

# Parallel slopes linear regression

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON



**Maarten Van den Broeck**

Content Developer at DataCamp

# The previous course

This course assumes knowledge from [Introduction to Regression in Python with statsmodels](#).

# From simple regression to multiple regression

*Multiple regression* is a regression model with more than one explanatory variable.

More explanatory variables can give **more insight** and **better predictions**.

# The course contents

## Chapter 1

- "Parallel slopes" regression

## Chapter 3

- More explanatory variables
- How linear regression works

## Chapter 2

- Interactions
- Simpson's Paradox

## Chapter 4

- Multiple logistic regression
- The logistic distribution
- How logistic regression works

# The fish dataset

mass_g	length_cm	species
242.0	23.2	Bream
5.9	7.5	Perch
200.0	30.0	Pike
40.0	12.9	Roach

- Each row represents a fish
- `mass_g` is the response variable
- 1 numeric, 1 categorical explanatory variable

# One explanatory variable at a time

```
from statsmodels.formula.api import ols

mdl_mass_vs_length = ols("mass_g ~ length_cm",
                          data=fish).fit()
```

```
print(mdl_mass_vs_length.params)
```

```
Intercept    -536.223947
length_cm      34.899245
dtype: float64
```

- 1 intercept coefficient
- 1 slope coefficient

```
mdl_mass_vs_species = ols("mass_g ~ species + 0",
                           data=fish).fit()

print(mdl_mass_vs_species.params)
```

```
species[Bream]    617.828571
species[Perch]    382.239286
species[Pike]     718.705882
species[Roach]    152.050000
dtype: float64
```

- 1 intercept coefficient for each category

# Both variables at the same time

```
mdl_mass_vs_both = ols("mass_g ~ length_cm + species + 0",  
                        data=fish).fit()
```

```
print(mdl_mass_vs_both.params)
```

```
species[Bream]    -672.241866  
species[Perch]    -713.292859  
species[Pike]     -1089.456053  
species[Roach]    -726.777799  
length_cm         42.568554  
dtype: float64
```

- 1 slope coefficient
- 1 intercept coefficient for each category

# Comparing coefficients

```
print mdl_mass_vs_length.params
```

```
Intercept    -536.223947  
length_cm     34.899245
```

```
print mdl_mass_vs_species.params
```

```
species[Bream]    617.828571  
species[Perch]    382.239286  
species[Pike]     718.705882  
species[Roach]    152.050000
```

```
print mdl_mass_vs_both.params
```

```
species[Bream]    -672.241866  
species[Perch]    -713.292859  
species[Pike]     -1089.456053  
species[Roach]    -726.777799  
length_cm         42.568554
```

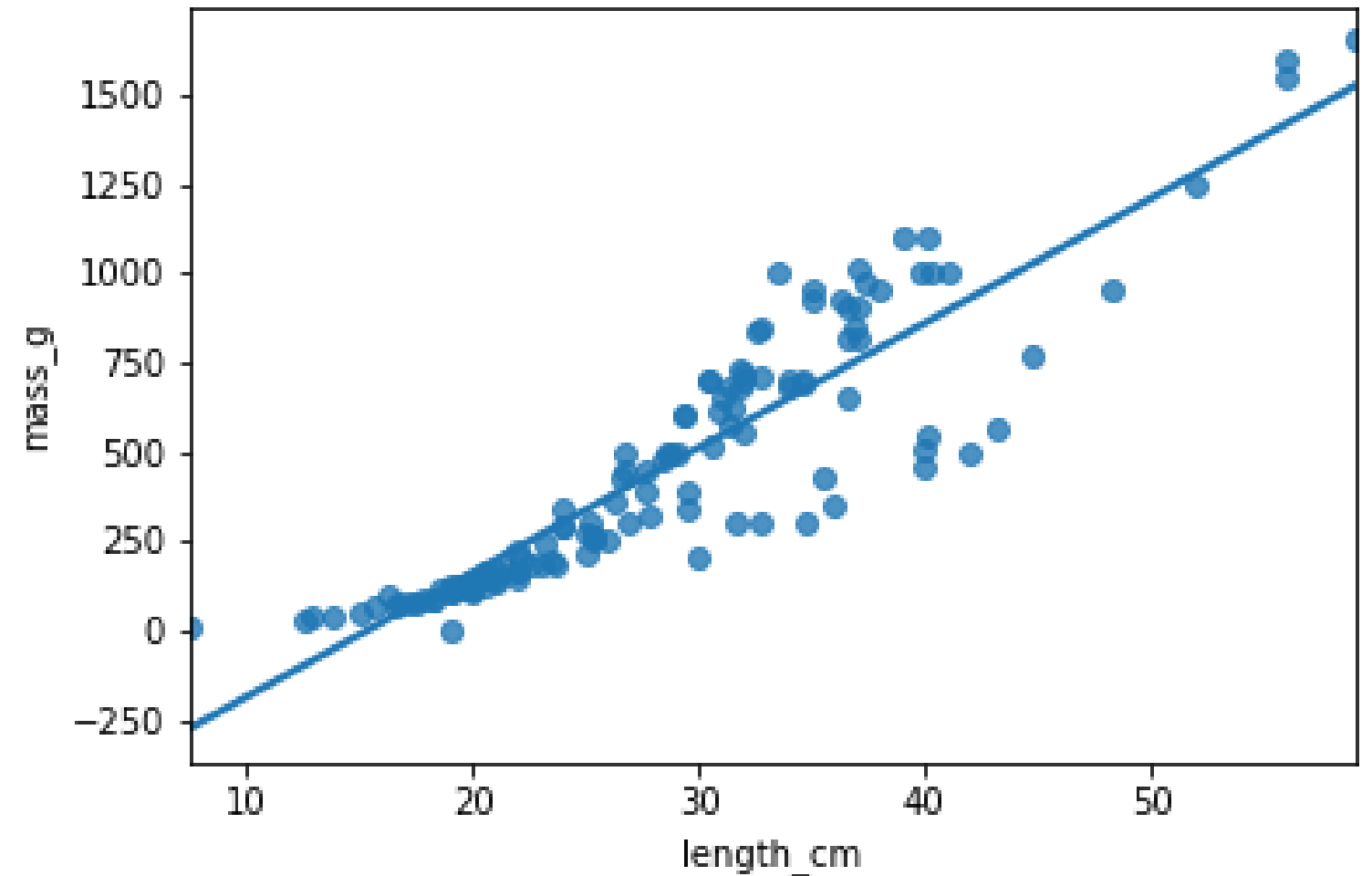


# Visualization: 1 numeric explanatory variable

```
import matplotlib.pyplot as plt
import seaborn as sns

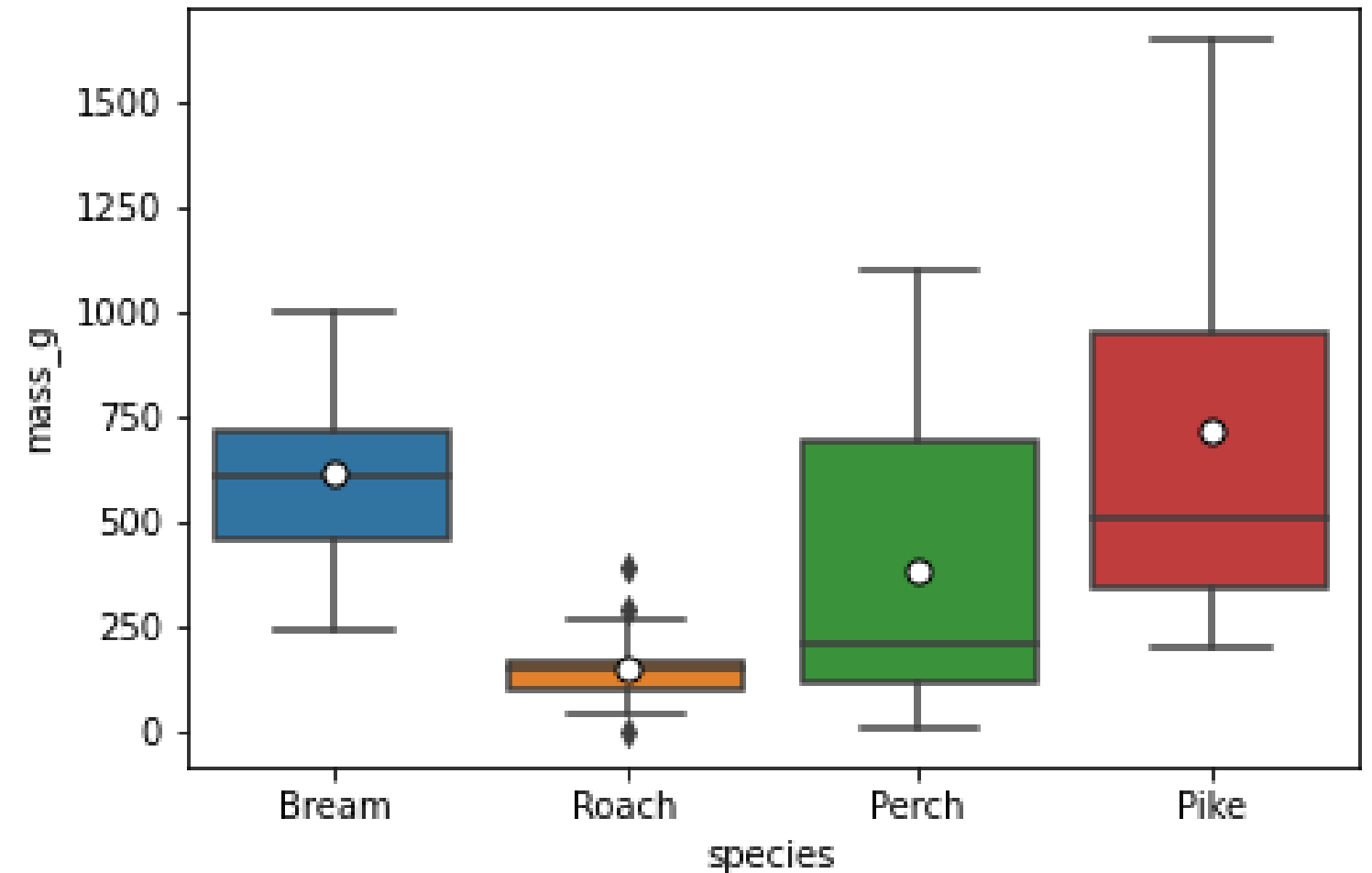
sns.regplot(x="length_cm",
            y="mass_g",
            data=fish,
            ci=None)

plt.show()
```



# Visualization: 1 categorical explanatory variable

```
sns.boxplot(x="species",  
            y="mass_g",  
            data=fish,  
            showmeans=True)
```



# Visualization: both explanatory variables

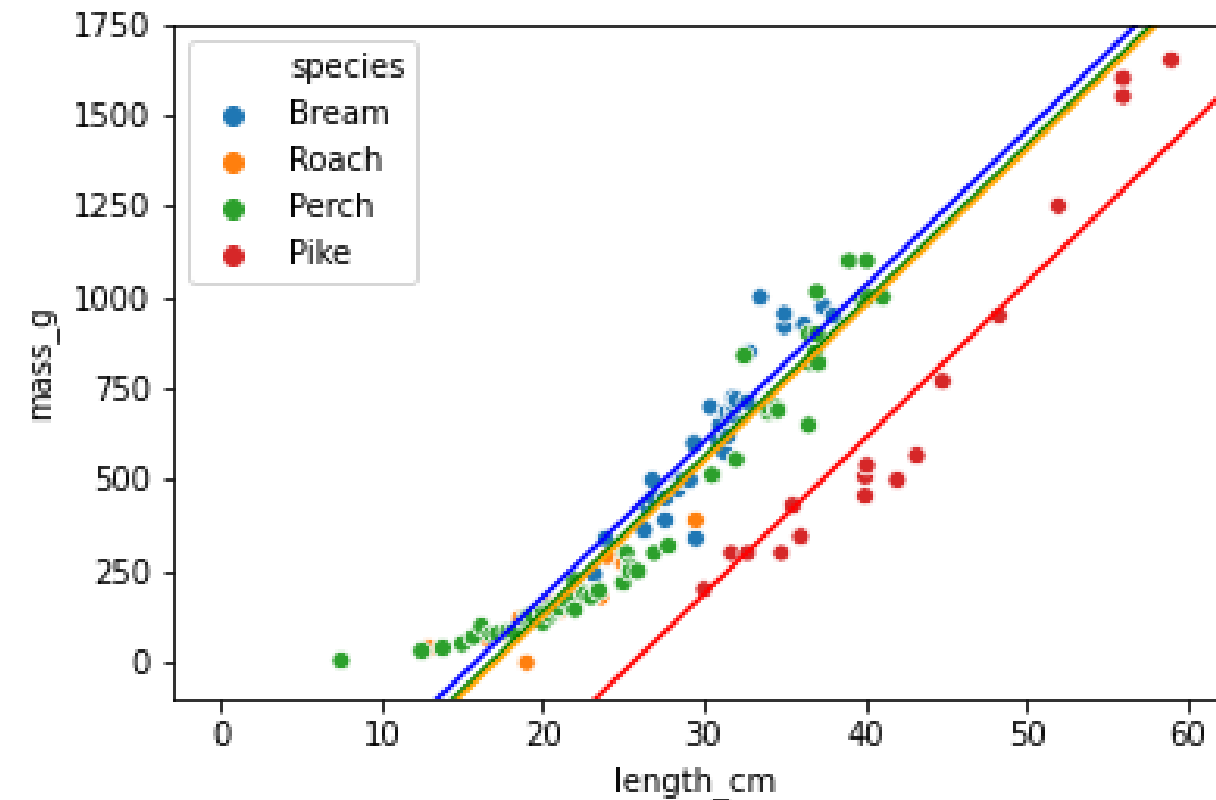
```
coeffs = mdl_mass_vs_both.params  
print(coeffs)
```

```
species[Bream]      -672.241866  
species[Perch]      -713.292859  
species[Pike]       -1089.456053  
species[Roach]      -726.777799  
length_cm           42.568554
```

```
ic_bream, ic_perch, ic_pike, ic_roach, sl = coeffs
```

```
sns.scatterplot(x="length_cm",  
                y="mass_g",  
                hue="species",  
                data=fish)
```

```
plt.axline(xy1=(0, ic_bream), slope=sl, color="blue")  
plt.axline(xy1=(0, ic_perch), slope=sl, color="green")  
plt.axline(xy1=(0, ic_pike), slope=sl, color="red")  
plt.axline(xy1=(0, ic_roach), slope=sl, color="orange")
```



# Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

# Predicting parallel slopes

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON



**Maarten Van den Broeck**

Content Developer at DataCamp

# The prediction workflow

```
import pandas as pd
import numpy as np
expl_data_length = pd.DataFrame(
    {"length_cm": np.arange(5, 61, 5)})
print(expl_data_length)
```

	length_cm
0	5
1	10
2	15
3	20
4	25
5	30
6	35
7	40
8	45
9	50
10	55
11	60

# The prediction workflow

```
[A, B, C] x [1, 2] ==> [A1, B1, C1, A2, B2, C2]
```

```
from itertools import product
product(["A", "B", "C"], [1, 2])
```

```
length_cm = np.arange(5, 61, 5)
species = fish["species"].unique()

p = product(length_cm, species)

expl_data_both = pd.DataFrame(p,
                              columns=['length_cm',
                                       'species'])

print(expl_data_both)
```

	length_cm	species
0	5	Bream
1	5	Roach
2	5	Perch
3	5	Pike
4	10	Bream
5	10	Roach
6	10	Perch
...		
41	55	Roach
42	55	Perch
43	55	Pike
44	60	Bream
45	60	Roach
46	60	Perch
47	60	Pike

# The prediction workflow

Predict `mass_g` from `length_cm` only

```
prediction_data_length = expl_data_length.assign(  
    mass_g = mdl_mass_vs_length.predict(expl_data)  
)
```

```
   length_cm  mass_g  
0         5 -361.7277  
1        10 -187.2315  
2        15  -12.7353  
3        20 161.7610  
4        25 336.2572  
5        30 510.7534  
... # number of rows: 12
```

Predict `mass_g` from both explanatory variables

```
prediction_data_both = expl_data_both.assign(  
    mass_g = mdl_mass_vs_both.predict(expl_data)  
)
```

```
   length_cm species  mass_g  
0         5   Bream -459.3991  
1         5   Roach -513.9350  
2         5   Perch -500.4501  
3         5    Pike -876.6133  
4        10   Bream -246.5563  
5        10   Roach -301.0923  
... # number of rows: 48
```

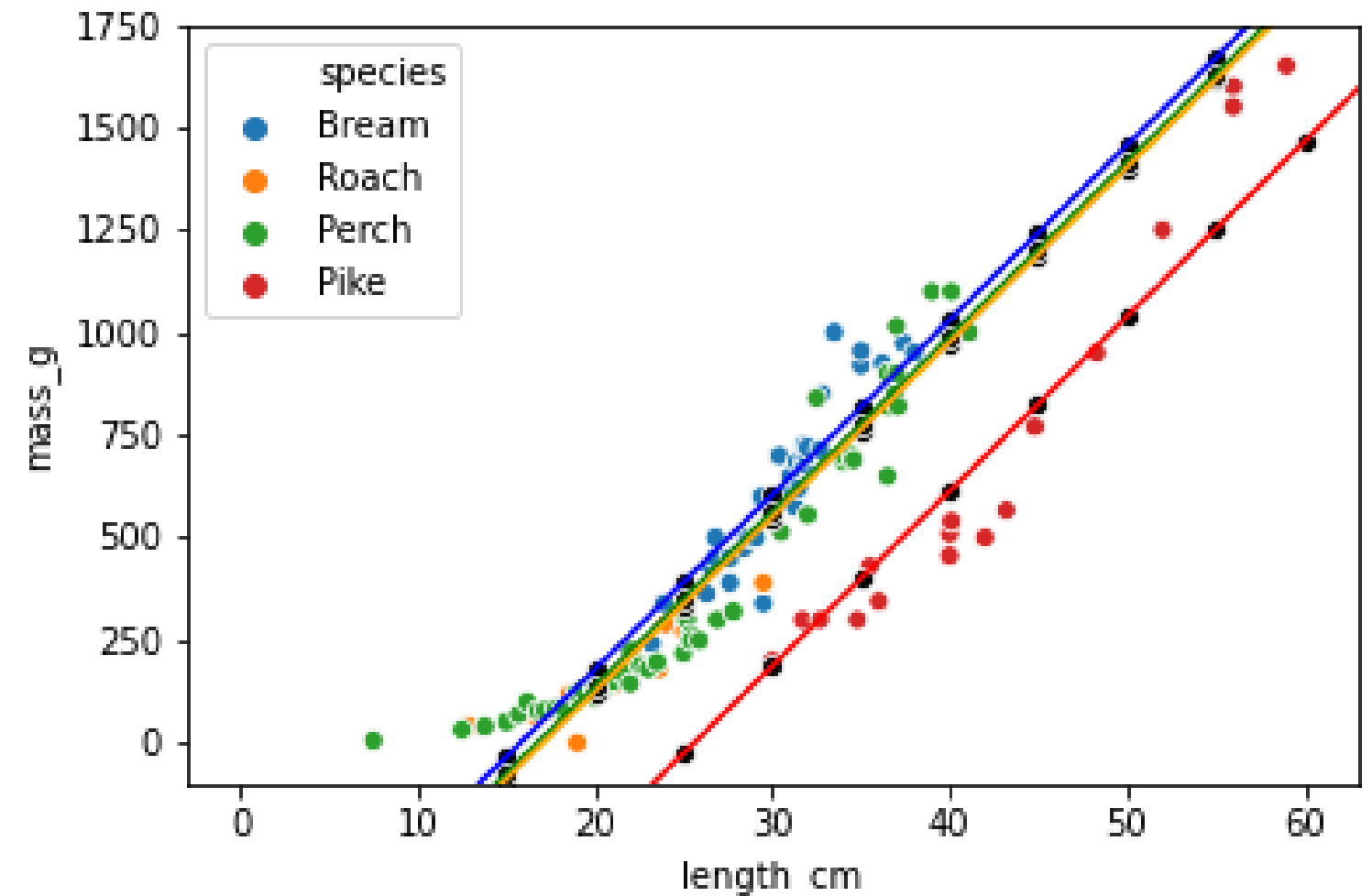


# Visualizing the predictions

```
plt.axline(xy1=(0, ic_bream), slope=sl, color="blue")
plt.axline(xy1=(0, ic_perch), slope=sl, color="green")
plt.axline(xy1=(0, ic_pike), slope=sl, color="red")
plt.axline(xy1=(0, ic_roach), slope=sl, color="orange")
```

```
sns.scatterplot(x="length_cm",
               y="mass_g",
               hue="species",
               data=fish)
```

```
sns.scatterplot(x="length_cm",
               y="mass_g",
               color="black",
               data=prediction_data)
```



# Manually calculating predictions for linear regression

```
coeffs = mdl_mass_vs_length.params  
print(coeffs)
```

```
intercept, slope = coeffs
```

```
explanatory_data = pd.DataFrame(  
    {"length_cm": np.arange(5, 61, 5)})  
  
prediction_data = explanatory_data.assign(  
    mass_g = intercept + slope * explanatory_data  
    )
```

```
print(prediction_data)
```

```
Intercept    -536.223947  
length_cm     34.899245
```

	length_cm	mass_g
0	5	-361.727721
1	10	-187.231494
2	15	-12.735268
3	20	161.760959
4	25	336.257185
5	30	510.753412
...		
9	50	1208.738318
10	55	1383.234545
11	60	1557.730771

# Manually calculating predictions for multiple regression

```
coeffs = mdl_mass_vs_both.params  
print(coeffs)
```

```
species[Bream]    -672.241866  
species[Perch]    -713.292859  
species[Pike]     -1089.456053  
species[Roach]    -726.777799  
length_cm         42.568554
```

```
ic_bream, ic_perch, ic_pike, ic_roach, slope = coeffs
```

# np.select()

```
conditions = [  
    condition_1,  
    condition_2,  
    # ...  
    condition_n  
]  
  
choices = [list_of_choices] # same length as conditions  
  
np.select(conditions, choices)
```

# Choosing an intercept with np.select()

```
conditions = [  
    explanatory_data["species"] == "Bream",  
    explanatory_data["species"] == "Perch",  
    explanatory_data["species"] == "Pike",  
    explanatory_data["species"] == "Roach"  
]
```

```
choices = [ic_bream, ic_perch, ic_pike, ic_roach]
```

```
intercept = np.select(conditions, choices)
```

```
print(intercept)
```

```
[ -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46  
 -672.24  -726.78  -713.29 -1089.46]
```

# The final prediction step

```
prediction_data = explanatory_data.assign(  
    intercept = np.select(conditions, choices),  
    mass_g = intercept + slope * explanatory_data["length_cm"])  
  
print(prediction_data)
```

	length_cm	species	intercept	mass_g
0	5	Bream	-672.2419	-459.3991
1	5	Roach	-726.7778	-513.9350
2	5	Perch	-713.2929	-500.4501
3	5	Pike	-1089.4561	-876.6133
4	10	Bream	-672.2419	-246.5563
5	10	Roach	-726.7778	-301.0923
6	10	Perch	-713.2929	-287.6073
7	10	Pike	-1089.4561	-663.7705
8	15	Bream	-672.2419	-33.7136
...				
40	55	Bream	-672.2419	1669.0286
41	55	Roach	-726.7778	1614.4927
42	55	Perch	-713.2929	1627.9776
43	55	Pike	-1089.4561	1251.8144
44	60	Bream	-672.2419	1881.8714
45	60	Roach	-726.7778	1827.3354
46	60	Perch	-713.2929	1840.8204
47	60	Pike	-1089.4561	1464.6572

# Compare to .predict()

```
mdl_mass_vs_both.predict(explanatory_data)
```

```
0      -459.3991
1      -513.9350
2      -500.4501
3      -876.6133
4      -246.5563
5      -301.0923
...
43     1251.8144
44     1881.8714
45     1827.3354
46     1840.8204
47     1464.6572
```

# Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON



# Assessing model performance

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON



**Maarten Van den Broeck**

Content Developer at DataCamp

# Model performance metrics

- *Coefficient of determination (R-squared)*: how well the linear regression line fits the observed values.
  - Larger is better.
- *Residual standard error (RSE)*: the typical size of the residuals.
  - Smaller is better.

# Getting the coefficient of determination

```
print mdl_mass_vs_length.rsquared)
```

```
0.8225689502644215
```

```
print(mdl_mass_vs_species.rsquared)
```

```
0.25814887709499157
```

```
print(mdl_mass_vs_both.rsquared)
```

```
0.9200433561156649
```

# Adjusted coefficient of determination

- More explanatory variables increases  $R^2$ .
- Too many explanatory variables causes overfitting.
- *Adjusted coefficient of determination* penalizes more explanatory variables.
- $\bar{R}^2 = 1 - (1 - R^2) \frac{n_{obs} - 1}{n_{obs} - n_{var} - 1}$
- Penalty is noticeable when  $R^2$  is small, or  $n_{var}$  is large fraction of  $n_{obs}$ .
- In `statsmodels`, it's contained in the `rsquared_adj` attribute.

# Getting the adjusted coefficient of determination

```
print("rsq_length: ", mdl_mass_vs_length.rsquared)
print("rsq_adj_length: ", mdl_mass_vs_length.rsquared_adj)
```

```
rsq_length: 0.8225689502644215
rsq_adj_length: 0.8211607673300121
```

```
print("rsq_species: ", mdl_mass_vs_species.rsquared)
print("rsq_adj_species: ", mdl_mass_vs_species.rsquared_adj)
```

```
rsq_species: 0.25814887709499157
rsq_adj_species: 0.24020086605696722
```

```
print("rsq_both: ", mdl_mass_vs_both.rsquared)
print("rsq_adj_both: ", mdl_mass_vs_both.rsquared_adj)
```

```
rsq_both: 0.9200433561156649
rsq_adj_both: 0.9174431400543857
```

# Getting the residual standard error

```
rse_length = np.sqrt mdl_mass_vs_length.mse_resid)
print("rse_length: ", rse_length)
```

```
rse_length: 152.12092835414788
```

```
rse_species = np.sqrt mdl_mass_vs_species.mse_resid)
print("rse_species: ", rse_species)
```

```
rse_species: 313.5501156682592
```

```
rse_both = np.sqrt mdl_mass_vs_both.mse_resid)
print("rse_both: ", rse_both)
```

```
rse_both: 103.35563303966488
```

# Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON