

# Tx\_Stats\_EDA\_code\_orosco

February 3, 2023

```
[ ]: #Christine Orosco  
#Texas demographic and health statistics EDA
```

```
[1]: import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from scipy import stats  
import math  
import numpy as np  
from sklearn.preprocessing import MinMaxScaler  
import statsmodels.stats.api as sms  
import statsmodels.formula.api as smf  
from scipy.stats import shapiro, normaltest, ttest_ind, ttest_rel
```

```
[2]: def Compute_Cohend(mean1, mean2, ser1, ser2, var1, title):  
    """Compute cohens'd to see the effect size"""  
  
    diff = mean1 - mean2  
    n1, n2, = len(ser1), len(ser2)  
  
    pooled_var = (n1 * var1 + n2) / (n1 + n2)  
    d = diff / math.sqrt(pooled_var)  
    print(f'{title} = {d} \n')  
    return d  
  
def Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2):  
    """Plot histogram using matplotlib in plotly plots two small plots  
    side by side for comparison"""  
  
    # Main Title  
    fig = plt.figure(figsize=(10,10))  
    title = fig.suptitle(subtitle, fontsize=14, fontweight="bold")  
  
    fig.subplots_adjust(top=0.88, wspace=0.3)  
  
    # Histogram 1
```

```

ax1 = ax1 = fig.add_subplot(1,2,1)
ax1.set_xlabel(xlab1)
ax1.set_ylabel("Frequency")

freq1, bins1, patches1 = ax1.hist(ser1, bins=10, color='darksalmon',
↪edgecolor='darkred', linewidth=1)

# Histogram 2
ax2 = ax2 = fig.add_subplot(1,2,2)
ax2.set_xlabel(xlab2)
ax2.set_ylabel("Frequency")

freq2, bins2, patches2 = ax2.hist(ser2, bins=10, color='green',
↪edgecolor='darkblue', linewidth=1)

return

def Sns_Kde(subtitle, xlab1, xlab2, ser1, mean1, ser2, mean2):
    """Plot KDE using seaborn- plot side by side for
        comparison"""

    # Main Title
    fig = plt.figure(figsize=(10,10))
    title = fig.suptitle(subtitle, \
                        fontsize=14, fontweight="bold")
    fig.subplots_adjust(top=0.88, wspace=0.3)

    # KDE 1
    ax1 = fig.add_subplot(1,2,1)
    ax1.set_xlabel(xlab1)
    ax1.set_ylabel("Density")
    sns.kdeplot(ser1, shade=True, color='darksalmon' )
    plt.axvline(x=mean1, color="black", linestyle="--")

    # KDE 2
    ax2 = ax2 = fig.add_subplot(1,2,2)
    ax2.set_xlabel(xlab2)
    ax2.set_ylabel("Density")
    sns.kdeplot(ser2, shade=True, color = 'green')
    plt.axvline(x=mean2, color="black", linestyle="--")
    return

```

```

def Cdf_Plot(subtitle, data):
    """Plot the CDF plots"""

    fig = plt.figure(figsize=(10,10))
    title = fig.suptitle(subtitle,\
                        fontsize=14, fontweight="bold")
    kwargs1 = {'cumulative': True}
    kwargs = {'cumulative': True, 'density': True}
    sns.distplot(data, hist_kws=kwargs, kde_kws=kwargs1)
    return

def Pmf_plot(subtitle, data, xlab):
    """Plot a PMF"""

    fig = plt.figure(figsize=(10,10))
    title = fig.suptitle(subtitle, \
                        fontsize=14, fontweight="bold")

    fig.subplots_adjust(top=0.88, wspace=0.3)
    ax1 = fig.add_subplot(1,1,1)
    ax1.set_xlabel(xlab)
    probs=data.value_counts(normalize=True)
    sns.barplot(probs.index, probs.values)
    return

def Heatmap_Plot(subtitle, xlab1, xlab2, data1, data2):
    """Plot a heatmap or Correlation Map"""
    fig = plt.figure(figsize=(13,13))
    title = fig.suptitle(subtitle, \
                        fontsize=14, fontweight="bold")
    fig.subplots_adjust(top=0.88, wspace=0.3)
    # Heatmap1
    ax1 = fig.add_subplot(1,2,1)
    ax1.set_xlabel(xlab1)
    sns.heatmap(
        data1,
        vmin=-1, vmax=1, center=0,
        cmap=sns.diverging_palette(20, 220, n=220),
        square=True)

    ax1.set_xticklabels(
        ax1.get_xticklabels(),
        rotation=45,

```

```

        horizontalalignment='right')

    #Heatmap2
    ax2 = fig.add_subplot(1,2,2)
    ax2.set_xlabel(xlab2)
    sns.heatmap(
        data2,
        vmin=-1, vmax=1, center=0,
        cmap=sns.diverging_palette(20, 220, n=220),
        square=True)

    ax2.set_xticklabels(
        ax2.get_xticklabels(),
        rotation=45,
        horizontalalignment='right')
    return

def sns_Scatter(subtitle, xlab, ylab, x_val, y_val, data):
    "Scatter plots for variables"
    fig = plt.figure(figsize=(10,10))
    title = fig.suptitle(subtitle, \
        fontsize=14, fontweight="bold")
    fig.subplots_adjust(top=0.88, wspace=0.3)
    # Scatter Plots
    ax1 = fig.add_subplot(1,1,1)
    ax1.set_xlabel(xlab)
    ax1.set_ylabel(ylab)
    sns.scatterplot(x = x_val, y = y_val, data=data)
    return

def sns_Lmplot(subtitle, x_val, y_val, data):
    """Plot a rgeression line with the obs"""
    fig = plt.figure(figsize=(10,10))
    title = fig.suptitle(subtitle, \
        fontsize=14, fontweight="bold")
    fig.subplots_adjust(top=0.88, wspace=0.3)

    # Scatter Plots
    ax1 = fig.add_subplot(1,4,1)
    sns.lmplot(x=x_val, y=y_val, data=data)

    return

```

```

[3]: # Set context to `paper`
sns.set(rc={"font.size":15,"axes.labelsize":10})

```

```
#fig, ax = plt.subplots(figsize=(10,10))
sns.set(color_codes=True)
```

```
[4]: # =====
# Using pg. 29 of your text as an example, compare two scenarios in your data,
# using a
# PMF. Reminder, this isn't comparing two variables against each other
# - it is the same variable, but a different scenario. Almost like a filter.
# The example in the book is first babies compared to all other babies, it
# is still the same variable, but breaking the data out based on criteria
# we are exploring (Chapter 3).
# =====
```

## 0.1 Read in Dataset

```
[5]: # =====
# Read csv file and drop first row which has texas cumulative dats
# =====
tx_data = pd.read_csv("~/tx_household.csv", sep = ",")
tx = pd.DataFrame(tx_data)
tx = tx.drop(tx.index[0])

# Create subsets

# RURAL
rural = tx[tx['Population'] <= 50000] # rural community

# URBAN
urban = tx[tx['Population'] > 50000]

# rename variables for ease
# Rural

s_percent = rural.percent_single
grad = rural.Grad_Rate
maths = rural.Math
read = rural.Read

# Urban

s_percent1 = urban.percent_single
grad1 = urban.Grad_Rate
maths1 = urban.Math
read1 = urban.Read
```

```
[6]: rural.describe()
```

```
[6]:      nbr_sing_household  nbr_household  percent_single      Rural  \
count      186.000000      186.000000      186.000000      186.000000
mean      1122.634409      3468.086022      31.521505      7986.478495
std       1049.116656      3022.640709      10.032492      7027.797094
min         0.000000         61.000000         1.000000         82.000000
25%        321.000000      1044.000000      25.250000      2954.000000
50%        768.000000      2715.000000      32.000000      4983.500000
75%       1685.000000      5078.250000      37.000000     11931.250000
max       4875.000000     14001.000000      65.000000     31172.000000

      Population      Math      Read  Grad_Rate  poor_health  \
count      186.000000     186.000000     186.000000     186.000000     186.000000
mean     14770.026882      2.941398      2.785484     94.392473     3064.365591
std     12605.944718      0.279684      0.237678      5.939492     2673.077929
min       152.000000      2.100000      2.000000     62.000000      23.000000
25%      4470.000000      2.700000      2.700000     93.000000     857.250000
50%     10989.000000      2.900000      2.800000     96.000000     2335.500000
75%     21421.000000      3.100000      2.900000     98.000000     4372.750000
max     49728.000000      4.000000      3.400000    100.000000    14489.000000

      poor_sleep  Cost_Burden  Uninsured_health  Physicians  food_Insecure  \
count      186.000000     186.000000         186.000000     186.000000     186.000000
mean     4692.655914     496.537634      2444.564516       5.317204     2178.870968
std     4040.692191     505.374179      2083.528517       6.139057     2018.025879
min        50.000000         0.000000         27.000000       0.000000      10.000000
25%     1418.500000     120.000000         812.750000       1.000000     572.500000
50%     3414.500000     336.000000      1827.500000       3.000000     1490.000000
75%     6814.750000     656.750000      3549.000000       8.000000     3105.000000
max     16987.000000     2451.000000      9222.000000      38.000000     9970.000000

      Life_Expectancy
count      186.000000
mean        77.393548
std         2.376930
min         71.900000
25%         75.800000
50%         77.300000
75%         78.975000
max         89.700000
```

```
[7]: urban.describe()
```

```
[7]:      nbr_sing_household  nbr_household  percent_single      Rural  \
count          68.000000      6.800000e+01      68.000000      68.000000
mean       31809.117647      9.693757e+04      31.264706     34735.838235
std       68062.434466      1.882317e+05       6.571348     18480.051069
min        2392.000000      9.762000e+03      17.000000      5022.000000
```

25%	5136.750000	1.750350e+04	26.750000	19776.500000
50%	9656.500000	3.161000e+04	32.000000	32598.000000
75%	25979.750000	8.180650e+04	37.000000	47476.750000
max	445154.000000	1.231476e+06	44.000000	103571.000000

	Population	Math	Read	Grad_Rate	poor_health \
count	6.800000e+01	68.000000	68.000000	68.000000	68.000000
mean	3.816856e+05	3.085294	2.894118	91.926471	74733.455882
std	7.236031e+05	0.197948	0.176941	4.341113	141018.568160
min	5.003100e+04	2.600000	2.500000	75.000000	8414.000000
25%	6.795200e+04	3.000000	2.800000	90.000000	14305.500000
50%	1.330275e+05	3.100000	2.900000	93.000000	23471.000000
75%	3.150315e+05	3.200000	3.000000	95.000000	68266.000000
max	4.698619e+06	3.600000	3.400000	97.000000	885812.000000

	poor_sleep	Cost_Burden	Uninsured_health	Physicians \
count	6.800000e+01	68.000000	68.000000	68.000000
mean	1.260119e+05	16881.529412	62499.058824	239.191176
std	2.434585e+05	36246.466308	135041.924584	462.947283
min	1.508600e+04	1182.000000	6687.000000	6.000000
25%	2.276400e+04	2361.000000	11253.500000	23.750000
50%	4.166000e+04	4829.500000	18019.500000	74.000000
75%	1.077918e+05	14472.000000	41986.500000	232.500000
max	1.593969e+06	240521.000000	908742.000000	2742.000000

	food_Insecure	Life_Expectancy
count	68.000000	68.000000
mean	54094.852941	78.020588
std	110360.809809	2.294286
min	4210.000000	73.300000
25%	10707.500000	76.200000
50%	17410.000000	78.300000
75%	47455.000000	79.425000
max	739120.000000	83.000000

**0.1.1 Is there a significant statistical difference between rural single parent**

**0.1.2 households and urban single parent households?**

```
[8]: # =====
# 1. Is there a significant statistical difference between rural single parent
# households and urban single parent households?
# look for counties that are considered rural 0 - 50K
# or nbr of households > 50k
# % single is based upon # hseholds
# Look at the normal distribution for the precent single
# In texas rural population is > than urban plus urban areas also
```

```

#   rural populations. Did not count these as I couldn't tell from
#   the numbers if the rural was counted seperately from the urban.
#   =====

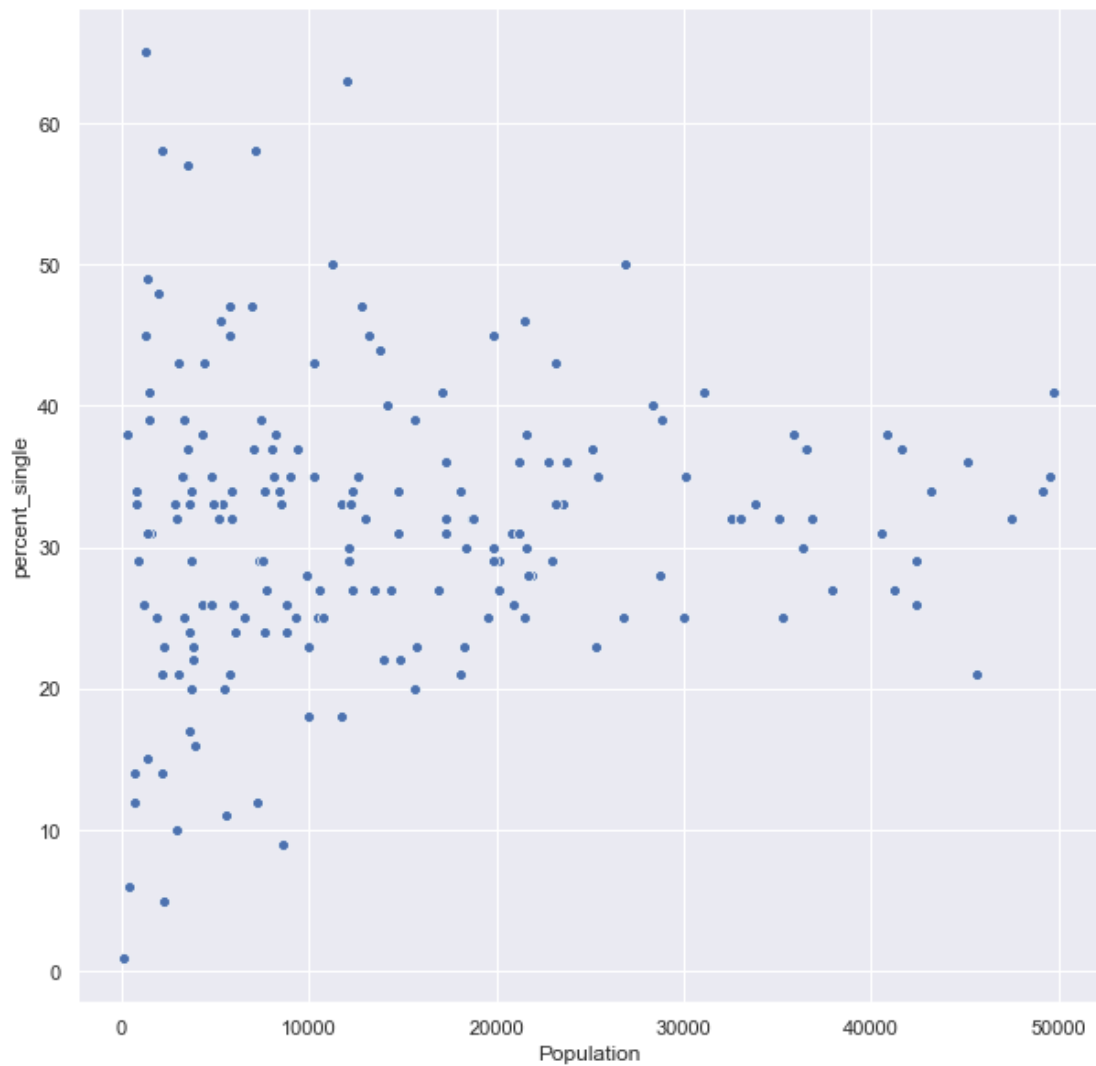
#   =====
#   Create two scatter plots comparing two variables and provide your analysis
#   on correlation and causation. Remember, covariance, Pearson's correlation,
#   and Non- Linear Relationships should also be considered during your analysis
#   (Chapter 7).
#   =====

subtitle = 'Single Parent Households Per County Population Rural'
x_val = 'Population'
y_val = 'percent_single'
xlab = 'Population'
ylab = 'Single Parent Household'
data = rural
sns_Scatter(subtitle, xlab, ylab, x_val, y_val, data)

```

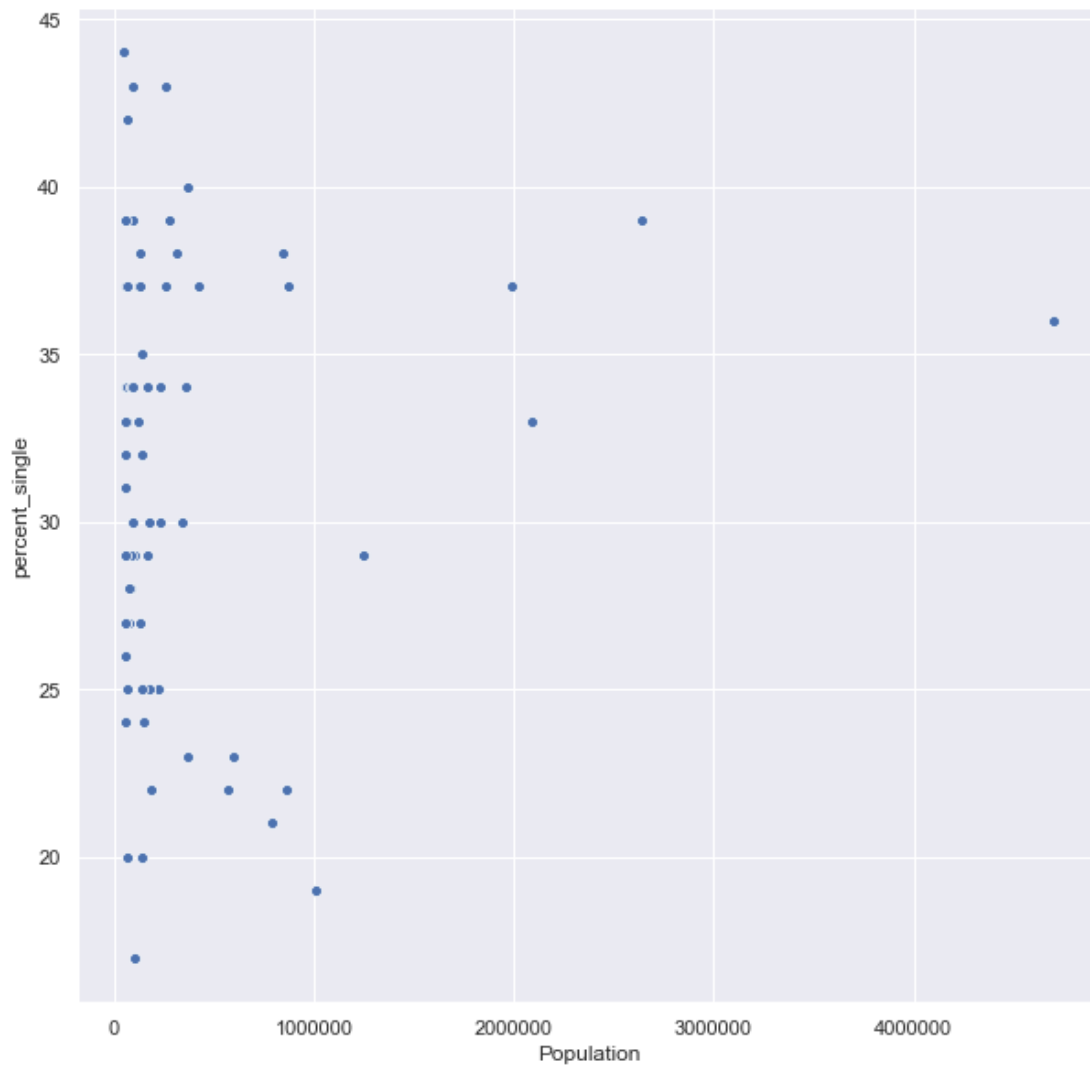


Single Parent Households Per County Population Rural



```
[9]: subtitle = 'Single Parent Households Per County Population Urban'
x_val = 'Population'
y_val = 'percent_single'
xlab = 'Population'
ylab = 'Single Parent Household'
data = urban
sns_Scatter(subtitle, xlab, ylab, x_val, y_val, data)
```

Single Parent Households Per County Population Urban



```
[10]: # =====
# Include the other descriptive characteristics about the variables: Mean,
# Mode, Spread, and Tails (Chapter 2)
# =====
# Compute Mean, Variance and Std.

rs_mean = s_percent.mean()
rs_var = s_percent.var()
rs_std = s_percent.std()
rs_mode = s_percent.mode()
```

```

print(f'Rural:  Mean = {rs_mean}  Var = {rs_var}  Std = {rs_std}  Mode = {rs_mode} \n')

us_mean = s_percent1.mean()
us_var = s_percent1.var()
us_std = s_percent1.std()
us_mode = s_percent1.mode()
print(f'Urban:  Mean = {us_mean}  Var = {us_var}  Std = {us_std} Mode = {us_mode} \n')

# Set values for plotting PDF and Cohen's d
mean1 = rs_mean
mean2 = us_mean
var1 = rs_var
ser1 = s_percent
ser2 = s_percent1

# =====
# Cohen's d
#
# What is the difference between single households in rural and urban
# couldn't derive the precentage of singles on rural area within urban
# counties.
# =====
ser1 = s_percent
ser2 = s_percent1
title = 'Cohens d for Single Parent Households Rural vs Urban'
cohen_d = Compute_Cohend(mean1, mean2, ser1, ser2, var1, title)

```

```

Rural:  Mean = 31.521505376344088  Var = 100.65088637024125  Std =
10.032491533524524  Mode = 0      32
dtype: int64

```

```

Urban:  Mean = 31.264705882352942  Var = 43.18261633011412  Std =
6.571348136426354  Mode = 0      37
dtype: int64

```

```

Cohens d for Single Parent Households Rural vs Urban = 0.029857843619705685

```

```

[11]: # =====
# Include a histogram of each of the 5 variables - in your summary and
# analysis, identify any outliers and explain the reasoning for them being
# outliers and how you believe they should be handled (Chapter 2).
# =====
# =====

```

```
# Plot 1 analytical distribution and provide your analysis on how it applies
# to the dataset you have chosen (Chapter 5).
# =====
```

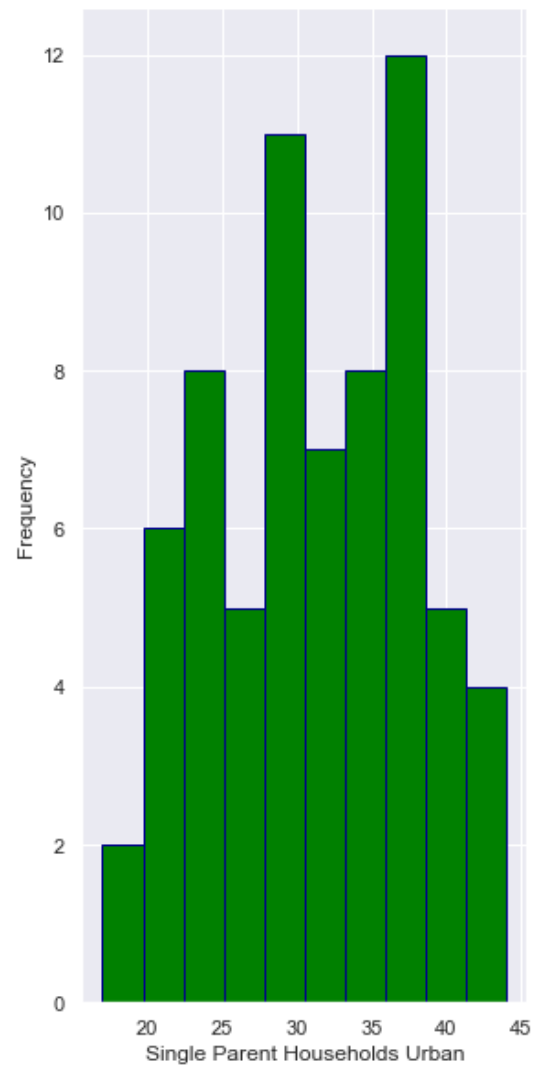
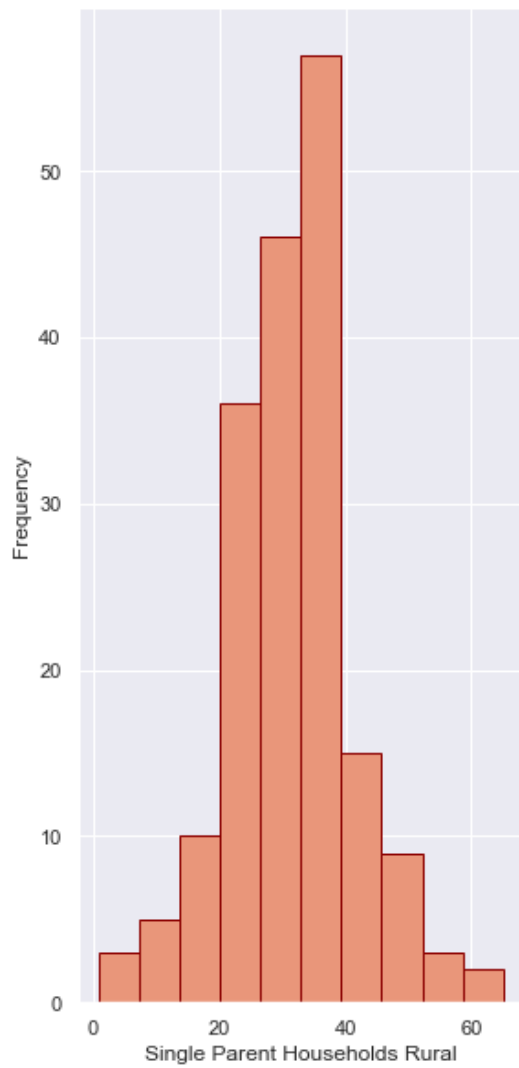
## 0.2 Histograms and PDFs

```
[12]: # Histogram and PDFs Single parent households rural vs urban
# =====
# plot side by side for comparison
subtitle = "Percent Single Parent Households Rural vs Urban"
xlab1 = "Single Parent Households Rural"
xlab2 = "Single Parent Households Urban"
binz = 10

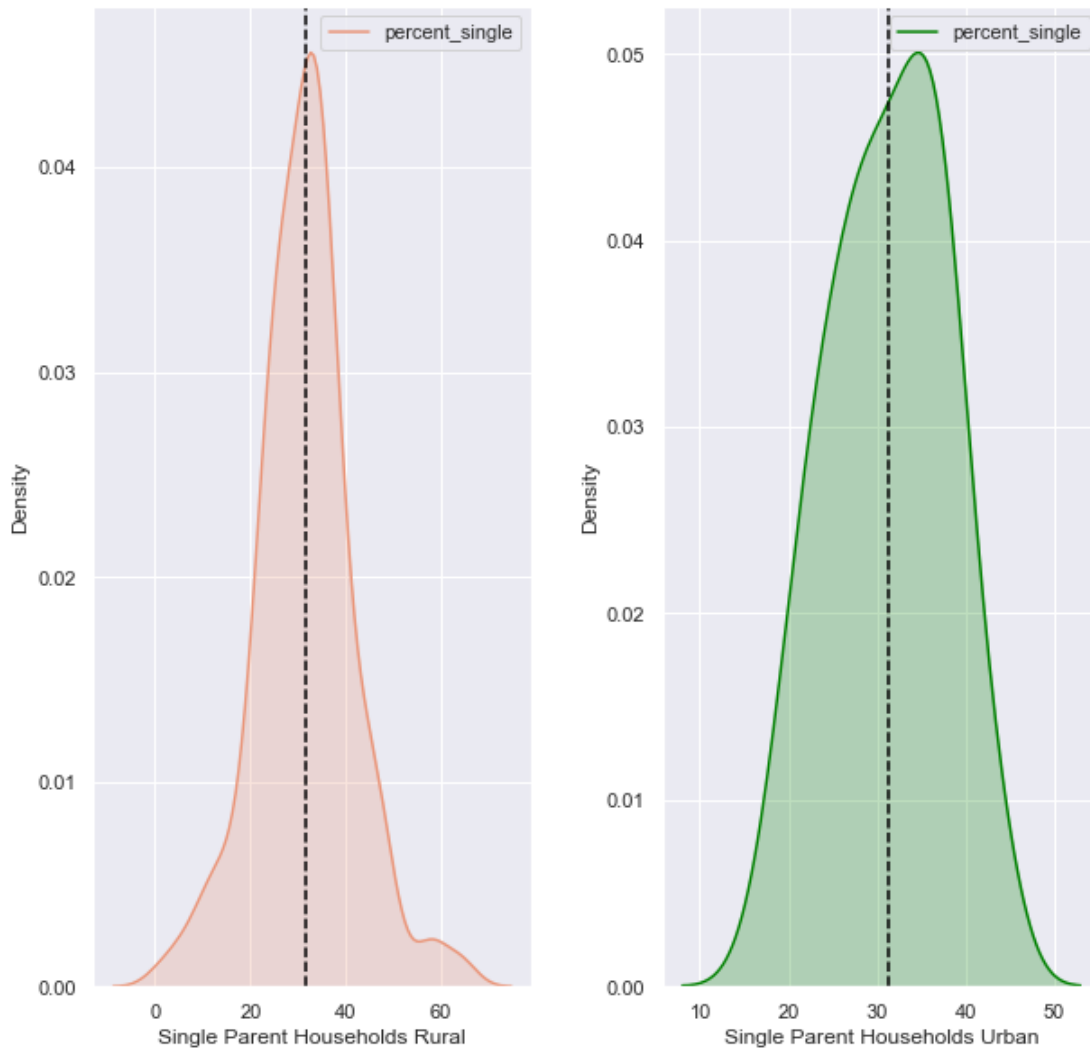
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

Percent Single Parent Households Rural vs Urban



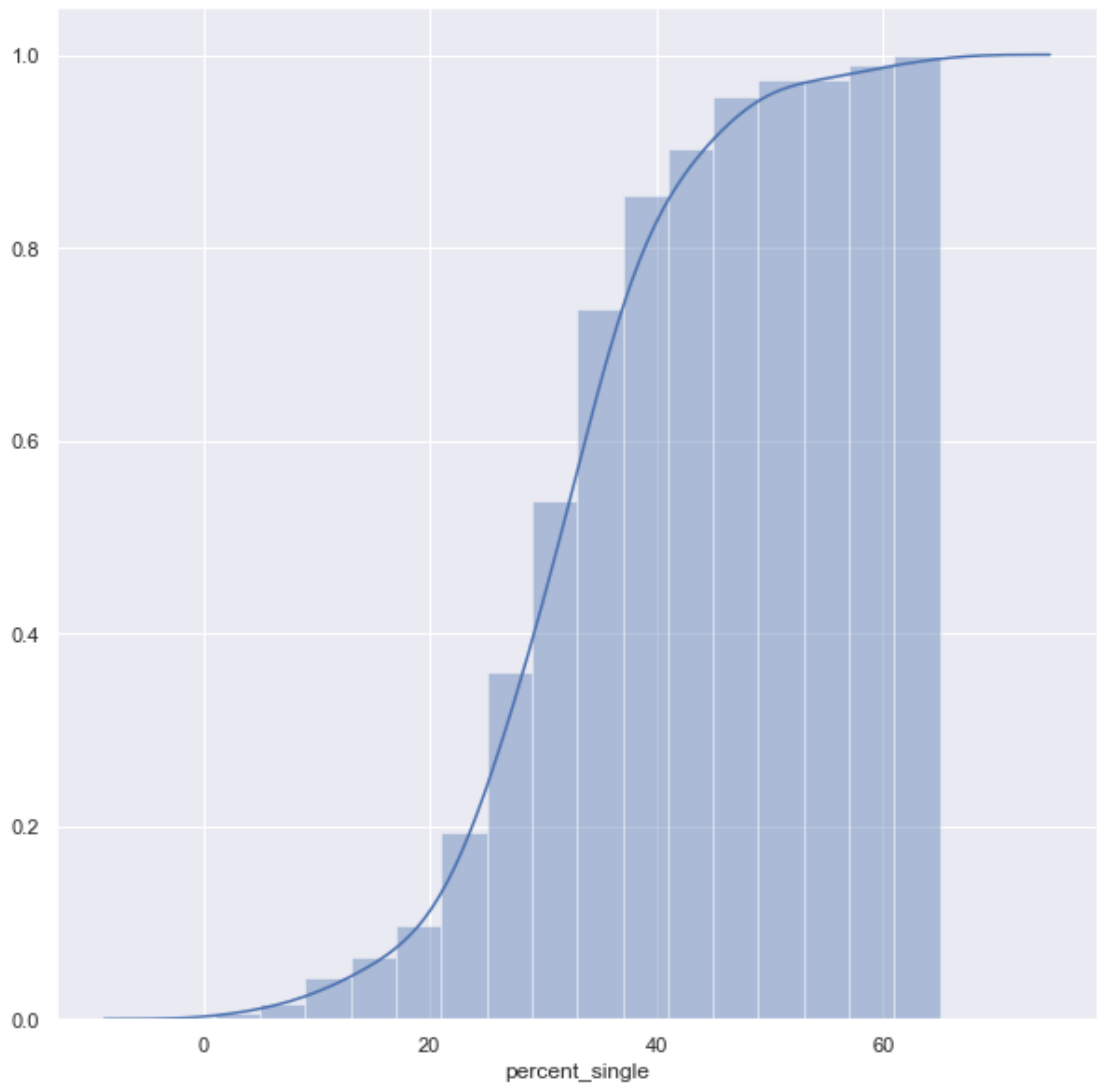
## Percent Single Parent Households Rural vs Urban



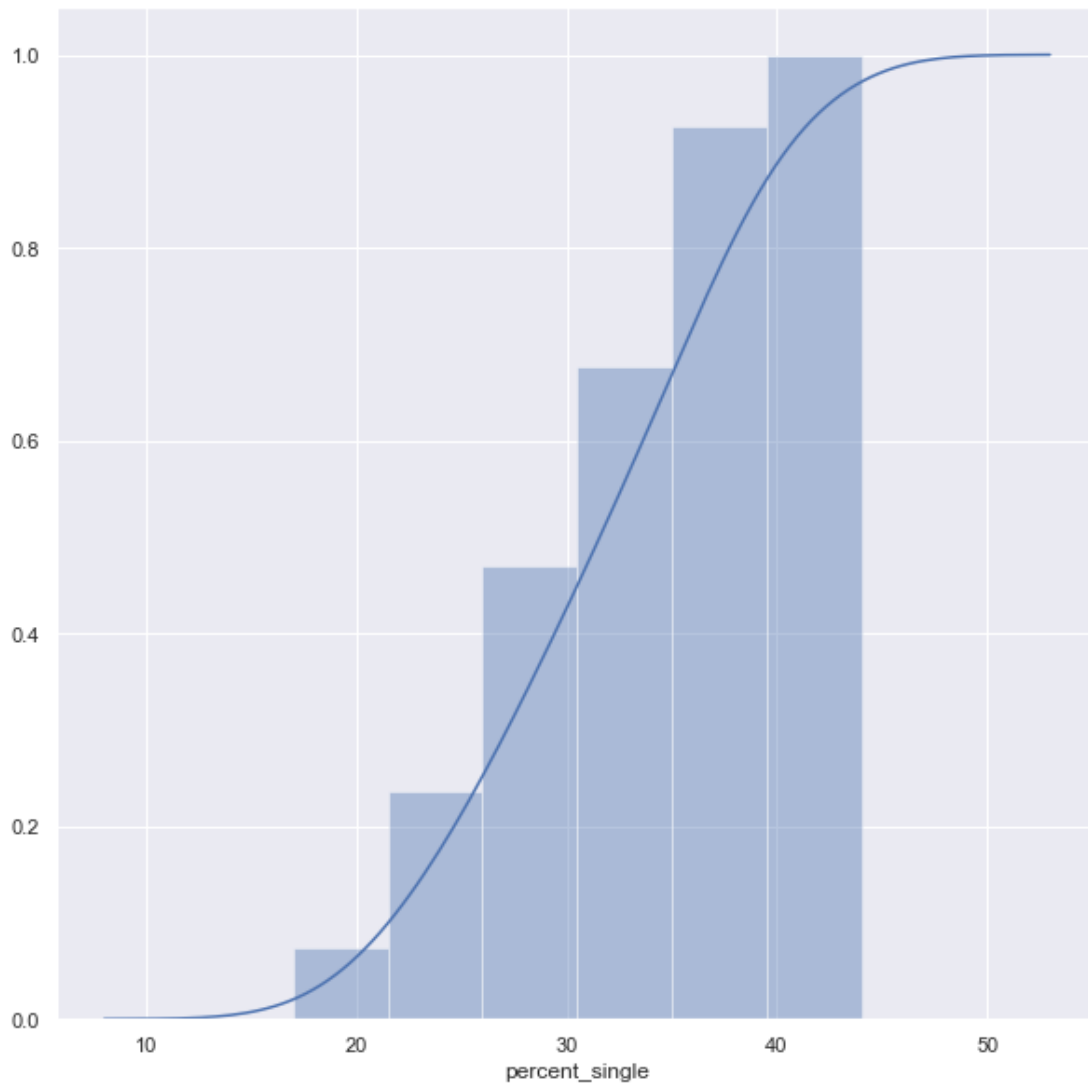
```
[13]: # =====
# Create 1 CDF with one of your variables, using page 41-44 as your guide,
# what does
# this tell you about your variable and how does
# it address the question you are trying to answer (Chapter 4).
# =====

subtitle = " CDF - Single Parent Households Rural"
Cdf_Plot(subtitle,ser1)
subtitle = " CDF - Single Parent Households Urban"
Cdf_Plot(subtitle, ser2)
```

CDF - Single Parent Households Rural



CDF - Single Parent Households Urban



0.3 Is the high school graduation rate and average math/reading scores

0.4 better in urban schools than rural schools?

[ ]:

```
[14]: # =====  
# 2.Is the high school graduation rate and average math/reading scores  
# better in urban schools than rural schools?  
#
```



```
# There is a myth that the best schools are in suburban areas and urban and
# rural schools are lacking in quality education. Do the high school
# graduation rates and 3rd grade reading/math scores reflect this?
# =====
```

```
[15]: # =====
# # Compute Stats
# =====

# Graduation Rate
rg_mean = grad.mean()
rg_var = grad.var()
rg_std = grad.std()
rg_mode = grad.mode()
print(f'Rural - Grad_Rate: Mean = {rg_mean} Var = {rg_var} Std = {rg_std} ␣
      ↪Mode = {rg_mode} \n')

ug_mean = grad1.mean()
ug_var = grad1.var()
ug_std = grad1.std()
ug_mode = grad1.mode()
print(f'Urban - Grad_Rate: Mean = {ug_mean} Var = {ug_var} Std = {ug_std}␣
      ↪Mode = {ug_mode} \n')

# Cohen's d
mean1 = rg_mean
mean2 = ug_mean
var1 = rg_var
ser1 = grad
ser2 = grad1
title = 'Cohens d for High School Graduation Rates Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)

# 3rd Grade Average Reading Scores
rr_mean = read.mean()
rr_var = read.var()
rr_std = read.std()
rr_mode = read.mode()
print('Rural - 3rd Grade Average Reading Scores')
print(f'Mean = {rr_mean} Var = {rr_var} Std = {rr_std} Mode = {rr_mode}\n')

ur_mean = read1.mean()
ur_var = read1.var()
ur_std = read1.std()
ur_mode = read1.mode()
print('Urban - 3rd Grade Average Reading Scores')
print(f'Mean = {ur_mean} Var = {ur_var} Std = {ur_std} Mode = {ur_mode} \n')
```

```

# Cohen's d
mean1 = rr_mean
mean2 = ur_mean
var1 = rr_var
ser1 = read
ser2 = read1
title = 'Cohens d for 3rd Grade Reading Levels Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)

# 3rd Grade Average Math Scores
rm_mean = maths.mean()
rm_var = maths.var()
rm_std = maths.std()
rm_mode = maths.mode()
print('Rural - 3rd Grade Average Math Scores')
print(f'Mean = {rm_mean} Var = {rm_var} Std = {rm_std} Mode = {rm_mode} \n')

um_mean = maths1.mean()
um_var = maths1.var()
um_std = maths1.std()
um_mode = maths1.mode()
print('Urban - 3rd Grade Average Math Scores')
print(f'Mean = {um_mean} Var = {um_var} Std = {um_std} Mode = {um_mode}\n')

# Cohen's d
mean1 = rm_mean
mean2 = um_mean
var1 = rm_var
ser1 = maths
ser2 = maths1
title = 'Cohens d for 3rd Grade Math Levels Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)

```

Rural - Grad\_Rate: Mean = 94.39247311827957 Var = 35.27756466143565 Std = 5.939491953141754 Mode = 0 98  
dtype: int64

Urban - Grad\_Rate: Mean = 91.92647058823529 Var = 18.845258999122024 Std = 4.341112645292912 Mode = 0 98  
dtype: int64

Cohens d for High School Graduation Rates Rural vs Urban = 0.4826872508130733

Rural - 3rd Grade Average Reading Scores  
Mean = 2.7854838709677425 Var = 0.056490845684394046 Std = 0.23767802945243813  
Mode = 0 2.7

dtype: float64

Urban - 3rd Grade Average Reading Scores

Mean = 2.894117647058824 Var = 0.031308165057067586 Std = 0.17694113444043358

Mode = 0 2.9

dtype: float64

Cohens d for 3rd Grade Reading Levels Rural vs Urban = -0.19540097249671604

Rural - 3rd Grade Average Math Scores

Mean = 2.9413978494623647 Var = 0.07822290031967445 Std = 0.2796835717729492

Mode = 0 3.0

dtype: float64

Urban - 3rd Grade Average Math Scores

Mean = 3.0852941176470585 Var = 0.03918349429323968 Std = 0.19794821113927674

Mode = 0 3.0

dtype: float64

Cohens d for 3rd Grade Math Levels Rural vs Urban = -0.2524115783211429

[15]: -0.2524115783211429

[16]: *# Descriptive Statistics*

```
hs_df_rural = rural[['Grad_Rate', 'Math', 'Read']]
hs_df_rural.describe()
```

[16]:

	Grad_Rate	Math	Read
count	186.000000	186.000000	186.000000
mean	94.392473	2.941398	2.785484
std	5.939492	0.279684	0.237678
min	62.000000	2.100000	2.000000
25%	93.000000	2.700000	2.700000
50%	96.000000	2.900000	2.800000
75%	98.000000	3.100000	2.900000
max	100.000000	4.000000	3.400000

[17]:

```
hs_df_urban = urban[['Grad_Rate', 'Math', 'Read']]
hs_df_urban.describe()
```

[17]:

	Grad_Rate	Math	Read
count	68.000000	68.000000	68.000000
mean	91.926471	3.085294	2.894118
std	4.341113	0.197948	0.176941
min	75.000000	2.600000	2.500000

25%	90.000000	3.000000	2.800000
50%	93.000000	3.100000	2.900000
75%	95.000000	3.200000	3.000000
max	97.000000	3.600000	3.400000

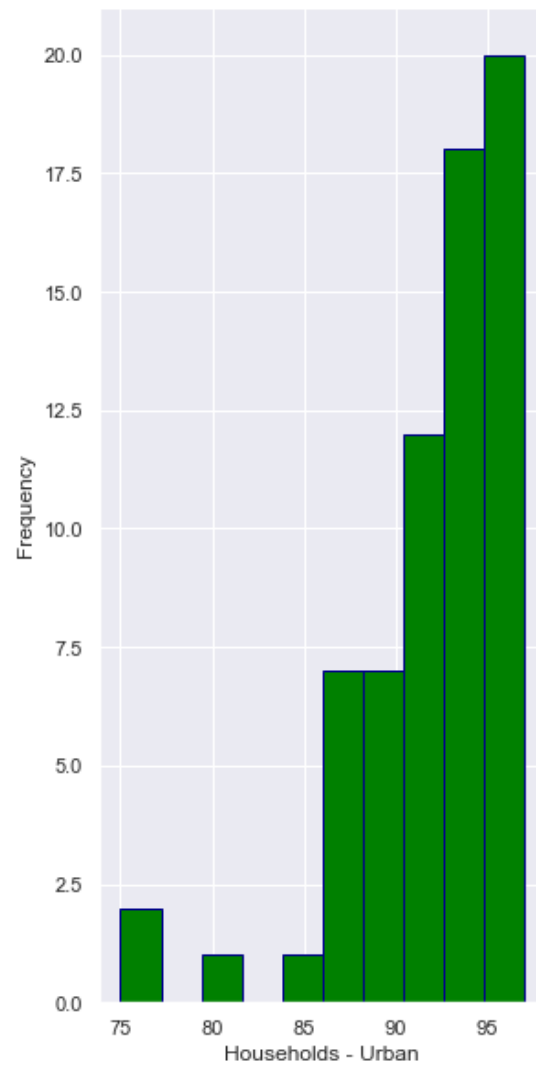
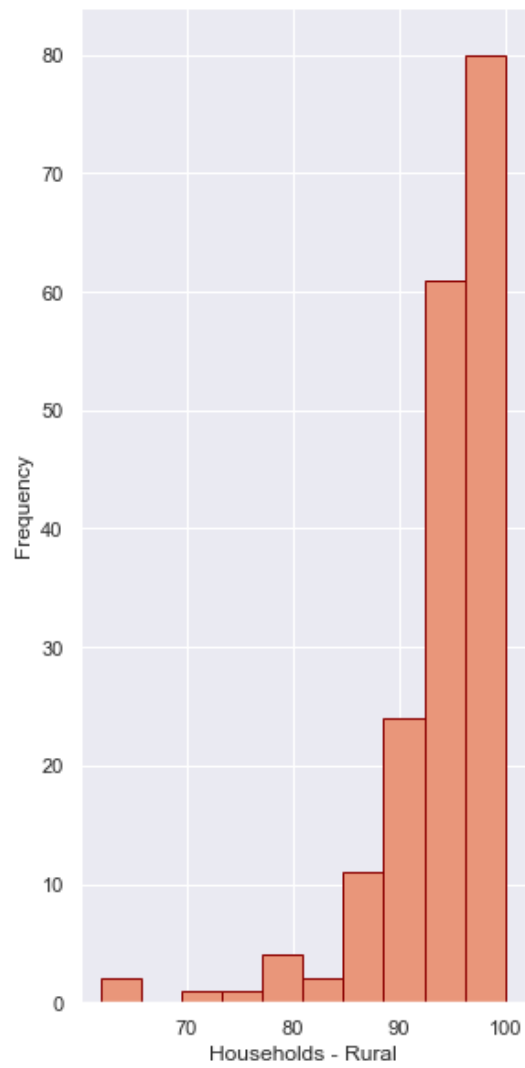
#### 0.4.1 Histograms and PDFs

```
[18]: # =====
# Include a histogram of each of the 5 variables - in your summary and
# analysis, identify any outliers and explain the reasoning for them being
# outliers and how you believe they should be handled (Chapter 2).
# =====
# Histogram for distribution - Placed these histograms and KDEs here because
# ↪ they
# use the raw data not the normalized data. Did this so I could compare the
# actual vlues and not the normalized values.
# =====

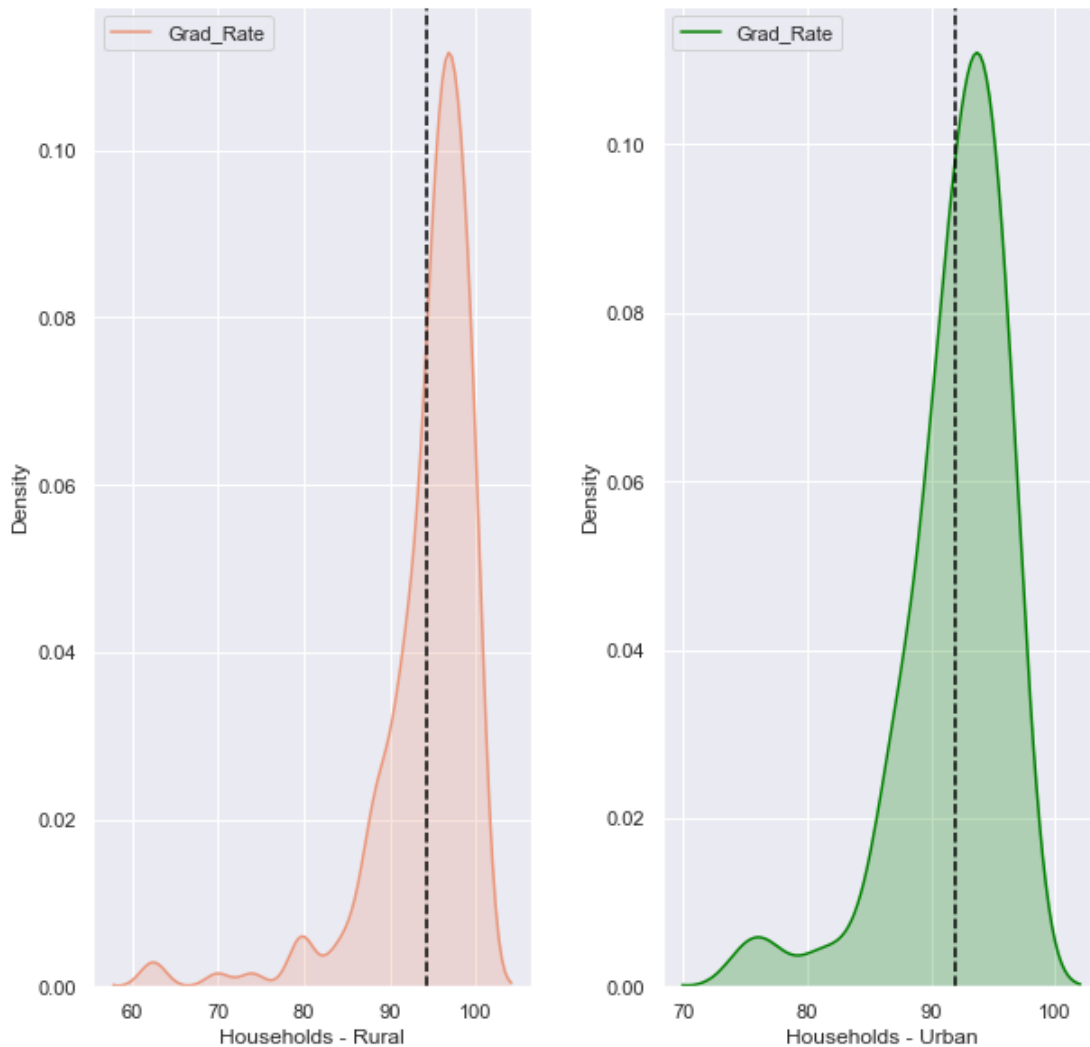
# =====
# # Graduation Rates Rural vs Urban
# =====
subtitle = "High School Graduation Rates for Rural vs Urban"
xlab1 = "Households - Rural"
xlab2 = "Households - Urban"
mean1 = rg_mean
mean2 = ug_mean
ser1 = grad
ser2 = grad1
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDF
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

## High School Graduation Rates for Rural vs Urban



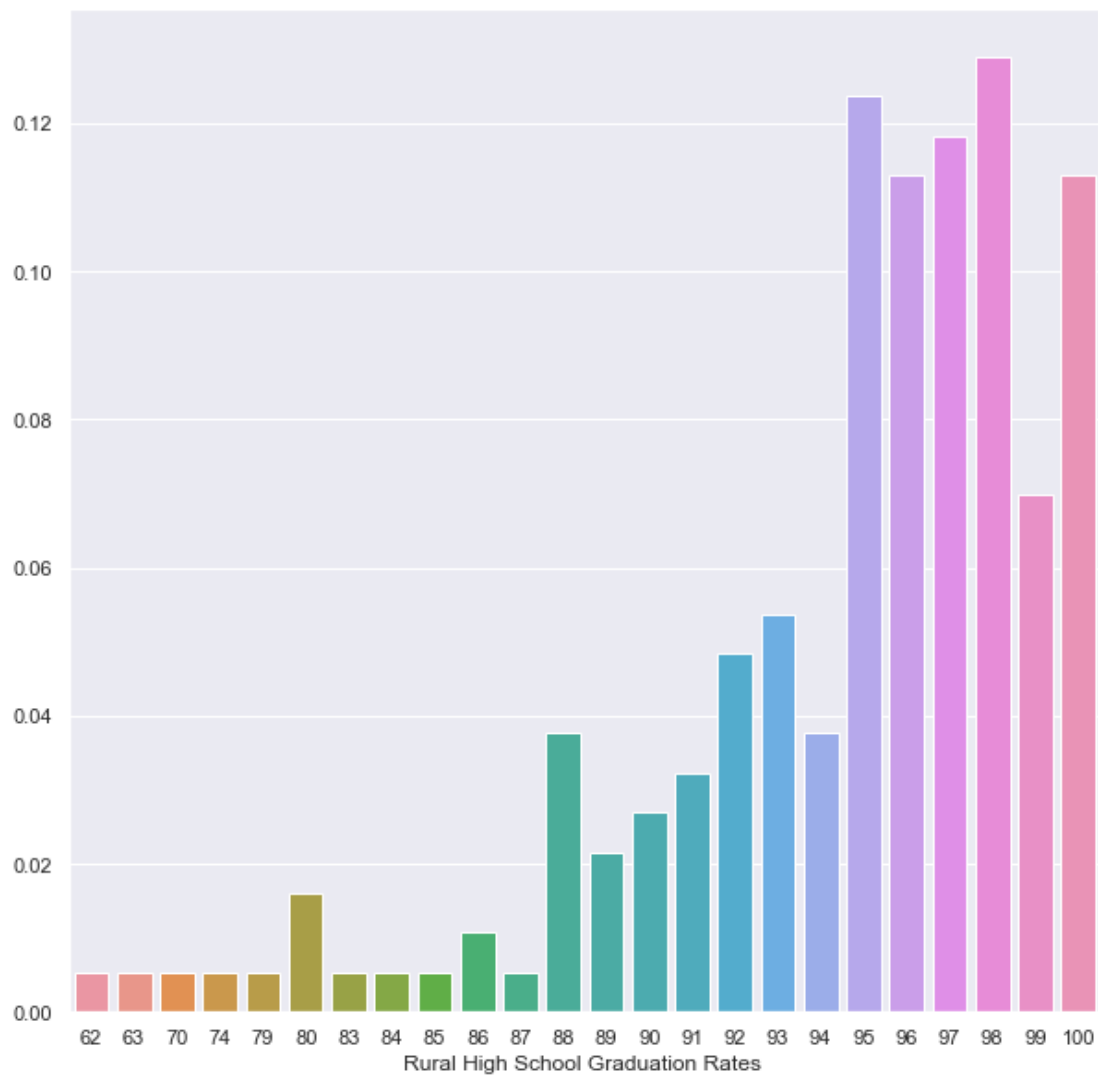
## High School Graduation Rates for Rural vs Urban



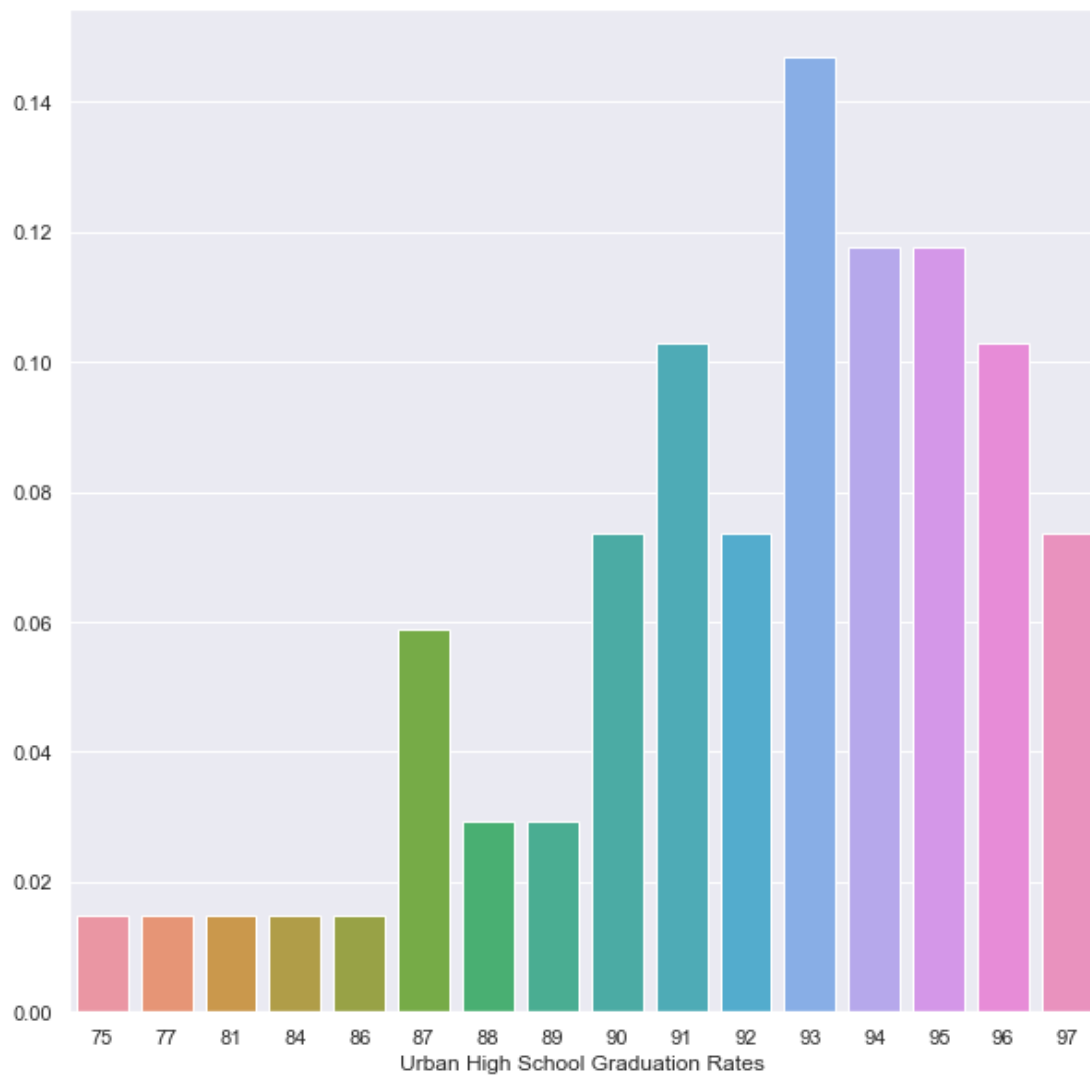
```
[19]: # =====
# Using pg. 29 of your text as an example, compare two scenarios in your data.
# using a
# PMF. Reminder, this isn't comparing two variables against each other
# - it is the same variable, but a different scenario. Almost like a filter.
# The example in the book is first babies compared to all other babies, it
# is still the same variable, but breaking the data out based on criteria
# we are exploring (Chapter 3).
# =====
```

```
[20]: subtitle = 'PMF Plot for High School Graduation Rates - Rural'
data = grad
xlab = 'Rural High School Graduation Rates '
Pmf_plot(subtitle, data, xlab)
subtitle = 'PMF Plot for High School Graduation Rates - Urban'
data = grad1
xlab = 'Urban High School Graduation Rates '
Pmf_plot(subtitle, data, xlab)
```

**PMF Plot for High School Graduation Rates - Rural**



PMF Plot for High School Graduation Rates - Urban



```
[21]: # =====
# # Histograms and PDFs 3rd grade Reading levels
# =====
subtitle = "3rd Grade Reading Levels for Rural vs Urban"
xlab1 = "Households - Rural"
xlab2 = "Households - Urban"
mean1 = rr_mean
mean2 = ur_mean
ser1 = read
ser2 = read1
```

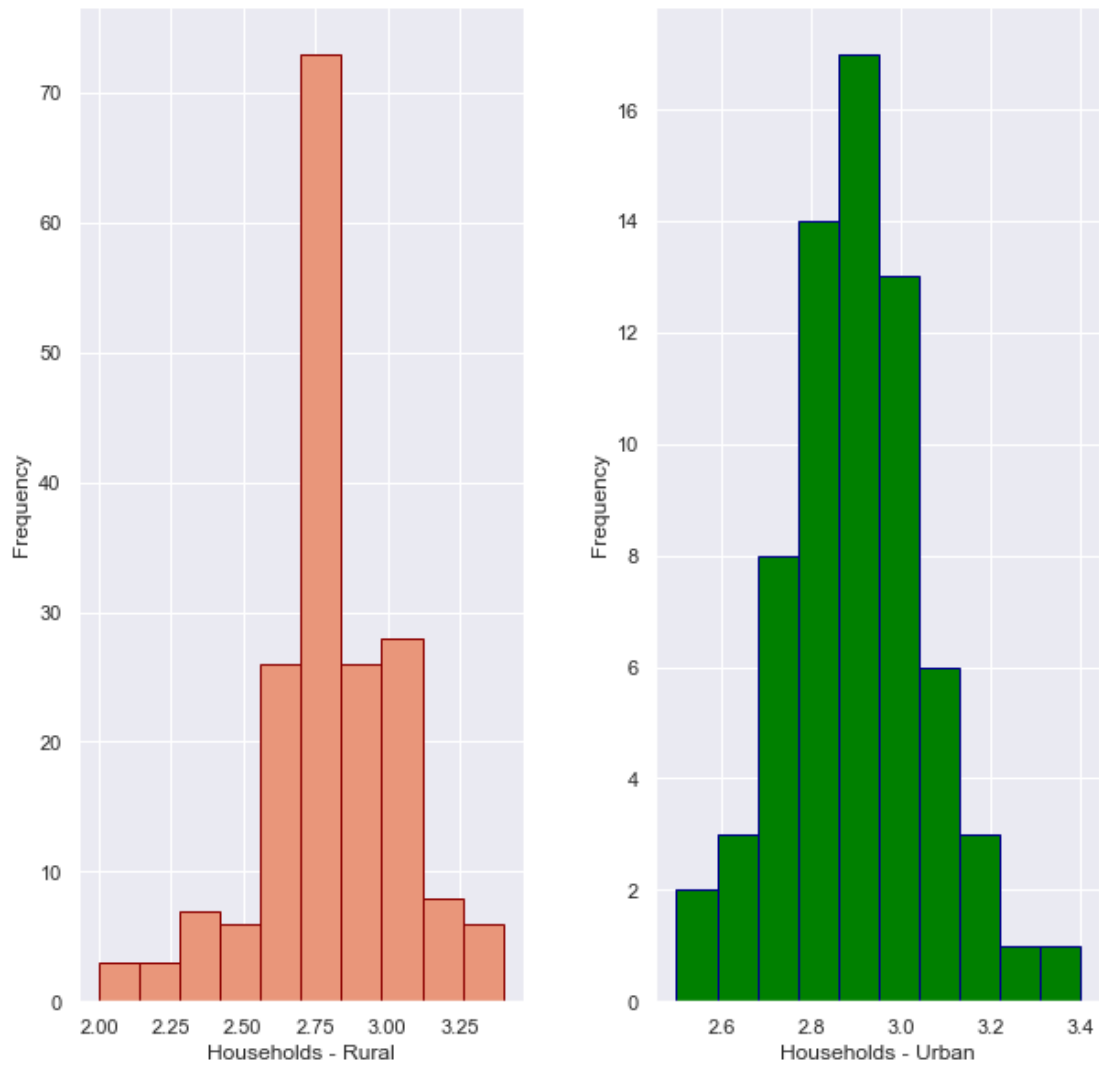


```
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)
```

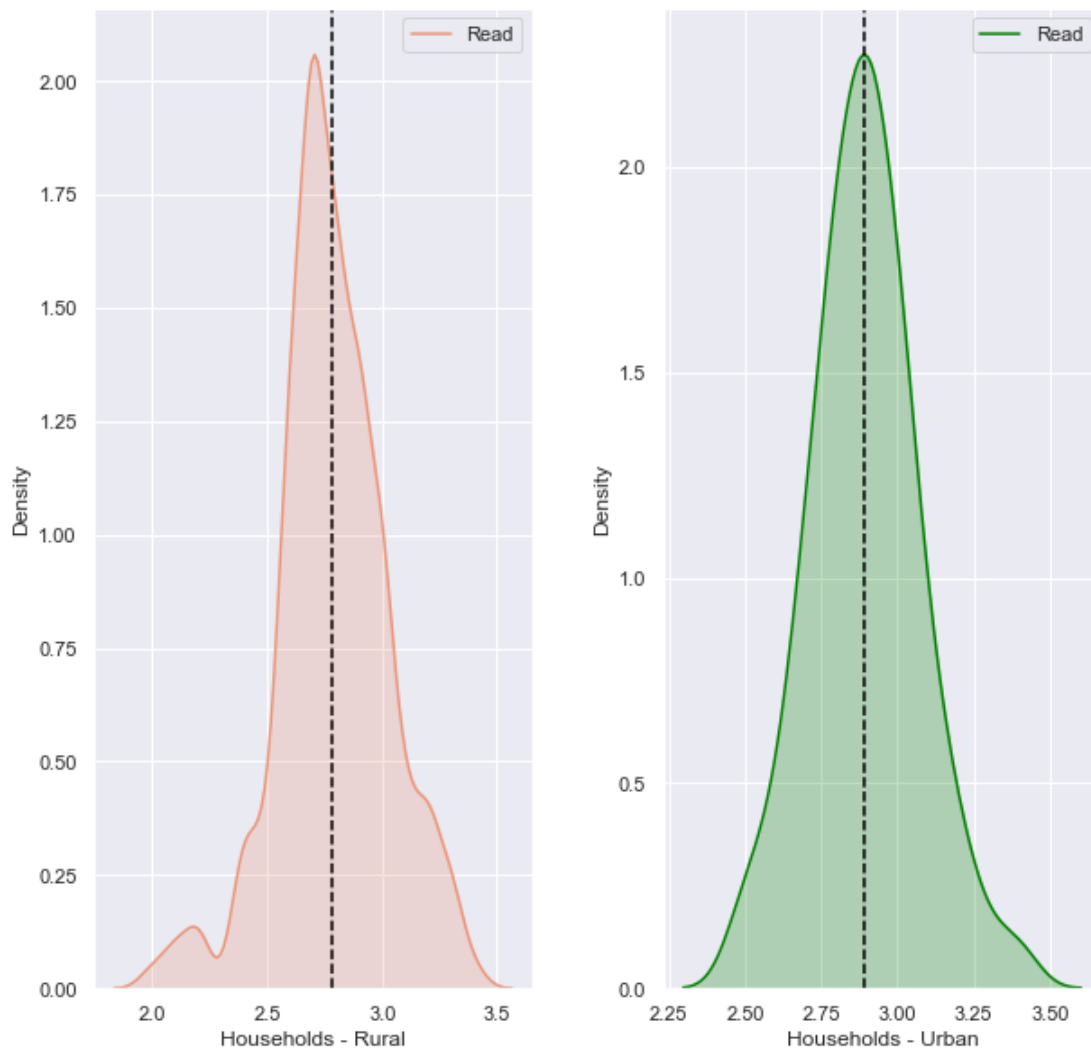
```
# KDE/PDF
```

```
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

**3rd Grade Reading Levels for Rural vs Urban**

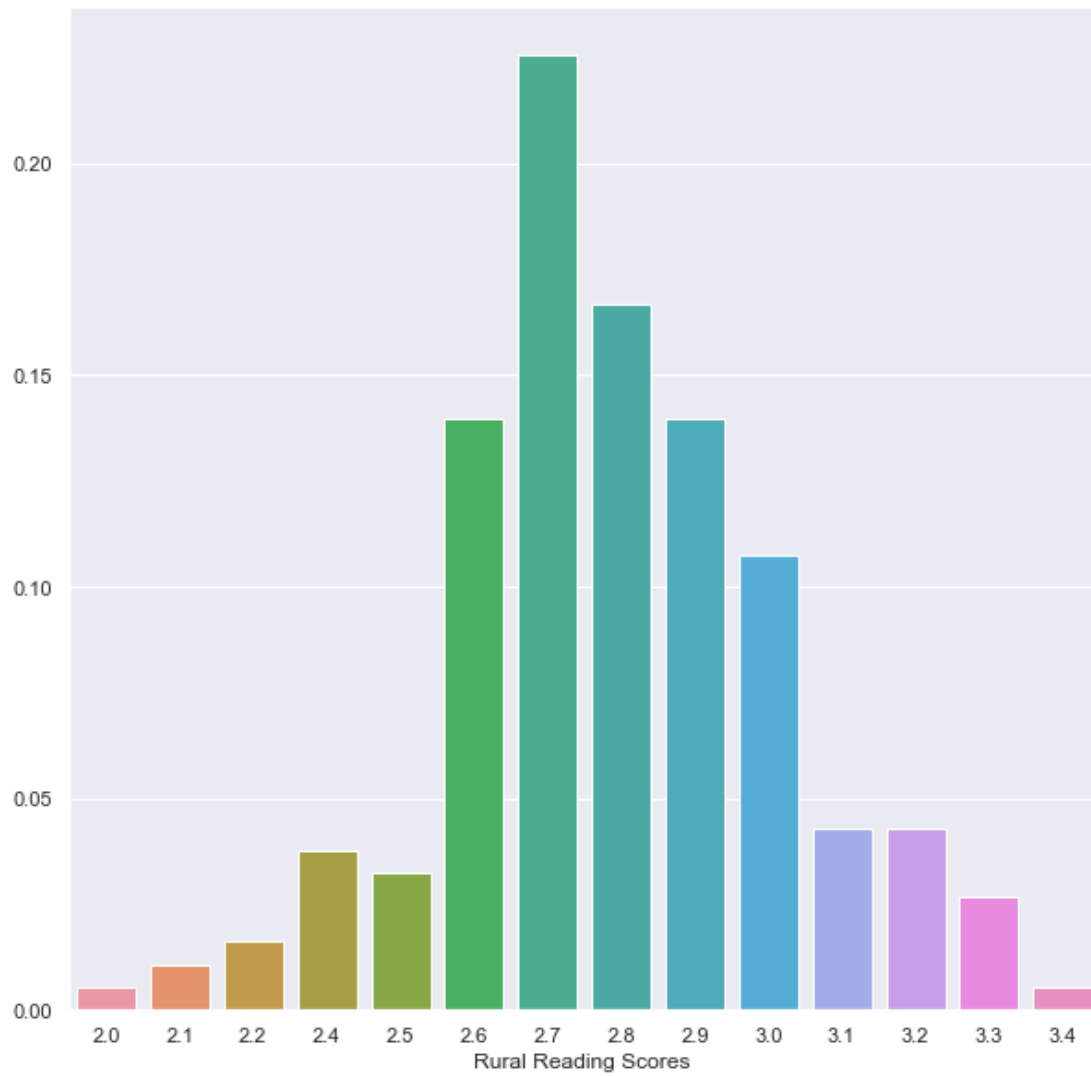


### 3rd Grade Reading Levels for Rural vs Urban

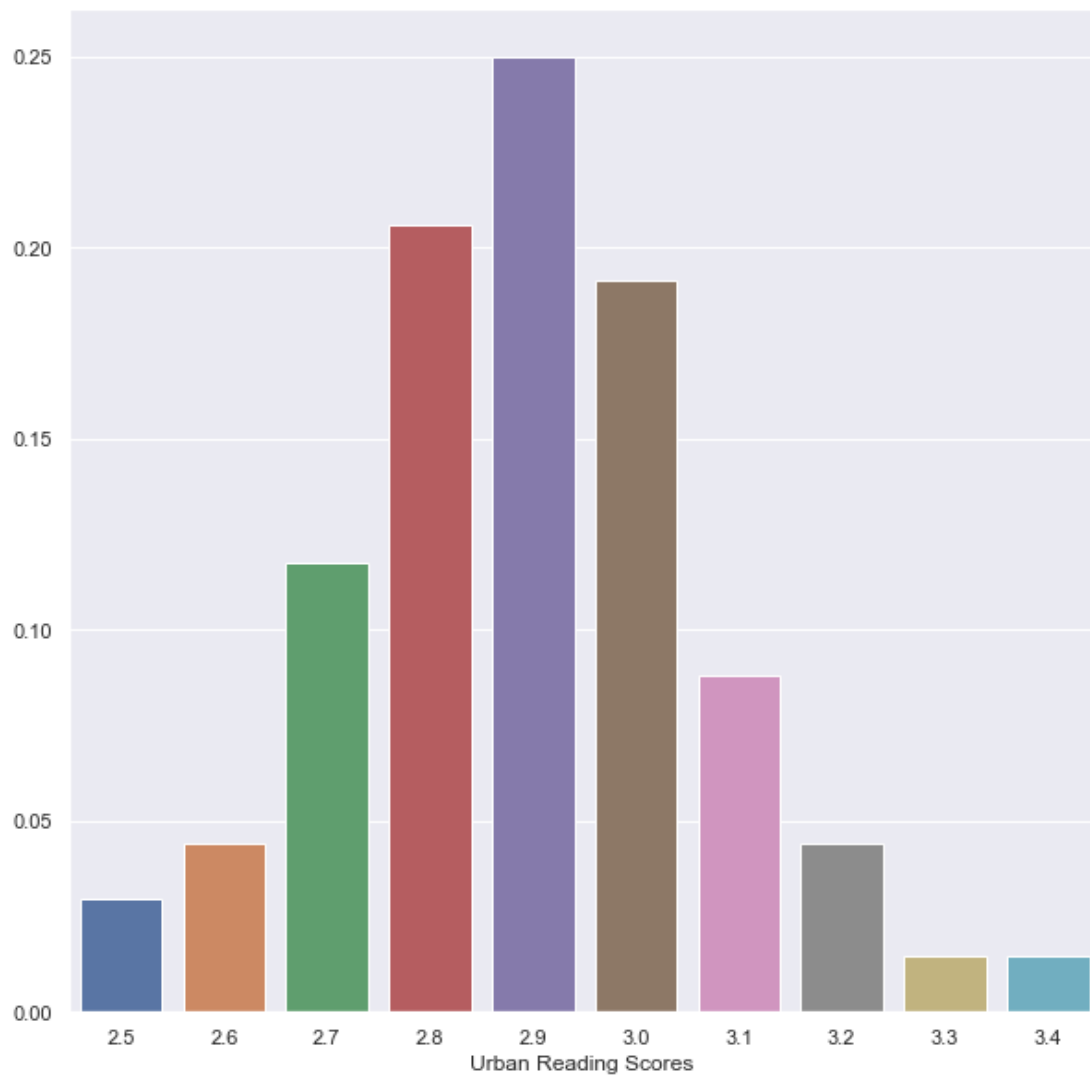


```
[22]: subtitle = 'PMF Plot for 3rd Grade Average Reading Scores - Rural'
data = read
xlab = 'Rural Reading Scores '
Pmf_plot(subtitle, data, xlab)
subtitle = 'PMF Plot for 3rd Grade Average Reading Scores - Urban'
data = read1
xlab = 'Urban Reading Scores '
Pmf_plot(subtitle, data, xlab)
```

PMF Plot for 3rd Grade Average Reading Scores - Rural



PMF Plot for 3rd Grade Average Reading Scores - Urban



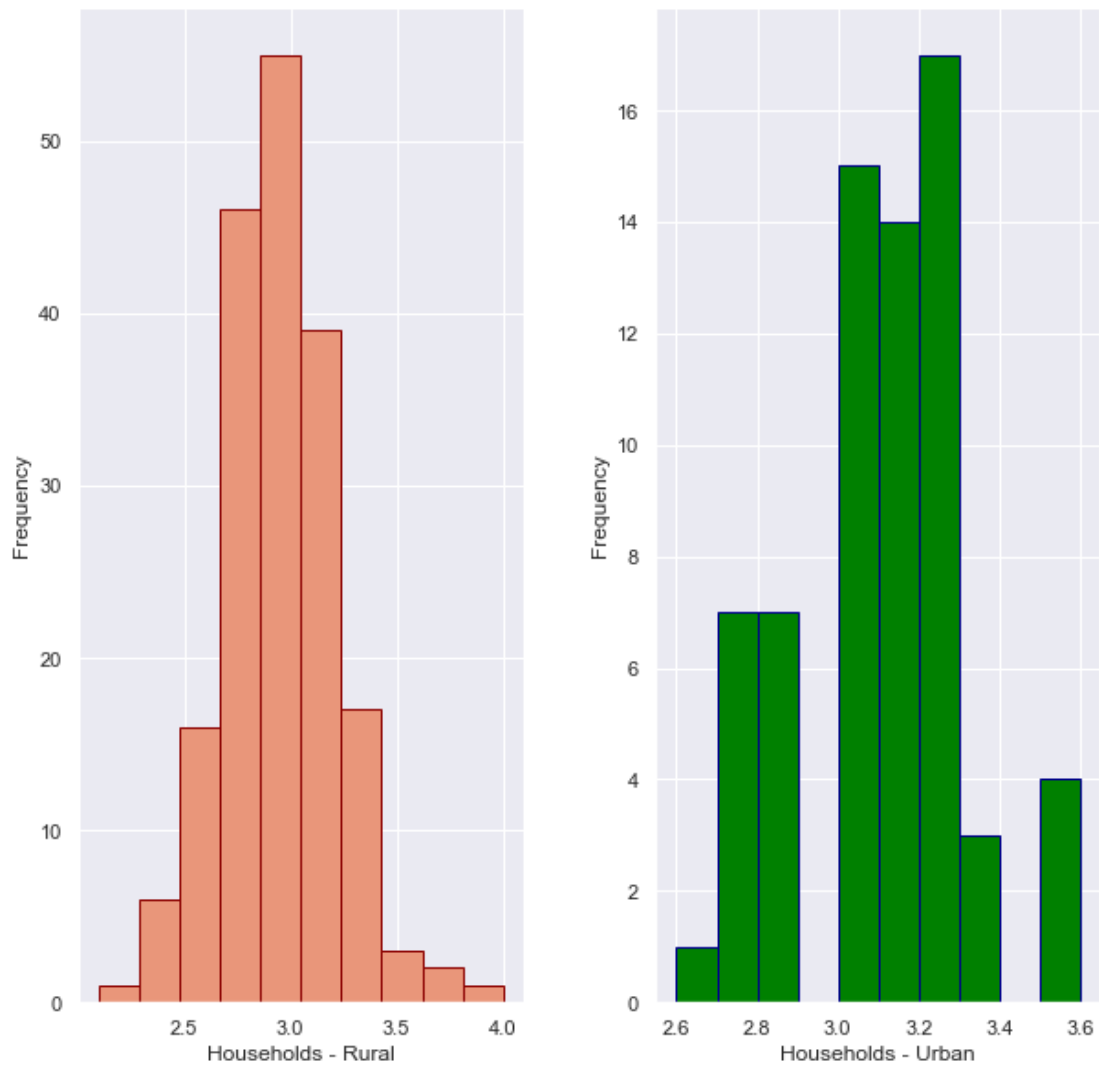
```
[23]: # =====
# # Histograms and PDFs 3rd grade Math levels
# =====
subtitle = "3rd Grade Math Levels for Rural vs Urban"
xlab1 = "Households - Rural"
xlab2 = "Households - Urban"
mean1 = rm_mean
mean2 = ur_mean
ser1 = maths
ser2 = maths1
```

```
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)
```

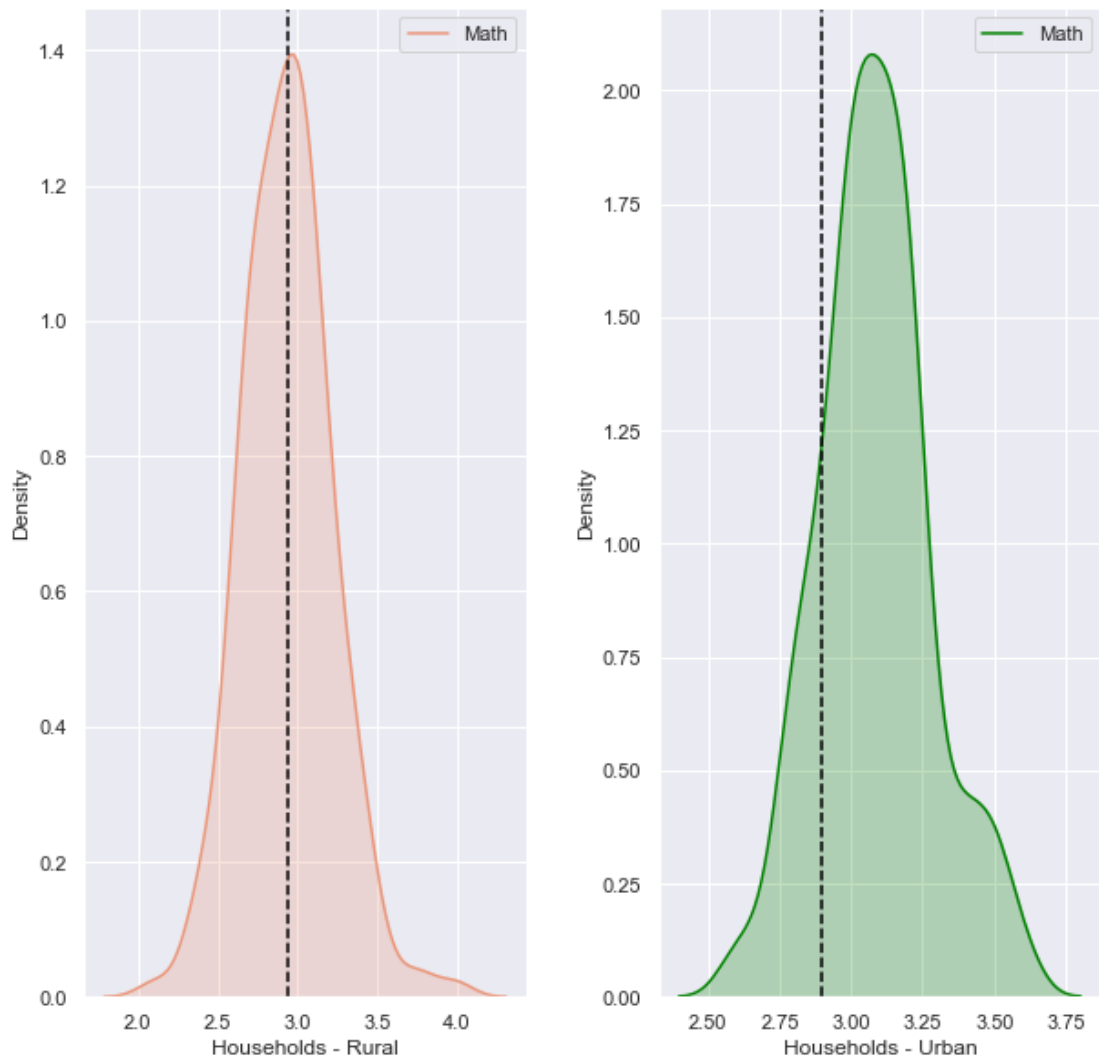
```
# KDE/PDFs
```

```
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

### 3rd Grade Math Levels for Rural vs Urban

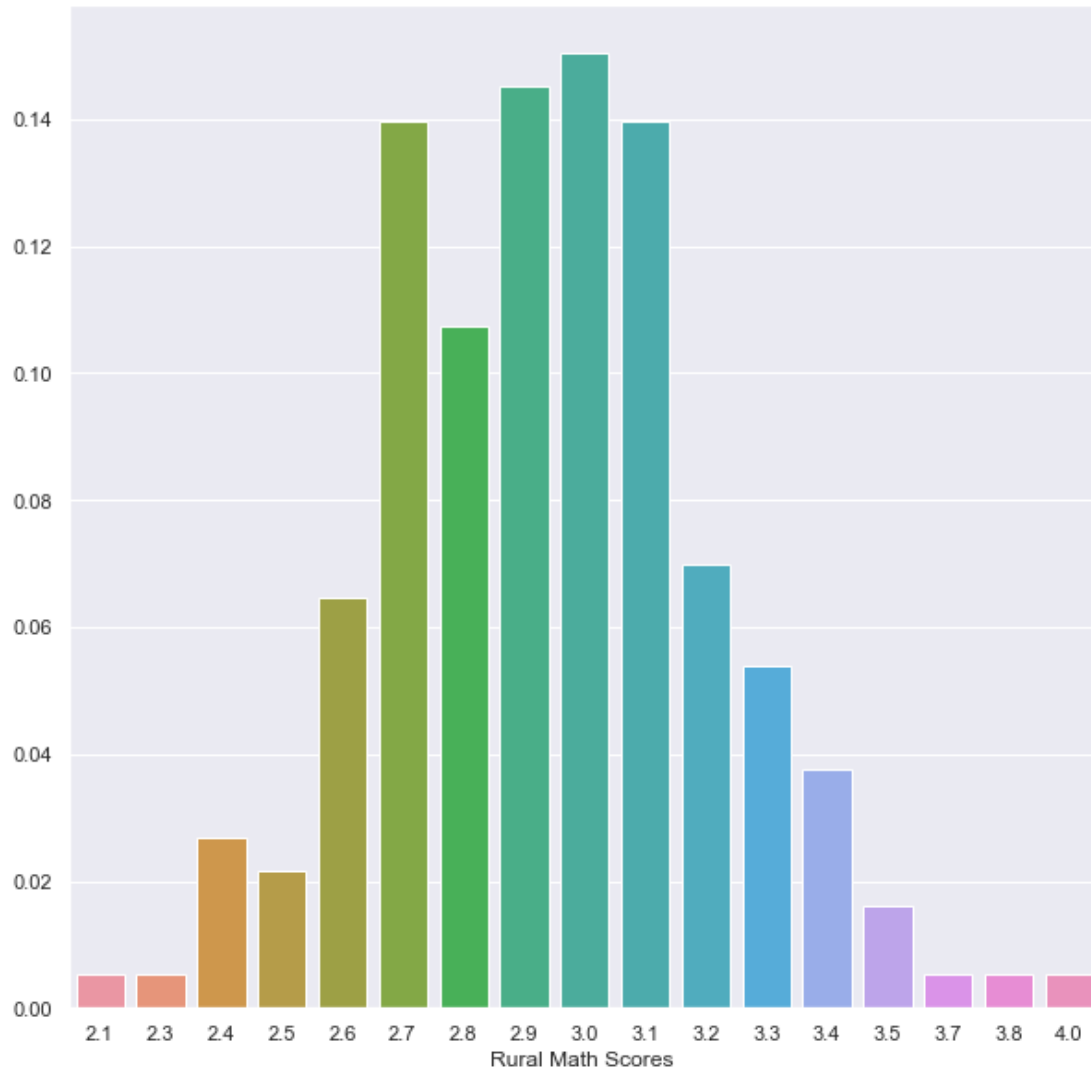


### 3rd Grade Math Levels for Rural vs Urban

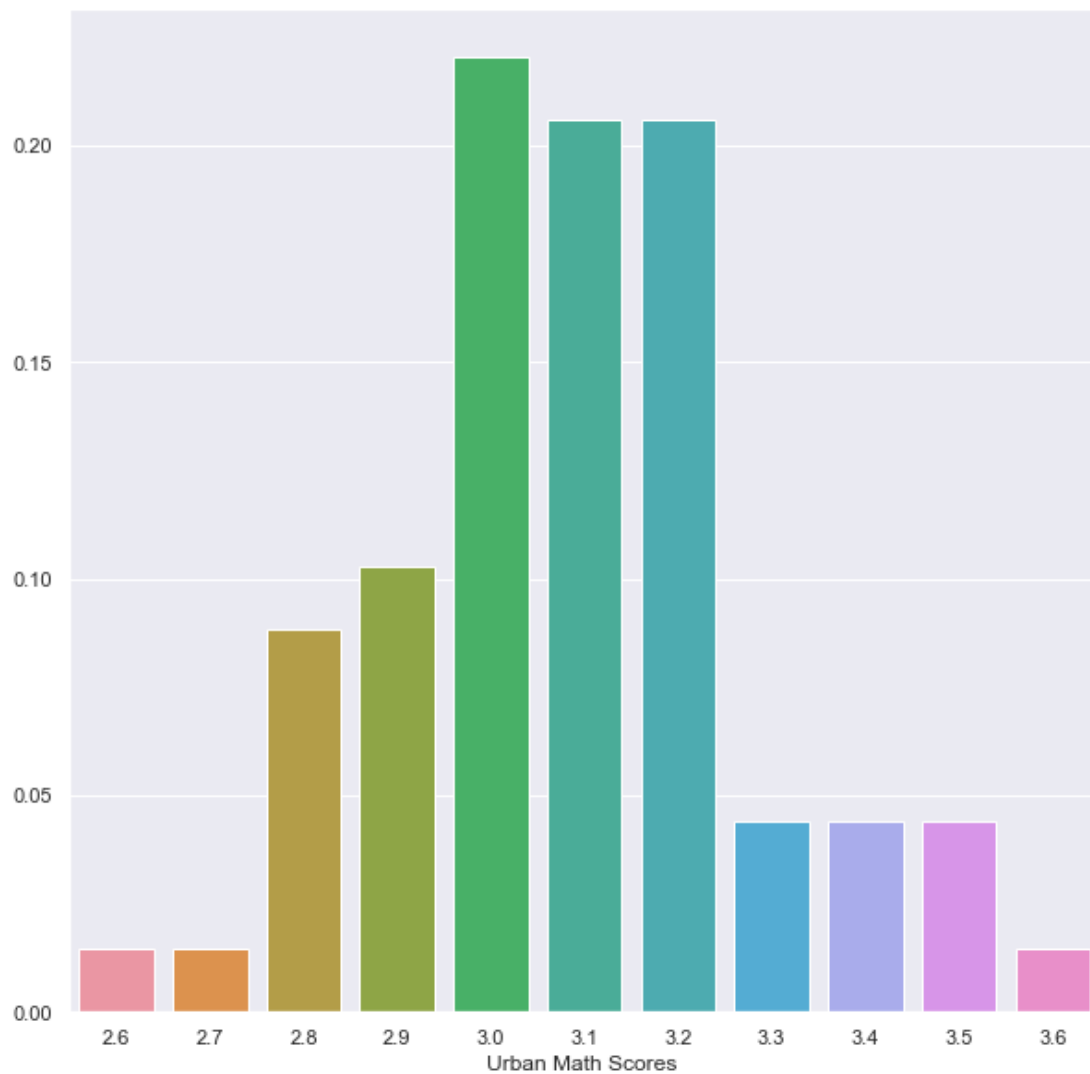


```
[24]: subtitle = 'PMF Plot for Math Scores - Rural'
data = maths
xlab = 'Rural Math Scores '
Pmf_plot(subtitle, data, xlab)
subtitle = 'PMF Plot for Math Scores - Urban'
data = maths1
xlab = 'Urban Math Scores '
Pmf_plot(subtitle, data, xlab)
```

PMF Plot for Math Scores - Rural



PMF Plot for Math Scores - Urban



```
[25]: # =====  
# Conduct Correlation Tests - Chapter 9  
# =====  
# what if any is the correlation between 3rd grade levels and high school  
# graduation  
# need to normalize the data  
# normalize the data sklearn  
  
scaler = MinMaxScaler()  
rural_school = rural[['Read', 'Math', 'Grad_Rate']].copy()
```



```

urban_school = urban[['Read', 'Math', 'Grad_Rate']].copy()

rural_school[['Read', 'Math', 'Grad_Rate']] = scaler.
    ↪fit_transform(rural_school[['Read',\
        'Math', 'Grad_Rate']])

urban_school[['Read', 'Math', 'Grad_Rate']] = scaler.
    ↪fit_transform(urban_school[['Read',\
        'Math', 'Grad_Rate']])

#Pearson
corr_rural1 = rural_school.corr(method="pearson")
print("Pearson correlation coefficient Rural :")
print(corr_rural1, "\n")

corr_urban1 = urban_school.corr(method="pearson")
print("Pearson correlation coefficient Urban:")
print(corr_urban1, "\n")

#Spearman
corr_rural2 = rural_school.corr(method="spearman")
print("Spearman correlation coefficient Rural :")
print(corr_rural2, "\n")

corr_urban2 = urban_school.corr(method="spearman")
print("Spearman correlation coefficient Urban :")
print(corr_urban2, "\n")

#Kendall's Tau
corr_rural3 = rural_school.corr(method="kendall")
print("Kendall Tau correlation coefficient Rural :")
print(corr_rural3, "\n")

corr_urban3 = urban_school.corr(method="kendall")
print("Kendall Tau correlation coefficient Urban:")
print(corr_urban3, "\n")

```

Pearson correlation coefficient Rural :

	Read	Math	Grad_Rate
Read	1.000000	0.739300	0.192063
Math	0.739300	1.000000	0.144729
Grad_Rate	0.192063	0.144729	1.000000

Pearson correlation coefficient Urban:

	Read	Math	Grad_Rate
Read	1.000000	0.875327	0.424968
Math	0.875327	1.000000	0.377365

```
Grad_Rate  0.424968  0.377365  1.000000
```

Spearman correlation coefficient Rural :

	Read	Math	Grad_Rate
Read	1.000000	0.725848	0.278714
Math	0.725848	1.000000	0.173201
Grad_Rate	0.278714	0.173201	1.000000

Spearman correlation coefficient Urban :

	Read	Math	Grad_Rate
Read	1.000000	0.838901	0.438993
Math	0.838901	1.000000	0.317306
Grad_Rate	0.438993	0.317306	1.000000

Kendall Tau correlation coefficient Rural :

	Read	Math	Grad_Rate
Read	1.000000	0.602315	0.211071
Math	0.602315	1.000000	0.134194
Grad_Rate	0.211071	0.134194	1.000000

Kendall Tau correlation coefficient Urban:

	Read	Math	Grad_Rate
Read	1.000000	0.745942	0.330658
Math	0.745942	1.000000	0.246029
Grad_Rate	0.330658	0.246029	1.000000

```
/Users/corosco/anaconda3/lib/python3.7/site-  
packages/sklearn/preprocessing/data.py:334: DataConversionWarning: Data with  
input dtype int64, float64 were all converted to float64 by MinMaxScaler.
```

```
    return self.partial_fit(X, y)
```

```
/Users/corosco/anaconda3/lib/python3.7/site-  
packages/sklearn/preprocessing/data.py:334: DataConversionWarning: Data with  
input dtype int64, float64 were all converted to float64 by MinMaxScaler.
```

```
    return self.partial_fit(X, y)
```

```
[26]: # Linear Relationship  
       subtitle = 'Graduation Rates and Reading Scores - Rural'  
       x_val = "Grad_Rate"  
       y_val = 'Math'  
       data = rural_school  
       sns.lmplot(x_val, y_val, data)  
  
       # Linear Relationship  
       subtitle = 'Graduation Rates and Reading Scores - Urban'  
       x_val = "Grad_Rate"  
       y_val = 'Math'  
       data = urban_school
```

```

sns.lmplot(x_val, y_val, data)

# Linear Relationship
subtitle = 'Graduation Rates and Reading Scores - Rural'
x_val = "Grad_Rate"
y_val = 'Read'
data = rural_school
sns.lmplot(x_val, y_val, data)

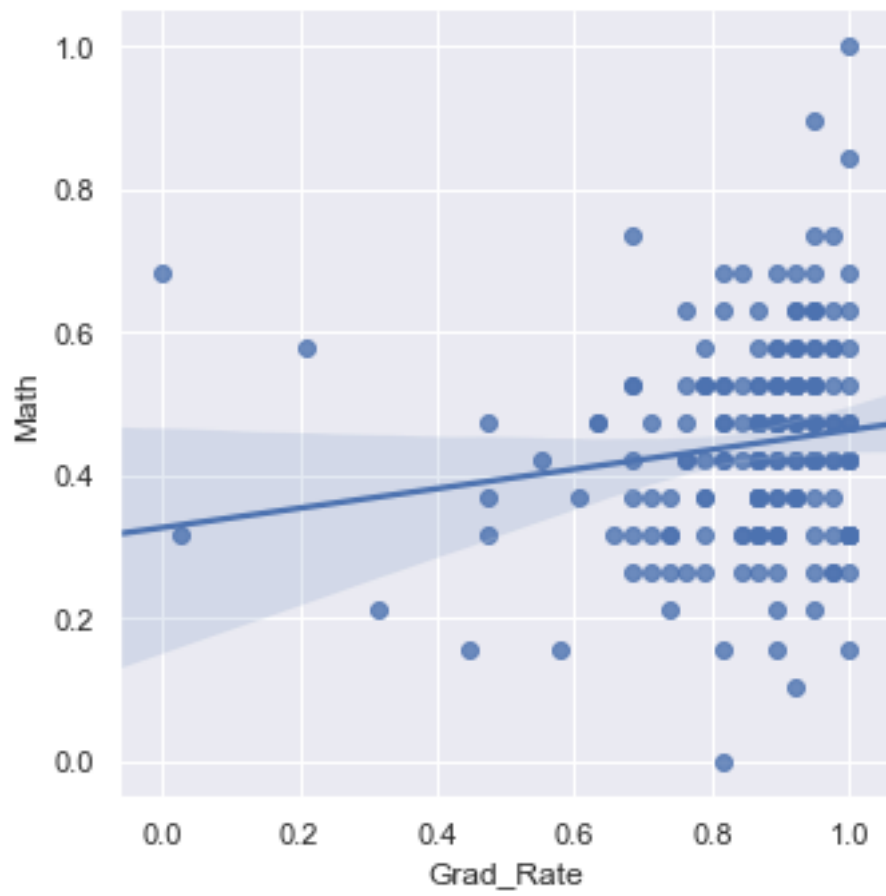
# Linear Relationship
subtitle = 'Graduation Rates and Reading Scores - Urban'
x_val = "Grad_Rate"
y_val = 'Read'
data = urban_school
sns.lmplot(x_val, y_val, data)

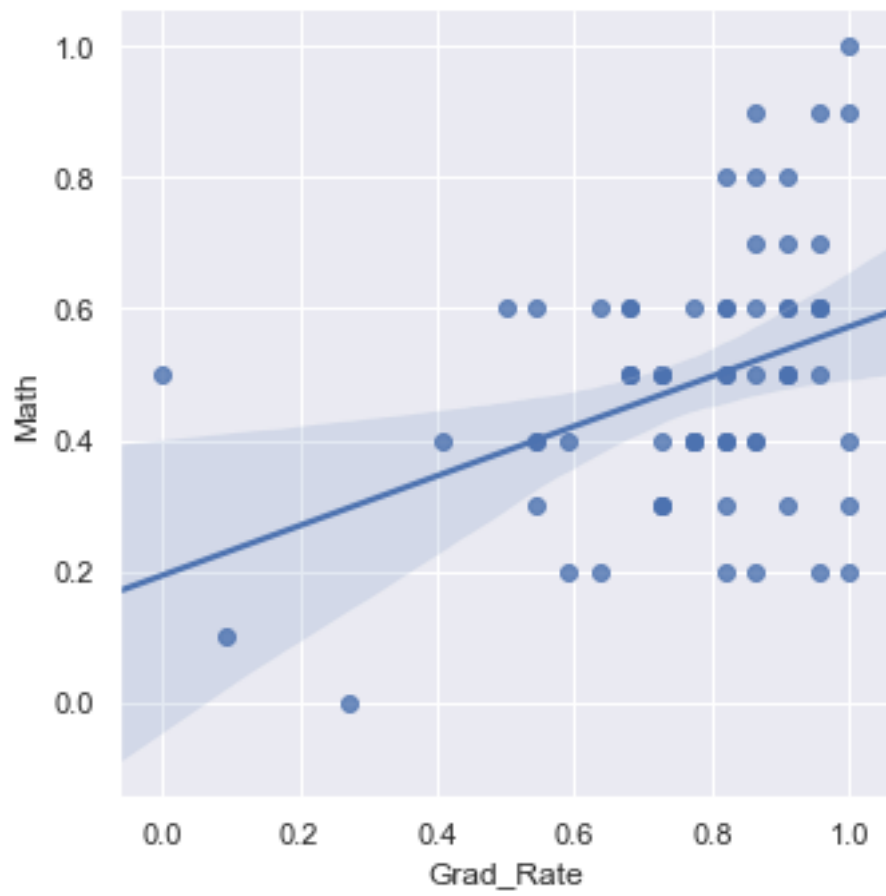
# Math and Reading
subtitle = 'Math and Reading Scores - Rural'
x_val = "Math"
y_val = 'Read'
data = rural_school
sns.lmplot(x_val, y_val, data)

# Math and Reading
subtitle = 'Math and Reading Scores - Urban'
x_val = "Math"
y_val = 'Read'
data = urban_school
sns.lmplot(x_val, y_val, data)

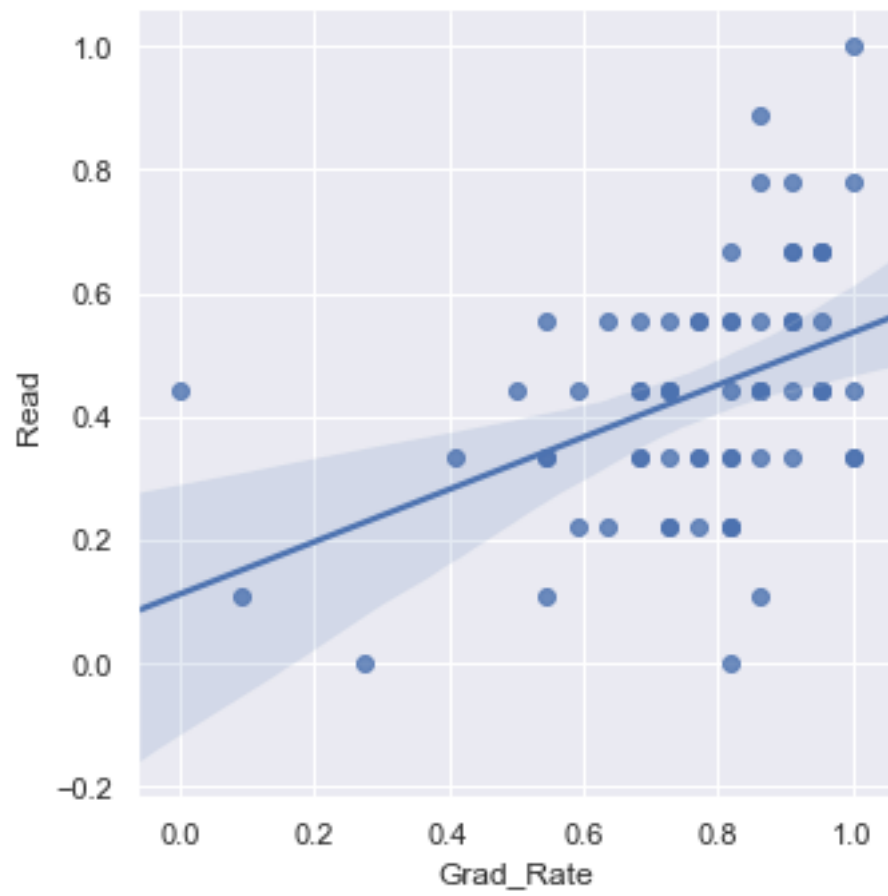
```

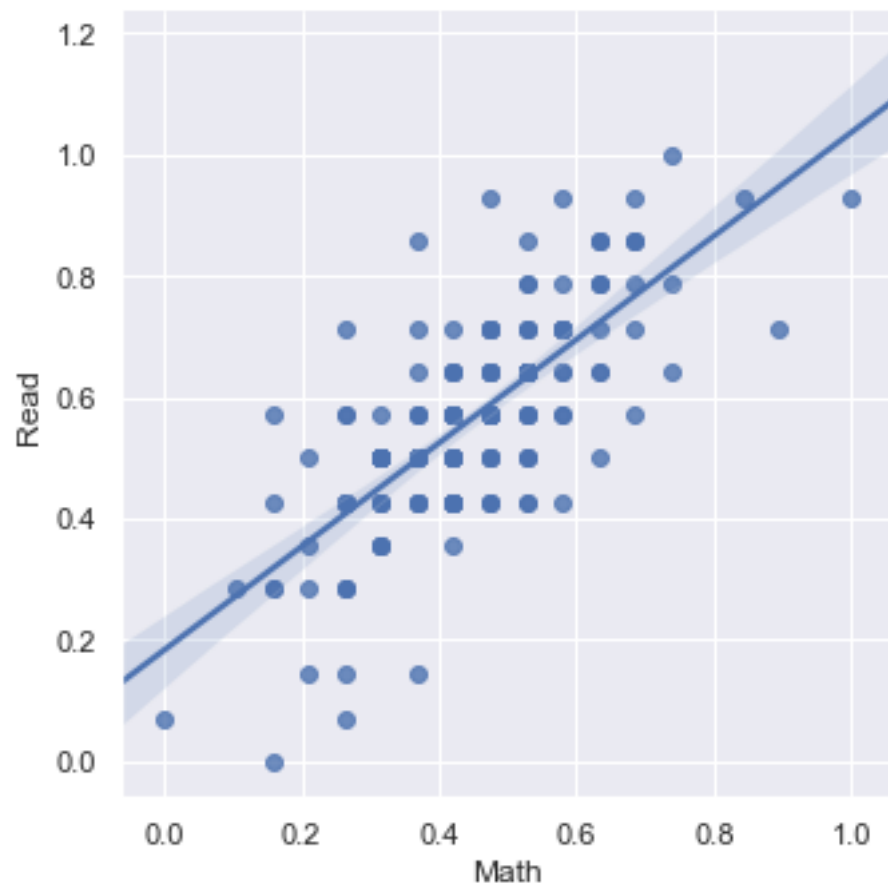
[26]: <seaborn.axisgrid.FacetGrid at 0x123cde5c0>



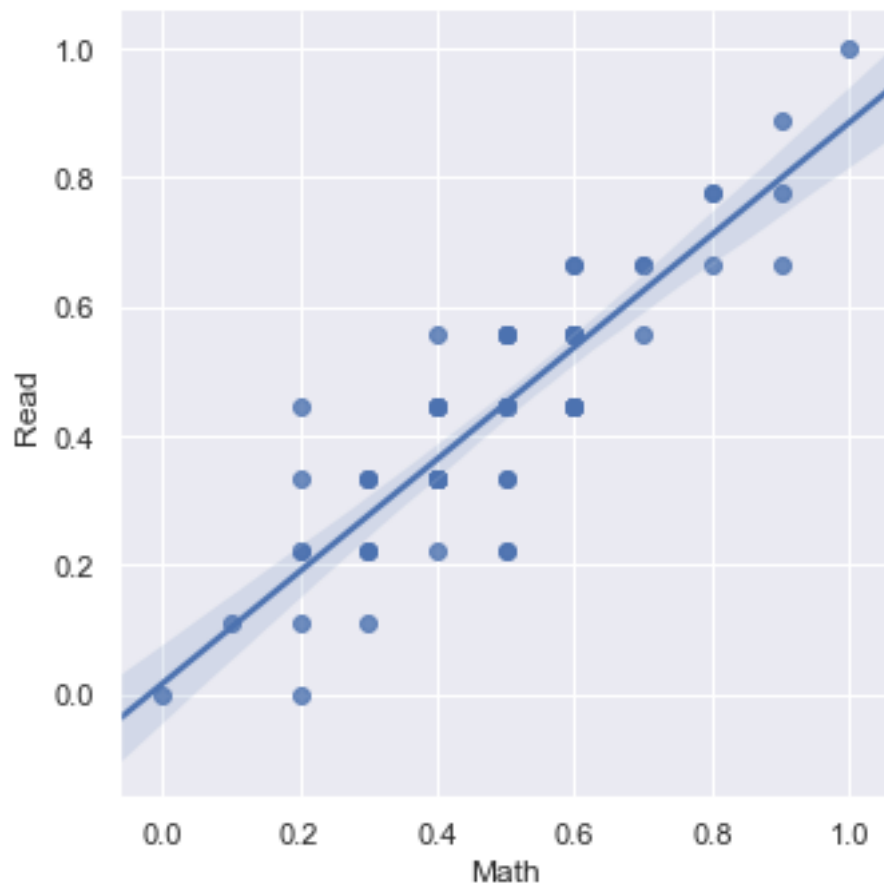












[27]: `#Include a histogram of each of the 5 variables - in your summary and analysis,␣  
 ↪identify  
 #any outliers and explain the reasoning for them being outliers and how you␣  
 ↪believe they  
 #should be handled (Chapter 2).`

- 0.5 Is there a correlation between life expectancy and poor health,
- 0.6 lack of sleep, no insurance, housing costs, access to food, and
- 0.7 availability of doctors?
- 0.8 Is there a correlation between poor health and availability of doctors,
- 0.9 lack of sleep, no insurance, housing costs, and access to food?

[28]: `# =====  
 # 3. Is there a correlation between life expectancy and poor health,  
 # lack of sleep, no insurance, housing costs, access to food, and  
 # availability of doctors?`

```
#
# 4. Is there a correlation between poor health and availability of doctors,
# lack of sleep, no insurance, housing costs, and access to food?
# =====
```

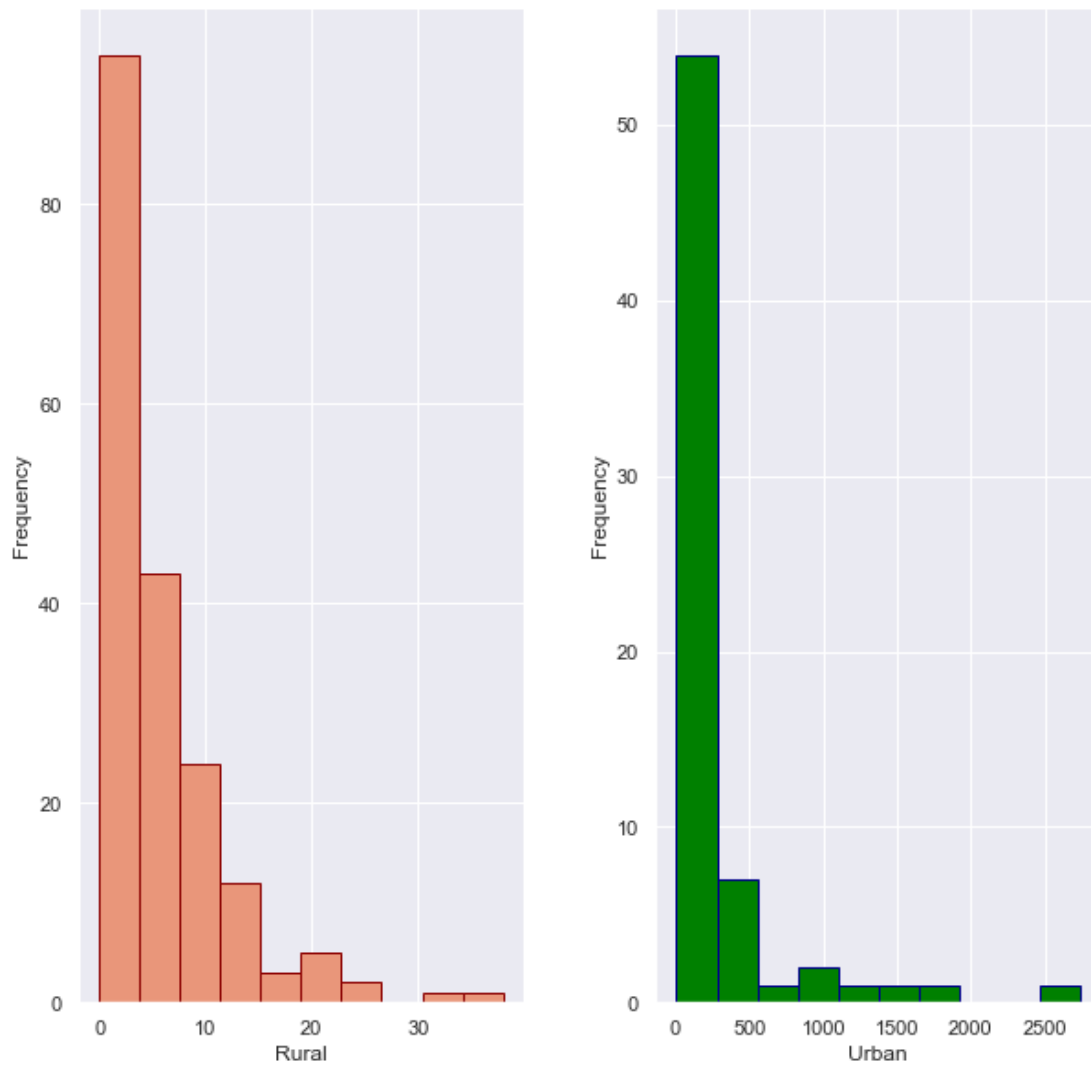
### 0.9.1 Histograms and PDFs

```
[29]: # Histogram and PDFs Physicians rural vs urban
# =====
# plot side by side for comparison
subtitle = "Number of Physicians per County Rural vs Urban"
xlab1 = "Rural"
xlab2 = "Urban"
ser1 = rural['Physicians']

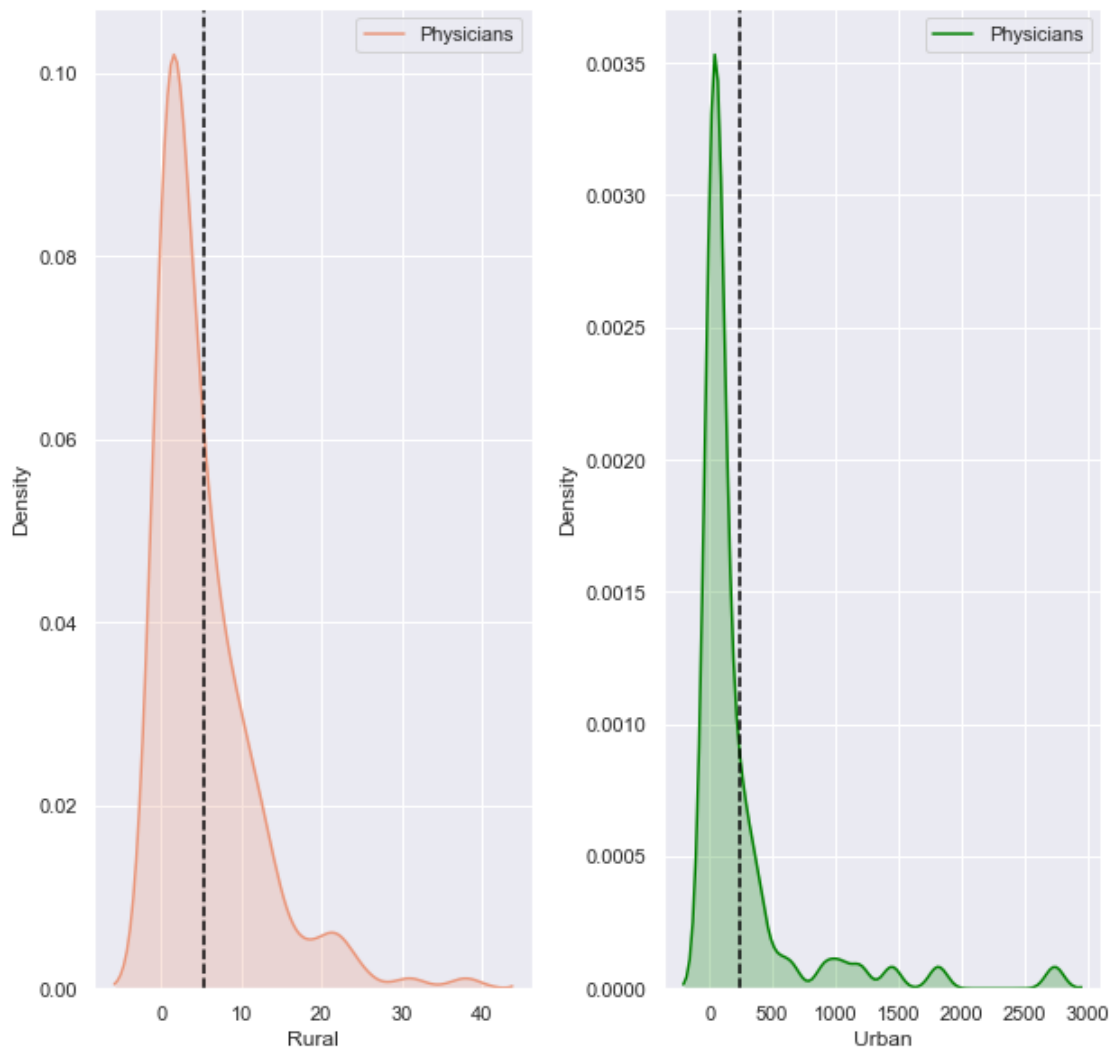
ser2 = urban.Physicians
mean1 = rural.Physicians.mean()
mean2 = urban.Physicians.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

Number of Physicians per County Rural vs Urban



### Number of Physicians per County Rural vs Urban



```
[30]: # Cohen's d while I have the mean
mean1 = rural.Physicians.var()
mean2 = urban.Physicians.var()
ser1 = rural.Physicians
ser2 = urban.Physicians
title = 'Cohens d for Physicians Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

Cohens d for Physicians Rural vs Urban = -375877.59850149235

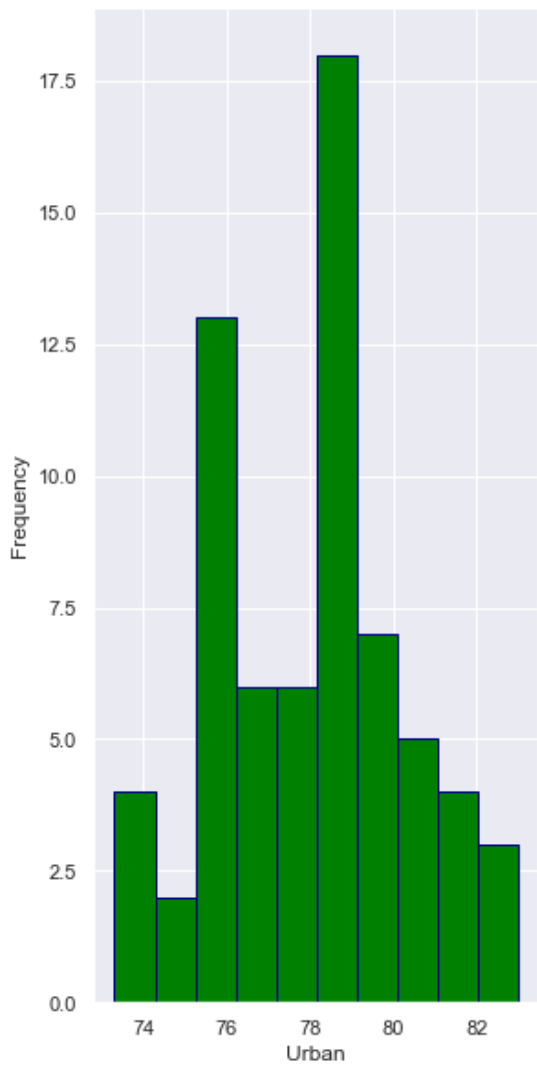
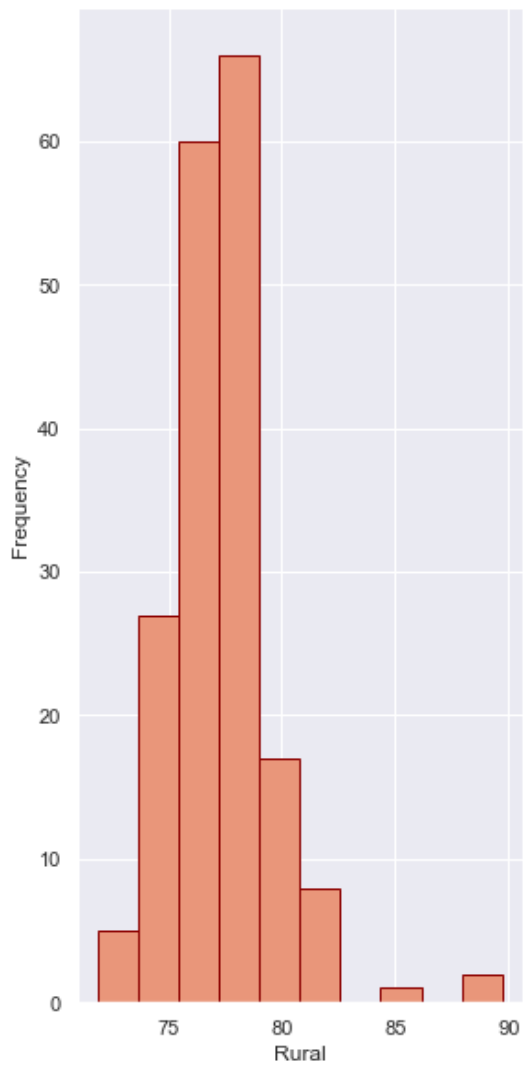
[30]: -375877.59850149235

```
[31]: # Histogram and PDFs Life_Expectancy rural vs urban
# =====
# plot side by side for comparison
subtitle = "Life_Expectancy Rural vs Urban"
ser1 = rural['Life_Expectancy']

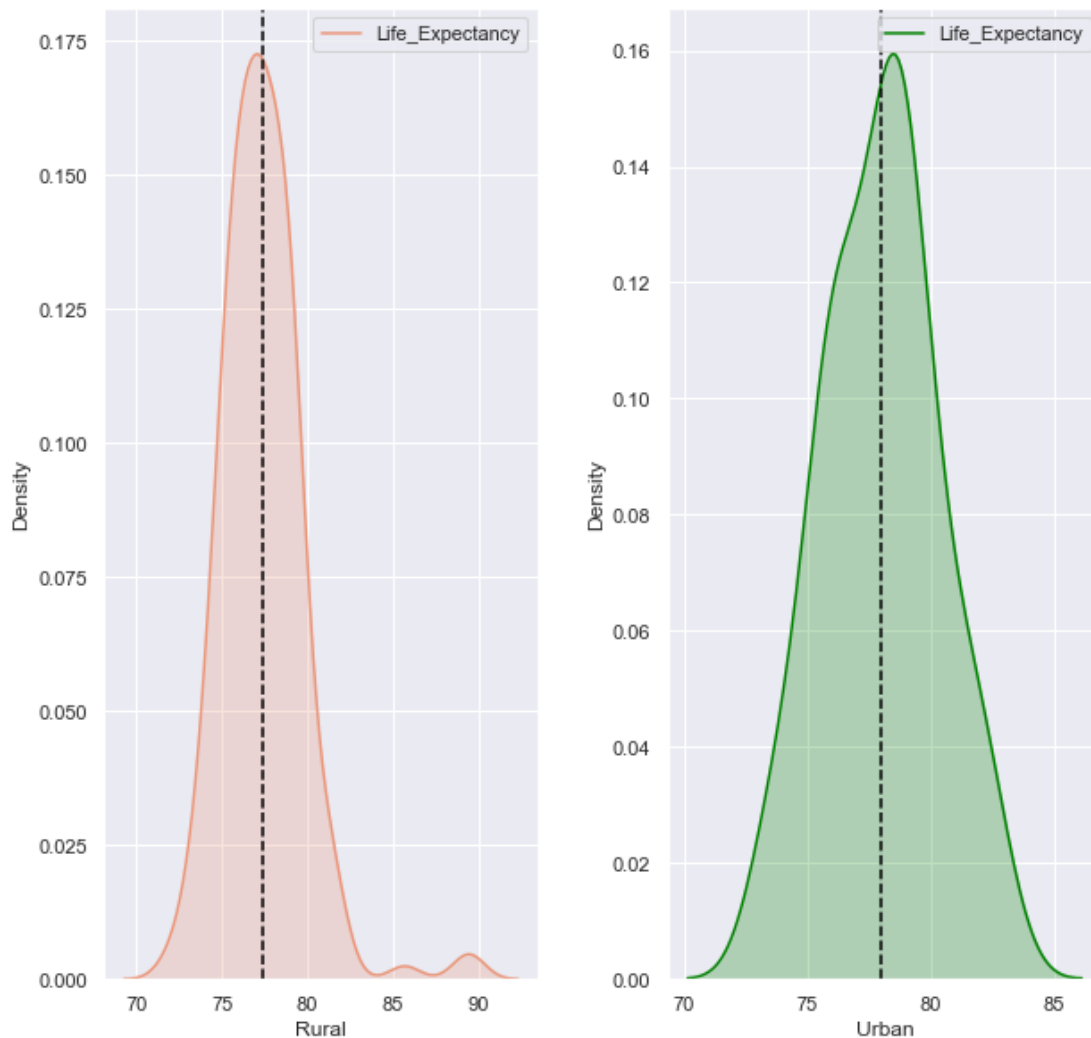
ser2 = urban.Life_Expectancy
mean1 = rural.Life_Expectancy.mean()
mean2 = urban.Life_Expectancy.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

### Life\_Expectancy Rural vs Urban



### Life\_Expectancy Rural vs Urban



```
[32]: # Cohen's d while I have the mean
rural.Life_Expectancy.var()
title = 'Cohens d for Life Expectancy Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

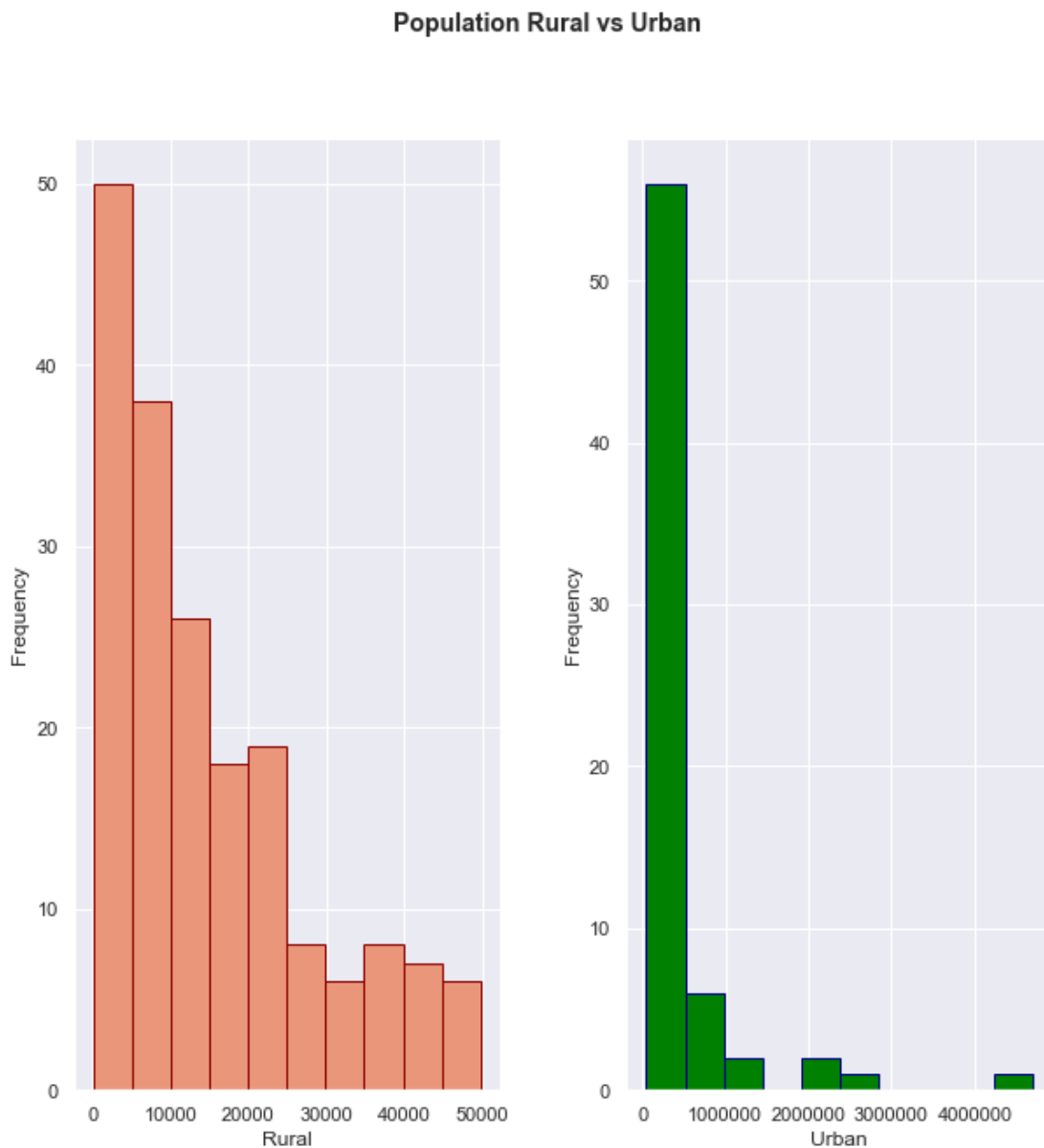
Cohens d for Life Expectancy Rural vs Urban = -1.099904255686382

```
[32]: -1.099904255686382
```

```
[33]: # Histogram and PDFs rural vs urban
# =====
# plot side by side for comparison
subtitle = "Population Rural vs Urban"
ser1 = rural['Population']

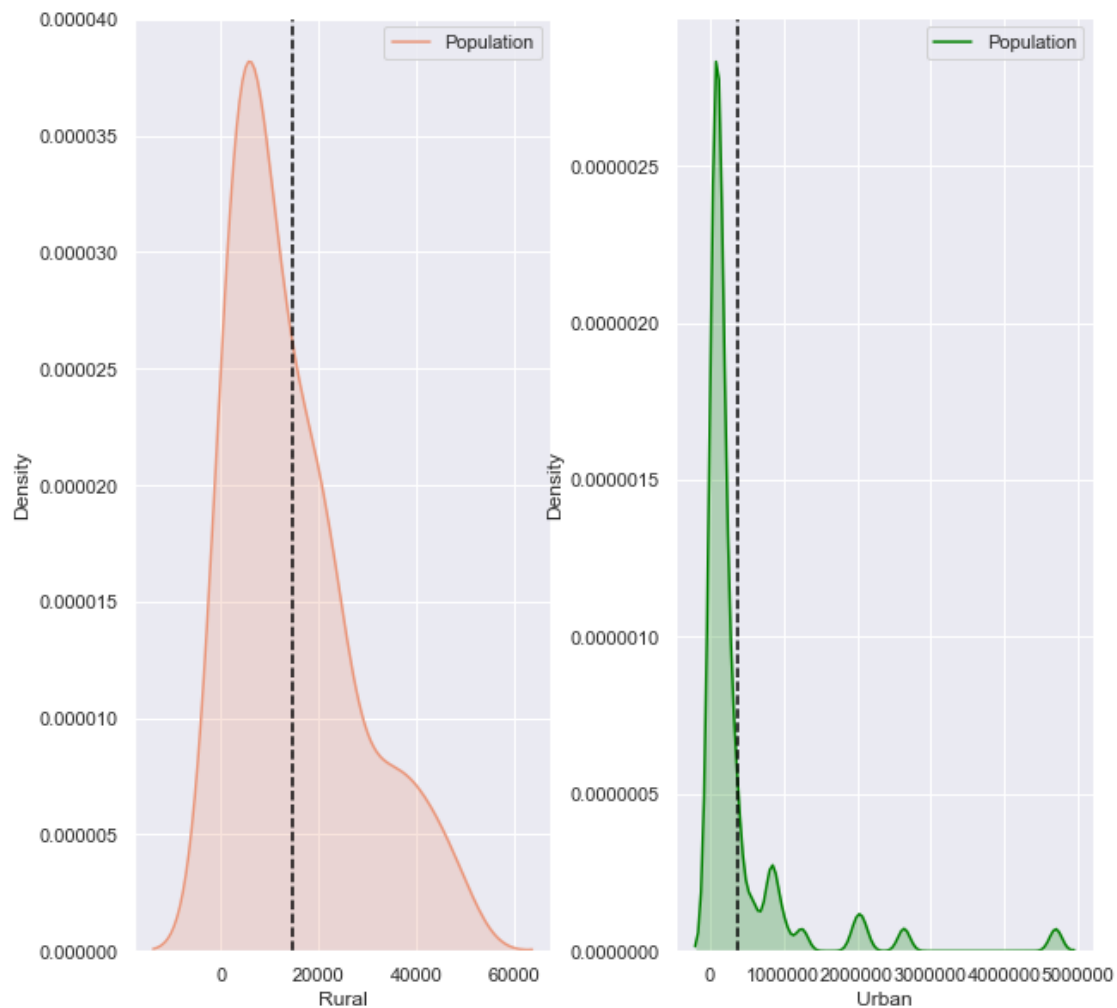
ser2 = urban.Population
mean1 = rural.Population.mean()
mean2 = urban.Population.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```





### Population Rural vs Urban



```
[34]: # Cohen's d while I have the mean
rural.Population.var()
title = 'Cohens d for Population Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

Cohens d for Population Rural vs Urban = -643614.5781972056

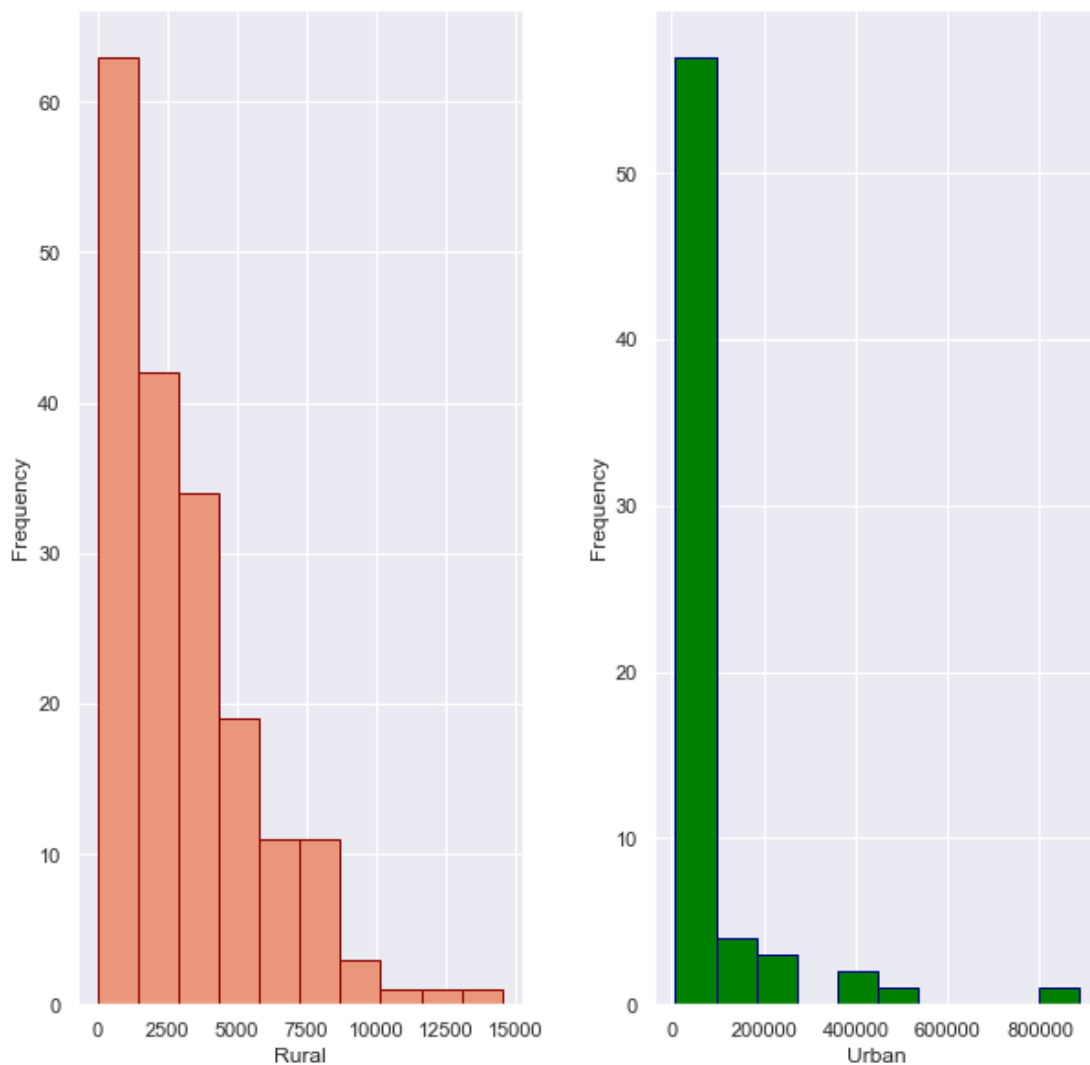
[34]: -643614.5781972056

```
[35]: # Histogram and PDFs rural vs urban
# =====
# plot side by side for comparison
subtitle = "poor_health vs Urban"
ser1 = rural['poor_health']

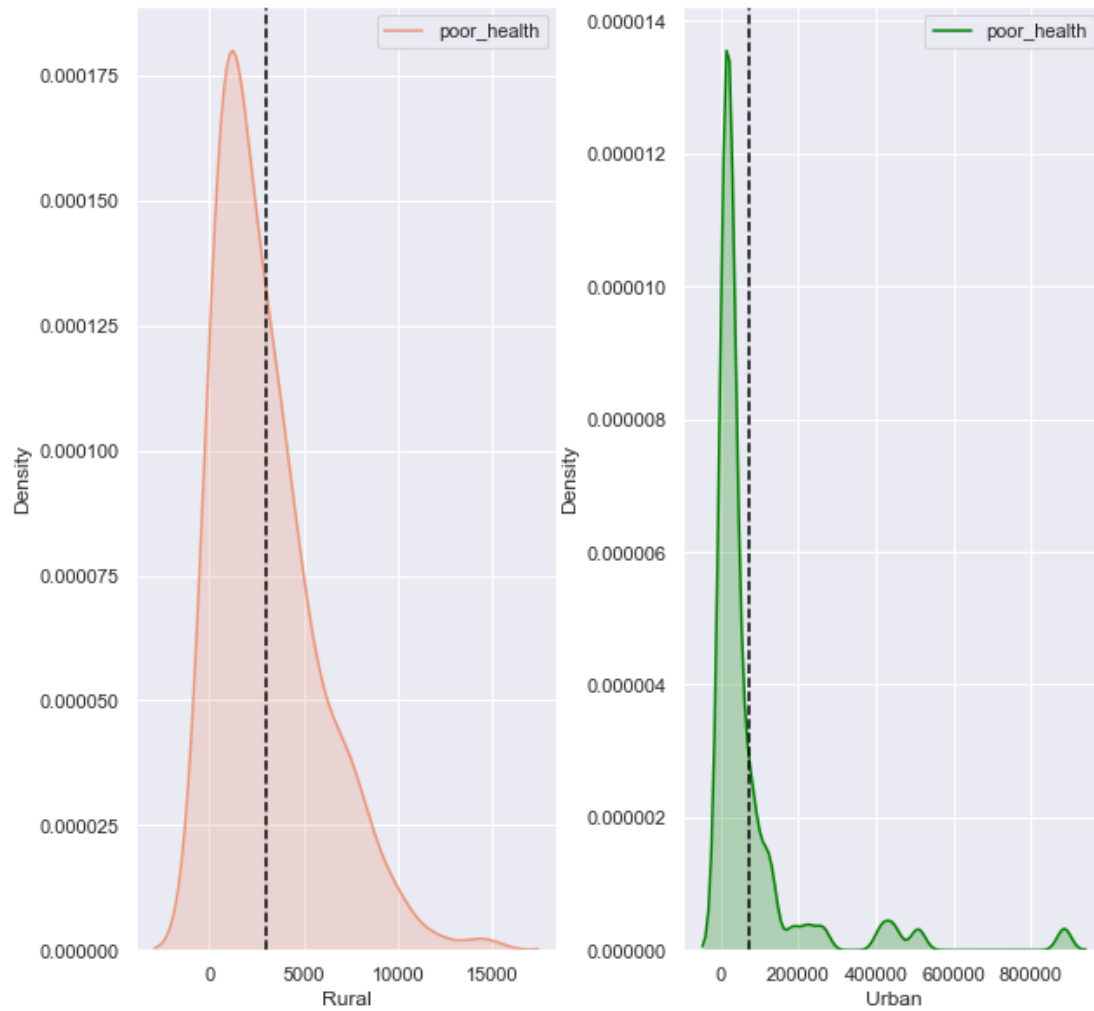
ser2 = urban.poor_health
mean1 = rural.poor_health.mean()
mean2 = urban.poor_health.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

**poor\_health vs Urban**



### poor\_health vs Urban



```
[36]: # Cohen's d while I have the mean
      var1 = rural.poor_health.var()
      title = 'Cohens d for Poor Health vs Urban'
      Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

Cohens d for Poor Health vs Urban = -31.33146101365113

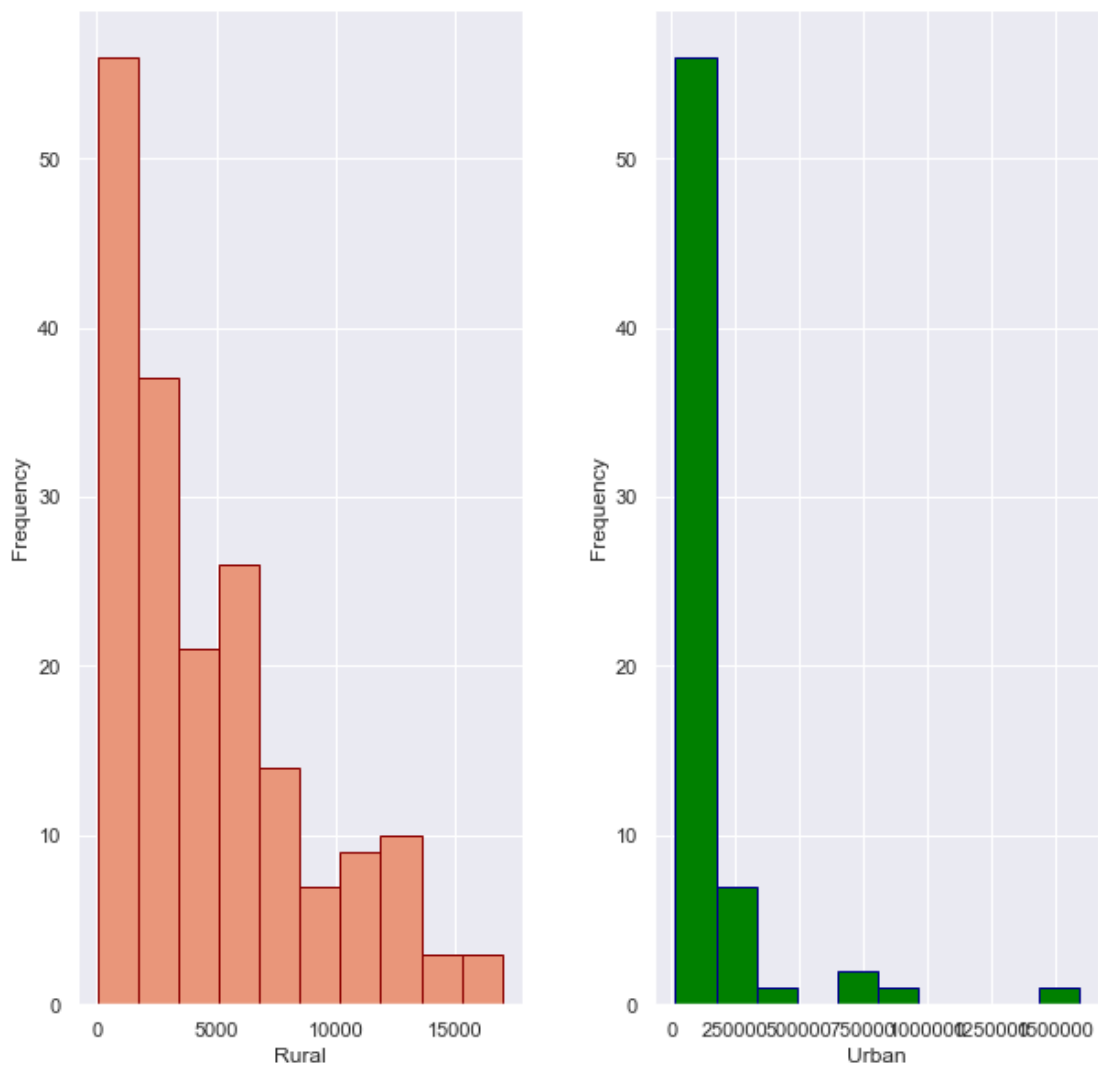
```
[36]: -31.33146101365113
```

```
[37]: # Histogram and PDFs rural vs urban
# =====
# plot side by side for comparison
subtitle = "poor_sleep vs Urban"
ser1 = rural['poor_sleep']

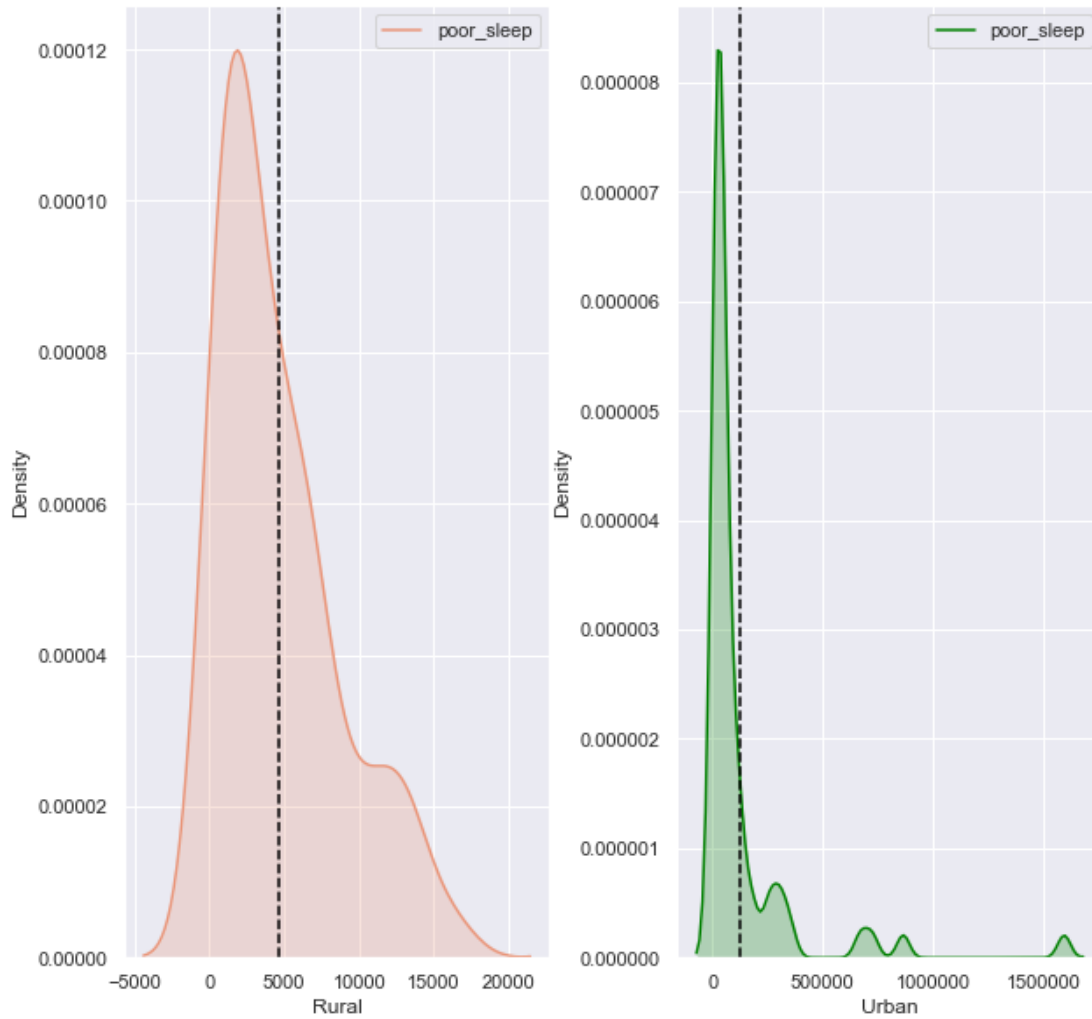
ser2 = urban.poor_sleep
mean1 = rural.poor_sleep.mean()
mean2 = urban.poor_sleep.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

poor\_sleep vs Urban



### poor\_sleep vs Urban



```
[38]: # Cohen's d while I have the mean
var1 = rural.poor_sleep.var()
title = 'Cohens d for Poor Sleep Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

Cohens d for Poor Sleep Rural vs Urban = -35.08603547270512

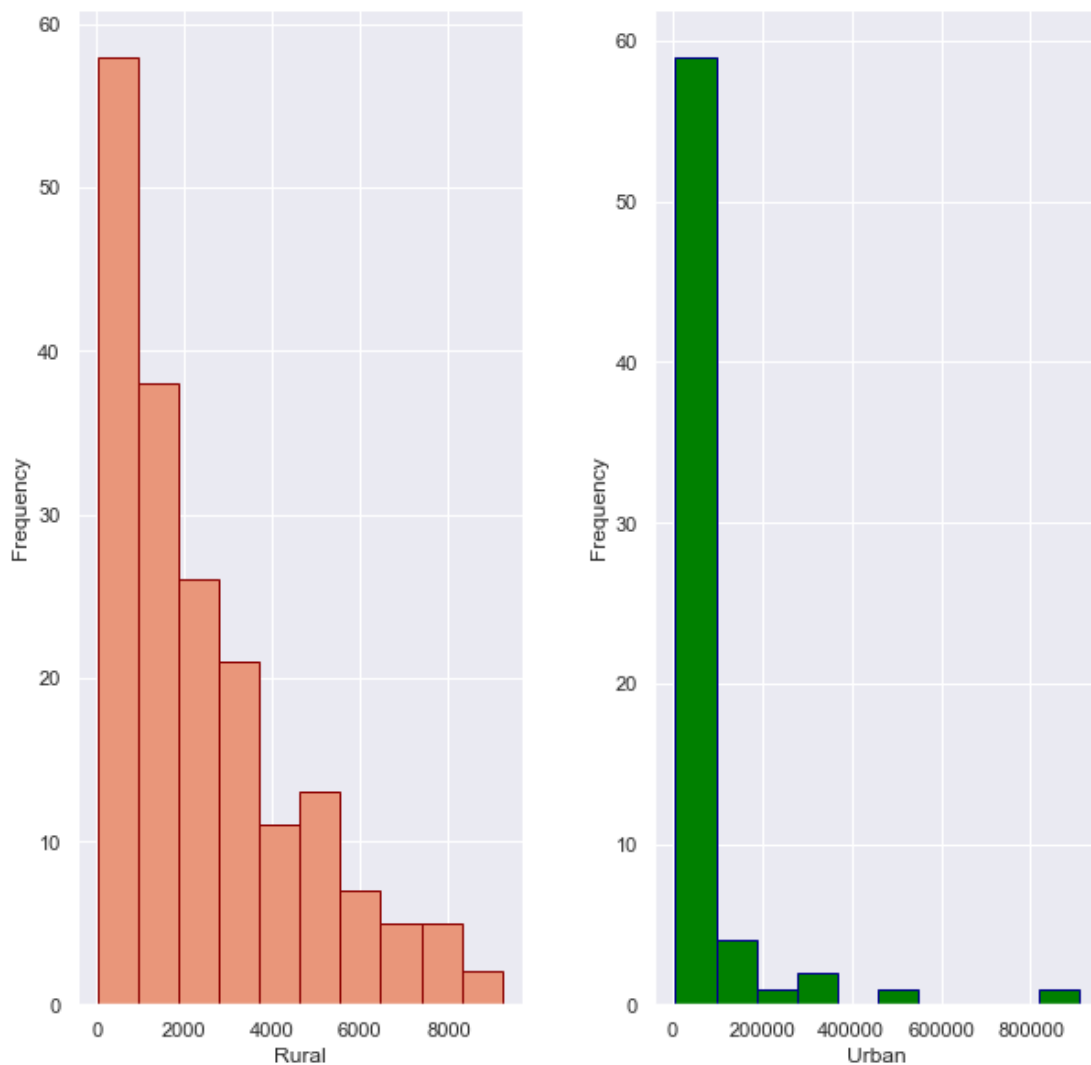
[38]: -35.08603547270512

```
[39]: # Histogram and PDFs rural vs urban
# =====
# plot side by side for comparison
subtitle = "Uninsured_health Rural vs Urban"
ser1 = rural['Uninsured_health']

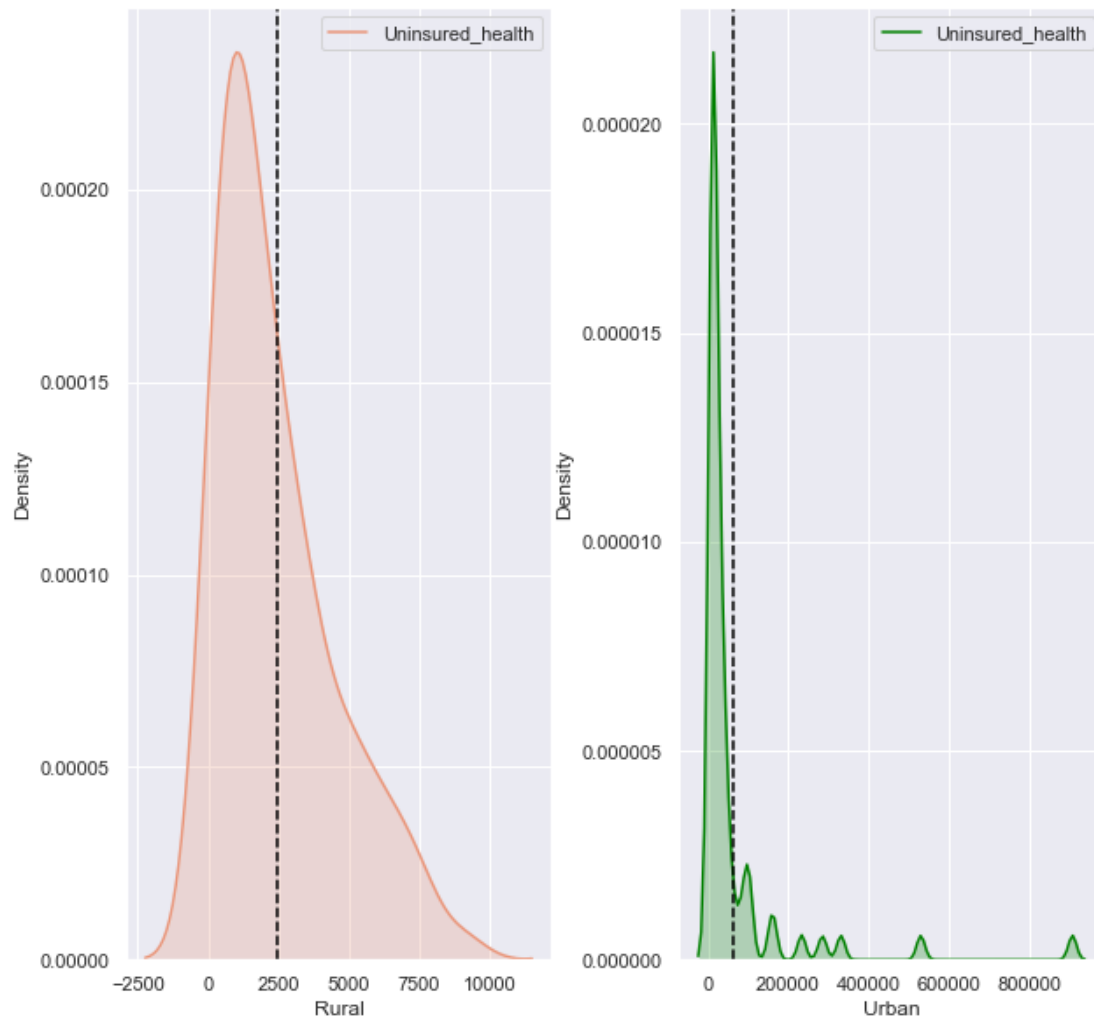
ser2 = urban.Uninsured_health
mean1 = rural.Uninsured_health.mean()
mean2 = urban.Uninsured_health.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

Uninsured\_health Rural vs Urban



### Uninsured\_health Rural vs Urban



```
[40]: # Cohen's d while I have the mean
var1 = rural.Uninsured_health.var()
title = 'Cohens d for Uninsured Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

Cohens d for Uninsured Rural vs Urban = -33.68266402418818

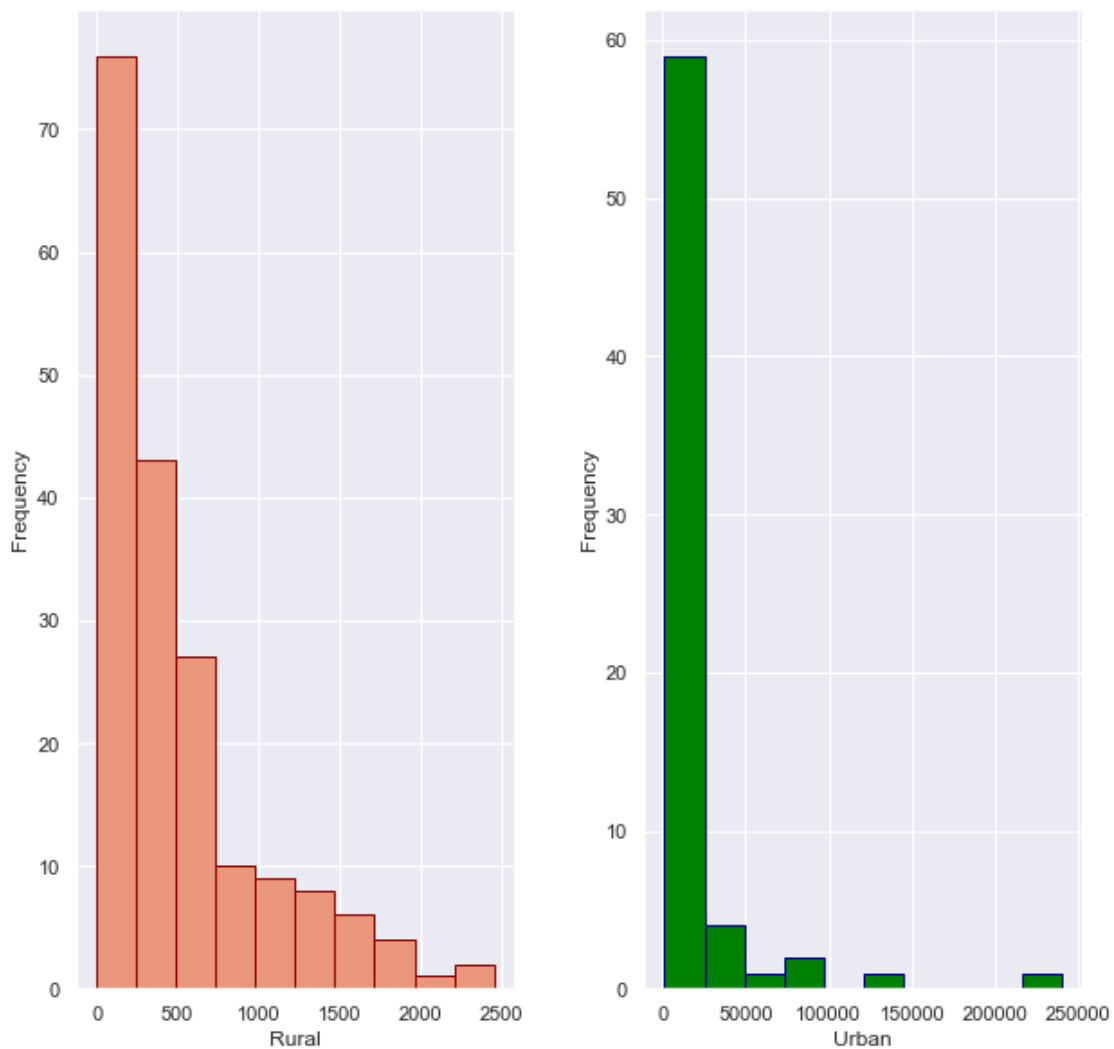
[40]: -33.68266402418818

```
[41]: # Histogram and PDFs  rural vs urban
# =====
# plot side by side for comparison
subtitle = "Cost_Burden vs Urban"
ser1 = rural.Cost_Burden

ser2 = urban.Cost_Burden
mean1 = rural.Cost_Burden.mean()
mean2 = urban.Cost_Burden.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

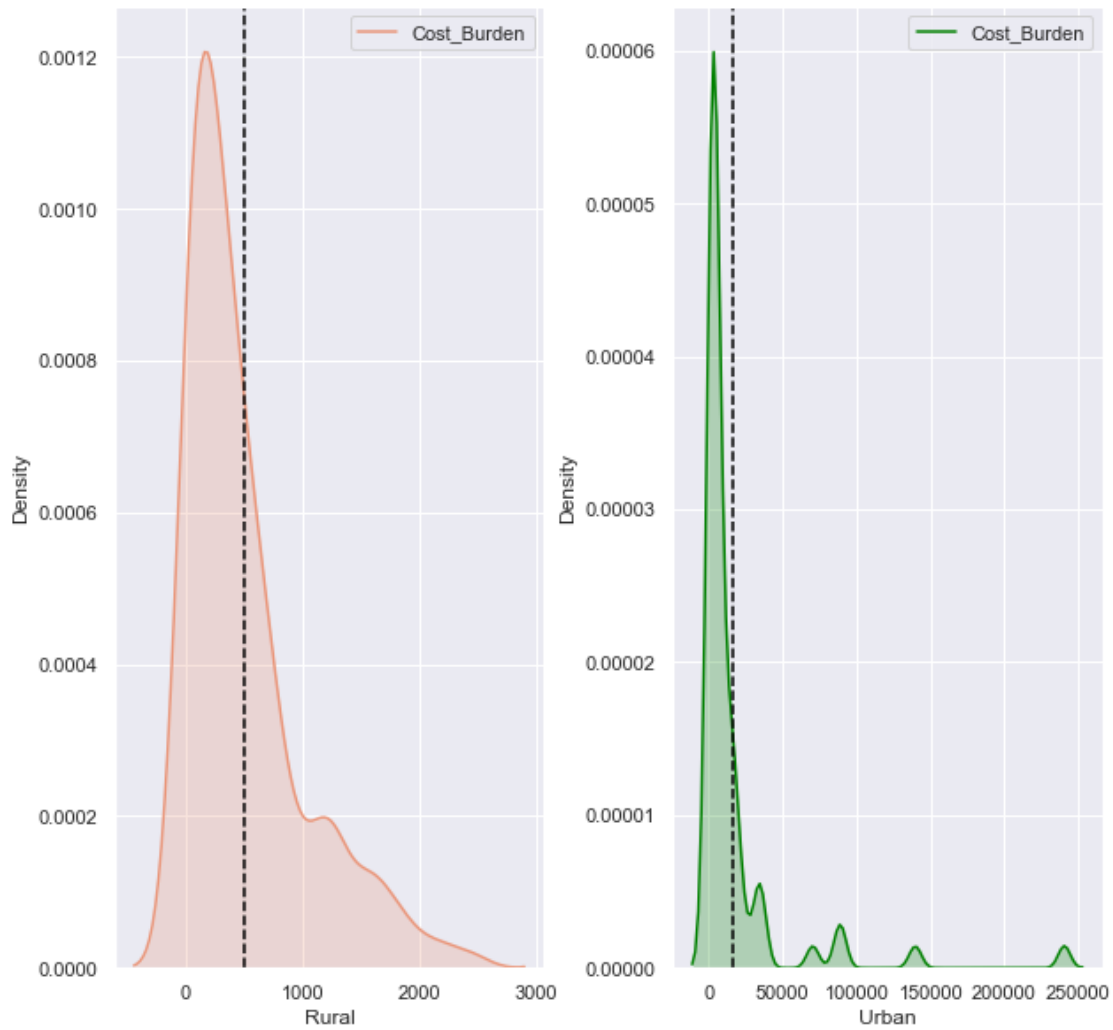
# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```

**Cost\_Burden vs Urban**





### Cost\_Burden vs Urban



```
[42]: # Cohen's d while I have the mean
var1 = rural.Cost_Burden.var()
title = 'Cohens d for Uninsured Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

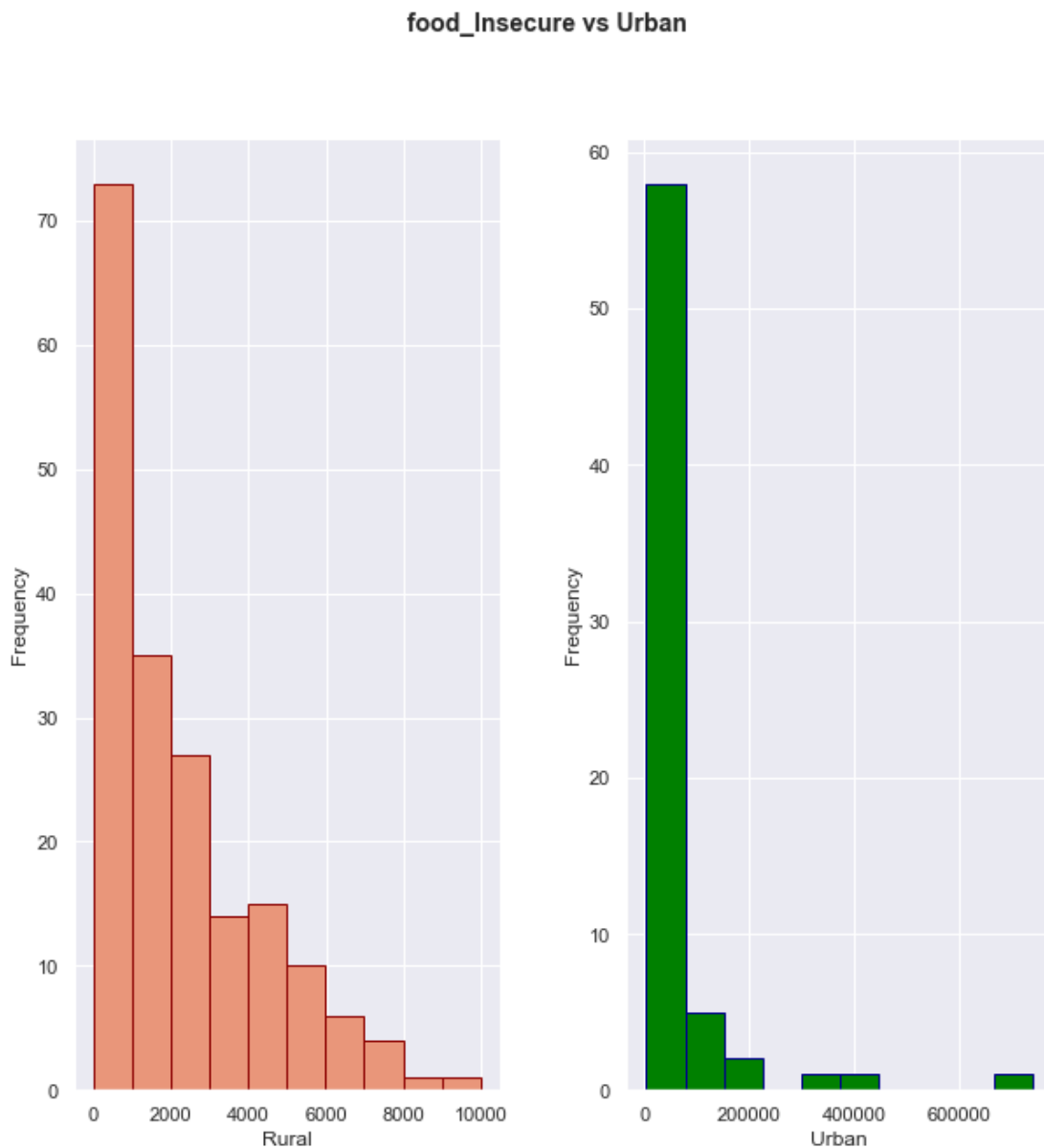
Cohens d for Uninsured Rural vs Urban = -37.88726489156317

[42]: -37.88726489156317

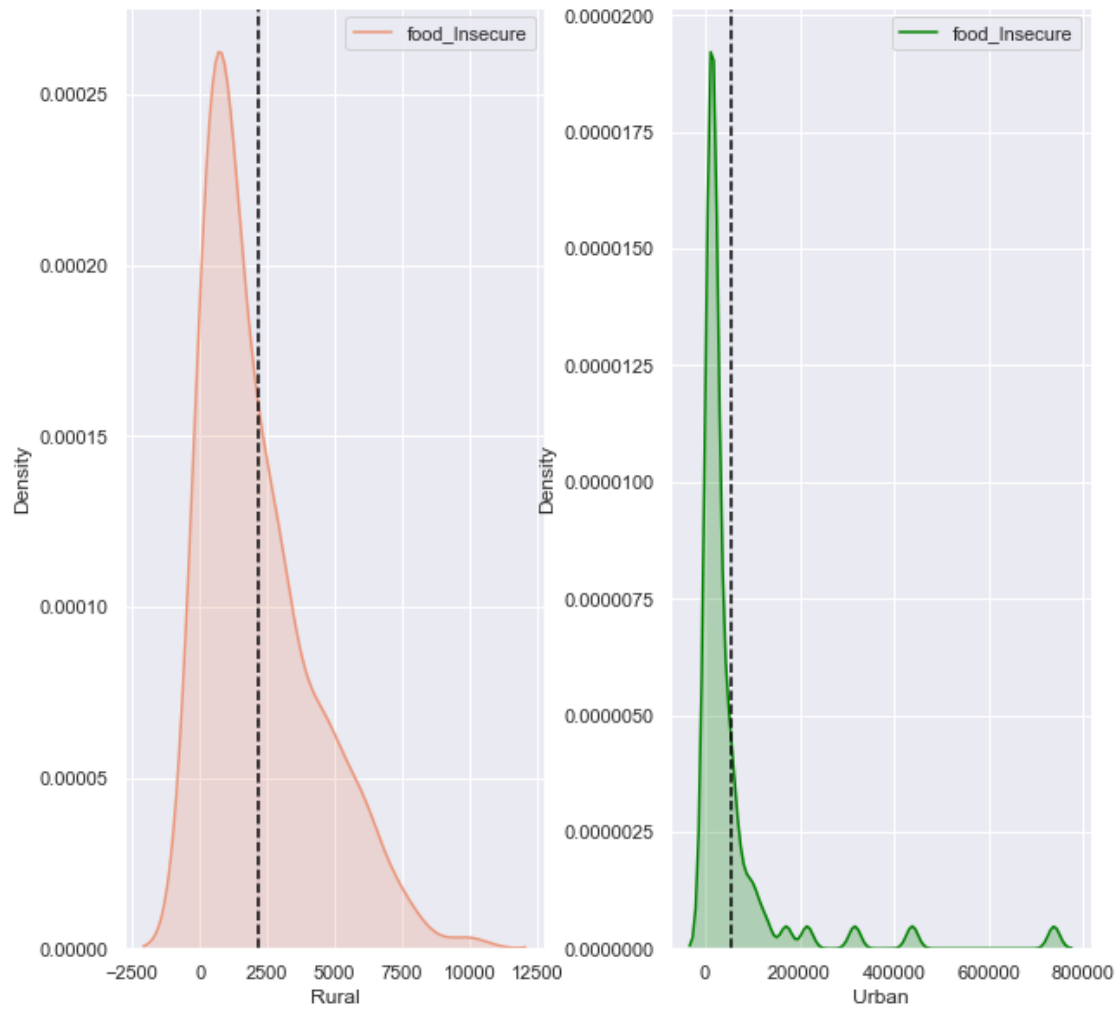
```
[43]: # Histogram and PDFs  rural vs urban
# =====
# plot side by side for comparison
subtitle = "food_Insecure vs Urban"
ser1 = rural.food_Insecure

ser2 = urban.food_Insecure
mean1 = rural.food_Insecure.mean()
mean2 = urban.food_Insecure.mean()
Hist_Plot(subtitle, xlab1, xlab2, ser1, ser2)

# KDE/PDFs
Sns_Kde(subtitle,xlab1, xlab2, ser1, mean1, ser2, mean2)
```



### food\_Insecure vs Urban



```
[44]: # Cohen's d while I have the mean
var1 = rural.food_Insecure.var()
title = 'Cohens d for Uninsured Rural vs Urban'
Compute_Cohend(mean1, mean2,ser1,ser2, var1, title)
```

Cohens d for Uninsured Rural vs Urban = -30.063165548759553

[44]: -30.063165548759553

## 0.9.2 Correlations

```
[45]: # =====  
# Run correlation for bad health habits rural vs urban. Hypothesis Do  
#   rural residents have worse health outcomes due to poor health, lack  
#   of availability of doctors, access to food due to housing costs  
#   which may have an impact on poor sleep and health?  
#  
# =====  
  
# normalize the data with sklearn  
scaler = MinMaxScaler()  
rural_df = rural[['Population', 'poor_health', 'poor_sleep',  
    ↪ 'Uninsured_health',  
    'Life_Expectancy', 'Physicians', 'food_Insecure', 'Cost_Burden']].copy()  
  
rural_df[['Population', 'poor_health', 'poor_sleep', 'Uninsured_health',  
    'Life_Expectancy', 'Physicians', 'food_Insecure', 'Cost_Burden']]  
    = scaler.fit_transform(rural_df[['Population', 'poor_health',  
    'poor_sleep', 'Uninsured_health', 'Life_Expectancy', 'Physicians',  
    'food_Insecure', 'Cost_Burden']])  
  
urban_df = urban[['Population', 'poor_health', 'poor_sleep',  
    ↪ 'Uninsured_health',  
    'Life_Expectancy', 'Physicians', 'food_Insecure', 'Cost_Burden']].copy()  
  
urban_df[['Population', 'poor_health', 'poor_sleep', 'Uninsured_health',  
    'Life_Expectancy', 'Physicians', 'food_Insecure', 'Cost_Burden']]  
    = scaler.fit_transform(urban_df[['Population', 'poor_health',  
    'poor_sleep', 'Uninsured_health', 'Life_Expectancy', 'Physicians',  
    'food_Insecure', 'Cost_Burden']])  
  
#Pearson Correlation  
corr_rural1 = rural_df.corr(method="pearson")  
  
corr_urban1 = urban_df.corr(method="pearson")  
  
print("Pearson correlation coefficient Rural : \n")  
print(corr_rural1, "\n")  
print("Pearson correlation coefficient Urban : \n")  
print(corr_urban1, "\n")
```

Pearson correlation coefficient Rural :

	Population	poor_health	poor_sleep	Uninsured_health	\
Population	1.000000	0.949782	0.997272	0.968661	
poor_health	0.949782	1.000000	0.961490	0.949579	
poor_sleep	0.997272	0.961490	1.000000	0.968746	
Uninsured_health	0.968661	0.949579	0.968746	1.000000	
Life_Expectancy	-0.169082	-0.157482	-0.179557	-0.147392	
Physicians	0.802969	0.716882	0.785917	0.765350	
food_Insecure	0.949488	0.866604	0.948382	0.901662	
Cost_Burden	0.941897	0.886859	0.938154	0.904360	

	Life_Expectancy	Physicians	food_Insecure	Cost_Burden	
Population	-0.169082	0.802969	0.949488	0.941897	
poor_health	-0.157482	0.716882	0.866604	0.886859	
poor_sleep	-0.179557	0.785917	0.948382	0.938154	
Uninsured_health	-0.147392	0.765350	0.901662	0.904360	
Life_Expectancy	1.000000	-0.047904	-0.253418	-0.171704	
Physicians	-0.047904	1.000000	0.738383	0.807168	
food_Insecure	-0.253418	0.738383	1.000000	0.927326	
Cost_Burden	-0.171704	0.807168	0.927326	1.000000	

Pearson correlation coefficient Urban :

	Population	poor_health	poor_sleep	Uninsured_health	\
Population	1.000000	0.986811	0.999027	0.983531	
poor_health	0.986811	1.000000	0.989802	0.985677	
poor_sleep	0.999027	0.989802	1.000000	0.985493	
Uninsured_health	0.983531	0.985677	0.985493	1.000000	
Life_Expectancy	0.343420	0.330408	0.330367	0.293629	
Physicians	0.985984	0.961588	0.980984	0.950631	
food_Insecure	0.989885	0.968455	0.989285	0.984261	
Cost_Burden	0.992718	0.978460	0.991086	0.986810	

	Life_Expectancy	Physicians	food_Insecure	Cost_Burden	
Population	0.343420	0.985984	0.989885	0.992718	
poor_health	0.330408	0.961588	0.968455	0.978460	
poor_sleep	0.330367	0.980984	0.989285	0.991086	
Uninsured_health	0.293629	0.950631	0.984261	0.986810	
Life_Expectancy	1.000000	0.368765	0.283458	0.299450	
Physicians	0.368765	1.000000	0.971714	0.978157	
food_Insecure	0.283458	0.971714	1.000000	0.993071	
Cost_Burden	0.299450	0.978157	0.993071	1.000000	

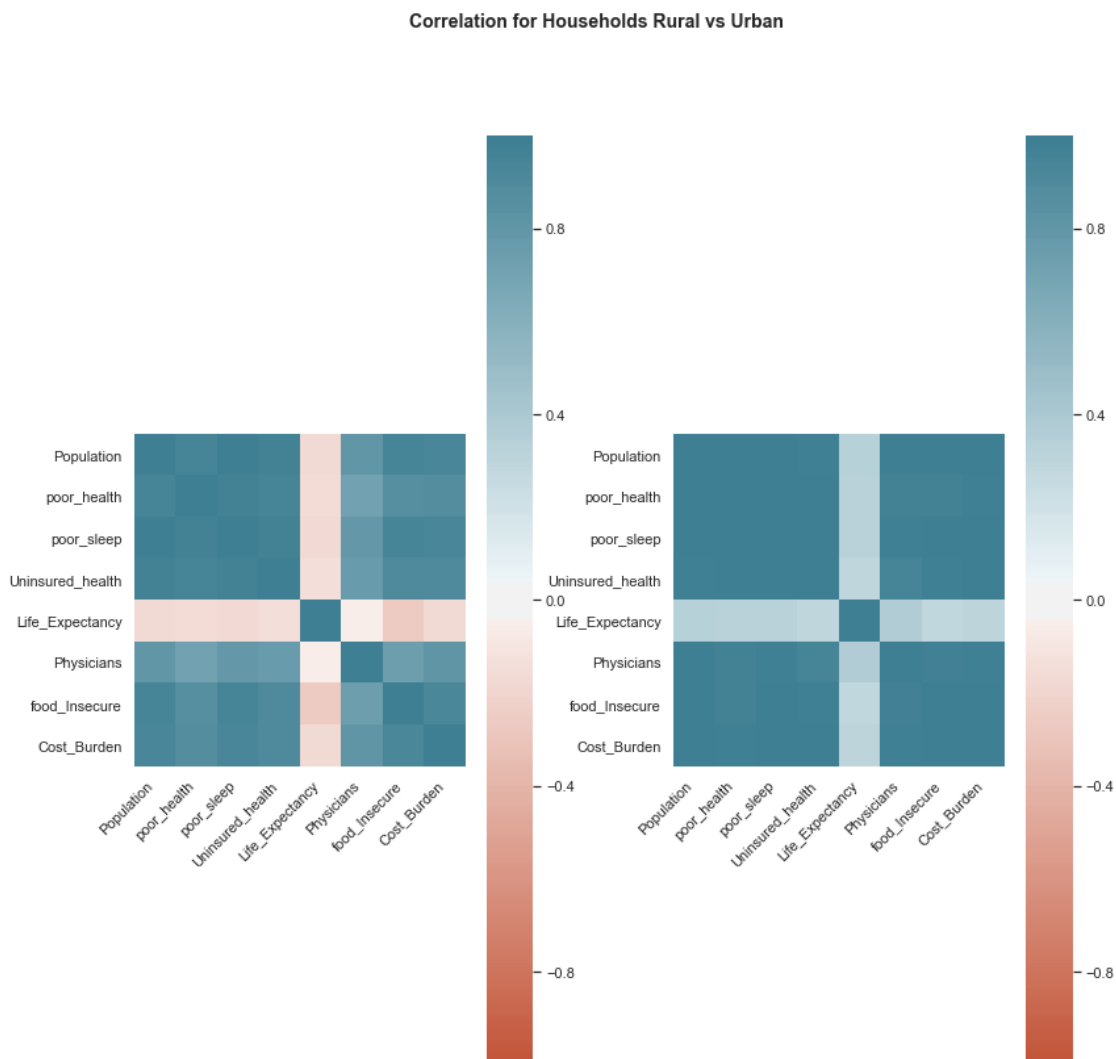
/Users/corosco/anaconda3/lib/python3.7/site-packages/sklearn/preprocessing/data.py:334: DataConversionWarning: Data with input dtype int64, float64 were all converted to float64 by MinMaxScaler.

return self.partial\_fit(X, y)

/Users/corosco/anaconda3/lib/python3.7/site-

packages/sklearn/preprocessing/data.py:334: DataConversionWarning: Data with input dtype int64, float64 were all converted to float64 by MinMaxScaler.  
 return self.partial\_fit(X, y)

```
[46]: # =====
# Display correlation
# =====
subtitle = "Correlation for Households Rural vs Urban"
xlab1 = "Rural"
xlab2 = "Urban"
data1 = corr_rural1
data2 = corr_urban1
Heatmap_Plot(subtitle, xlab1, xlab2, data1, data2)
```



```
[47]: # =====
# Include the other descriptive characteristics about the variables: Mean,
# Mode, Spread, and Tails (Chapter 2)
# =====
# # Display descriptive statistics
# =====
rural_cp = rural[['Population', 'poor_health', 'poor_sleep',
↳ 'Uninsured_health',\
    'Life_Expectancy', 'Physicians', 'food_Insecure', 'Cost_Burden']].copy()

urban_cp = urban[['Population', 'poor_health', 'poor_sleep',
↳ 'Uninsured_health',\
    'Life_Expectancy', 'Physicians', 'food_Insecure', 'Cost_Burden']].copy()

# before scaling
print('RURAL \n')
print(rural_cp.describe(), '\n')
print('URBAN \n')
print(urban_cp.describe(), '\n')
```

#### RURAL

	Population	poor_health	poor_sleep	Uninsured_health \
count	186.000000	186.000000	186.000000	186.000000
mean	14770.026882	3064.365591	4692.655914	2444.564516
std	12605.944718	2673.077929	4040.692191	2083.528517
min	152.000000	23.000000	50.000000	27.000000
25%	4470.000000	857.250000	1418.500000	812.750000
50%	10989.000000	2335.500000	3414.500000	1827.500000
75%	21421.000000	4372.750000	6814.750000	3549.000000
max	49728.000000	14489.000000	16987.000000	9222.000000

	Life_Expectancy	Physicians	food_Insecure	Cost_Burden
count	186.000000	186.000000	186.000000	186.000000
mean	77.393548	5.317204	2178.870968	496.537634
std	2.376930	6.139057	2018.025879	505.374179
min	71.900000	0.000000	10.000000	0.000000
25%	75.800000	1.000000	572.500000	120.000000
50%	77.300000	3.000000	1490.000000	336.000000
75%	78.975000	8.000000	3105.000000	656.750000

max	89.700000	38.000000	9970.000000	2451.000000
-----	-----------	-----------	-------------	-------------

URBAN

	Population	poor_health	poor_sleep	Uninsured_health \
count	6.800000e+01	68.000000	6.800000e+01	68.000000
mean	3.816856e+05	74733.455882	1.260119e+05	62499.058824
std	7.236031e+05	141018.568160	2.434585e+05	135041.924584
min	5.003100e+04	8414.000000	1.508600e+04	6687.000000
25%	6.795200e+04	14305.500000	2.276400e+04	11253.500000
50%	1.330275e+05	23471.000000	4.166000e+04	18019.500000
75%	3.150315e+05	68266.000000	1.077918e+05	41986.500000
max	4.698619e+06	885812.000000	1.593969e+06	908742.000000

	Life_Expectancy	Physicians	food_Insecure	Cost_Burden
count	68.000000	68.000000	68.000000	68.000000
mean	78.020588	239.191176	54094.852941	16881.529412
std	2.294286	462.947283	110360.809809	36246.466308
min	73.300000	6.000000	4210.000000	1182.000000
25%	76.200000	23.750000	10707.500000	2361.000000
50%	78.300000	74.000000	17410.000000	4829.500000
75%	79.425000	232.500000	47455.000000	14472.000000
max	83.000000	2742.000000	739120.000000	240521.000000

```
[48]: # after scaling
print('RURAL \n')
print(rural_df.describe(), '\n')
print('URBAN \n')
print(urban_df.describe(), '\n')
```

RURAL

	Population	poor_health	poor_sleep	Uninsured_health	Life_Expectancy \
count	186.000000	186.000000	186.000000	186.000000	186.000000
mean	0.294861	0.210242	0.274113	0.262922	0.308626
std	0.254275	0.184783	0.238572	0.226594	0.133535
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.087099	0.057670	0.080799	0.085454	0.219101
50%	0.218594	0.159858	0.198648	0.195813	0.303371
75%	0.429018	0.300688	0.399407	0.383034	0.397472
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Physicians	food_Insecure	Cost_Burden
count	186.000000	186.000000	186.000000
mean	0.139926	0.217758	0.202586
std	0.161554	0.202613	0.206191
min	0.000000	0.000000	0.000000



25%	0.026316	0.056476	0.048960
50%	0.078947	0.148594	0.137087
75%	0.210526	0.310743	0.267952
max	1.000000	1.000000	1.000000

URBAN

	Population	poor_health	poor_sleep	Uninsured_health	Life_Expectancy \
count	68.000000	68.000000	68.000000	68.000000	68.000000
mean	0.071345	0.075587	0.070256	0.061872	0.486659
std	0.155661	0.160724	0.154197	0.149705	0.236524
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.003855	0.006715	0.004863	0.005062	0.298969
50%	0.017854	0.017161	0.016831	0.012563	0.515464
75%	0.057007	0.068215	0.058716	0.039132	0.631443
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Physicians	food_Insecure	Cost_Burden
count	68.000000	68.000000	68.000000
mean	0.085231	0.067879	0.065595
std	0.169206	0.150169	0.151444
min	0.000000	0.000000	0.000000
25%	0.006488	0.008841	0.004926
50%	0.024854	0.017961	0.015240
75%	0.082785	0.058844	0.055528
max	1.000000	1.000000	1.000000

```
[49]: for x in rural_df:
      print(f'Mode for {x} = {rural_cp[x].mode()} \n')
```

```
Mode for Population = 0      152
1      277
2      442
3      648
4      726
5      749
6      823
7      903
8     1200
9     1234
10    1311
11    1362
12    1388
13    1389
14    1515
15    1516
16    1522
```

17	1892
18	1928
19	2131
20	2139
21	2204
22	2249
23	2252
24	2836
25	2895
26	2962
27	3028
28	3079
29	3253

...

156	28360
157	28719
158	28875
159	29989
160	30119
161	31129
162	32587
163	33033
164	33830
165	35108
166	35286
167	35872
168	36354
169	36459
170	36552
171	36810
172	37924
173	40574
174	40822
175	41260
176	41619
177	42446
178	42454
179	43247
180	45129
181	45641
182	47542
183	49208
184	49565
185	49728

Length: 186, dtype: int64

Mode for poor\_health = 0  
1 48

23

2	82
3	118
4	122
5	134
6	136
7	193
8	206
9	218
10	232
11	236
12	243
13	256
14	279
15	280
16	293
17	304
18	348
19	392
20	414
21	468
22	486
23	530
24	537
25	553
26	572
27	586
28	600
29	614
...	
156	5618
157	5677
158	5845
159	6091
160	6332
161	6372
162	6560
163	6605
164	6669
165	6825
166	7006
167	7191
168	7207
169	7256
170	7265
171	7530
172	7593
173	7773
174	7862

```

175      8064
176      8103
177      8582
178      8691
179      8695
180      9580
181      9726
182      9972
183     10456
184     11985
185     14489
Length: 186, dtype: int64

```

```

Mode for poor_sleep = 0      50
1          83
2         145
3         185
4         206
5         226
6         257
7         263
8         356
9         365
10        412
11        427
12        428
13        437
14        438
15        443
16        462
17        543
18        568
19        616
20        666
21        676
22        707
23        732
24        878
25        950
26        951
27        957
28        970
29        977
...
156       9041
157       9102
158       9160
159       9355

```

```

160      9747
161     10314
162     10697
163     10756
164     10898
165     11119
166     11123
167     11172
168     11496
169     11762
170     12105
171     12330
172     12594
173     12803
174     13014
175     13035
176     13220
177     13328
178     13467
179     13576
180     13619
181     14246
182     14571
183     15987
184     16130
185     16987
Length: 186, dtype: int64

Mode for Uninsured_health = 0      767
1      1896
2      3023
dtype: int64

Mode for Life_Expectancy = 0      79.0
dtype: float64

Mode for Physicians = 0      1
dtype: int64

Mode for food_Insecure = 0      320
1      910
2      1130
dtype: int64

Mode for Cost_Burden = 0      120
dtype: int64

```

```
[50]: for x in rural_df:
      print(f'Mode for {x} = {urban_cp[x].mode()} \n')
```

```
Mode for Population = 0      50031
1      50224
2      50310
3      50921
4      52405
5      52592
6      53126
7      54450
8      56019
9      57207
10     58057
11     58485
12     60537
13     64525
14     65711
15     66726
16     66893
17     68305
18     72480
19     74808
20     82299
21     83572
22     86323
23     86976
24     87092
25     92035
26     94324
27     96493
28    100657
29    118189
...
38    148373
39    162124
40    163694
41    171361
42    172578
43    179436
44    222631
45    226758
46    230221
47    254607
48    255001
49    275910
50    307412
```

```
51      337890
52      355642
53      362265
54      370200
55      423908
56      566719
57      590925
58      787858
59      840758
60      859064
61      865939
62     1005146
63     1248743
64     1986049
65     2084931
66     2637772
67     4698619
```

```
Length: 68, dtype: int64
```

```
Mode for poor_health = 0      8414
```

```
1       8712
2       9253
3       9300
4       9585
5       9991
6      10335
7      11250
8      11290
9      11310
10     11453
11     11585
12     11913
13     12292
14     13583
15     13785
16     14142
17     14360
18     14900
19     15368
20     15380
21     17568
22     17694
23     17798
24     17823
25     18203
26     19091
27     20089
28     20212
```

29	20567
	...
38	28581
39	28787
40	28948
41	29108
42	30214
43	33891
44	40119
45	40756
46	42572
47	51644
48	53336
49	63214
50	68029
51	68977
52	72158
53	84503
54	85533
55	88737
56	95886
57	121121
58	123953
59	125970
60	132923
61	185748
62	225358
63	264110
64	415851
65	446703
66	509327
67	885812

Length: 68, dtype: int64

Mode for poor_sleep = 0	15086
1	15135
2	15302
3	16181
4	16246
5	16618
6	17000
7	17291
8	17911
9	18041
10	18346
11	19510
12	20464
13	20981



14	21030
15	21403
16	22542
17	22838
18	24462
19	24752
20	26805
21	27961
22	28220
23	28577
24	29626
25	30307
26	30384
27	30538
28	33839
29	39214

...

38	49183
39	50714
40	51023
41	51336
42	54293
43	60314
44	66437
45	76393
46	76815
47	81512
48	89747
49	98229
50	105184
51	115615
52	117231
53	118163
54	129043
55	146194
56	180301
57	188136
58	251871
59	255958
60	288446
61	306354
62	324257
63	357721
64	672071
65	723866
66	865481
67	1593969

Length: 68, dtype: int64

Mode for Uninsured\_health = 0                      6687

1	6717
2	6890
3	7161
4	7653
5	7924
6	8075
7	8086
8	8108
9	8418
10	8753
11	8985
12	9311
13	9427
14	9875
15	10748
16	10811
17	11401
18	11728
19	11910
20	12598
21	13186
22	13501
23	14200
24	14374
25	14751
26	14914
27	15267
28	15499
29	16015
	...
38	21633
39	23333
40	25349
41	27461
42	29225
43	29371
44	30839
45	32046
46	36912
47	37306
48	40118
49	41152
50	41367
51	43845
52	44276
53	54780

54 57906  
 55 69387  
 56 85425  
 57 92077  
 58 96650  
 59 104621  
 60 104680  
 61 158704  
 62 163112  
 63 233591  
 64 285140  
 65 331573  
 66 530742  
 67 908742

Length: 68, dtype: int64

Mode for Life\_Expectancy = 0 78.4

dtype: float64

Mode for Physicians = 0 20

dtype: int64

Mode for food\_Insecure = 0 15950

dtype: int64

Mode for Cost\_Burden = 0 1182

1 1340  
 2 1348  
 3 1486  
 4 1638  
 5 1676  
 6 1808  
 7 1861  
 8 1867  
 9 1881  
 10 1902  
 11 1915  
 12 2070  
 13 2153  
 14 2197  
 15 2249  
 16 2331  
 17 2371  
 18 2485  
 19 2507  
 20 2794  
 21 3115  
 22 3378

```
23      3576
24      3577
25      3695
26      4050
27      4096
28      4119
29      4299
```

```
...
38      5483
39      5498
40      5760
41      6036
42      6515
43      6806
44      9782
45     10936
46     11094
47     11150
48     12132
49     13384
50     14406
51     14670
52     17388
53     17445
54     17459
55     17587
56     18634
57     20379
58     25022
59     32496
60     34287
61     35537
62     35969
63     70498
64     88496
65     89430
66     139743
67     240521
```

```
Length: 68, dtype: int64
```

```
[51]: urban_df
```

```
[51]:
```

	Population	poor_health	poor_sleep	Uninsured_health	Life_Expectancy	\
1	0.001727	0.003464	0.003406	0.000525	0.010309	
3	0.007973	0.011157	0.009640	0.009769	0.288660	
7	0.000060	0.003988	0.000970	0.001371	0.515464	

11	0.007948	0.010577	0.008545	0.009512	0.443299
14	0.065743	0.072651	0.072176	0.037061	0.525773
15	0.416474	0.499533	0.416107	0.308687	0.577320
19	0.009528	0.010724	0.011877	0.005790	0.237113
20	0.068874	0.067945	0.063671	0.041670	0.597938
21	0.038017	0.036861	0.038829	0.028112	0.824742
31	0.080428	0.141907	0.083038	0.108568	0.783505
37	0.000551	0.003232	0.001872	0.002290	0.247423
43	0.205463	0.133983	0.195816	0.108633	0.958763
46	0.021155	0.016095	0.018916	0.010926	0.680412
50	0.005330	0.006122	0.005938	0.001919	0.432990
57	0.556672	0.570907	0.538605	0.580957	0.577320
61	0.174038	0.131684	0.149970	0.099731	0.876289
68	0.024113	0.029037	0.022761	0.025147	0.206186
70	0.027837	0.022985	0.028646	0.023030	0.515464
71	0.170100	0.247258	0.184477	0.173410	0.742268
79	0.158721	0.128456	0.152558	0.094662	1.000000
84	0.061924	0.069026	0.064694	0.041193	0.453608
91	0.018061	0.016158	0.017106	0.015675	0.298969
92	0.015849	0.018029	0.016556	0.016569	0.247423
94	0.024451	0.023586	0.022565	0.013589	0.659794
100	0.001544	0.000000	0.001397	0.000000	0.247423
101	1.000000	1.000000	1.000000	1.000000	0.649485
102	0.003591	0.004420	0.004722	0.003038	0.309278
105	0.037130	0.036135	0.032524	0.026774	0.731959
107	0.006941	0.007392	0.007422	0.007205	0.144330
108	0.175517	0.291425	0.173135	0.251541	0.907216
..	...	...	...	...	...
155	0.044008	0.049271	0.042072	0.033944	0.494845
162	0.001819	0.013851	0.002802	0.007554	0.536082
163	0.000191	0.002189	0.000031	0.000033	0.608247
165	0.026362	0.023220	0.024832	0.024985	0.567010
170	0.116357	0.091547	0.109603	0.087287	0.659794
174	0.003373	0.006777	0.003734	0.005588	0.422680
178	0.067167	0.086721	0.065285	0.056780	0.525773
181	0.007215	0.006528	0.008154	0.004572	0.072165
184	0.019004	0.012169	0.016053	0.011725	0.525773
187	0.000000	0.001797	0.000694	0.001539	0.000000
188	0.014976	0.023403	0.016037	0.018453	0.103093
191	0.018552	0.013306	0.017913	0.009120	0.587629
199	0.010891	0.005891	0.009689	0.005226	0.783505
201	0.000951	0.003614	0.000735	0.001551	0.360825
205	0.003627	0.007939	0.004001	0.004502	0.298969
212	0.038762	0.038931	0.039097	0.033507	0.515464
214	0.003118	0.019008	0.004910	0.010341	0.597938
220	0.437746	0.464370	0.448912	0.360162	0.597938
221	0.018846	0.019813	0.017776	0.013474	0.257732

226	0.014662	0.016292	0.015282	0.011589	0.463918
227	0.257866	0.202114	0.217011	0.168523	0.876289
234	0.001288	0.000956	0.001212	0.002909	0.319588
235	0.009036	0.013447	0.009209	0.008329	0.525773
236	0.004829	0.007926	0.006122	0.002548	0.525773
237	0.000666	0.003278	0.002065	0.003534	0.628866
240	0.048591	0.099695	0.052659	0.069508	0.721649
243	0.017647	0.024846	0.021596	0.012309	0.226804
246	0.111149	0.087895	0.104640	0.053315	0.896907
247	0.000042	0.000340	0.000137	0.000225	0.639175
249	0.003931	0.003301	0.003765	0.006553	0.298969

	Physicians	food_Insecure	Cost_Burden
1	0.005117	0.008586	0.001270
3	0.016813	0.016818	0.010007
7	0.000731	0.000000	0.000000
11	0.006213	0.007933	0.004968
14	0.101608	0.079942	0.055252
15	0.527778	0.288484	0.364813
19	0.020833	0.021731	0.011983
20	0.083333	0.060416	0.040754
21	0.067251	0.053585	0.068543
31	0.073465	0.059273	0.068008
37	0.005482	0.006667	0.002616
43	0.333333	0.171286	0.145346
46	0.033626	0.015036	0.015309
50	0.004020	0.013172	0.004458
57	0.662281	0.591392	0.578932
61	0.191520	0.146481	0.130835
68	0.031798	0.015117	0.016629
70	0.023392	0.024642	0.017970
71	0.145833	0.102938	0.143541
79	0.235380	0.128710	0.099608
84	0.087354	0.066756	0.056355
91	0.020468	0.023363	0.016796
92	0.036915	0.028180	0.020281
94	0.018275	0.017853	0.014281
100	0.001827	0.007538	0.003063
101	1.000000	1.000000	1.000000
102	0.004751	0.012246	0.005444
105	0.031798	0.029296	0.041414
107	0.006944	0.013594	0.008076
108	0.139254	0.124124	0.138318
..	...	...	...
155	0.066155	0.055449	0.050982
162	0.003289	0.002082	0.003710
163	0.002924	0.000313	0.000694

165	0.019371	0.016614	0.018033
170	0.127193	0.094569	0.080208
174	0.014620	0.013498	0.010003
178	0.107091	0.050959	0.067950
181	0.004386	0.015975	0.006735
184	0.019371	0.018165	0.013383
187	0.008406	0.006355	0.002921
188	0.030702	0.021581	0.022282
191	0.027047	0.018070	0.013023
199	0.025585	0.008926	0.009175
201	0.001827	0.007361	0.002064
205	0.004386	0.005538	0.004801
212	0.082602	0.049380	0.035932
214	0.002558	0.005416	0.001905
220	0.434942	0.425590	0.368716
221	0.033260	0.025105	0.019128
226	0.026316	0.015049	0.015171
227	0.379020	0.227960	0.289614
234	0.001462	0.006096	0.002862
235	0.024123	0.010355	0.010500
236	0.005117	0.015975	0.012175
237	0.000000	0.006205	0.003008
240	0.030336	0.028629	0.045751
243	0.032529	0.027486	0.023498
246	0.135234	0.081017	0.067711
247	0.006579	0.000435	0.000660
249	0.007675	0.006722	0.004241

[68 rows x 8 columns]

[52]: rural\_df

[52]:	Population	poor_health	poor_sleep	Uninsured_health	Life_Expectancy \
2	0.362595	0.247200	0.336837	0.376183	0.314607
4	0.476844	0.336375	0.422625	0.440566	0.337079
5	0.174157	0.082746	0.147370	0.142904	0.387640
6	0.035098	0.014448	0.029108	0.024687	0.308989
8	0.601844	0.353242	0.530850	0.528004	0.393258
9	0.138676	0.129545	0.135443	0.173573	0.325843
10	0.457318	0.265588	0.400189	0.325829	0.432584
12	0.069187	0.040854	0.060459	0.052529	0.101124
13	0.654248	0.535739	0.653776	0.427297	0.258427
16	0.232976	0.130375	0.196906	0.203263	0.387640
17	0.010005	0.004079	0.007971	0.006199	0.398876
18	0.373951	0.225702	0.321367	0.330179	0.247191
22	0.183859	0.136458	0.160949	0.153127	0.505618
23	0.027513	0.018664	0.024325	0.035998	0.398876

24	0.140431	0.186368	0.146602	0.102664	0.292135
25	0.761901	0.482718	0.752967	0.594236	0.213483
26	0.367859	0.222453	0.327508	0.305275	0.269663
27	0.955906	0.541891	0.838165	0.819902	0.410112
28	0.869271	0.687751	0.857354	0.819250	0.308989
29	0.431842	0.310659	0.412293	0.370092	0.365169
30	0.279208	0.160653	0.237882	0.221642	0.247191
32	0.259823	0.184156	0.249159	0.237847	0.202247
33	0.118061	0.052744	0.099250	0.088091	0.348315
34	0.604466	0.367897	0.572534	0.403371	0.230337
35	0.151545	0.126434	0.145480	0.203263	0.443820
36	0.853276	0.500622	0.798607	0.649918	0.337079
38	0.144001	0.099751	0.136506	0.098097	0.230337
39	0.207843	0.116826	0.185688	0.162153	0.342697
40	0.054139	0.056270	0.053551	0.069386	0.213483
41	0.064910	0.041477	0.054732	0.049701	0.269663
..	...	...	...	...	...
210	0.509642	0.384764	0.515085	0.557151	0.179775
211	0.059041	0.037951	0.053138	0.080479	0.460674
213	0.178796	0.109775	0.162189	0.150952	0.303371
215	0.187208	0.128785	0.175120	0.192061	0.235955
216	0.023378	0.017766	0.022259	0.023056	0.398876
217	0.024407	0.014724	0.021373	0.021968	0.398876
218	0.072737	0.047422	0.064415	0.070038	0.533708
219	0.147450	0.127748	0.139104	0.132028	0.219101
222	0.013535	0.012650	0.012222	0.014682	0.398876
223	0.244776	0.203926	0.233867	0.255900	0.202247
224	0.027493	0.016107	0.022849	0.030016	0.398876
225	0.663244	0.518941	0.640491	0.757042	0.320225
228	0.294255	0.204065	0.270060	0.235454	0.044944
229	0.434565	0.254251	0.384661	0.273953	0.213483
230	0.829192	0.459422	0.765425	0.661120	0.174157
231	0.070982	0.053436	0.066127	0.073844	0.348315
232	0.538446	0.500000	0.511189	0.536378	0.398876
233	0.989511	1.000000	0.940958	1.000000	0.404494
238	0.233339	0.168257	0.214678	0.222295	0.162921
239	0.705099	0.438891	0.628624	0.552909	0.466292
241	0.836433	0.660653	0.792171	0.810223	0.280899
242	0.101642	0.066639	0.090630	0.113214	0.146067
244	0.255527	0.184571	0.234398	0.241218	0.230337
245	0.430914	0.555855	0.448899	0.363567	0.325843
248	0.152655	0.125190	0.142469	0.150843	0.151685
250	0.907233	0.496613	0.777588	0.688961	0.286517
251	0.170223	0.131273	0.156698	0.195541	0.353933
252	0.360921	0.224043	0.308791	0.353888	0.140449
253	0.283161	0.347712	0.287359	0.388146	0.359551
254	0.238644	0.337965	0.244081	0.199891	0.219101



	Physicians	food_Insecure	Cost_Burden
2	0.236842	0.147590	0.138311
4	0.236842	0.410643	0.482252
5	0.131579	0.122490	0.098327
6	0.000000	0.024096	0.026520
8	0.157895	0.430723	0.403509
9	0.078947	0.079317	0.045288
10	0.157895	0.309237	0.251326
12	0.131579	0.058233	0.028968
13	0.184211	0.454819	0.517340
16	0.131579	0.144578	0.174215
17	0.000000	0.007028	0.000816
18	0.263158	0.279116	0.263566
22	0.157895	0.112450	0.168095
23	0.000000	0.020080	0.010608
24	0.026316	0.106426	0.139535
25	0.500000	0.599398	0.508772
26	0.184211	0.284137	0.218278
27	0.578947	0.611446	0.641779
28	0.315789	0.508032	0.686659
29	0.289474	0.272088	0.255406
30	0.026316	0.223896	0.220726
32	0.289474	0.231928	0.267646
33	0.026316	0.076305	0.032640
34	0.210526	0.635542	0.420645
35	0.026316	0.074297	0.068951
36	0.105263	0.596386	0.630763
38	0.263158	0.127510	0.095471
39	0.105263	0.166667	0.129743
40	0.026316	0.037149	0.018768
41	0.026316	0.050201	0.046920
..	...	...	...
210	0.131579	0.529116	0.354549
211	0.000000	0.031124	0.025296
213	0.157895	0.147590	0.134231
215	0.078947	0.155622	0.115055
216	0.000000	0.012048	0.005304
217	0.026316	0.015060	0.008568
218	0.078947	0.030120	0.025296
219	0.026316	0.113454	0.096695
222	0.000000	0.008032	0.011832
223	0.105263	0.145582	0.157895
224	0.026316	0.025100	0.019176
225	0.473684	0.500000	0.383517
228	0.052632	0.280120	0.213790
229	0.131579	0.415663	0.268054

230	0.315789	0.728916	0.542636
231	0.052632	0.042169	0.031824
232	0.315789	0.242972	0.303550
233	0.368421	0.458835	0.643003
238	0.078947	0.115462	0.132191
239	0.552632	0.586345	0.597715
241	0.421053	0.626506	0.719298
242	0.052632	0.090361	0.096695
244	0.157895	0.213855	0.170951
245	0.210526	0.307229	0.234190
248	0.052632	0.081325	0.082823
250	0.552632	0.723896	0.858833
251	0.078947	0.060241	0.048960
252	0.315789	0.282129	0.251326
253	0.026316	0.161647	0.187271
254	0.052632	0.172691	0.133415

[186 rows x 8 columns]

### 0.9.3 Hypothesis Test

```
[66]: # =====
# Chapter 9 Test on Hypothesis.
# Normality Test
# =====
# # Shapiro-Wilk Test
# # Sample has a Gaussian Distribution
# =====
# create same sample size

s_rural = rural_df.sample(n=60, random_state=1)
s_rural.reset_index(drop=True, inplace=True)

s_urban = urban_df.sample(n=60, random_state=1)
s_urban.reset_index(drop=True, inplace=True)

stat, p = shapiro(rural_df)
print(f"Shapiro-Wilk Test - Rural \n")
print(f'stat= {stat} p = {p} \n')
if p > 0.05:
    print('Probably Gaussian \n')
else:
    print('Probably not Gaussian \n')

print(f"Shapiro-Wilk Test - Urban \n")
stat, p = shapiro(urban_df)
```

```

print(f'stat= {stat} p = {p} \n')

if p > 0.05:
    print('Probably Gaussian \n')
else:
    print('Probably not Gaussian \n')

# =====
# #-D'Agostinos K^2 test
# # p > 0.05
# =====
stat, p = normaltest(rural_df)

print(f"D'Agostinos K^2 test - Rural \n")
print(f'stat= {stat}\n')
print(f'p = {p} \n')

stat, p = normaltest(urban_df)

print(f"D'Agostinos K^2 test - Urban \n")
print(f'stat= {stat}\n')
print(f'p = {p} \n')

# =====
# Check the means of the samples
# H0 = mean = mean independent and identical distribution
# show the how significant the difference is between the means. Meaning
# could the differences have happened by chance. Larger the t-score the more
# difference between the groups.
# https://towardsdatascience.com/
#   ↳inferential-statistics-series-t-test-using-numpy-2718f8f9bf2f
# A t-score of 3 means the groups are 3x as different from each other
# small t = more similar the groups are.
# low p = data did not occur by chance baseline = .05%
# =====
t, p = ttest_ind(rural_df, urban_df)
print( "Student's t-test Rural and Urban \n")

print(f't= {t}\n')
print(f'p = {p} \n')

# =====
# # Calculate CI for difference of the means
# =====
ci = sms.CompareMeans(sms.DescrStatsW(rural_df), sms.DescrStatsW(urban_df))
print('Difference between the means CI')
print(ci.tconfint_diff(usevar='unequal'), '\n')

```

```
# =====
# Paired Student's t-test
# =====

t, p = ttest_rel(s_rural, s_urban)
print( "Paired Student's t-test Rural and Urban \n")
print(f't= {t}\n')
print(f'p = {p} \n')
```

Shapiro-Wilk Test - Rural

stat= 0.8858976364135742 p = 9.19647064970336e-32

Probably not Gaussian

Shapiro-Wilk Test - Urban

stat= 0.6095304489135742 p = 3.1120191727302785e-33

Probably not Gaussian

D'Agostinos K<sup>2</sup> test - Rural

stat= [25.79760919 44.65363099 26.77023061 29.80436106 77.21659653 96.44366863  
35.93508877 56.87876197]

p = [2.50103829e-06 2.01181125e-10 1.53786525e-06 3.37337971e-07  
1.70855557e-17 1.14162494e-21 1.57323870e-08 4.45588290e-13]

D'Agostinos K<sup>2</sup> test - Urban

stat= [89.55656942 84.98611489 91.2761902 99.01825946 0.70700863 77.26905094  
98.02255886 96.24116205]

p = [3.57304133e-20 3.51155627e-19 1.51226233e-20 3.15107074e-22  
7.02222963e-01 1.66432746e-17 5.18408116e-22 1.26327295e-21]

Student's t-test Rural and Urban

t= [ 6.79323141 5.31725255 6.55877629 6.79038486 -7.51255979 2.35885894  
5.56395097 5.00468522]

p = [7.85440886e-11 2.32278625e-07 3.05527250e-10 7.98651687e-11  
1.00808010e-12 1.90950891e-02 6.72848316e-08 1.05122247e-06]

Difference between the means CI

```
(array([ 0.17118857,  0.08771362,  0.15333928,  0.15249002, -0.23831231,
        0.00776219,  0.10346571,  0.09001562]), array([ 0.27584289,  0.18159802,
        0.25437533,  0.24960902, -0.11775222,
        0.10162928,  0.19629283,  0.18396512]))
```

Paired Student's t-test Rural and Urban

```
t= [ 6.79875785  5.51777929  6.56017156  6.5975982 -7.00952344  2.69525087
     5.53527142  5.12666291]
```

```
p = [5.92225149e-09 8.03380888e-07 1.49548235e-08 1.29351795e-08
     2.60678077e-09 9.14884204e-03 7.52365180e-07 3.42963082e-06]
```

```
[ ]: t, p = ttest_rel(s_rural, s_urban)
```

```
[67]: # =====
# Conduct linear regression analysis
# Using life expectancy as the reponse how do
# poor health, lack of sleep, no insuranceand availability of Physicians
# affect LE. Use poor_health as the response variable, how do the
# factors affect poor_health.
# =====
# =====
# For this project, conduct a regression analysis on either one dependent and
# one explanatory variable, or multiple explanatory variables (Chapter 10 & 11)
# =====
# =====
# Create two scatter plots comparing two variables and provide your analysis
# on correlation and causation. Remember, covariance, Pearson's correlation,
# and Non- Linear Relationships should also be considered during your analysis
# (Chapter 7).
# =====
# =====
# How do the factors affect life expectancy
# =====
```

```
[55]: # Rename variables
life = s_rural.Life_Expectancy
health = s_rural.poor_health
sleep = s_rural.poor_sleep
uninsured = s_rural.Uninsured_health
docs = s_rural.Physicians
food = s_rural.food_Insecure
burden = s_rural.Cost_Burden

life1 = s_urban.Life_Expectancy
```

```

health1 = s_urban.poor_health
sleep1 = s_urban.poor_sleep
uninsured1 = s_urban.Uninsured_health
docs1 = s_urban.Physicians
food1 = s_urban.food_Insecure
burden1 = s_urban.Cost_Burden

# Fit the models
# Rural
mod1 = smf.ols('life ~ health + sleep + uninsured + docs\
               + food + burden',
               data=s_rural).fit()
print('RURAL', '\n')
print(mod1.summary(), '\n')

# Urban
mod2 = smf.ols('life1 ~ health1 + sleep1 + uninsured1 + docs1\
               + food1 + burden1',
               data=s_urban).fit()
print('-URBAN', '\n')
print(mod2.summary(), '\n')

```

## RURAL

### OLS Regression Results

=====						
Dep. Variable:	life	R-squared:	0.212			
Model:	OLS	Adj. R-squared:	0.123			
Method:	Least Squares	F-statistic:	2.376			
Date:	Fri, 29 May 2020	Prob (F-statistic):	0.0417			
Time:	08:32:57	Log-Likelihood:	49.587			
No. Observations:	60	AIC:	-85.17			
Df Residuals:	53	BIC:	-70.51			
Df Model:	6					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
-----						
Intercept	0.2873	0.024	12.035	0.000	0.239	0.335
health	-0.0988	0.396	-0.249	0.804	-0.894	0.696
sleep	0.0571	0.487	0.117	0.907	-0.920	1.034
uninsured	0.1549	0.314	0.494	0.623	-0.474	0.784
docs	0.3275	0.200	1.634	0.108	-0.074	0.729
food	-0.4867	0.362	-1.344	0.185	-1.213	0.240
burden	0.0758	0.341	0.223	0.825	-0.608	0.759
=====						
Omnibus:	2.409	Durbin-Watson:	2.265			

```

Prob(Omnibus):          0.300    Jarque-Bera (JB):          1.619
Skew:                   0.220    Prob(JB):             0.445
Kurtosis:                3.674    Cond. No.              54.7
=====

```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- URBAN

#### OLS Regression Results

```

=====
Dep. Variable:          life1    R-squared:                0.492
Model:                  OLS      Adj. R-squared:           0.434
Method:                  Least Squares    F-statistic:             8.541
Date:                    Fri, 29 May 2020    Prob (F-statistic):       1.67e-06
Time:                    08:32:57    Log-Likelihood:          24.084
No. Observations:        60    AIC:                     -34.17
Df Residuals:            53    BIC:                     -19.51
Df Model:                 6
Covariance Type:         nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.4333	0.029	14.940	0.000	0.375	0.491
health1	0.0309	1.951	0.016	0.987	-3.883	3.944
sleep1	4.2851	2.546	1.683	0.098	-0.822	9.393
uninsured1	0.4955	1.930	0.257	0.798	-3.375	4.366
docs1	3.2563	0.914	3.562	0.001	1.422	5.090
food1	-8.2411	2.112	-3.902	0.000	-12.477	-4.005
burden1	0.4006	1.819	0.220	0.827	-3.249	4.050

```

=====
Omnibus:                0.808    Durbin-Watson:           2.328
Prob(Omnibus):          0.668    Jarque-Bera (JB):        0.555
Skew:                   -0.236    Prob(JB):                0.758
Kurtosis:               2.998    Cond. No.                161.
=====

```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```

[56]: # =====
      # How do the factors affect health
      # =====

```

```
# Rural
mod2 = smf.ols('health ~ life + sleep + uninsured + docs\
               + food + burden', data=s_rural).fit()
print('                RURAL', '\n')
print(mod2.summary(), '\n')

# Urban
mod2 = smf.ols('health1 ~ life1 + sleep1 + uninsured1 + docs1\
               + food1 + burden1', data=s_urban).fit()
print('                URBAN', '\n')
print(mod2.summary(), '\n')
```

## RURAL

### OLS Regression Results

Dep. Variable:	health	R-squared:	0.962			
Model:	OLS	Adj. R-squared:	0.958			
Method:	Least Squares	F-statistic:	226.4			
Date:	Fri, 29 May 2020	Prob (F-statistic):	6.13e-36			
Time:	08:32:57	Log-Likelihood:	113.20			
No. Observations:	60	AIC:	-212.4			
Df Residuals:	53	BIC:	-197.7			
Df Model:	6					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
-----						
Intercept	0.0134	0.016	0.843	0.403	-0.018	0.045
life	-0.0119	0.048	-0.249	0.804	-0.107	0.084
sleep	0.7822	0.130	6.010	0.000	0.521	1.043
uninsured	0.3364	0.099	3.412	0.001	0.139	0.534
docs	-0.2776	0.060	-4.621	0.000	-0.398	-0.157
food	-0.5702	0.101	-5.662	0.000	-0.772	-0.368
burden	0.2940	0.111	2.648	0.011	0.071	0.517
=====						
Omnibus:	7.787	Durbin-Watson:	2.153			
Prob(Omnibus):	0.020	Jarque-Bera (JB):	8.483			
Skew:	0.546	Prob(JB):	0.0144			
Kurtosis:	4.483	Cond. No.	41.8			
=====						

#### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

## URBAN



### OLS Regression Results

```

=====
Dep. Variable:          health1    R-squared:                0.995
Model:                  OLS        Adj. R-squared:            0.994
Method:                 Least Squares    F-statistic:              1720.
Date:                  Fri, 29 May 2020    Prob (F-statistic):       7.26e-59
Time:                  08:32:57    Log-Likelihood:           183.30
No. Observations:      60    AIC:                      -352.6
Df Residuals:          53    BIC:                      -337.9
Df Model:               6
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.0067	0.005	1.473	0.147	-0.002	0.016
life1	0.0002	0.010	0.016	0.987	-0.019	0.020
sleep1	0.9546	0.129	7.396	0.000	0.696	1.214
uninsured1	0.7020	0.096	7.328	0.000	0.510	0.894
docs1	-0.0116	0.072	-0.162	0.872	-0.155	0.132
food1	-0.5735	0.149	-3.845	0.000	-0.873	-0.274
burden1	-0.0463	0.128	-0.362	0.719	-0.303	0.210

```

=====
Omnibus:                12.072    Durbin-Watson:            1.818
Prob(Omnibus):           0.002    Jarque-Bera (JB):         23.230
Skew:                    0.543    Prob(JB):                  9.03e-06
Kurtosis:                5.848    Cond. No.                  140.
=====

```

#### Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

## 0.9.4 Linear Relationships

[ ]:

```

[57]: # Linear Relationship
      subtitle = 'Rural'
      x_val = "Life_Expectancy"
      y_val = 'poor_health'
      data = s_rural
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

      subtitle = 'Urban'
      x_val = "Life_Expectancy"

```

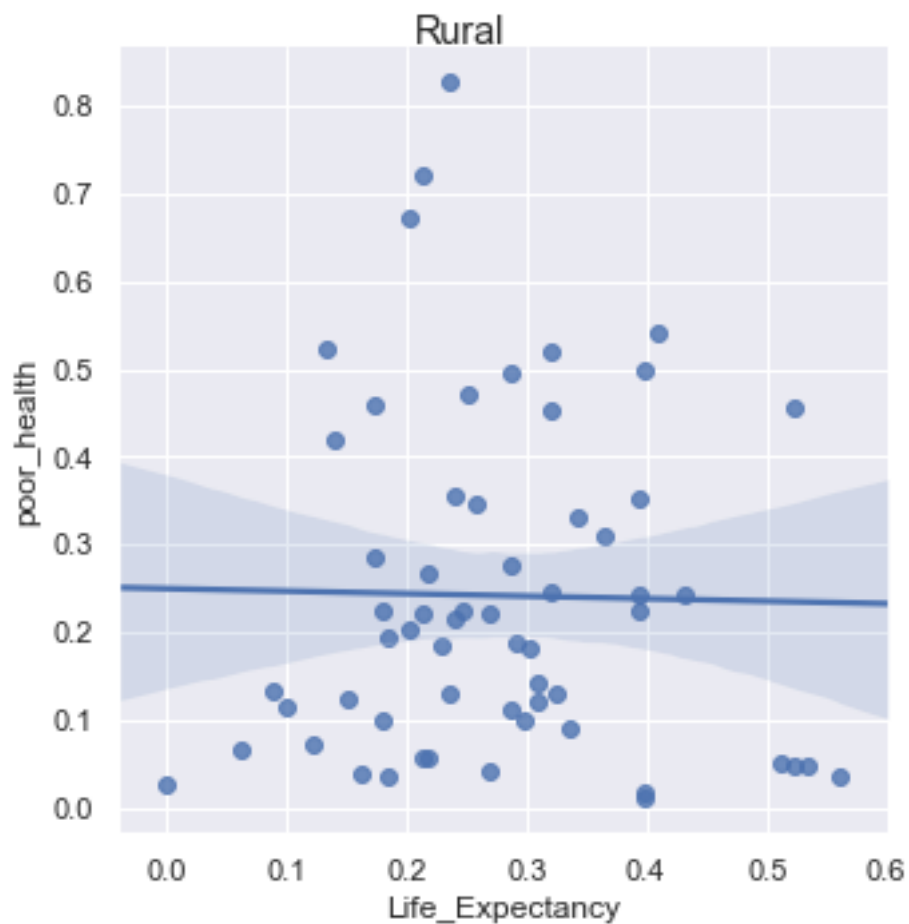
```

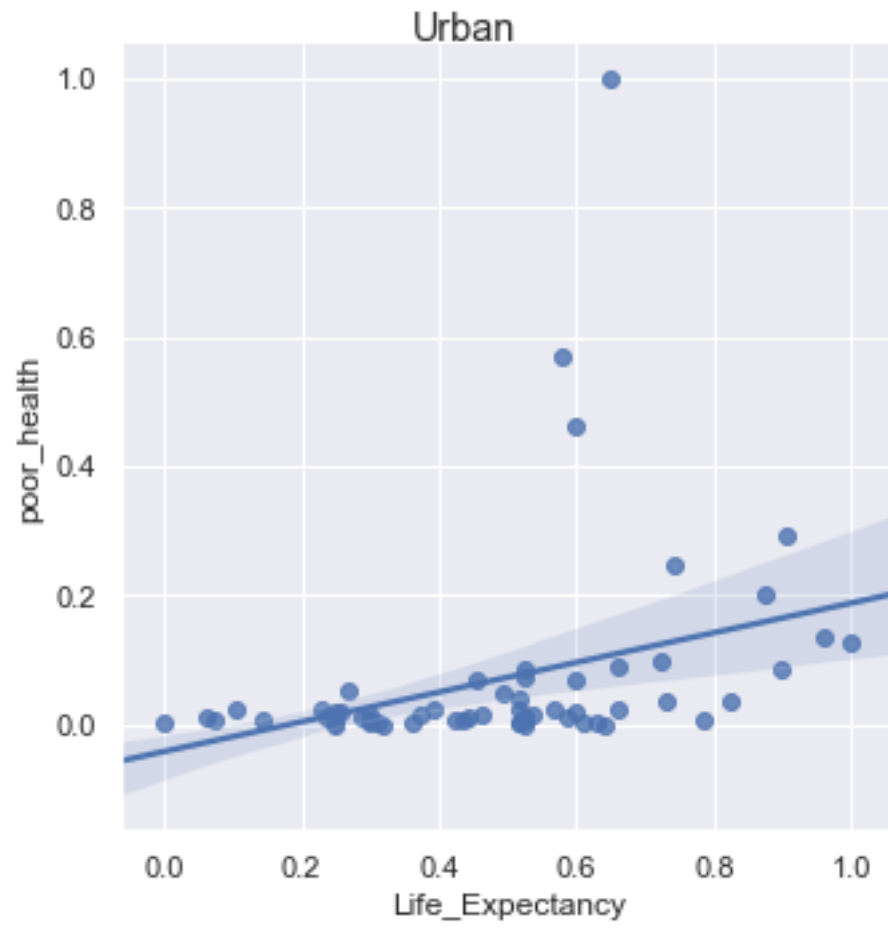
y_val = 'poor_health'
data = s_urban
fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

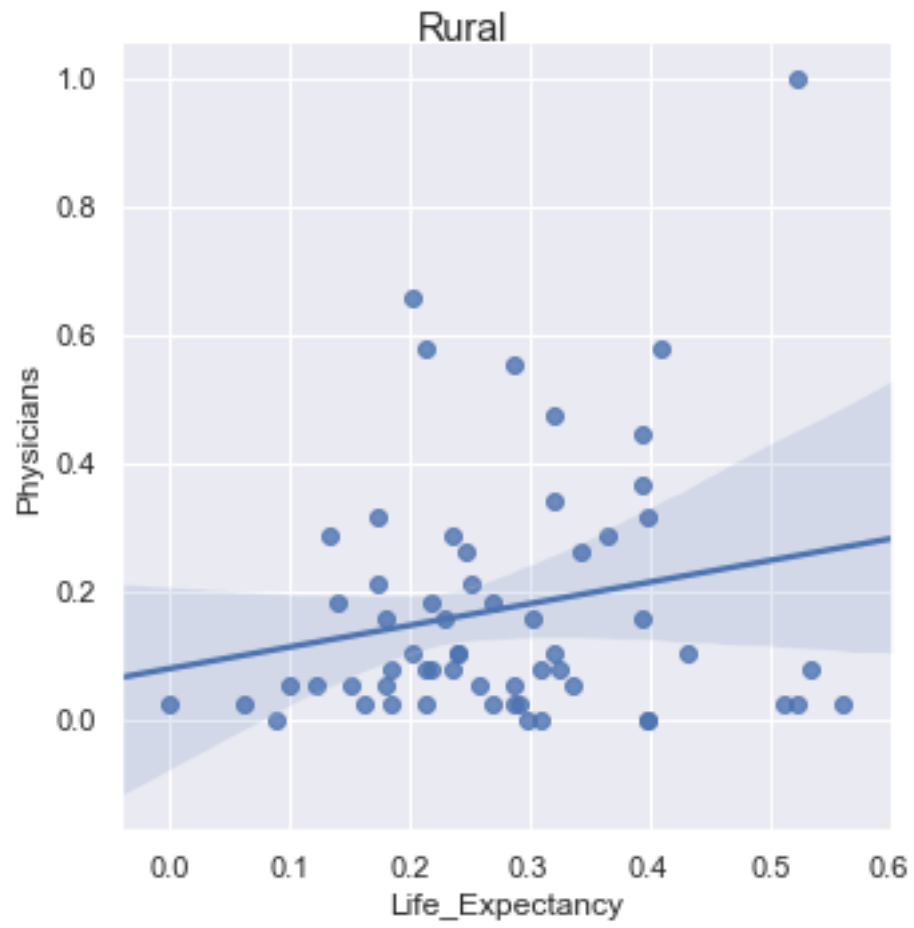
subtitle = 'Rural'
x_val = "Life_Expectancy"
y_val = 'Physicians'
data = s_rural
fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

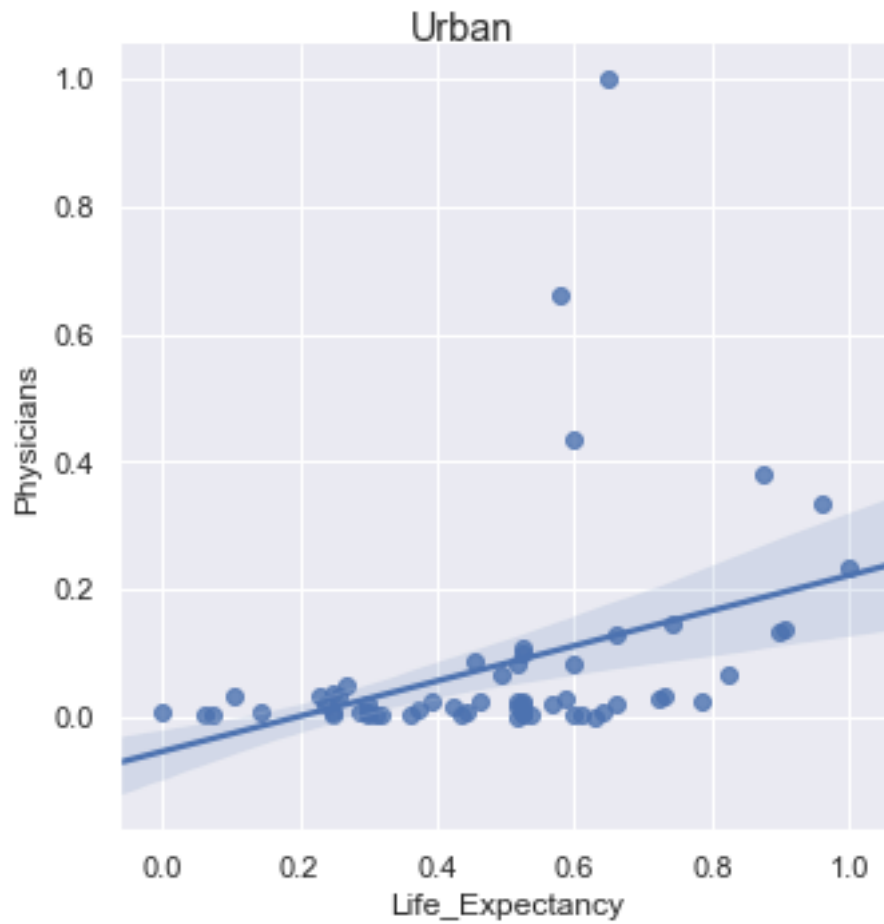
subtitle = 'Urban'
x_val = "Life_Expectancy"
y_val = 'Physicians'
data = s_urban
fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

```



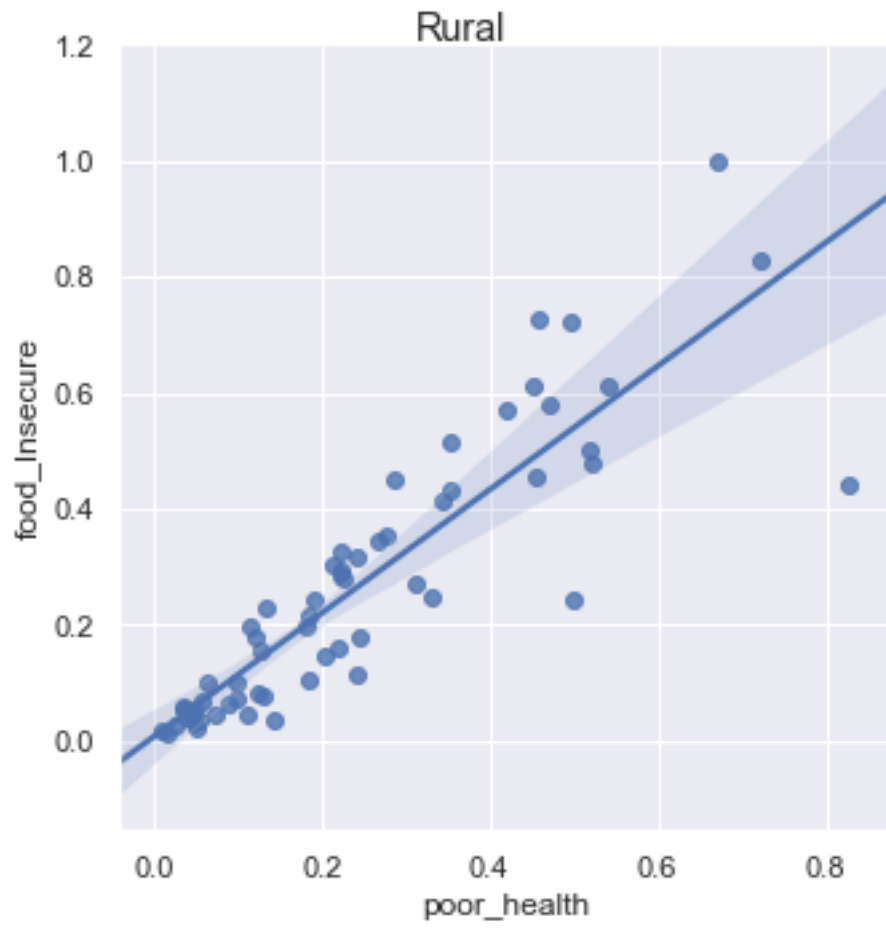


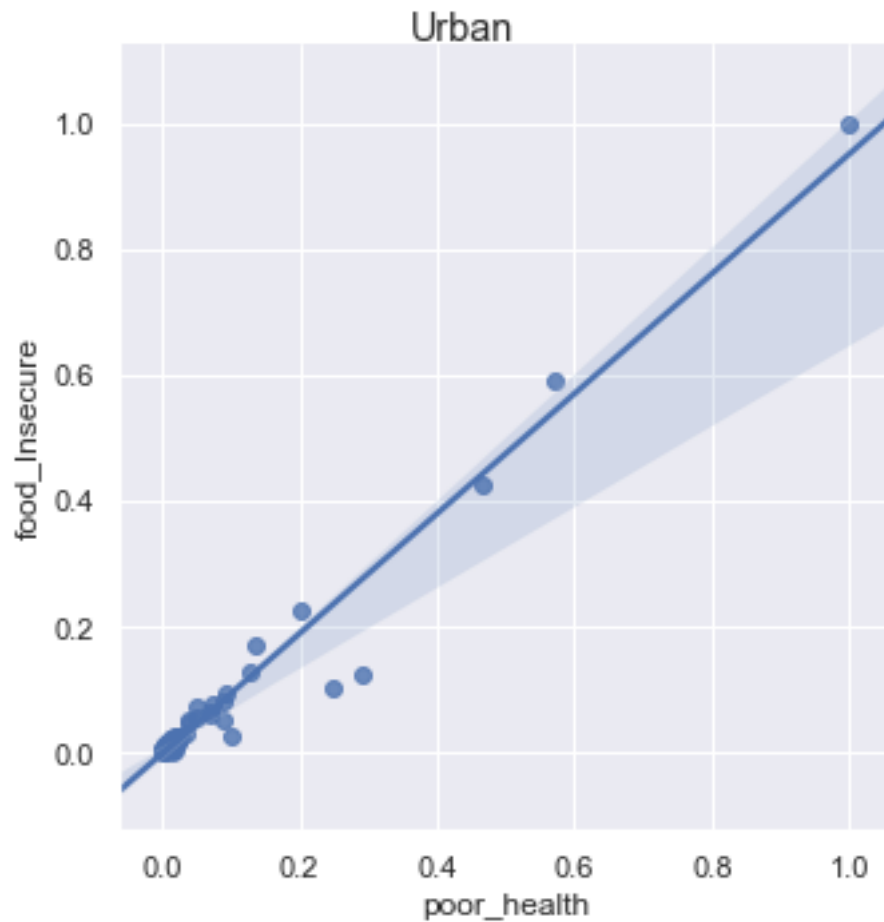




```
[58]: # Linear Relationship
      subtitle = 'Rural'
      x_val = 'poor_health'
      y_val = 'food_Insecure'
      data = s_rural
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

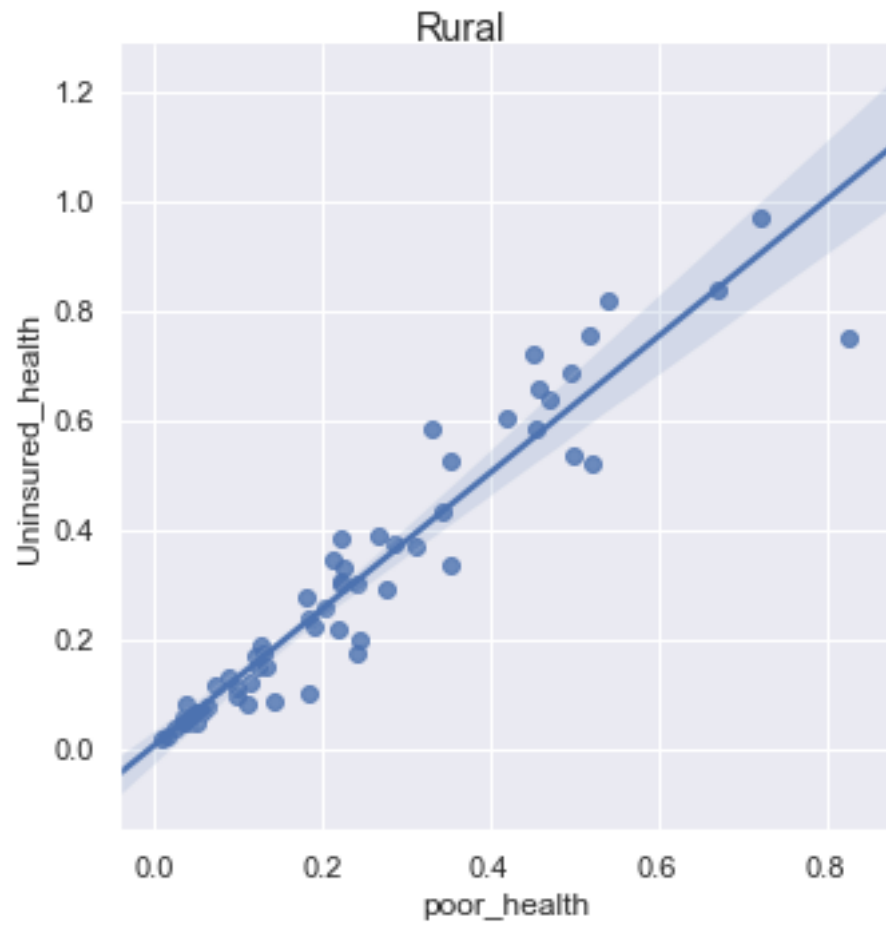
      subtitle = 'Urban'
      x_val = 'poor_health'
      y_val = 'food_Insecure'
      data = s_rural
      data = s_urban
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)
```



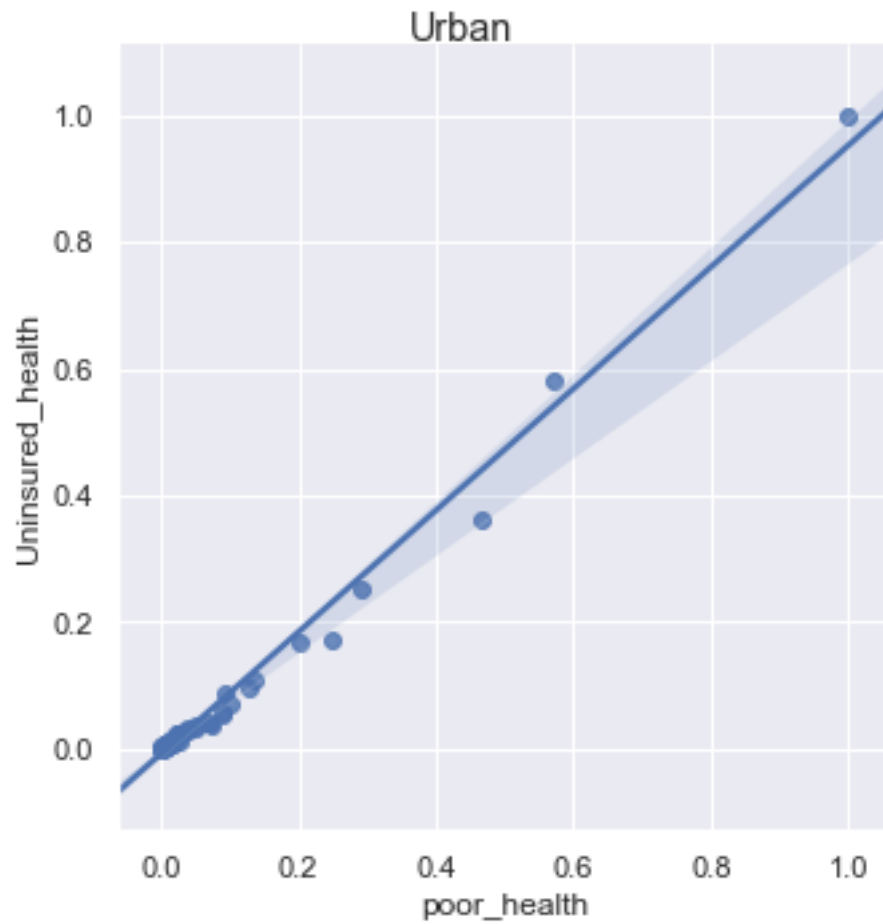


```
[59]: # Linear Relationship
      subtitle = 'Rural'
      x_val = 'poor_health'
      y_val = 'Uninsured_health'
      data = s_rural
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

      subtitle = 'Urban'
      x_val = 'poor_health'
      y_val = 'Uninsured_health'
      data = s_rural
      data = s_urban
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)
```



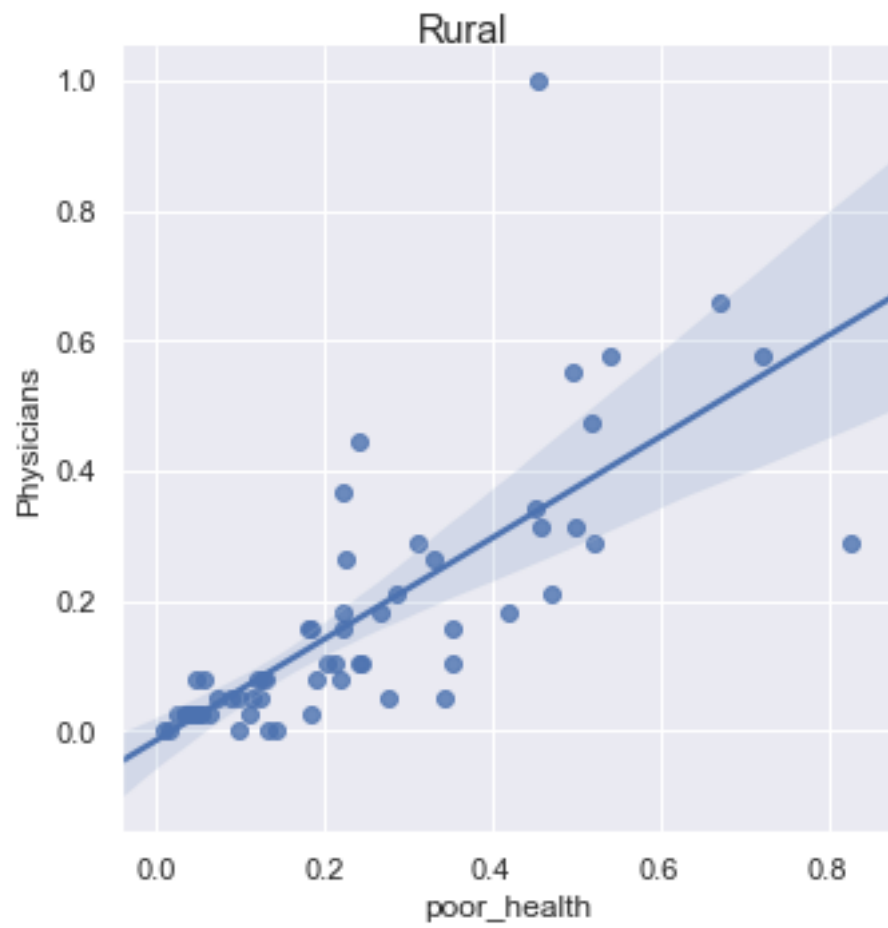


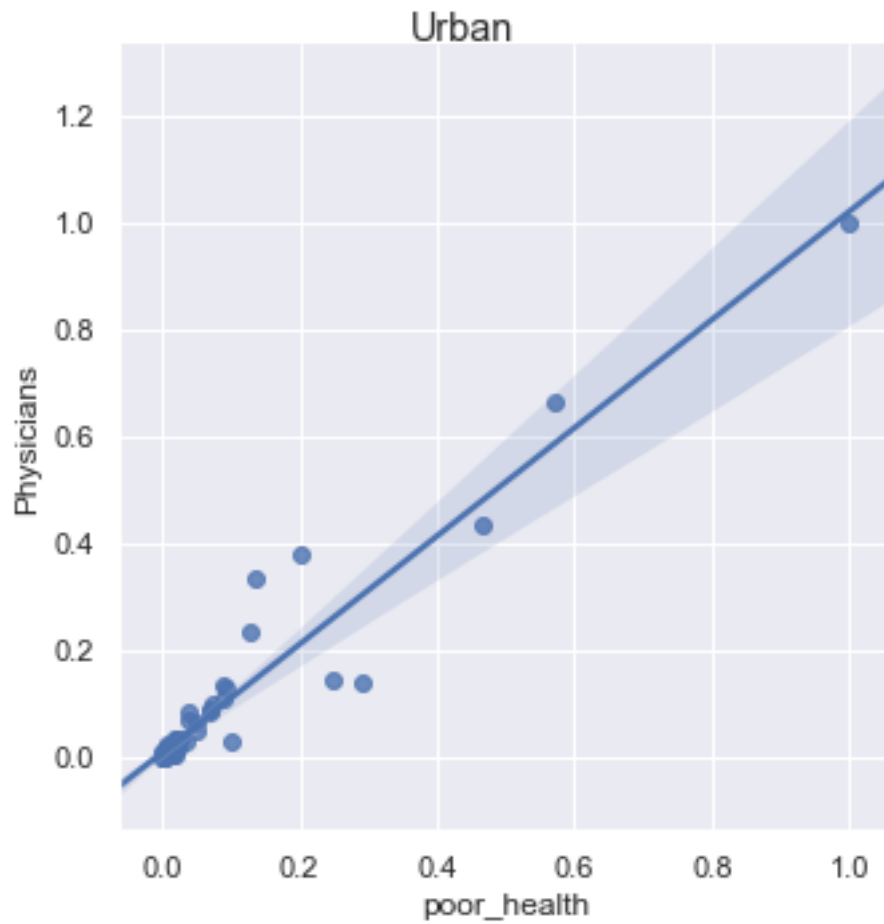


```
[60]: # Linear Relationship
      subtitle = 'Rural'
      x_val = 'poor_health'
      y_val = 'Physicians'
      data = s_rural
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

      subtitle = 'Urban'
      x_val = 'poor_health'
      y_val = 'Physicians'
      data = s_urban

      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)
```

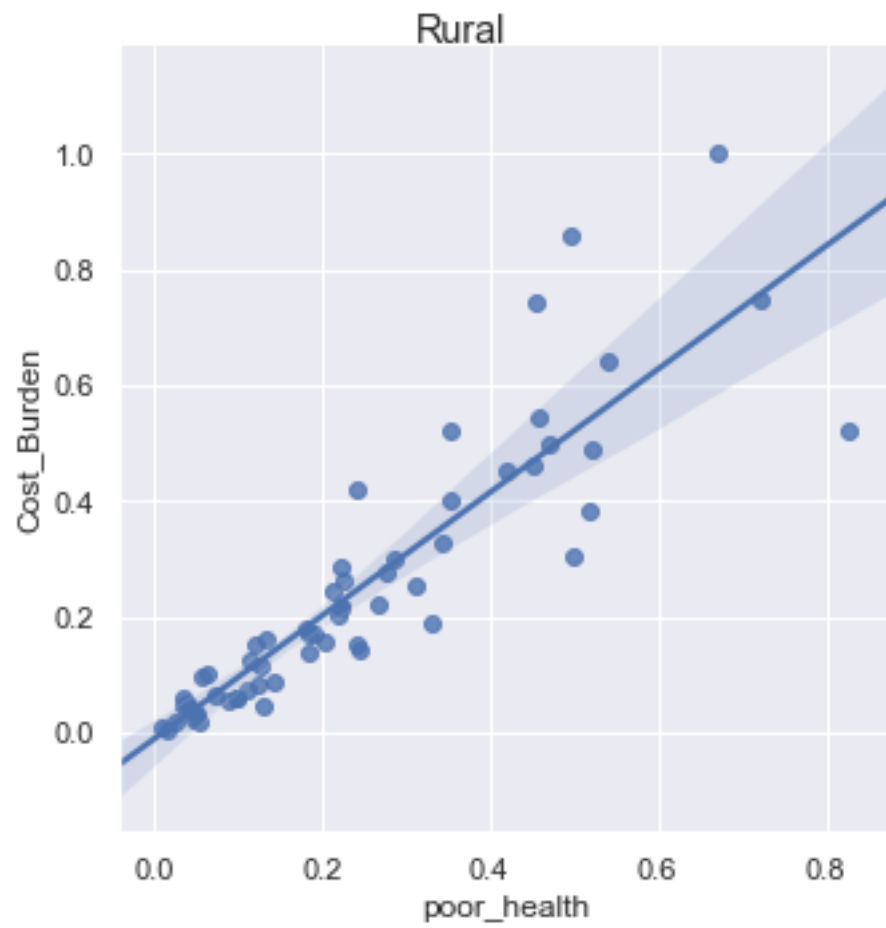


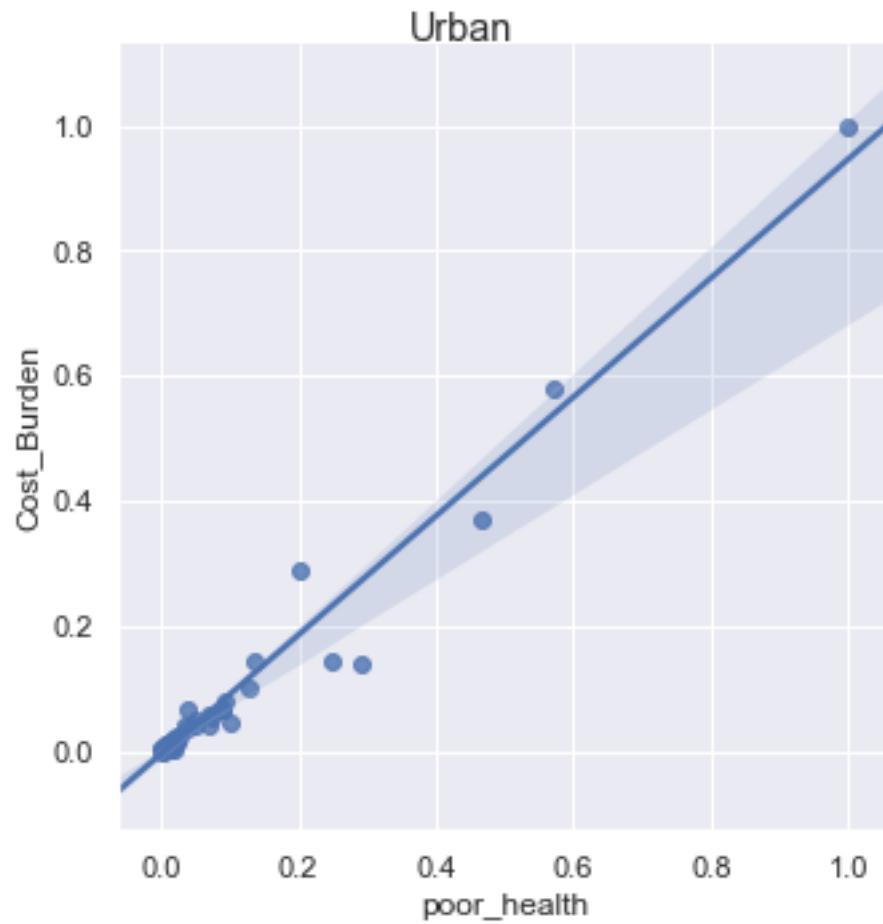


```
[61]: # Linear Relationship
      subtitle = 'Rural'
      x_val = 'poor_health'
      y_val = 'Cost_Burden'
      data = s_rural
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)

      subtitle = 'Urban'
      x_val = 'poor_health'
      y_val = 'Cost_Burden'
      data = s_urban

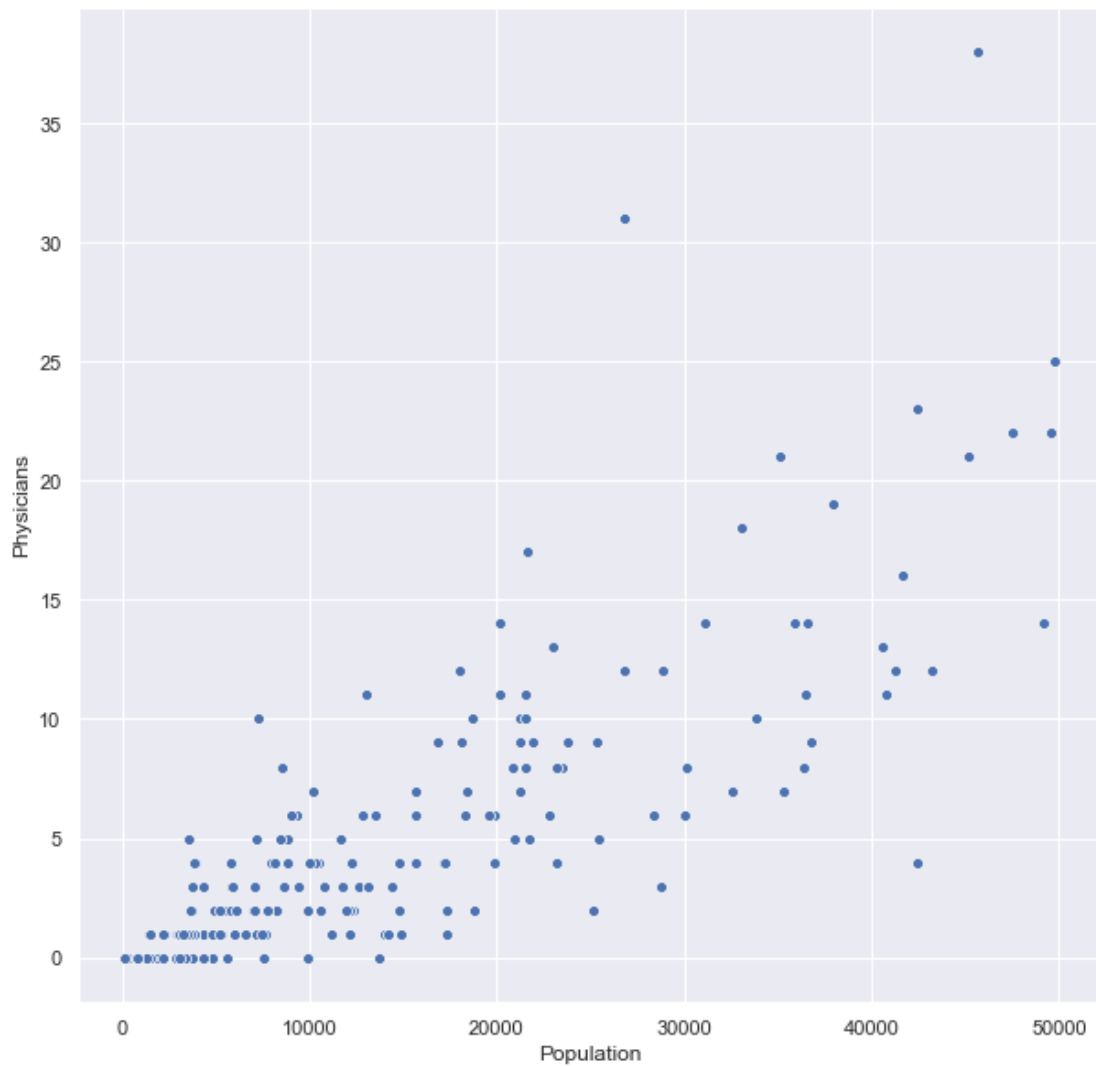
      fig = sns.lmplot(x_val, y_val, data).fig.suptitle(subtitle)
```





```
[62]: subtitle = 'Number of Doctors Per County Population - Rural'
x_val = 'Population'
y_val = 'Physicians'
xlab = 'Population per County'
ylab = '# of Physicians'
data = rural
sns_Scatter(subtitle, xlab, ylab, x_val, y_val, data)
```

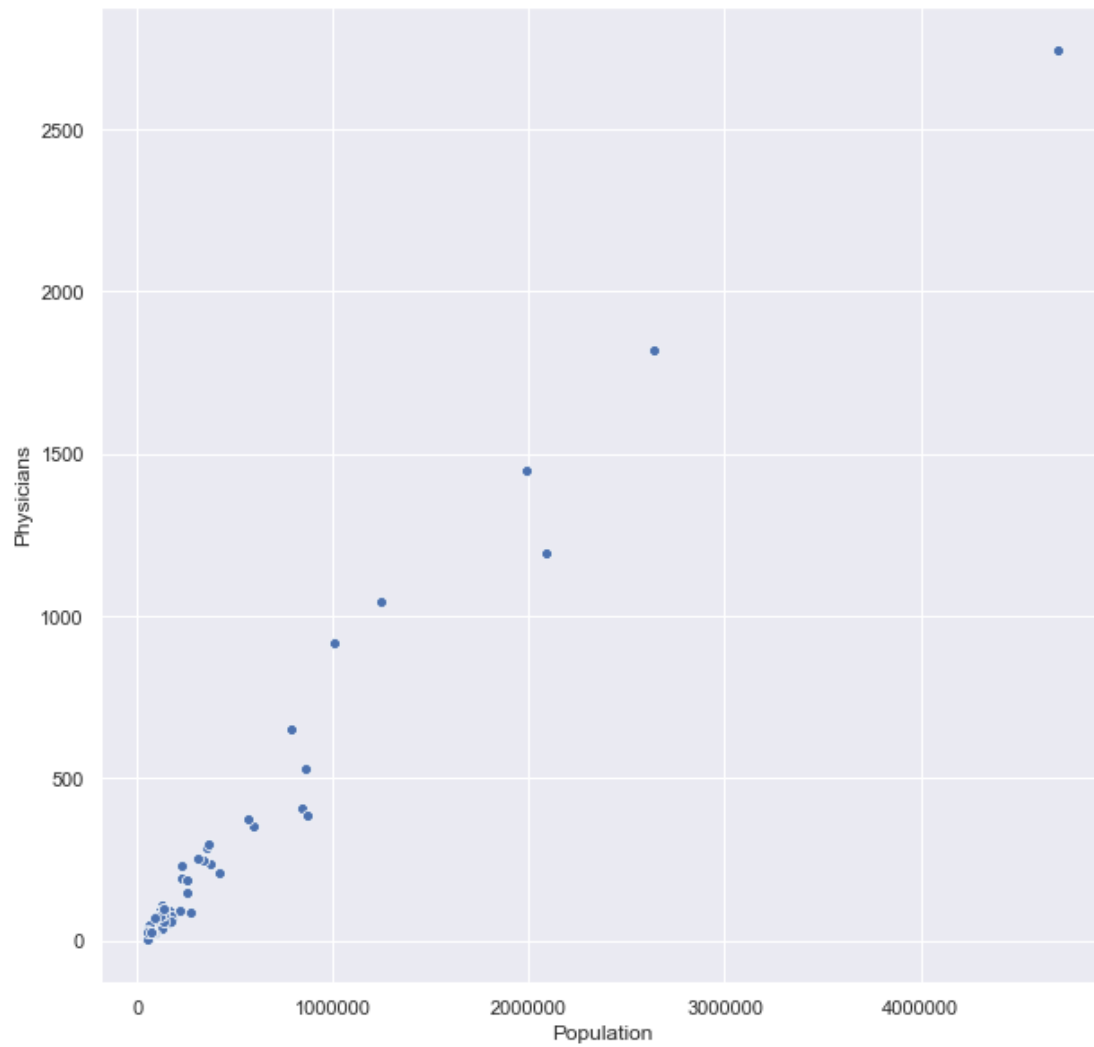
Number of Doctors Per County Population - Rural



[ ]:

```
[63]: subtitle = 'Number of Doctors Per County Population - urban'
x_val = 'Population'
y_val = 'Physicians'
xlab = 'Population per County'
ylab = '# of Physicians'
data = urban
sns_Scatter(subtitle, xlab, ylab, x_val, y_val, data)
```

Number of Doctors Per County Population - urban



[ ]: