**Randall C. Crawford**

**Student ID: 901012194**

**Email:** [**r.crawford2194@student.nu.edu**](mailto:r.crawford2194@student.nu.edu)

**Course: ANA 620 Continuous Data Methods, Appl**

# Import required packages for this project.  
from pathlib import Path  
  
import math  
import statistics  
  
import os  
import calendar  
import numpy as np  
import networkx as nx  
import pandas as pd  
from pandas.plotting import scatter\_matrix, parallel\_coordinates  
  
import seaborn as sns  
  
from sklearn import preprocessing  
  
import scipy.stats  
from scipy import stats  
from scipy.stats import anderson  
from scipy.stats import ttest\_ind  
  
import matplotlib as mpl  
import matplotlib.pyplot as plt  
import matplotlib.patches as mpatches  
  
import statsmodels.formula.api as smf  
import statsmodels.api as sm  
import statsmodels.stats.api as sms  
from statsmodels.stats.diagnostic import het\_breuschpagan  
from statsmodels.stats.diagnostic import het\_white  
from statsmodels.graphics.gofplots import qqplot  
from statsmodels.stats.outliers\_influence import OLSInfluence  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  
  
from IPython.display import Image  
Image("img/Model\_Comparison.png")  
  
%matplotlib inline

# Established path to access data for this assignment.  
DATA = Path("C:\\Users\\rcc\_0\\OneDrive\\Documents\\ANA 620\\Assignment\_3")

# Load the required data for the first part of assigment.  
Wages\_df = pd.read\_csv(DATA / 'HW\_3\_SLIDLogWagesClean.csv')

# View some initial records.  
Wages\_df.head()

wages education age sex language logwages  
0 10.56 15.0 40 Male English 2.357073  
1 11.00 13.2 19 Male English 2.397895  
2 17.76 14.0 46 Male Other 2.876949  
3 14.00 16.0 50 Female English 2.639057  
4 8.20 15.0 31 Male English 2.104134

# Generate dataframe dimensions.  
Wages\_df.shape

(3987, 6)

# Generate variable data types.  
Wages\_df.dtypes

wages float64  
education float64  
age int64  
sex object  
language object  
logwages float64  
dtype: object

# Generate number of missing values.  
Wages\_df.isna().sum()

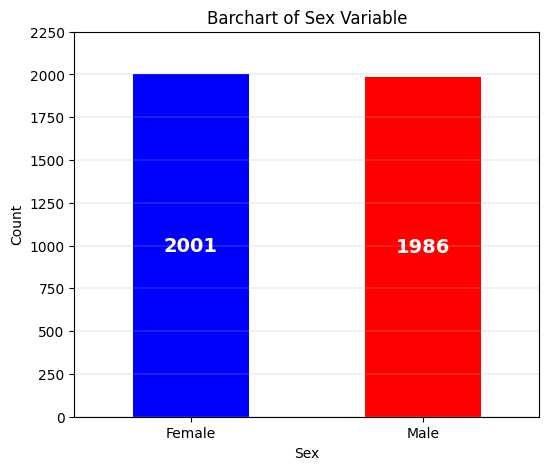
wages 0  
education 0  
age 0  
sex 0  
language 0  
logwages 0  
dtype: int64

Specific Evaluation and Preparation of Dataset Variables

# Evaluate the 'sex' categorical variable.  
Wages\_df['sex'].value\_counts()

sex  
Female 2001  
Male 1986  
Name: count, dtype: int64

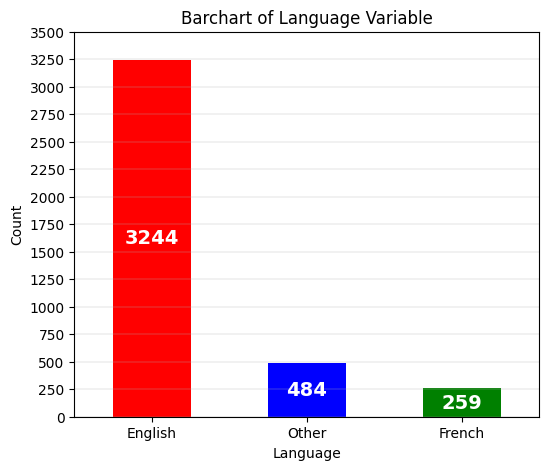
# Barchart for the 'sex' categorical variable.  
dfg = Wages\_df['sex'].value\_counts()  
ax = dfg.plot(kind='bar', title='Barchart of Sex Variable', ylabel='Count', xlabel='Sex', color= ['b','r'], rot=0, figsize=(6,5))  
ax.bar\_label(ax.containers[0], label\_type='center', color='w', fontsize=14, fontweight='bold')  
plt.yticks(np.arange(0,2251,step=250))  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



# Evaluate the 'language' categorical variable.  
Wages\_df['language'].value\_counts()

language  
English 3244  
Other 484  
French 259  
Name: count, dtype: int64

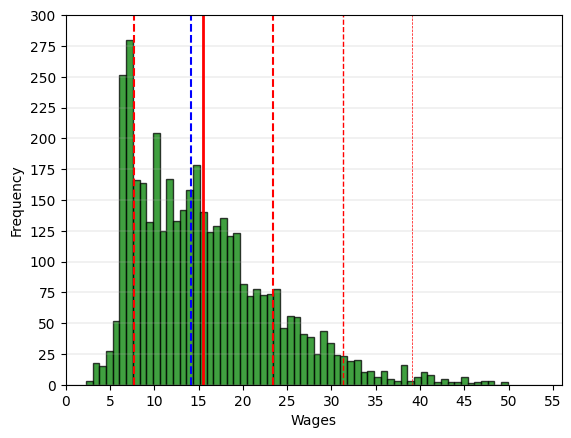
# Barchart for the 'language' categorical variable.  
dfg = Wages\_df['language'].value\_counts()  
ax = dfg.plot(kind='bar', title='Barchart of Language Variable', ylabel='Count', xlabel='Language', color= ['r','b', 'g'], rot=0, figsize=(6,5))  
ax.bar\_label(ax.containers[0], label\_type='center', color='w', fontsize=14, fontweight='bold')  
plt.yticks(np.arange(0,3501,step=250))  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



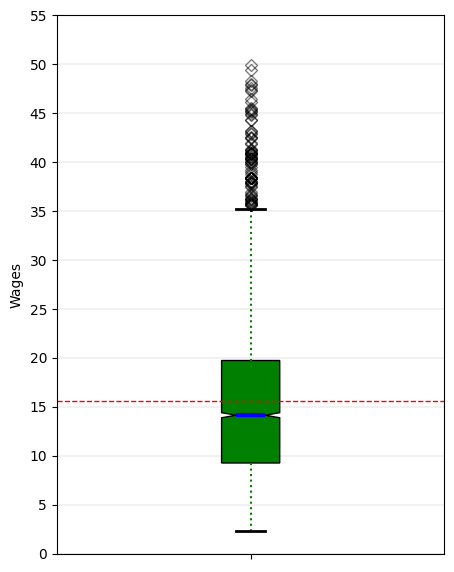
Wages\_df.describe().map('{:,.3f}'.format)

wages education age logwages  
count 3,987.000 3,987.000 3,987.000 3,987.000  
mean 15.539 13.337 37.098 2.619  
std 7.869 3.038 12.137 0.503  
min 2.300 0.000 16.000 0.833  
25% 9.250 12.000 28.000 2.225  
50% 14.130 13.000 36.000 2.648  
75% 19.720 15.100 46.000 2.982  
max 49.920 20.000 69.000 3.910

# Histogram for the 'wages' continuous variable.  
plt.hist(Wages\_df['wages'], bins = 63, alpha = 0.75, color = 'green', edgecolor = 'black')  
plt.xlabel('Wages')  
plt.ylabel('Frequency')  
plt.xlim(0, 56)  
plt.xticks(np.arange(0,56,step=5))  
plt.yticks(np.arange(0,301,step=25))  
  
mean\_value = Wages\_df['wages'].mean()  
median\_value = Wages\_df['wages'].median()  
std\_value = Wages\_df['wages'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



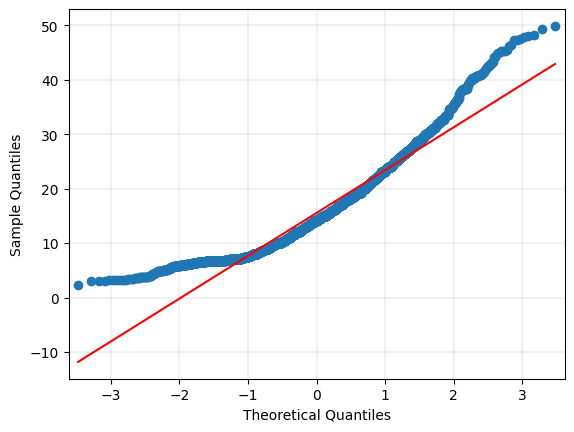
# Boxplot for the 'wages' continuous variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Wages\_df['wages'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('Wages')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(0,56,step=5))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



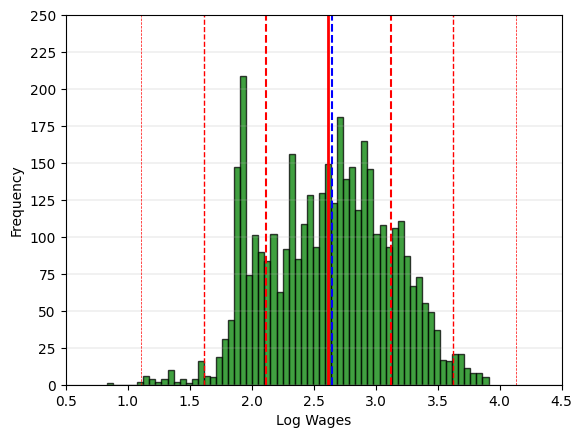
# 'wages' Anderson-Darling Normality Test  
result = anderson(Wages\_df['wages'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 70.761  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.786, data does not look normal (reject H0)  
2.500: 0.917, data does not look normal (reject H0)  
1.000: 1.091, data does not look normal (reject H0)

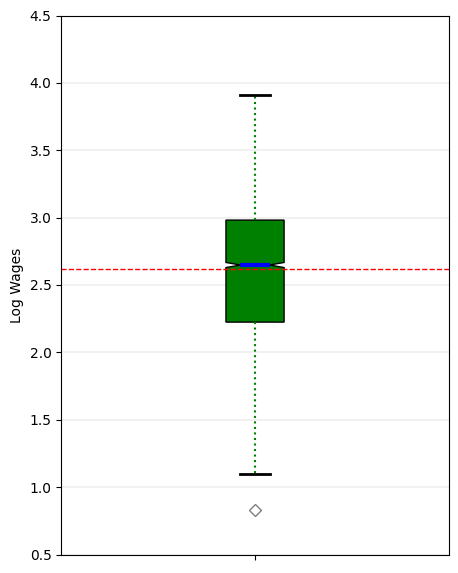
# 'wages' QQ plot  
qqplot(Wages\_df['wages'], line='s')  
plt.grid(linewidth=0.25)  
plt.show()



# Histogram for the 'logwages' continuous transformed variable.  
plt.hist(Wages\_df['logwages'], bins = 63, alpha = 0.75, color = 'green', edgecolor = 'black')  
plt.xlabel('Log Wages')  
plt.ylabel('Frequency')  
plt.xticks(np.arange(0.5,4.51,step=0.50))  
plt.yticks(np.arange(0,251,step=25))  
  
mean\_value = Wages\_df['logwages'].mean()  
median\_value = Wages\_df['logwages'].median()  
std\_value = Wages\_df['logwages'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



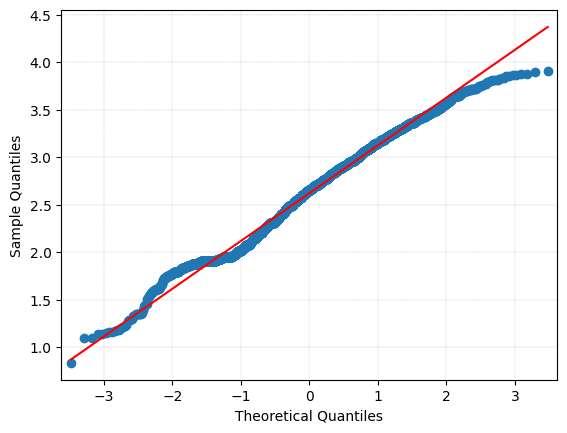
# Boxplot for the 'logwages' continuous transformed variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Wages\_df['logwages'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('Log Wages')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(0.5,4.51,step=0.50))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



# 'logwages' Anderson-Darling Normality Test  
result = anderson(Wages\_df['logwages'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 14.847  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.786, data does not look normal (reject H0)  
2.500: 0.917, data does not look normal (reject H0)  
1.000: 1.091, data does not look normal (reject H0)

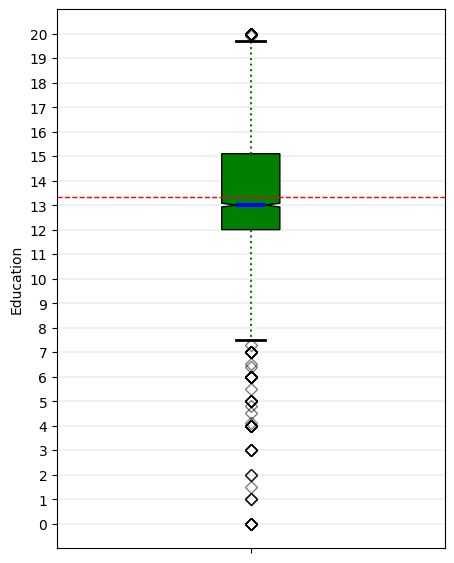
# 'logwages' QQ plot  
qqplot(Wages\_df['logwages'], line='s')  
plt.grid(linestyle='dashed', linewidth=0.25)  
plt.show()



# Histogram for the 'education' continuous variable.  
plt.hist(Wages\_df['education'], bins = 20, alpha = 0.75, color = 'green', edgecolor = 'black', align='mid')  
plt.xlabel('Edcuation')  
plt.ylabel('Frequency')  
plt.xlim(-1,21)  
plt.xticks(np.arange(0,21,step=1))  
plt.yticks(np.arange(0,1001,step=100))  
  
mean\_value = Wages\_df['education'].mean()  
median\_value = Wages\_df['education'].median()  
std\_value = Wages\_df['education'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



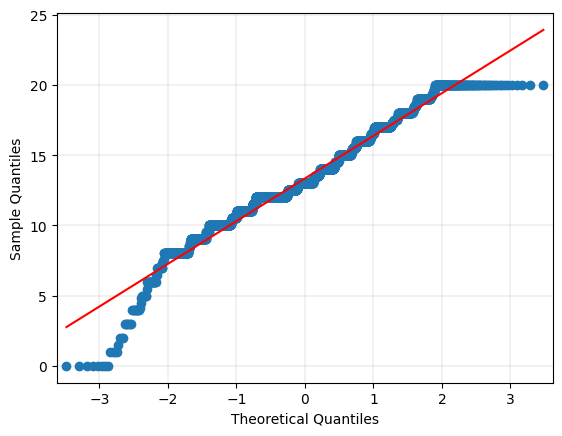
# Boxplot for the 'education' continuous variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Wages\_df['education'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('Education')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(0,21,step=1))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



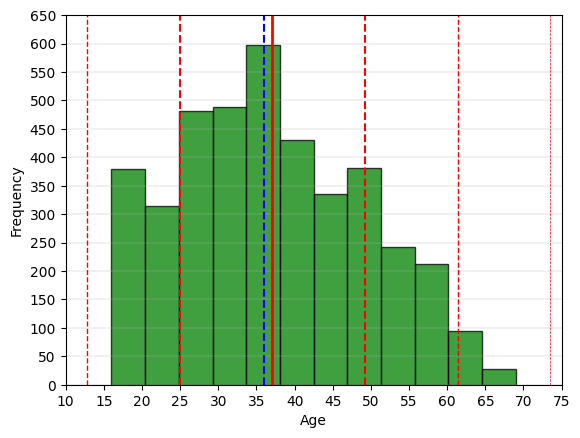
# 'education' Anderson-Darling Normality Test  
result = anderson(Wages\_df['education'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 26.697  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.786, data does not look normal (reject H0)  
2.500: 0.917, data does not look normal (reject H0)  
1.000: 1.091, data does not look normal (reject H0)

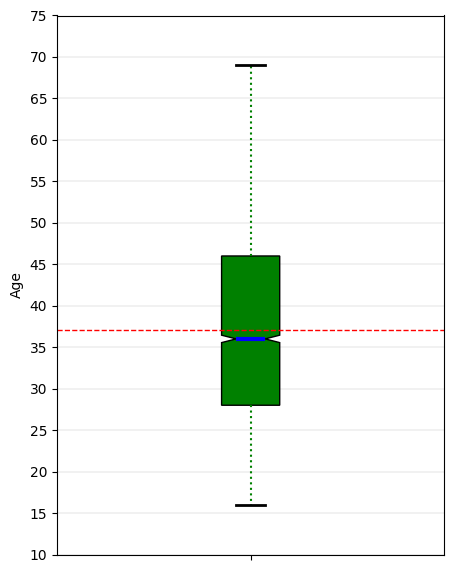
# 'education' QQ plot  
qqplot(Wages\_df['education'], line='s')  
plt.grid(linewidth=0.25)  
plt.show()



# Histogram for the 'age' continuous variable.  
plt.hist(Wages\_df['age'], bins = 12, alpha = 0.75, color = 'green', edgecolor = 'black')  
plt.xlabel('Age')  
plt.ylabel('Frequency')  
plt.xlim(10,75)  
plt.xticks(np.arange(10,76,step=5))  
plt.yticks(np.arange(0,651,step=50))  
  
mean\_value = Wages\_df['age'].mean()  
median\_value = Wages\_df['age'].median()  
std\_value = Wages\_df['age'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



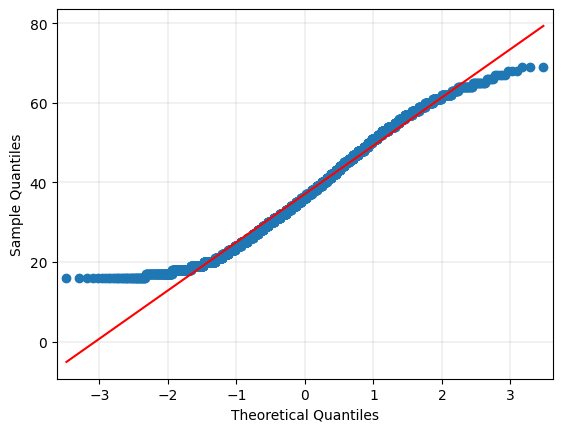
# Boxplot for the 'age' continuous variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Wages\_df['age'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('Age')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(10,76,step=5))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



# 'age' Anderson-Darling Normality Test  
result = anderson(Wages\_df['age'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

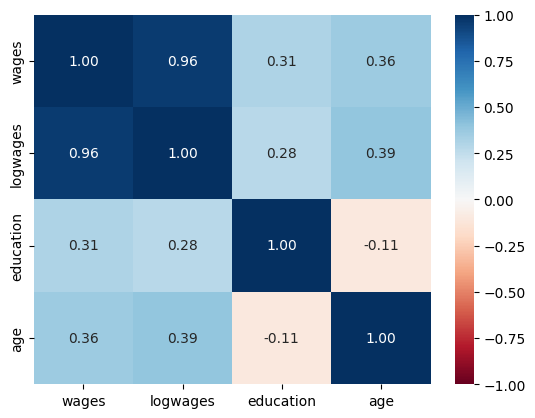
Statistic: 19.460  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.786, data does not look normal (reject H0)  
2.500: 0.917, data does not look normal (reject H0)  
1.000: 1.091, data does not look normal (reject H0)

# 'age' QQ plot  
qqplot(Wages\_df['age'], line='s')  
plt.grid(linewidth=0.25)  
plt.show()



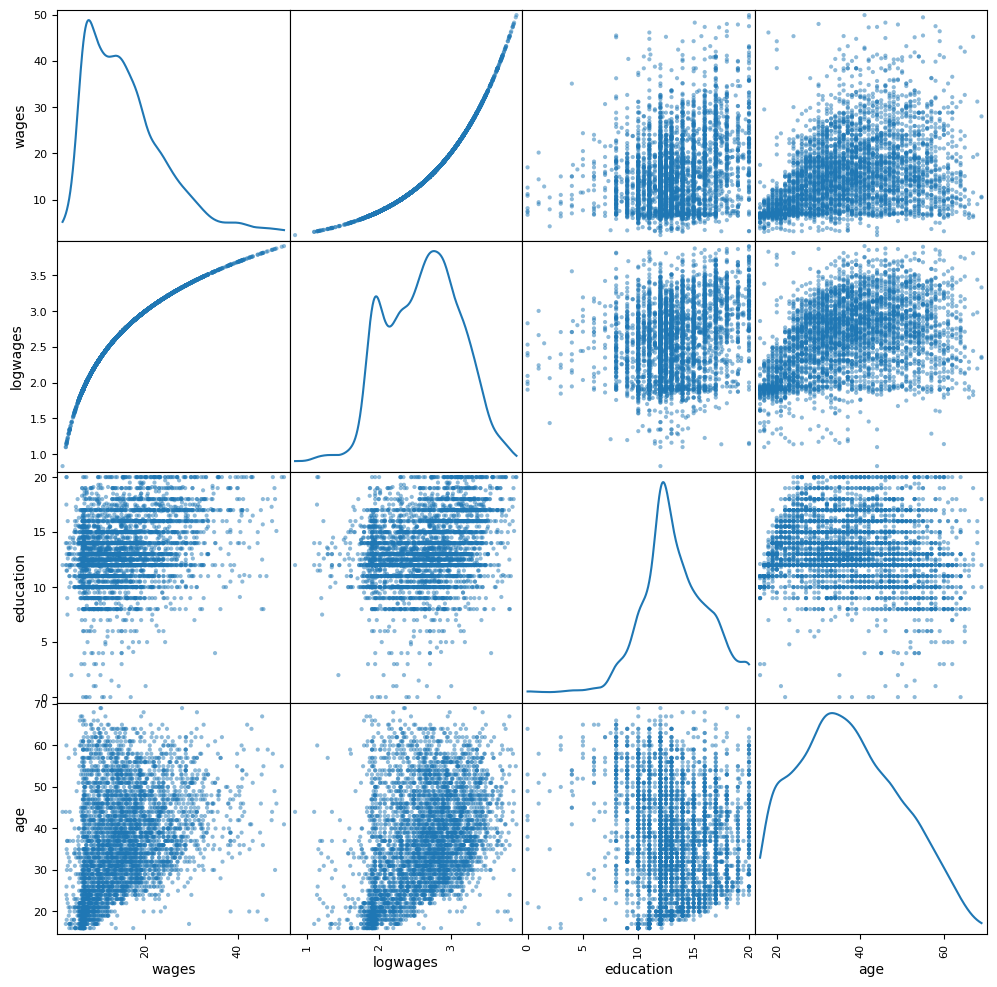
# Create dataframe of continuous variables for correlation analysis.  
Wages\_corr = Wages\_df[['wages','logwages','education', 'age']].copy()

# Generate heatmap for correlation matrix analysis.  
corr = Wages\_corr.corr()  
fig, ax = plt.subplots()  
sns.heatmap(corr, annot=True, fmt=".2f", xticklabels=corr.columns, yticklabels=corr.columns, vmin=-1, vmax=1, cmap="RdBu", ax=ax)  
plt.show()



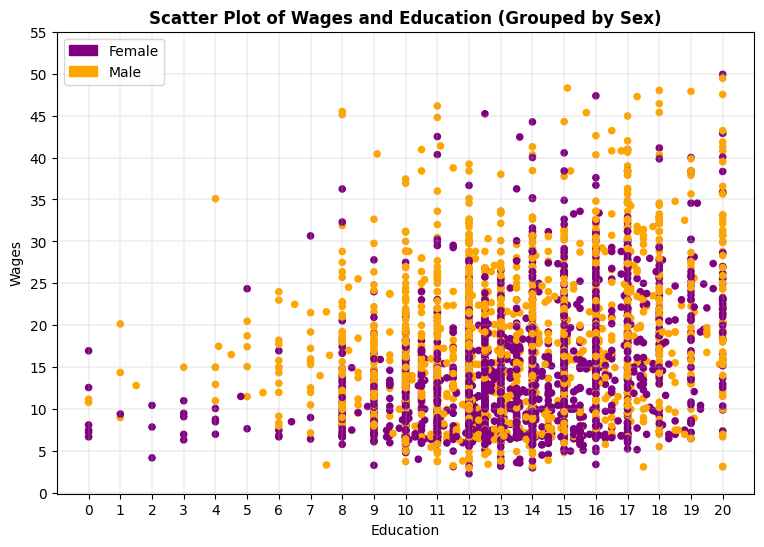
1. Create a scatter-plot matrix for the continuous variables: wages, log wages, education, and age. (10 points)

# Generate requested scatter plot matrix from required continuous variables.  
scatter\_matrix(Wages\_corr, diagonal="kde", figsize =(12, 12))  
plt.show()



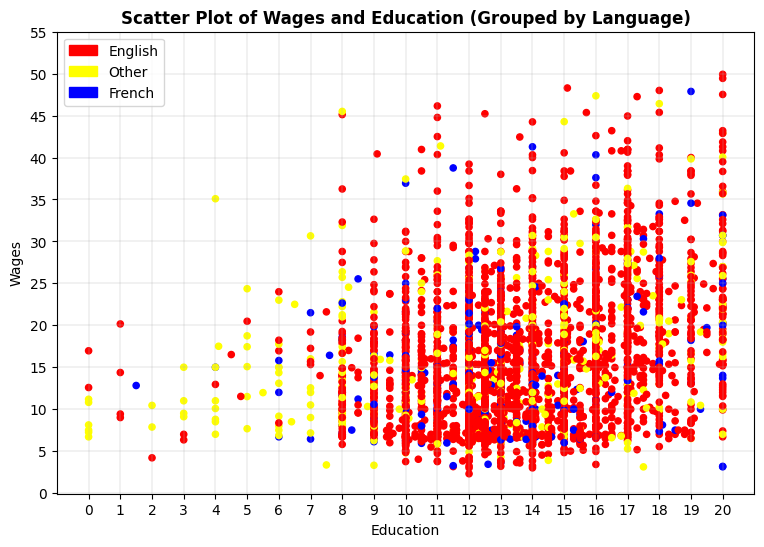
1. Create two grouped scatter plots for wages and education. One uses sex as the labeled variable, and the other uses language as the labeled variable (20 points).

# 2.1 Generate a grouped scatter plot for the continuous variables 'wages' and 'education', using the categorical label variable 'sex'.  
# Get color for each data point.  
colors = {'Female': 'purple', 'Male': 'orange'}  
color\_list = [colors[group] for group in Wages\_df['sex']]  
  
# Create a scatter plot with color-coding based on 'sex'.  
ax = Wages\_df.plot.scatter('education',  
 'wages',  
 c=color\_list,  
 grid=True,  
 figsize=(9,6))  
  
# Create legend handles, labels for each group and add legend to the plot.  
legend\_handles = [  
 mpatches.Patch(color=colors['Female'], label='Female'),  
 mpatches.Patch(color=colors['Male'], label='Male')  
]  
ax.legend(handles=legend\_handles,  
 loc='upper left')  
  
# Add title and labels.  
ax.set\_title('Scatter Plot of Wages and Education (Grouped by Sex)',  
 weight='bold')  
ax.set\_xlabel('Education')  
ax.set\_ylabel('Wages')  
  
plt.xticks(np.arange(0,21,step=1))   
plt.yticks(np.arange(0,56,step=5))   
plt.grid(linewidth=0.25)  
plt.show()



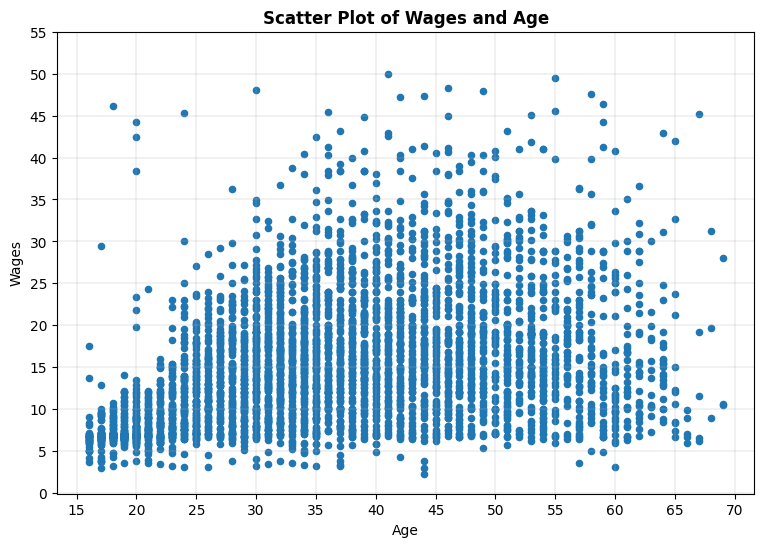
# 2.2 Generate a grouped scatter plot for the continuous variables 'wages' and 'education', using the categorical label variable 'language'.  
# Get color for each data point.  
colors = {'English': 'red', 'Other': 'yellow', 'French': 'blue'}  
color\_list = [colors[group] for group in Wages\_df['language']]  
  
# Create a scatter plot with color-coding based on 'language'.  
ax = Wages\_df.plot.scatter('education',  
 'wages',  
 c=color\_list,  
 grid=True,  
 figsize=(9,6))  
  
# Create legend handles, labels for each group and add legend to the plot.  
legend\_handles = [  
 mpatches.Patch(color=colors['English'], label='English'),  
 mpatches.Patch(color=colors['Other'], label='Other'),  
 mpatches.Patch(color=colors['French'], label='French'),  
]  
ax.legend(handles=legend\_handles,  
 loc='upper left')

# Add title and labels.  
ax.set\_title('Scatter Plot of Wages and Education (Grouped by Language)',  
 weight='bold')  
ax.set\_xlabel('Education')  
ax.set\_ylabel('Wages')  
  
plt.xticks(np.arange(0,21,step=1))   
plt.yticks(np.arange(0,56,step=5))   
plt.grid(linewidth=0.25)  
plt.show()



1. Create a regression model using wages as the dependent variable with education and age as the independent variables. Generate an ANOVA table describing overall performance. Is the model a good fit? Explain your hypothesis testing, f-test, and plots generated. (25 points)

# Generate a scatter plot for the continuous variables 'wages' and 'age'.  
ax = Wages\_df.plot.scatter('age',  
 'wages',  
 grid=True,  
 figsize=(9,6))  
  
ax.set\_title('Scatter Plot of Wages and Age',  
 weight='bold')  
ax.set\_xlabel('Age')  
ax.set\_ylabel('Wages')  
  
plt.xticks(np.arange(15,71,step=5))   
plt.yticks(np.arange(0,56,step=5))   
plt.grid(linewidth=0.25)  
plt.show()



# Generate 3D plot of two significant explanatory variables and the outcome variable.  
fig = plt.figure(figsize=(12,12))  
ax = plt.axes(projection="3d")  
  
x = Wages\_df['education'].tolist()  
y = Wages\_df['age'].tolist()  
z = Wages\_df['wages'].tolist()  
  
my\_cmap = plt.cm.magma  
norm = mpl.colors.Normalize(vmin=10, vmax=60)  
  
sctt = ax.scatter3D(x, y, z, c=z, alpha = 1, cmap=plt.cm.magma, marker = 'd')  
  
ax.set\_title('Wages by Education & Age - 3D Scatterplot', fontweight='bold', fontsize=16)  
ax.set\_xlabel("Education", fontweight='bold')  
ax.set\_ylabel("Age", fontweight='bold')  
ax.set\_zlabel("Wages", fontweight='bold')  
ax.set\_zticks([0,5,10,15,20,25,30,35,40,45,50,55])  
ax.set\_box\_aspect(aspect = (1,1,1))  
ax.view\_init(elev=15, azim=240)  
  
fig.colorbar(mpl.cm.ScalarMappable(norm = norm, cmap = my\_cmap), ax = ax, shrink = 0.5, aspect = 5)  
  
plt.xticks(np.arange(0,21,step=1))   
plt.yticks(np.arange(15,71,step=5))  
plt.grid(linewidth=0.25)  
plt.show()



# Set Alpha standard for regression model testing.  
alpha = 0.05

# Generate required multiple linear regression model.  
# Fit linear regression model.  
lm = smf.ols("wages ~ education + age", data = Wages\_df).fit()  
# View model coefficients.  
print(lm.params)

Intercept -6.021653  
education 0.901464  
age 0.257090  
dtype: float64

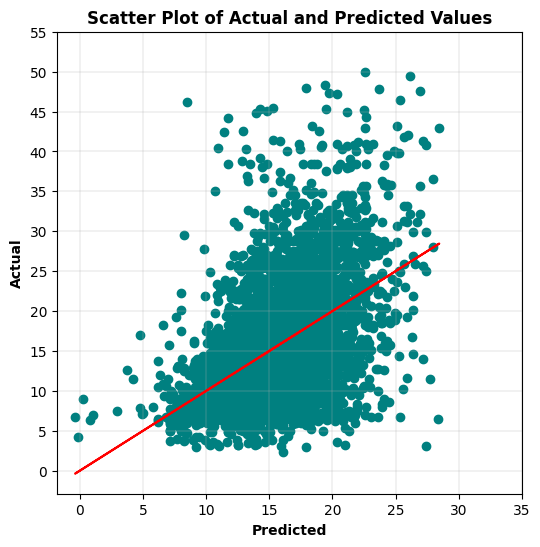
# Calculate the Total Sum of Squares (SST) for an empty model for 'wages'.  
Y = Wages\_df['wages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 246,790.472

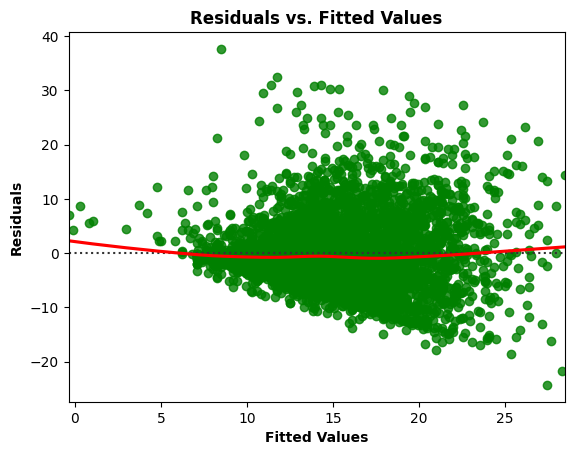
# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Assess multicollinearity with VIF table.  
X = Wages\_df[['education','age']]  
# Create VIF dataframe.  
# vif\_data = pd.DataFrame()  
# vif\_data["feature"] = X.columns  
# Calculate VIF for explanatory variable.  
# vif\_data["VIF"] = [variance\_inflation\_factor(X.values, i)  
# for i in range(len(X.columns))]  
# print(vif\_data, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}",'\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
predicted = lm.predict(X)

# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(0,36,step=5))   
plt.yticks(np.arange(0,56,step=5))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: wages R-squared: 0.249  
Model: OLS Adj. R-squared: 0.249  
Method: Least Squares F-statistic: 660.7  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.56e-248  
Time: 09:57:20 Log-Likelihood: -13310.  
No. Observations: 3987 AIC: 2.663e+04  
Df Residuals: 3984 BIC: 2.665e+04  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept -6.0217 0.619 -9.729 0.000 -7.235 -4.808  
education 0.9015 0.036 25.209 0.000 0.831 0.972  
age 0.2571 0.009 28.721 0.000 0.240 0.275  
=============================================================================  
Omnibus: 561.750 Durbin-Watson: 1.959  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 1092.696  
Skew: 0.878 Prob(JB): 5.30e-238  
Kurtosis: 4.870 Cond. No. 236.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 185,322.444  
SSE = 61,468.028   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



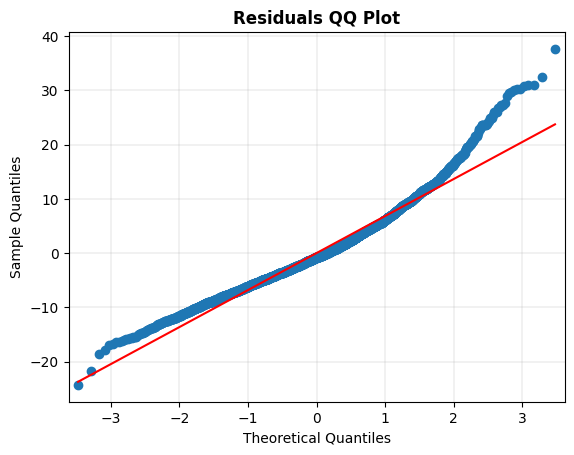
# Display Residuals vs. Fitted values scatter plot.  
fitted = lm.fittedvalues  
residuals = lm.resid  
sns.residplot(x=fitted, y=residuals, lowess=True, line\_kws={'color': 'red'}, color='green')   
plt.title('Residuals vs. Fitted Values', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel('Residuals', fontweight='bold')  
plt.show()



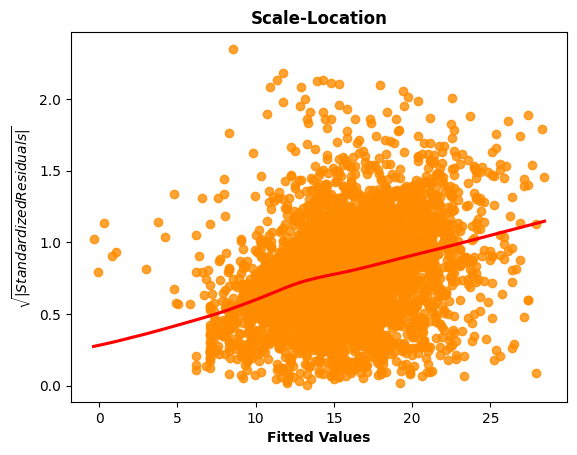
# Residuals Anderson-Darling Normality Test  
result = anderson(residuals)  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 32.202  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.786, data does not look normal (reject H0)  
2.500: 0.917, data does not look normal (reject H0)  
1.000: 1.091, data does not look normal (reject H0)

# Residuals QQ Plot  
qqplot(residuals, line='s')  
plt.title('Residuals QQ Plot', fontweight='bold')  
plt.grid(linewidth=0.25)  
plt.show()



# Scale-Location Plot  
resid\_standardized = lm.get\_influence().resid\_studentized\_internal  
  
sns.regplot(x=fitted, y=np.sqrt(np.abs(resid\_standardized)), color='darkorange',  
 ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-Location', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel(r'$\sqrt{|Standardized Residuals|}$', fontweight='bold')  
plt.show()



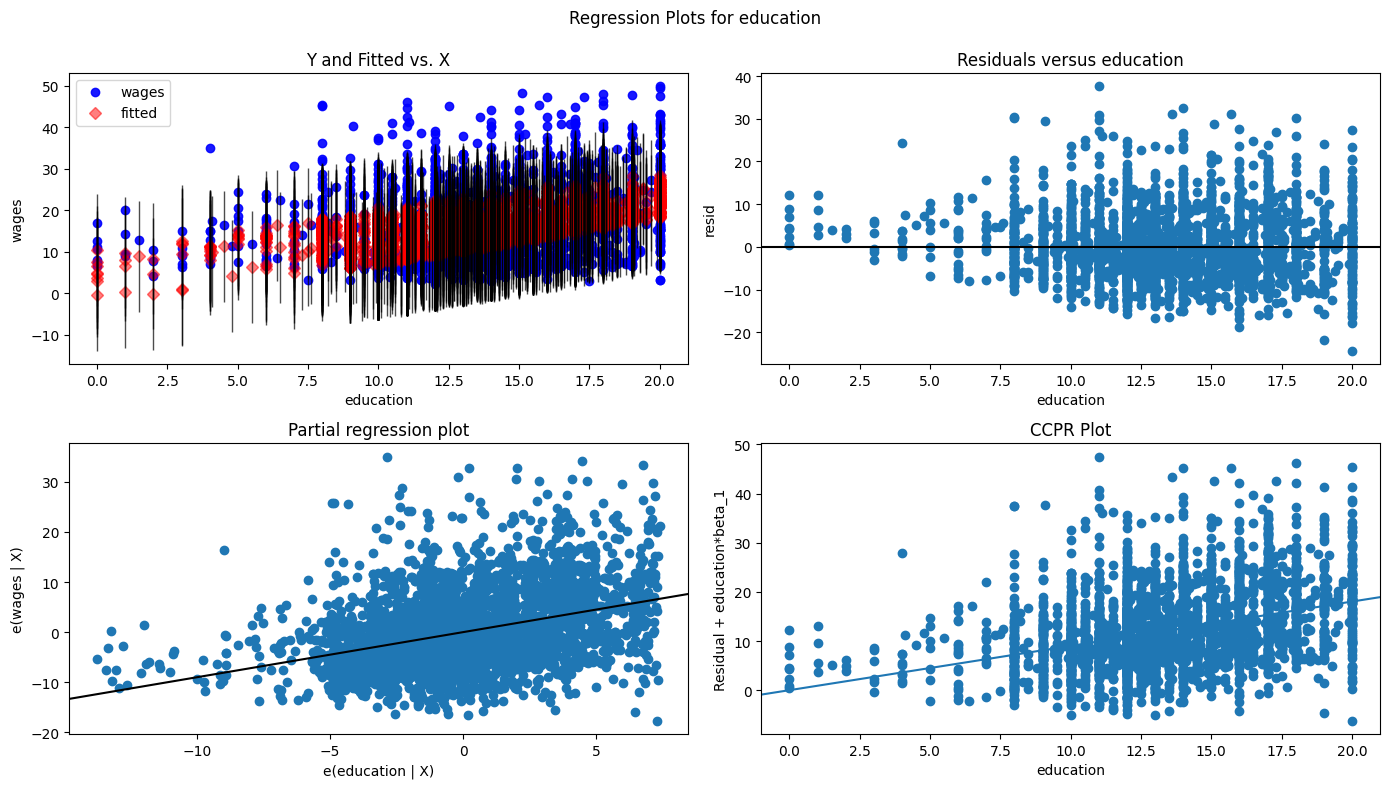
# Generate the Breusch-Pagan test for heteroscedasticity.  
bp\_test = het\_breuschpagan(lm.resid, lm.model.exog)  
print("lm:", f"{bp\_test[0]:0,.3f}", "lm\_pvalue:", f"{bp\_test[1]:0,.3f}")

lm: 166.360 lm\_pvalue: 0.000

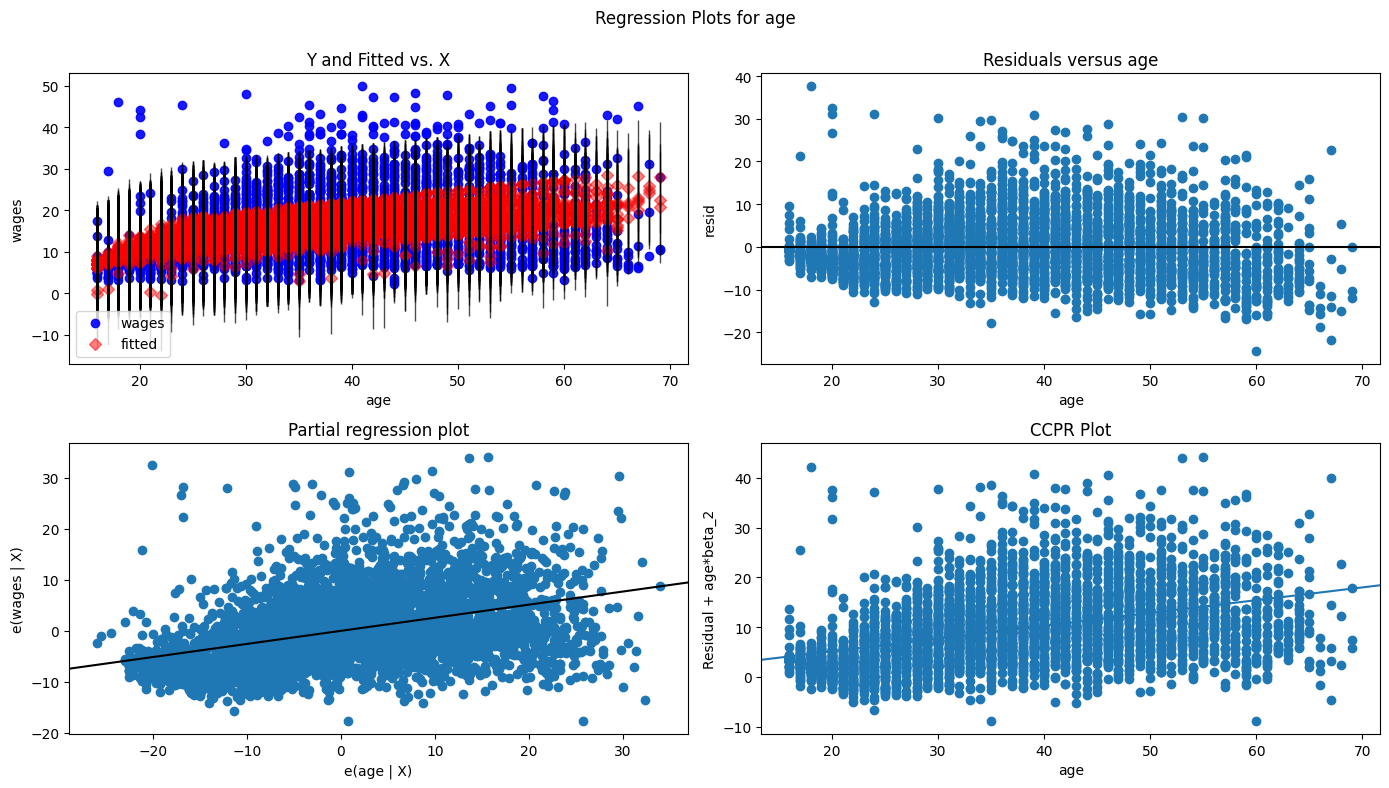
# Generate the White test for heteroscedasticity.  
# w\_test = het\_white(lm.resid, lm.model.exog)  
# print("lm:", w\_test[0], "lm\_pvalue:", w\_test[1])

# Residuals vs. Leverage Plot  
# fig, ax = plt.subplots(figsize=(12,12))  
# fig = sm.graphics.influence\_plot(lm, alpha=0.05, ax=ax, criterion="cooks")  
# plt.show()

# Display model regression plots for the 'education' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'education',fig=fig)  
plt.show()



# Display model regression plots for the 'age' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'age',fig=fig)  
plt.show()



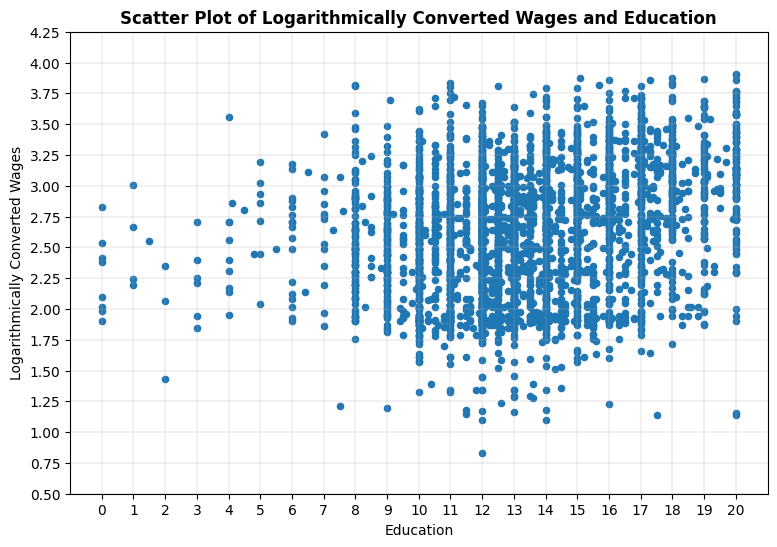
3 Interpret the results for wages MLR model:

The required OLS regression model was run with wages as the outcome variable. The R-squared value of 0.249 indicates that 24.9% of the variation in the predicted wage values can be explained by this multiple linear regression model with education and age as explanatory variables. The Y intercept, beta0, equals -6.021653 USD/hr, education, beta1, equals 0.9015 USD/hr per 1 level of education, and age, beta2, equals 0.2571 USD/hr per 1 year of age. The F-statistic of 660.7 easily exceeded the F-critical of 9.552, and the F-statistic probability was near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating clearly that the model relationship between the outcome and explanatory variables is statistically significant. The education and age variables had a t-statistic that exceeded the absolute t-critical of 4.303, and p-values near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating clearly that their relationships with wages are statistically significant.

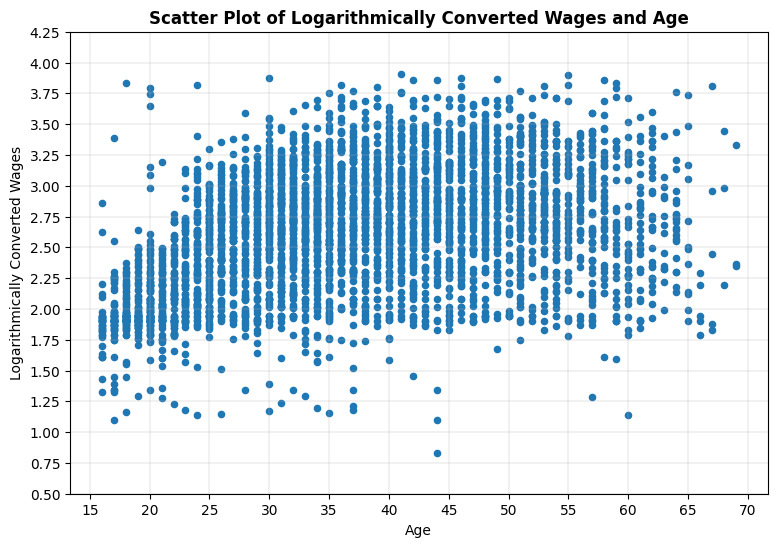
With wages having a SST of 246,790.472, the wages MLR model using education and age variables reduced it 61,468.028 to a SSR of 185,322.444. When plotting the residuals and the fitted value, there seems to be a reasonable amount of linearity in the model. Concerning normality of the residuals, the Jarque-Bera test statistic of 1,092.696 and Jarque-Bera probability near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicates a lack of normality. The Anderson-Darling result of 32.202 and QQ plot support that conclusion.  
The Scale-Location chart showed an incline that indicates notable heteroscedasticity in the model, and the Breusch-Pagan test statistic of 166.36 and p-value of nearly zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicate that heteroscedasticity is present in the model. Believe that the significantly right skewed distribution of the wages outcome variable and the somewhat left tailed distribution for the education explanatory variable were primarily responsible for issues with MLR model normality and heteroscedasticity. The low R squared value, extremely low Log-Likelihood value, and very high AIC/BIC values are an indicator that this MLR model would not be a good predictor of wages.

1. Create a regression model using log wages as the dependent variable with education and age as the independent variables. Generate an ANOVA table describing overall performance. Is the model a good fit? Explain your hypothesis testing results (25 points).

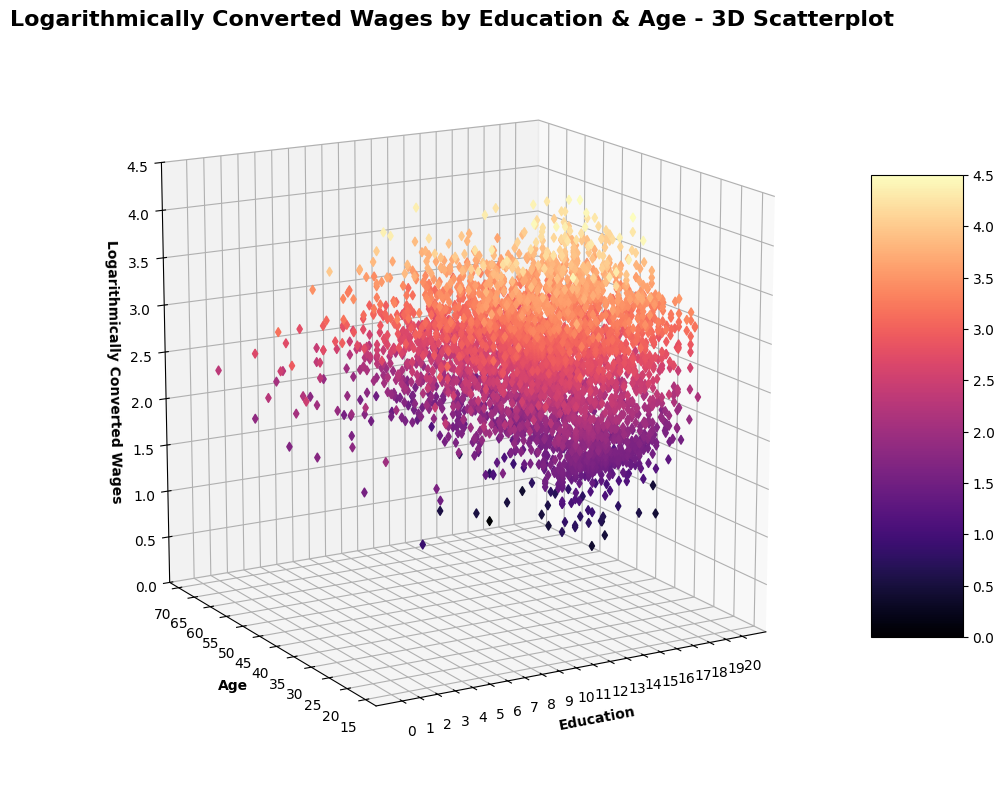
# Generate a scatter plot for the continuous variables 'logwages' and 'education'.  
ax = Wages\_df.plot.scatter('education',  
 'logwages',  
 grid=True,  
 figsize=(9,6))  
  
ax.set\_title('Scatter Plot of Logarithmically Converted Wages and Education',  
 weight='bold')  
ax.set\_xlabel('Education')  
ax.set\_ylabel('Logarithmically Converted Wages')  
  
plt.xticks(np.arange(0,21,step=1))   
plt.yticks(np.arange(0.50,4.50,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()



# Generate a scatter plot for the continuous variables 'logwages' and 'age'.  
ax = Wages\_df.plot.scatter('age',  
 'logwages',  
 grid=True,  
 figsize=(9,6))  
  
ax.set\_title('Scatter Plot of Logarithmically Converted Wages and Age',  
 weight='bold')  
ax.set\_xlabel('Age')  
ax.set\_ylabel('Logarithmically Converted Wages')  
  
plt.xticks(np.arange(15,71,step=5))   
plt.yticks(np.arange(0.50,4.50,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()



# Generate 3D plot of two significant explanatory variables and the outcome variable.  
fig = plt.figure(figsize=(12,12))  
ax = plt.axes(projection="3d")  
  
x = Wages\_df['education'].tolist()  
y = Wages\_df['age'].tolist()  
z = Wages\_df['logwages'].tolist()  
  
my\_cmap = plt.cm.magma  
norm = mpl.colors.Normalize(vmin=0, vmax=4.5)  
  
sctt = ax.scatter3D(x, y, z, c=z, alpha = 1, cmap=plt.cm.magma, marker = 'd')  
  
ax.set\_title('Logarithmically Converted Wages by Education & Age - 3D Scatterplot', fontweight='bold', fontsize=16)  
ax.set\_xlabel("Education", fontweight='bold')  
ax.set\_ylabel("Age", fontweight='bold')  
ax.set\_zlabel("Logarithmically Converted Wages", fontweight='bold')  
ax.set\_zticks([0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5])  
ax.set\_box\_aspect(aspect = (1,1,1))  
ax.view\_init(elev=15, azim=240)  
  
fig.colorbar(mpl.cm.ScalarMappable(norm = norm, cmap = my\_cmap), ax = ax, shrink = 0.5, aspect = 5)  
  
plt.xticks(np.arange(0,21,step=1))   
plt.yticks(np.arange(15,71,step=5))  
plt.grid(linewidth=0.25)  
plt.show()



# Generate required multiple linear regression model.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_df).fit()  
# View model coefficients.  
print(lm.params)

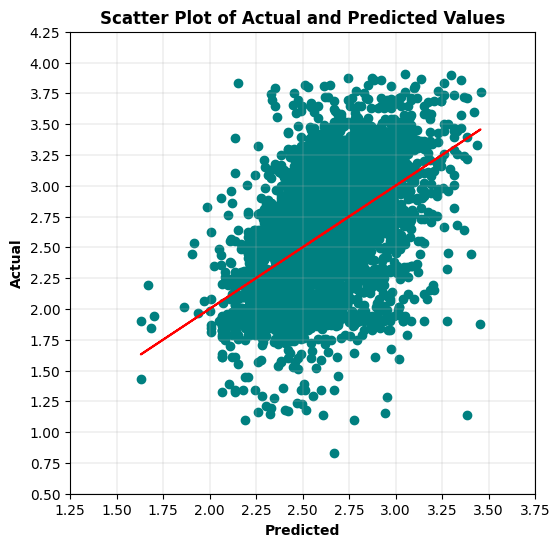
Intercept 1.240207  
education 0.054032  
age 0.017751  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_df['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

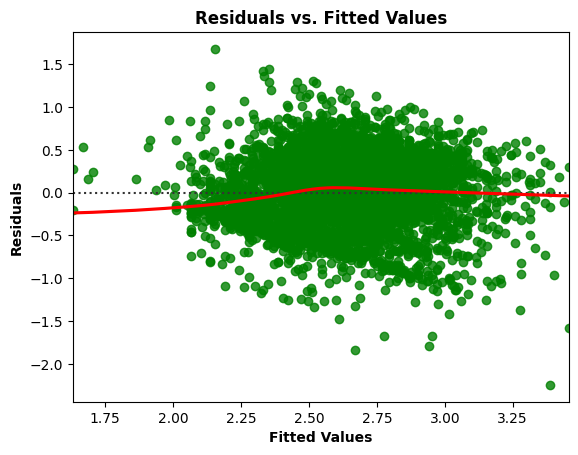
SST = 1,010.056

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_df[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,4.0,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.260  
Model: OLS Adj. R-squared: 0.259  
Method: Least Squares F-statistic: 699.4  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 4.67e-261  
Time: 09:59:43 Log-Likelihood: 2320.3  
No. Observations: 3987 AIC: 4647.  
Df Residuals: 3984 BIC: 4665.  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.2402 0.039 31.549 0.000 1.163 1.317  
education 0.0540 0.002 23.790 0.000 0.050 0.058  
age 0.0178 0.001 31.224 0.000 0.017 0.019  
=============================================================================  
Omnibus: 55.164 Durbin-Watson: 2.002  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 61.517  
Skew: -0.245 Prob(JB): 4.38e-14  
Kurtosis: 3.360 Cond. No. 236.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 747.580  
SSE = 262.477   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Display Residuals vs. Fitted values scatter plot.  
fitted = lm.fittedvalues  
residuals = lm.resid  
sns.residplot(x=fitted, y=residuals, lowess=True, line\_kws={'color': 'red'}, color='green')   
plt.title('Residuals vs. Fitted Values', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel('Residuals', fontweight='bold')  
plt.show()



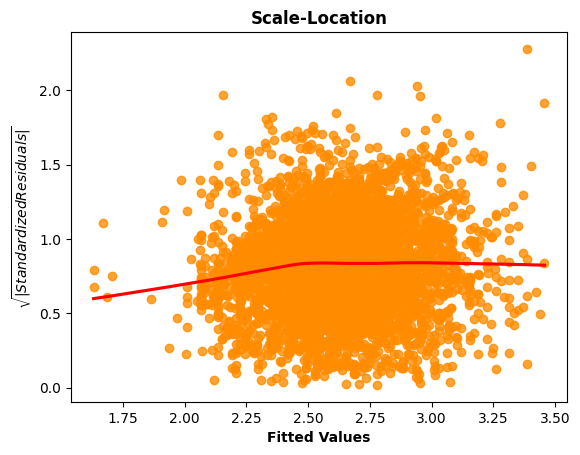
# Residuals Anderson-Darling Normality Test  
result = anderson(residuals)  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 4.048  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.786, data does not look normal (reject H0)  
2.500: 0.917, data does not look normal (reject H0)  
1.000: 1.091, data does not look normal (reject H0)

# Residuals QQ Plot  
qqplot(residuals, line='s')  
plt.title('Residuals QQ Plot', fontweight='bold')  
plt.grid(linewidth=0.25)  
plt.show()



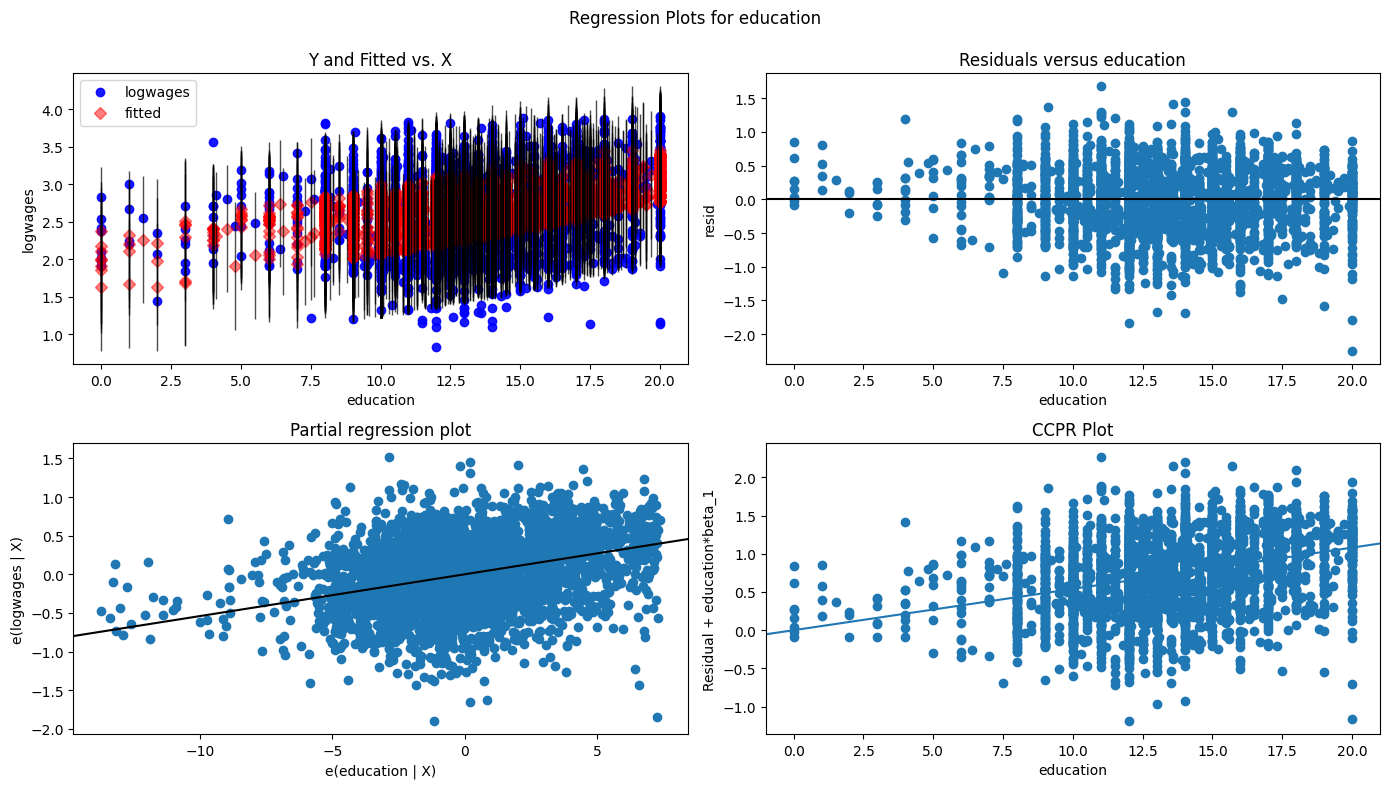
# Scale-Location Plot  
resid\_standardized = lm.get\_influence().resid\_studentized\_internal  
  
sns.regplot(x=fitted, y=np.sqrt(np.abs(resid\_standardized)), color='darkorange',  
 ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-Location', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel(r'$\sqrt{|Standardized Residuals|}$', fontweight='bold')  
plt.show()



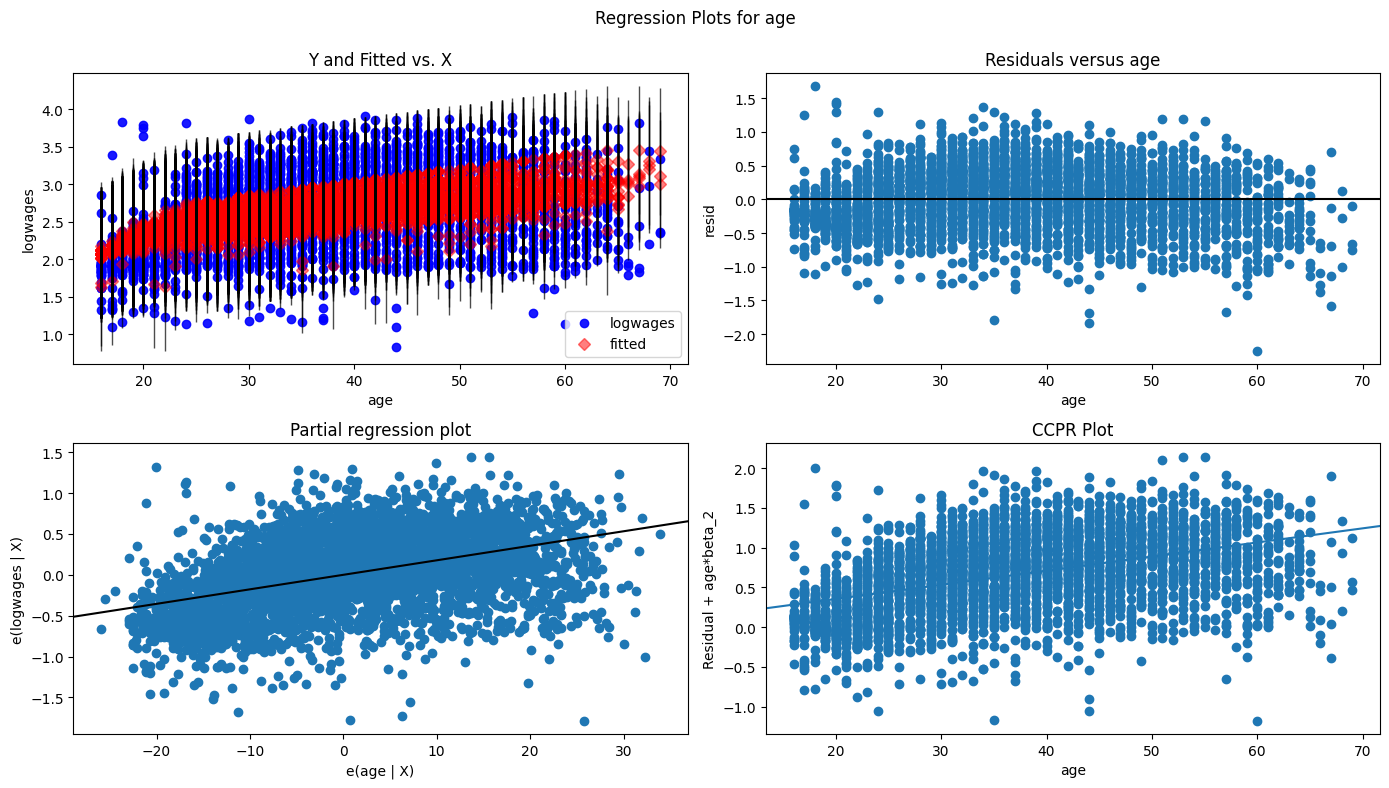
# Generate the Breusch-Pagan test for heteroscedasticity.  
bp\_test = het\_breuschpagan(lm.resid, lm.model.exog)  
print("lm:", f"{bp\_test[0]:0,.3f}", "lm\_pvalue:", f"{bp\_test[1]:0,.3f}")

lm: 43.336 lm\_pvalue: 0.000

# Display model regression plots for the 'education' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'education',fig=fig)  
plt.show()



# Display model regression plots for the 'age' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'age',fig=fig)  
plt.show()



4 Interpret the results for the logarithmically converted wages MLR model:

The required OLS regression model was run with logwages as the outcome variable. The R-squared value of 0.260 indicates that 26.0% of the variation in the predicted logwages values can be explained by this multiple linear regression model with education and age as explanatory variables. The Y intercept, beta0, equals 1.240207 log(USD/hr), education, beta1, equals 0.054032 log(USD/hr) per 1 level of education, and age, beta2, equals 0.017751 log(USD/hr) per 1 year of age. The F-statistic of 699.4 easily exceeded the F-critical of 9.552, and the F-statistic probability was near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating clearly that the model relationship between the outcome and explanatory variables is statistically significant. The education and age variables had a t-statistic that exceeded the absolute t-critical of 4.303, and p-values near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating clearly that their relationships with logwages are statistically significant.

With logwages having a SST of 1,010.056, the logwages MLR model using education and age variables reduced it 262.477 to a SSR of 747.580. When plotting the residuals and the fitted value, there seems to be a reasonable amount of linearity in the model. Concerning normality of the residuals, the Jarque-Bera test statistic of 61.517 and Jarque-Bera probability near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicates a lack of normality. Although, the Anderson-Darling result of 4.048 somewhat supports that conclusion, and QQ plot rendering shows some acceptable normality. The Scale-Location chart showed some fluctuation that indicates potential heteroscedasticity in the model, and the Breusch-Pagan test statistic of 43.336 and p-value of nearly zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicate that heteroscedasticity is present in the model. On the whole, the logarithmically transformed wages outcome variable slightly improved model predictions, significantly improved the residual normality, and helped correct some heteroscedasticity. Although, the low R squared value, low Log-Likelihood value, and high AIC/BIC values are an indicator that this MLR model would not be a good predictor of logwages.

1. Use grouping variables for sex and language, and add these to model. Select the grouping variable that creates the best model. Explain the performance difference (if any) over the previous models. (30 points)

5a. My first interpretation of this problem was to add One-Hot encoded variables for sex and language values to the logwages MLR model, generate modeling results with them added, and then generate modeling results by adding sex and language One-Hot encoded variables exclusively in two different MLR models.

# Use One-Hot Encoding to create a new binary column for Female and Male in the 'sex' variable.  
Wages\_binary = pd.get\_dummies(Wages\_df, columns=['sex'])

# Use One-Hot Encoding to create a new binary column for English, French, and Other in the 'language' variable.  
Wages\_binary = pd.get\_dummies(Wages\_binary, columns=['language'])  
Wages\_binary.head()

wages education age logwages sex\_Female sex\_Male language\_English   
0 10.56 15.0 40 2.357073 False True True   
1 11.00 13.2 19 2.397895 False True True   
2 17.76 14.0 46 2.876949 False True False   
3 14.00 16.0 50 2.639057 True False True   
4 8.20 15.0 31 2.104134 False True True   
  
 language\_French language\_Other   
0 False False   
1 False False   
2 False True   
3 False False   
4 False False

# Generate required multiple linear regression model.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age + sex\_Female + sex\_Male + language\_English + language\_French + language\_Other", data = Wages\_binary).fit()  
# View model coefficients.  
print(lm.params)

Intercept 0.673905  
sex\_Female[T.True] 0.224823  
sex\_Male[T.True] 0.449082  
language\_English[T.True] 0.219685  
language\_French[T.True] 0.224607  
language\_Other[T.True] 0.229612  
education 0.055035  
age 0.017621  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_binary['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 1,010.056

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,5,6)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=5))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_binary[['education','age', 'sex\_Female', 'sex\_Male', 'language\_English', 'language\_French', 'language\_Other']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,4.0,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.309  
Model: OLS Adj. R-squared: 0.309  
Method: Least Squares F-statistic: 356.9  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 8.36e-317  
Time: 10:00:54 Log-Likelihood: 2181.9  
No. Observations: 3987 AIC: 4376.  
Df Residuals: 3981 BIC: 4414.  
Df Model: 5   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 0.6739 0.021 31.803 0.000 0.632 0.715  
sex\_Female[T.True]

0.2248 0.013 17.867 0.000 0.200 0.249  
sex\_Male[T.True]

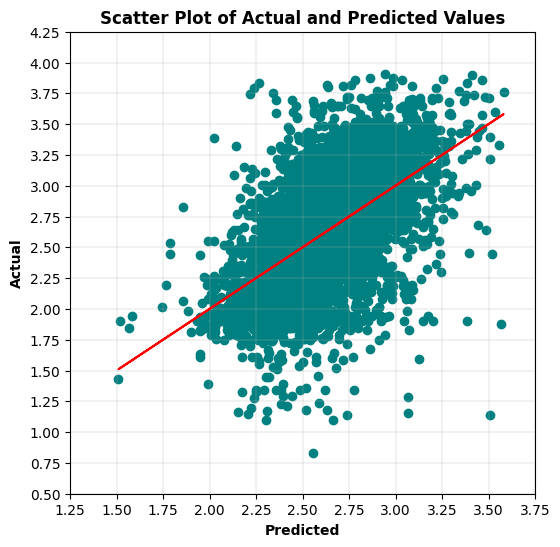
0.4491 0.012 36.170 0.000 0.425 0.473  
language\_English[T.True]

0.2197 0.012 17.578 0.000 0.195 0.244  
language\_French[T.True]

0.2246 0.021 10.815 0.000 0.184 0.265  
language\_Other[T.True]

0.2296 0.017 13.361 0.000 0.196 0.263  
education 0.0550 0.002 24.966 0.000 0.051 0.059  
age 0.0176 0.001 31.888 0.000 0.017 0.019  
=============================================================================  
Omnibus: 85.406 Durbin-Watson: 1.987  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 116.359  
Skew: -0.256 Prob(JB): 5.41e-26  
Kurtosis: 3.663 Cond. No. 2.40e+16  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The smallest eigenvalue is 1.16e-26. This might indicate that there are  
strong multicollinearity problems or that the design matrix is singular.  
  
 ====================

RSS = 697.448  
SSE = 312.608   
  
F\_critical: 4.387  
Absolute t\_critical: 2.571  
Alpha Standard: 0.05



# Generate required multiple linear regression model with 'sex' One-Hot encoded variables.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age + sex\_Female + sex\_Male", data = Wages\_binary).fit()  
# View model coefficients.  
print(lm.params)

Intercept 0.821543  
sex\_Female[T.True] 0.298643  
sex\_Male[T.True] 0.522900  
education 0.054935  
age 0.017651  
dtype: float64

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,3,4)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=3))  
print(f"t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_binary[['education','age', 'sex\_Female', 'sex\_Male']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,4.0,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.309  
Model: OLS Adj. R-squared: 0.309  
Method: Least Squares F-statistic: 595.0  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.59e-319  
Time: 10:01:22 Log-Likelihood: -2182.0  
No. Observations: 3987 AIC: 4372.  
Df Residuals: 3983 BIC: 4397.  
Df Model: 3   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 0.8215 0.025 32.448 0.000 0.772 0.871  
sex\_Female[T.True]

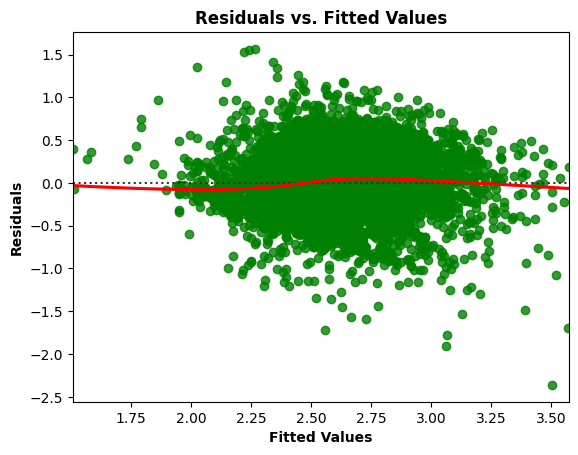
0.2986 0.014 20.794 0.000 0.270 0.327

sex\_Male[T.True]

0.5229 0.014 36.777 0.000 0.495 0.551  
education 0.0549 0.002 25.030 0.000 0.051 0.059  
age 0.0177 0.001 32.136 0.000 0.017 0.019  
=============================================================================  
Omnibus: 85.002 Durbin-Watson: 1.987  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 115.380  
Skew: -0.256 Prob(JB): 8.82e-26  
Kurtosis: 3.658 Cond. No. 1.30e+16  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The smallest eigenvalue is 3.97e-26. This might indicate that there are  
strong multicollinearity problems or that the design matrix is singular.  
  
 ====================   
  
RSS = 697.492  
SSE = 312.565   
  
F\_critical: 6.591  
t\_critical: 3.182  
Alpha Standard: 0.05



# Display Residuals vs. Fitted values scatter plot.  
fitted = lm.fittedvalues  
residuals = lm.resid  
sns.residplot(x=fitted, y=residuals, lowess=True, line\_kws={'color': 'red'}, color='green')   
plt.title('Residuals vs. Fitted Values', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel('Residuals', fontweight='bold')  
plt.show()



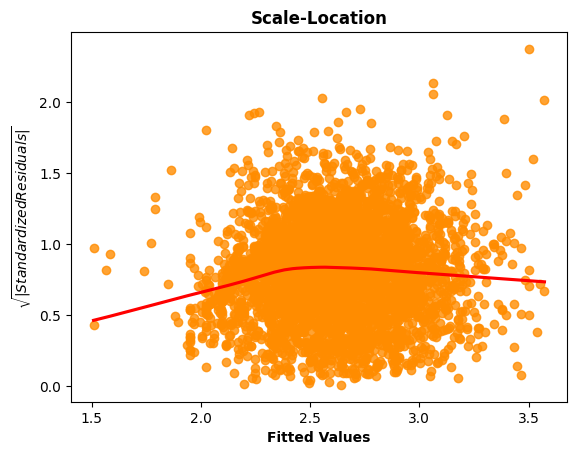
# Residuals Anderson-Darling Normality Test  
result = anderson(residuals)  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 3.219  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.786, data does not look normal (reject H0)  
2.500: 0.917, data does not look normal (reject H0)  
1.000: 1.091, data does not look normal (reject H0)

# Residuals QQ Plot  
qqplot(residuals, line='s')  
plt.title('Residuals QQ Plot', fontweight='bold')  
plt.grid(linewidth=0.25)  
plt.show()



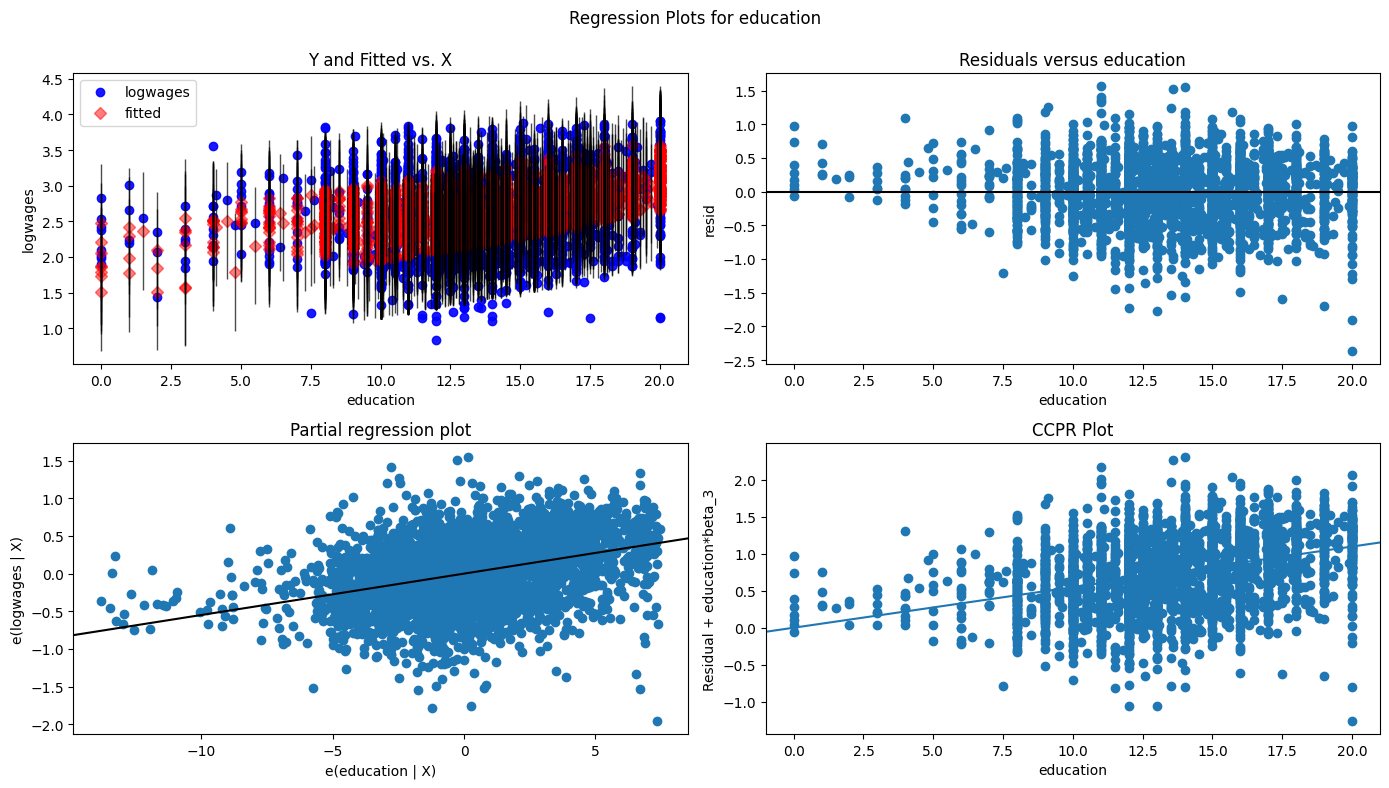
# Scale-Location Plot  
resid\_standardized = lm.get\_influence().resid\_studentized\_internal  
  
sns.regplot(x=fitted, y=np.sqrt(np.abs(resid\_standardized)), color='darkorange',  
 ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-Location', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel(r'$\sqrt{|Standardized Residuals|}$', fontweight='bold')  
plt.show()



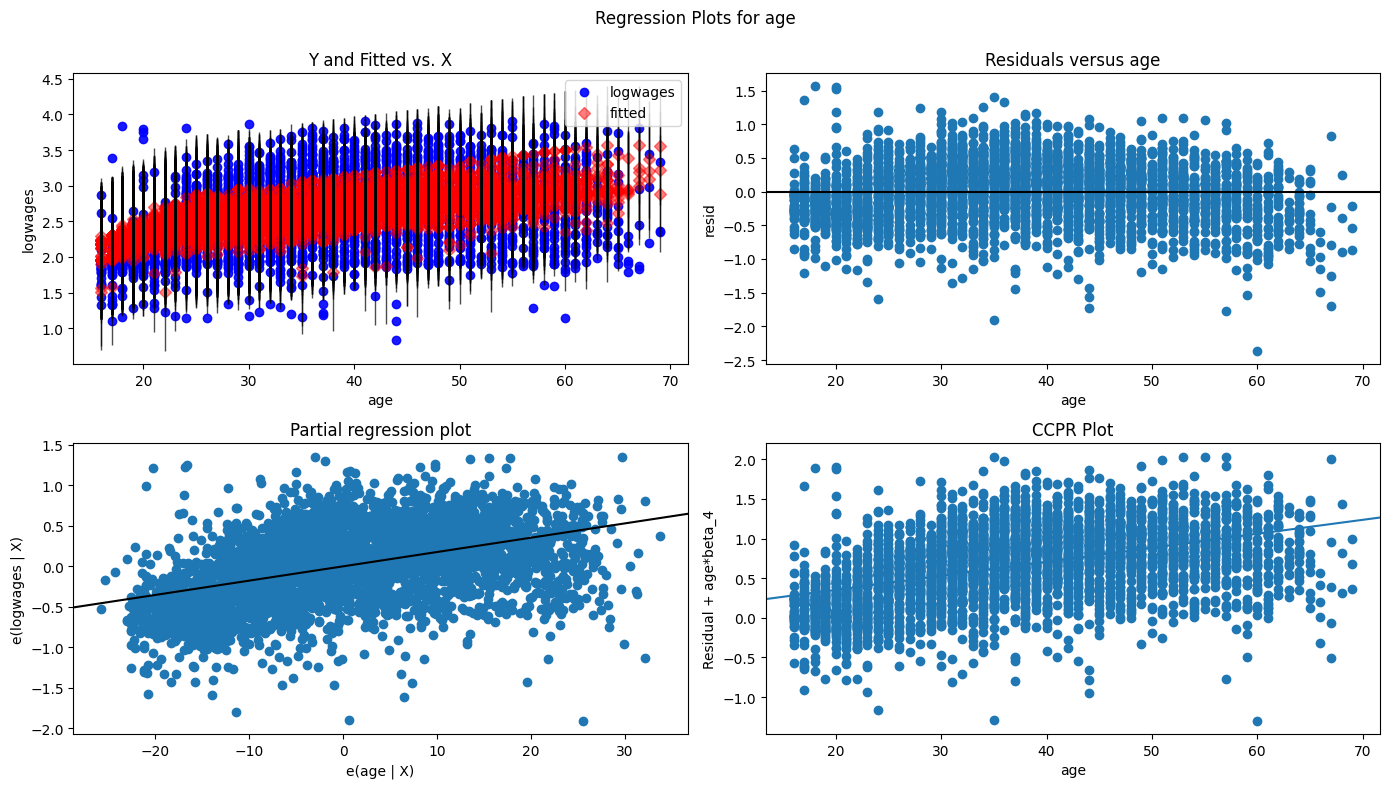
# Generate the Breusch-Pagan test for heteroscedasticity.  
bp\_test = het\_breuschpagan(lm.resid, lm.model.exog)  
print("lm:", f"{bp\_test[0]:0,.3f}", "lm\_pvalue:", f"{bp\_test[1]:0,.3f}")

lm: 28.438 lm\_pvalue: 0.000

# Display model regression plots for the 'education' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'education',fig=fig)  
plt.show()



# Display model regression plots for the 'age' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'age',fig=fig)  
plt.show()



# Generate required multiple linear regression model with 'language' One-Hot encoded variables.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age + language\_English + language\_French + language\_Other", data = Wages\_binary).fit()  
# View model coefficients.  
print(lm.params)

Intercept 0.933951  
language\_English[T.True] 0.303931  
language\_French[T.True] 0.318326  
language\_Other[T.True] 0.311695  
education 0.054136  
age 0.017726  
dtype: float64

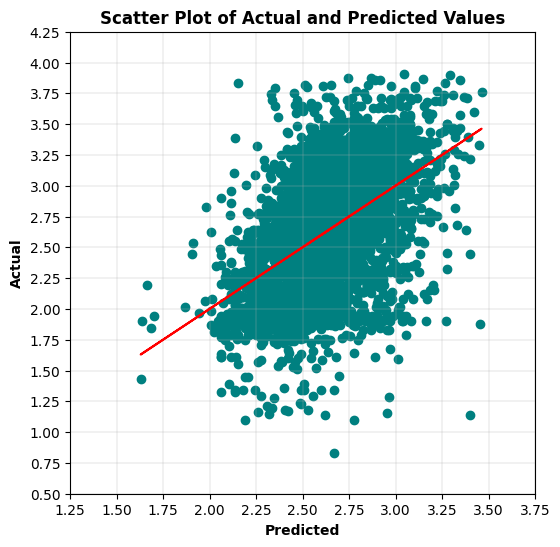
# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,4,5)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=4))  
print(f"t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_binary[['education','age', 'language\_English', 'language\_French', 'language\_Other']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,4.0,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.260  
Model: OLS Adj. R-squared: 0.259  
Method: Least Squares F-statistic: 349.6  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 2.73e-258  
Time: 10:03:06 Log-Likelihood: -2320.1  
No. Observations: 3987 AIC: 4650.  
Df Residuals: 3982 BIC: 4682.  
Df Model: 4   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 0.9340 0.030 30.970 0.000 0.875 0.993  
language\_English[T.True]

0.3039 0.014 21.443 0.000 0.276 0.332  
language\_French[T.True]

0.3183 0.023 13.917 0.000 0.273 0.363  
language\_Other[T.True]

0.3117 0.019 16.329 0.000 0.274 0.349  
education 0.0541 0.002 23.731 0.000 0.050 0.059  
age 0.0177 0.001 30.990 0.000 0.017 0.019  
=============================================================================  
Omnibus: 55.918 Durbin-Watson: 2.002  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 62.638  
Skew: -0.246 Prob(JB): 2.50e-14  
Kurtosis: 3.368 Cond. No. 1.56e+16  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The smallest eigenvalue is 2.78e-26. This might indicate that there are  
strong multicollinearity problems or that the design matrix is singular.  
  
 ====================   
  
RSS = 747.512  
SSE = 262.545   
  
F\_critical: 5.192  
t\_critical: 2.776  
Alpha Standard: 0.05



5b. After discussing this with the professor, I believe he wanted individual data groups associated with sex and language values to have a MLR model estimating logarithmically converted wages with age and education variables.

# Create individual dataframes associated with Females, Males, English, French, and Other values.  
Wages\_female = Wages\_df[Wages\_df.sex == 'Female']  
Wages\_male = Wages\_df[Wages\_df.sex == 'Male']  
Wages\_english = Wages\_df[Wages\_df.language == 'English']  
Wages\_french = Wages\_df[Wages\_df.language == 'French']  
Wages\_other = Wages\_df[Wages\_df.language == 'Other']

# Generate required multiple linear regression model for Female group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_female).fit()  
# View model coefficients.  
print(lm.params)

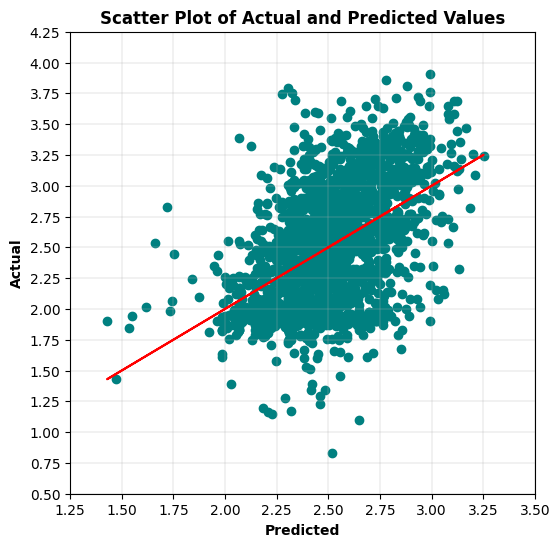
Intercept 1.112596  
education 0.064476  
age 0.014391  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_female['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 468.137

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_female[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,3.75,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.248  
Model: OLS Adj. R-squared: 0.248  
Method: Least Squares F-statistic: 330.3  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.21e-124  
Time: 10:03:30 Log-Likelihood: -1100.2  
No. Observations: 2001 AIC: 2206.  
Df Residuals: 1998 BIC: 2223.  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.1126 0.056 19.999 0.000 1.003 1.222  
education 0.0645 0.003 19.906 0.000 0.058 0.071  
age 0.0144 0.001 18.428 0.000 0.013 0.016  
=============================================================================  
Omnibus: 0.932 Durbin-Watson: 2.038  
Prob(Omnibus): 0.627 Jarque-Bera (JB): 0.849  
Skew: -0.013 Prob(JB): 0.654  
Kurtosis: 3.098 Cond. No. 243.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 351.828  
SSE = 116.309   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Generate required multiple linear regression model for Male group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_male).fit()  
# View model coefficients.  
print(lm.params)

Intercept 1.331881  
education 0.046639  
age 0.020940  
dtype: float64

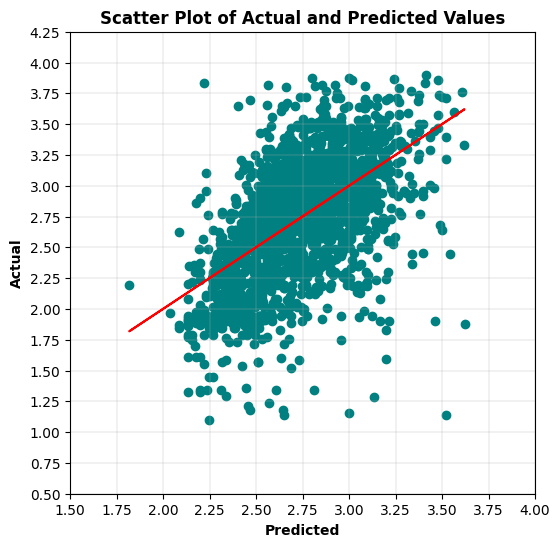
# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_male['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 493.026

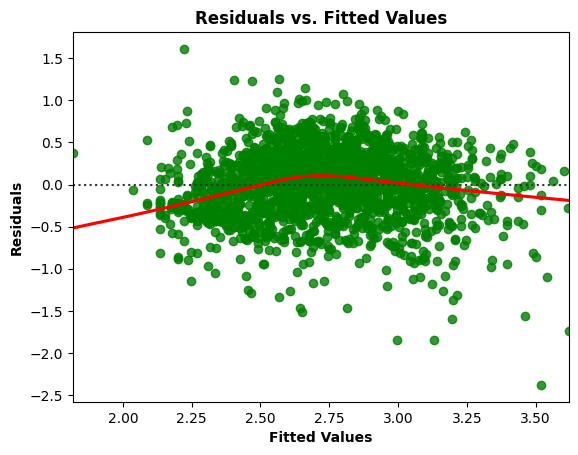
# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_male[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.5,4.25,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.319  
Model: OLS Adj. R-squared: 0.319  
Method: Least Squares F-statistic: 465.3  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 2.03e-166  
Time: 10:03:45 Log-Likelihood: -1052.4  
No. Observations: 1986 AIC: 2111.  
Df Residuals: 1983 BIC: 2127.  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.3319 0.051 25.979 0.000 1.231 1.432  
education 0.0466 0.003 15.861 0.000 0.041 0.052  
age 0.0209 0.001 27.491 0.000 0.019 0.022  
=============================================================================

Omnibus: 137.144 Durbin-Watson: 2.041  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 243.827  
Skew: -0.501 Prob(JB): 1.13e53  
Kurtosis: 4.393 Cond. No. 229.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 335.550  
SSE = 157.476   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



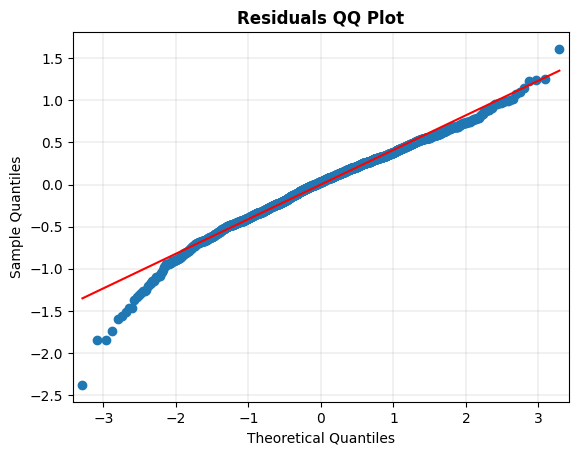
# Display Residuals vs. Fitted values scatter plot.  
fitted = lm.fittedvalues  
residuals = lm.resid  
sns.residplot(x=fitted, y=residuals, lowess=True, line\_kws={'color': 'red'}, color='green')   
plt.title('Residuals vs. Fitted Values', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel('Residuals', fontweight='bold')  
plt.show()



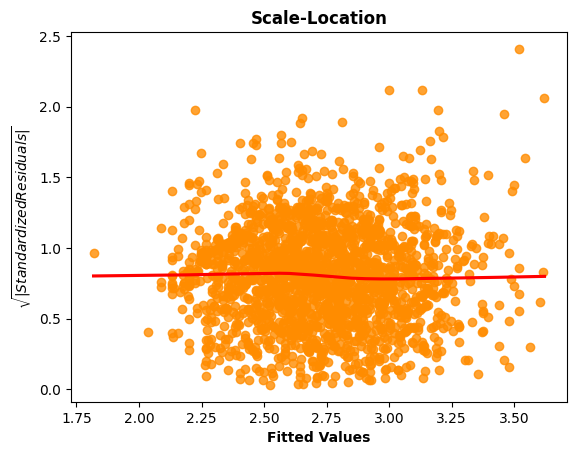
# Residuals Anderson-Darling Normality Test  
result = anderson(residuals)  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 4.415  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.655, data does not look normal (reject H0)  
5.000: 0.785, data does not look normal (reject H0)  
2.500: 0.916, data does not look normal (reject H0)  
1.000: 1.090, data does not look normal (reject H0)

# Residuals QQ Plot  
qqplot(residuals, line='s')  
plt.title('Residuals QQ Plot', fontweight='bold')  
plt.grid(linewidth=0.25)  
plt.show()



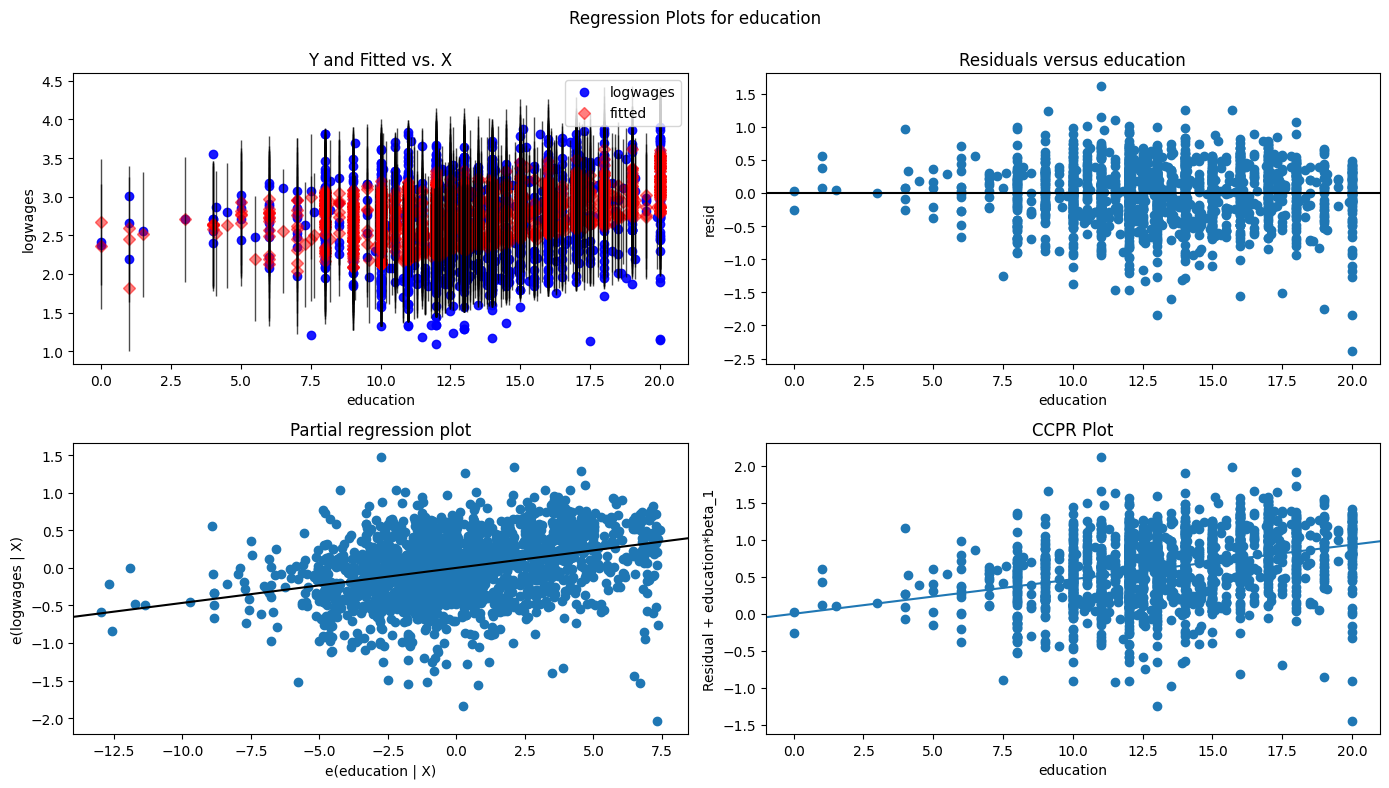
# Scale-Location Plot  
resid\_standardized = lm.get\_influence().resid\_studentized\_internal  
  
sns.regplot(x=fitted, y=np.sqrt(np.abs(resid\_standardized)), color='darkorange',  
 ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-Location', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel(r'$\sqrt{|Standardized Residuals|}$', fontweight='bold')  
plt.show()



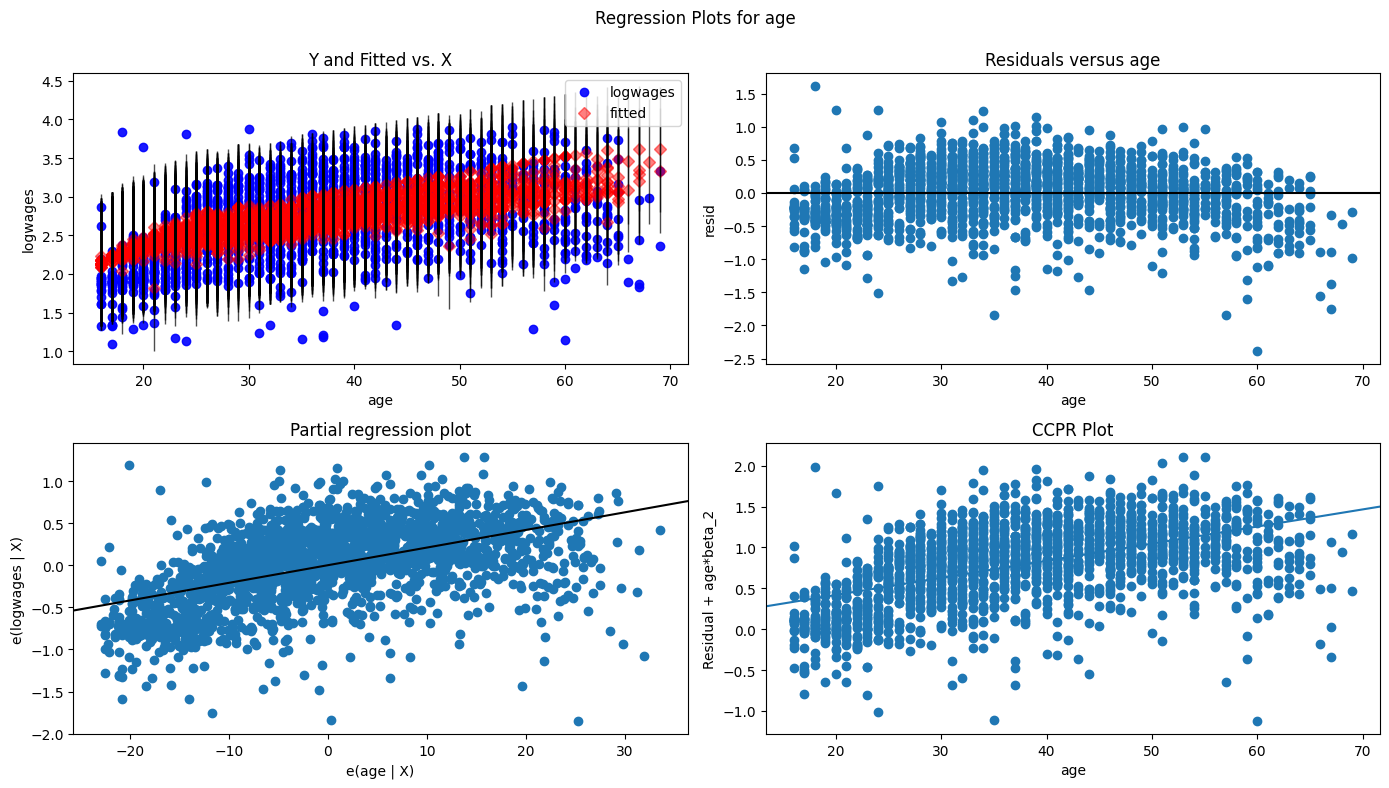
# Generate the Breusch-Pagan test for heteroscedasticity.  
bp\_test = het\_breuschpagan(lm.resid, lm.model.exog)  
print("lm:", f"{bp\_test[0]:0,.3f}", "lm\_pvalue:", f"{bp\_test[1]:0,.3f}")

lm: 10.484 lm\_pvalue: 0.005

# Display model regression plots for the 'education' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'education',fig=fig)  
plt.show()



# Display model regression plots for the 'age' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'age',fig=fig)  
plt.show()



# Generate required multiple linear regression model for English group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_english).fit()  
# View model coefficients.  
print(lm.params)

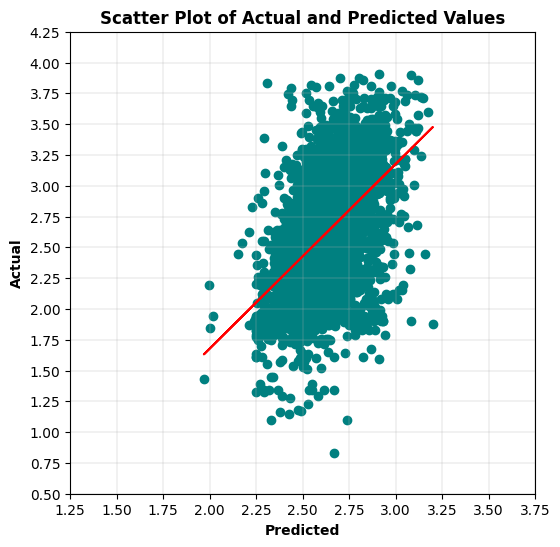
Intercept 1.182472  
education 0.057239  
age 0.018098  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_english['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 831.404

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_english[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,4,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.114  
Model: OLS Adj. R-squared: 0.107  
Method: Least Squares F-statistic: 16.48  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.86e-07  
Time: 10:05:14 Log-Likelihood: -169.36  
No. Observations: 259 AIC: 344.7  
Df Residuals: 256 BIC: 355.4  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.6991 0.170 10.018 0.000 1.365 2.033  
education 0.0347 0.009 3.723 0.000 0.016 0.053  
age 0.0126 0.003 4.968 0.000 0.008 0.018  
=============================================================================  
Omnibus: 31.458 Durbin-Watson: 1.960  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 50.652  
Skew: -0.714 Prob(JB): 1.00e-11  
Kurtosis: 4.630 Cond. No. 242.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 56.077  
SSE = 775.326   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Generate required multiple linear regression model for French group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_french).fit()  
# View model coefficients.  
print(lm.params)

Intercept 1.699127  
education 0.034713  
age 0.012568  
dtype: float64

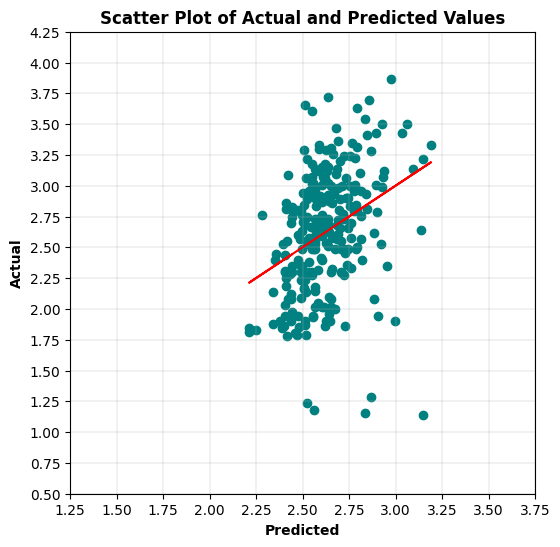
# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_french['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 63.295

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_french[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,4,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.114  
Model: OLS Adj. R-squared: 0.107  
Method: Least Squares F-statistic: 16.48  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.86e07  
Time: 10:05:27 Log-Likelihood: -169.36  
No. Observations: 259 AIC: 344.7  
Df Residuals: 256 BIC: 355.4  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.6991 0.170 10.018 0.000 1.365 2.033  
education 0.0347 0.009 3.723 0.000 0.016 0.053  
age 0.0126 0.003 4.968 0.000 0.008 0.018  
=============================================================================

Omnibus: 31.458 Durbin-Watson: 1.960  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 50.652  
Skew: -0.714 Prob(JB): 1.00e-11  
Kurtosis: 4.630 Cond. No. 242.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 56.077  
SSE = 7.218   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Generate required multiple linear regression model for Other group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_other).fit()  
# View model coefficients.  
print(lm.params)

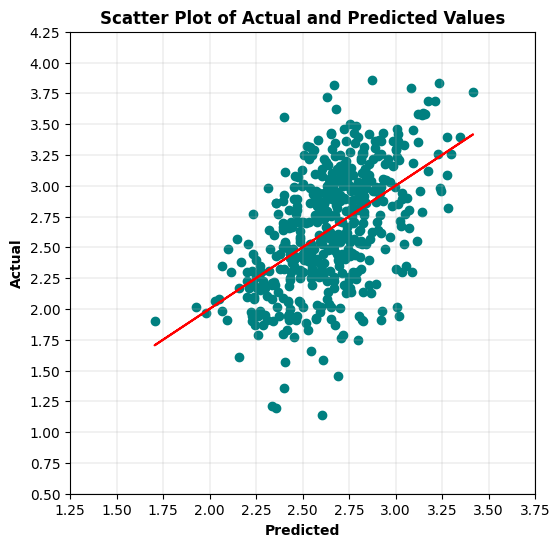
Intercept 1.324193  
education 0.049325  
age 0.017283  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_other['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 114.817

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_other[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,4,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.276  
Model: OLS Adj. R-squared: 0.273  
Method: Least Squares F-statistic: 91.64  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.91e-34  
Time: 10:05:43 Log-Likelihood: -260.46  
No. Observations: 484 AIC: 526.9  
Df Residuals: 481 BIC: 539.5  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.3242 0.100 13.276 0.000 1.128 1.520  
education 0.0493 0.005 10.319 0.000 0.040 0.059  
age 0.0173 0.002 10.875 0.000 0.014 0.020  
=============================================================================  
Omnibus: 10.025 Durbin-Watson: 1.829  
Prob(Omnibus): 0.007 Jarque-Bera (JB): 10.043  
Skew: -0.331 Prob(JB): 0.00659  
Kurtosis: 3.242 Cond. No. 234.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 83.139  
SSE = 31.678   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



5c. After reviewing the question semantically, and thinking about what the professor had mentioned, I suspected that he may have wanted to see a MLR model estimating logarithmically converted wages with age and education variables for each grouped combination of the sex and language values.

# Create individual dataframes associated with each grouped combination of sex and language values.  
Wages\_f\_english = Wages\_female[Wages\_female.language == 'English']  
Wages\_f\_french = Wages\_female[Wages\_female.language == 'French']  
Wages\_f\_other = Wages\_female[Wages\_female.language == 'Other']  
Wages\_m\_english = Wages\_male[Wages\_male.language == 'English']  
Wages\_m\_french = Wages\_male[Wages\_male.language == 'French']  
Wages\_m\_other = Wages\_male[Wages\_male.language == 'Other']

# Generate required multiple linear regression model for the Female and English group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_f\_english).fit()  
# View model coefficients.  
print(lm.params)

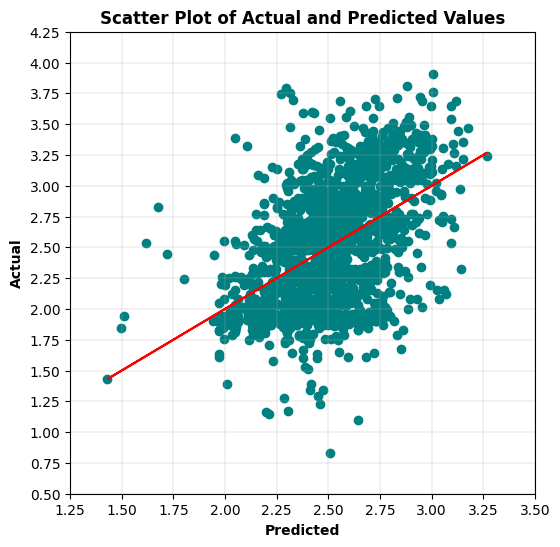
Intercept 1.064363  
education 0.067182  
age 0.014537  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_f\_english['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 394.759

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_f\_english[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,3.75,step=0.25))   
plt.yticks(np.arange(0.5,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.249  
Model: OLS Adj. R-squared: 0.248  
Method: Least Squares F-statistic: 270.4  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 3.61e-102  
Time: 10:06:06 Log-Likelihood: 924.40  
No. Observations: 1636 AIC: 1855.  
Df Residuals: 1633 BIC: 1871.  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.0644 0.064 16.585 0.000 0.938 1.190  
education 0.0672 0.004 17.285 0.000 0.060 0.075  
age 0.0145 0.001 16.689 0.000 0.013 0.016  
=============================================================================  
Omnibus: 1.101 Durbin-Watson: 2.074  
Prob(Omnibus): 0.577 Jarque-Bera (JB): 1.021  
Skew: -0.000 Prob(JB): 0.600  
Kurtosis: 3.122 Cond. No. 247.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 296.546  
SSE = 98.214   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Generate required multiple linear regression model for the Female and French group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_f\_french).fit()  
# View model coefficients.  
print(lm.params)

Intercept 1.191763  
education 0.063748  
age 0.012791  
dtype: float64

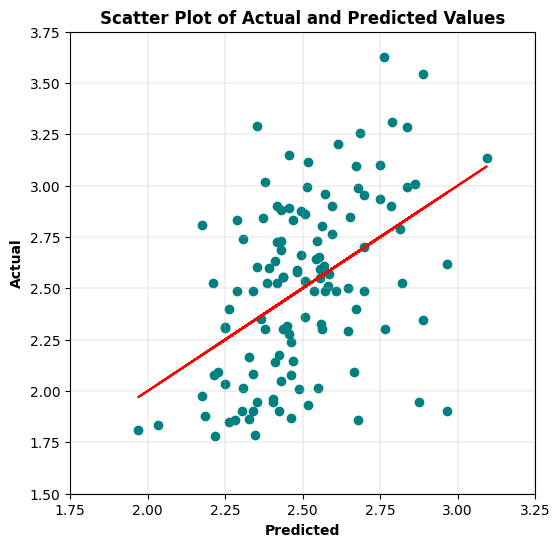
# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_f\_french['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 22.062

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_f\_french[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.75,3.5,step=0.25))   
plt.yticks(np.arange(1.5,4,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.217  
Model: OLS Adj. R-squared: 0.204  
Method: Least Squares F-statistic: 16.09  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 6.78e-07  
Time: 10:06:23 Log-Likelihood: -54.008  
No. Observations: 119 AIC: 114.0  
Df Residuals: 116 BIC: 122.4  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.1918 0.234 5.101 0.000 0.729 1.655  
education 0.0637 0.013 4.850 0.000 0.038 0.090  
age 0.0128 0.003 4.112 0.000 0.007 0.019  
=============================================================================

Omnibus: 0.266 Durbin-Watson: 2.239  
Prob(Omnibus): 0.876 Jarque-Bera (JB): 0.434  
Skew: -0.064 Prob(JB): 0.805  
Kurtosis: 2.734 Cond. No. 270.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 17.270  
SSE = 4.792   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Generate required multiple linear regression model for the Female and Other group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_f\_other).fit()  
# View model coefficients.  
print(lm.params)

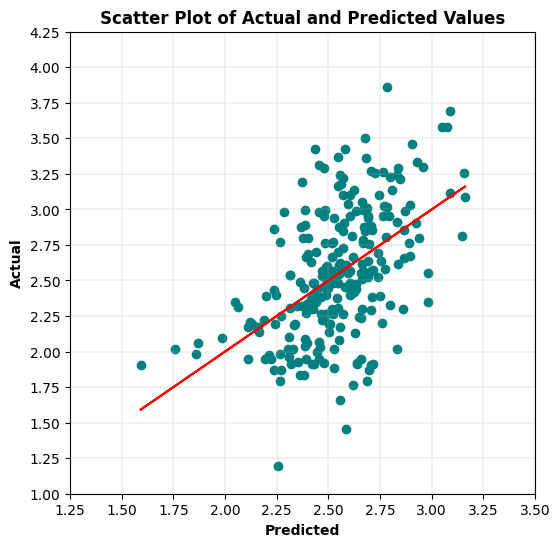
Intercept 1.310070  
education 0.056881  
age 0.012777  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_f\_other['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 51.008

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_f\_other[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.25,3.75,step=0.25))   
plt.yticks(np.arange(1,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.270  
Model: OLS Adj. R-squared: 0.264  
Method: Least Squares F-statistic: 44.94  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 2.47e-17  
Time: 10:06:39 Log-Likelihood: -116.82  
No. Observations: 246 AIC: 239.6  
Df Residuals: 243 BIC: 250.2  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.3101 0.138 9.509 0.000 1.039 1.581  
education 0.0569 0.007 8.718 0.000 0.044 0.070  
age 0.0128 0.002 5.743 0.000 0.008 0.017  
=============================================================================  
Omnibus: 0.331 Durbin-Watson: 1.770  
Prob(Omnibus): 0.848 Jarque-Bera (JB): 0.142  
Skew: -0.030 Prob(JB): 0.932  
Kurtosis: 3.101 Cond. No. 240.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 37.235  
SSE = 13.773   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Generate required multiple linear regression model for the Male and English group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_m\_english).fit()  
# View model coefficients.  
print(lm.params)

