Exercise1

Using StatsModels

Let's run the same regression using SciPy and StatsModels, and confirm we get the same results.

Instructions

- Compute the regression of '_VEGESU1' as a function of 'INCOME2' using SciPy's linregress().
- Compute the regression of '_VEGESU1' as a function of 'INCOME2' using StatsModels' smf.ols().

In [ ]:

```
from scipy.stats import linregress
import statsmodels.formula.api as smf

# Run regression with linregress
subset = brfss.dropna(subset=['INCOME2', '_VEGESU1'])
xs = subset['INCOME2']
ys = subset['_VEGESU1']
res = ____
print(res)

# Run regression with StatsModels
results = smf.ols('____', ____ = ____).fit()
print(results.params)

#_____________________#
#Solutions

from scipy.stats import linregress
import statsmodels.formula.api as smf

# Run regression with linregress
subset = brfss.dropna(subset=['INCOME2', '_VEGESU1'])
xs = subset['INCOME2']
ys = subset['_VEGESU1']
res = linregress(xs, ys)
print(res)

# Run regression with StatsModels
results = smf.ols('_VEGESU1 ~ INCOME2', data=brfss).fit()
print(results.params)
```

Exercise2

Plot income and education

To get a closer look at the relationship between income and education, let's use the variable 'educ' to group the data, then plot mean income in each group.

Here, the GSS dataset has been pre-loaded into a DataFrame called gss.

Instructions

- Group gss by 'educ'. Store the result in grouped.
- From grouped, extract 'realinc' and compute the mean.
- Plot mean_income_by_educ as a scatter plot. Specify 'o' and alpha=0.5.

In [ ]:

```
# Group by educ
grouped = ____

# Compute mean income in each group
mean_income_by_educ = ____

# Plot mean income as a scatter plot
plt.plot(____)

# Label the axes
plt.xlabel('Education (years)')
plt.ylabel('Income (1986 $)')
plt.show()

#____________________#
#Solutions

# Group by educ
grouped = gss.groupby('educ')

# Compute mean income in each group
mean_income_by_educ = grouped['realinc'].mean()

# Plot mean income as a scatter plot
plt.clf()
plt.plot(mean_income_by_educ, 'o', alpha=0.5)

# Label the axes
plt.xlabel('Education (years)')
plt.ylabel('Income (1986 $)')
plt.show()
```

Exercise3

Non-linear model of education

The graph in the previous exercise suggests that the relationship between income and education is non-linear. So let's try fitting a non-linear model.

Instructions

- Add a column named 'educ2' to the gss DataFrame; it should contain the values from 'educ' squared.
- Run a regression model that uses 'educ', 'educ2', 'age', and 'age2' to predict 'realinc'.

In [ ]:

```
import statsmodels.formula.api as smf

# Add a new column with educ squared
gss['educ2'] = ____

# Run a regression model with educ, educ2, age, and age2
results = ____

# Print the estimated parameters
print(results.params)

#_____________________#
#Solutions

import statsmodels.formula.api as smf

# Add a new column with educ squared
gss['educ2'] = gss['educ']**2

# Run a regression model with educ, educ2, age, and age2
results = smf.ols('realinc ~ educ + educ2 + age + age2', data=gss).fit()

# Print the estimated parameters
print(results.params)
```

Exercise4

Making predictions

At this point, we have a model that predicts income using age, education, and sex.

Let's see what it predicts for different levels of education, holding age constant.

Instructions

- Using np.linspace(), add a variable named 'educ' to df with a range of values from 0 to 20.
- Add a variable named 'age' with the constant value 30.
- Use df to generate predicted income as a function of education.

In [ ]:

```
# Run a regression model with educ, educ2, age, and age2
results = smf.ols('realinc ~ educ + educ2 + age + age2', data=gss).fit()

# Make the DataFrame
df = pd.DataFrame()
df['educ'] = ____
df['age'] = ____
df['educ2'] = df['educ']**2
df['age2'] = df['age']**2

# Generate and plot the predictions
pred = ____
print(pred.head())

#_____________________#
#Solutions

# Run a regression model with educ, educ2, age, and age2
results = smf.ols('realinc ~ educ + educ2 + age + age2', data=gss).fit()

# Make the DataFrame
df = pd.DataFrame()
df['educ'] = np.linspace(0, 20)
df['age'] = 30
df['educ2'] = df['educ']**2
df['age2'] = df['age']**2

# Generate and plot the predictions
pred = results.predict(df)
print(pred.head())
```

Exercise5

Visualizing predictions

Now let's visualize the results from the previous exercise!

