

# main

August 21, 2021

## 1 PCA analysis of predictiveness of DEG for gender

```
[1]: #load required packages
import functools
import numpy as np
import pandas as pd
from plotnine import *
from scipy.stats import linregress
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

from warnings import filterwarnings
from matplotlib.cbook import mplDeprecation

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

### 1.1 Configuration and functions

#### 1.1.1 Configuration

```
[3]: feature = 'genes'
config = {
    'deg_file': '../_m/%s/diffExpr_EAvsAA_full.txt' % feature,
    'res_file': '../_m/%s/residualized_expression.tsv' % feature,
    'pheno_file': '../input/phenotypes/_m/dlpfc_phenotypes.csv',
    'ancestry': "../input/ancestry_structure/structure.
↳out_ancestry_proportion_raceDemo_compare",
}
```

#### 1.1.2 Cached functions

```
[4]: @functools.lru_cache()
def get_ancestry():
    """
    Loads admixture ancestry from STRUCTURE.
    """
```

```

return pd.read_csv(config["ancestry"], sep='\t')

@functools.lru_cache()
def get_deg():
    ''' Take DE genes obtained from limma-voom pipeline.
    '''
    deg = pd.read_csv(config['deg_file'], sep='\t', index_col=0).
    ↪sort_values('adj.P.Val')
    return deg[(deg['adj.P.Val'] < 0.05)]

@functools.lru_cache()
def get_residualized():
    '''Load residualization file.
    '''
    return pd.read_csv(config['res_file'], sep='\t', index_col=0).transpose()

@functools.lru_cache()
def get_deg_res_df(num):
    de_df = get_deg()
    res_df = get_residualized()
    if num == 0:
        geneList = np.array(de_df.index)
    else:
        geneList = np.array(de_df.head(num).index)
    return res_df[geneList]

@functools.lru_cache()
def get_pheno_data():
    return pd.read_csv(config['pheno_file'], index_col=0)\
        .merge(get_ancestry(), left_on="BrNum", right_on="id")\
        .drop_duplicates(subset="BrNum")

```

### 1.1.3 Simple functions

```

[5]: def get_explained_variance(df):
    x = StandardScaler().fit_transform(df)
    pca = PCA(n_components=2).fit(x)
    pc1 = pca.explained_variance_ratio_[0]
    pc2 = pca.explained_variance_ratio_[1]
    print("Explained Variance\nPC1:\t%0.5f\nPC2:\t%0.5f" % (pc1, pc2))

def cal_pca(df):

```

```

x = StandardScaler().fit_transform(df)
pca = PCA(n_components=2).fit_transform(x)
return pd.DataFrame(data=pca, columns=['PC1', 'PC2'], index=df.index)

def get_pca_df(num, deg_fnc):
    """
    num: refers to how many genes we want to include
    if 0, look at all genes, if not 0, look at only those specified genes
    new_pheno: this is the correct size of samples using the the first two
    → columns of residualized expression
        - the residualized expression data frame, has the correct samples
        - output new_pheno shape row numbers should be the same as res_df row
    → numbers
    """
    expr_res = deg_fnc(num)
    pheno_df = get_pheno_data()
    # Generate pheno data frame with correct samples
    new_pheno = pheno_df.merge(expr_res.iloc[:, 0:1], right_index=True,
    → left_on="RNum")\
        .drop(expr_res.iloc[:, 0:1].columns, axis=1)\
        .set_index("RNum")
    principalDf = cal_pca(expr_res)
    get_explained_variance(expr_res)
    return pd.concat([principalDf, new_pheno], axis = 1)

def calculate_corr(xx, yy):
    """This calculates R2 correlation via linear regression:
        - used to calculate relationship between 2 arrays
        - the arrays are principal components 1 or 2 (PC1, PC2) AND ancestry
        - calculated on a scale of 0 to 1 (with 0 being no correlation)
    Inputs:
        x: array of variable of interest (continous or binary)
        y: array of PC
    Outputs:
        1. r2
        2. p-value, two-sided test
            - whose null hypothesis is that two sets of data are uncorrelated
        3. slope (beta): directory of correlations
    """
    slope, intercept, r_value, p_value, std_err = linregress(xx, yy)
    return slope, r_value, p_value

def corr_annotation(dft):
    xx = dft.Eur

```

```

yy = dft.PC1
zz = dft.PC2
slope1, r_value1, p_value1 = calculate_corr(xx, yy)
slope2, r_value2, p_value2 = calculate_corr(xx, zz)
label = 'PC1 R2: %.2f\nP-value: %.2e' % (r_value1**2, p_value1)
print('PC2 R2: %.4f Pval: %.3e' % (r_value2**2, p_value2))
return label

```

### 1.1.4 Plotting functions

```

[6]: def plot_corr_impl(num, deg_fnc):
    pca_df = get_pca_df(num, deg_fnc)
    pca_df['Race'] = pca_df.Race.astype('category').cat\
        .rename_categories({'AA': 'AA', 'CAUC': 'EA'})
    title = '\n'.join([corr_annotation(pca_df)])
    pp = ggplot(pca_df, aes(x='PC1', y='PC2', fill='Race'))\
        + geom_point(alpha=0.75, size=4)\
        + theme_matplotlib()\
        + theme(axis_text_x=element_blank(),
                axis_text_y=element_text(size=18),
                axis_title=element_text(size=21),
                plot_title=element_text(size=22),
                legend_text=element_text(size=16),
                legend_title=element_blank(),
                legend_position="bottom")
    pp += ggtitle(title)
    return pp

def plot_corr(num, deg_fnc):
    return plot_corr_impl(num, deg_fnc)

def save_plot(p, fn, width=7, height=7):
    '''Save plot as svg, png, and pdf with specific label and dimension.'''
    for ext in ['.svg', '.png', '.pdf']:
        p.save(fn+ext, width=width, height=height)

```

## 1.2 PCA

```

[7]: pp = plot_corr(0, get_deg_res_df)
pp

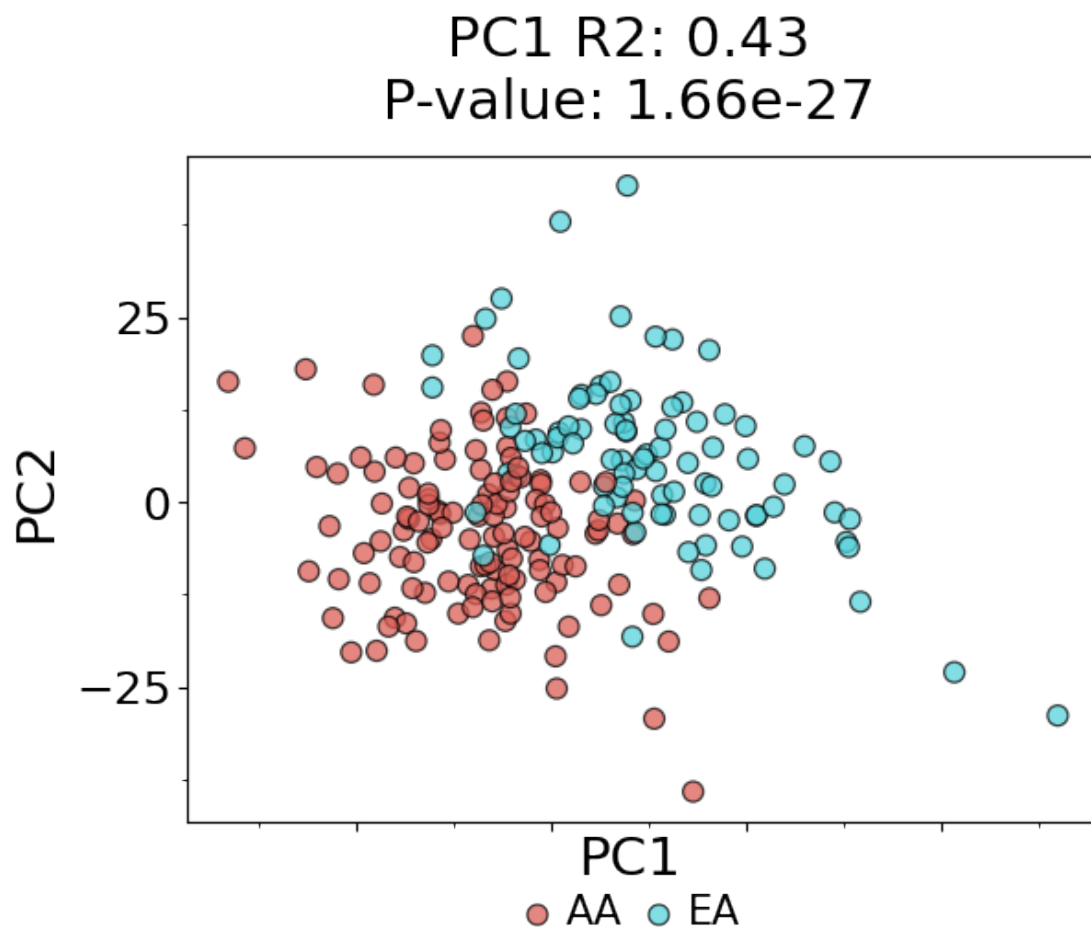
```

Explained Variance

PC1: 0.10080

PC2: 0.04936

PC2 R2: 0.1954 Pval: 1.666e-11



[7]: <ggplot: (8754266681688)>

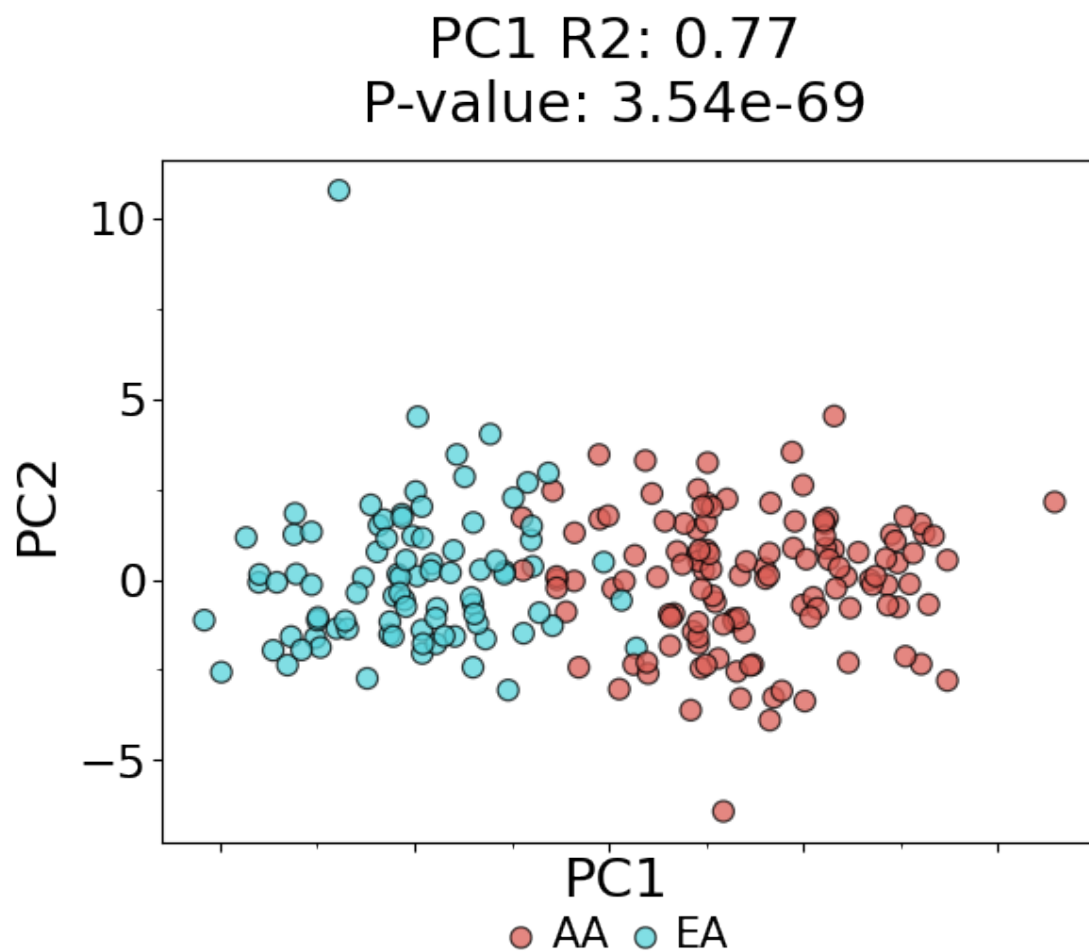
```
[8]: qq = plot_corr(100, get_deg_res_df)
      qq
```

Explained Variance

PC1: 0.25187

PC2: 0.03508

PC2 R2: 0.0005 Pval: 7.367e-01



[8]: <ggplot: (8754267588247)>

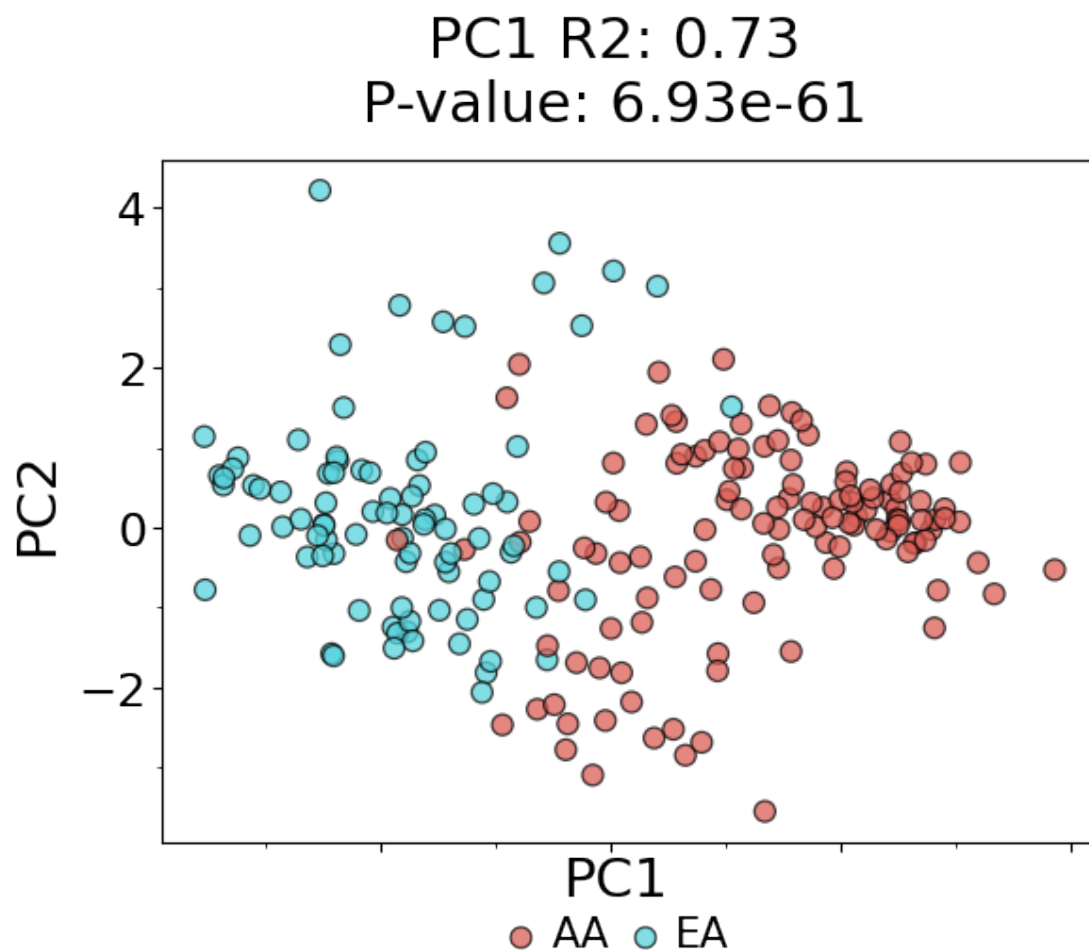
```
[9]: rr = plot_corr(20, get_deg_res_df)
      rr
```

Explained Variance

PC1: 0.39533

PC2: 0.07556

PC2 R2: 0.0109 Pval: 1.301e-01



[9]: <ggplot: (8754267591247)>

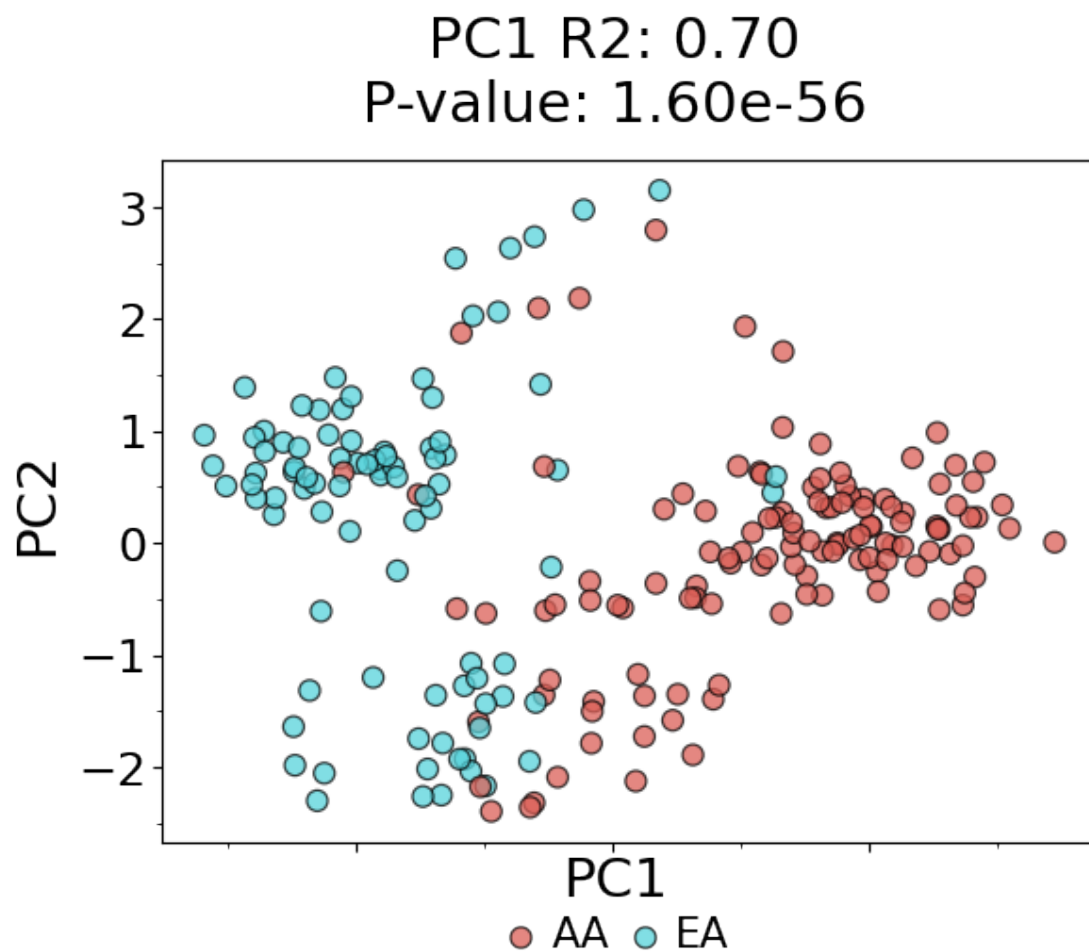
```
[10]: ww = plot_corr(10, get_deg_res_df)
      ww
```

Explained Variance

PC1: 0.49231

PC2: 0.12566

PC2 R2: 0.0109 Pval: 1.303e-01



[10]: <ggplot: (8754267591262)>

```
[11]: save_plot(pp, 'deg_pca_all')
      save_plot(qq, 'deg_pca_top100')
      save_plot(ww, 'deg_pca_top10')
      save_plot(rr, 'deg_pca_top20')
```

**FDR < 0.05 separates out ancestry**

[ ]: