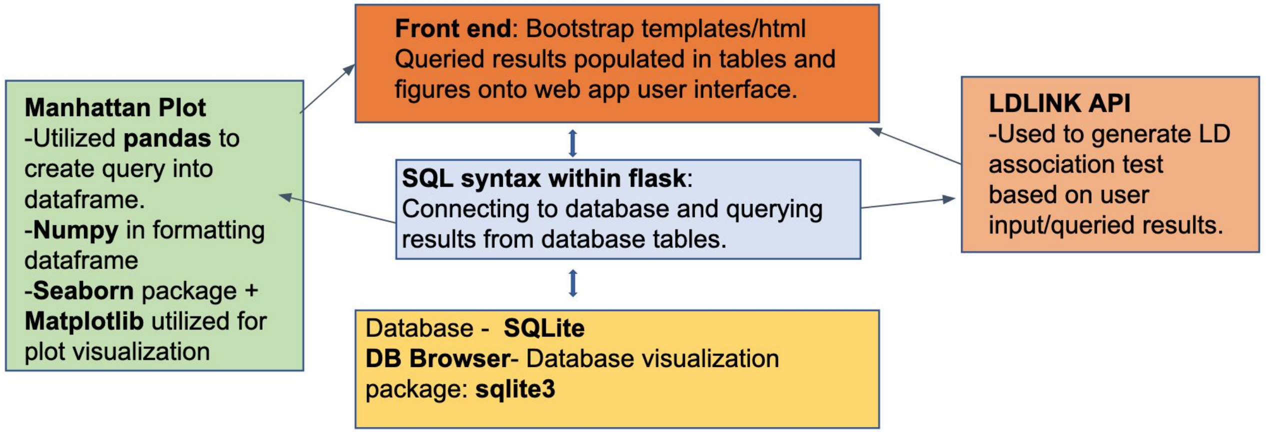
Type I Diabetes Web Application Presentation

Team Google Chrome

Software Architecture


Front end: Bootstrap templates/html
Queried results populated in tables and figures onto web app user interface.
Manhattan Plot
-Utilized **pandas** to create query into dataframe.
-**Numpy** in formatting dataframe
-**Seaborn** package + **Matplotlib** utilized for plot visualization
SQL syntax within flask:
Connecting to database and querying results from database tables.
LDLINK API
-Used to generate LD association test based on user input/queried results.
Database - **SQLite**
DB Browser- Database visualization
package: **sqlite3**

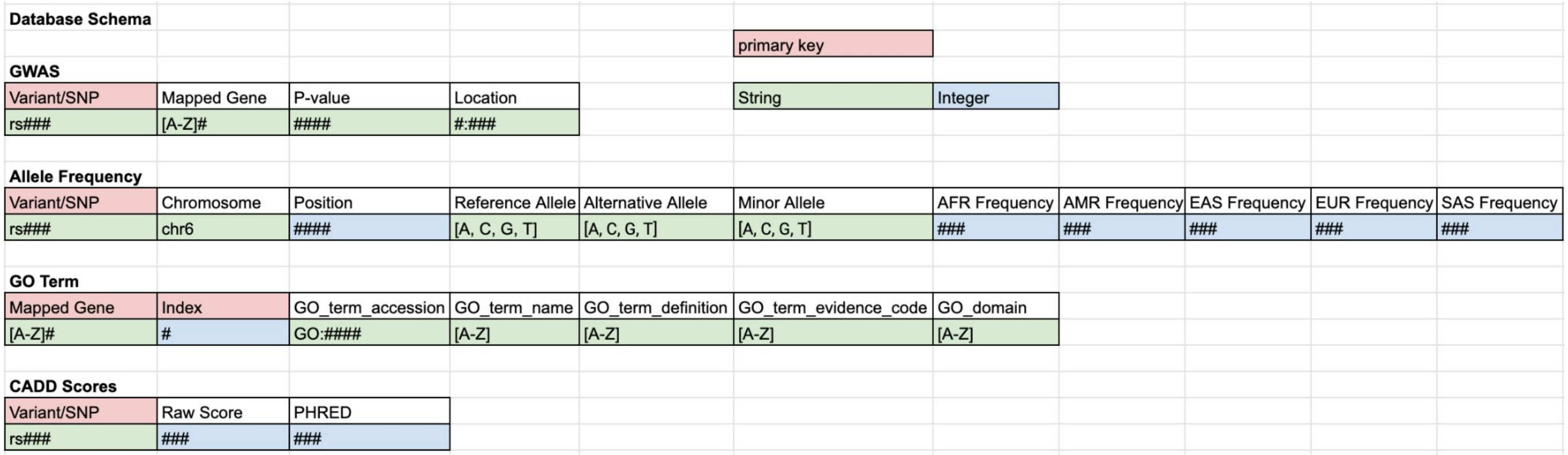
Data Sources

❑ GWAS data EMBL-EBI

❑ Gene Ontology data Ensembl

❑ Population Allele Frequencies and CADD data SNPnexus

Database Schema



Database Schema

primary key	
String	Integer

GWAS

Variant/SNP	Mapped Gene	P-value	Location
rs###	[A-Z]#	####	#:###

Allele Frequency

Variant/SNP	Chromosome	Position	Reference Allele	Alternative Allele	Minor Allele	AFR Frequency	AMR Frequency	EAS Frequency	EUR Frequency	SAS Frequency
rs###	chr6	####	[A, C, G, T]	[A, C, G, T]	[A, C, G, T]	###	###	###	###	###

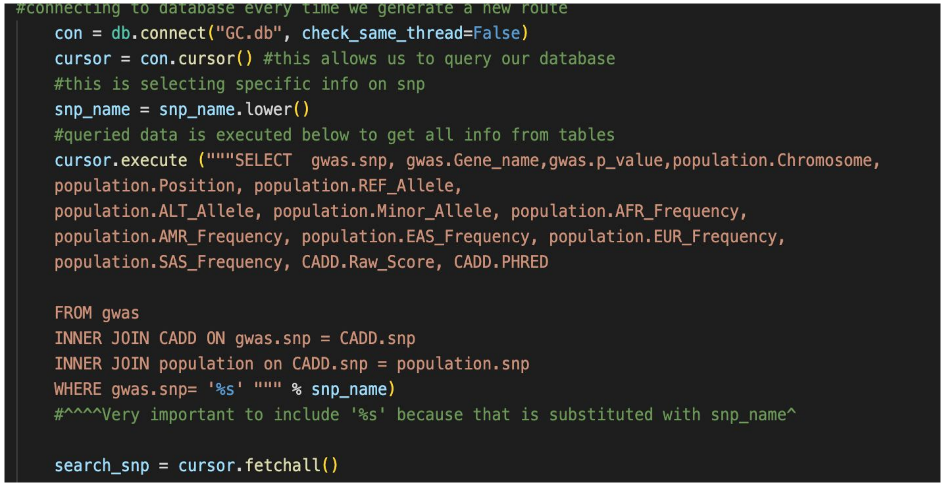
GO Term

Mapped Gene	Index	GO_term_accession	GO_term_name	GO_term_definition	GO_term_evidence_code	GO_domain
[A-Z]#	#	GO:####	[A-Z]	[A-Z]	[A-Z]	[A-Z]

CADD Scores

Variant/SNP	Raw Score	PHRED
rs###	###	###

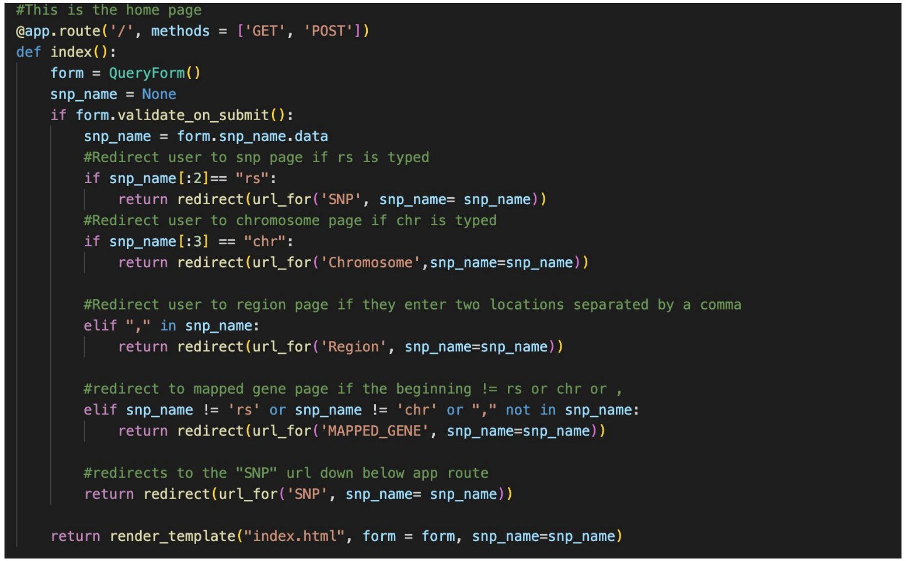
Connecting the Database with Flask



```
#connecting to database every time we generate a new route
    con = db.connect("GC.db", check_same_thread=False)
    cursor = con.cursor() #this allows us to query our database
    #this is selecting specific info on snp
    snp_name = snp_name.lower()
    #queried data is executed below to get all info from tables
    cursor.execute ("""SELECT  gwas.snp, gwas.Gene_name,gwas.p_value,population.Chromosome,
    population.Position, population.REF_Allele,
    population.ALT_Allele, population.Minor_Allele, population.AFR_Frequency,
    population.AMR_Frequency, population.EAS_Frequency, population.EUR_Frequency,
    population.SAS_Frequency, CADD.Raw_Score, CADD.PHRED

    FROM gwas
    INNER JOIN CADD ON gwas.snp = CADD.snp
    INNER JOIN population on CADD.snp = population.snp
    WHERE gwas.snp= '%s' """ % snp_name)
    #^^^^Very important to include '%s' because that is substituted with snp_name^

    search_snp = cursor.fetchall()
```

1. Connection
2. Cursor for Query
3. Execution of Query
   a. ‘SELECT’ - Columns
   b. ‘INNER JOIN’ - Selects tables with matching values
   c. WHERE - filters the results based on user input.

Flask redirecting/routing

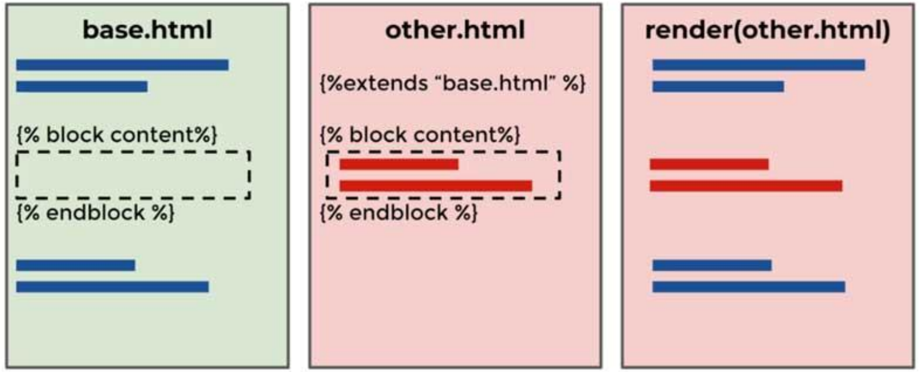


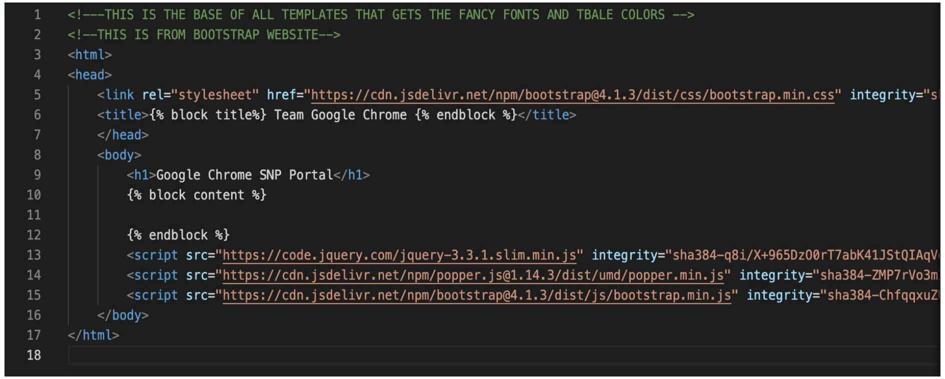
```python
#This is the home page
@app.route('/', methods = ['GET', 'POST'])
def index():
    form = QueryForm()
    snp_name = None
    if form.validate_on_submit():
        snp_name = form.snp_name.data
        #Redirect user to snp page if rs is typed
        if snp_name[:2]== "rs":
            return redirect(url_for('SNP', snp_name= snp_name))
        #Redirect user to chromosome page if chr is typed
        if snp_name[:3] == "chr":
            return redirect(url_for('Chromosome',snp_name=snp_name))

        #Redirect user to region page if they enter two locations separated by a comma
        elif "," in snp_name:
            return redirect(url_for('Region', snp_name=snp_name))

        #redirect to mapped gene page if the beginning != rs or chr or ,
        elif snp_name != 'rs' or snp_name != 'chr' or "," not in snp_name:
            return redirect(url_for('MAPPED_GENE', snp_name=snp_name))

        #redirects to the "SNP" url down below app route
        return redirect(url_for('SNP', snp_name= snp_name))

    return render_template("index.html", form = form, snp_name=snp_name)
```

Flask
web development,
one drop at a time

- Redirect to route based on user input
- Mapped genes unique so if user input does not = to any statement above then redirect to ‘MAPPED_GENE’
- Comma included to separate two positions.

HTML template inheritance

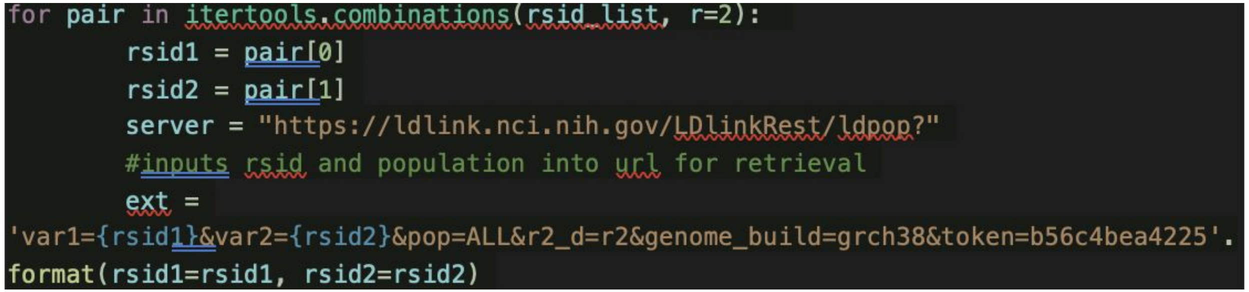




```
<!---THIS IS THE BASE OF ALL TEMPLATES THAT GETS THE FANCY FONTS AND TBALE COLORS -->
<!--THIS IS FROM BOOTSTRAP WEBSITE-->
<html>
<head>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css" integrity="s
    <title>{% block title%} Team Google Chrome {% endblock %}</title>
    </head>
    <body>
        <h1>Google Chrome SNP Portal</h1>
        {% block content %}

        {% endblock %}
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqV
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.js" integrity="sha384-ZMP7rVo3m
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min.js" integrity="sha384-ChfqqxuZ
    </body>
</html>
```

Limitations:
Limited to only just chromosome 6 information
Future development:
Allow clinician to filter chromosome at top of tables.
Condensed tables for gene ontology terms.

Linkage Disequilibrium - retrieving data



```
for pair in itertools.combinations(rsid_list, r=2):
        rsid1 = pair[0]
        rsid2 = pair[1]
        server = "https://ldlink.nci.nih.gov/LDlinkRest/ldpop?"
        #inputs rsid and population into url for retrieval
        ext =
'var1={rsid1}&var2={rsid2}&pop=ALL&r2_d=r2&genome_build=grch38&token=b56c4bea4225'.
format(rsid1=rsid1, rsid2=rsid2)
```

LD - Pros and Cons of LDlink API

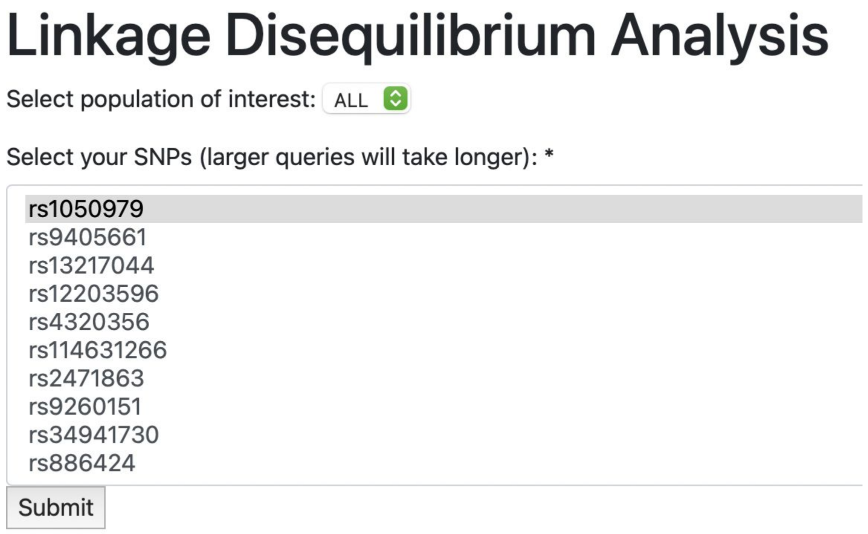
Pros

- Apply to whole genome easily

Cons

- Efficiency - long queries take time
- API blocking
- Token-based access

Linkage Disequilibrium - User queries



Linkage Disequilibrium Analysis

Select population of interest: ALL

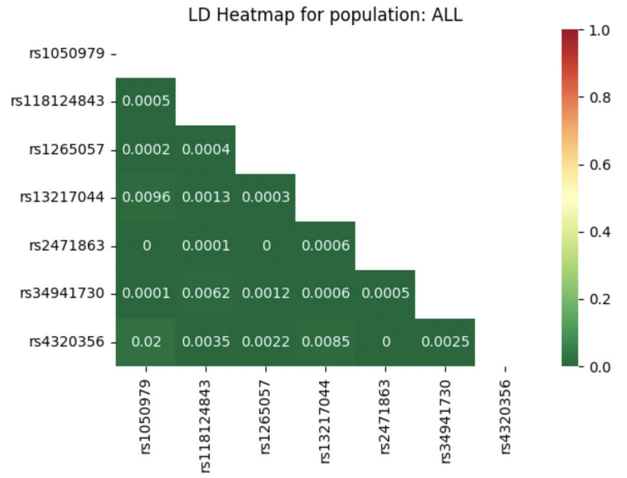
Select your SNPs (larger queries will take longer): *

rs1050979
rs9405661
rs13217044
rs12203596
rs4320356
rs114631266
rs2471863
rs9260151
rs34941730
rs886424

Linkage Disequilibrium - displaying and downloading

SNP Pair	r^2 Value
('rs1050979', 'rs13217044')	0.0096
('rs1050979', 'rs4320356')	0.0197
('rs1050979', 'rs2471863')	0.0
('rs1050979', 'rs34941730')	0.0001
('rs1050979', 'rs118124843')	0.0005
('rs1050979', 'rs1265057')	0.0002
('rs13217044', 'rs4320356')	0.0085
('rs13217044', 'rs2471863')	0.0006
('rs13217044', 'rs34941730')	0.0006
('rs13217044', 'rs118124843')	0.0013

Heatmap of LD values



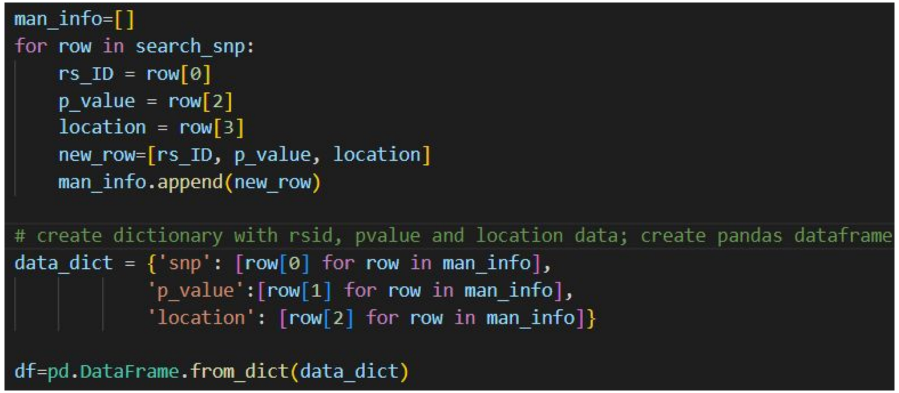
seaborn

Download text file here

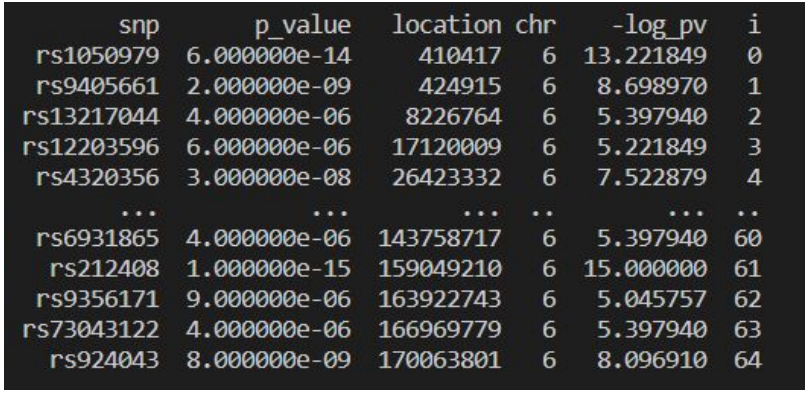
Click here to download a text file of the LD values in a matrix

Manhattan Plot- Retrieving Data

Creation of dataframe from SQL Query



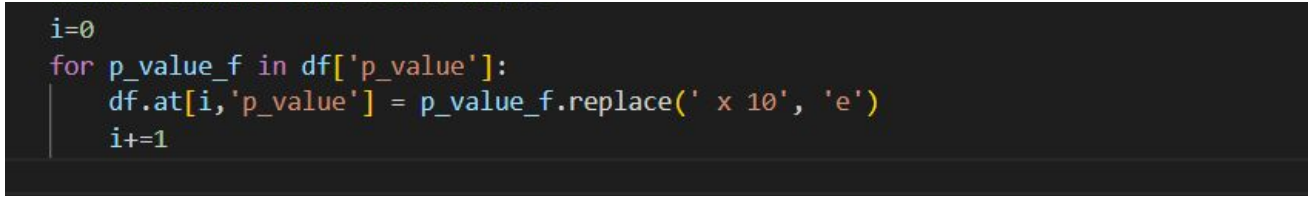
```
man_info=[]
for row in search_snp:
    rs_ID = row[0]
    p_value = row[2]
    location = row[3]
    new_row=[rs_ID, p_value, location]
    man_info.append(new_row)

# create dictionary with rsid, pvalue and location data; create pandas dataframe
data_dict = {'snp': [row[0] for row in man_info],
            'p_value':[row[1] for row in man_info],
            'location': [row[2] for row in man_info]}

df=pd.DataFrame.from_dict(data_dict)
```



```
         snp       p_value   location chr     -log_pv   i
   rs1050979  6.000000e-14     410417   6   13.221849   0
   rs9405661  2.000000e-09     424915   6    8.698970   1
  rs13217044  4.000000e-06    8226764   6    5.397940   2
  rs12203596  6.000000e-06   17120009   6    5.221849   3
   rs4320356  3.000000e-08   26423332   6    7.522879   4
         ...           ...        ...  ..         ...  ..
   rs6931865  4.000000e-06  143758717   6    5.397940  60
    rs212408  1.000000e-15  159049210   6   15.000000  61
   rs9356171  9.000000e-06  163922743   6    5.045757  62
  rs73043122  4.000000e-06  166969779   6    5.397940  63
    rs924043  8.000000e-09  170063801   6    8.096910  64
```

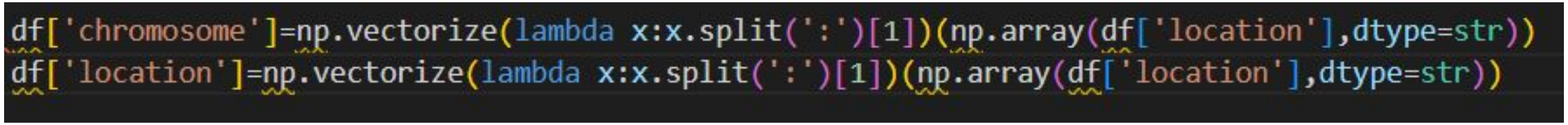
Manhattan Plot- Formatting

To write P-values in E- notation



```
i=0
for p_value_f in df['p_value']:
    df.at[i,'p_value'] = p_value_f.replace(' x 10', 'e')
    i+=1
```

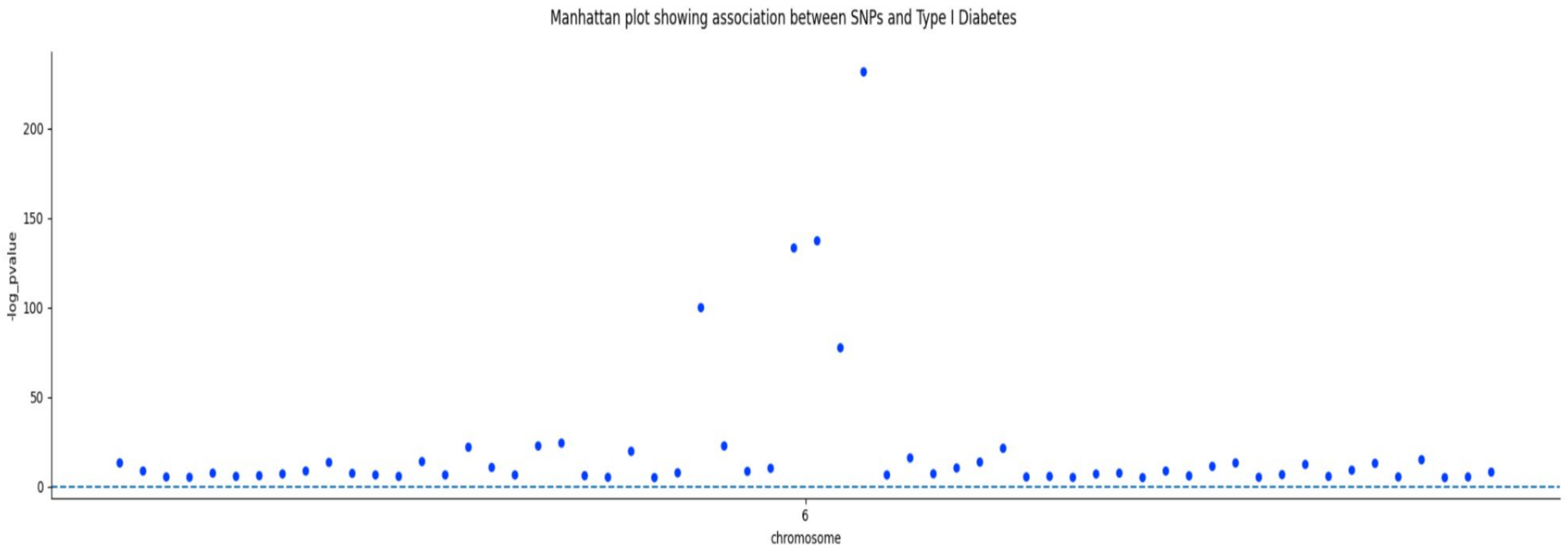
To separate the original location column into two: Chromosome number and location



```
df['chromosome']=np.vectorize(lambda x:x.split(':')[1])(np.array(df['location'],dtype=str))
df['location']=np.vectorize(lambda x:x.split(':')[1])(np.array(df['location'],dtype=str))
```

seaborn

Manhattan Plot



Limitations

- Expand data to other chromosomes
- Make the tables easier to read and filterable
- Make the plots interactive and more informative(e.g. Annotation)
- Improve LD retrieval method

Demonstration

Any questions?