main_transcripts

September 2, 2021

1 Enrichment and Overlap of PGC2+CLOZUK

[1]:
```
import re
import os, errno
import functools
import numpy as np
import pandas as pd
from plotnine import *
from pandas_plink import read_plink
from warnings import filterwarnings
from matplotlib.cbook import mplDeprecation
from scipy.stats import fisher_exact, binom_test

filterwarnings("ignore",category=mplDeprecation)
filterwarnings('ignore', category=UserWarning, module='plotnine.*')
filterwarnings('ignore', category=DeprecationWarning, module='plotnine.*')
```

1.1 Config and Functions

[2]:
```
config = {
    'biomart_file': '../_h/biomart.csv',
    'phenotype_file': '/ceph/projects/v4_phase3_paper/inputs/phenotypes/_m/
 ↪merged_phenotypes.csv',
    'plink_file_prefix': '/ceph/projects/v4_phase3_paper/inputs/genotypes/_m/
 ↪LIBD_Brain_TopMed',
    'gwas_snp_file': '/ceph/projects/v4_phase3_paper/inputs/sz_gwas/pgc2_clozuk/
 ↪map_phase3/_m/libd_hg38_pgc2sz_snps.tsv'
}

config_feature = {
    'de_file': '../../differential_expression/_m/transcripts/
 ↪diffExpr_szVctl_full.txt',
    'residual_expression_file': '../../differential_expression/_m/transcripts/
 ↪residualized_expression.tsv',
    'fastqtl_output_file': '../../eqtl/caudate/summary_table/_m/
 ↪Brainseq_LIBD_caudate_4features.signifpairs.txt.gz',
}
```

```
feature = "transcripts"
```

[3]:
```
@functools.lru_cache()
def feature_map(feature):
    return {"genes": "Gene", "transcripts": "Transcript",
            "exons": "Exon", "junctions": "Junction"}[feature]


@functools.lru_cache()
def get_de_df():
    """
    Load DE analysis
    """
    return pd.read_csv(config_feature['de_file'], sep='\t', index_col=0)


@functools.lru_cache()
def get_eqtl_df():
    eqtl_df = pd.read_csv(config_feature['fastqtl_output_file'], sep='\t')
    return eqtl_df[(eqtl_df["Type"] == feature_map(feature))]


@functools.lru_cache()
def get_gwas_snps():
    return pd.read_csv(config['gwas_snp_file'], sep='\t', index_col=0,␣
 ↪low_memory=False)


@functools.lru_cache()
def get_integration_df():
    return get_gwas_snps().merge(get_eqtl_df(), left_on='our_snp_id',␣
 ↪right_on='variant_id',
                                 suffixes=['_PGC2', '_eQTL'])\
                        .merge(get_de_df(), left_on='gene_id',␣
 ↪right_index=True)


@functools.lru_cache()
def get_residual_expression_df():
    return pd.read_csv(config_feature['residual_expression_file'],
                       sep='\t', index_col=0).transpose()


@functools.lru_cache()
def get_pheno_df():
    return pd.read_csv(config['phenotype_file'], index_col=0)
```

[4]:
```
def agree_direction(row):
    return [-1, 1][row['pgc2_a1_same_as_our_counted']] * np.sign(row['OR'] - 1)␣
 ↪* np.sign(row['slope']) * np.sign(row['t'])


def letter_snp(number, a0, a1):
    '''
    Example:
    letter_snp(0, 'A', 'G') is 'AA'
    letter_snp(1, 'A', 'G') is 'AG'
    letter_snp(2, 'A', 'G') is 'GG'

    '''
    if np.isnan(number):
        return np.nan
    if len(a0)==1 and len(a1)==1:
        sep = ''
    else:
        sep = ' '
    return sep.join(sorted([a0]*int(number) + [a1]*(2-int(number))))


def get_gwas_snp(snp_id):
    gwas = get_gwas_snps()
    r = gwas[gwas['our_snp_id']==snp_id]
    assert len(r) == 1
    return r
```

[5]:
```
@functools.lru_cache()
def get_expression_and_pheno_df():
    return pd.merge(get_pheno_df(), get_residual_expression_df(),␣
 ↪left_index=True, right_index=True)


@functools.lru_cache()
def get_plink_tuple():
    '''
    Usage: (bim, fam, bed) = get_plink_tuple()
    '''
    return read_plink(config['plink_file_prefix'])


@functools.lru_cache()
def subset_bed():
    """
    This subsets the bed and bim file and returns the new subsetted
    data with shared brain_ids.
```

```
    This is to speed up accessing the bed file.
    """
    (bim, fam, bed) = get_plink_tuple()
    brain_ids = list(set(get_expression_and_pheno_df()['BrNum']).
↪intersection(set(fam['fid'])))
    fam_pos = list(fam[(fam["fid"].isin(brain_ids))].
↪drop_duplicates(subset="fid").loc[:, 'i'])
    unique_snps = get_eqtl_df().variant_id.unique()
    snp_info = bim[(bim["snp"].isin(unique_snps))].copy()
    snp_pos = list(snp_info.loc[:, "i"])
    new_bed = bed[snp_pos].compute()[:,fam_pos]
    new_bim = bim[(bim["i"].isin(snp_pos))].reset_index(drop=True)
    new_bim['ii'] = new_bim.index
    return new_bed, new_bim, brain_ids


@functools.lru_cache()
def get_snp_df(snp_id):
    '''
    Returns a dataframe containing the genotype on snp snp_id.
    The allele count is the same as in the plink files.

    Example:
    get_snp_df('rs653953').head(5)

            rs653953_num rs653953_letter rs653953
    Br5168             0              GG    0\nGG
    Br2582             1              AG    1\nAG
    Br2378             1              AG    1\nAG
    Br5155             2              AA    2\nAA
    Br5182             2              AA    2\nAA
    '''
    bed, bim, brain_ids = subset_bed()
    snp_info = bim[bim['snp']==snp_id]
    snp_pos = snp_info.iloc[0]['ii']
    dfsnp = pd.DataFrame(bed[[snp_pos]], columns=brain_ids, index=[snp_id +␣
↪'_num']).transpose().dropna()
    my_letter_snp = functools.partial(letter_snp, a0=snp_info.iloc[0]['a0'],␣
↪a1=snp_info.iloc[0]['a1'])
    # the 2 - in next line is to workaround a possible bug in pandas_plink? a1␣
↪and a0 inverted
    dfsnp[[snp_id + '_num']] = 2 - dfsnp[[snp_id + '_num']].astype('int')
    dfsnp[snp_id + '_letter'] = dfsnp[snp_id + '_num'].apply(my_letter_snp)
    dfsnp[snp_id] = (dfsnp[snp_id + '_num'].astype('str') + '\n' +
                     dfsnp[snp_id + '_letter'].astype('str')).astype('category')
    return dfsnp
```

```
@functools.lru_cache()
def get_gwas_ordered_snp_df(snp_id):
    '''
    Returns a dataframe containing the genotype on snp snp_id.
    The allele count is the number of risk alleles according to GWAS.

    Example:
    get_gwas_ordered_snp_df('rs653953').head(5)

            rs653953_num rs653953_letter rs653953
    Br5168             2              GG    2\nGG
    Br2582             1              AG    1\nAG
    Br2378             1              AG    1\nAG
    Br5155             0              AA    0\nAA
    Br5182             0              AA    0\nAA
    '''
    pgc = get_gwas_snps()
    dfsnp = get_snp_df(snp_id).copy()
    gwas_snp = get_gwas_snp(snp_id)

    if gwas_snp['pgc2_a1_same_as_our_counted'].iloc[0]:
        if gwas_snp['OR'].iloc[0] > 1:
            pass
        else:
            dfsnp[[snp_id + '_num']] = 2 - dfsnp[[snp_id + '_num']]
    else:
        if gwas_snp['OR'].iloc[0] > 1:
            dfsnp[[snp_id + '_num']] = 2 - dfsnp[[snp_id + '_num']]
        else:
            pass

    dfsnp[snp_id] = (dfsnp[snp_id + '_num'].astype('str') + '\n' +
                     dfsnp[snp_id + '_letter'].astype('str')).astype('category')
    return dfsnp
```

[6]:
```
@functools.lru_cache()
def get_biomart_df():
    biomart = pd.read_csv(config['biomart_file'])
    biomart['description'] = biomart['description'].str.replace('\[Source.
 ↪*$','', regex=True)
    return biomart


def get_gene_symbol(gene_id, biomart=get_biomart_df()):
    ensge = re.sub('\..+$','', gene_id)
```

```
    ggg = biomart[biomart['ensembl_gene_id']==ensge]
    if ggg.shape[0]==0:
        return '', ''

    gs = ggg['external_gene_name'].values[0]
    de = ggg['description'].values[0]
    if type(de)!=str:
        de = ''
    de = re.sub('\[Source:.*$','',de)
    return gs, de


@functools.lru_cache()
def get_risk_allele(snp_id):
    gwas_snp = get_gwas_snp(snp_id)
    if gwas_snp['OR'].iloc[0] > 1:
        ra = gwas_snp['A1'].iloc[0]
    else:
        ra = gwas_snp['A2'].iloc[0]

    return ra
```

[7]:
```
def get_snp_gene_pheno_df(snp_id, gene_id, snp_df_func):
    pheno_columns = list(get_pheno_df().columns)
    expr_df = get_expression_and_pheno_df()[pheno_columns + [gene_id]]
    snp_df =  snp_df_func(snp_id)
    return expr_df.merge(snp_df, left_on='BrNum', right_index=True)


def simple_snp_expression_pheno_plot_impl(snp_id, gene_id, snp_df_func,␣
 ↪pheno_var):
    df = get_snp_gene_pheno_df(snp_id, gene_id, snp_df_func)
    df['Dx'] = df.Dx.astype('category').cat.rename_categories({'Control':␣
 ↪'CTL', 'Schizo': 'SZ'})
    y0 = df[gene_id].quantile(.01) - 0.26
    y1 = df[gene_id].quantile(.99) + 0.26
    pjd = position_jitterdodge(jitter_width=0.27)
    p = ggplot(df, aes(x=snp_id, y=gene_id, fill=pheno_var)) \
    + geom_boxplot(alpha=0.4, outlier_alpha=0) \
    + geom_jitter(position=pjd, stroke=0, alpha=0.6) + ylim(y0, y1) \
    + labs(y='Residualized expression', fill='Diagnosis') \
    + theme_bw(base_size=20)\
    + theme(legend_title=element_text(face='bold'),
            panel_grid_major=element_blank(),
            panel_grid_minor=element_blank())
    return p
```

```
def simple_gwas_ordered_snp_expression_pheno_plot(snp_id, gene_id, pheno_var):
    return simple_snp_expression_pheno_plot_impl(snp_id, gene_id,␣
 ↪get_gwas_ordered_snp_df, pheno_var)
```

[8]:
```
def save_plot(p, fn):
    for ext in ['png', 'pdf', 'svg']:
        p.save(fn + '.' + ext)


def gwas_annotation(snp_id):
    return 'SZ GWAS pvalue: %.1e' % get_gwas_snp(snp_id).iloc[0]['P']


def eqtl_annotation(snp_id, gene_id):
    r = get_eqtl_df()[(get_eqtl_df()['variant_id']==snp_id) &
                      (get_eqtl_df()['gene_id']==gene_id)]
    assert len(r)==1
    return 'eQTL nominal p-value: %.1e' % r.iloc[0]['pval_nominal']


def de_annotation(gene_id):
    g = get_de_df()[(get_de_df()['transcript_id'] == gene_id)]
    return 'DE adj.P.Val: %.3f' % g.iloc[0]['adj.P.Val']


def risk_allele_annotation(snp_id):
    return 'SZ risk allele: %s' % get_risk_allele(snp_id)


def gwas_annotated_eqtl_pheno_plot(snp_id, gene_id, pheno_var):
    p = simple_gwas_ordered_snp_expression_pheno_plot(snp_id, gene_id,␣
 ↪pheno_var)
    de_df = get_de_df()[(get_de_df()['transcript_id'] == gene_id)]
    gene_symbol, gene_description = get_gene_symbol(de_df.iloc[0]['gene_id'])

    title ="\n".join([gene_symbol,
                      gene_id,
                      gwas_annotation(snp_id),
                      risk_allele_annotation(snp_id),
                      eqtl_annotation(snp_id, gene_id),
                      de_annotation(gene_id)])

    p += ggtitle(title)
    return p
```

