In [1]:
```
import pandas as pd
```

In [3]:
```
# Read the Excel workbook
df = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls')
```

In [4]:
```
df.head()
```

Out[4]:

	Food Environment Atlas data download
0	Notes about the Food Environment Atlas downloa...
1	This file contains multiple spreadsheets:
2	1. A variable list that includes metadata abo...
3	2. Spreadsheets that contain data for each of...
4	3. County and State-level supplemental data t...

In [5]:
```
# Load Access, Stores, Assistance, Insecurity, Local, Health, Restaurants, Socioeconomic data sheets as dataframes

access = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'ACCESS')
stores = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'STORES')
assistance = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'ASSISTANCE')
insecurity = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'INSECURITY')
local = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'LOCAL')
health = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'HEALTH')
restaurants = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'RESTAURANTS')
socioeconomic = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'SOCIOECONOMIC')
```

In [6]:
```
# prices = pd.read_excel('data/2018-usda-food-environment-atlas-dataset.xls', 'FOOD')
```

In [7]:
```
restaurants.head()
restaurants_MA = restaurants[restaurants['State'] == 'MA']
restaurant_cols = ['FIPS', 'State', 'County', 'FFR14', 'FFRPTH14', 'FSR14', 'FSRPT
H14']
restaurants_MA = restaurants_MA[restaurant_cols]
restaurants_MA
```

Out[7]:

	FIPS	State	County	FFR14	FFRPTH14	FSR14	FSRPTH14
1217	25001	MA	Barnstable	216	1.005053	427	1.986841
1218	25003	MA	Berkshire	118	0.916754	215	1.670357
1219	25005	MA	Bristol	386	0.696507	468	0.844470
1220	25007	MA	Dukes	24	1.382807	60	3.457018
1221	25009	MA	Essex	576	0.748936	621	0.807447
1222	25011	MA	Franklin	35	0.493918	65	0.917276
1223	25013	MA	Hampden	293	0.625853	335	0.715566
1224	25015	MA	Hampshire	99	0.615140	168	1.043874
1225	25017	MA	Middlesex	1213	0.772456	1303	0.829770
1226	25019	MA	Nantucket	17	1.565954	54	4.974208
1227	25021	MA	Norfolk	471	0.680386	601	0.868178
1228	25023	MA	Plymouth	306	0.603524	410	0.808643
1229	25025	MA	Suffolk	745	0.970995	918	1.196475
1230	25027	MA	Worcester	581	0.714220	574	0.705615

In [8]:
```
stores_MA = stores[stores['State'] == 'MA']
stores_cols = ['FIPS', 'State', 'County', 'GROCPTH14', 'SUPERCPTH14', 'CONVSPTH14'
, 'SPECSPTH14', 'SNAPSPTH16']
stores_MA = stores_MA[stores_cols]
stores_MA
```

Out[8]:

	FIPS	State	County	GROCPTH14	SUPERCPTH14	CONVSPTH14	SPECSPTH14	SNAPSPTH16
1217	25001	MA	Barnstable	0.348977	0.004653	0.581628	0.251263	0.833038
1218	25003	MA	Berkshire	0.264149	0.007769	0.536068	0.100998	0.742037
1219	25005	MA	Bristol	0.187660	0.012631	0.481781	0.075786	0.881209
1220	25007	MA	Dukes	0.691404	0.000000	0.460936	0.633787	0.497700
1221	25009	MA	Essex	0.243144	0.003901	0.357565	0.100118	0.696817
1222	25011	MA	Franklin	0.282239	0.014112	0.493918	0.098784	0.736457
1223	25013	MA	Hampden	0.209330	0.008544	0.508372	0.061944	1.020705
1224	25015	MA	Hampshire	0.229901	0.006214	0.316890	0.074562	0.606141
1225	25017	MA	Middlesex	0.182766	0.005731	0.362348	0.083423	0.522978
1226	25019	MA	Nantucket	0.368460	0.000000	0.552690	0.552690	0.363372
1227	25021	MA	Norfolk	0.171902	0.010112	0.365473	0.082340	0.489710
1228	25023	MA	Plymouth	0.167646	0.005917	0.455602	0.102560	0.618552
1229	25025	MA	Suffolk	0.282827	0.001303	0.431409	0.099055	0.864011
1230	25027	MA	Worcester	0.167184	0.009834	0.413043	0.057777	0.742344

In [ ]:
```
```

In [9]:
```
# We're going to join all these sheets, so we'll drop redundant state and county c
ols from all but one.
# We'll also set the FIPS ID col to be index so we can use that for the join.
dfs = [stores, assistance, insecurity, local, health, restaurants, socioeconomic]
for df in dfs:
    df.drop(columns=['State', 'County'], axis=1, inplace=True)
    df.set_index('FIPS', inplace=True)

# Then, we'll also set the index on the Access df. This will be the dataframe we j
oin the others onto.
access.set_index('FIPS', inplace=True)
```

In [10]:
```
# Combine all sheets into one dataframe by joining on FIPS col.
# Now we have a master dataframe containing the cols from all sheets.
master_df = access.join(dfs)
```

In [11]:
```
# These are the features I was interested in. Feel free to change them aroud as yo
u see fit.
# The sheeet 'Variable List' contains descriptions of all available vars.
cols_of_interest = ['PCT_LACCESS_POP15', 'GROCPTH14', 'CONVSPTH14', 'SNAPSPTH12',
'WICSPTH12', 'FFRPTH14',
                    'FOODINSEC_10_12', 'FMRKTPTH16', 'CSA12', 'FARM_TO_SCHOOL09',
'PCT_DIABETES_ADULTS13',
                    'PCT_OBESE_ADULTS13', 'POVRATE15', 'MEDHHINC15', 'RECFACPTH14'
]

# Create a dataframe with just the cols of interest.
df = master_df[cols_of_interest]
df.shape
```

Out[11]:
```
(3143, 15)
```

In [12]:
```
# Check for null values...
df.isnull().sum()
```

Out[12]:
```
PCT_LACCESS_POP15        19
GROCPTH14                 0
CONVSPTH14                0
SNAPSPTH12                0
WICSPTH12                 0
FFRPTH14                  0
FOODINSEC_10_12           0
FMRKTPTH16                2
CSA12                    63
FARM_TO_SCHOOL09          5
PCT_DIABETES_ADULTS13     1
PCT_OBESE_ADULTS13        1
POVRATE15                 4
MEDHHINC15                4
RECFACPTH14               0
dtype: int64
```

In [13]:
```
# ...and drop them
df = df.dropna()
df.isnull().sum()
```

Out[13]:
```
PCT_LACCESS_POP15        0
GROCPTH14                0
CONVSPTH14               0
SNAPSPTH12               0
WICSPTH12                0
FFRPTH14                 0
FOODINSEC_10_12          0
FMRKTPTH16               0
CSA12                    0
FARM_TO_SCHOOL09         0
PCT_DIABETES_ADULTS13    0
PCT_OBESE_ADULTS13       0
POVRATE15                0
MEDHHINC15               0
RECFACPTH14              0
dtype: int64
```

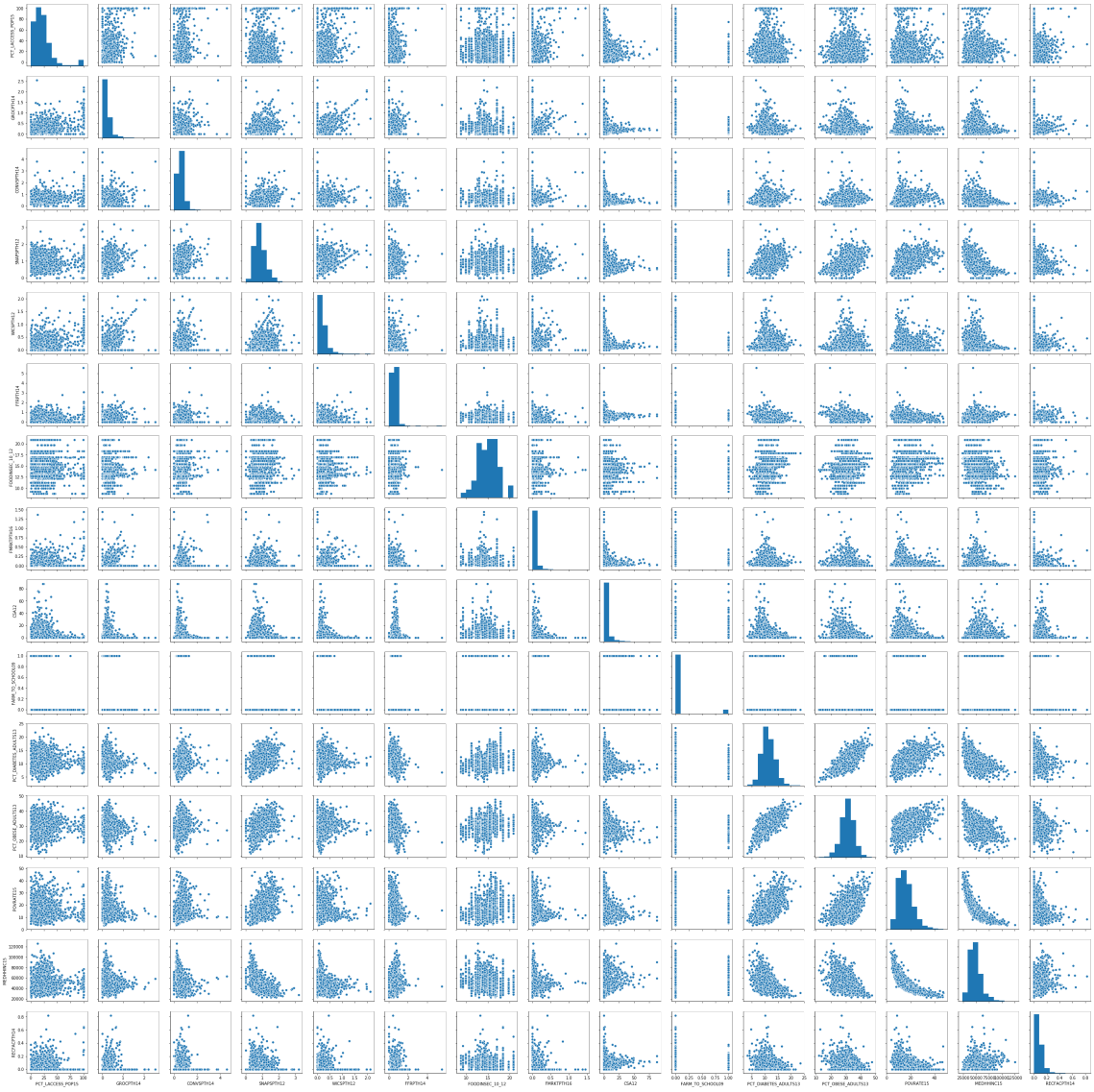
In [14]:
```
# dtypes look good
print(df.dtypes)
```

```
PCT_LACCESS_POP15        float64
GROCPTH14                float64
CONVSPTH14               float64
SNAPSPTH12               float64
WICSPTH12                float64
FFRPTH14                 float64
FOODINSEC_10_12          float64
FMRKTPTH16               float64
CSA12                    float64
FARM_TO_SCHOOL09         float64
PCT_DIABETES_ADULTS13    float64
PCT_OBESE_ADULTS13       float64
POVRATE15                float64
MEDHHINC15               float64
RECFACPTH14              float64
dtype: object
```

In [15]:
```
import seaborn as sns
```

In [16]:
```
# We can look at scatter plots of each feature in pairs, to see if there is a corr
elation between them.
# You can double-click on a plot to enlarge.
# We may need to back-track and look at some other variables.
# This is the part where it would be good to get some input from the TAs.
sns.pairplot(df)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x1182faf98>
```



In [17]:
```
from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

In [18]:
```
features = ['PCT_LACCESS_POP15', 'GROCPTH14', 'CONVSPTH14', 'SNAPSPTH12', 'WICSPTH
12', 'FFRPTH14',
            'FOODINSEC_10_12', 'FMRKTPTH16', 'CSA12', 'FARM_TO_SCHOOL09', 'POVRATE
15', 'MEDHHINC15',
            'PCT_DIABETES_ADULTS13', 'RECFACPTH14']
target = ['PCT_OBESE_ADULTS13']
```

In [19]:
```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test
_size=0.33)

# Create linear regression object
regr = linear_model.LinearRegression()

# Train our model
regr.fit(X_train, y_train)

# Make predictions
y_pred = regr.predict(X_test)

# R2 scores
print(r2_score(y_train, regr.predict(X_train)))
print(r2_score(y_test, y_pred))
```

```
0.5472865099050459
0.5013684111771843

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sk
learn/linear_model/base.py:509: RuntimeWarning: internal gelsd driver lwork quer
y error, required iwork dimension not returned. This is likely the result of LAP
ACK bug 0038, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling back to 'g
elss' driver.
  linalg.lstsq(X, y)
```

In [ ]:
```
```