

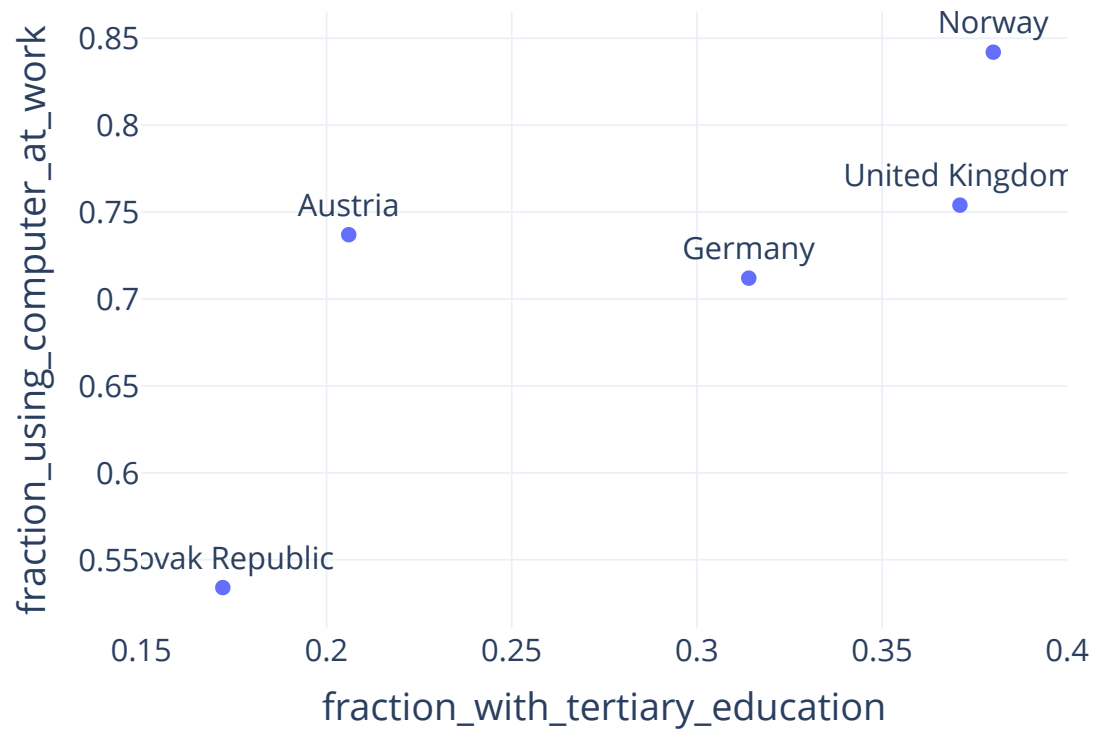
Effective Programming Practices for Economists

Data Analysis in Python

Working with statsmodels' results objects

Janoś Gabler, Hans-Martin von Gaudecker, and Tim Mensinger

Example



Model and Results Objects

```
>>> model = smf.ols(  
...     data=df,  
...     formula="fraction_using_computer_at_work ~ fraction_with_tertiary_education",  
... )  
>>> model  
<statsmodels.regression.linear_model.OLS at 0x7fb56c905250>  
  
>>> all_results = model.fit(cov_type="nonrobust")  
>>> all_results  
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f84b22e7490>
```

RegressionResultsWrapper contains methods and attributes for all results

- Coefficient estimates
- Predictions / Residuals
- Variance-covariance matrix of estimates
- Many tests

Summarising Regression Results

```
>>> all_results.summary()
```

Dep. Variable:	fraction_using_computer_at_work				R-squared:	0.628	
Model:	OLS				Adj. R-squared:	0.505	
Method:	Least Squares				F-statistic:	5.074	
No. Observations:	5				Df Residuals:	3	
		coef	std err	t	P> t 	[0.025	0.975]
	Intercept	0.4445	0.126	3.541	0.038	0.045	0.844
	fraction_with_tertiary_education	0.9399	0.417	2.253	0.110	-0.388	2.268

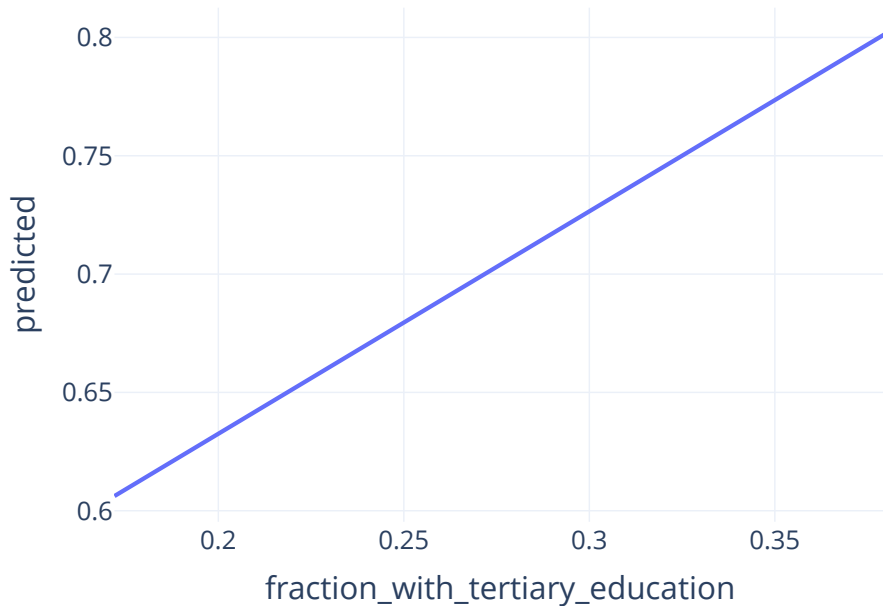
Add Mean Prediction to Data

```
>>> df["predicted"] = all_results.predict(df)
>>> df
```

country	fraction_with_tertiary_education	fraction_using_computer_at_work	predicted
Slovak Republic	0.172	0.534	0.606
Austria	0.206	0.737	0.638
Germany	0.314	0.712	0.74
United Kingdom	0.371	0.754	0.793
Norway	0.38	0.842	0.802

Plot the Regression Line

```
>>> line_fig = df.plot(x="fraction_with_tertiary_education", y="predicted")  
>>> line_fig.show()
```



Add Regression Line to Scatter Plot

```
>>> fig = df.reset_index().plot.scatter(  
...     x="fraction_with_tertiary_education",  
...     y="fraction_using_computer_at_work",  
...     text="country",  
... )  
>>> # Add the regression line  
>>> fig.add_traces(line_fig.data)  
>>> # Nicer formatting  
>>> fig.update_traces(textposition="bottom center")  
>>> fig.update_xaxes(range=(0.15, 0.4))  
>>> fig.show()
```

Data Points and Regression Line