Instructions

- Plot mean_income_by_educ using circles ('o'). Specify an alpha of 0.5.
- Plot the prediction results with a line, with df['educ'] on the x-axis and pred on the y-axis.

In [ ]:

```
# Plot mean income in each age group
plt.clf()
grouped = gss.groupby('educ')
mean_income_by_educ = grouped['realinc'].mean()
____

# Plot the predictions
pred = results.predict(df)
plt.plot(____, ____, label='Age 30')

# Label axes
plt.xlabel('Education (years)')
plt.ylabel('Income (1986 $)')
plt.legend()
plt.show()

#_____________________#
#Solutions

# Plot mean income in each age group
plt.clf()
grouped = gss.groupby('educ')
mean_income_by_educ = grouped['realinc'].mean()
plt.plot(mean_income_by_educ, 'o', alpha=0.5)

# Plot the predictions
pred = results.predict(df)
plt.plot(df['educ'], pred, label='Age 30')

# Label axes
plt.xlabel('Education (years)')
plt.ylabel('Income (1986 $)')
plt.legend()
plt.show()
```

Exercise6

Predicting a binary variable

Let's use logistic regression to predict a binary variable. Specifically, we'll use age, sex, and education level to predict support for legalizing cannabis (marijuana) in the U.S.

In the GSS dataset, the variable grass records the answer to the question "Do you think the use of marijuana should be made legal or not?"

Instructions

- Fill in the parameters of smf.logit() to predict grass using the variables age, age2, educ, and educ2, along with sex as a categorical
- Add a column called educ and set it to 12 years; then compute a second column, educ2, which is the square of educ.
- Generate separate predictions for men and women.

- Fill in the missing code to compute the mean of 'grass' for each age group, and then the arguments of plt.plot() to plot pred2 versus df['age'] with the label 'Female'.

In [ ]:

```
# Recode grass
gss['grass'].replace(2, 0, inplace=True)

# Run logistic regression
results = smf.logit(____).fit()
results.params

# Make a DataFrame with a range of ages
df = pd.DataFrame()
df['age'] = np.linspace(18, 89)
df['age2'] = df['age']**2

# Set the education level to 12
df['educ'] = ____
df['educ2'] = ____

# Generate predictions for men and women
df['sex'] = 1
pred1 = results.____(____)

df['sex'] = 2
pred2 = results.____(____)

plt.clf()
grouped = gss.groupby('age')
favor_by_age = ____
plt.plot(favor_by_age, 'o', alpha=0.5)

plt.plot(df['age'], pred1, label='Male')
plt.plot(____)

plt.xlabel('Age')
plt.ylabel('Probability of favoring legalization')
plt.legend()
plt.show()

#_____#
#Solutions

# Recode grass
gss['grass'].replace(2, 0, inplace=True)

# Run logistic regression
results = smf.logit('grass ~ age + age2 + educ + educ2 + C(sex)', data=gss).fit()
results.params

# Make a DataFrame with a range of ages
df = pd.DataFrame()
df['age'] = np.linspace(18, 89)
df['age2'] = df['age']**2

# Set the education level to 12
df['educ'] = 12
df['educ2'] = df['educ']**2

# Generate predictions for men and women
df['sex'] = 1
pred1 = results.predict(df)
```

```
df['sex'] = 2
pred2 = results.predict(df)

plt.clf()
grouped = gss.groupby('age')
favor_by_age = grouped['grass'].mean()
plt.plot(favor_by_age, 'o', alpha=0.5)

plt.plot(df['age'], pred1, label='Male')
plt.plot(df['age'], pred2, label='Female')

plt.xlabel('Age')
plt.ylabel('Probability of favoring legalization')
plt.legend()
plt.show()
```