Multiple Regression

Today's Outline

- Multiple Regression
- Omitted variable bias
- Higher order terms
- ANOVA, ANCOVA, MANOVA
- Interactions

Note: Assignment 3 will be posted this Weekend

Multiple Regression in Python

- Consider the following population regression model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + u$$

- The tilde "~" separates the dependent variable from the regressors
- The regressors are separated by "+"
- A constant (intercept) is added by default

```
reg = smf.ols(formula='y ~ x1 + x2 + x3', data=sample)
results = reg.fit()
```

Multiple Regression in Practice

- Consider the following population regression model:

$$\log(salary) = \beta_0 + \beta_1 \log(sales) + \beta_2 roe + \beta_3 consprod + u$$

```
reg = smf.ols('np.log(salary) ~ np.log(sales) + roe + consprod', ceo)
results = reg.fit()
results.summary()
```

OLS Regression Results

Dep. Variable:	np.log(salary)	**R-squared:**	0.299
Model:	OLS	**Adj. R-squared:**	0.289
Method:	Least Squares	**F-statistic:**	29.18
Date:	Thu, 20 Oct 2022	**Prob (F-statistic):**	9.38e-16
Time:	21:18:52	**Log-Likelihood:**	-140.08
No. Observations:	209	**AIC:**	288.2
Df Residuals:	205	**BIC:**	301.5
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.3994	0.292	15.092	0.000	3.825	4.974
np.log(sales)	0.2726	0.033	8.271	0.000	0.208	0.338
roe	0.0139	0.004	3.243	0.001	0.005	0.022
consprod	0.1799	0.080	2.247	0.026	0.022	0.338

How can we interpret roe in the multiple regression?

Manual Multiple Regression

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}.$$

- Multiple regression is relatively straightforward to implement with basic linear algebra operations
- These linear algebra operations may be useful in other contexts as well
- Regressors are stores in an n x (k+1) matrix that has a column for each regressor and a constant

```python
# make a subset of the predictors
X = ceo[['lsales', 'roe', 'consprod']].copy()
X["intercept"] = 1

# pull the dependent variable
Y = ceo[["lsalary"]].copy()

# determine sample size and number of regressors
n = ceo.shape[0]
k = 3
```

```python
# calculate the parameters
b_hat = np.linalg.inv(X.T@X)@X.T@y
b.index = x.columns
b
```

	lsalary
lsales	0.272554
roe	0.013923
consprod	0.179890
intercept	4.399396

Manual Multiple Regression (Continued)

$$\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}\hat{\beta}$$

```
# get the vector of residuals
u_hat = y - X@b
u_hat.columns = ["residuals"]
u_hat
```

	residuals
0	-0.384172
1	-0.151570

$$\hat{\sigma}^2 = \frac{1}{n-k-1}\hat{\mathbf{u}}'\hat{\mathbf{u}}$$

```
error_var = (u_hat.T@u_hat)/(n-k-1)
error_var
```

	residuals
residuals	0.228075

$$\widehat{\mathrm{Var}(\hat{\beta})} = \hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1}$$

```
se_reg = error_var.values * np.linalg.inv(X.T@X)
se_reg
```

```
array([[ 1.08577283e-03,  1.77888993e-05, -9.02465726e-05,
        -9.28329591e-03],
       [ 1.77888993e-05,  1.84291873e-05, -1.40663816e-04,
        -4.23819340e-04],
       [-9.02465726e-05, -1.40663816e-04,  6.40753719e-03,
```

```
# the standard errors of the regression parameters
# can be found along the diagonal of the Variance- Covariance matrix
se = np.sqrt(np.diagonal(se_reg))
se
```

```
5]: array([0.03295107, 0.00429292, 0.08004709, 0.29150151])
```

Calculating Marginal Effects

- We will often want to visualize the marginal effect of a predictor on our dependent variable
- This requires that we fix the values of the other regressors
- In a simple model, changing where the covariates are fixed will change the intercept
- The slope remains the same

```
results.params
```

```
Intercept        4.399396
np.log(sales)    0.272554
roe              0.013923
consprod         0.179890
dtype: float64
```

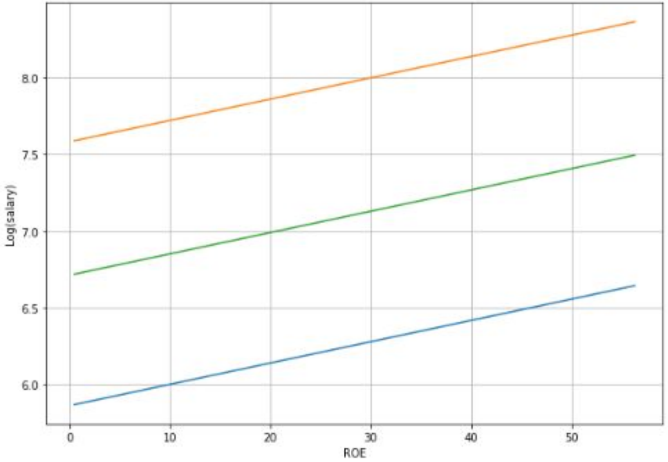
```
xroe = np.linspace(ceo.roe.min(),ceo.roe.max(), 50)

# sales fixed values
minsales = np.log(ceo.sales).min()
maxsales = np.log(ceo.sales).max()
meansales = np.log(ceo.sales).mean()

# we fix consprod at the mean
meancons = np.mean(ceo.consprod)
```

```
# form the fitted values at each different fixed value
y1 = results.params[0] + results.params[1]*minsales +results.params[2]*xroe +results.params[3]*meancons
y2 = results.params[0] + results.params[1]*maxsales +results.params[2]*xroe +results.params[3]*meancons
y3 =  results.params[0] + results.params[1]*meansales +results.params[2]*xroe +results.params[3]*meancons
```

Effects Plots

- The R implementations of effects plots will automatically use the average across the covariates
- Confidence intervals are also included
- It will also automatically calculate the marginal effects with interactions and higher order terms
- There is not an analogous implementation in python
- However we can implement a workaround

```python
plt.figure(figsize = (10, 7))

# plot x against each y predicted at different fixed levels of log(sales)
plt.plot(xroe, y1)
plt.plot(xroe, y2)
plt.plot(xroe, y3)

plt.xlabel("ROE")
plt.ylabel("Log(salary)")

plt.grid()
```



Quadratics and Polynomials

- Quadratic or higher power terms can make our models much more flexible
- Can allow us to model decreasing, increasing, s-shaped relationships, etc.
- Many economic variables have these types of relationships between variables
- Statsmodels lets us add a polynomial term using the following syntax:

  I(x**p) where x is the variable name and p is the power you want it raised to

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + u$$

‘y ~ x + I(x2) + I(X**3)’**

Quadratic Statsmodels

- On the right I regress the sales on a quadratic
- We are using the ‘andy’ dataset from 430
- What do you notice about the coefficients?
- What does this tell you about the relationship between advertising and sales?

```
# include all individual and interaction effects
mreg = smf.ols('sales ~ price + advert + I(advert**2)',data = andy).fit()
mreg.summary()
```

OLS Regression Results

Dep. Variable:	sales	**R-squared:**	0.508
Model:	OLS	**Adj. R-squared:**	0.487
Method:	Least Squares	**F-statistic:**	24.46
Date:	Thu, 26 Oct 2023	**Prob (F-statistic):**	5.60e-11
Time:	14:06:10	**Log-Likelihood:**	-219.55
No. Observations:	75	**AIC:**	447.1
Df Residuals:	71	**BIC:**	456.4
Df Model:	3		
Covariance Type:	nonrobust		

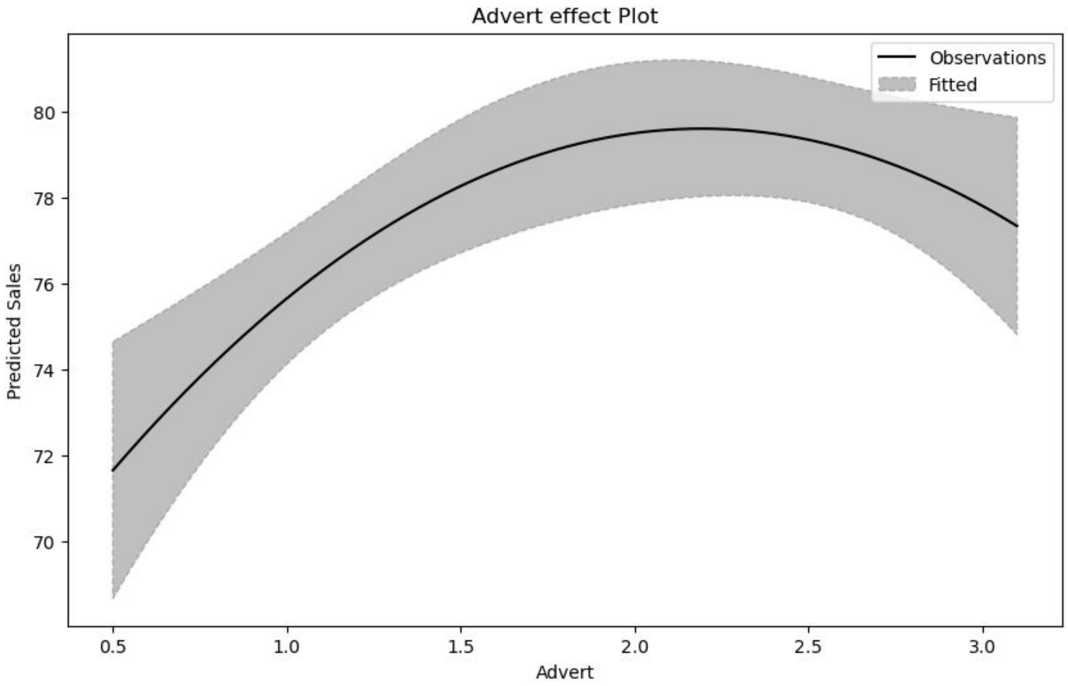
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	109.7190	6.799	16.137	0.000	96.162	123.276
price	-7.6400	1.046	-7.304	0.000	-9.726	-5.554
advert	12.1512	3.556	3.417	0.001	5.060	19.242
I(advert ** 2)	-2.7680	0.941	-2.943	0.004	-4.644	-0.892

Omnibus:	1.004	**Durbin-Watson:**	2.043
Prob(Omnibus):	0.605	**Jarque-Bera (JB):**	0.455
Skew:	-0.088	**Prob(JB):**	0.797
Kurtosis:	3.339	**Cond. No.**	101.

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

Quadratic Effects Plot

- An effects plot may also be drawn for a polynomial
- Fix the terms that are not included in the polynomial at their mean
- Make a prediction for y over a range of possible values for the desired variable (In this case sales)

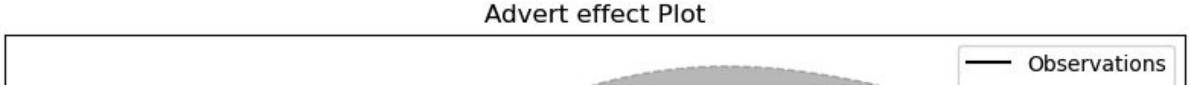


Plotting Effects Workaround

```python
# Generate predictions over range
xrange = np.linspace(andy.advert.min(), andy.advert.max(), 2000).reshape(2000,1)
new_data = pd.DataFrame(xrange, columns = ['advert'])
new_data['price'] = andy.price.mean()

predictions = mreg.get_prediction(new_data)

# Generate table with intervals for each x
predictions = predictions.summary_frame(alpha=0.05)
```

```python
plt.figure(figsize = (10, 6))

plt.plot(new_data['advert'], predictions["mean"], color = "black")

plt.title('Advert effect Plot')
plt.xlabel("Advert")
plt.ylabel("Predicted Sales")

# confidence Intervals
plt.fill_between(new_data['advert'], predictions["mean_ci_lower"], predictions["mean_ci_upper"],
                 color = "grey", linestyle = '--', alpha = .5)

# Fun fact - the legend is labelled in the order you draw each plot element!
plt.legend(["Observations", "Fitted", "Lower CI", "Upper CI","Lower PI", "Upper PI"])
```

```
<matplotlib.legend.Legend at 0x7fa7de139910>
```



Exercise 1

- Use the wage1 data from the wooldridge package
- Specify a model with wage as the dependent variable and two predictors. Use your economic intuition to choose one predictor that should be a quadratic such as:

$$wage = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + u$$

- Visualize the effects of x1 in the regression above over a range of possible values

Omitted Variable Bias

- Omitted variable bias occurs when we leave relevant variables out of our model
- This will make our estimated parameters inaccurate
- For example regressing wage on education may lead us to overestimate the positive effects of education
  - Leaves out measures of inherent ability (like IQ)
  - The indirect IQ effect will be misattributed to education, inflating the coefficient
- Hence the effect of education on wage is the sum of its direct effect and the indirect effect of ability
- Consider the regression on the right

```python
reg = smf.ols('np.log(salary) ~ np.log(sales) + roe', ceo)
results = reg.fit()
results.summary()
```

