alleleDiff_motif + ":" + "%.2f" % best_alleleDiff

#information about best motifs for each SNP will be stored in a dict in which␣
↪SNP_ids will be keys
best_motifs_dict = {}
```

```python
for df in [enhancer_SNPs_motifbreakr_strong, promoter_SNPs_motifbreakr_strong]:
    for snp_id, snp_records in df.groupby("SNP_id"):
        best_match, highest_diff = find_best_matching_motif(snp_records)
        best_motifs_dict[snp_id] = {"best_match" : best_match, "highest_diff" :␣
 ↪highest_diff}

#extract information from the dict to fill appropriate columns in enhancer and␣
 ↪promoter SNPs dataframes
enhancer_SNPs_motifbreakr_strong["motif_best_match"] = ␣
 ↪enhancer_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:␣
 ↪best_motifs_dict[x]["best_match"])
enhancer_SNPs_motifbreakr_strong["motif_highest_diff"] = ␣
 ↪enhancer_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:␣
 ↪best_motifs_dict[x]["highest_diff"])

promoter_SNPs_motifbreakr_strong["motif_best_match"] = ␣
 ↪promoter_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:␣
 ↪best_motifs_dict[x]["best_match"])
promoter_SNPs_motifbreakr_strong["motif_highest_diff"] = ␣
 ↪promoter_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:␣
 ↪best_motifs_dict[x]["highest_diff"])
```

```
<ipython-input-43-c5359b181366>:27: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  enhancer_SNPs_motifbreakr_strong["motif_best_match"] =
enhancer_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:
best_motifs_dict[x]["best_match"])
<ipython-input-43-c5359b181366>:28: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  enhancer_SNPs_motifbreakr_strong["motif_highest_diff"] =
enhancer_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:
best_motifs_dict[x]["highest_diff"])
<ipython-input-43-c5359b181366>:30: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  promoter_SNPs_motifbreakr_strong["motif_best_match"] =
promoter_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:
best_motifs_dict[x]["best_match"])
<ipython-input-43-c5359b181366>:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  promoter_SNPs_motifbreakr_strong["motif_highest_diff"] =
promoter_SNPs_motifbreakr_strong.SNP_id.apply(lambda x:
best_motifs_dict[x]["highest_diff"])
```

Finally keep one record per SNP.

[44]:
```
#extract information about SNP location and best motifs, drop duplicates
promoter_SNPs_motifbreakr_strong_snps_only =␣
 ↪promoter_SNPs_motifbreakr_strong[["seqnames", "start", "REF", "ALT",␣
 ↪"motif_best_match", "motif_highest_diff"]].drop_duplicates()
enhancer_SNPs_motifbreakr_strong_snps_only =␣
 ↪enhancer_SNPs_motifbreakr_strong[["seqnames", "start", "REF", "ALT",␣
 ↪"motif_best_match", "motif_highest_diff"]].drop_duplicates()

#change column names to keep the convention used in the whole notebook
promoter_SNPs_motifbreakr_strong_snps_only =␣
 ↪promoter_SNPs_motifbreakr_strong_snps_only.rename(columns = {"seqnames":
 ↪"CHROM", "start":"POS"})
enhancer_SNPs_motifbreakr_strong_snps_only =␣
 ↪enhancer_SNPs_motifbreakr_strong_snps_only.rename(columns = {"seqnames":
 ↪"CHROM", "start":"POS"})
```

[45]:
```
print(len(promoter_SNPs_motifbreakr_strong_snps_only), "and",␣
 ↪len(enhancer_SNPs_motifbreakr_strong_snps_only),
     "SNPs in promoters and enhancers have predicted strong effect of motif␣
 ↪binding (out of %s and %s, respectively)." %␣
 ↪(str(len(rare_enriched_promoter_snps)),␣
 ↪str(len(rare_enriched_enhancer_snps))))
```

```
925 and 449 SNPs in promoters and enhancers have predicted strong effect of
motif binding (out of 1313 and 784, respectively).
```

Merge information about allele counts and frequencies with selected SNPs.

[46]:
```
rare_enriched_promoter_snps_motif = pd.merge(rare_enriched_promoter_snps,␣
 ↪promoter_SNPs_motifbreakr_strong_snps_only, how = "right", on = ["CHROM",␣
 ↪"POS", "REF", "ALT"])
```

```
rare_enriched_enhancer_snps_motif = pd.merge(rare_enriched_enhancer_snps,␣
↪enhancer_SNPs_motifbreakr_strong_snps_only, how = "right", on = ["CHROM",␣
↪"POS", "REF", "ALT"])
```

Check how SNP data look like now.

```
[47]: rare_enriched_promoter_snps_motif.head()
```

```
[47]:    CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
      0  chr19   45783029   T   G   2  0.043  46           0.000076
      1   chr2  222320241   T   C   4  0.087  46           0.000200
      2  chr17    6556629   G   C   2  0.091  22           0.001300
      3   chr2   26692643   C   G   2  0.048  42           0.000046
      4  chr12  106247724   G   T   2  0.063  32           0.000200

         gnomAD_genome_NFE  binom_pval  B-H_reject_H0  corrected_binom_pval  \
      0             0.0000    0.000000           True              0.000000
      1             0.0000    0.000000           True              0.000000
      2             0.0017    0.000653           True              0.004048
      3             0.0000    0.000000           True              0.000000
      4             0.0003    0.000044           True              0.000320

                   motif_best_match          motif_highest_diff
      0  KLF15_HUMAN.H11MO.0.A:0.93  ZN770_HUMAN.H11MO.0.C:3.80
      1   E2F6_HUMAN.H11MO.0.A:0.98   E2F6_HUMAN.H11MO.0.A:3.84
      2    MAZ_HUMAN.H11MO.1.A:0.96    MAZ_HUMAN.H11MO.1.A:7.56
      3  ZN740_HUMAN.H11MO.0.D:0.99  ZN740_HUMAN.H11MO.0.D:4.41
      4   OSR2_HUMAN.H11MO.0.C:0.95    WT1_HUMAN.H11MO.0.C:3.25
```

```
[48]: rare_enriched_enhancer_snps_motif.head()
```

```
[48]:    CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
      0   chr2   90397298   T   G   5  0.109  46           0.000088
      1   chr2   90397732   G   A  11  0.239  46           0.000400
      2  chr12  116359966   T   C   1  0.022  46           0.000000
      3  chr13   26475911   A   C   1  0.024  42           0.000065
      4   chr4    4857069   C   A   2  0.043  46           0.001100

         gnomAD_genome_NFE    binom_pval  B-H_reject_H0  corrected_binom_pval  \
      0             0.0000  0.000000e+00           True          0.000000e+00
      1             0.0005  6.410374e-27           True          8.314926e-26
      2             0.0000  0.000000e+00           True          0.000000e+00
      3             0.0000  0.000000e+00           True          0.000000e+00
      4             0.0000  0.000000e+00           True          0.000000e+00

                   motif_best_match          motif_highest_diff
      0  ZN394_HUMAN.H11MO.1.D:0.99  SMCA5_HUMAN.H11MO.0.C:4.32
```

```
1  ZN394_HUMAN.H11MO.1.D:1.00  ZN394_HUMAN.H11MO.1.D:4.09
2   ETS2_HUMAN.H11MO.0.B:0.96   ETV2_HUMAN.H11MO.0.B:5.76
3    SP4_HUMAN.H11MO.0.A:0.87    SP4_HUMAN.H11MO.0.A:1.10
4  SALL4_HUMAN.H11MO.0.B:1.00  ZN320_HUMAN.H11MO.0.C:4.26
```

8 Assign target genes

8.1 Promoter SNPs

[49]:
```
rare_enriched_promoter_snps_motif["genomic element"] = "promoter"
```

We will intersect the promoter SNPs data with promoters data to be able to assign genes to SNPs. This will be done with pybedtools package - a python interface to bedtools.

[50]:
```
#create BedTool object from promoter regions bed
promoters_info = pbt.BedTool(PROMOTER_REGIONS)

#create BedTool object from dataframe with selected promoter SNPs
rare_enriched_promoter_snps_motif["POS-1"] =␣
 ↪rare_enriched_promoter_snps_motif["POS"] - 1
rare_enriched_promoter_snps_motif_bedtool = pbt.BedTool.
 ↪from_dataframe(rare_enriched_promoter_snps_motif[["CHROM", "POS-1", "POS"]])
rare_enriched_promoter_snps_motif = rare_enriched_promoter_snps_motif.
 ↪drop(labels = ["POS-1"], axis=1)

#intersect promoters and SNPs
rare_enriched_promoter_snps_motif_intersection =␣
 ↪rare_enriched_promoter_snps_motif_bedtool.intersect(promoters_info, wa=True,␣
 ↪wb=True)

#create a dataframe from the intersection results, keep only columns with SNP␣
 ↪location and gene(s) name(s)
rare_enriched_promoter_snps_motif_intersection_df =␣
 ↪rare_enriched_promoter_snps_motif_intersection.to_dataframe(names =␣
 ↪["CHROM", "POS", "Gene"], usecols = [0, 2, 6]).drop_duplicates()
rare_enriched_promoter_snps_motif_intersection_df.head()
```

```
/usr/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering
(buffering=1) isn't supported in binary mode, the default buffer size will be
used
  self.stderr = io.open(errread, 'rb', bufsize)
```

[50]:
```
   CHROM        POS                                        Gene
0  chr19   45783029                         ENSG00000104936/DMPK
1   chr2  222320241                   ENSG00000237732/AC010980.2
2  chr17    6556629                      ENSG00000091622/PITPNM3
3   chr2   26692643                        ENSG00000171303/KCNK3
```

```
4  chr12  106247724  ENSG00000136026/CKAP4,ENSG00000258355/RP11-651…
```

[51]:
```
#merge the intersection dataframe with df containing frequency information
rare_enriched_promoter_snps_motif_gene = pd.
 ↪merge(rare_enriched_promoter_snps_motif,␣
 ↪rare_enriched_promoter_snps_motif_intersection_df, how = "left", on =␣
 ↪["CHROM", "POS"])
rare_enriched_promoter_snps_motif_gene.head()
```

[51]:
```
   CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
0  chr19   45783029   T   G   2  0.043  46           0.000076
1   chr2  222320241   T   C   4  0.087  46           0.000200
2  chr17    6556629   G   C   2  0.091  22           0.001300
3   chr2   26692643   C   G   2  0.048  42           0.000046
4  chr12  106247724   G   T   2  0.063  32           0.000200

   gnomAD_genome_NFE  binom_pval  B-H_reject_H0  corrected_binom_pval  \
0             0.0000    0.000000           True              0.000000
1             0.0000    0.000000           True              0.000000
2             0.0017    0.000653           True              0.004048
3             0.0000    0.000000           True              0.000000
4             0.0003    0.000044           True              0.000320

             motif_best_match          motif_highest_diff genomic element  \
0  KLF15_HUMAN.H11MO.0.A:0.93  ZN770_HUMAN.H11MO.0.C:3.80        promoter
1   E2F6_HUMAN.H11MO.0.A:0.98   E2F6_HUMAN.H11MO.0.A:3.84        promoter
2    MAZ_HUMAN.H11MO.1.A:0.96    MAZ_HUMAN.H11MO.1.A:7.56        promoter
3  ZN740_HUMAN.H11MO.0.D:0.99  ZN740_HUMAN.H11MO.0.D:4.41        promoter
4   OSR2_HUMAN.H11MO.0.C:0.95    WT1_HUMAN.H11MO.0.C:3.25        promoter

                                                Gene
0                           ENSG00000104936/DMPK
1                     ENSG00000237732/AC010980.2
2                        ENSG00000091622/PITPNM3
3                          ENSG00000171303/KCNK3
4  ENSG00000136026/CKAP4,ENSG00000258355/RP11-651…
```

8.2 Enhancer SNPs

8.2.1 Intronic enhancers

For enhancers located inside genes the gene in which they reside is treated as candidate. Provided bed file with active enhancers contains information about intersecting gene in the last column. If enhancer is intergenic then the last column contains "." .

[52]:
```
enh_genes = pd.read_csv(ENHANCER_REGIONS, sep='\t', names = ['chr', 'start',␣
 ↪'end', 'Gene'])
enh_genes[:10]
```

```
[52]:     chr    start      end                        Gene
      0  chr1    19482    22400      ENSG00000227232/WASH7P
      1  chr1   189803   193125  ENSG00000279457/FO538757.2
      2  chr1  1134548  1135587                           .
      3  chr1  1156466  1159847                           .
      4  chr1  1504484  1506451                           .
      5  chr1  1550199  1551447       ENSG00000160075/SSU72
      6  chr1  1567417  1568787       ENSG00000160075/SSU72
      7  chr1  1806062  1808150        ENSG00000078369/GNB1
      8  chr1  1846213  1848350        ENSG00000078369/GNB1
      9  chr1  1857785  1859590        ENSG00000078369/GNB1
```

```
[53]: #reformat to have one gene ID in cell
      enh_gene = pd.DataFrame()
      for i, row in enh_genes.iterrows():
          genes = row['Gene'].split(',')
          if len(genes) == 1:
              enh_gene = enh_gene.append(row, ignore_index=True)
          else:
              for gene in genes:
                  new_row = row
                  new_row['Gene'] = gene
                  enh_gene = enh_gene.append(new_row, ignore_index=True)
                  enh_gene = enh_gene.reindex(enh_genes.columns, axis=1)
                  enh_gene["start"] = enh_gene["start"].astype(int)
                  enh_gene["end"] = enh_gene["end"].astype(int)
      enh_gene[:10]
```

```
[53]:     chr    start      end                        Gene
      0  chr1    19482    22400      ENSG00000227232/WASH7P
      1  chr1   189803   193125  ENSG00000279457/FO538757.2
      2  chr1  1134548  1135587                           .
      3  chr1  1156466  1159847                           .
      4  chr1  1504484  1506451                           .
      5  chr1  1550199  1551447       ENSG00000160075/SSU72
      6  chr1  1567417  1568787       ENSG00000160075/SSU72
      7  chr1  1806062  1808150        ENSG00000078369/GNB1
      8  chr1  1846213  1848350        ENSG00000078369/GNB1
      9  chr1  1857785  1859590        ENSG00000078369/GNB1
```

Intersect information about enhancers with SNPs to assign gene names to SNPs.

```
[54]: #prepare bedtool objects
      rare_enriched_enhancer_snps_motif["POS-1"] =␣
       ↪rare_enriched_enhancer_snps_motif["POS"] - 1
```

```
rare_enriched_enhancer_snps_motif_bedtool = pbt.BedTool.
↪from_dataframe(rare_enriched_enhancer_snps_motif[["CHROM", "POS-1", "POS",␣
↪"REF", "ALT", "AC", "AF", "AN",
                                                                              ␣
↪                           "gnomAD_genome_ALL", "gnomAD_genome_NFE",␣
↪"binom_pval",
                                                                              ␣
↪                           "B-H_reject_H0", "corrected_binom_pval",␣
↪"motif_best_match",
                                                                              ␣
↪                           "motif_highest_diff"]])
enh_genes_bedtool = pbt.BedTool.from_dataframe(enh_gene)

#intersect
rare_enriched_enhancer_snps_motif_intersection =␣
↪rare_enriched_enhancer_snps_motif_bedtool.intersect(enh_genes_bedtool,␣
↪wa=True, wb=True, loj=True)

#reformat intersection to dataframe, keep columns with enhancer coordinates -␣
↪they will be usefull in the next step
rare_enriched_enhancer_snps_motif_gene =␣
↪rare_enriched_enhancer_snps_motif_intersection.to_dataframe(usecols =␣
↪[0,2,3,4,5,6,7,8,9,10,11,12,13,14,16,17,18], names = ["CHROM", "POS", "REF",␣
↪"ALT", "AC", "AF", "AN",
                                                                              ␣
↪                           "gnomAD_genome_ALL", "gnomAD_genome_NFE",␣
↪"binom_pval",
                                                                              ␣
↪                           "B-H_reject_H0", "corrected_binom_pval",␣
↪"motif_best_match",
                                                                              ␣
↪                           "motif_highest_diff", "enh_start",␣
↪"enh_end","Gene"])
rare_enriched_enhancer_snps_motif_gene.head()
```

```
/usr/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering
(buffering=1) isn't supported in binary mode, the default buffer size will be
used
  self.stderr = io.open(errread, 'rb', bufsize)
```

[54]:

```
   CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
0   chr2   90397298   T   G   5  0.109  46           0.000088
1   chr2   90397732   G   A  11  0.239  46           0.000400
2  chr12  116359966   T   C   1  0.022  46           0.000000
3  chr13   26475911   A   C   1  0.024  42           0.000065
4   chr4    4857069   C   A   2  0.043  46           0.001100
```

```
   gnomAD_genome_NFE    binom_pval  B-H_reject_H0  corrected_binom_pval  \
0             0.0000  0.000000e+00           True          0.000000e+00
1             0.0005  6.410374e-27           True          8.314926e-26
2             0.0000  0.000000e+00           True          0.000000e+00
3             0.0000  0.000000e+00           True          0.000000e+00
4             0.0000  0.000000e+00           True          0.000000e+00

             motif_best_match          motif_highest_diff  enh_start  \
0  ZN394_HUMAN.H11MO.1.D:0.99  SMCA5_HUMAN.H11MO.0.C:4.32   90397237
1  ZN394_HUMAN.H11MO.1.D:1.00  ZN394_HUMAN.H11MO.1.D:4.09   90397237
2   ETS2_HUMAN.H11MO.0.B:0.96   ETV2_HUMAN.H11MO.0.B:5.76  116358414
3    SP4_HUMAN.H11MO.0.A:0.87    SP4_HUMAN.H11MO.0.A:1.10   26475763
4  SALL4_HUMAN.H11MO.0.B:1.00  ZN320_HUMAN.H11MO.0.C:4.26    4855091

     enh_end Gene
0   90398831    .
1   90398831    .
2  116360334    .
3   26476965    .
4    4857297    .
```

Add column which will inform if enhancer is intronic or intergenic.

```
[55]: rare_enriched_enhancer_snps_motif_gene["genomic element"] =␣
       ↪rare_enriched_enhancer_snps_motif_gene.Gene.apply(lambda x: "enhancer␣
       ↪intergenic" if x == "." else "enhancer intronic")
      rare_enriched_enhancer_snps_motif_gene.head()
```

```
[55]:    CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
      0   chr2   90397298   T   G   5  0.109  46           0.000088
      1   chr2   90397732   G   A  11  0.239  46           0.000400
      2  chr12  116359966   T   C   1  0.022  46           0.000000
      3  chr13   26475911   A   C   1  0.024  42           0.000065
      4   chr4    4857069   C   A   2  0.043  46           0.001100

         gnomAD_genome_NFE    binom_pval  B-H_reject_H0  corrected_binom_pval  \
      0             0.0000  0.000000e+00           True          0.000000e+00
      1             0.0005  6.410374e-27           True          8.314926e-26
      2             0.0000  0.000000e+00           True          0.000000e+00
      3             0.0000  0.000000e+00           True          0.000000e+00
      4             0.0000  0.000000e+00           True          0.000000e+00

                   motif_best_match          motif_highest_diff  enh_start  \
      0  ZN394_HUMAN.H11MO.1.D:0.99  SMCA5_HUMAN.H11MO.0.C:4.32   90397237
      1  ZN394_HUMAN.H11MO.1.D:1.00  ZN394_HUMAN.H11MO.1.D:4.09   90397237
      2   ETS2_HUMAN.H11MO.0.B:0.96   ETV2_HUMAN.H11MO.0.B:5.76  116358414
```

```
3    SP4_HUMAN.H11MO.0.A:0.87    SP4_HUMAN.H11MO.0.A:1.10   26475763
4  SALL4_HUMAN.H11MO.0.B:1.00  ZN320_HUMAN.H11MO.0.C:4.26    4855091

     enh_end Gene      genomic element
0   90398831    .  enhancer intergenic
1   90398831    .  enhancer intergenic
2  116360334    .  enhancer intergenic
3   26476965    .  enhancer intergenic
4    4857297    .  enhancer intergenic
```

8.2.2 Intergenic enhancers

For enhancers located in intergenic regions targets can be assigned based on distance - by selecting the closest genes or based on chromatin contacts information inferred from Hi-C data.

Closest gene To find closest genes we will first obtain TSS locations from hg38 genome annotation from Ensembl (GRCh38.p5). The gtf file used here contains only "gene" records.

[56]:
```
full_annot = pd.read_csv('data/hg38_full.genes.gtf', sep='\t', usecols = [0, 2,␣
↪3, 4, 6, 8], skiprows = 5,
                          names = ["chr", "type", "start", "end", "strand",␣
↪'info'])
genes_info = full_annot[full_annot["type"] == "gene"]
genes_info['ID'] = genes_info['info'].str.split('"').str[1]
genes_info['Gene'] = genes_info['ID'] + "/" + genes_info['info'].str.split('"').
↪str[5]

genes_info[:10]
```

[56]:
```
    chr  type  start     end strand  \
0  chr1  gene  11869   14409      +
1  chr1  gene  14404   29570      -
2  chr1  gene  17369   17436      -
3  chr1  gene  29554   31109      +
4  chr1  gene  30366   30503      +
5  chr1  gene  34554   36081      -
6  chr1  gene  52473   53312      +
7  chr1  gene  62948   63887      +
8  chr1  gene  69091   70008      +
9  chr1  gene  89295  133723      -

                                                info               ID  \
0  gene_id "ENSG00000223972"; gene_version "5"; g…  ENSG00000223972
1  gene_id "ENSG00000227232"; gene_version "5"; g…  ENSG00000227232
2  gene_id "ENSG00000278267"; gene_version "1"; g…  ENSG00000278267
3  gene_id "ENSG00000243485"; gene_version "3"; g…  ENSG00000243485
4  gene_id "ENSG00000274890"; gene_version "1"; g…  ENSG00000274890
```

```
5  gene_id "ENSG00000237613"; gene_version "2"; g…  ENSG00000237613
6  gene_id "ENSG00000268020"; gene_version "3"; g…  ENSG00000268020
7  gene_id "ENSG00000240361"; gene_version "1"; g…  ENSG00000240361
8  gene_id "ENSG00000186092"; gene_version "4"; g…  ENSG00000186092
9  gene_id "ENSG00000238009"; gene_version "6"; g…  ENSG00000238009

                           Gene
0       ENSG00000223972/DDX11L1
1        ENSG00000227232/WASH7P
2     ENSG00000278267/MIR6859-1
3  ENSG00000243485/RP11-34P13.3
4     ENSG00000274890/MIR1302-2
5       ENSG00000237613/FAM138A
6        ENSG00000268020/OR4G4P
7       ENSG00000240361/OR4G11P
8         ENSG00000186092/OR4F5
9  ENSG00000238009/RP11-34P13.7
```

[57]:
```
def find_tss(row):
    if row['strand'] == '+':
        return row['start']
    else:
        return row['end']
```

[58]:
```
genes_info["tss"] = genes_info.apply(find_tss, axis=1)
genes_info.head()
```

[58]:
```
    chr  type  start    end strand  \
0  chr1  gene  11869  14409      +
1  chr1  gene  14404  29570      -
2  chr1  gene  17369  17436      -
3  chr1  gene  29554  31109      +
4  chr1  gene  30366  30503      +

                                                info               ID  \
0  gene_id "ENSG00000223972"; gene_version "5"; g…  ENSG00000223972
1  gene_id "ENSG00000227232"; gene_version "5"; g…  ENSG00000227232
2  gene_id "ENSG00000278267"; gene_version "1"; g…  ENSG00000278267
3  gene_id "ENSG00000243485"; gene_version "3"; g…  ENSG00000243485
4  gene_id "ENSG00000274890"; gene_version "1"; g…  ENSG00000274890

                           Gene    tss
0       ENSG00000223972/DDX11L1  11869
1        ENSG00000227232/WASH7P  29570
2     ENSG00000278267/MIR6859-1  17436
3  ENSG00000243485/RP11-34P13.3  29554
4     ENSG00000274890/MIR1302-2  30366
```

Now we will use bedtools closest tool to identify TSS closest to each enhancer containing analyzed SNPs. In cases when more than one TSS can be found at the shortest distance all results will be reported.

```
[59]: genes_info_tss_bed = pbt.BedTool.from_dataframe(genes_info[['chr', 'tss',␣
       ↪'tss', 'Gene']])
      genes_info_tss_bed_sorted = genes_info_tss_bed.sort()

      enhancers_bed = pbt.BedTool.
       ↪from_dataframe(rare_enriched_enhancer_snps_motif_gene[['CHROM', 'enh_start',␣
       ↪'enh_end']].drop_duplicates()).sort()

      tss_closest_to_enh = enhancers_bed.closest(genes_info_tss_bed_sorted, t='all',␣
       ↪d=True)

      tss_closest_to_enh_df = tss_closest_to_enh.to_dataframe(names = ['CHROM',␣
       ↪'enh_start', 'enh_end', "closest gene", "distance to closest gene"],
                                                                  usecols = [0,1,2,6,7])
      tss_closest_to_enh_df.head()
```

```
/usr/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering
(buffering=1) isn't supported in binary mode, the default buffer size will be
used
  self.stderr = io.open(errread, 'rb', bufsize)
/usr/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering
(buffering=1) isn't supported in binary mode, the default buffer size will be
used
  self.stderr = io.open(errread, 'rb', bufsize)
/usr/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering
(buffering=1) isn't supported in binary mode, the default buffer size will be
used
  self.stderr = io.open(errread, 'rb', bufsize)
```

```
[59]:   CHROM  enh_start   enh_end                 closest gene  \
      0  chr1    6868223   6870996  ENSG00000227950/RP11-312B8.2
      1  chr1    8888663   8891414      ENSG00000238249/HMGN2P17
      2  chr1   10143598  10144774     ENSG00000201746/RNU6-828P
      3  chr1   21253526  21255567  ENSG00000236936/RP3-329E20.2
      4  chr1   24288566  24291017    ENSG00000266511/AL590683.2

         distance to closest gene
      0                     33605
      1                      1995
      2                     18494
      3                     10515
      4                      8711
```

Merge information about closest TSS with previously collected enhancer SNPs annotations.

```
[60]: rare_enriched_enhancer_snps_motif_gene_closest = pd.
       ↪merge(rare_enriched_enhancer_snps_motif_gene, tss_closest_to_enh_df,␣
       ↪how="left", on = ["CHROM", "enh_start", "enh_end"])
      rare_enriched_enhancer_snps_motif_gene_closest.head()
```

```
[60]:   CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
     0   chr2   90397298   T   G   5  0.109  46           0.000088
     1   chr2   90397732   G   A  11  0.239  46           0.000400
     2  chr12  116359966   T   C   1  0.022  46           0.000000
     3  chr13   26475911   A   C   1  0.024  42           0.000065
     4   chr4    4857069   C   A   2  0.043  46           0.001100

        gnomAD_genome_NFE    binom_pval  B-H_reject_H0  corrected_binom_pval  \
     0             0.0000  0.000000e+00           True          0.000000e+00
     1             0.0005  6.410374e-27           True          8.314926e-26
     2             0.0000  0.000000e+00           True          0.000000e+00
     3             0.0000  0.000000e+00           True          0.000000e+00
     4             0.0000  0.000000e+00           True          0.000000e+00

                  motif_best_match          motif_highest_diff  enh_start  \
     0  ZN394_HUMAN.H11MO.1.D:0.99  SMCA5_HUMAN.H11MO.0.C:4.32   90397237
     1  ZN394_HUMAN.H11MO.1.D:1.00  ZN394_HUMAN.H11MO.1.D:4.09   90397237
     2   ETS2_HUMAN.H11MO.0.B:0.96   ETV2_HUMAN.H11MO.0.B:5.76  116358414
     3    SP4_HUMAN.H11MO.0.A:0.87    SP4_HUMAN.H11MO.0.A:1.10   26475763
     4  SALL4_HUMAN.H11MO.0.B:1.00  ZN320_HUMAN.H11MO.0.C:4.26    4855091

          enh_end Gene      genomic element                  closest gene  \
     0   90398831    .  enhancer intergenic  ENSG00000281904/CH17-132F21.5
     1   90398831    .  enhancer intergenic  ENSG00000281904/CH17-132F21.5
     2  116360334    .  enhancer intergenic   ENSG00000258346/RP11-148B3.2
     3   26476965    .  enhancer intergenic          ENSG00000132970/WASF3
     4    4857297    .  enhancer intergenic           ENSG00000163132/MSX1

        distance to closest gene
     0                     31500
     1                     31500
     2                      8430
     3                     80738
     4                      2369
```

```
[61]: print(len(rare_enriched_enhancer_snps_motif_gene_closest),␣
       ↪len(rare_enriched_enhancer_snps_motif_gene))
```

```
477 477
```

Chromatin contacts predicted from Hi-C data To predict target genes we will also use chromatin contacts predicted based on Hi-C data from developing human brain (Won et al., 2016) using HiCEnterprise software.

```
[62]: # Read bed file with predicted contacts, select one record for each␣
       ↪enganger-gene pair
      contacts = pd.read_csv(CHROMATIN_CONTACTS, sep=' ')
      contacts_to_genes = contacts[contacts["ENSG"] != '-']
      contacts_to_genes = contacts_to_genes.drop_duplicates(subset = ["chr", "start",␣
       ↪"end", "ENSG"])
      contacts_to_genes = contacts_to_genes.rename(columns = {"chr":"CHROM",
                                                              "start":"enh_start",
                                                              "end":"enh_end",
                                                              "ENSG":"contacting␣
       ↪gene"})
      contacts_to_genes.head()
```

```
[62]:    CHROM  enh_start  enh_end             ENST  contacting gene  -log10(qval)  \
      1   chr1    1024210  1025994  ENST00000606034  ENSG00000272512      8.697366
      2   chr1    1024210  1025994  ENST00000484667  ENSG00000188290      8.697366
      6   chr1    1024210  1025994  ENST00000624697  ENSG00000187608      8.697366
      9   chr1    1024210  1025994  ENST00000330388  ENSG00000184163      2.648810
      12  chr1    1024210  1025994  ENST00000478065  ENSG00000131584      3.934685

          confirmed_both_ways
      1                     1
      2                     1
      6                     1
      9                     1
      12                    1
```

Merge enhancer variants with contacts.

```
[63]: rare_enriched_enhancer_snps_motif_gene_closest_contacts = pd.
       ↪merge(rare_enriched_enhancer_snps_motif_gene_closest,
                                                                               ␣
       ↪contacts_to_genes[["CHROM", "enh_start", "enh_end", "contacting gene"]],
                                                                                on =␣
       ↪["CHROM", "enh_start", "enh_end"], how = "left").fillna('.')
      rare_enriched_enhancer_snps_motif_gene_closest_contacts.head()
```

```
[63]:    CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
      0   chr2   90397298   T   G   5  0.109  46           0.000088
      1   chr2   90397732   G   A  11  0.239  46           0.000400
      2  chr12  116359966   T   C   1  0.022  46           0.000000
      3  chr13   26475911   A   C   1  0.024  42           0.000065
      4   chr4    4857069   C   A   2  0.043  46           0.001100

         gnomAD_genome_NFE    binom_pval  …  corrected_binom_pval  \
0              0.0000  0.000000e+00  ...          0.000000e+00
1              0.0005  6.410374e-27  ...          8.314926e-26
2              0.0000  0.000000e+00  ...          0.000000e+00
3              0.0000  0.000000e+00  ...          0.000000e+00
4              0.0000  0.000000e+00  ...          0.000000e+00

            motif_best_match          motif_highest_diff  enh_start  \
0  ZN394_HUMAN.H11MO.1.D:0.99  SMCA5_HUMAN.H11MO.0.C:4.32   90397237
1  ZN394_HUMAN.H11MO.1.D:1.00  ZN394_HUMAN.H11MO.1.D:4.09   90397237
2   ETS2_HUMAN.H11MO.0.B:0.96   ETV2_HUMAN.H11MO.0.B:5.76  116358414
3    SP4_HUMAN.H11MO.0.A:0.87    SP4_HUMAN.H11MO.0.A:1.10   26475763
4  SALL4_HUMAN.H11MO.0.B:1.00  ZN320_HUMAN.H11MO.0.C:4.26    4855091

     enh_end  Gene      genomic element                    closest gene  \
0   90398831     .  enhancer intergenic  ENSG00000281904/CH17-132F21.5
1   90398831     .  enhancer intergenic  ENSG00000281904/CH17-132F21.5
2  116360334     .  enhancer intergenic   ENSG00000258346/RP11-148B3.2
3   26476965     .  enhancer intergenic         ENSG00000132970/WASF3
4    4857297     .  enhancer intergenic          ENSG00000163132/MSX1

  distance to closest gene  contacting gene
0                    31500                .
1                    31500                .
2                     8430                .
3                    80738  ENSG00000132964
4                     2369                .

[5 rows x 21 columns]
```

Replace gene ID from the “contacting gene” column with ID/name.

[64]:
```python
def add_gene_name(gene_id):
    if len(genes_info[genes_info["ID"]==gene_id]) != 0:
        return genes_info[genes_info["ID"]==gene_id]["Gene"].values[0]
    else:
        return '.'
```

[65]:
```python
rare_enriched_enhancer_snps_motif_gene_closest_contacts["contacting gene"] =␣
 ↪rare_enriched_enhancer_snps_motif_gene_closest_contacts["contacting gene"].
 ↪apply(add_gene_name)
rare_enriched_enhancer_snps_motif_gene_closest_contacts.head()
```

[65]:
```
   CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
0   chr2   90397298   T   G   5  0.109  46           0.000088
1   chr2   90397732   G   A  11  0.239  46           0.000400
2  chr12  116359966   T   C   1  0.022  46           0.000000
3  chr13   26475911   A   C   1  0.024  42           0.000065
```

```
4   chr4    4857069   C   A   2  0.043  46           0.001100

   gnomAD_genome_NFE    binom_pval  …  corrected_binom_pval  \
0             0.0000  0.000000e+00  …          0.000000e+00
1             0.0005  6.410374e-27  …          8.314926e-26
2             0.0000  0.000000e+00  …          0.000000e+00
3             0.0000  0.000000e+00  …          0.000000e+00
4             0.0000  0.000000e+00  …          0.000000e+00

             motif_best_match          motif_highest_diff   enh_start  \
0  ZN394_HUMAN.H11MO.1.D:0.99  SMCA5_HUMAN.H11MO.0.C:4.32   90397237
1  ZN394_HUMAN.H11MO.1.D:1.00  ZN394_HUMAN.H11MO.1.D:4.09   90397237
2   ETS2_HUMAN.H11MO.0.B:0.96   ETV2_HUMAN.H11MO.0.B:5.76  116358414
3    SP4_HUMAN.H11MO.0.A:0.87    SP4_HUMAN.H11MO.0.A:1.10   26475763
4  SALL4_HUMAN.H11MO.0.B:1.00  ZN320_HUMAN.H11MO.0.C:4.26    4855091

     enh_end  Gene      genomic element                   closest gene  \
0   90398831     .  enhancer intergenic  ENSG00000281904/CH17-132F21.5
1   90398831     .  enhancer intergenic  ENSG00000281904/CH17-132F21.5
2  116360334     .  enhancer intergenic   ENSG00000258346/RP11-148B3.2
3   26476965     .  enhancer intergenic          ENSG00000132970/WASF3
4    4857297     .  enhancer intergenic           ENSG00000163132/MSX1

  distance to closest gene       contacting gene
0                    31500                     .
1                    31500                     .
2                     8430                     .
3                    80738  ENSG00000132964/CDK8
4                     2369                     .

[5 rows x 21 columns]
```

8.2.3 Reformat to have all target genes in one cell

Now we have separate columns for different sources of target gene predictions (containing, closest, contacts) and multiple rows for each variant can be present if more than one closest or contacting gene was found for a particular enhancer. We will collect all predicted targets in the Gene column.

[66]:
```
rare_enriched_enhancer_snps_motif_genes_collected = pd.DataFrame()

for name, group in rare_enriched_enhancer_snps_motif_gene_closest_contacts.
↪groupby(["CHROM", "POS", "REF", "ALT"]):
    containing_genes = [gene + "(containing)" for gene in group["Gene"].
↪unique() if gene != "."]
    closest_genes = [gene + "(closest)" for gene in group["closest gene"].
↪unique() if gene != "."]
```

```
    contacting_genes = [gene + "(contacting)" for gene in group["contacting␣
↪gene"].unique() if gene != "."]

    all_genes = []
    all_genes.extend(containing_genes)
    all_genes.extend(closest_genes)
    all_genes.extend(contacting_genes)

    group["Gene"] = ";".join(all_genes)

    rare_enriched_enhancer_snps_motif_genes_collected =␣
↪rare_enriched_enhancer_snps_motif_genes_collected.append(group[['CHROM',␣
↪'POS', 'REF', 'ALT',
                                                                               ␣
↪                                         'AC', 'AF', 'AN',␣
↪'gnomAD_genome_ALL',
                                                                               ␣
↪                                         'gnomAD_genome_NFE', 'binom_pval',␣
↪'B-H_reject_H0',
                                                                               ␣
↪                                         'corrected_binom_pval',␣
↪'motif_best_match', 'motif_highest_diff',
                                                                               ␣
↪                                         'enh_start', 'enh_end', 'Gene',␣
↪'genomic element']].drop_duplicates())
```

[67]:
```
rare_enriched_enhancer_snps_motif_genes_collected.head()
```

[67]:
```
    CHROM       POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
189  chr1   6868940   G   C   1  0.022  46           0.000200
82   chr1   8890584   T   C   1  0.022  46           0.000100
298  chr1  10144750   G   A   2  0.043  46           0.000100
326  chr1  10144754   G   A   2  0.043  46           0.000035
378  chr1  21254094   A   G   2  0.048  42           0.000037

     gnomAD_genome_NFE  binom_pval  B-H_reject_H0  corrected_binom_pval  \
189           0.000000    0.000000           True              0.000000
82            0.000000    0.000000           True              0.000000
298           0.000200    0.000041           True              0.000384
326           0.000071    0.000005           True              0.000055
378           0.000073    0.000005           True              0.000050

               motif_best_match          motif_highest_diff  enh_start  \
189   FLI1_HUMAN.H11MO.1.A:0.96   ETS1_HUMAN.H11MO.0.A:1.60    6868223
82   PRDM6_HUMAN.H11MO.0.C:0.96   IRF2_HUMAN.H11MO.0.A:3.86    8888663
298  CPEB1_HUMAN.H11MO.0.D:0.90  ZFP28_HUMAN.H11MO.0.C:1.89   10143598
326  PRDM6_HUMAN.H11MO.0.C:1.00  CPEB1_HUMAN.H11MO.0.D:3.01   10143598
```

```
378   IRF2_HUMAN.H11MO.0.A:0.89   IRF5_HUMAN.H11MO.0.D:2.20   21253526

      enh_end                                                 Gene  \
189   6870996  ENSG00000171735/CAMTA1(containing);ENSG0000022…
82    8891414  ENSG00000238249/HMGN2P17(closest);ENSG00000116…
298  10144774  ENSG00000130939/UBE4B(containing);ENSG00000201…
326  10144774  ENSG00000130939/UBE4B(containing);ENSG00000201…
378  21255567  ENSG00000117298/ECE1(containing);ENSG000002369…

         genomic element
189    enhancer intronic
82   enhancer intergenic
298    enhancer intronic
326    enhancer intronic
378    enhancer intronic
```

8.2.4 Check correlation between H3K27ac in enhancer and gene expression

We expect that enhancer activity should positively correlate with gene expression. Therefore we will use information about coverage from ChIP-seq on H3K27ac as enhancer activity measure and check if it correlates with expression of putative target genes. Putative targets with positive correlation and p-value < 0.15 will be reported. Spearman correlation will be calculated.

The ChIP-seq and RNA-seq data used here are from Stepniak et al [ref]

[68]:
```
# H3K27ac coverage
h3k27ac_cov = pd.read_csv(ENHANCER_ACTIVITY, sep = "\t")
h3k27ac_cov = h3k27ac_cov.rename(columns = {"chr":"CHROM",
                                             "start":"enh_start",
                                             "end":"enh_end"})

h3k27ac_cov.head()
```

[68]:
```
  CHROM  enh_start  enh_end      GB08      GB02      GB01      PA04      PA01  \
0  chr1      19482    22400  0.510363  0.834031  0.698638  0.988984  0.756236
1  chr1     183942   184568  0.671342  0.697314  0.402064  0.463842  0.788018
2  chr1     189803   193125  0.877638  0.989761  0.888490  1.141639  1.091900
3  chr1    1024210  1025994  0.793004  0.777391  0.688113  0.785313  0.587288
4  chr1    1031391  1031884  0.905488  0.851251  0.894948  1.180019  0.951712

       PA02      GB04      GB05      DA03      GB06      DA04      GB07  \
0  1.132902  0.657006  0.566208  0.528208  0.451753  0.574673  0.526820
1  1.099185  0.665276  0.743958  0.581258  0.533743  0.658219  0.651496
2  1.293182  0.825588  0.921340  0.935483  0.614575  0.967135  0.663150
3  0.661786  0.584530  0.526820  0.855769  0.487969  0.661978  0.602755
4  1.047810  0.822269  0.311449  0.755113  0.372133  0.701185  0.663920

       DA05      GB03      DA01      DA06
0  0.966401  0.426569  0.689541  0.638432
```

```
1  0.687957  0.541331  0.785030  0.634088
2  1.189149  0.475793  1.106543  0.791204
3  0.899062  0.581647  0.603402  0.811515
4  0.967906  0.420525  0.440595  0.634359
```

[69]:
```
# Genes/transcripts normalized counts
counts = pd.read_csv(GENE_EXPRESSION, sep='\t')
counts['Gene'] = counts.Transcript.apply(lambda x: '_'.join(x.split('_')[1:]))
counts.head()
```

[69]:
```
                Transcript    DA02    GB08   GB02    GB01   PA10   PA11  \
0     ENST00000000233_ARF5   800.0  1820.0  551.0  1412.0  922.0  869.0
1     ENST00000000412_M6PR  1302.0   485.0  790.0  1131.0  952.0  840.0
2    ENST00000000442_ESRRA   347.0   393.0  214.0   261.0  208.0  183.0
3    ENST00000001008_FKBP4   556.0   682.0  704.0  2868.0  300.0  462.0
4  ENST00000001146_CYP26B1    24.0    22.0   53.0    46.0  110.0  166.0

    PA08    PA03    PA09  …    GB06    DA04    GB10    GB07   DA05    GB03  \
0  713.0   912.0   798.0  …  1010.0  1195.0   802.0   901.0  664.0  1051.0
1  831.0  1044.0  1316.0  …  1209.0   772.0  1287.0  1131.0  699.0  1250.0
2  159.0   241.0   220.0  …   422.0   298.0   308.0   301.0  232.0   286.0
3  539.0   739.0   414.0  …   542.0   658.0   590.0   595.0  710.0   935.0
4  126.0   188.0    74.0  …    57.0    42.0    24.0    47.0   28.0    11.0

    DA01   DA07    DA06     Gene
0  986.0  843.0  1127.0     ARF5
1  707.0  849.0   912.0     M6PR
2  323.0  223.0   295.0    ESRRA
3  652.0  744.0  1076.0    FKBP4
4   13.0   31.0     9.0  CYP26B1

[5 rows x 35 columns]
```

[70]:
```
#merge SNPs with H3K27ac coverage on enhancers
rare_enriched_enhancer_snps_motif_genes_collected_coverage = pd.
 ↪merge(rare_enriched_enhancer_snps_motif_genes_collected,                          ␣
 ↪h3k27ac_cov, on = ["CHROM", "enh_start", "enh_end"], how = "left")

rare_enriched_enhancer_snps_motif_genes_collected_coverage.head()
```

[70]:
```
  CHROM       POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
0  chr1   6868940   G   C   1  0.022  46           0.000200
1  chr1   8890584   T   C   1  0.022  46           0.000100
2  chr1  10144750   G   A   2  0.043  46           0.000100
3  chr1  10144754   G   A   2  0.043  46           0.000035
4  chr1  21254094   A   G   2  0.048  42           0.000037
```

```
   gnomAD_genome_NFE  binom_pval  …      GB04      GB05      DA03      GB06  \
0           0.000000    0.000000  …  0.766341  0.885556  0.899062  1.049015
1           0.000000    0.000000  …  0.718616  0.772834  0.924597  1.050524
2           0.000200    0.000041  …  0.787857  0.409702  0.679355  0.392537
3           0.000071    0.000005  …  0.787857  0.409702  0.679355  0.392537
4           0.000073    0.000005  …  0.753438  0.799153  0.877427  0.955926

       DA04      GB07      DA05      GB03      DA01      DA06
0  0.979862  0.946164  0.878843  0.705809  0.757225  0.907536
1  0.808201  0.989958  0.997923  0.631530  1.012666  1.006235
2  0.693585  0.384560  0.552377  0.351660  0.295737  0.423495
3  0.693585  0.384560  0.552377  0.351660  0.295737  0.423495
4  1.046168  1.056799  0.937268  0.765077  0.804443  0.834471

[5 rows x 34 columns]
```

```
[71]: def calculate_correlation(row, sample_names):
          correlations = ""

          enh_act_vector = row[sample_names].values

          genes = set([el.split("(")[0] for el in row["Gene"].split(';')])

          #iterate over all target genes assigned to this variant
          for gene in genes:
              gene_name = gene.split('/')[1]
              gene_expr_rows = counts[counts["Gene"] == gene_name]
              if len(gene_expr_rows) != 0:

                  #calculate correlations for each transcript of the analyzed gene
                  gene_correlations = {}
                  for j, expr_row in gene_expr_rows.iterrows():
                      expr_vector = expr_row[sample_names].values
                      rho, pval = spearmanr(enh_act_vector, expr_vector)
                      #collect pvalues for positive correlations
                      if str(rho) != 'nan' and rho > 0:
                          gene_correlations[pval] = [rho, expr_row["Transcript"]]

                  #find best correlating transcript
                  if len(gene_correlations.keys()) > 0:
                      min_pval = min(gene_correlations.keys())
                      if min_pval < 0.15:
                          correlations += gene_name + "/" +␣
↪gene_correlations[min_pval][1].split("_")[0] + "/" + "%.5f" % min_pval + ";"

              else:
```

```
            pass
    if len(correlations) == 0:
        return "."
    else:
        return correlations.rstrip(";")
```

[72]:
```
samples = h3k27ac_cov.columns[3:]
rare_enriched_enhancer_snps_motif_genes_collected_coverage["H3K27ac-expression␣
↪correlation p-values"] =␣
↪rare_enriched_enhancer_snps_motif_genes_collected_coverage.
↪apply(calculate_correlation, args = (samples,), axis=1)
rare_enriched_enhancer_snps_motif_genes_collected_corelations =␣
↪rare_enriched_enhancer_snps_motif_genes_collected_coverage.drop(labels =␣
↪samples, axis=1)
rare_enriched_enhancer_snps_motif_genes_collected_corelations.head()
```

```
/home/researcher/.local/lib/python3.8/site-packages/scipy/stats/stats.py:4264:
SpearmanRConstantInputWarning: An input array is constant; the correlation
coefficent is not defined.
  warnings.warn(SpearmanRConstantInputWarning())
```

[72]:
```
  CHROM       POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
0  chr1   6868940   G   C   1  0.022  46           0.000200
1  chr1   8890584   T   C   1  0.022  46           0.000100
2  chr1  10144750   G   A   2  0.043  46           0.000100
3  chr1  10144754   G   A   2  0.043  46           0.000035
4  chr1  21254094   A   G   2  0.048  42           0.000037

   gnomAD_genome_NFE  binom_pval  B-H_reject_H0  corrected_binom_pval  \
0           0.000000    0.000000           True              0.000000
1           0.000000    0.000000           True              0.000000
2           0.000200    0.000041           True              0.000384
3           0.000071    0.000005           True              0.000055
4           0.000073    0.000005           True              0.000050

             motif_best_match          motif_highest_diff  enh_start  \
0   FLI1_HUMAN.H11MO.1.A:0.96   ETS1_HUMAN.H11MO.0.A:1.60    6868223
1  PRDM6_HUMAN.H11MO.0.C:0.96   IRF2_HUMAN.H11MO.0.A:3.86    8888663
2  CPEB1_HUMAN.H11MO.0.D:0.90  ZFP28_HUMAN.H11MO.0.C:1.89   10143598
3  PRDM6_HUMAN.H11MO.0.C:1.00  CPEB1_HUMAN.H11MO.0.D:3.01   10143598
4   IRF2_HUMAN.H11MO.0.A:0.89   IRF5_HUMAN.H11MO.0.D:2.20   21253526

    enh_end                                               Gene  \
0   6870996  ENSG00000171735/CAMTA1(containing);ENSG0000022…
1   8891414  ENSG00000238249/HMGN2P17(closest);ENSG00000116…
2  10144774  ENSG00000130939/UBE4B(containing);ENSG00000201…
3  10144774  ENSG00000130939/UBE4B(containing);ENSG00000201…
```

```
4  21255567  ENSG00000117298/ECE1(containing);ENSG000002369…

      genomic element H3K27ac-expression correlation p-values
0    enhancer intronic                                      .
1  enhancer intergenic                                      .
2    enhancer intronic                                      .
3    enhancer intronic                                      .
4    enhancer intronic           ECE1/ENST00000415912/0.06776
```

[73]:
```
def find_best_candidate_target(putative_targets):
    if putative_targets != ".":
        putative_targets_list = putative_targets.split(';')
        pvalues = [float(target.split('/')[2]) for target in␣
 ↪putative_targets_list]
        min_pval = min(pvalues)
        best_candidate = putative_targets_list[pvalues.index(min_pval)]
        return best_candidate
    else:
        return '.'
```

[74]:
```
rare_enriched_enhancer_snps_motif_genes_collected_corelations["Putative target␣
 ↪with highest correlation"] =␣
 ↪rare_enriched_enhancer_snps_motif_genes_collected_corelations["H3K27ac-expression␣
 ↪correlation p-values"].apply(find_best_candidate_target)
rare_enriched_enhancer_snps_motif_genes_collected_corelations.head()
```

[74]:
```
  CHROM       POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
0  chr1   6868940   G   C   1  0.022  46           0.000200
1  chr1   8890584   T   C   1  0.022  46           0.000100
2  chr1  10144750   G   A   2  0.043  46           0.000100
3  chr1  10144754   G   A   2  0.043  46           0.000035
4  chr1  21254094   A   G   2  0.048  42           0.000037

  gnomAD_genome_NFE  binom_pval  B-H_reject_H0  corrected_binom_pval  \
0          0.000000    0.000000           True              0.000000
1          0.000000    0.000000           True              0.000000
2          0.000200    0.000041           True              0.000384
3          0.000071    0.000005           True              0.000055
4          0.000073    0.000005           True              0.000050

            motif_best_match          motif_highest_diff  enh_start  \
0  FLI1_HUMAN.H11MO.1.A:0.96   ETS1_HUMAN.H11MO.0.A:1.60    6868223
1  PRDM6_HUMAN.H11MO.0.C:0.96   IRF2_HUMAN.H11MO.0.A:3.86    8888663
2  CPEB1_HUMAN.H11MO.0.D:0.90  ZFP28_HUMAN.H11MO.0.C:1.89   10143598
3  PRDM6_HUMAN.H11MO.0.C:1.00  CPEB1_HUMAN.H11MO.0.D:3.01   10143598
4   IRF2_HUMAN.H11MO.0.A:0.89   IRF5_HUMAN.H11MO.0.D:2.20   21253526
```

```
    enh_end                                                              Gene  \
0   6870996  ENSG00000171735/CAMTA1(containing);ENSG0000022…
1   8891414  ENSG00000238249/HMGN2P17(closest);ENSG00000116…
2  10144774  ENSG00000130939/UBE4B(containing);ENSG00000201…
3  10144774  ENSG00000130939/UBE4B(containing);ENSG00000201…
4  21255567  ENSG00000117298/ECE1(containing);ENSG000002369…

     genomic element H3K27ac-expression correlation p-values  \
0    enhancer intronic                                     .
1  enhancer intergenic                                     .
2    enhancer intronic                                     .
3    enhancer intronic                                     .
4    enhancer intronic             ECE1/ENST00000415912/0.06776

  Putative target with highest correlation
0                                        .
1                                        .
2                                        .
3                                        .
4             ECE1/ENST00000415912/0.06776
```

9 Check expression of TFs and target genes in brain

[75]:
```
gtex = pd.read_csv(GTEX, sep='\t', skiprows=[0,1])
brain_columns = [col for col in list(gtex.columns) if "Brain" in col]
```

[76]:
```
def get_gene_names(genes_string):
    #promoters will have ENSG00000136026/CKAP4, comma separated
    #enhancers will have ";"-separated lists with the following format:␣
↪ENSG00000171735/CAMTA1(containing)
    if genes_string:
        if "(" not in genes_string:
            return [el.split('/')[1] for el in genes_string.split(',')]
        else:
            return [el.split('/')[1].split('(')[0] for el in genes_string.
↪split(';')]
    else:
        return ""

def check_expression_in_brain(genes):

    gene_names_list = get_gene_names(genes)
    expression_list = []
    for gene in gene_names_list:
        if gene != "" and gene != ".":
            try:
                mean_median_tpm = sum(gtex[gtex['Description'] == gene.
↪strip()][brain_columns].values[0]) / float(len(brain_columns))
                expression_list.append(gene + ':' + "%.2f" % mean_median_tpm)
            except:
                print("no gtex brain data for:", gene)

    return ",".join(expression_list)
```

[77]:
```
rare_enriched_promoter_snps_motif_gene["Median TPM in brain tissues in GTEx"] =␣
 ↪rare_enriched_promoter_snps_motif_gene.Gene.apply(check_expression_in_brain)
rare_enriched_enhancer_snps_motif_genes_collected_corelations["Median TPM in␣
 ↪brain tissues in GTEx"] =␣
 ↪rare_enriched_enhancer_snps_motif_genes_collected_corelations.Gene.
 ↪apply(check_expression_in_brain)
```

```
no gtex brain data for: AL121992.1
no gtex brain data for: RP13-582O9.6
no gtex brain data for: AC138430.4
no gtex brain data for: AC138028.1
no gtex brain data for: RP11-460N20.3
no gtex brain data for: FAM150B
no gtex brain data for: AL590233.1
no gtex brain data for: FAM150B
no gtex brain data for: RP11-327F22.2
no gtex brain data for: AL022326.1
no gtex brain data for: CTD-3105H18.14
no gtex brain data for: RP4-665J23.2
no gtex brain data for: TMEM56-RWDD3
no gtex brain data for: BORCS8-MEF2B
no gtex brain data for: CTC-435M10.3
no gtex brain data for: TCEB2
no gtex brain data for: AC226119.5
no gtex brain data for: FAM150B
no gtex brain data for: RP4-665J23.2
no gtex brain data for: C1orf95
no gtex brain data for: AC138028.1
no gtex brain data for: AC226119.5
no gtex brain data for: TGIF2-C20orf24
no gtex brain data for: AC138430.4
no gtex brain data for: RP13-395E19.3
no gtex brain data for: RP13-395E19.3
no gtex brain data for: RP11-697E2.6
no gtex brain data for: FLJ35934
no gtex brain data for: XRCC6BP1
no gtex brain data for: MIR3650
no gtex brain data for: MIR1302-11
no gtex brain data for: AL132780.1
no gtex brain data for: CH17-232I21.1
no gtex brain data for: MLLT4
no gtex brain data for: MLLT4-AS1
no gtex brain data for: MIR1184-3
no gtex brain data for: GUSBP11
no gtex brain data for: KB-1572G7.2
no gtex brain data for: RN7SL671P
no gtex brain data for: DHFRL1
no gtex brain data for: RP13-539J13.1
no gtex brain data for: EBLN3
no gtex brain data for: RP11-396C23.2
no gtex brain data for: MIR1292
no gtex brain data for: NAMA_2
no gtex brain data for: AL356020.1
no gtex brain data for: STX16-NPEPL1
no gtex brain data for: RP13-395E19.3
no gtex brain data for: MIR1302-2
no gtex brain data for: RP11-34P13.3
no gtex brain data for: AC007128.1
no gtex brain data for: AC007040.11
no gtex brain data for: AL590683.2
no gtex brain data for: AL391730.1
no gtex brain data for: AL355795.1
no gtex brain data for: RP11-25K21.6
no gtex brain data for: RP11-222A5.1
no gtex brain data for: AL590085.1
no gtex brain data for: RP11-134G8.6
no gtex brain data for: AL138925.1
no gtex brain data for: RP11-420K10.1
no gtex brain data for: RP11-548K23.11
no gtex brain data for: OBFC1
no gtex brain data for: OBFC1
no gtex brain data for: RP11-179H18.5
no gtex brain data for: AP002498.1
no gtex brain data for: RP11-95F22.1
no gtex brain data for: AL132988.1
no gtex brain data for: AL358340.1
no gtex brain data for: AF111168.4
no gtex brain data for: AC104002.1
no gtex brain data for: RP11-680G10.1
no gtex brain data for: AC010311.1
no gtex brain data for: AC010311.1
no gtex brain data for: CTC-548K16.2
no gtex brain data for: CTC-548K16.2
no gtex brain data for: AC092580.4
no gtex brain data for: AC096772.6
no gtex brain data for: AC093865.1
no gtex brain data for: AL035106.1
no gtex brain data for: AP000320.7
no gtex brain data for: AC121332.1
no gtex brain data for: NPHP3-ACAD11
no gtex brain data for: AC063932.1
no gtex brain data for: RP11-215A19.2
no gtex brain data for: CTD-3224K15.2
no gtex brain data for: CTB-35F21.2
no gtex brain data for: CTD-3224K15.2
no gtex brain data for: CTB-35F21.2
no gtex brain data for: CTD-3224K15.2
no gtex brain data for: AC005592.1
no gtex brain data for: AC005592.1
no gtex brain data for: RP1-167F1.2
no gtex brain data for: ATP6V1G2-DDX39B
no gtex brain data for: XXbac-BPG32J3.22
no gtex brain data for: MSH5-SAPCD1
no gtex brain data for: AC006483.1
no gtex brain data for: AC004520.1
no gtex brain data for: AC004520.1
no gtex brain data for: RP4-777O23.3
no gtex brain data for: RP5-1165K10.2
no gtex brain data for: RP5-1165K10.2
no gtex brain data for: WBSCR16
no gtex brain data for: WBSCR16
no gtex brain data for: AC006014.7
no gtex brain data for: RP11-514P8.8
no gtex brain data for: MESTIT1_1
no gtex brain data for: MESTIT1_2
no gtex brain data for: MESTIT1_3
no gtex brain data for: AC104133.1
no gtex brain data for: AP003356.1
no gtex brain data for: AC022909.1
no gtex brain data for: KIAA0196
no gtex brain data for: GS1-393G12.12
no gtex brain data for: MIR6847
no gtex brain data for: GS1-393G12.12
no gtex brain data for: MIR6847
no gtex brain data for: GS1-393G12.12
no gtex brain data for: CDKN2B-AS_3
no gtex brain data for: RP11-15J10.1
no gtex brain data for: RP11-548B3.3
no gtex brain data for: PPP2R4
no gtex brain data for: RP11-247A12.8
no gtex brain data for: AF241734.1
```

[78]:
```
rare_enriched_promoter_snps_motif_gene.head()
```

```
[78]:    CHROM        POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
      0  chr19   45783029   T   G   2  0.043  46           0.000076
      1   chr2  222320241   T   C   4  0.087  46           0.000200
      2  chr17    6556629   G   C   2  0.091  22           0.001300
      3   chr2   26692643   C   G   2  0.048  42           0.000046
      4  chr12  106247724   G   T   2  0.063  32           0.000200

         gnomAD_genome_NFE  binom_pval  B-H_reject_H0  corrected_binom_pval  \
      0             0.0000    0.000000           True              0.000000
      1             0.0000    0.000000           True              0.000000
      2             0.0017    0.000653           True              0.004048
      3             0.0000    0.000000           True              0.000000
      4             0.0003    0.000044           True              0.000320

                   motif_best_match         motif_highest_diff genomic element  \
      0  KLF15_HUMAN.H11MO.0.A:0.93  ZN770_HUMAN.H11MO.0.C:3.80        promoter
      1   E2F6_HUMAN.H11MO.0.A:0.98   E2F6_HUMAN.H11MO.0.A:3.84        promoter
      2    MAZ_HUMAN.H11MO.1.A:0.96    MAZ_HUMAN.H11MO.1.A:7.56        promoter
      3  ZN740_HUMAN.H11MO.0.D:0.99  ZN740_HUMAN.H11MO.0.D:4.41        promoter
      4   OSR2_HUMAN.H11MO.0.C:0.95    WT1_HUMAN.H11MO.0.C:3.25        promoter

                                                    Gene  \
      0                            ENSG00000104936/DMPK
      1                      ENSG00000237732/AC010980.2
      2                         ENSG00000091622/PITPNM3
      3                           ENSG00000171303/KCNK3
      4  ENSG00000136026/CKAP4,ENSG00000258355/RP11-651…

        Median TPM in brain tissues in GTEx
      0                          DMPK:37.81
      1                     AC010980.2:2.01
      2                       PITPNM3:29.02
      3                          KCNK3:6.22
      4       CKAP4:11.70,RP11-651L5.2:0.02
```

```
[79]: rare_enriched_enhancer_snps_motif_genes_collected_corelations.head()
```

```
[79]:   CHROM       POS REF ALT  AC     AF  AN  gnomAD_genome_ALL  \
      0  chr1   6868940   G   C   1  0.022  46           0.000200
      1  chr1   8890584   T   C   1  0.022  46           0.000100
      2  chr1  10144750   G   A   2  0.043  46           0.000100
      3  chr1  10144754   G   A   2  0.043  46           0.000035
      4  chr1  21254094   A   G   2  0.048  42           0.000037

         gnomAD_genome_NFE  binom_pval  …  corrected_binom_pval  \
      0           0.000000    0.000000  …              0.000000
      1           0.000000    0.000000  …              0.000000
```

```
2          0.000200    0.000041  …            0.000384
3          0.000071    0.000005  …            0.000055
4          0.000073    0.000005  …            0.000050

            motif_best_match          motif_highest_diff enh_start   enh_end  \
0   FLI1_HUMAN.H11MO.1.A:0.96   ETS1_HUMAN.H11MO.0.A:1.60   6868223   6870996
1  PRDM6_HUMAN.H11MO.0.C:0.96   IRF2_HUMAN.H11MO.0.A:3.86   8888663   8891414
2  CPEB1_HUMAN.H11MO.0.D:0.90  ZFP28_HUMAN.H11MO.0.C:1.89  10143598  10144774
3  PRDM6_HUMAN.H11MO.0.C:1.00  CPEB1_HUMAN.H11MO.0.D:3.01  10143598  10144774
4   IRF2_HUMAN.H11MO.0.A:0.89   IRF5_HUMAN.H11MO.0.D:2.20  21253526  21255567

                                                 Gene      genomic element  \
0  ENSG00000171735/CAMTA1(containing);ENSG0000022…    enhancer intronic
1  ENSG00000238249/HMGN2P17(closest);ENSG00000116…  enhancer intergenic
2  ENSG00000130939/UBE4B(containing);ENSG00000201…    enhancer intronic
3  ENSG00000130939/UBE4B(containing);ENSG00000201…    enhancer intronic
4  ENSG00000117298/ECE1(containing);ENSG000002369…    enhancer intronic

  H3K27ac-expression correlation p-values  \
0                                       .
1                                       .
2                                       .
3                                       .
4            ECE1/ENST00000415912/0.06776

  Putative target with highest correlation  \
0                                        .
1                                        .
2                                        .
3                                        .
4             ECE1/ENST00000415912/0.06776

            Median TPM in brain tissues in GTEx
0                CAMTA1:19.49,RP11-312B8.2:0.00
1  HMGN2P17:0.02,ERRFI1:16.11,RP11-431K24.1:0.16
2                    UBE4B:18.16,RNU6-828P:0.02
3                    UBE4B:18.16,RNU6-828P:0.02
4    ECE1:13.67,RP3-329E20.2:0.01,RAP1GAP:103.43

[5 rows x 21 columns]
```

10 Save output to files

[80]:
```
rare_enriched_promoter_snps_motif_gene.to_csv(ANNOTATED_PROMOTER_SNPs, sep =␣
↪"\t", index = False)
rare_enriched_enhancer_snps_motif_genes_collected_corelations.
↪to_csv(ANNOTATED_ENHANCER_SNPs, sep = "\t", index = False)
```

[81]:
```
t2 = pd.to_datetime('today')
t2
```

[81]:
```
Timestamp('2021-03-15 12:55:41.025907')
```

[ ]:
```
```