Intercept 1.255115  
education 0.050261  
age 0.021827  
dtype: float64

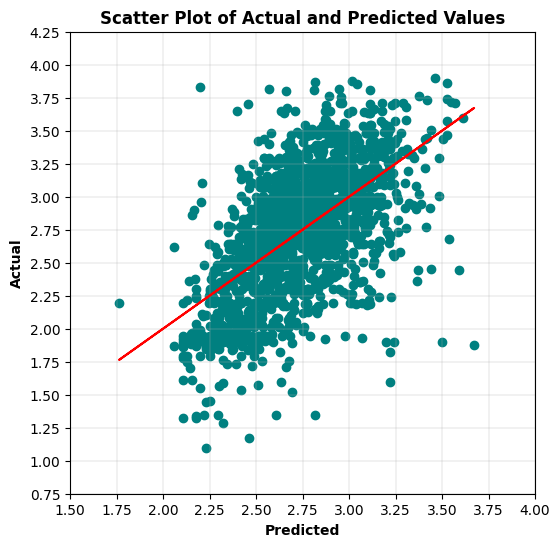
# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_m\_english['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 396.891

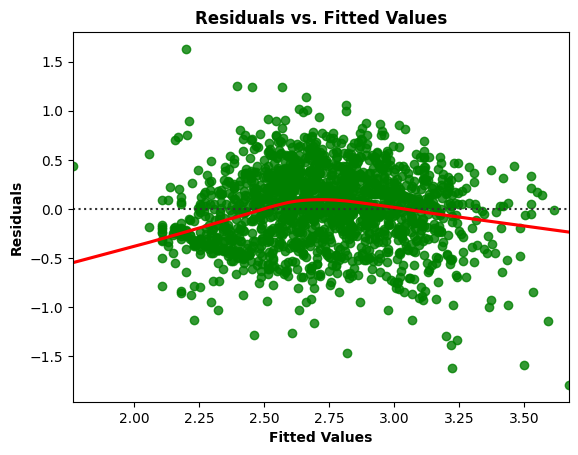
# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_m\_english[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.5,4.25,step=0.25))   
plt.yticks(np.arange(0.75,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.348  
Model: OLS Adj. R-squared: 0.347  
Method: Least Squares F-statistic: 428.0  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.03e-149  
Time: 10:06:55 Log-Likelihood: 813.09  
No. Observations: 1608 AIC: 1632.  
Df Residuals: 1605 BIC: 1648.  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.2551 0.057 21.834 0.000 1.142 1.368  
education 0.0503 0.003 14.645 0.000 0.044 0.057  
age 0.0218 0.001 26.090 0.000 0.020 0.023  
=============================================================================

Omnibus: 37.494 Durbin-Watson: 2.097  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 48.787  
Skew: -0.277 Prob(JB): 2.55e11  
Kurtosis: 3.650 Cond. No. 233.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 258.832  
SSE = 138.059   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



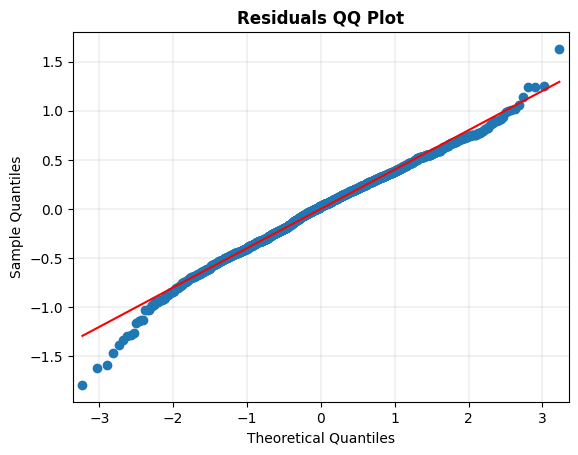
# Display Residuals vs. Fitted values scatter plot.  
fitted = lm.fittedvalues  
residuals = lm.resid  
sns.residplot(x=fitted, y=residuals, lowess=True, line\_kws={'color': 'red'}, color='green')   
plt.title('Residuals vs. Fitted Values', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel('Residuals', fontweight='bold')  
plt.show()



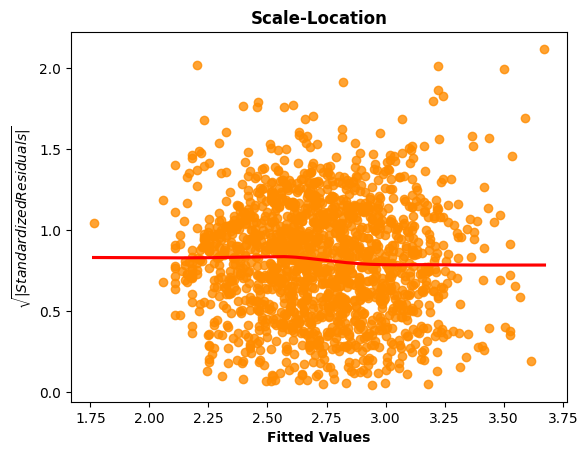
# Residuals Anderson-Darling Normality Test  
result = anderson(residuals)  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 1.738  
15.000: 0.575, data does not look normal (reject H0)  
10.000: 0.654, data does not look normal (reject H0)  
5.000: 0.785, data does not look normal (reject H0)  
2.500: 0.916, data does not look normal (reject H0)  
1.000: 1.089, data does not look normal (reject H0)

# Residuals QQ Plot  
qqplot(residuals, line='s')  
plt.title('Residuals QQ Plot', fontweight='bold')  
plt.grid(linewidth=0.25)  
plt.show()



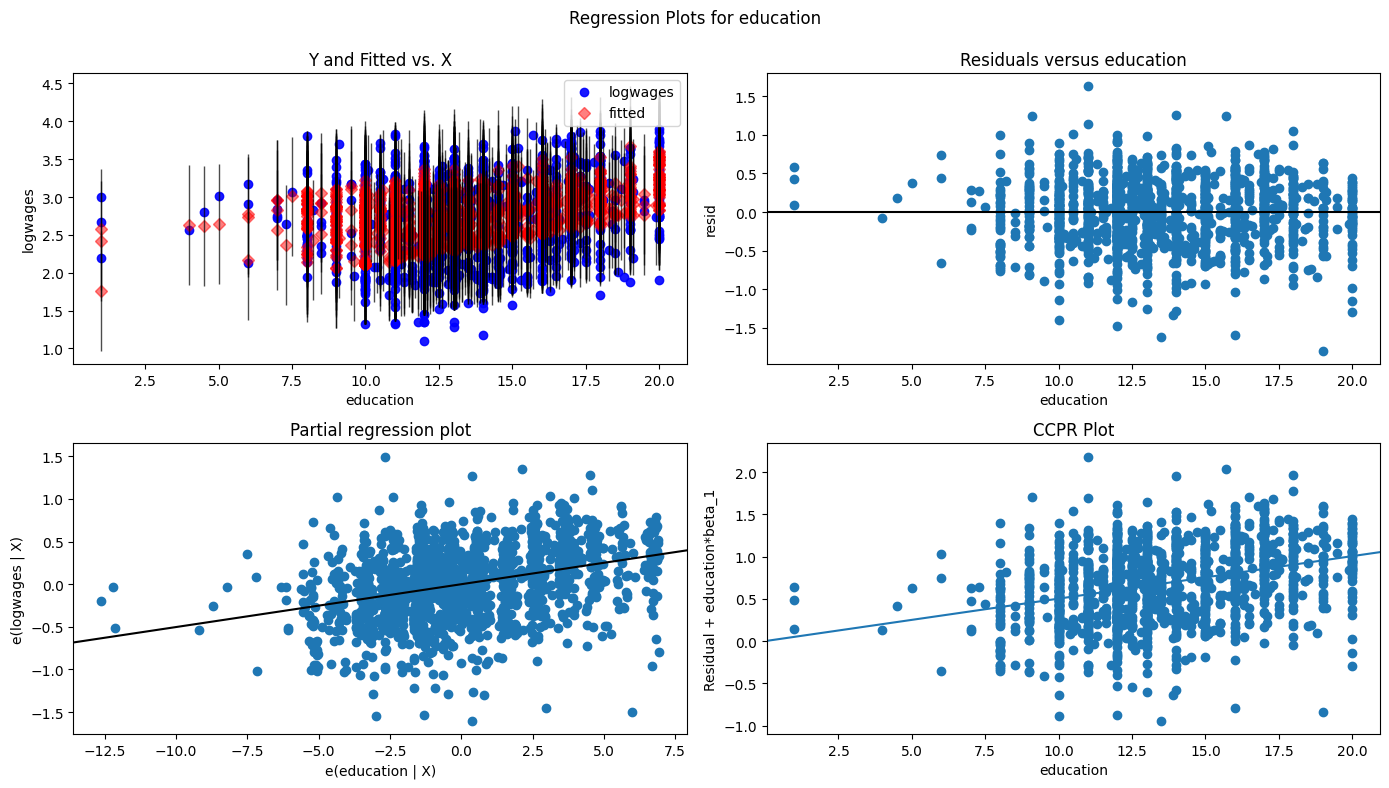
# Scale-Location Plot  
resid\_standardized = lm.get\_influence().resid\_studentized\_internal  
  
sns.regplot(x=fitted, y=np.sqrt(np.abs(resid\_standardized)), color='darkorange',  
 ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-Location', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel(r'$\sqrt{|Standardized Residuals|}$', fontweight='bold')  
plt.show()



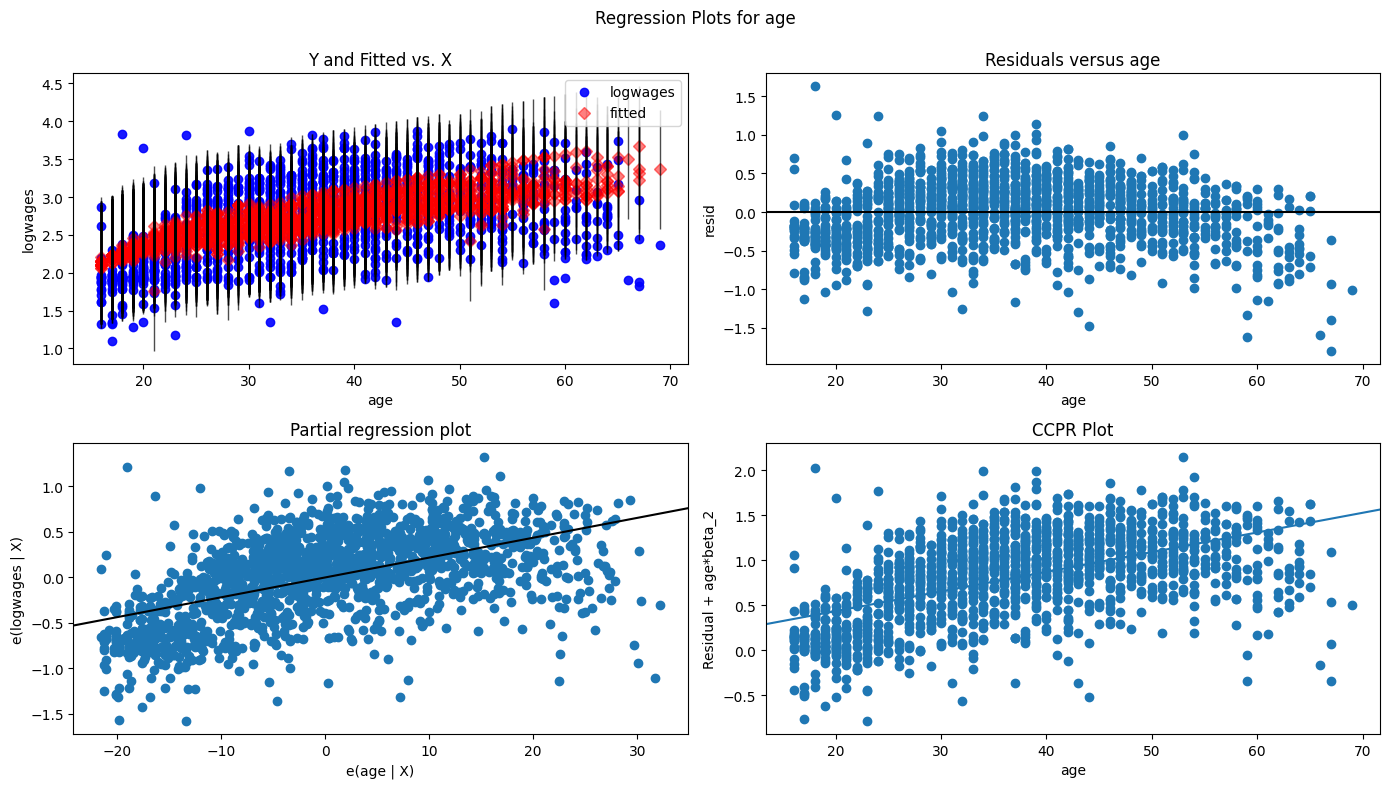
# Generate the Breusch-Pagan test for heteroscedasticity.  
bp\_test = het\_breuschpagan(lm.resid, lm.model.exog)  
print("lm:", f"{bp\_test[0]:0,.3f}", "lm\_pvalue:", f"{bp\_test[1]:0,.3f}")

lm: 5.679 lm\_pvalue: 0.058

# Display model regression plots for the 'education' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'education',fig=fig)  
plt.show()



# Display model regression plots for the 'age' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'age',fig=fig)  
plt.show()



# Generate required multiple linear regression model for the Male and French group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_m\_french).fit()  
# View model coefficients.  
print(lm.params)

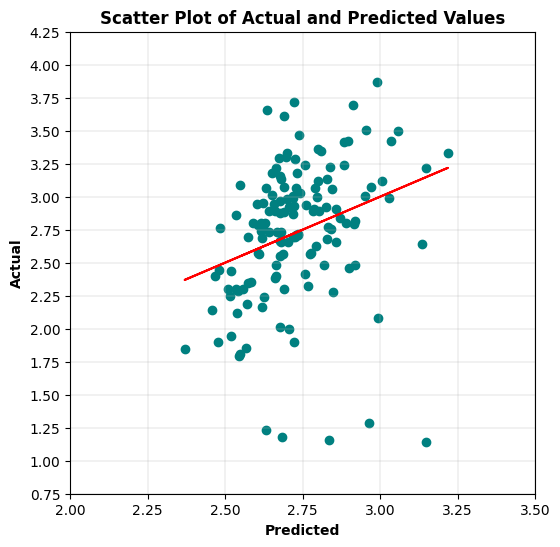
Intercept 1.983874  
education 0.020630  
age 0.012507  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_m\_french['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 37.890

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_m\_french[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(2,3.75,step=0.25))   
plt.yticks(np.arange(0.75,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.087  
Model: OLS Adj. R-squared: 0.074  
Method: Least Squares F-statistic: 6.521  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 0.00197  
Time: 10:07:51 Log-Likelihood: -100.80  
No. Observations: 140 AIC: 207.6  
Df Residuals: 137 BIC: 216.4  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.9839 0.230 8.636 0.000 1.530 2.438  
education 0.0206 0.012 1.668 0.098 -0.004 0.045  
age 0.0125 0.004 3.383 0.001 0.005 0.020  
=============================================================================  
Omnibus: 38.978 Durbin-Watson: 1.999  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 80.949  
Skew: -1.199 Prob(JB): 2.64e-18  
Kurtosis: 5.850 Cond. No. 227.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 34.596  
SSE = 3.294   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05



# Generate required multiple linear regression model for the Male and Other group.  
# Fit linear regression model.  
lm = smf.ols("logwages ~ education + age", data = Wages\_m\_other).fit()  
# View model coefficients.  
print(lm.params)

Intercept 1.394755  
education 0.042414  
age 0.020101  
dtype: float64

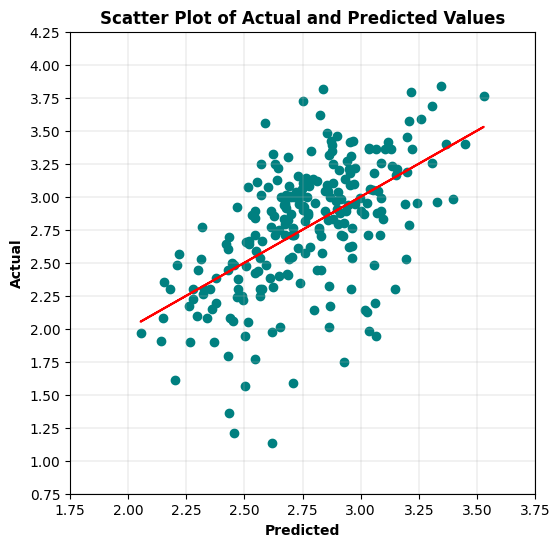
# Calculate the Total Sum of Squares (SST) for an empty model for 'logwages'.  
Y = Wages\_m\_other['logwages'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 57.983

# Generate OLS Regression results for the multiple linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,2,3)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=2))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Wages\_m\_other[['education','age']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, 1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(1.75,4,step=0.25))   
plt.yticks(np.arange(0.75,4.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: logwages R-squared: 0.321  
Model: OLS Adj. R-squared: 0.316  
Method: Least Squares F-statistic: 55.64  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.65e-20  
Time: 10:08:06 Log-Likelihood: -123.53  
No. Observations: 238 AIC: 253.1  
Df Residuals: 235 BIC: 263.5  
Df Model: 2   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 1.3948 0.135 10.367 0.000 1.130 1.660  
education 0.0424 0.007 6.523 0.000 0.030 0.055  
age 0.0201 0.002 9.471 0.000 0.016 0.024  
=============================================================================

Omnibus: 26.836 Durbin-Watson: 1.854  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 35.104  
Skew: -0.757 Prob(JB): 2.38e-08  
Kurtosis: 4.116 Cond. No. 229.  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 39.350  
SSE = 18.633   
  
F\_critical: 9.552  
Absolute t\_critical: 4.303  
Alpha Standard: 0.05

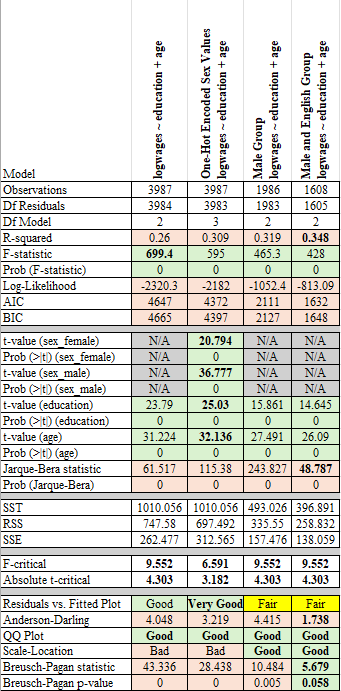


5 Select the grouping variable that creates the best model. Explain the performance difference (if any) over the previous models.

Multiple OLS regression models were run with logwages as the outcome variable along with education and age as explanatory variables. Two MLR models caught my eye. First, Model A had One-Hot Encoded Sex values added to the MLR model, and the second, Model B used the Male and English group to generate the MLR model. Model A used the same number of observations (3,987) as the original, and Model B used far less observations (1,608). Model B had the best R-square result of 0.348, Model A R-square result was 0.309, and the original model had 0.26 as a R-square result. The original model had the best F-statistic of 699.4, but Model A had 595 as a F-statistic with an additional degree of freedom. All three models had a F-statistic probability near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating clearly that the model relationship between the outcome and explanatory variables is statistically significant. Model A had the best t-statistic values for the education and age variables, and solid t-statistic values for the binary sex variables as well. All three model's t-statistics exceeded the absolute t-critical, and had p-values near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating clearly that their relationships with logwages are statistically significant.

With logwages having a SST of 1,010.056, Model A outperformed the original model by reducing it 312.565 to a SSR of 697.492, but given Model B had far fewer observations, it did not make sense to add it to this comparison. When plotting the residuals and the fitted value, Model A depicted very good linearity, the orignal model depicted reasonably good linearity, and Model B depicted was fair at best. Concerning normality of the residuals, Model B had the best Jarque-Bera test statistic of 48.787, but all three models had a Jarque-Bera probability near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating a lack of normality. Although, Model B's Anderson-Darling result of 1.738 barely supports that conclusion, and QQ plot rendering shows potentially acceptable normality for all three models. Of the three models, only Model B's Scale-Location chart showed a good horizontal progression that indicates potential homoscedasticity in the model, and the Breusch-Pagan test statistic of 5.679 and p-value of 0.058, slightly greater than the 0.05 alpha standard (accepting H0 the null hypothesis), indicate that some level of homoscedasticity is present in the model. All three models had a low R squared value, low Log-Likelihood value, and high AIC/BIC values basically indicates that the MLR models would not be good predictors of logwages.

A B



I guess if had to choose one of the models, it would probably be Model A, which had One-Hot Encoded Sex values added to the model. Merely because it uses all the observations and has very good model linearity without sacrificing much otherwise. Was highly tempted to run a transformation on the education explanatory variable, even though it has some zero values, just to see what that might yield. But ended up making the judgement call that more variables would be necessary to improve this model enough for any realistic use.

# Load the required data for the second part of assigment.  
Fert\_df = pd.read\_csv(DATA / 'Fertility\_UN.csv')  
  
# View some initial records.  
Fert\_df.head()

Fertility PPgdp LogFertility LogPPgdp  
0 6.80 98 1.916923 4.584967  
1 2.28 1317 0.824175 7.183112  
2 2.80 1784 1.029619 7.486613  
3 7.20 739 1.974081 6.605298  
4 2.44 7163 0.891998 8.876684

# Generate dataframe dimensions.  
Fert\_df.shape

(193, 4)

# Generate variable data types.  
Fert\_df.dtypes

Fertility float64  
PPgdp int64  
LogFertility float64  
LogPPgdp float64  
dtype: object

# Generate number of missing values.  
Fert\_df.isna().sum()

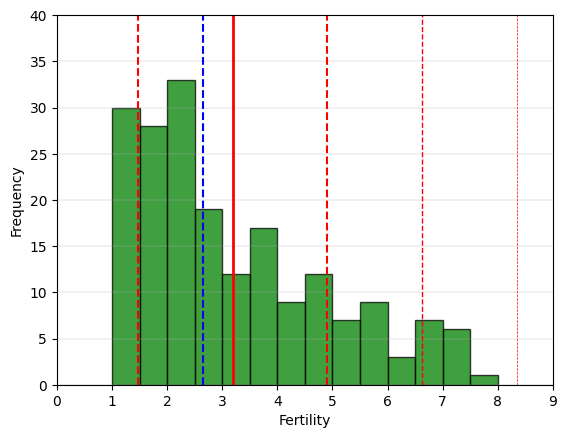
Fertility 0  
PPgdp 0  
LogFertility 0  
LogPPgdp 0  
dtype: int64

Specific Evaluation and Preparation of Dataset Variables

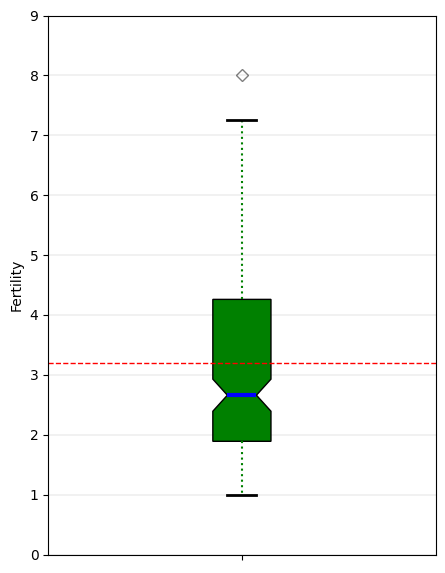
Fert\_df.describe().map('{:,.3f}'.format)

Fertility PPgdp LogFertility LogPPgdp  
count 193.000 193.000 193.000 193.000  
mean 3.189 6,408.425 1.018 7.620  
std 1.717 9,377.946 0.537 1.646  
min 1.000 90.000 0.000 4.500  
25% 1.890 467.000 0.637 6.146  
50% 2.660 1,938.000 0.978 7.569  
75% 4.260 7,850.000 1.449 8.968  
max 8.000 44,579.000 2.079 10.705

# Histogram for the 'Fertility' continuous variable.  
plt.hist(Fert\_df['Fertility'], bins = 14, alpha = 0.75, color = 'green', edgecolor = 'black')  
plt.xlabel('Fertility')  
plt.ylabel('Frequency')  
plt.xlim(0, 9)  
plt.xticks(np.arange(0,10,step=1))  
plt.yticks(np.arange(0,45,step=5))  
  
mean\_value = Fert\_df['Fertility'].mean()  
median\_value = Fert\_df['Fertility'].median()  
std\_value = Fert\_df['Fertility'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



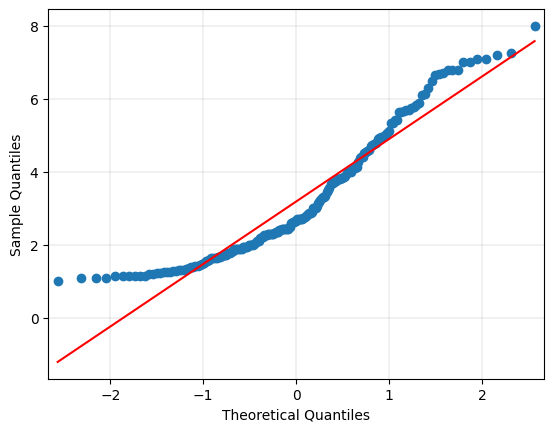
# Boxplot for the 'Fertility' continuous variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Fert\_df['Fertility'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('Fertility')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(0,10,step=1))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



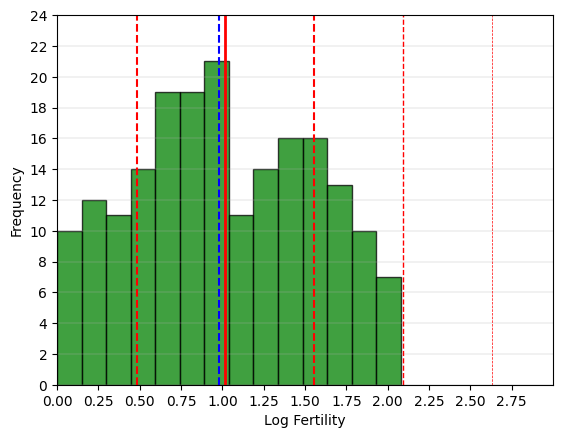
# 'Fertility' Anderson-Darling Normality Test  
result = anderson(Fert\_df['Fertility'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 5.570  
15.000: 0.565, data does not look normal (reject H0)  
10.000: 0.643, data does not look normal (reject H0)  
5.000: 0.772, data does not look normal (reject H0)  
2.500: 0.900, data does not look normal (reject H0)  
1.000: 1.071, data does not look normal (reject H0)

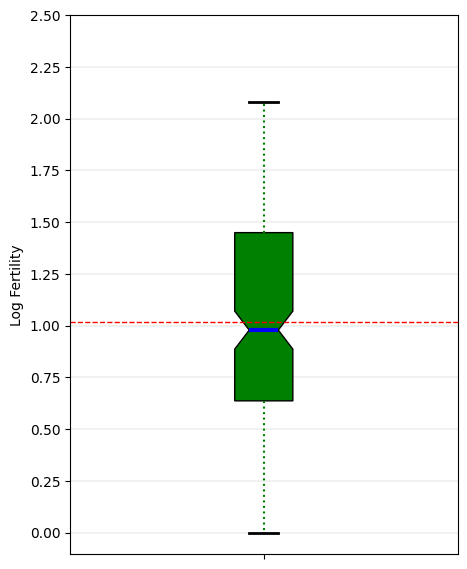
# 'Fertility' QQ plot  
qqplot(Fert\_df['Fertility'], line='s')  
plt.grid(linewidth=0.25)  
plt.show()



# Histogram for the 'LogFertility' continuous transformed variable.  
plt.hist(Fert\_df['LogFertility'], bins = 14, alpha = 0.75, color = 'green', edgecolor = 'black')  
plt.xlabel('Log Fertility')  
plt.ylabel('Frequency')  
plt.xlim(0,3)  
plt.xticks(np.arange(0,3,step=0.25))  
plt.yticks(np.arange(0,26,step=2))  
mean\_value = Fert\_df['LogFertility'].mean()  
median\_value = Fert\_df['LogFertility'].median()  
std\_value = Fert\_df['LogFertility'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



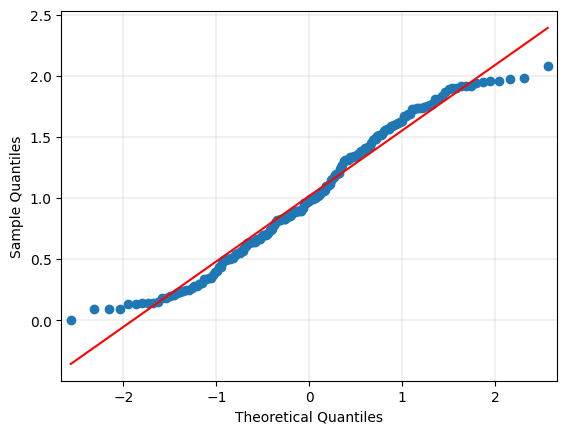
# Boxplot for the 'LogFertility' continuous variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Fert\_df['LogFertility'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('Log Fertility')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(0,2.75,step=0.25))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



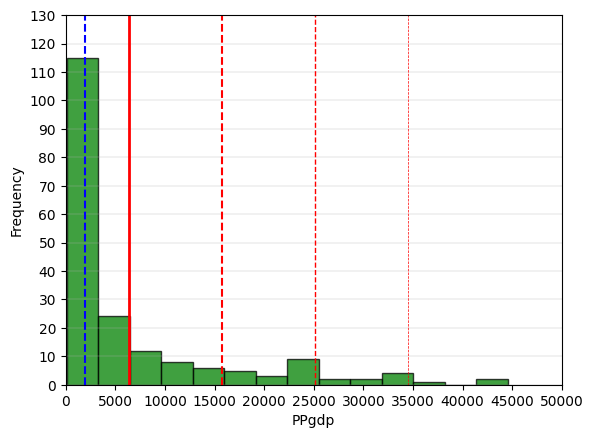
# 'LogFertility' Anderson-Darling Normality Test  
result = anderson(Fert\_df['LogFertility'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 1.362  
15.000: 0.565, data does not look normal (reject H0)  
10.000: 0.643, data does not look normal (reject H0)  
5.000: 0.772, data does not look normal (reject H0)  
2.500: 0.900, data does not look normal (reject H0)  
1.000: 1.071, data does not look normal (reject H0)

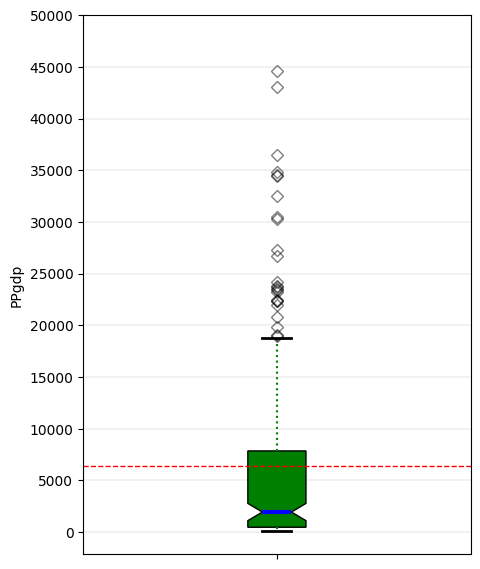
# 'LogFertility' QQ plot  
qqplot(Fert\_df['LogFertility'], line='s')  
plt.grid(linewidth=0.25)  
plt.show()



# Histogram for the 'PPgdp' continuous variable.  
plt.hist(Fert\_df['PPgdp'], bins = 14, alpha = 0.75, color = 'green', edgecolor = 'black')  
plt.xlabel('PPgdp')  
plt.ylabel('Frequency')  
plt.xlim(0, 9)  
plt.xticks(np.arange(0,50001,step=5000))  
plt.yticks(np.arange(0,140,step=10))  
  
mean\_value = Fert\_df['PPgdp'].mean()  
median\_value = Fert\_df['PPgdp'].median()  
std\_value = Fert\_df['PPgdp'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



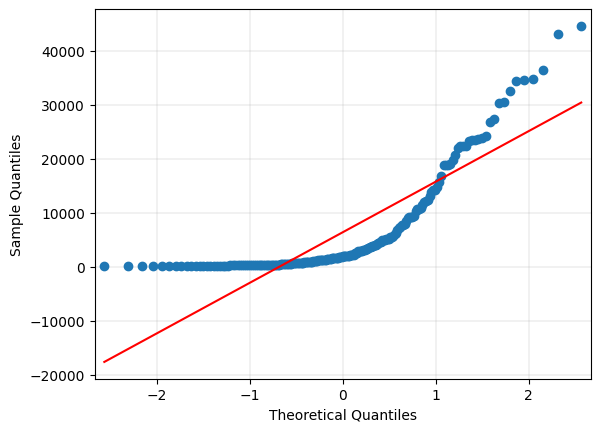
# Boxplot for the 'PPgdp' continuous variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Fert\_df['PPgdp'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('PPgdp')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(0,50001,step=5000))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



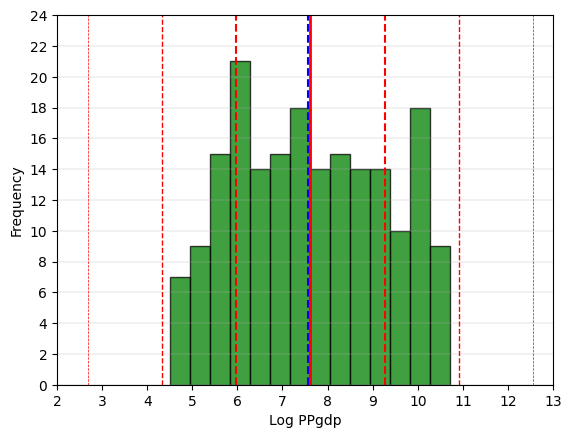
# 'PPgdp' Anderson-Darling Normality Test  
result = anderson(Fert\_df['PPgdp'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 22.292  
15.000: 0.565, data does not look normal (reject H0)  
10.000: 0.643, data does not look normal (reject H0)  
5.000: 0.772, data does not look normal (reject H0)  
2.500: 0.900, data does not look normal (reject H0)  
1.000: 1.071, data does not look normal (reject H0)

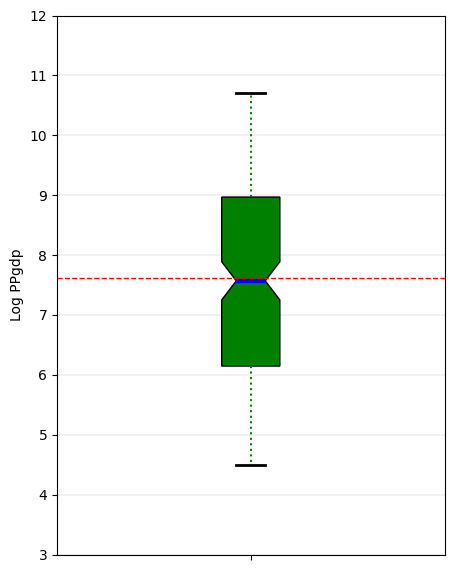
# 'PPgdp' QQ plot  
qqplot(Fert\_df['PPgdp'], line='s')  
plt.grid(linewidth=0.25)  
plt.show()



# Histogram for the 'LogPPgdp' continuous transformed variable.  
plt.hist(Fert\_df['LogPPgdp'], bins = 14, alpha = 0.75, color = 'green', edgecolor = 'black')  
plt.xlabel('Log PPgdp')  
plt.ylabel('Frequency')  
plt.xlim(2,12)  
plt.xticks(np.arange(2,14,step=1))  
plt.yticks(np.arange(0,26,step=2))  
mean\_value = Fert\_df['LogPPgdp'].mean()  
median\_value = Fert\_df['LogPPgdp'].median()  
std\_value = Fert\_df['LogPPgdp'].std()  
  
plt.axvline(mean\_value, color='red', linewidth=2, label=f'Mean: {mean\_value: .2f}')  
plt.axvline(median\_value, color='blue', linestyle='dashed', linewidth=1.5, label=f'Median: {median\_value: .2f}')  
plt.axvline(mean\_value+std\_value, color='red', linestyle='dashed', linewidth=1.5, label=f'1 SD')  
plt.axvline(mean\_value-std\_value, color='red', linestyle='dashed', linewidth=1.5)  
plt.axvline(mean\_value+2\*std\_value, color='red', linestyle='dashed', linewidth=1, label=f'2 SD')  
plt.axvline(mean\_value-2\*std\_value, color='red', linestyle='dashed', linewidth=1)  
plt.axvline(mean\_value+3\*std\_value, color='red', linestyle='dashed', linewidth=0.5, label=f'3 SD')  
plt.axvline(mean\_value-3\*std\_value, color='red', linestyle='dashed', linewidth=0.5)  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



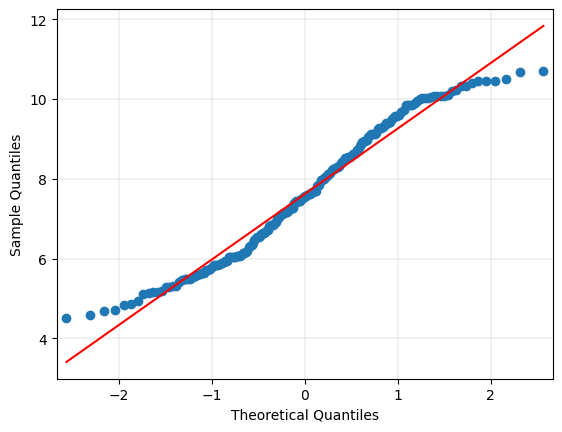
# Boxplot for the 'LogPPgdp' continuous variable.  
fig = plt.figure(figsize =(5, 7))  
ax = fig.add\_subplot(111)  
  
# Creating axes instance  
bp = ax.boxplot(Fert\_df['LogPPgdp'], patch\_artist = True,  
 notch ='True')  
  
for patch, color in zip(bp['boxes'], 'green'):  
 patch.set\_facecolor('green')  
  
# Changing color and linewidth of whiskers.  
for whisker in bp['whiskers']:  
 whisker.set(color ='green',  
 linewidth = 1.5,  
 linestyle =":")  
  
# Changing color and linewidth of caps.  
for cap in bp['caps']:  
 cap.set(color ='black',  
 linewidth = 2)  
  
# Changing color and linewidth of median.  
for median in bp['medians']:  
 median.set(color ='blue',  
 linewidth = 3)  
   
# Changing style of fliers.  
for flier in bp['fliers']:  
 flier.set(marker ='D',  
 alpha = 0.5)  
   
# Set axis labels.  
ax.set\_ylabel('Log PPgdp')   
ax.set\_xticklabels('')  
  
# Set axis limits.  
plt.yticks(np.arange(3,13,step=1))   
  
# Display the mean.  
plt.axhline(mean\_value, color='red', linewidth=1, linestyle='dashed', label=f'Mean: {mean\_value: .2f}')  
  
plt.grid(axis='y', linewidth=0.25)  
plt.show()



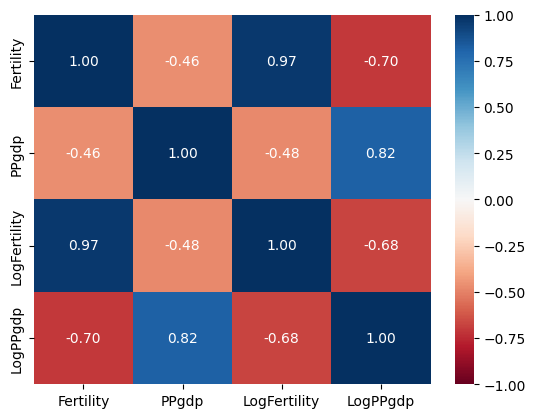
# 'LogPPgdp' Anderson-Darling Normality Test  
result = anderson(Fert\_df['LogPPgdp'])  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 1.738  
15.000: 0.565, data does not look normal (reject H0)  
10.000: 0.643, data does not look normal (reject H0)  
5.000: 0.772, data does not look normal (reject H0)  
2.500: 0.900, data does not look normal (reject H0)  
1.000: 1.071, data does not look normal (reject H0)

# 'LogPPgdp' QQ plot  
qqplot(Fert\_df['LogPPgdp'], line='s')  
plt.grid(linewidth=0.25)  
plt.show()



# Generate heatmap for correlation matrix analysis.  
corr = Fert\_df.corr()  
fig, ax = plt.subplots()  
sns.heatmap(corr, annot=True, fmt=".2f", xticklabels=corr.columns, yticklabels=corr.columns, vmin=-1, vmax=1, cmap="RdBu", ax=ax)  
plt.show()

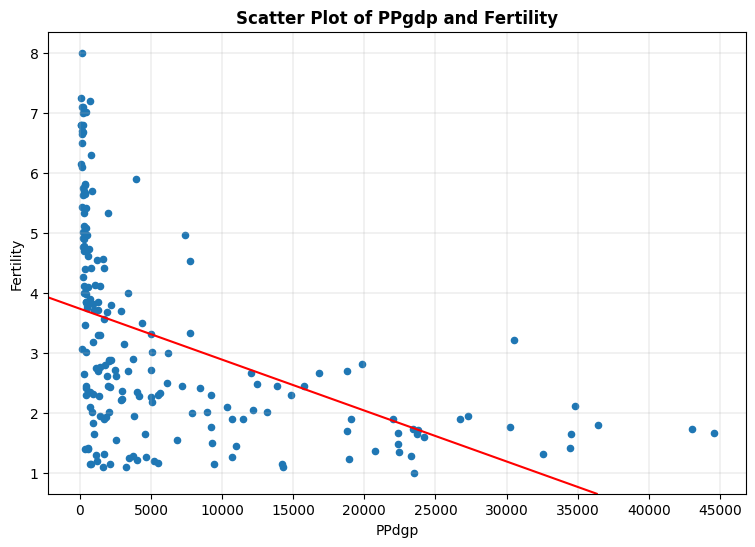


1. The data for the UN study on fertility contains GDP and fertility rates for 193 nations. It also contains log transforms for both variables. Generate two scatter plots of PPgdp vs. Fertility, one using linear scaling for both variables and the other using log scaling for both variables.

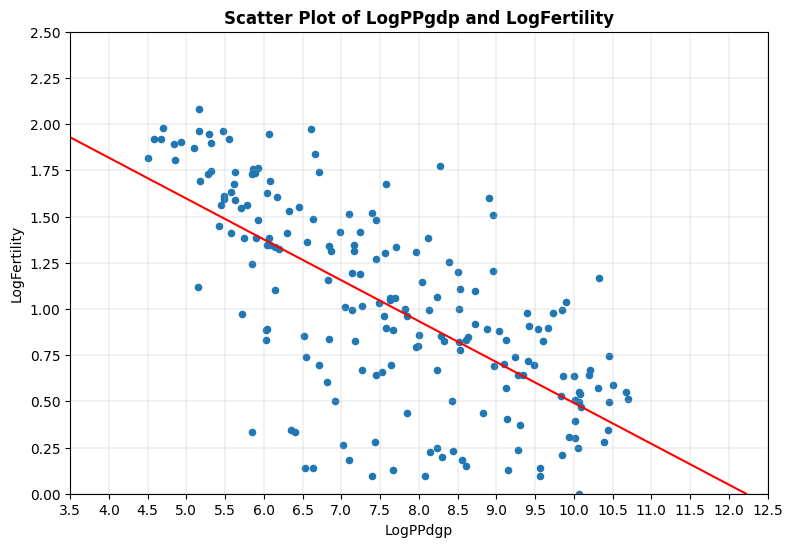
* a. Perform regressions using both log and linear transformations of the data.

b. Which regression exhibits a better fit (log or linear)? Use the ANOVA summary to support your reasoning.

# Generate a scatter plot for the continuous variables 'PPgdp' and 'Fertility'.  
ax = Fert\_df.plot.scatter('PPgdp',  
 'Fertility',  
 grid=True,  
 figsize=(9,6))  
  
ax.set\_title('Scatter Plot of PPgdp and Fertility',  
 weight='bold')  
ax.set\_xlabel('PPdgp')  
ax.set\_ylabel('Fertility')  
  
plt.xticks(np.arange(0,45001,step=5000))   
plt.yticks(np.arange(0,9,step=1))   
plt.grid(linewidth=0.25)  
  
X = Fert\_df['PPgdp'].tolist()  
Y = Fert\_df['Fertility'].tolist()  
  
m, b = np.polyfit(X, Y, deg=1)  
plt.axline(xy1=(0, b), slope=m, color='r')  
plt.show()



# Generate a scatter plot for the continuous variables 'LogPPgdp' and 'LogFertility'.  
ax = Fert\_df.plot.scatter('LogPPgdp',  
 'LogFertility',  
 grid=True,  
 figsize=(9,6))  
  
ax.set\_title('Scatter Plot of LogPPgdp and LogFertility',  
 weight='bold')  
ax.set\_xlabel('LogPPdgp')  
ax.set\_ylabel('LogFertility')  
  
plt.xlim(3.5,12.5)  
plt.xticks(np.arange(3.5,13,step=0.5))   
plt.ylim(0,2.50)  
plt.yticks(np.arange(0,2.75,step=0.25))   
plt.grid(linewidth=0.25)  
  
X = Fert\_df['LogPPgdp'].tolist()  
Y = Fert\_df['LogFertility'].tolist()  
  
m, b = np.polyfit(X, Y, deg=1)  
plt.axline(xy1=(0, b), slope=m, color='r')  
plt.show()



6a. Perform regressions using both log and linear transformations of the data.

# Generate required linear regression model for 'Fertility' outcome variable and 'PPgdp' explanatory variable.  
# Fit linear regression model.  
lm = smf.ols("Fertility ~ PPgdp", data = Fert\_df).fit()  
# View model coefficients.  
print(lm.params)

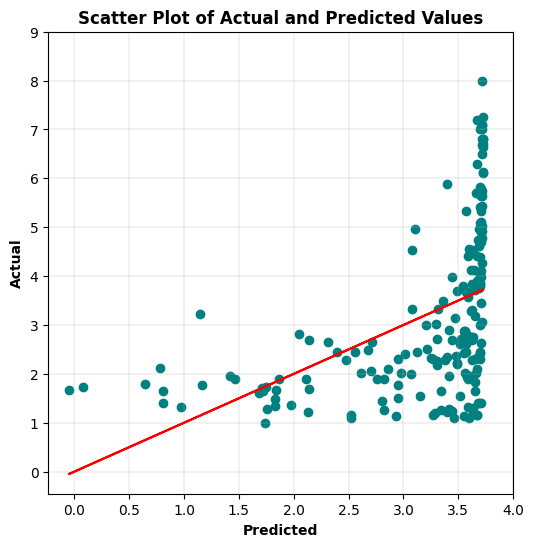
Intercept 3.733282  
PPgdp -0.000085  
dtype: float64

# Calculate the Total Sum of Squares (SST) for an empty model for 'Fertility'.  
Y = Fert\_df['Fertility'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

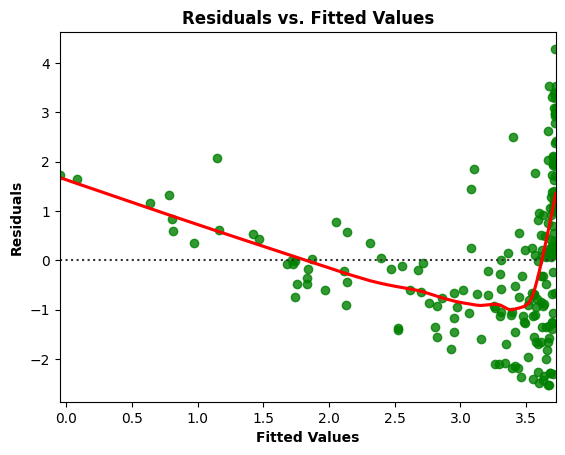
SST = 566.292

# Generate OLS Regression results for the linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,1,2)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=1))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Fert\_df[['PPgdp']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, deg=1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(0,4.5,step=0.5))   
plt.yticks(np.arange(0,10,step=1))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: Fertility R-squared: 0.215  
Model: OLS Adj. R-squared: 0.211  
Method: Least Squares F-statistic: 52.22  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 1.15e-11  
Time: 12:32:16 Log-Likelihood: -354.41  
No. Observations: 193 AIC: 712.8  
Df Residuals: 191 BIC: 719.3  
Df Model: 1   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 3.7333 0.133 28.040 0.000 3.471 3.996  
PPgdp -8.486e-05 1.17e-05 -7.226 0.000 -0.000 -6.17e-05  
=============================================================================  
Omnibus: 9.811 Durbin-Watson: 1.845  
Prob(Omnibus): 0.007 Jarque-Bera (JB): 10.371  
Skew: 0.548 Prob(JB): 0.00560  
Kurtosis: 2.704 Cond. No. 1.37e+04  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The condition number is large, 1.37e+04. This might indicate that there are  
strong multicollinearity or other numerical problems.  
  
 ====================   
  
RSS = 444.703  
SSE = 121.588   
  
F\_critical: 18.513  
Absolute t\_critical: 12.706  
Alpha Standard: 0.05



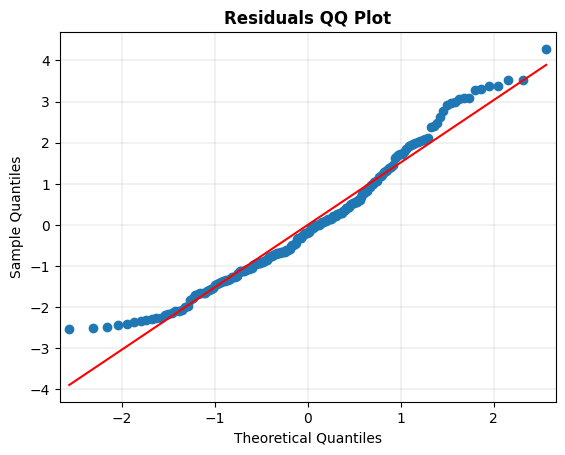
# Display Residuals vs. Fitted values scatter plot.  
fitted = lm.fittedvalues  
residuals = lm.resid  
sns.residplot(x=fitted, y=residuals, lowess=True, line\_kws={'color': 'red'}, color='green')   
plt.title('Residuals vs. Fitted Values', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel('Residuals', fontweight='bold')  
plt.show()



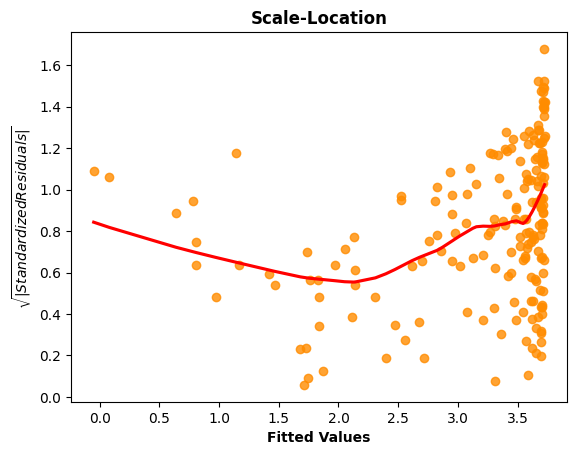
# Residuals Anderson-Darling Normality Test  
result = anderson(residuals)  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 1.714  
15.000: 0.565, data does not look normal (reject H0)  
10.000: 0.643, data does not look normal (reject H0)  
5.000: 0.772, data does not look normal (reject H0)  
2.500: 0.900, data does not look normal (reject H0)  
1.000: 1.071, data does not look normal (reject H0)

# Residuals QQ Plot  
qqplot(residuals, line='s')  
plt.title('Residuals QQ Plot', fontweight='bold')  
plt.grid(linewidth=0.25)  
plt.show()



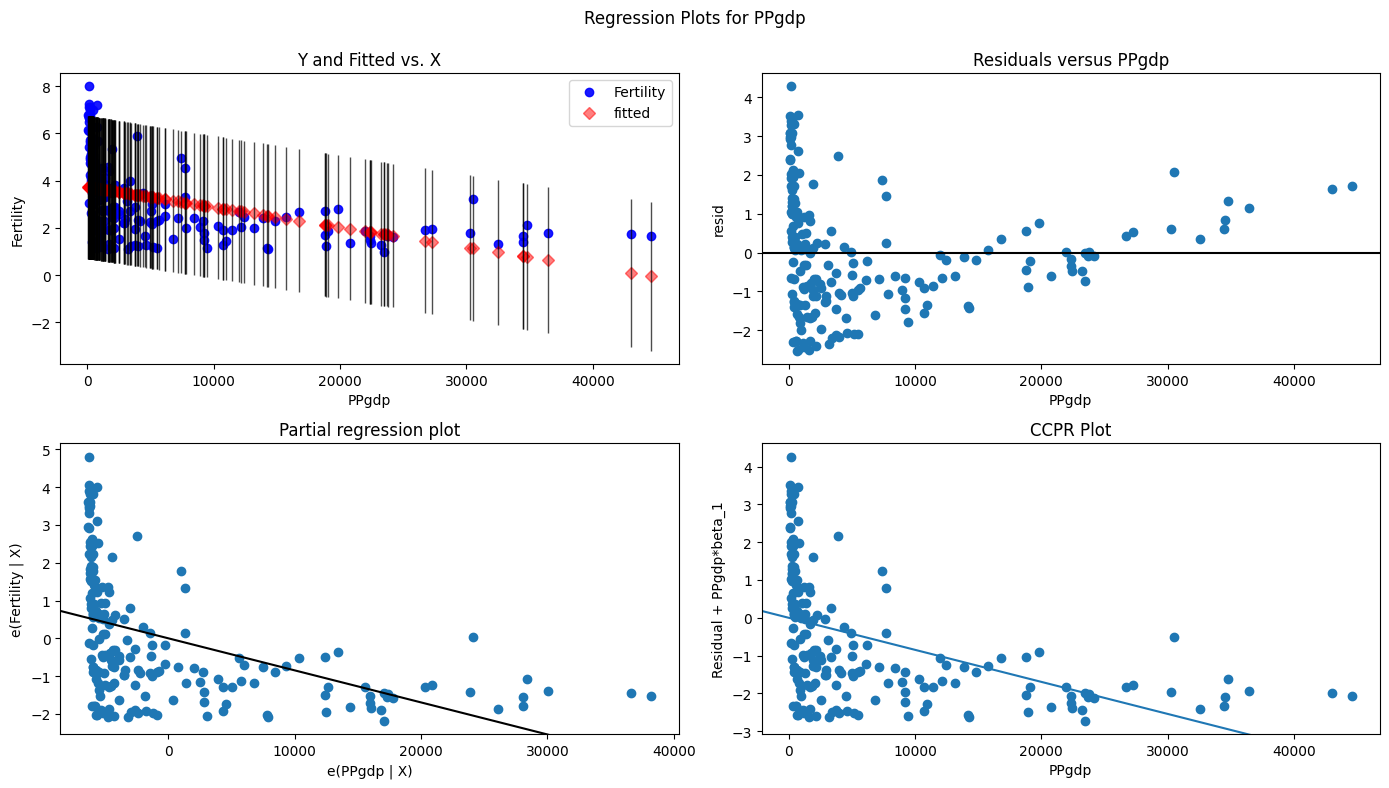
# Scale-Location Plot  
resid\_standardized = lm.get\_influence().resid\_studentized\_internal  
  
sns.regplot(x=fitted, y=np.sqrt(np.abs(resid\_standardized)), color='darkorange',  
 ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-Location', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel(r'$\sqrt{|Standardized Residuals|}$', fontweight='bold')  
plt.show()



# Generate the Breusch-Pagan test for heteroscedasticity.  
bp\_test = het\_breuschpagan(lm.resid, lm.model.exog)  
print("lm:", f"{bp\_test[0]:0,.3f}", "lm\_pvalue:", f"{bp\_test[1]:0,.3f}")

lm: 14.641 lm\_pvalue: 0.000

# Display model regression plots for the 'PPgdp' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'PPgdp',fig=fig)  
plt.show()



# Generate required linear regression model for 'LogFertility' outcome variable and 'LogPPgdp' explanatory variable.  
# Fit linear regression model.  
lm = smf.ols("LogFertility ~ LogPPgdp", data = Fert\_df).fit()  
# View model coefficients.  
print(lm.params)

Intercept 2.703218  
LogPPgdp -0.221160  
dtype: float64

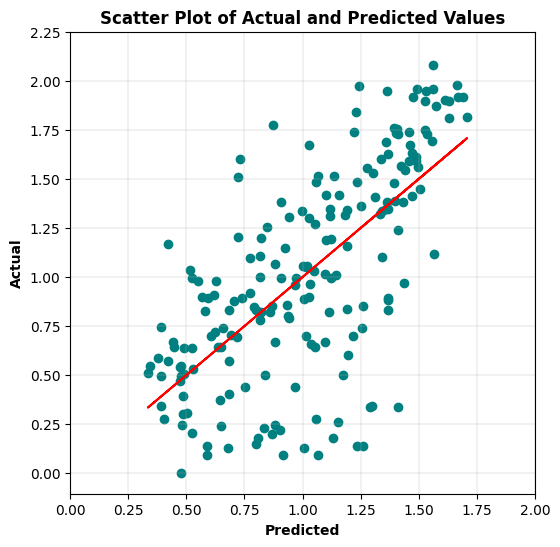
# Calculate the Total Sum of Squares (SST) for an empty model for 'LogFertility'.  
Y = Fert\_df['LogFertility'].tolist()  
Y\_mean = np.mean(Y)  
sst = np.sum((Y - Y\_mean) \*\* 2)  
print(f"SST = {sst:0,.3f}")

SST = 55.432

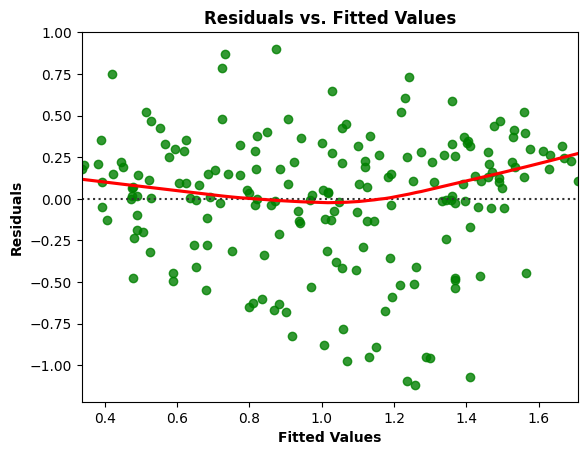
# Generate OLS Regression results for the linear regression model.  
print(lm.summary())  
print('\n', '='\*20, '\n')  
# Residual Sum of Squares  
print(f"RSS = {lm.ssr:0,.3f}")  
# Sum of Squares Error  
print(f"SSE = {sst-lm.ssr:0,.3f}", '\n')  
# Generate the critical F-statistic.  
F\_critical = scipy.stats.f.ppf(1-alpha,1,2)  
print(f"F\_critical: {F\_critical:.3f}")  
# Generate the absolute critical t-statistic.  
t\_critical = abs(scipy.stats.t.ppf(q=alpha/2,df=1))  
print(f"Absolute t\_critical: {t\_critical:.3f}")  
# Display expected Alpha standard.  
print(f"Alpha Standard: {alpha:.2f}")  
# Display Predicted vs. Actual values scatter plot.  
X = Fert\_df[['LogPPgdp']]  
predicted = lm.predict(X)  
# Generate a scatter plot for fitted and actual values.  
plt.figure(figsize=(6,6))  
plt.scatter(predicted, Y, c='teal')  
m, b = np.polyfit(predicted, Y, deg=1)  
plt.plot(predicted, m\*predicted+b, color='red')  
plt.title('Scatter Plot of Actual and Predicted Values', fontweight='bold')  
plt.xlabel('Predicted', fontweight='bold')  
plt.ylabel('Actual', fontweight='bold')  
plt.xticks(np.arange(0,2.25,step=0.25))   
plt.yticks(np.arange(0,2.5,step=0.25))   
plt.grid(linewidth=0.25)  
plt.show()

OLS Regression Results   
=============================================================================  
Dep. Variable: LogFertility R-squared: 0.459  
Model: OLS Adj. R-squared: 0.456  
Method: Least Squares F-statistic: 162.1  
Date: Sun, 23 Feb 2025 Prob (F-statistic): 2.73e-27  
Time: 12:46:35 Log-Likelihood: -94.158  
No. Observations: 193 AIC: 192.3  
Df Residuals: 191 BIC: 198.8  
Df Model: 1   
Covariance Type: nonrobust   
=============================================================================  
 coef std err t P>|t| [0.025 0.975]  
-----------------------------------------------------------------------------  
Intercept 2.7032 0.135 19.968 0.000 2.436 2.970  
LogPPgdp -0.2212 0.017 -12.734 0.000 -0.255 -0.187  
=============================================================================

Omnibus: 15.343 Durbin-Watson: 1.976  
Prob(Omnibus): 0.000 Jarque-Bera (JB): 16.598  
Skew: -0.693 Prob(JB): 0.000249  
Kurtosis: 3.379 Cond. No. 37.6  
=============================================================================  
  
Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
  
 ====================   
  
RSS = 29.980  
SSE = 25.451   
  
F\_critical: 18.513  
Absolute t\_critical: 12.706  
Alpha Standard: 0.05



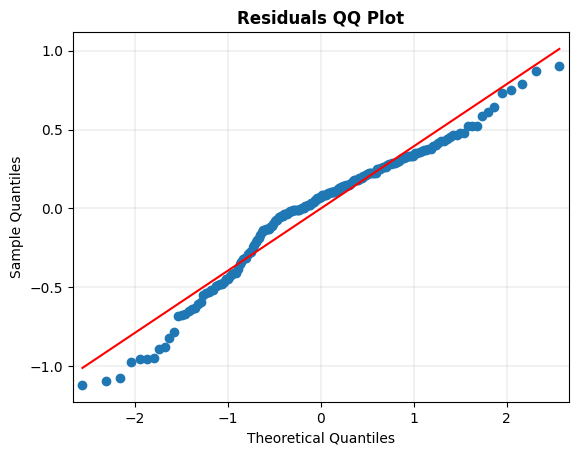
# Display Residuals vs. Fitted values scatter plot.  
fitted = lm.fittedvalues  
residuals = lm.resid  
sns.residplot(x=fitted, y=residuals, lowess=True, line\_kws={'color': 'red'}, color='green')   
plt.title('Residuals vs. Fitted Values', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel('Residuals', fontweight='bold')  
plt.show()



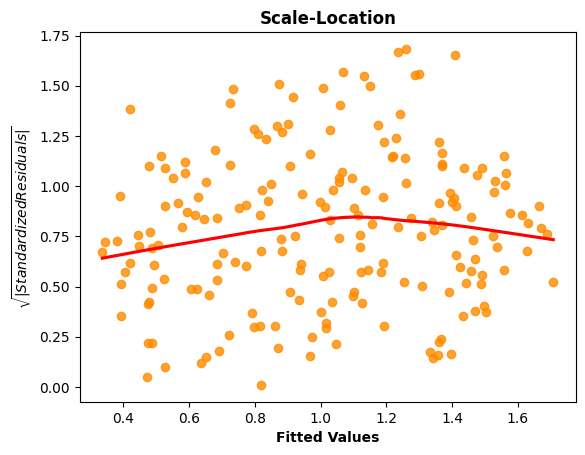
# Residuals Anderson-Darling Normality Test  
result = anderson(residuals)  
print('Statistic: %.3f' % result.statistic)  
p = 0  
for i in range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[i], result.critical\_values[i]  
 if result.statistic < result.critical\_values[i]:  
 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))  
 else:  
 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

Statistic: 3.206  
15.000: 0.565, data does not look normal (reject H0)  
10.000: 0.643, data does not look normal (reject H0)  
5.000: 0.772, data does not look normal (reject H0)  
2.500: 0.900, data does not look normal (reject H0)  
1.000: 1.071, data does not look normal (reject H0)

# Residuals QQ Plot  
qqplot(residuals, line='s')  
plt.title('Residuals QQ Plot', fontweight='bold')  
plt.grid(linewidth=0.25)  
plt.show()



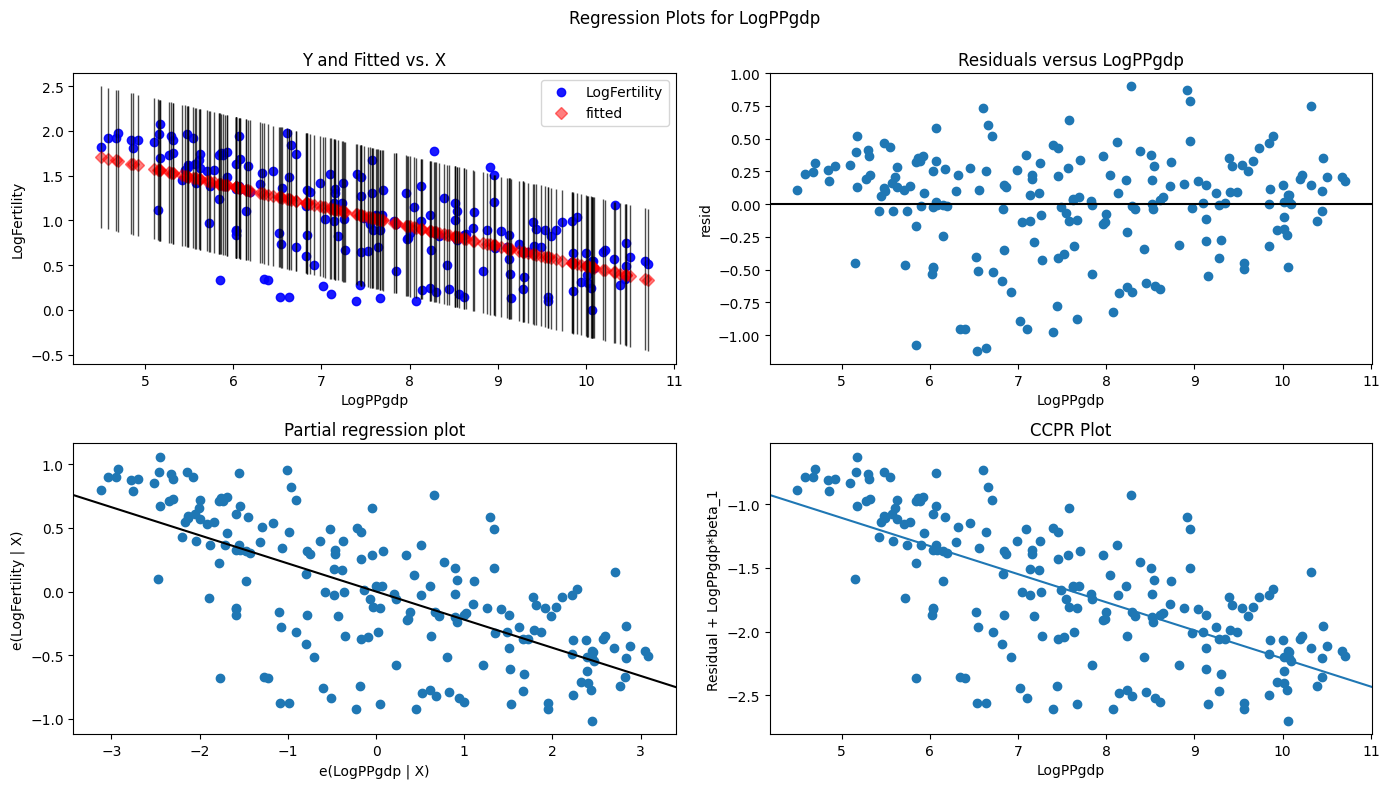
# Scale-Location Plot  
resid\_standardized = lm.get\_influence().resid\_studentized\_internal  
  
sns.regplot(x=fitted, y=np.sqrt(np.abs(resid\_standardized)), color='darkorange',  
 ci=None, lowess=True, line\_kws={'color': 'red'})  
plt.title('Scale-Location', fontweight='bold')  
plt.xlabel('Fitted Values', fontweight='bold')  
plt.ylabel(r'$\sqrt{|Standardized Residuals|}$', fontweight='bold')  
plt.show()



# Generate the Breusch-Pagan test for heteroscedasticity.  
bp\_test = het\_breuschpagan(lm.resid, lm.model.exog)  
print("lm:", f"{bp\_test[0]:0,.3f}", "lm\_pvalue:", f"{bp\_test[1]:0,.3f}")

lm: 1.211 lm\_pvalue: 0.271

# Display model regression plots for the 'LogPPgdp' variable.  
fig = plt.figure(figsize=(14,8))  
fig = sm.graphics.plot\_regress\_exog(lm,'LogPPgdp',fig=fig)  
plt.show()



6b. Which regression exhibits a better fit (log or linear)? Use the ANOVA summary to support your reasoning.

The logarithmic data yielded a significantly better regression model result. The log data regression model had the better R-square result of 0.459, while the linear data regression model had a 0.215 R-square result. The log data regression model had the better F-statistic of 162.1, while the linear data regression model had a 52.22 F-statistic. Both models had a F-statistic probability near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating that the model relationship between the outcome and explanatory variable is statistically significant.  
Only the log data regression model had a t-statistic value that exceeded the absolute t-critical, but both models had p-values near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating that their relationships with the outcome variable (LogFertility or Fertility) were statistically significant.

It did not make sense to add a SST comparison. When plotting the residuals and the fitted value, the log data regression model showed some linearity, while the linear data regression model showed no linearity. Concerning normality of the residuals, the linear data regression model had the better Jarque-Bera test statistic of 10.371, while the log data regression model had a 16.598 Jarque-Bera test statistic. Both models had a Jarque-Bera probability near zero, safely under the 0.05 alpha standard (rejecting H0 the null hypothesis), indicating a lack of normality. The linear data regression model had an Anderson-Darling result of 1.714, while the log data regression model had a 3.206 Anderson-Darling result, somewhat supporting that conclusion. Although, the QQ plot renderings showed potentially acceptable normality for both models. The Scale-Location chart for the log data regression model showed a reasonably horizontal progression that indicates potential homoscedasticity in the model, while the linear data regression model clearly showed heteroscedasticity. The log data regression model had a Breusch-Pagan test statistic of 1.211 and p-value of 0.271, greater than the 0.05 alpha standard (accepting H0 the null hypothesis), indicating that some level of homoscedasticity is present in the model.