OLS Regression Results

Dep. Variable:	np.log(salary)	**R-squared:**	0.282
Model:	OLS	**Adj. R-squared:**	0.275
Method:	Least Squares	**F-statistic:**	40.45
Date:	Thu, 20 Oct 2022	**Prob (F-statistic):**	1.52e-15
Time:	21:18:52	**Log-Likelihood:**	-142.62
No. Observations:	209	**AIC:**	291.2
Df Residuals:	206	**BIC:**	301.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	4.3622	0.294	14.843	0.000	3.783	4.942
np.log(sales)	0.2751	0.033	8.272	0.000	0.210	0.341
roe	0.0179	0.004	4.519	0.000	0.010	0.026

Omnibus:	97.850	**Durbin-Watson:**	2.033
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	564.485
Skew:	1.727	**Prob(JB):**	2.65e-123
Kurtosis:	10.273	**Cond. No.**	183.

Omitted Variable Bias

- Leaving out the covariates, we can see how the value changes

```
# the coefficient on log(sales) is smaller now
smf.ols('np.log(salary) ~ np.log(sales)', ceo).fit().params
```

```
6]: Intercept         4.821996
    np.log(sales)     0.256672
    dtype: float64
```

- We can calculate Beta1 by considering how y changes in response to the direct and indirect effects of x:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$$

$$\tilde{y} = \tilde{\beta}_0 + \tilde{\beta}_1 x_1$$

$$\tilde{\beta}_1 = \hat{\beta}_1 + \hat{\beta}_2 \tilde{\delta}$$

$$x_2 = \tilde{\delta}_0 + \tilde{\delta}_1 x_1$$

Predicted y increases by $\hat{\beta}_1$ units. This is also associated with a predicted increase in x_2 of $\tilde{\delta}_1$ units. Each unit increase of x_2 leads to a corresponding increase in y of $\hat{\beta}_2$. Therefore $\tilde{\beta}_1$ is the sum of the direct and indirect effects of sales on log(salary).

Omitted Variable Bias (In Python)

```
# the coefficient on log(sales) is smaller now
smf.ols('np.log(salary) ~ np.log(sales)', ceo).fit().params
```

```
6]: Intercept        4.821996
    np.log(sales)    0.256672
    dtype: float64
```

```
]:
# pull the hats from the multiple regression
beta1_hat = results.params[1]
beta2_hat = results.params[2]

# Calculate how roe responds to changes in log(sales)
delta1_tilde = smf.ols('roe ~ + np.log(sales)', ceo).fit().params[1]
```

```
]:
# calculate the association between log(sales) and salary if we leave out ROE
# This icnludes the direct and indirect effects on teh left and right
test_beta1_tilde = beta1_hat + beta2_hat*delta1_tilde
```

```
]:
test_beta1_tilde
```

```
ıt[8]: 0.25667169166414966
```

ANOVA (Building Groups)

- Using statsmodels we can build a model with no predictors
- We want to understand the differences in the dependent variable between groups
- The one-way ANOVA allows us to perform hypothesis tests to see if the differences are statistically significant
- On the right we construct an occupation variable by combining the various CEO industries

```python
# there are four different business types in the dataset
ceo.columns
```

```
: Index(['salary', 'pcsalary', 'sales', 'roe', 'pcroe', 'ros', 'indus',
         'finance', 'consprod', 'utility', 'lsalary', 'lsales', 'occupation'],
        dtype='object')
```

```python
# I terate through teh data and combine these conditionally
# into one column
occupation = []
for i in range(ceo.shape[0]):
    if ceo.indus[i] == 1:
        occupation.append("indus")
    elif ceo.finance[i] == 1:
        occupation.append("finance")
    elif ceo.consprod[i] == 1:
        occupation.append("consprod")
    else:
        occupation.append("utility")
```

```python
# this column is added to the original dataset
ceo["occupation"] = occupation
```

ANOVA

- The data type of our new column is now “object” and not “numeric”
- If we regress on an “object” type column, statsmodels splits up the groups automatically into their unique values
- Statsmodels automatically chooses one group to omit
- This is the reference (or control) group against which the others are compared

```python
# this column is added to the original dataset
ceo["occupation"] = occupation
```

```python
ceo.occupation.dtype
```

```
dtype('O')
```

```python
ceo.occupation.unique()
```

```
array(['indus', 'finance', 'utility', 'consprod'], dtype=object)
```

```python
anova1 = smf.ols('np.log(salary) ~ occupation',data = ceo).fit()
anova1.summary()
```

OLS Regression Results

Dep. Variable:	np.log(salary)	**R-squared:**	0.147
Model:	OLS	**Adj. R-squared:**	0.134
Method:	Least Squares	**F-statistic:**	11.76
Date:	Fri, 21 Oct 2022	**Prob (F-statistic):**	3.87e-07
Time:	11:40:24	**Log-Likelihood:**	-160.65
No. Observations:	209	**AIC:**	329.3
Df Residuals:	205	**BIC:**	342.7
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	7.1465	0.068	105.047	0.000	7.012	7.281
occupation[T.finance]	-0.0889	0.103	-0.861	0.390	-0.292	0.115
occupation[T.indus]	-0.2094	0.094	-2.236	0.026	-0.394	-0.025
occupation[T.utility]	-0.6354	0.111	-5.719	0.000	-0.854	-0.416

ANOVA (Change Reference Group)

- We will often have a control group
- This is the baseline that we usually want to compare other groups against
- Statsmodels allows us to change the reference groups by including the following syntax in our formula:

C(column, Treatment(reference = “control”)

```
anova1 = smf.ols('np.log(salary) ~ C(occupation, Treatment(reference="indus"))',
                 data = ceo).fit()
anova1.summary()
```

OLS Regression Results

Dep. Variable:	np.log(salary)	R-squared:	0.147
Model:	OLS	Adj. R-squared:	0.134
Method:	Least Squares	F-statistic:	11.76
Date:	Thu, 20 Oct 2022	Prob (F-statistic):	3.87e-07
Time:	23:01:49	Log-Likelihood:	-160.65
No. Observations:	209	AIC:	329.3
Df Residuals:	205	BIC:	342.7
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.9371	0.064	107.753	0.000	6.810	7.064
C(occupation, Treatment(reference="indus"))[T.consprod]	0.2094	0.094	2.236	0.026	0.025	0.394
C(occupation, Treatment(reference="indus"))[T.finance]	0.1205	0.101	1.195	0.234	-0.078	0.319
C(occupation, Treatment(reference="indus"))[T.utility]	-0.4259	0.109	-3.911	0.000	-0.641	-0.211

Omnibus:	43.446	Durbin-Watson:	2.126
Prob(Omnibus):	0.000	Jarque-Bera (JB):	152.360
Skew:	0.785	Prob(JB):	8.23e-34
Kurtosis:	6.877	Cond. No.	4.37

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

ANOVA Table

- An anova is essentially just a test that **none** of the coefficients for any of the categorical variable are statistically significant
- We do this by performing a ∫joint hypothesis or **f-test** (covered later)
- A type 2 anova table will show the results of such an f-test for each variable in the model

```
import statsmodels.api as sm
sm.stats.anova_lm(anova1, typ = 2)
```

	sum_sq	df	F	PR(>F)
C(occupation, Treatment(reference="indus"))	9.794386	3.0	11.756703	3.868312e-07
Residual	56.927780	205.0	NaN	NaN

$$H_0 : \beta_{consprod} = \beta_{finance} = \beta_{utility} = 0$$

ANCOVA

- We may want to include a factor and quantitative variables in a model
- The additional quantitative variables are added to the formula just like any other covariate
- On the right, I replicate the regression from ECON 430 in python

```
utown = pd.read_csv("utown.csv")
```

```
mreg_mod = smf.ols('price~utown+sqft', data=utown).fit()
mreg_mod.summary()
```

5]:

OLS Regression Results

Dep. Variable:	price	R-squared:	0.865
Model:	OLS	Adj. R-squared:	0.865
Method:	Least Squares	F-statistic:	3192.
Date:	Thu, 20 Oct 2022	Prob (F-statistic):	0.00
Time:	22:53:27	Log-Likelihood:	-4159.7
No. Observations:	1000	AIC:	8325.
Df Residuals:	997	BIC:	8340.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	5.6809	4.290	1.324	0.186	-2.738	14.100
utown	60.3690	0.983	61.432	0.000	58.441	62.297
sqft	8.3557	0.168	49.642	0.000	8.025	8.686

Omnibus:	0.818	Durbin-Watson:	2.011
Prob(Omnibus):	0.664	Jarque-Bera (JB):	0.709
Skew:	-0.056	Prob(JB):	0.702
Kurtosis:	3.068	Cond. No.	222.

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

ANCOVA Table

```
sm.stats.anova_lm(mreg_mod, typ = 2)
```

	sum_sq	df	F	PR(>F)
utown	909292.379689	1.0	3773.898474	0.000000e+00
sqft	593749.764695	1.0	2464.280336	1.037732e-271
Residual	240219.631988	997.0	NaN	NaN

Adjusted Means

- We also may want to look at the adjusted means
- These are the fitted values at various levels of the factor
- Utown takes the values (0,1)
- In this case we can observe how price changes as we move away from 0 towards one

```python
# we fix the other variables in the plot at their mean to generate the effects
# we start with utown = 0
min_town = mreg_mod.params[0]+mreg_mod.params[1]*0 + mreg_mod.params[2]*utown.sqft.mean()
```

```python
# we fix the orger variables in the plot at their mean to generate the effects
# we start with utown = 1
max_town = mreg_mod.params[0]+mreg_mod.params[1]*1 + mreg_mod.params[2]*utown.sqft.mean()
```

```python
# We can look at the adjusted means (fitted values at the various levels of the factor)
values = [0, .2, .5, .8, 1]

[mreg_mod.params[0]+mreg_mod.params[1]*i + mreg_mod.params[2]*utown.sqft.mean() for i in values]
```

```
]: [216.32419472277348,
    228.39800060995142,
    246.50870944071832,
    264.6194182714852,
    276.69322415866316]
```

MANOVA

- We can also specify multifactor regression models
- This enables to examine variations between and within different groups (for example race and gender and how these variables interact)
- We replicate the example from ECON 430, where conformity in the moore study is regressed on:
  - Their partner's social status
  - Fcategory (a measure of self-esteem)

```
1 moore = pd.read_csv('moore.csv', index_col = 0)
```

```
1 moore
```

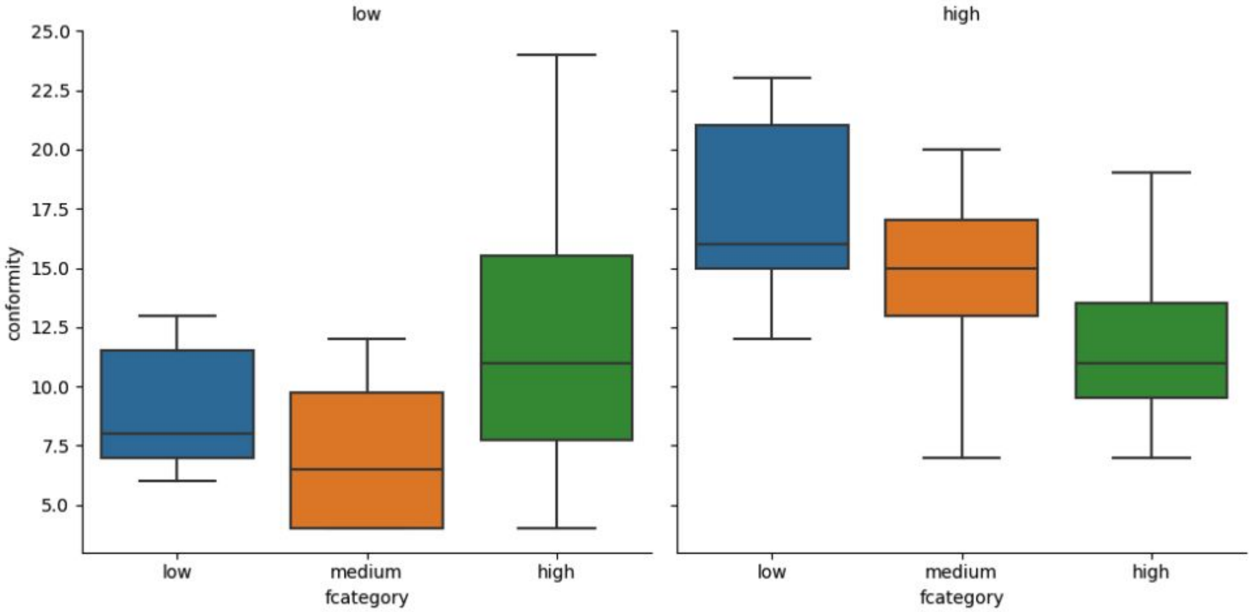
8]:

	partner_status	conformity	fcategory	fscore
NaN	partner.status	conformity	fcategory	fscore
1.0	low	8	low	37
2.0	low	4	high	57
3.0	low	8	high	65
4.0	low	7	low	20
5.0	low	10	low	36
6.0	low	6	low	18
7.0	low	12	medium	51
8.0	low	4	medium	44
9.0	low	13	low	31
10.0	low	12	low	36
11.0	low	4	medium	42
12.0	low	13	high	56

Box Plots For Interactions

- Below we split the boxplots between groups with partners who have high and low social status
- What does the following tell us about the effect of the effect of self esteem on conformity?

```
: a = sns.catplot(data = moore,
                  x = 'fcategory',
                  y = 'conformity',
                  kind = 'box', # type of plot
                  col = 'partner_status',
                  order = ['low', # custom order of boxplots
                           'medium',
                           'high']).set_titles('{col_name}')
```



MANOVA Interactions (statsmodels)

- In order to study the between and within effects of the two factor variables we need to include interactions
- Interactions are specified in statsmodels using the “*” operator
- This tells statsmodels to include all interactions as well as the individual effects of each variable

```
manova1 = smf.ols('conformity ~ C(fcategory, Treatment("low"))*C(partner_status, Treatment("low"))',data = moore).fit()
manova1.summary()
```

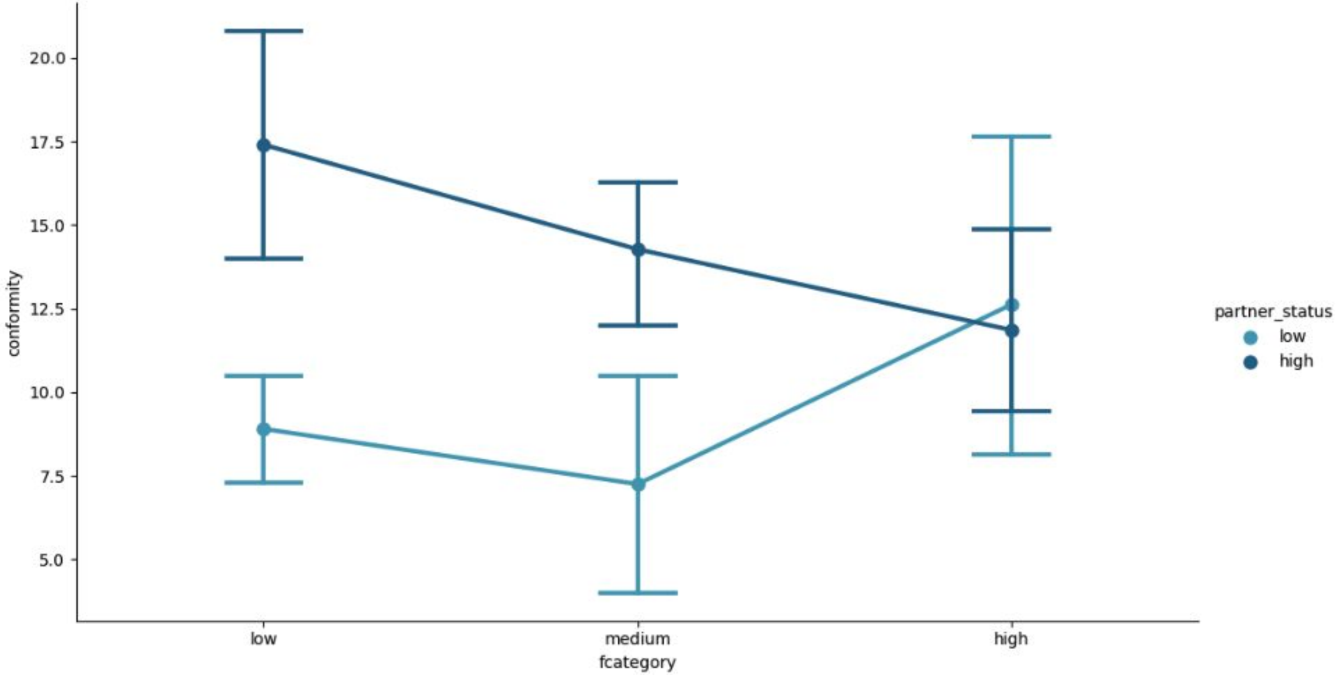
OLS Regression Results

Dep. Variable:	conformity	**R-squared:**	0.324
Model:	OLS	**Adj. R-squared:**	0.237
Method:	Least Squares	**F-statistic:**	3.734
Date:	Thu, 20 Oct 2022	**Prob (F-statistic):**	0.00740
Time:	23:42:49	**Log-Likelihood:**	-129.10
No. Observations:	45	**AIC:**	270.2
Df Residuals:	39	**BIC:**	281.0
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	8.9000	1.448	6.146	0.000	5.971	11.829
C(fcategory, Treatment("low"))[T.high]	3.7250	2.172	1.715	0.094	-0.668	8.118
C(fcategory, Treatment("low"))[T.medium]	-1.6500	2.709	-0.609	0.546	-7.130	3.830
C(partner_status, Treatment("low"))[T.high]	8.5000	2.508	3.389	0.002	3.427	13.573
C(fcategory, Treatment("low"))[T.high]:C(partner_status, Treatment("low"))[T.high]	-9.2679	3.451	-2.686	0.011	-16.247	-2.288
C(fcategory, Treatment("low"))[T.medium]:C(partner_status, Treatment("low"))[T.high]	-1.4773	3.666	-0.403	0.689	-8.892	5.938

Box Plots For Interactions

```
: g = sns.catplot(
    data=moore, x="fcategory", y="conformity", hue="partner_status", order = ['low', 'medium', 'high'],
    capsize=.2, palette="YlGnBu_d",
    kind="point", height=6, aspect=1.75,
)
```



MANOVA (Main Effects)

```
# include only main  effects
manova1 = smf.ols('conformity ~ C(fcategory, Treatment("low"))+C(partner_status, Treatment("low"))',data = moore).fit()
manova1.summary()
```

OLS Regression Results

Dep. Variable:	conformity	**R-squared:**	0.179
Model:	OLS	**Adj. R-squared:**	0.118
Method:	Least Squares	**F-statistic:**	2.971
Date:	Fri, 21 Oct 2022	**Prob (F-statistic):**	0.0428
Time:	12:19:25	**Log-Likelihood:**	-133.47
No. Observations:	45	**AIC:**	274.9
Df Residuals:	41	**BIC:**	282.2
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	10.1978	1.373	7.429	0.000	7.426	12.970
C(fcategory, Treatment("low"))[T.high]	-0.0809	1.809	-0.045	0.965	-3.735	3.573
C(fcategory, Treatment("low"))[T.medium]	-1.1760	1.902	-0.618	0.540	-5.017	2.665
C(partner_status, Treatment("low"))[T.high]	4.6067	1.556	2.960	0.005	1.463	7.750

Omnibus:	8.365	**Durbin-Watson:**	2.756
Prob(Omnibus):	0.015	**Jarque-Bera (JB):**	7.335
Skew:	0.882	**Prob(JB):**	0.0255
Kurtosis:	3.895	**Cond. No.**	4.25

Exercise 2

- Import the mtcars dataset using following code

```
import statsmodels.api as sm
mtcars = sm.datasets.get_rdataset('mtcars').data
```

- Convert the “gear” variable to an object data type:

```
mtcars["gear"] = mtcars.gear.astype('str')
```

- Run a type II anova using a gear = 5 as the reference group
- What is the predicted MPG for a car with 4 gears?
- Is there a statistically significant difference between each group and the control?

A data frame with 32 observations on 11 (numeric) variables.

[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (1000 lbs)
[, 7]	qsec	1/4 mile time
[, 8]	vs	Engine (0 = V-shaped, 1 = straight)
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears

Exercise 2

- Create a MANOVA model that investigates how MPG compares across different *gear* and *vs* configurations
  - Let the reference group be “3”
- Make sure to include an interaction between the two groups
- Based on our results, what do we expect the value of MPG for the following combinations:
  - V-shaped engine with gear = 3
  - Straight engine with gear = 4
  - V-shaped engine with gear = 5

A data frame with 32 observations on 11 (numeric) variables.

[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (1000 lbs)
[, 7]	qsec	1/4 mile time
[, 8]	vs	Engine (0 = V-shaped, 1 = straight)
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears