# Assignment 1 - Linear Regression


## Prajwal Sahu || ME18B114


## Importing Libraries

We'll be using common python libraries like numpy, pandas, seaborn etc.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

## Importing the Data-sets

We have been provided with cancer incidence and death rate for US counties. Each county has a unique FIPS (Federal Information Processing Standards) Code, and the data is sorted by it.

### Data Notes

`FIPS` - Federal Information Processing Standard. Each area has a unique code.

`Incidence_rates` - Cases per 100,000 population per year which are age-adjusted to the 2000 US standard population.

`Mortality Rate` - The measure of number of deaths.

`All_Poverty` - Population For whom income in the past 12 months is below poverty level.

`M_Poverty` - Male Population For whom income in the past 12 months is below poverty level.

`F_Poverty` - Female Population For whom income in the past 12 months is below poverty level.

`Med_Income` - Median household income in the past 12 months

`Med_Income_X` - Median household income in the past 12 months for the X ethnicity ( X - White, Black, Asian, Hispanic etc.).

`All_With` - Population covered by health insurance.

`All_Without` - Population not covered by health insurance.

`M_With, M_Without` - Male population covered and not covered by health insurance respectively.

`F_With, F_Without` - Female population covered and not covered by health insurance respectively.

`Avg_Ann_Deaths` - Average lung cancer mortalities

`Avg_Ann_Incidence` - Average lung cancer incidence rate

# Our Data Set

We have a merged data set containing Income, Poverty and Health Insurance data as well as the Mortality rate in the various counties with FIPS.

In [3]:
```
mergeddf = pd.read_excel('merged_data.xlsx')
```

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

Out[4]:

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | Med_Incoı |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | AK | Aleutians East Borough, Alaska | 553 | 334 | 219 | 2013 | 61518.0 | |
| **1** | 1 | AK | Aleutians West Census Area, Alaska | 499 | 273 | 226 | 2016 | 84306.0 | |
| **2** | 2 | AK | Anchorage Municipality, Alaska | 23914 | 10698 | 13216 | 2020 | 78326.0 | |
| **3** | 3 | AK | Bethel Census Area, Alaska | 4364 | 2199 | 2165 | 2050 | 51012.0 | |
| **4** | 4 | AK | Bristol Bay Borough, Alaska | 69 | 33 | 36 | 2060 | 79750.0 | |

5 rows × 26 columns

# Population data

Acquired a data-set from data.world for population estimate in counties in 2015, the year our dataset was collected.

In [5]:
```
popdf = pd.read_csv('popdata.csv')
popdf.head()
```

Out[5]:

| | SUMLEV | REGION | DIVISION | STATE | COUNTY | STNAME | CTYNAME | POPESTIMATE2015 |
|---|---|---|---|---|---|---|---|---|
| **0** | 40 | 3 | 6 | 1 | 0 | Alabama | Alabama | 4858979 |
| **1** | 50 | 3 | 6 | 1 | 1 | Alabama | Autauga County | 55347 |

| | SUMLEV | REGION | DIVISION | STATE | COUNTY | STNAME | CTYNAME | POPESTIMATE2015 |
|---|---|---|---|---|---|---|---|---|
| **2** | 50 | 3 | 6 | 1 | 3 | Alabama | Baldwin County | 203709 |
| **3** | 50 | 3 | 6 | 1 | 5 | Alabama | Barbour County | 26489 |
| **4** | 50 | 3 | 6 | 1 | 7 | Alabama | Bibb County | 22583 |

In [6]:

```
#adding zeroes to adjust for our FIPS

state = popdf.STATE.apply(lambda x: str(x))\

county = popdf.COUNTY.apply(lambda x: str(x))\
                            .str.pad(3, 'left', '0')
```

In [7]:

```
state
```

Out[7]:

```
0       1
1       1
2       1
3       1
4       1
       ..
3188    56
3189    56
3190    56
3191    56
3192    56
Name: STATE, Length: 3193, dtype: object
```

In [8]:

```
county
```

Out[8]:

```
0       000
1       001
2       003
3       005
4       007
       ...
3188    037
3189    039
3190    041
3191    043
3192    045
Name: COUNTY, Length: 3193, dtype: object
```

In [9]:

```
# Creating a new column for merging
popdf['FIPS']= state + county
popdf
```

Out[9]:

| | SUMLEV | REGION | DIVISION | STATE | COUNTY | STNAME | CTYNAME | POPESTIMATE2015 | F |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 40 | 3 | 6 | 1 | 0 | Alabama | Alabama | 4858979 | 1( |
| **1** | 50 | 3 | 6 | 1 | 1 | Alabama | Autauga County | 55347 | 1( |
| **2** | 50 | 3 | 6 | 1 | 3 | Alabama | Baldwin County | 203709 | 1( |
| **3** | 50 | 3 | 6 | 1 | 5 | Alabama | Barbour County | 26489 | 1( |

| | SUMLEV | REGION | DIVISION | STATE | COUNTY | STNAME | CTYNAME | POPESTIMATE2015 | F |
|---|---|---|---|---|---|---|---|---|---|
| **4** | 50 | 3 | 6 | 1 | 7 | Alabama | Bibb County | 22583 | 1( |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3188** | 50 | 4 | 8 | 56 | 37 | Wyoming | Sweetwater County | 44626 | 56( |
| **3189** | 50 | 4 | 8 | 56 | 39 | Wyoming | Teton County | 23125 | 56( |
| **3190** | 50 | 4 | 8 | 56 | 41 | Wyoming | Uinta County | 20822 | 56( |
| **3191** | 50 | 4 | 8 | 56 | 43 | Wyoming | Washakie County | 8328 | 56( |
| **3192** | 50 | 4 | 8 | 56 | 45 | Wyoming | Weston County | 7234 | 56( |

3193 rows × 9 columns

# Adding Population Estimates to Socio-Economic Data

We merge the 2 dataframes on the FIPS column

In [10]:

```
popdf['FIPS']=popdf['FIPS'].astype(int)
```

In [11]:

```
# Checking whether both dfs match

print(sum(pd.Series(mergeddf.FIPS.unique()).isin(popdf.FIPS)), 'matches out of')
print("%d unique values" % len(mergeddf.FIPS.unique()))
```

```
3134 matches out of
3134 unique values
```

In [12]:

```
# merging

mergeddf = mergeddf.merge(popdf ,right_on = 'FIPS',left_on = 'FIPS')
mergeddf
```

Out[12]:

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | Med_ |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | AK | Aleutians East Borough, Alaska | 553 | 334 | 219 | 2013 | 61518.0 | |
| **1** | 1 | AK | Aleutians West Census Area, Alaska | 499 | 273 | 226 | 2016 | 84306.0 | |
| **2** | 2 | AK | Anchorage Municipality, Alaska | 23914 | 10698 | 13216 | 2020 | 78326.0 | |

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | Med_ |
|---|---|---|---|---|---|---|---|---|---|
| **3** | 3 | AK | Bethel Census Area, Alaska | 4364 | 2199 | 2165 | 2050 | 51012.0 | |
| **4** | 4 | AK | Bristol Bay Borough, Alaska | 69 | 33 | 36 | 2060 | 79750.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3129** | 3129 | WY | Sweetwater County, Wyoming | 5058 | 2177 | 2881 | 56037 | 69022.0 | |
| **3130** | 3130 | WY | Teton County, Wyoming | 1638 | 1026 | 612 | 56039 | 75325.0 | |
| **3131** | 3131 | WY | Uinta County, Wyoming | 2845 | 1453 | 1392 | 56041 | 56569.0 | |
| **3132** | 3132 | WY | Washakie County, Wyoming | 1137 | 489 | 648 | 56043 | 47652.0 | |
| **3133** | 3133 | WY | Weston County, Wyoming | 958 | 354 | 604 | 56045 | 57738.0 | |

3134 rows × 34 columns

In [13]:

```
# dropping extra columns
mergeddf.drop(['SUMLEV','STATE','REGION','DIVISION','COUNTY','STNAME','CTYNAME'], ax
```

# Data Preprocessing

## Checking for Null Values in our dataframe

In [14]:

```
mergeddf
```

Out[14]:

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | Med_ |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | AK | Aleutians East Borough, Alaska | 553 | 334 | 219 | 2013 | 61518.0 | |
| **1** | 1 | AK | Aleutians West Census Area, Alaska | 499 | 273 | 226 | 2016 | 84306.0 | |

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | Med_ |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 2 | AK | Anchorage Municipality, Alaska | 23914 | 10698 | 13216 | 2020 | 78326.0 | |
| **3** | 3 | AK | Bethel Census Area, Alaska | 4364 | 2199 | 2165 | 2050 | 51012.0 | |
| **4** | 4 | AK | Bristol Bay Borough, Alaska | 69 | 33 | 36 | 2060 | 79750.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3129** | 3129 | WY | Sweetwater County, Wyoming | 5058 | 2177 | 2881 | 56037 | 69022.0 | |
| **3130** | 3130 | WY | Teton County, Wyoming | 1638 | 1026 | 612 | 56039 | 75325.0 | |
| **3131** | 3131 | WY | Uinta County, Wyoming | 2845 | 1453 | 1392 | 56041 | 56569.0 | |
| **3132** | 3132 | WY | Washakie County, Wyoming | 1137 | 489 | 648 | 56043 | 47652.0 | |
| **3133** | 3133 | WY | Weston County, Wyoming | 958 | 354 | 604 | 56045 | 57738.0 | |

3134 rows × 27 columns

In [15]:

```
# missing value count

for col in mergeddf.columns:
    print((col, sum(mergeddf[col].isnull())))
```

```
('Unnamed: 0', 0)
('State', 0)
('AreaName', 0)
('All_Poverty', 0)
('M_Poverty', 0)
('F_Poverty', 0)
('FIPS', 0)
('Med_Income', 1)
('Med_Income_White', 2)
('Med_Income_Black', 1210)
('Med_Income_Nat_Am', 1660)
('Med_Income_Asian', 1757)
('Hispanic', 681)
('M_With', 0)
('M_Without', 0)
('F_With', 0)
('F_Without', 0)
('All_With', 0)
('All_Without', 0)
('fips_x', 0)
('Incidence_Rate', 0)
```

```
('Avg_Ann_Incidence', 0)
('recent_trend', 0)
('fips_y', 0)
('Mortality_Rate', 0)
('Avg_Ann_Deaths', 0)
('POPESTIMATE2015', 0)
```

As a significant amount of data is missing, we should drop the income ethnicity data.

In [16]:

```
mergeddf.drop(['Med_Income_White', 'Med_Income_Black', 'Med_Income_Nat_Am',
             'Med_Income_Asian', 'Hispanic'], axis=1, inplace=True)
```

# Dealing with Strings

We notice some strings in the Data. Let's find out the data types

In [17]:

```
def get_types(col_name):
    ts = (pd.Series([type(i) for i in mergeddf[col_name]]).value_counts())
    print("%s\n" % feature, ts, "\n", "-"*20)

for feature in mergeddf.columns:
    get_types(feature)
```

```
Unnamed: 0
 <class 'int'>    3134
dtype: int64
 --------------------
State
 <class 'str'>    3134
dtype: int64
 --------------------
AreaName
 <class 'str'>    3134
dtype: int64
 --------------------
All_Poverty
 <class 'int'>    3134
dtype: int64
 --------------------
M_Poverty
 <class 'int'>    3134
dtype: int64
 --------------------
F_Poverty
 <class 'int'>    3134
dtype: int64
 --------------------
FIPS
 <class 'int'>    3134
dtype: int64
 --------------------
Med_Income
 <class 'float'>    3134
dtype: int64
 --------------------
M_With
 <class 'int'>    3134
dtype: int64
 --------------------
M_Without
 <class 'int'>    3134
dtype: int64
 --------------------
F_With
 <class 'int'>    3134
```

```
dtype: int64
 --------------------
F_Without
 <class 'int'>    3134
dtype: int64
 --------------------
All_With
 <class 'int'>    3134
dtype: int64
 --------------------
All_Without
 <class 'int'>    3134
dtype: int64
 --------------------
fips_x
 <class 'int'>    3134
dtype: int64
 --------------------
Incidence_Rate
 <class 'float'>    2361
<class 'str'>       499
<class 'int'>       274
dtype: int64
 --------------------
Avg_Ann_Incidence
 <class 'int'>    2714
<class 'str'>     420
dtype: int64
 --------------------
recent_trend
 <class 'str'>    3134
dtype: int64
 --------------------
fips_y
 <class 'int'>    3134
dtype: int64
 --------------------
Mortality_Rate
 <class 'float'>    2539
<class 'str'>       325
<class 'int'>       270
dtype: int64
 --------------------
Avg_Ann_Deaths
 <class 'int'>    2809
<class 'str'>     325
dtype: int64
 --------------------
POPESTIMATE2015
 <class 'int'>    3134
dtype: int64
 --------------------
```

# Cleaning the Data Strings

## Mortality Rate

We need to deal with the columns containing the string *.

Our target variable is the Mortality Rate. Therefore the * values in Mortality rate should be dropped

In [18]:
```
mergeddf = mergeddf[mergeddf.Mortality_Rate != '*']
```

In [19]:
```
mergeddf.shape
```

Out[19]:
```
(2809, 22)
```

## Med_Income

In [20]:
```
mergeddf['Med_Income'] = pd.to_numeric(mergeddf.Med_Income)
```

## Incidence Rate

We first find out the various nan values

In [21]:
```
values = []
for _, j in enumerate(mergeddf.Incidence_Rate):
    try:
        pd.to_numeric(j)
    except:
        values.append(j)

pd.Series(values).value_counts()[:10]
```

Out[21]:
```
_          151
__          12
*            5
73.6 #       2
68.2 #       2
71.1 #       2
97 #         1
62.4 #       1
64.9 #       1
69.5 #       1
dtype: int64
```

We will replace string with na.

In [22]:
```
mergeddf['Incidence_Rate'] = pd.to_numeric(mergeddf.Incidence_Rate, errors='coerce')
```

We fill the empty data with the median.

In [23]:
```
mergeddf['Incidence_Rate'] = mergeddf.Incidence_Rate.fillna(mergeddf.Incidence_Rate.
print(sum(mergeddf.Incidence_Rate.isnull()))
```

```
0
```

## Avg_Ann_Incidence

Following similar procedure as done above in Incidence_Rate

In [24]:
```
values = []
for _, j in enumerate(mergeddf.Avg_Ann_Incidence):
    try:
        pd.to_numeric(j)
    except:
        values.append(j)

pd.Series(values,dtype = str).value_counts()[:10]
```

Out[24]:
```
_              151
__              12
3 or fewer       5
dtype: int64
```

In [25]:
```
mergeddf['Avg_Ann_Incidence'] = pd.to_numeric(mergeddf.Incidence_Rate, errors='coerc
mergeddf['Avg_Ann_Incidence'] = mergeddf.Incidence_Rate.fillna(mergeddf.Incidence_Ra
print(sum(mergeddf.Incidence_Rate.isnull()))
```

```
0
```

## Avg_Ann_Deaths

Following similar procedure as done above in Incidence_Rate

In [26]:
```
values = []
for _, k in enumerate(mergeddf.Avg_Ann_Deaths):
    try:
        pd.to_numeric(k)
    except:
        values.append(k)

pd.Series(values).value_counts()[:10]
```

Out[26]:
```
Series([], dtype: int64)
```

In [27]:
```
mergeddf['Avg_Ann_Deaths'] = pd.to_numeric(mergeddf.Incidence_Rate, errors='coerce')
mergeddf['Avg_Ann_Deaths'] = mergeddf.Incidence_Rate.fillna(mergeddf.Incidence_Rate.
print(sum(mergeddf.Incidence_Rate.isnull()))
```

```
0
```

# Dealing with Categorical Variables

We have the recent trend variable which we have to convert to numerical values using dummy variables.

In [28]:
```
mergeddf
```

Out[28]:

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | M_W |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 2 | AK | Anchorage Municipality, Alaska | 23914 | 10698 | 13216 | 2020 | 78326.0 | 1207 |
| **3** | 3 | AK | Bethel Census Area, Alaska | 4364 | 2199 | 2165 | 2050 | 51012.0 | 63 |
| **7** | 7 | AK | Fairbanks North Star Borough, Alaska | 7752 | 3523 | 4229 | 2090 | 71068.0 | 406 |
| **9** | 9 | AK | Juneau City and Borough, Alaska | 2110 | 1145 | 965 | 2110 | 85746.0 | 137 |

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | M_W |
|---|---|---|---|---|---|---|---|---|---|
| **10** | 10 | AK | Kenai Peninsula Borough, Alaska | 5558 | 2596 | 2962 | 2122 | 63684.0 | 223 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3129** | 3129 | WY | Sweetwater County, Wyoming | 5058 | 2177 | 2881 | 56037 | 69022.0 | 198 |
| **3130** | 3130 | WY | Teton County, Wyoming | 1638 | 1026 | 612 | 56039 | 75325.0 | 89 |
| **3131** | 3131 | WY | Uinta County, Wyoming | 2845 | 1453 | 1392 | 56041 | 56569.0 | 91 |
| **3132** | 3132 | WY | Washakie County, Wyoming | 1137 | 489 | 648 | 56043 | 47652.0 | 33 |
| **3133** | 3133 | WY | Weston County, Wyoming | 958 | 354 | 604 | 56045 | 57738.0 | 29 |

2809 rows × 22 columns

In [29]:
```
# getting value count

mergeddf['recent_trend'].value_counts()
```

Out[29]:
```
stable     2382
falling     197
_           151
rising       39
*            28
__           12
Name: recent_trend, dtype: int64
```

We need to replace the * values with stable.

In [30]:
```
mergeddf.replace({'recent_Trend' : {'*':'stable'}}, inplace=True)
```

Stable trends doesn't affect our model, so we create dummy variables for rising and falling

In [31]:
```
def f(x, term):
    if x == term:
        return 1
    else:
        return 0
```

In [32]:
```
mergeddf['rising'] = mergeddf.recent_trend.apply(lambda x: f(x, term='rising'))
```

In [33]:
```
mergeddf['falling'] = mergeddf.recent_trend.apply(lambda x: f(x, term='falling'))
```

In [34]:

```
mergeddf
```

Out[34]:

| | Unnamed: 0 | State | AreaName | All_Poverty | M_Poverty | F_Poverty | FIPS | Med_Income | M_W |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 2 | AK | Anchorage Municipality, Alaska | 23914 | 10698 | 13216 | 2020 | 78326.0 | 1207 |
| **3** | 3 | AK | Bethel Census Area, Alaska | 4364 | 2199 | 2165 | 2050 | 51012.0 | 63 |
| **7** | 7 | AK | Fairbanks North Star Borough, Alaska | 7752 | 3523 | 4229 | 2090 | 71068.0 | 406 |
| **9** | 9 | AK | Juneau City and Borough, Alaska | 2110 | 1145 | 965 | 2110 | 85746.0 | 137 |
| **10** | 10 | AK | Kenai Peninsula Borough, Alaska | 5558 | 2596 | 2962 | 2122 | 63684.0 | 223 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3129** | 3129 | WY | Sweetwater County, Wyoming | 5058 | 2177 | 2881 | 56037 | 69022.0 | 198 |
| **3130** | 3130 | WY | Teton County, Wyoming | 1638 | 1026 | 612 | 56039 | 75325.0 | 89 |
| **3131** | 3131 | WY | Uinta County, Wyoming | 2845 | 1453 | 1392 | 56041 | 56569.0 | 91 |
| **3132** | 3132 | WY | Washakie County, Wyoming | 1137 | 489 | 648 | 56043 | 47652.0 | 33 |
| **3133** | 3133 | WY | Weston County, Wyoming | 958 | 354 | 604 | 56045 | 57738.0 | 29 |

2809 rows × 24 columns

In [35]:

```
# original column no longer needed
mergeddf.drop(['recent_trend'],axis=1,inplace = True)
```

Data is cleaned.

# Dropping unnecessary columns

In [36]:
```
finaldf = mergeddf.drop(['Unnamed: 0','State','AreaName','FIPS','fips_x','fips_y'],a
```

In [37]:
```
finaldf.head()
```

Out[37]:

| | All_Poverty | M_Poverty | F_Poverty | Med_Income | M_With | M_Without | F_With | F_Without | All_V |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 23914 | 10698 | 13216 | 78326.0 | 120747 | 23245 | 122426 | 21393 | 243 |
| **3** | 4364 | 2199 | 2165 | 51012.0 | 6396 | 2708 | 6627 | 1774 | 13 |
| **7** | 7752 | 3523 | 4229 | 71068.0 | 40605 | 6957 | 40210 | 5322 | 80 |
| **9** | 2110 | 1145 | 965 | 85746.0 | 13739 | 2433 | 13582 | 2213 | 27 |
| **10** | 5558 | 2596 | 2962 | 63684.0 | 22391 | 6435 | 21668 | 5433 | 44 |

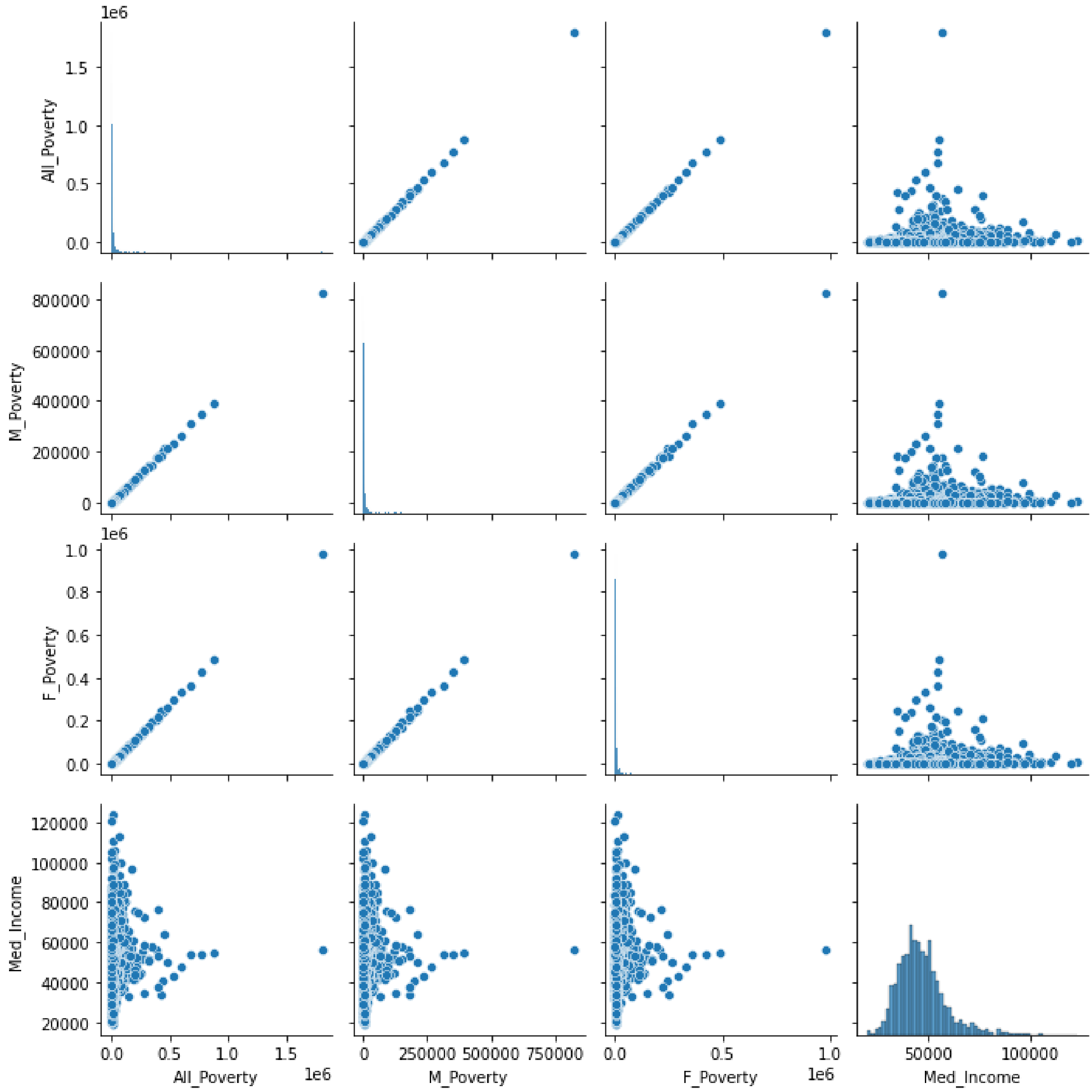
# Determining Correlation

We'll check correlation between similar variables to decide which ones to use for our model

## Poverty

In [38]:
```
sns.pairplot(finaldf[['All_Poverty','M_Poverty','F_Poverty','Med_Income']])
```

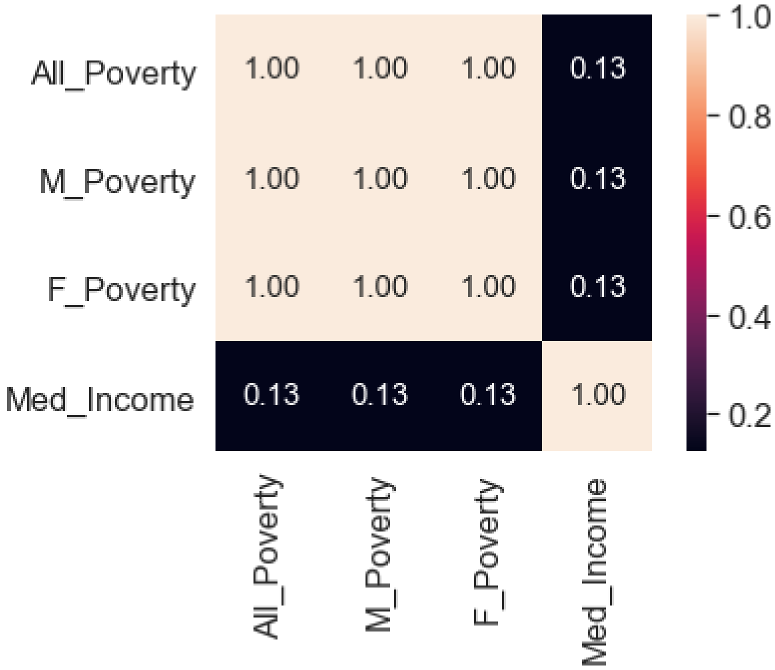
Out[38]:
```
<seaborn.axisgrid.PairGrid at 0x2c5ba0e5a30>
```




In [39]:

```
# heatmap for correlation

cols = ['All_Poverty', 'M_Poverty', 'F_Poverty', 'Med_Income']
cm = np.corrcoef(finaldf[['All_Poverty', 'M_Poverty', 'F_Poverty', 'Med_Income']].va
sns.set(font_scale=1.5)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size
                 yticklabels=cols, xticklabels=cols)
plt.show()
```



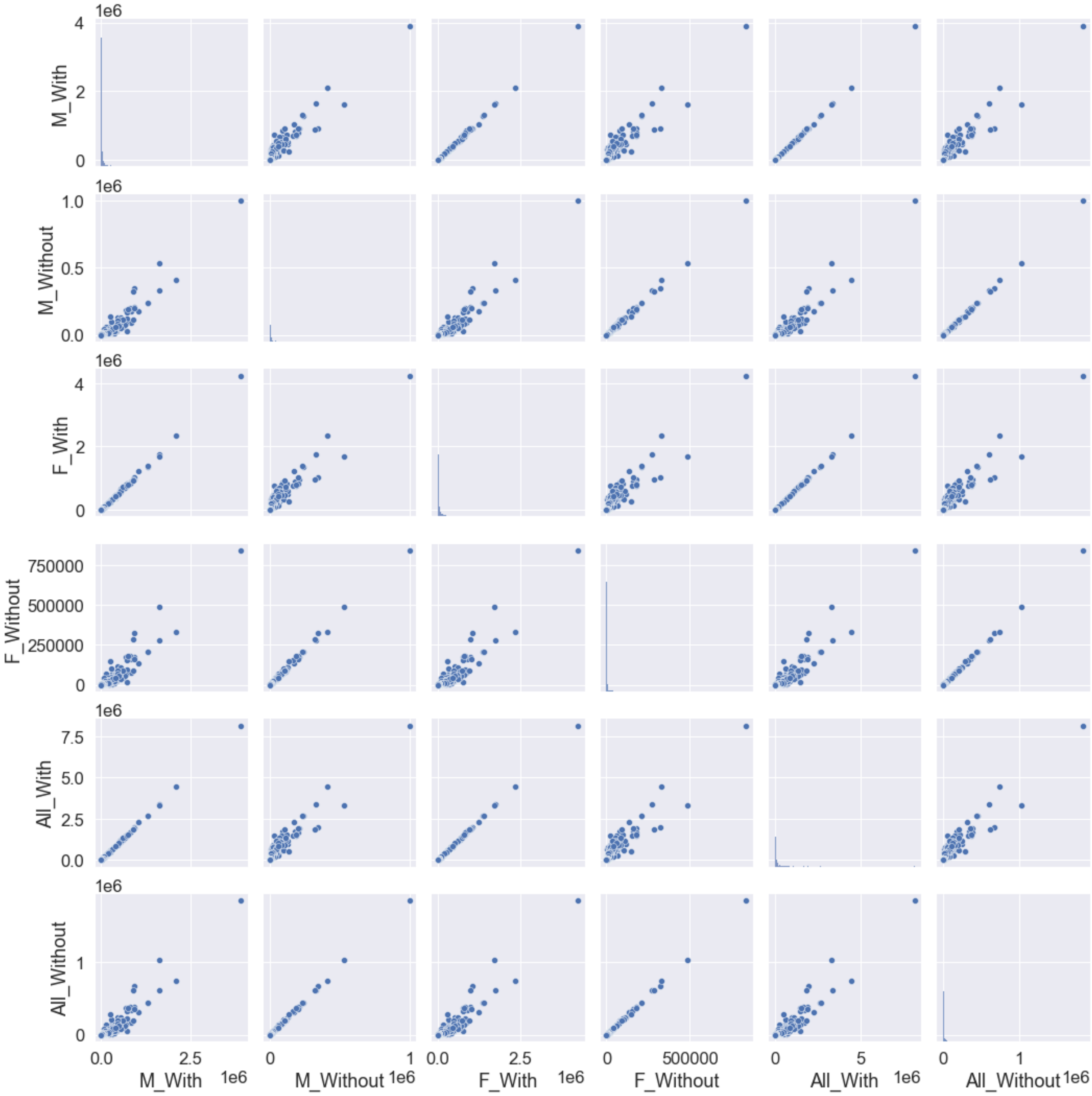

We can drop M_Poverty and F_Poverty as we see very high correlation between the gender related Poverty data.

In [40]:
```
finaldf = finaldf.drop(['M_Poverty', 'F_Poverty'],axis=1)
```

## Health Insurance

In [41]:
```
sns.pairplot(finaldf[['M_With','M_Without','F_With','F_Without','All_With','All_With
```

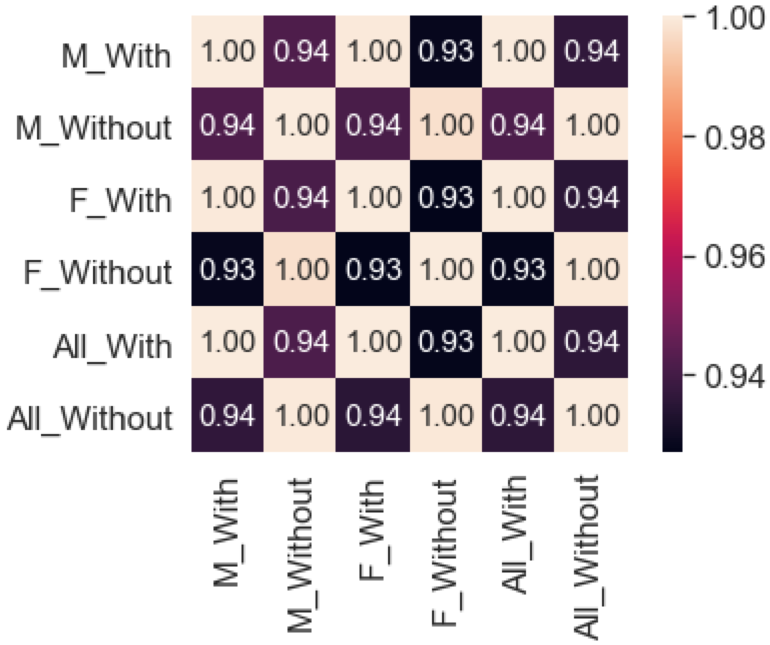
Out[41]:
```
<seaborn.axisgrid.PairGrid at 0x2c5bcab7340>
```




In [42]:

```
# heatmap for correlation

cols = ['M_With','M_Without','F_With','F_Without','All_With','All_Without']
cm = np.corrcoef(finaldf[['M_With','M_Without','F_With','F_Without','All_With','All_
sns.set(font_scale=1.5)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size
                 yticklabels=cols, xticklabels=cols)
plt.show()
```




Similar to Poverty, We can drop M_With, M_Without, F_With, F_Without due to high correlation.

In [43]:
```
finaldf = finaldf.drop(['M_With','M_Without','F_With','F_Without'],axis=1)
```

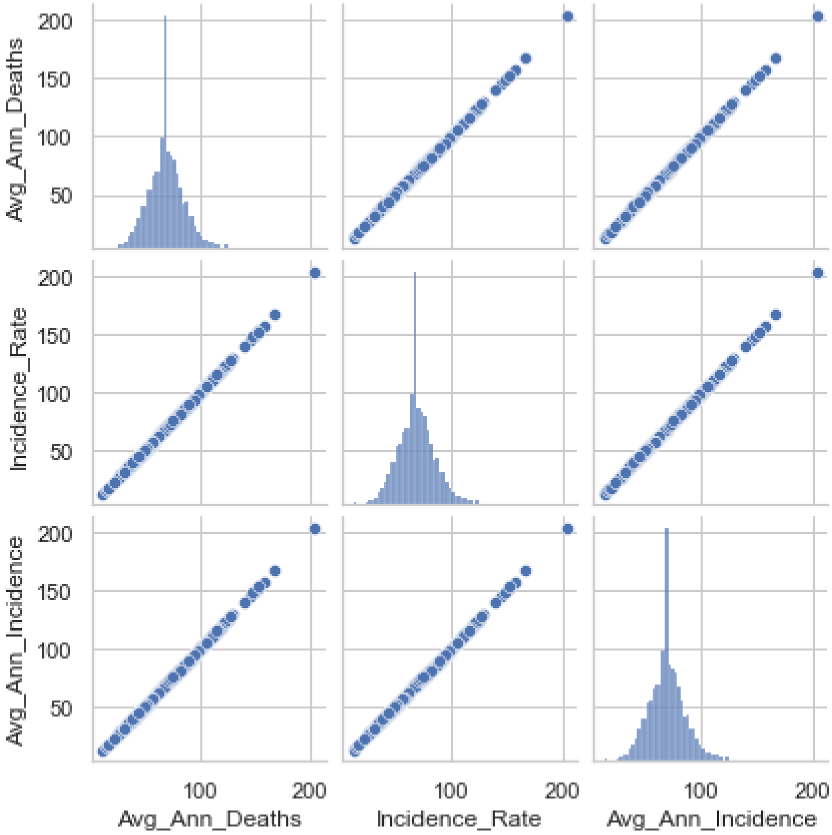
In [44]:
```
finaldf.head()
```

Out[44]:

| | All_Poverty | Med_Income | All_With | All_Without | Incidence_Rate | Avg_Ann_Incidence | Mortality_I |
|---|---|---|---|---|---|---|---|
| **2** | 23914 | 78326.0 | 243173 | 44638 | 61.5 | 61.5 | |
| **3** | 4364 | 51012.0 | 13023 | 4482 | 62.7 | 62.7 | |
| **7** | 7752 | 71068.0 | 80815 | 12279 | 58.1 | 58.1 | |
| **9** | 2110 | 85746.0 | 27321 | 4646 | 35.1 | 35.1 | |
| **10** | 5558 | 63684.0 | 44059 | 11868 | 64.9 | 64.9 | |

# Incidence, Death and Incidence Rate

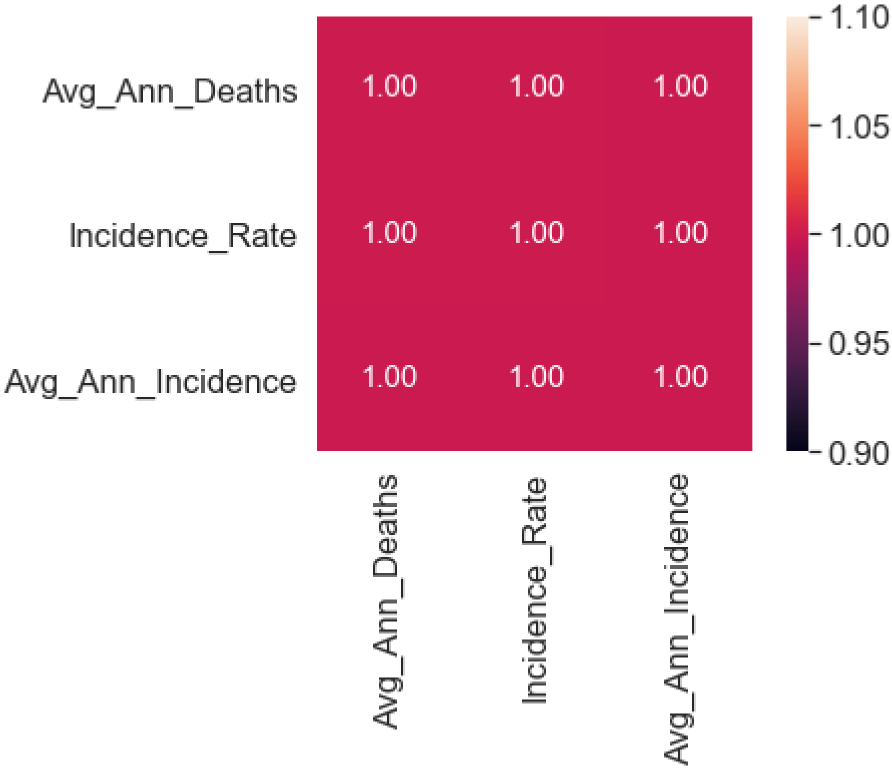
In [45]:
```
cols = ['Avg_Ann_Deaths','Incidence_Rate','Avg_Ann_Incidence']
sns.set(style='whitegrid', context='notebook')
sns.pairplot(finaldf[cols],size=2)
plt.show()
```




In [46]:

```
# heatmap for correlation

cols = ['Avg_Ann_Deaths','Incidence_Rate','Avg_Ann_Incidence']
cm = np.corrcoef(finaldf[['Avg_Ann_Deaths','Incidence_Rate','Avg_Ann_Incidence']].va
sns.set(font_scale=1.5)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size
                 yticklabels=cols, xticklabels=cols)
plt.show()
```




Since Avg_Ann_Deaths, Incidence_Rate and Avg_Ann_Incidence Represent same thing (confirmed by the correlation heatmap), we can keep the already cleaned Incidence_Rate and drop the other 2.

In [47]:
```
finaldf = finaldf.drop(['Avg_Ann_Deaths','Avg_Ann_Incidence'],axis=1)
```

# Normalizing Data by Population

We create new columns with Population Normalization for Poverty and Health Insurance Data. We use the Population estimates we acquired.

In [48]:
```
finaldf
```

Out[48]:

| | All_Poverty | Med_Income | All_With | All_Without | Incidence_Rate | Mortality_Rate | POPESTIMAT |
|---|---|---|---|---|---|---|---|
| **2** | 23914 | 78326.0 | 243173 | 44638 | 61.5 | 47.3 | 2 |
| **3** | 4364 | 51012.0 | 13023 | 4482 | 62.7 | 58.3 | |
| **7** | 7752 | 71068.0 | 80815 | 12279 | 58.1 | 54 | |
| **9** | 2110 | 85746.0 | 27321 | 4646 | 35.1 | 34.4 | |
| **10** | 5558 | 63684.0 | 44059 | 11868 | 64.9 | 50.1 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **3129** | 5058 | 69022.0 | 38491 | 6001 | 39.9 | 28.4 | |
| **3130** | 1638 | 75325.0 | 18503 | 3750 | 23.7 | 29.1 | |
| **3131** | 2845 | 56569.0 | 17843 | 2916 | 31.7 | 22.1 | |
| **3132** | 1137 | 47652.0 | 6839 | 1394 | 50.0 | 38.2 | |
| **3133** | 958 | 57738.0 | 6014 | 768 | 44.9 | 43.5 | |

2809 rows × 9 columns

In [49]:
```
for col in ['All_Poverty','All_With', 'All_Without']:
    finaldf[col + "_PN"] = finaldf[col] / finaldf.POPESTIMATE2015 * 10**5
```

In [50]:
```
finaldf
```

Out[50]:

| | All_Poverty | Med_Income | All_With | All_Without | Incidence_Rate | Mortality_Rate | POPESTIMAT |
|---|---|---|---|---|---|---|---|
| **2** | 23914 | 78326.0 | 243173 | 44638 | 61.5 | 47.3 | 2 |
| **3** | 4364 | 51012.0 | 13023 | 4482 | 62.7 | 58.3 | |
| **7** | 7752 | 71068.0 | 80815 | 12279 | 58.1 | 54 | |
| **9** | 2110 | 85746.0 | 27321 | 4646 | 35.1 | 34.4 | |
| **10** | 5558 | 63684.0 | 44059 | 11868 | 64.9 | 50.1 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **3129** | 5058 | 69022.0 | 38491 | 6001 | 39.9 | 28.4 | |

| | All_Poverty | Med_Income | All_With | All_Without | Incidence_Rate | Mortality_Rate | POPESTIMAT |
|---|---|---|---|---|---|---|---|
| **3130** | 1638 | 75325.0 | 18503 | 3750 | 23.7 | 29.1 | |
| **3131** | 2845 | 56569.0 | 17843 | 2916 | 31.7 | 22.1 | |
| **3132** | 1137 | 47652.0 | 6839 | 1394 | 50.0 | 38.2 | |
| **3133** | 958 | 57738.0 | 6014 | 768 | 44.9 | 43.5 | |

2809 rows × 12 columns

Dropping the non-normalized columns

In [51]:

```
finaldf = finaldf.drop(['All_Poverty','All_With','All_Without'],axis=1)
```

In [52]:

```
finaldf
```

Out[52]:

| | Med_Income | Incidence_Rate | Mortality_Rate | POPESTIMATE2015 | rising | falling | All_Poverty_P |
|---|---|---|---|---|---|---|---|
| **2** | 78326.0 | 61.5 | 47.3 | 298695 | 0 | 0 | 8006.1601 |
| **3** | 51012.0 | 62.7 | 58.3 | 17946 | 0 | 0 | 24317.3966 |
| **7** | 71068.0 | 58.1 | 54 | 99631 | 0 | 0 | 7780.7108 |
| **9** | 85746.0 | 35.1 | 34.4 | 32756 | 0 | 0 | 6441.5679 |
| **10** | 63684.0 | 64.9 | 50.1 | 58059 | 0 | 0 | 9573.0205 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **3129** | 69022.0 | 39.9 | 28.4 | 44626 | 0 | 0 | 11334.1997 |
| **3130** | 75325.0 | 23.7 | 29.1 | 23125 | 0 | 0 | 7083.2432 |
| **3131** | 56569.0 | 31.7 | 22.1 | 20822 | 0 | 0 | 13663.4329 |
| **3132** | 47652.0 | 50.0 | 38.2 | 8328 | 0 | 0 | 13652.7377 |
| **3133** | 57738.0 | 44.9 | 43.5 | 7234 | 0 | 0 | 13243.0190 |

2809 rows × 9 columns

# Visualising Final Data and Exploratory Analysis

In [53]:

```
finaldf.describe()
```

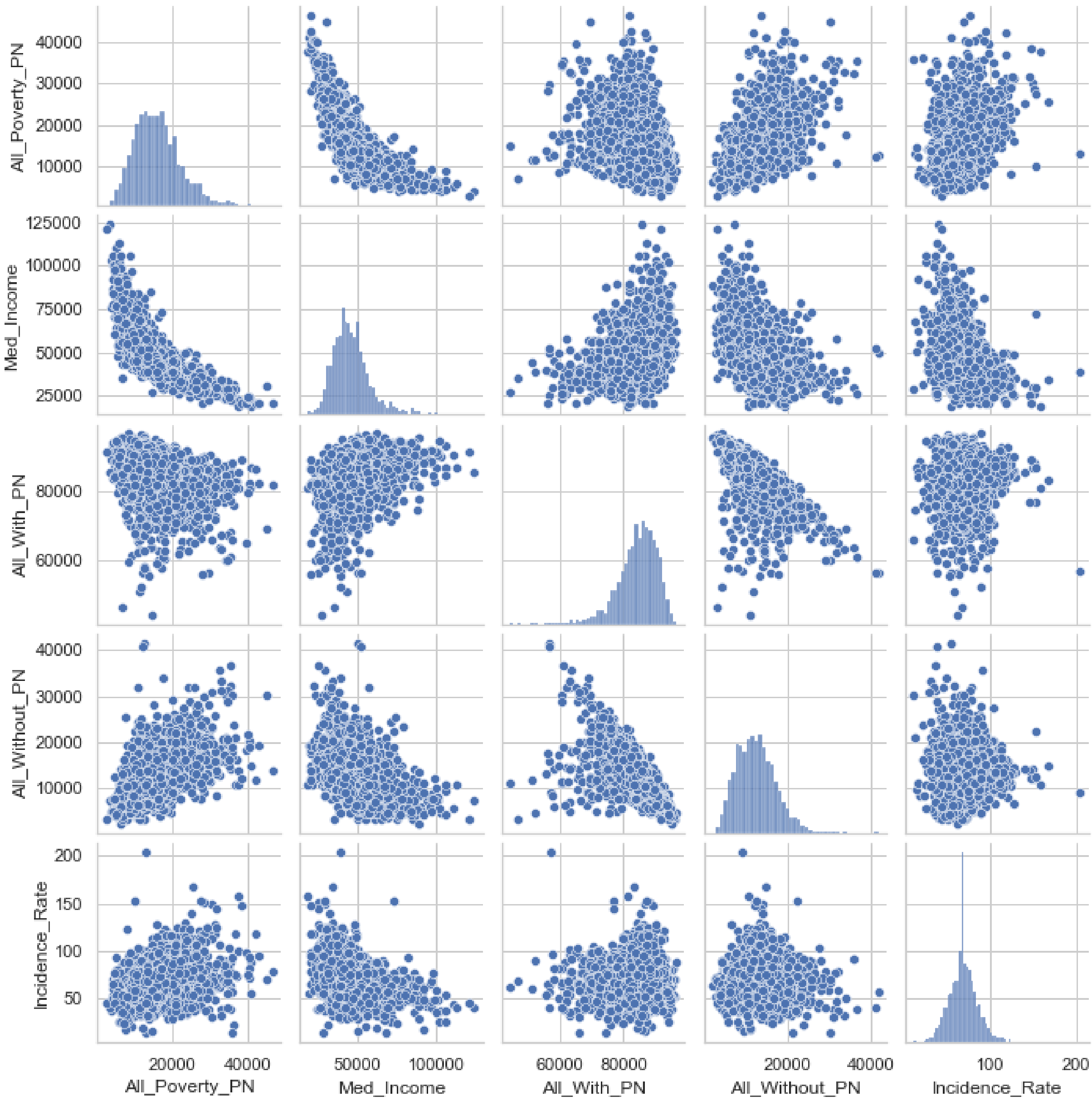
Out[53]:

| | Med_Income | Incidence_Rate | POPESTIMATE2015 | rising | falling | All_Poverty_PN |
|---|---|---|---|---|---|---|
| **count** | 2809.000000 | 2809.000000 | 2.809000e+03 | 2809.000000 | 2809.000000 | 2809.000000 |
| **mean** | 46812.890352 | 70.174404 | 1.139303e+05 | 0.013884 | 0.070132 | 16251.736718 |
| **std** | 12420.272665 | 17.008618 | 3.463393e+05 | 0.117030 | 0.255414 | 6119.212012 |
| **min** | 19328.000000 | 13.500000 | 2.302000e+03 | 0.000000 | 0.000000 | 2598.617910 |
| **25%** | 38698.000000 | 59.800000 | 1.484400e+04 | 0.000000 | 0.000000 | 11867.195818 |

| | Med_Income | Incidence_Rate | POPESTIMATE2015 | rising | falling | All_Poverty_PN |
|---|---|---|---|---|---|---|
| **50%** | 45048.000000 | 69.700000 | 3.118300e+04 | 0.000000 | 0.000000 | 15640.873409 |
| **75%** | 52149.000000 | 79.100000 | 7.916100e+04 | 0.000000 | 0.000000 | 19775.868774 |
| **max** | 123453.000000 | 203.700000 | 1.017029e+07 | 1.000000 | 1.000000 | 46489.942720 |

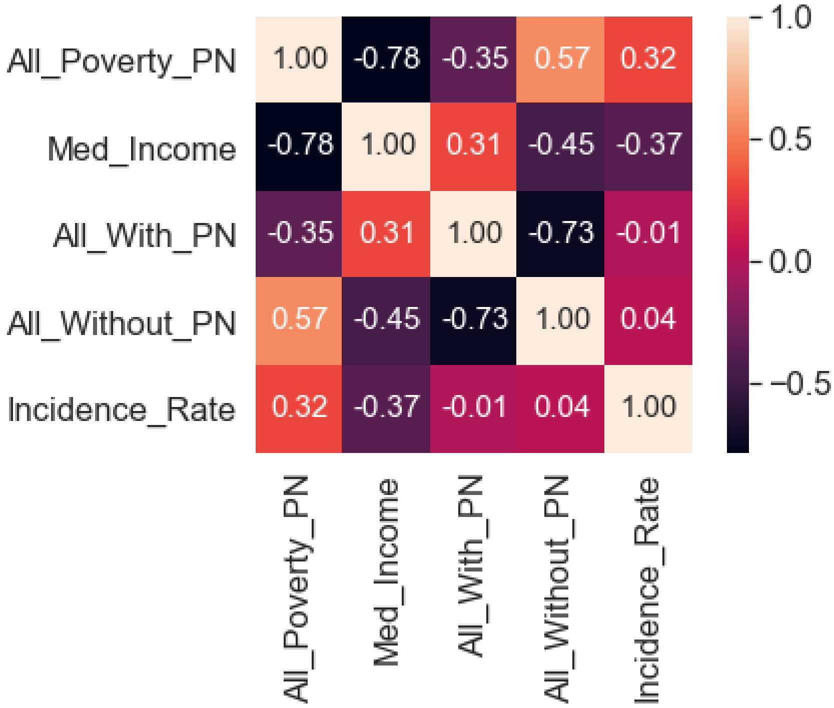
In [54]:

```
cols = ['All_Poverty_PN', 'Med_Income', 'All_With_PN', 'All_Without_PN',
                'Incidence_Rate']
sns.set(style='whitegrid', context='notebook')
sns.pairplot(finaldf[cols],height=2)
plt.show()
```




In [55]:

```
cm = np.corrcoef(finaldf[cols].values.T)
sns.set(font_scale=1.5)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size
                 yticklabels=cols, xticklabels=cols)
plt.show()
```

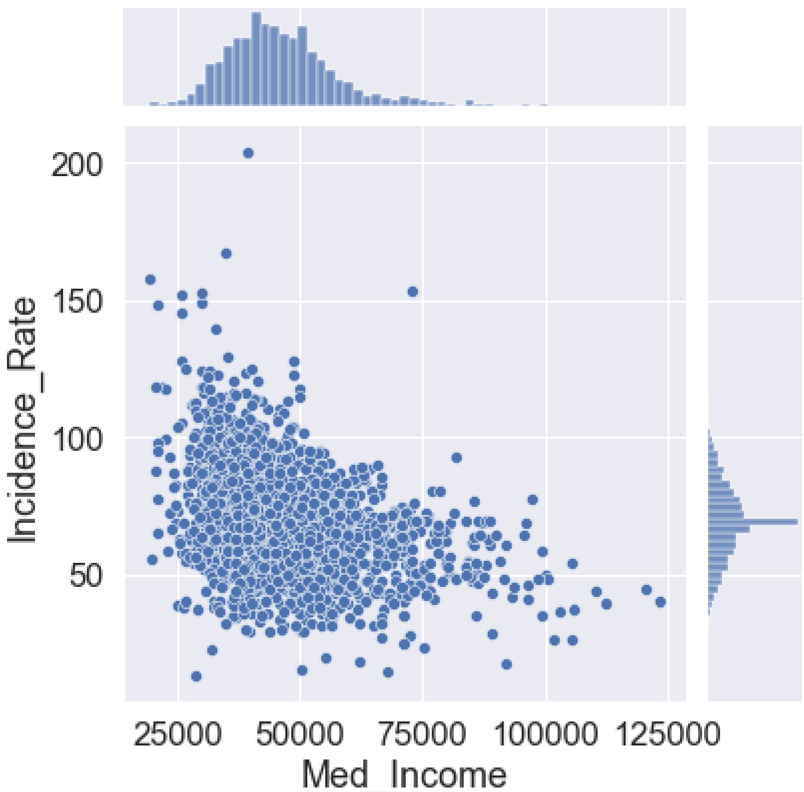



We don't have any high correlation values, which shows all variables are significant.

# Income Trends with Incidence Rate

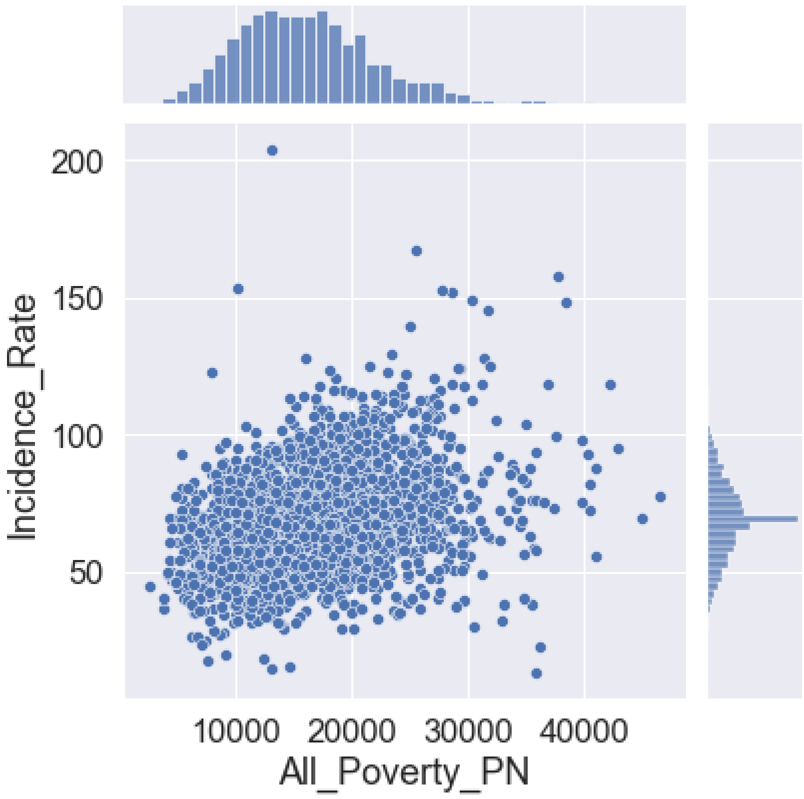
In [56]:
```
sns.jointplot(x='Med_Income',y='Incidence_Rate' ,data=finaldf)
```

Out[56]: <seaborn.axisgrid.JointGrid at 0x2c5c96c6400>




In [57]:
```
sns.jointplot(x='All_Poverty_PN',y='Incidence_Rate' ,data=finaldf)
```

Out[57]: <seaborn.axisgrid.JointGrid at 0x2c5c9862c70>



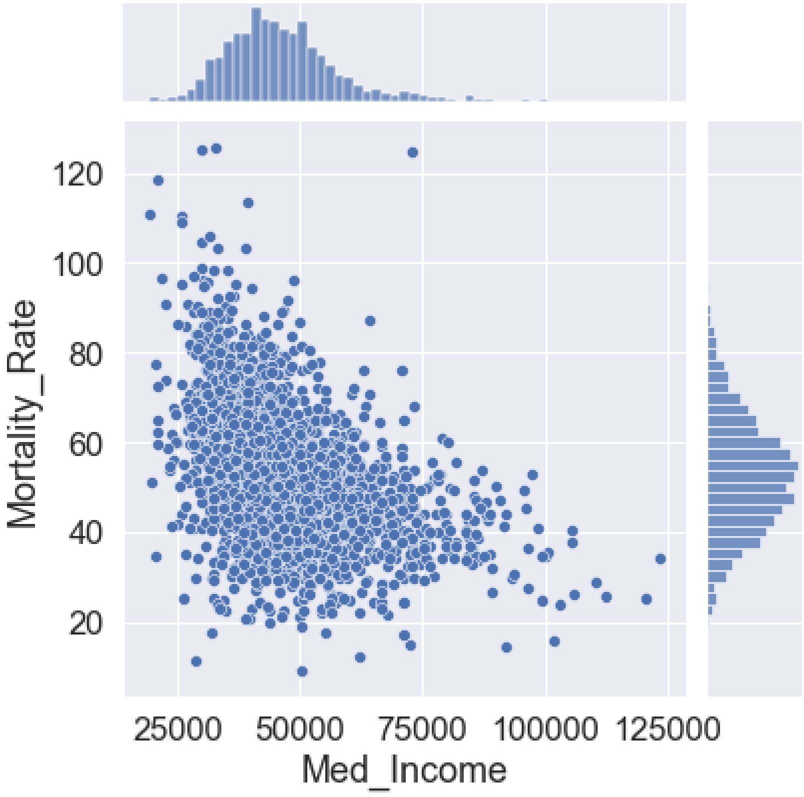

There is clear visual evidence that lower income groups have higher incidence rate.

## Trends with Mortality Rate

We use scatterplots for our columns with mortality rate to visualize trends.

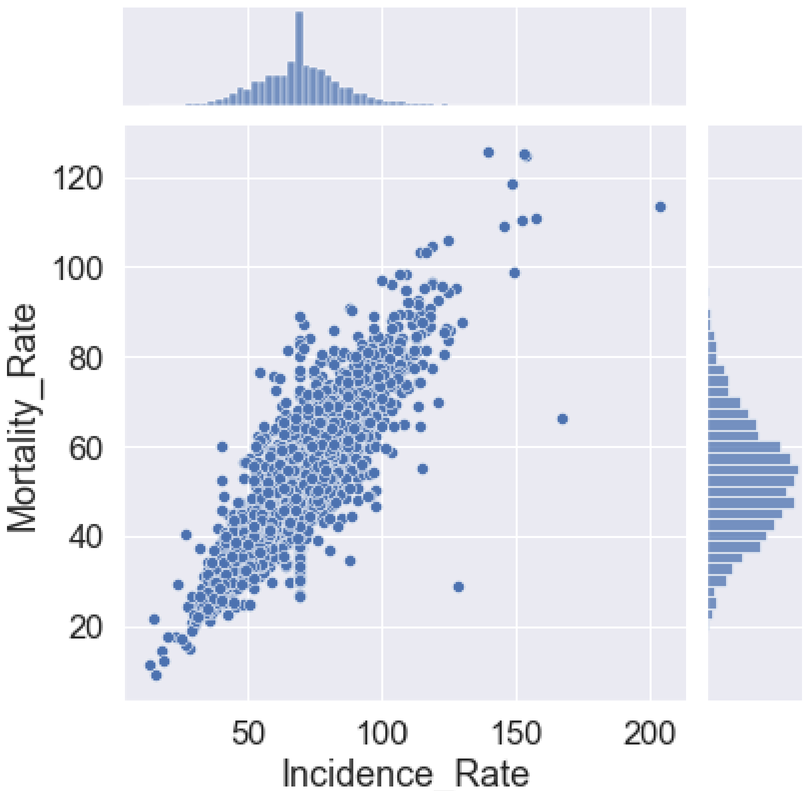
In [58]:
```
sns.jointplot(x='Med_Income',y='Mortality_Rate' ,data=finaldf)
```

Out[58]: <seaborn.axisgrid.JointGrid at 0x2c5c9b53190>



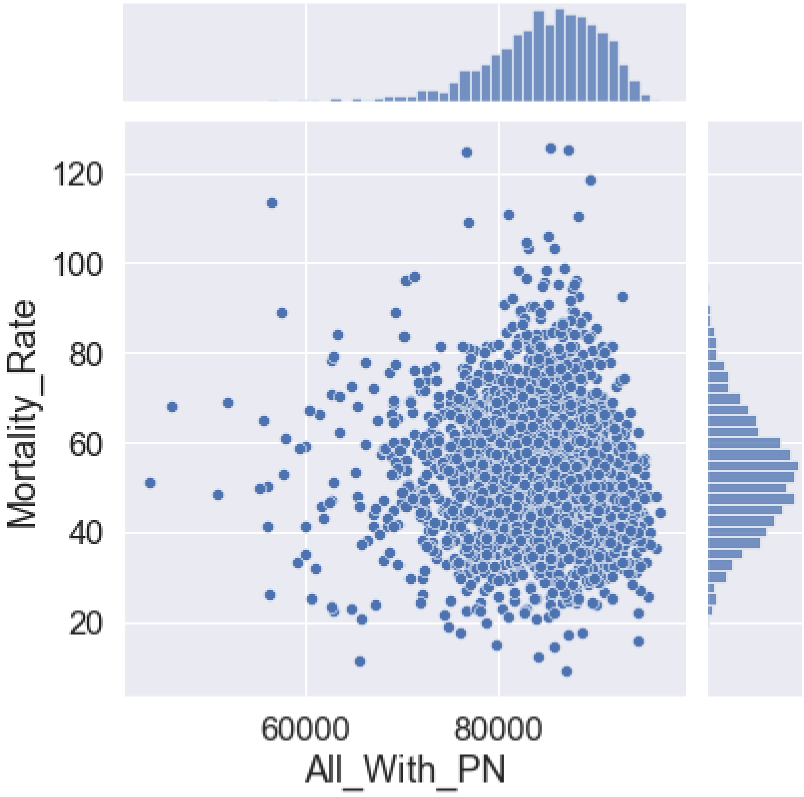

In [59]:
```
sns.jointplot(x='Incidence_Rate',y='Mortality_Rate',data=finaldf)
```

Out[59]: <seaborn.axisgrid.JointGrid at 0x2c5c9d24f40>



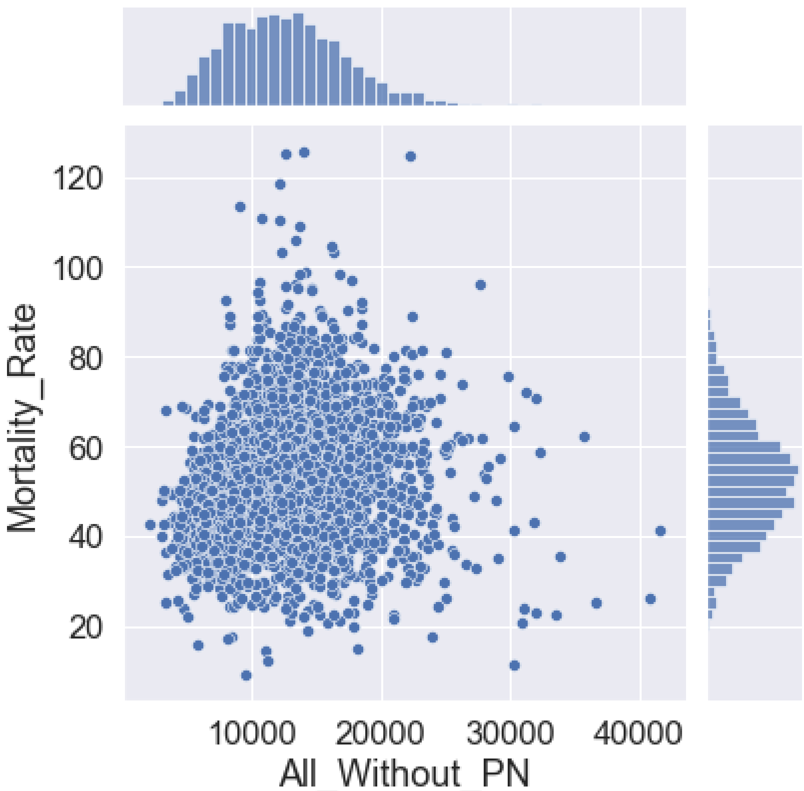

In [60]:
```
sns.jointplot(x='All_With_PN',y='Mortality_Rate',data=finaldf)
```

Out[60]: <seaborn.axisgrid.JointGrid at 0x2c5c9f568b0>



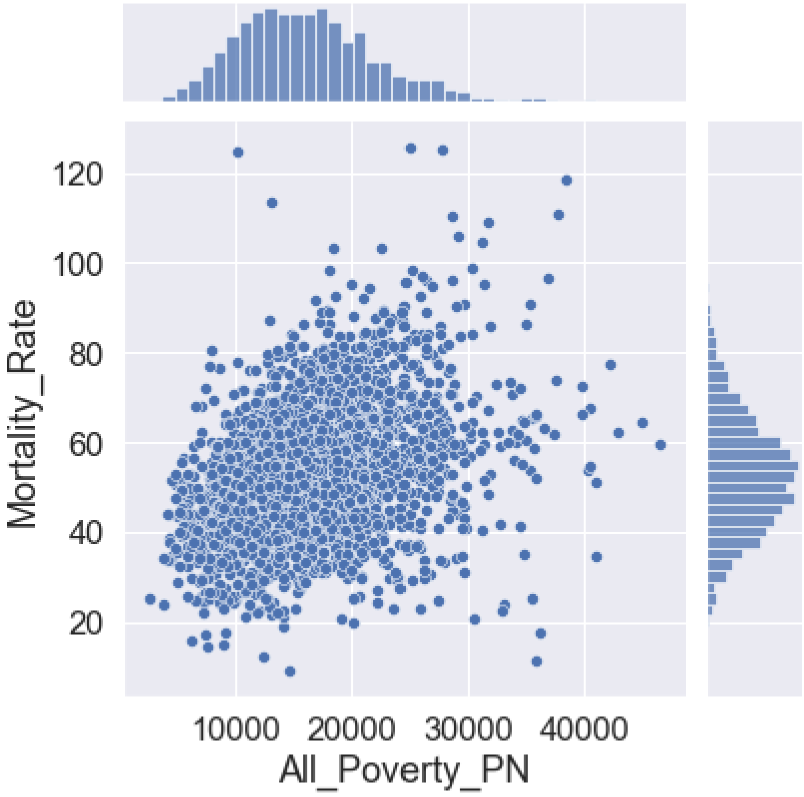

In [61]:
```
sns.jointplot(x='All_Without_PN',y='Mortality_Rate',data=finaldf)
```

Out[61]: <seaborn.axisgrid.JointGrid at 0x2c5cb1113a0>



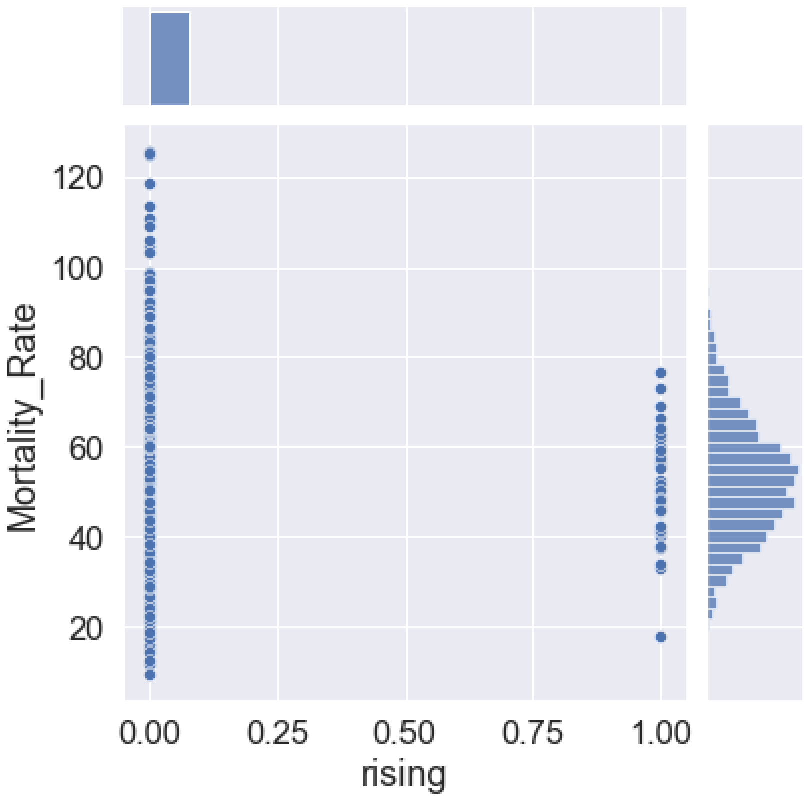

In [62]:
```
sns.jointplot(x='All_Poverty_PN',y='Mortality_Rate',data=finaldf)
```

Out[62]: <seaborn.axisgrid.JointGrid at 0x2c5cb2c2fa0>



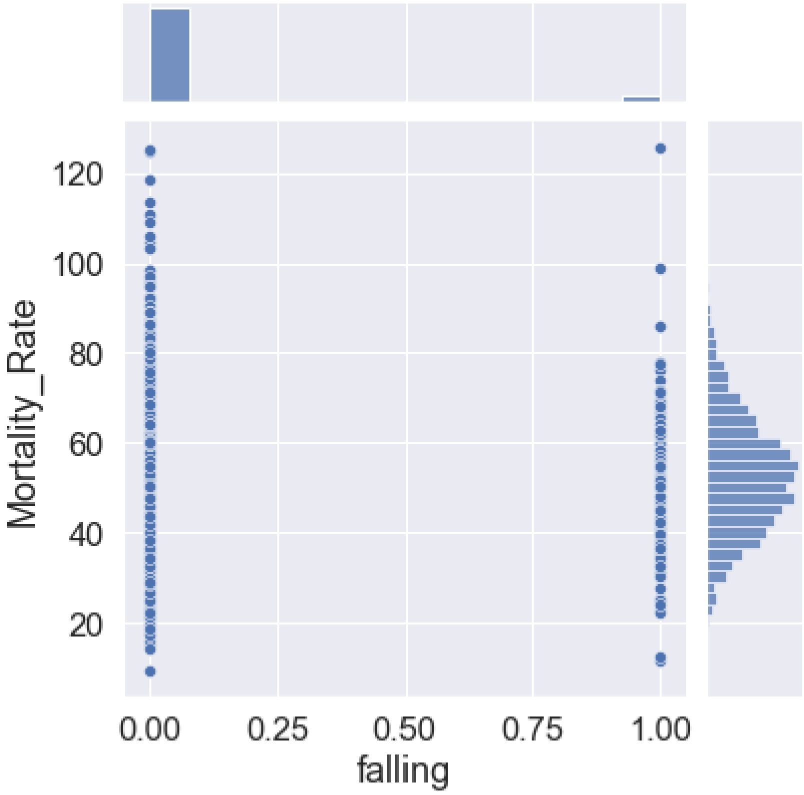

In [63]:
```
sns.jointplot(x='rising',y='Mortality_Rate',data=finaldf)
```

Out[63]: <seaborn.axisgrid.JointGrid at 0x2c5cb3a85e0>




In [64]:
```
sns.jointplot(x='falling',y='Mortality_Rate',data=finaldf)
```

Out[64]: <seaborn.axisgrid.JointGrid at 0x2c5c9ca9790>




# Model Building

In [65]:
```
y = finaldf['Mortality_Rate']
X = finaldf[['All_Poverty_PN', 'Med_Income', 'All_With_PN', 'All_Without_PN',
                'Incidence_Rate','rising','falling']]
```

## Training the Model

In [66]:
```
from sklearn.linear_model import LinearRegression
```

In [67]:
```
lm = LinearRegression()
```

In [68]:
```
model = lm.fit(X,y)
```

## Intercept and Coefficients

In [69]:
```
print(lm.intercept_)
```

```
6.144721764929308
```

In [70]:
```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

Out[70]:

| | Coefficient |
|---|---|
| **All_Poverty_PN** | 0.000057 |

| | Coefficient |
|---|---|
| **Med_Income** | -0.000116 |
| **All_With_PN** | 0.000028 |
| **All_Without_PN** | 0.000240 |
| **Incidence_Rate** | 0.656103 |
| **rising** | -0.986183 |
| **falling** | 0.682052 |

# Metrics and Model Evaluation

In [75]:

```
#R2 Score of the model

model.score(X,y)
```

Out[75]: 0.736365906887327

In [72]:

```
predictions = model.predict(X)
```

In [73]:

```
from sklearn import metrics
```

In [74]:

```
# various types of cost functions

print('MAE:', metrics.mean_absolute_error(y, predictions))
print('MSE:', metrics.mean_squared_error(y, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y, predictions)))
```

```
MAE: 5.240005014185609
MSE: 51.96484297101189
RMSE: 7.208664437398365
```

# ------------------------ THE END ------------------------