1.2 Transcripts

[9]:
```
try:
    os.makedirs(feature)
except OSError as e:
    if e.errno != errno.EEXIST:
        raise
```

1.2.1 Enrichment

Integrate DEG with PGC2+CLOZUK SNPs

[10]:
```
dft = get_integration_df()
dft.shape
```

```
/home/jbenja13/.local/lib/python3.9/site-packages/numpy/lib/arraysetops.py:583:
FutureWarning: elementwise comparison failed; returning scalar instead, but in
the future will perform elementwise comparison
```

[10]:
```
(2280801, 65)
```

[11]:
```
agreement = {1: 'Yes', -1: 'No', 0: 0}
dft['agree_direction'] = dft.apply(agree_direction, axis=1)
dft.agree_direction = [agreement[item] for item in dft['agree_direction']]

table =  [[np.sum((dft['P']<5e-8) & ((dft['adj.P.Val']<.05))),
           np.sum((dft['P']<5e-8) & ((dft['adj.P.Val']>=.05)))],
          [np.sum((dft['P']>=5e-8) & ((dft['adj.P.Val']<.05))),
           np.sum((dft['P']>=5e-8) & ((dft['adj.P.Val']>=.05)))]]
print(table)
fisher_exact(table)
```

```
[[989, 54364], [48243, 2177205]]
```

[11]:
```
(0.8210127548449392, 4.079815690149912e-10)
```

[12]:
```
dft1 = dft[(dft['P']<5e-8) & ((dft['adj.P.Val']<.05))]
df = dft1.groupby('agree_direction').size().reset_index()
df
```

[12]:

	agree_direction	0
0	No	231
1	Yes	758

[13]:
```
binom_test(df[0].iloc[1], df[0].sum())
```

[13]:
```
4.805232318453914e-66
```

[14]:
```
dft2 = dft[(dft['P']<=5e-8) & (dft['adj.P.Val'] < 0.05)].copy()
dft2['risk_allele'] = dft2['our_snp_id'].apply(get_risk_allele)
```

```
direction = {-1: 'Down', 1: 'Up'}
boolean_conv = {True: 1, False: -1}
dft2.pgc2_a1_same_as_our_counted = [boolean_conv[item] for item in␣
↪dft2['pgc2_a1_same_as_our_counted']]
dft2['eqtl_gwas_dir'] = [direction[item] for item in np.
↪sign(dft2['pgc2_a1_same_as_our_counted']) * np.sign(dft2['slope']) * np.
↪sign(dft2['OR'] - 1)]
dft2['de_dir'] = [direction[item] for item in np.sign(dft2['t'])]
dft2['eqtl_slope'] = np.sign(dft2['pgc2_a1_same_as_our_counted']) * np.
↪sign(dft2['OR'] - 1) * dft2['slope']
dft2 = dft2[['gene_id', 'gene_name', 'variant_id', 'A1', 'A2', 'risk_allele',␣
↪'OR',
            'P', 'pval_nominal', 'adj.P.Val', 'logFC', 't', 'eqtl_slope',
            'de_dir', 'eqtl_gwas_dir', 'agree_direction']]
dft2.to_csv('%s/integration_by_symbol.txt' % feature, sep='\t', index=False)
```

[15]:
```
df2 = dft2.groupby(['gene_id']).first().reset_index().sort_values('P')
df2.groupby(['agree_direction']).size()
```

[15]:
```
agree_direction
No     3
Yes    6
dtype: int64
```

[16]:
```
df2.set_index('gene_name')
```

[16]:

gene_name	gene_id	variant_id	A1	A2	risk_allele	OR	P	pval_nominal	adj.P.Val	logFC	t
HCG11	ENST00000411553.2	chr6:26466161:G:A	G	A	A	0.91432	1.020000e-14	1.890010e-05	0.002565	0.102920	4.413076
ZNF391	ENST00000244576.8	chr6:27280994:A:C	A	C	A	1.07340	1.610000e-13	1.931590e-04	0.036305	0.086460	3.470055
PLCH2	ENST00000378486.7	chr1:2440958:A:G	A	G	G	0.92873	4.630000e-11	1.180570e-08	0.010192	-0.193953	-3.957680
ZSCAN26	ENST00000617168.4	chr6:27910960:T:G	T	G	T	1.07250	6.280000e-11	1.656890e-04	0.016166	-0.419030	-3.793507
CNNM2	ENST00000369878.8	chr10:102825368:C:A	C	A	C	1.06040	1.120000e-09	4.928250e-11	0.000852	0.084962	4.753266
IP6K3	ENST00000293756.4	chr6:33741539:G:A	G	A	G	1.07670	4.780000e-09	6.262500e-08	0.005209	-0.262250	-4.180640
SREBF2	ENST00000361204.8	chr22:41885425:A:G	A	G	A	1.05670					
HCG4	ENST00000418983.1	chr6:29388857:G:C	G	C	C	0.93808					
ZNF14	ENST00000344099.3	chr19:19623068:T:C	T	C	T	1.06130					

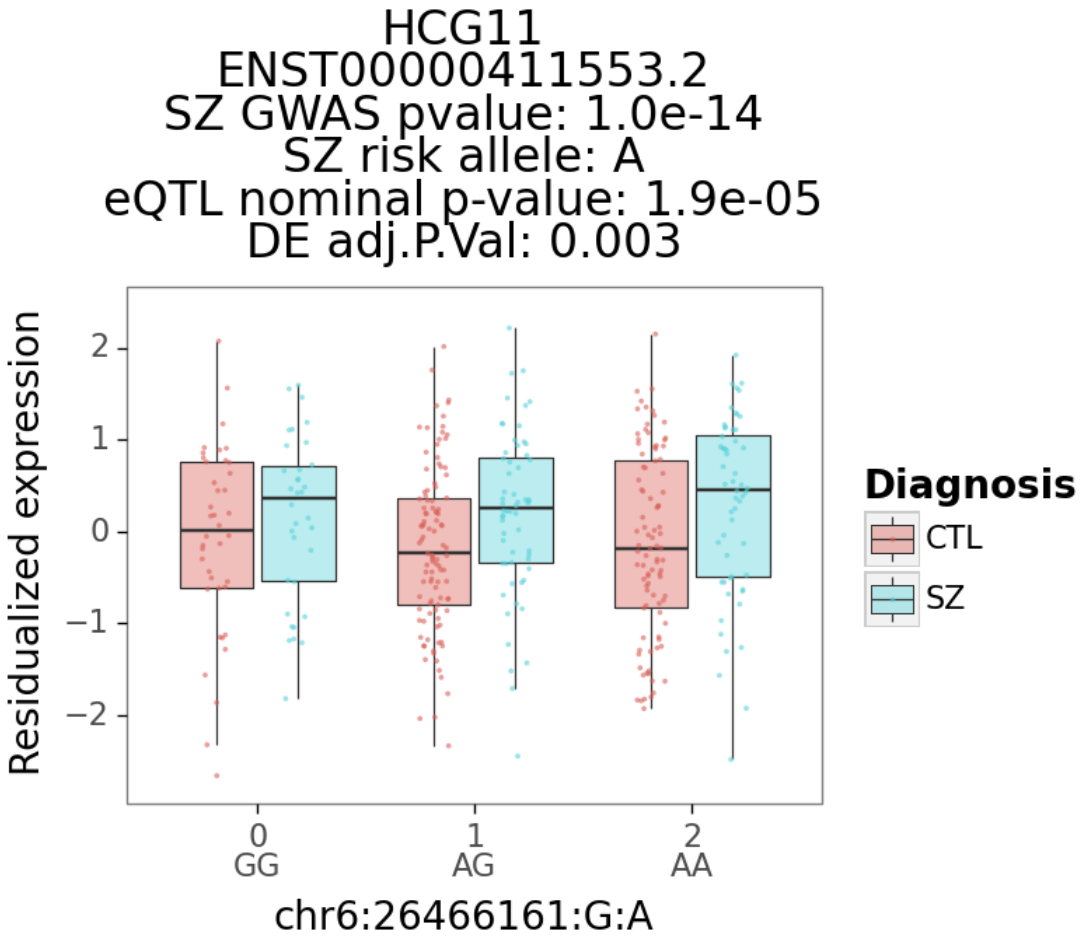
```
SREBF2     8.110000e-09  6.456540e-05   0.018145 -0.073932 -3.747975
HCG4       2.870000e-08  7.612650e-05   0.028722  0.251395  3.568189
ZNF14      4.300000e-08  2.146240e-05   0.047456  0.077718  3.357046

           eqtl_slope de_dir eqtl_gwas_dir agree_direction
gene_name
HCG11        0.201059     Up            Up             Yes
ZNF391      -0.170013     Up          Down              No
PLCH2       -0.266826   Down          Down             Yes
ZSCAN26     -0.238187   Down          Down             Yes
CNNM2       -0.179745     Up          Down              No
IP6K3        0.221272   Down            Up              No
SREBF2      -0.182903   Down          Down             Yes
HCG4         0.220492     Up            Up             Yes
ZNF14        0.156479     Up            Up             Yes
```

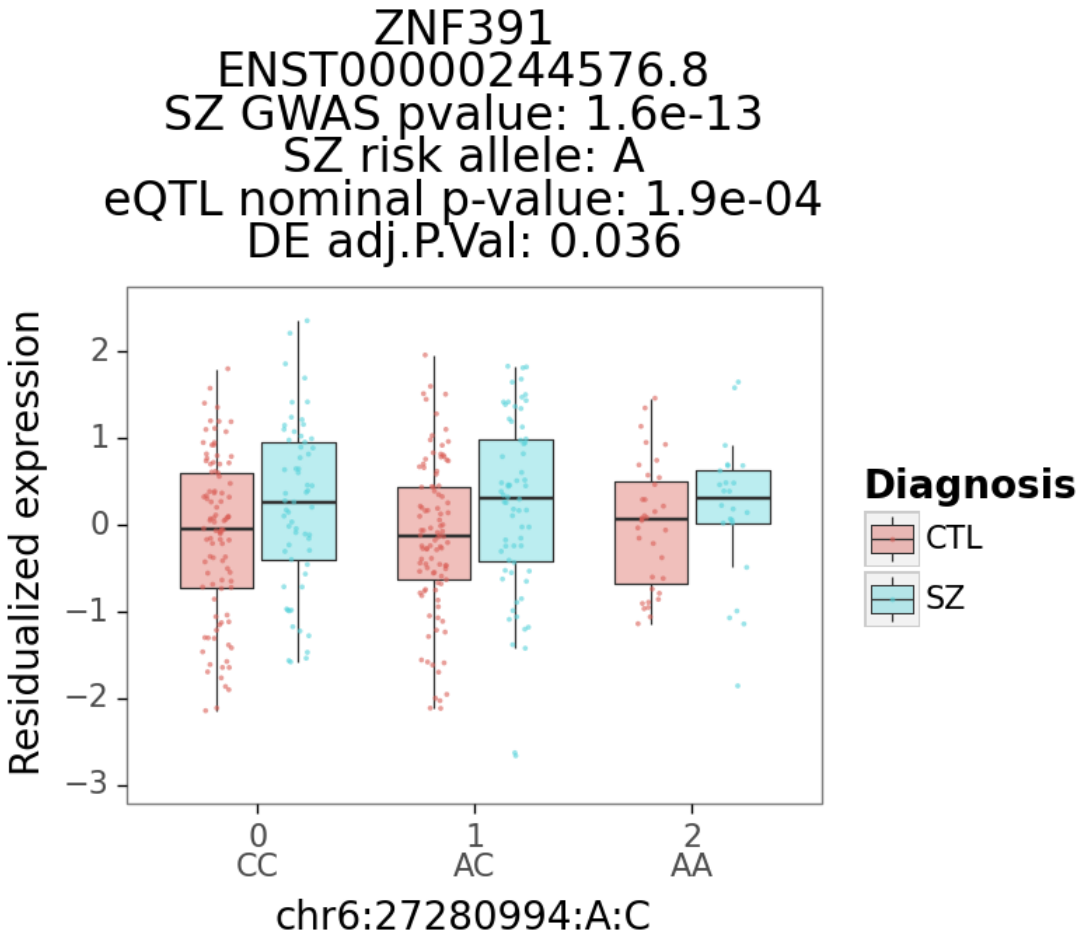
1.2.2 Plot with PGC2 risk allele

[17]:
```
for xx in range(df2.shape[0]):
    gg = gwas_annotated_eqtl_pheno_plot(df2.iloc[xx, :].variant_id, df2.
 ↪iloc[xx, :].gene_id, 'Dx')
    print(gg)
    label = '%s/eqtl_gwas_%s' % (feature, df2.iloc[xx, :].gene_name)
    save_plot(gg, label)
```

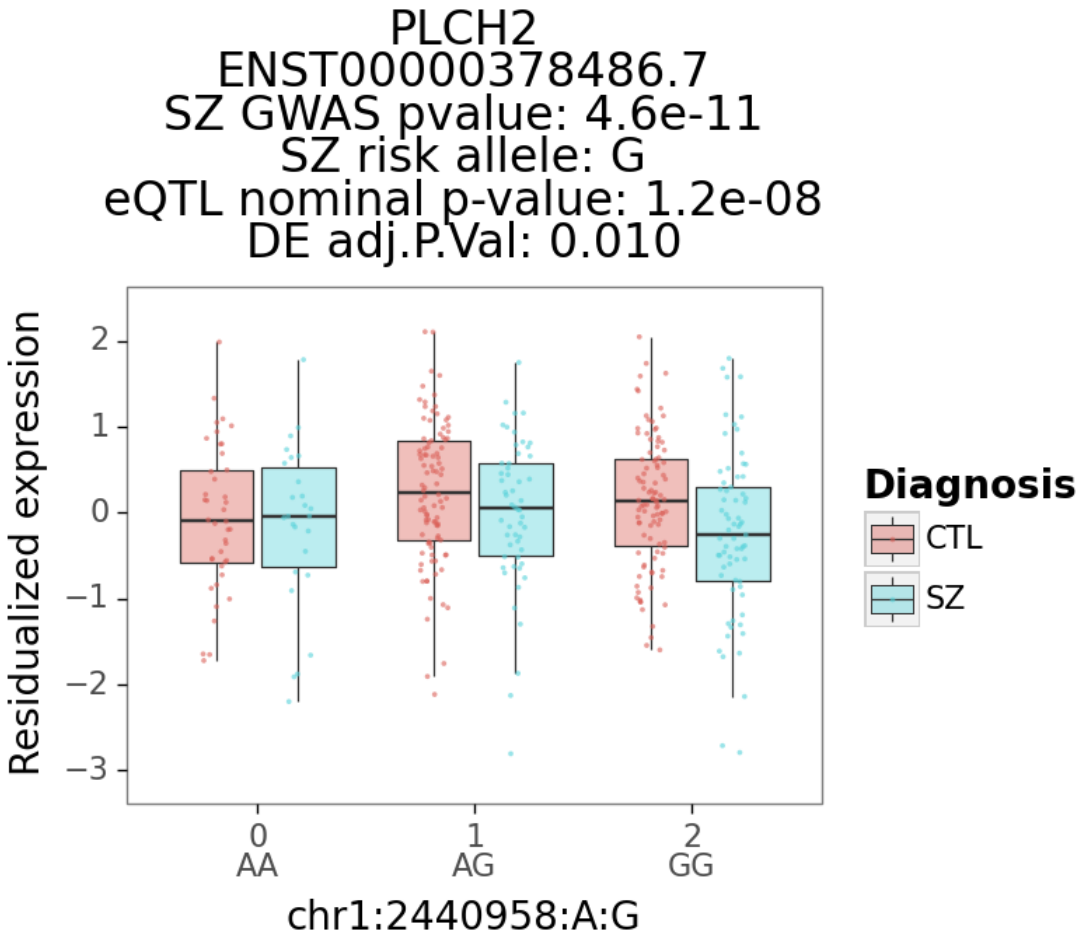
```
Mapping files: 100%|      | 3/3 [00:25<00:00,  8.64s/it]
```



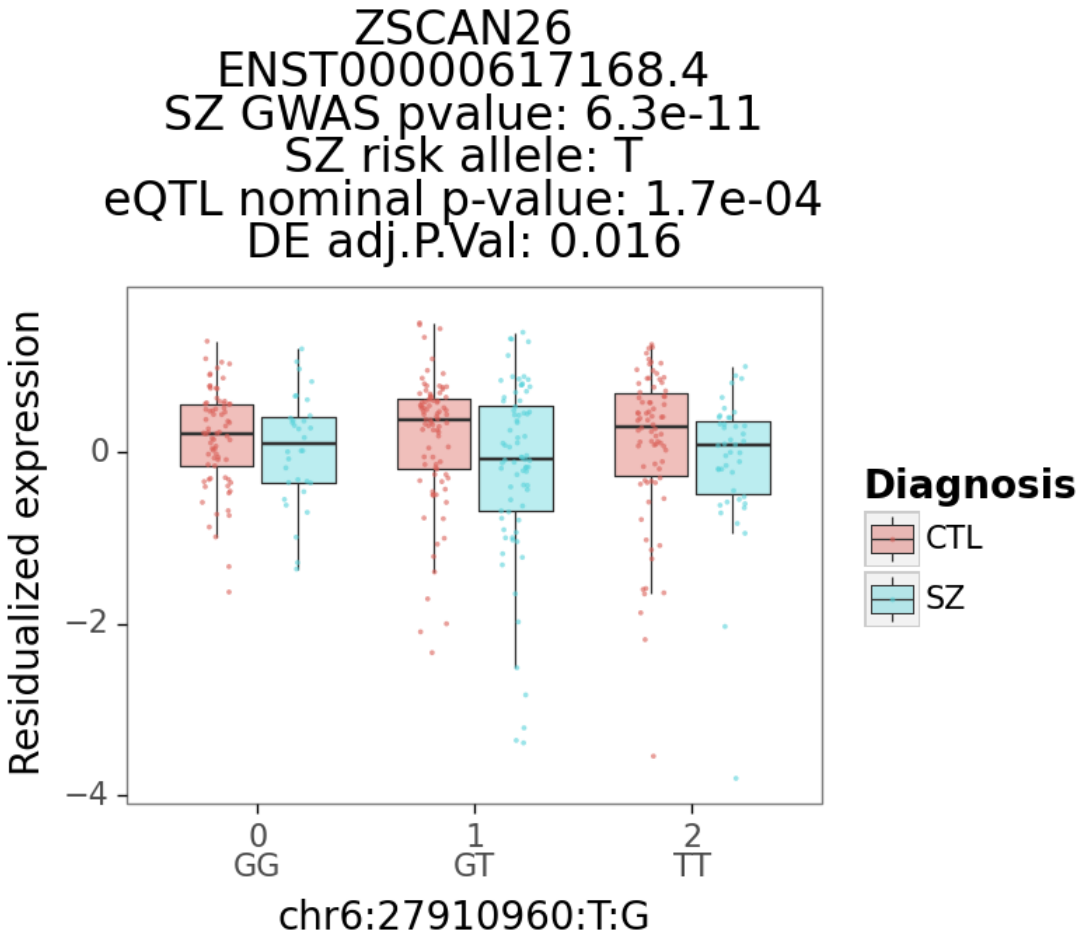

```
<ggplot: (8729546831080)>
```



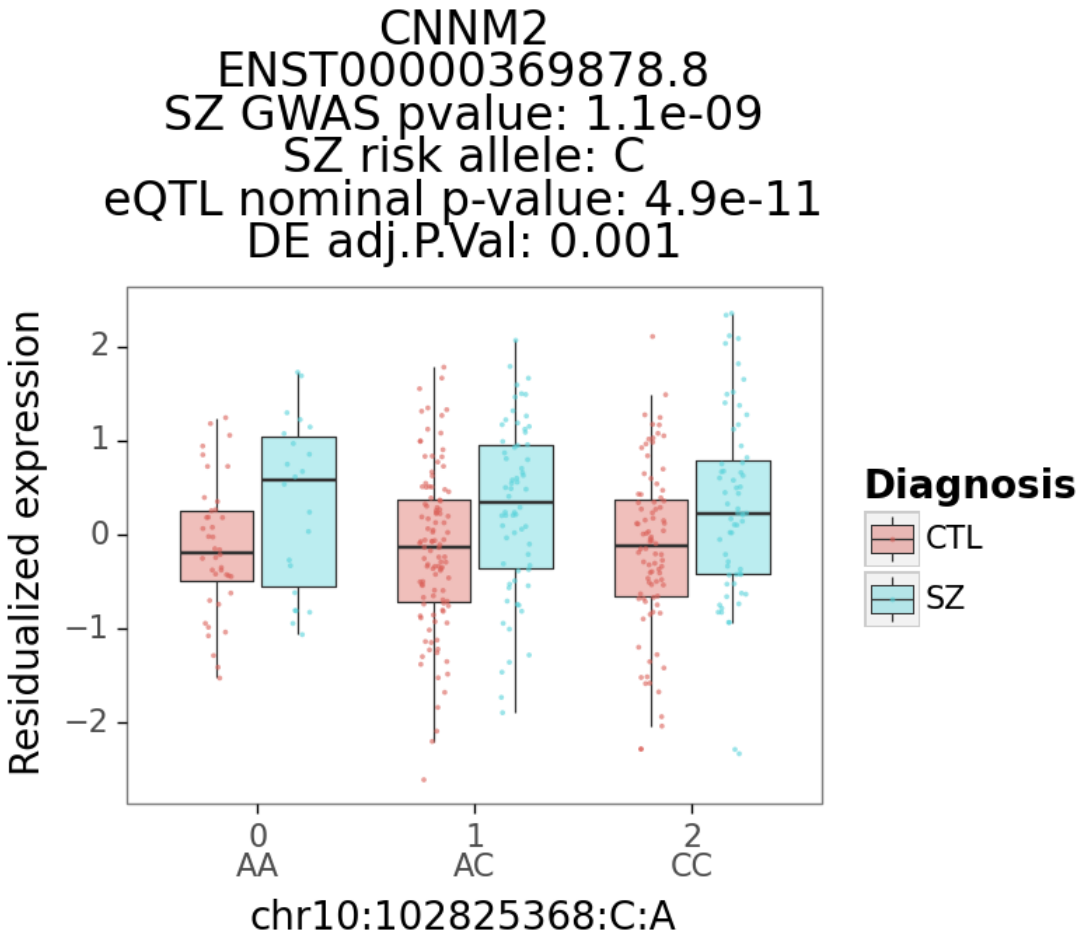

```
<ggplot: (8729548902684)>
```



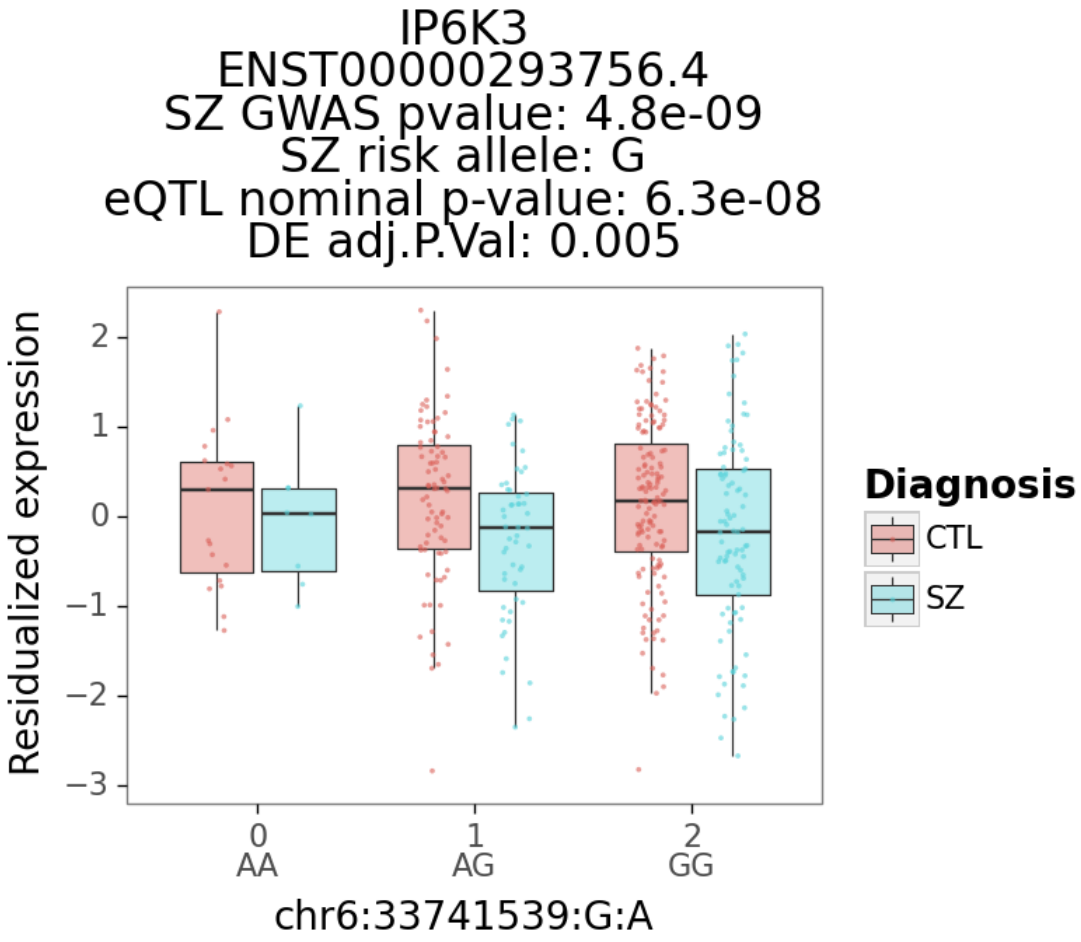

```
<ggplot: (8729548537971)>
```



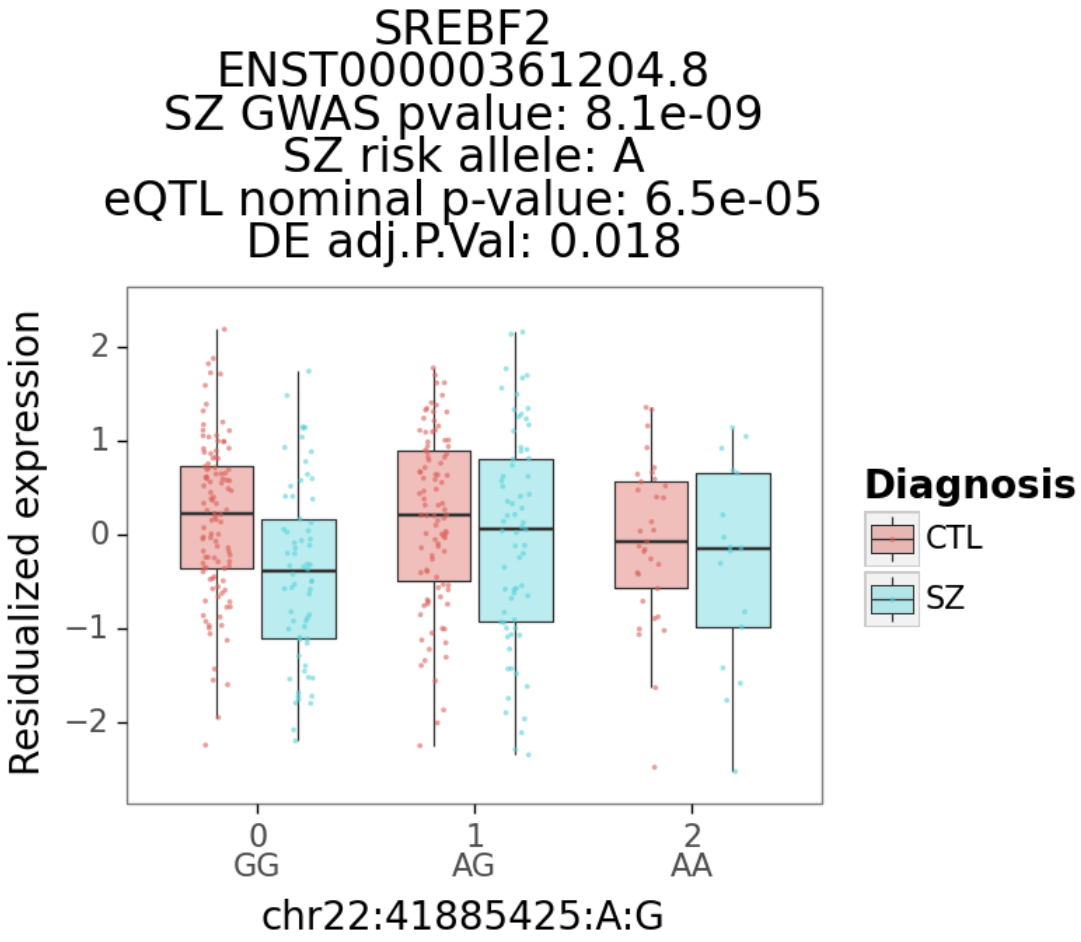

```
<ggplot: (8729549316563)>
```



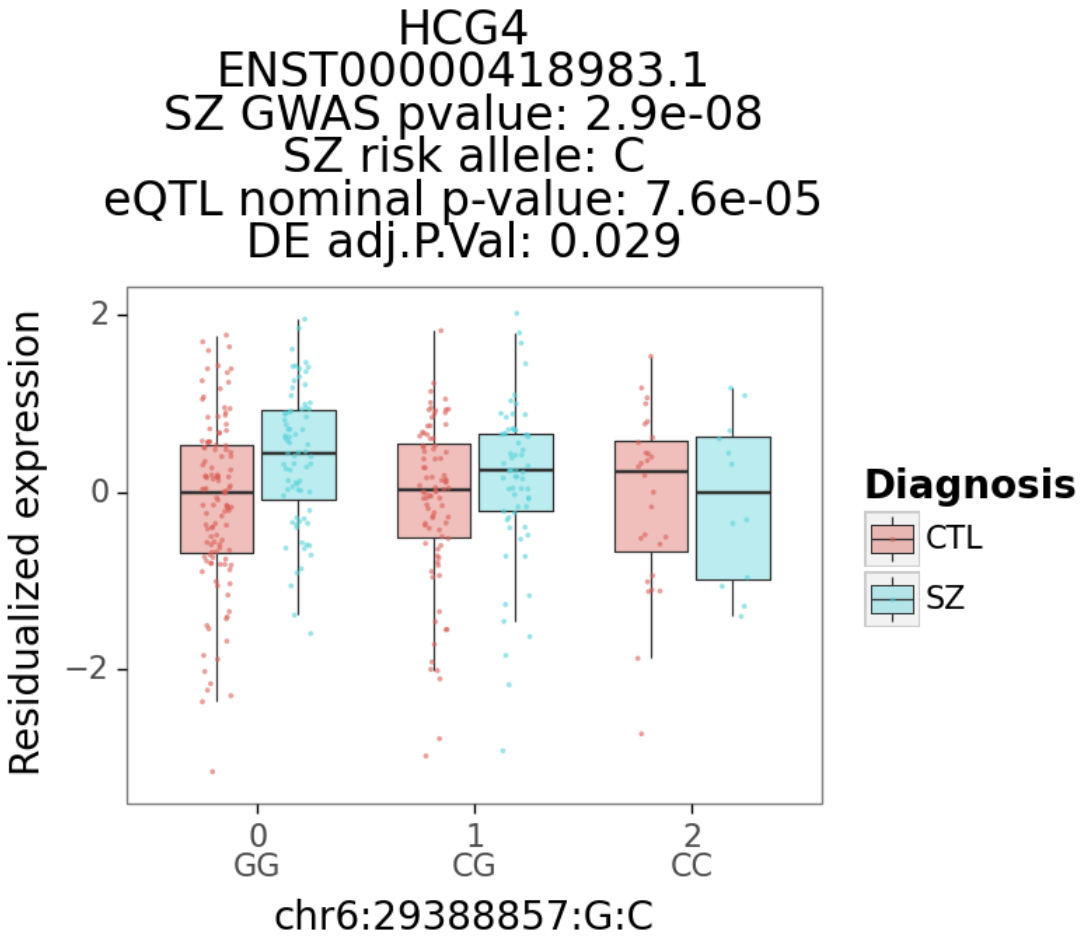

```
<ggplot: (8729515558193)>
```



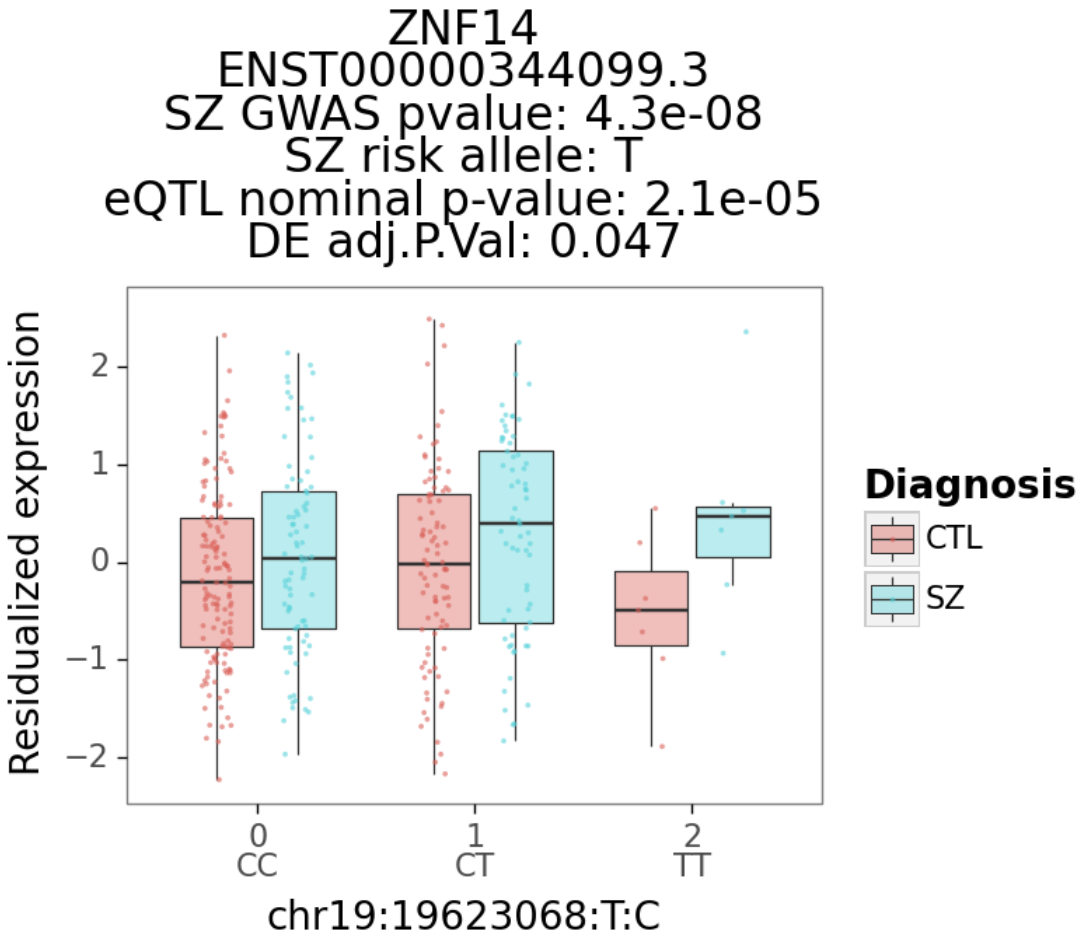

```
<ggplot: (8729515224149)>
```




```
<ggplot: (8729514559657)>
```




```
<ggplot: (8729513879065)>
```




```
<ggplot: (8729515214743)>
```