# Two numeric explanatory variables

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

**Maarten Van den Broeck**
Content Developer at DataCamp

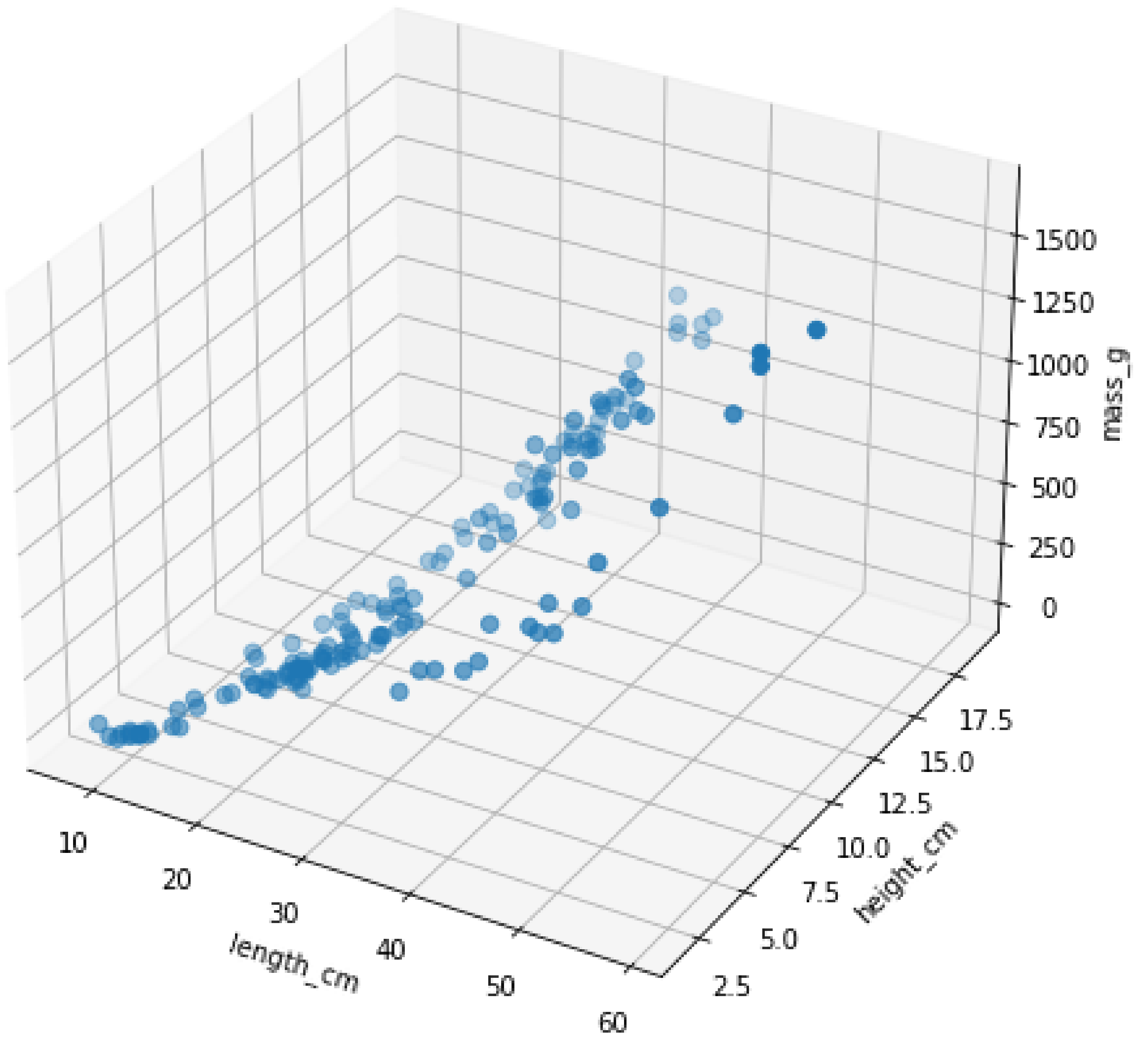
# Visualizing three numeric variables

- 3D scatter plot
- 2D scatter plot with response as color

# Another column for the fish dataset

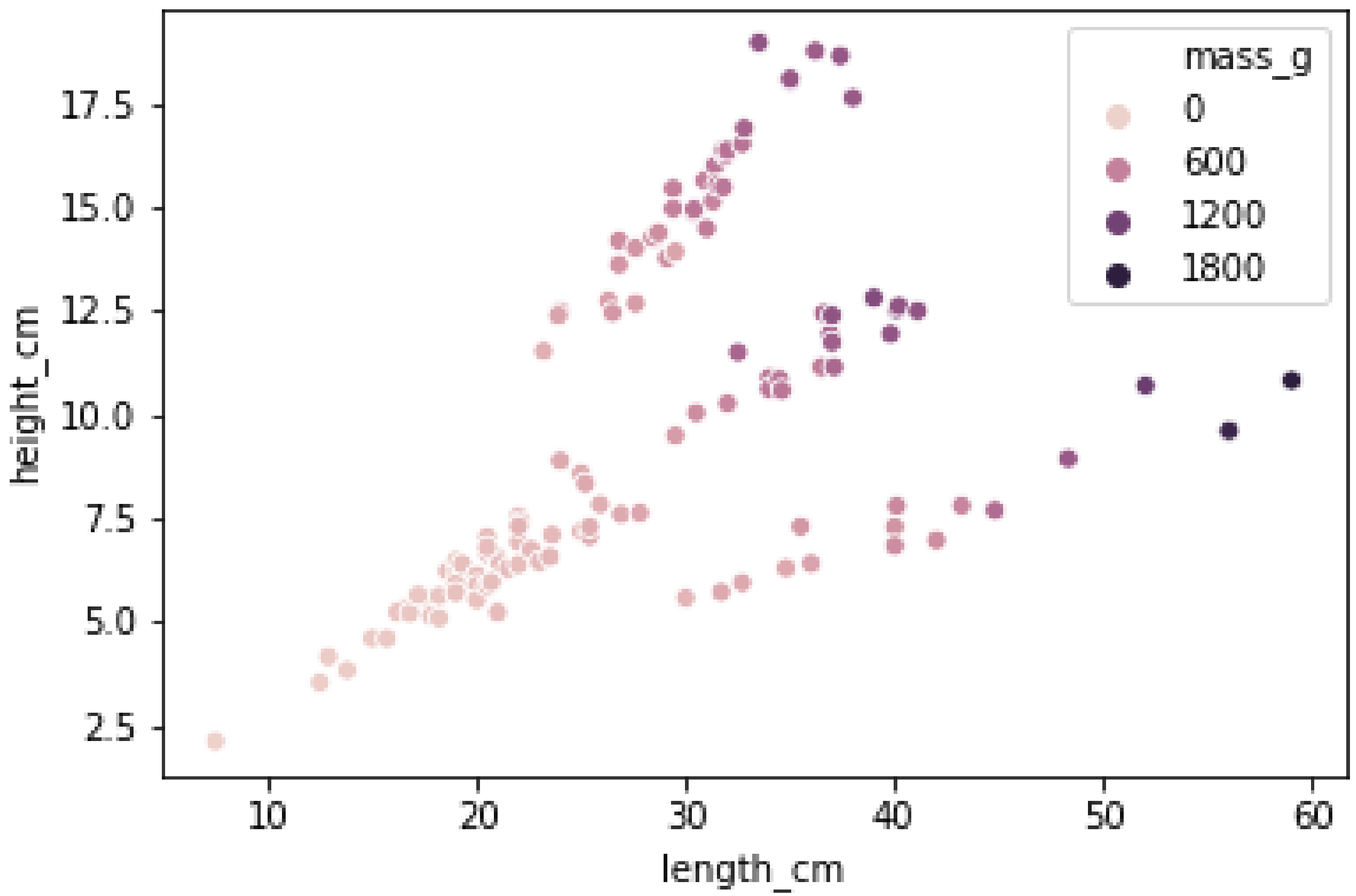
| species | mass_g | length_cm | height_cm |
|---|---|---|---|
| Bream | 1000 | 33.5 | 18.96 |
| Bream | 925 | 36.2 | 18.75 |
| Roach | 290 | 24.0 | 8.88 |
| Roach | 390 | 29.5 | 9.48 |
| Perch | 1100 | 39.0 | 12.80 |
| Perch | 1000 | 40.2 | 12.60 |
| Pike | 1250 | 52.0 | 10.69 |
| Pike | 1650 | 59.0 | 10.81 |

# 3D scatter plot



# 2D scatter plot, color for response

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



# Modeling with two numeric explanatory variables

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


print(mdl_mass_vs_both.params)
```

```
Intercept   -622.150234
length_cm     28.968405
height_cm     26.334804
```

# The prediction flow

```python
from itertools import product

length_cm = np.arange(5, 61, 5)
height_cm = np.arange(2, 21, 2)

p = product(length_cm, height_cm)

explanatory_data = pd.DataFrame(p,
                                columns=["length_cm",
                                         "height_cm"])
```

```python
prediction_data = explanatory_data.assign(
  mass_g = mdl_mass_vs_both.predict(explanatory_data))
```

```python
print(prediction_data)
```

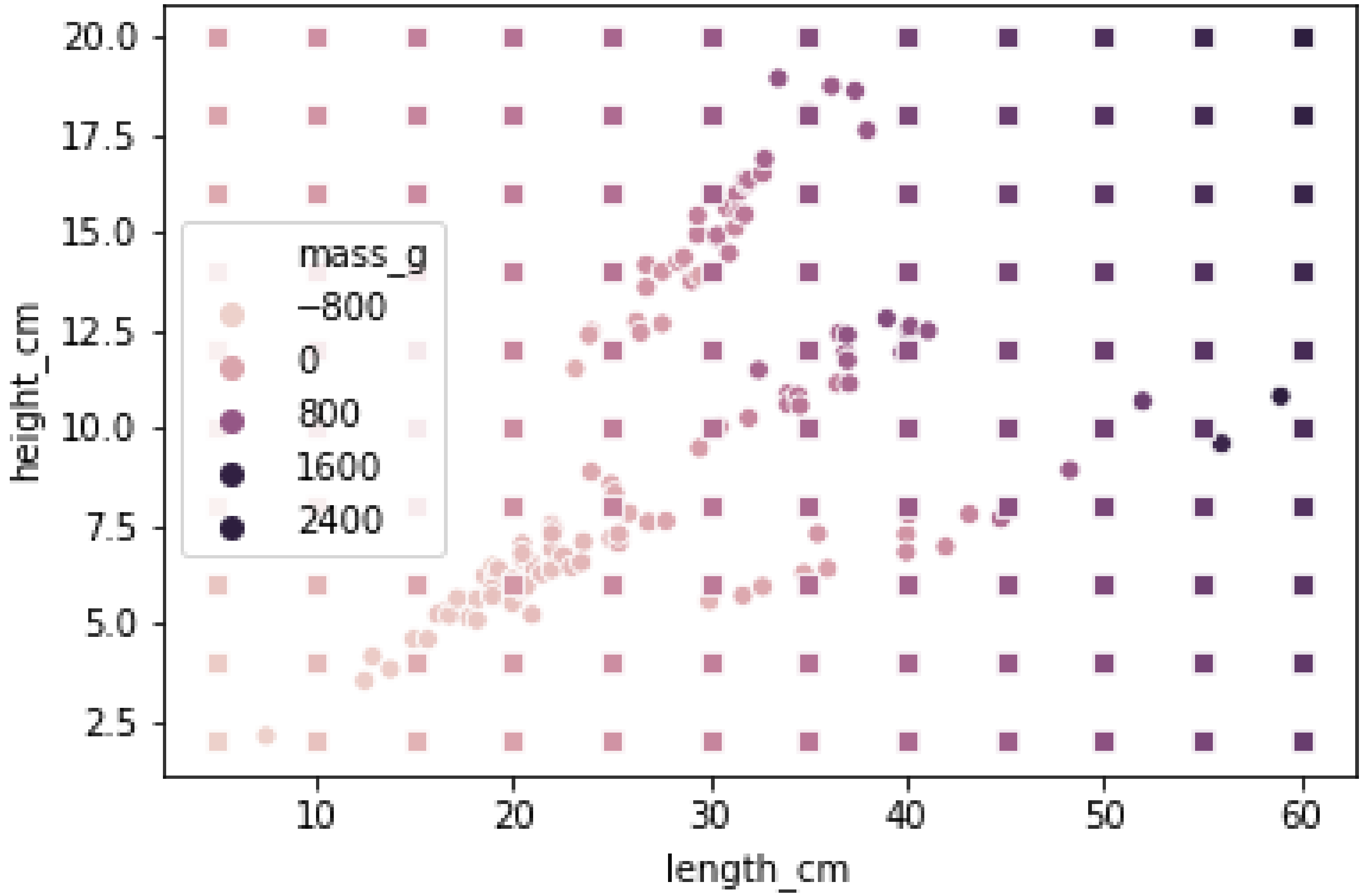
```
     length_cm  height_cm       mass_g
0            5          2  -424.638603
1            5          4  -371.968995
2            5          6  -319.299387
3            5          8  -266.629780
4            5         10  -213.960172
..         ...        ...          ...
115         60         12  1431.971694
116         60         14  1484.641302
117         60         16  1537.310909
118         60         18  1589.980517
119         60         20  1642.650125

[120 rows x 3 columns]
```

# Plotting the predictions

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

```
sns.scatterplot(x="length_cm",
                y="height_cm",
                data=prediction_data,
                hue="mass_g",
                legend=False,
                marker="s")
```

```
plt.show()
```



# Including an interaction

```
mdl_mass_vs_both_inter = ols("mass_g ~ length_cm * height_cm",
                              data=fish).fit()
```

```
print(mdl_mass_vs_both_inter.params)
```

```
Intercept                 159.107480
length_cm                   0.301426
height_cm                 -78.125178
length_cm:height_cm         3.545435
```

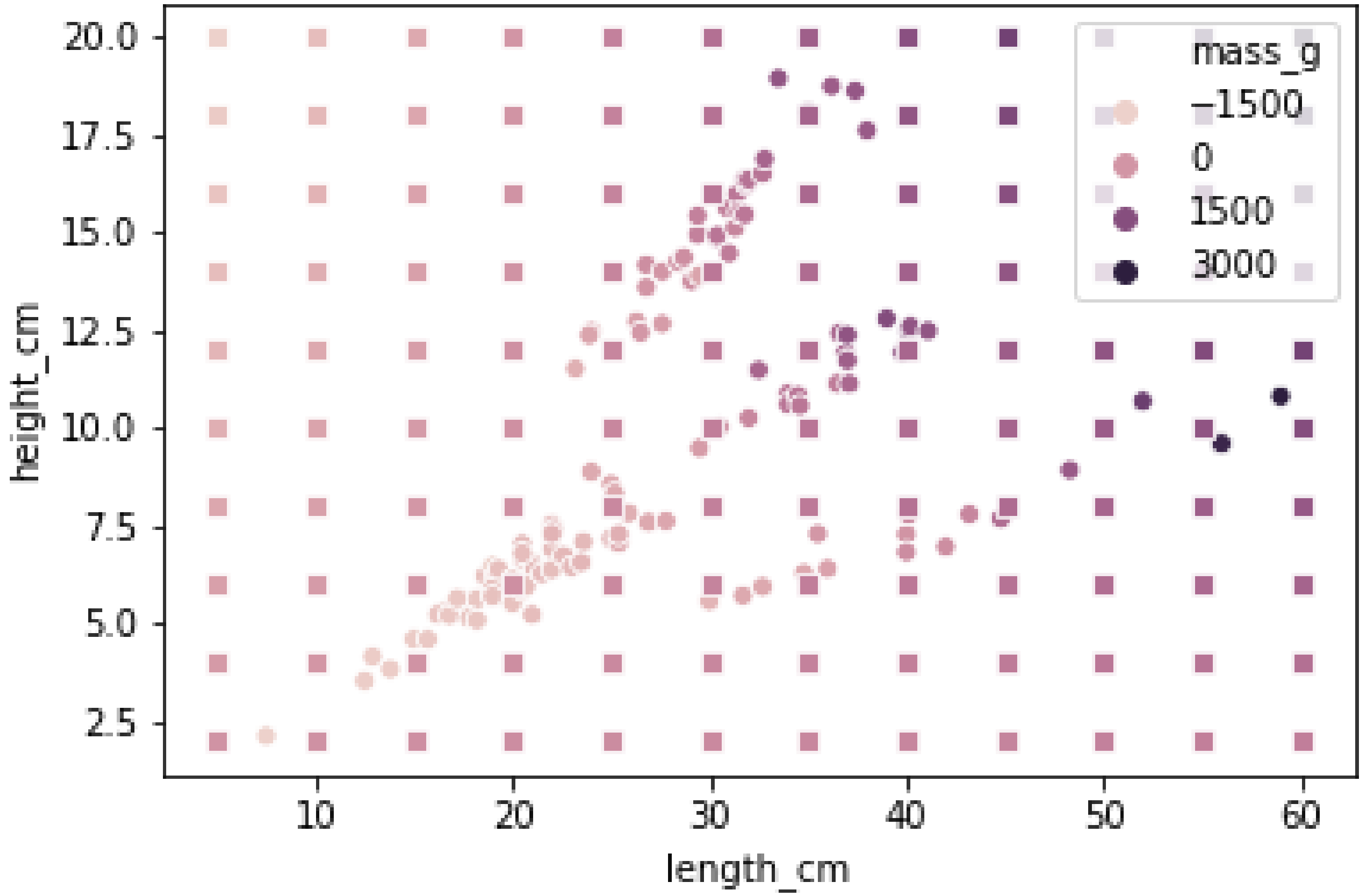
# The prediction flow with an interaction

```
length_cm = np.arange(5, 61, 5)
height_cm = np.arange(2, 21, 2)

p = product(length_cm, height_cm)

explanatory_data = pd.DataFrame(p,
                                columns=["length_cm",
                                         "height_cm"])

prediction_data = explanatory_data.assign(
  mass_g = mdl_mass_vs_both_inter.predict(explanatory_data))
```

# Plotting the predictions

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

sns.scatterplot(x="length_cm",
                y="height_cm",
                data=prediction_data,
                hue="mass_g",
                legend=False,
                marker="s")
```

```
plt.show()
```



# Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

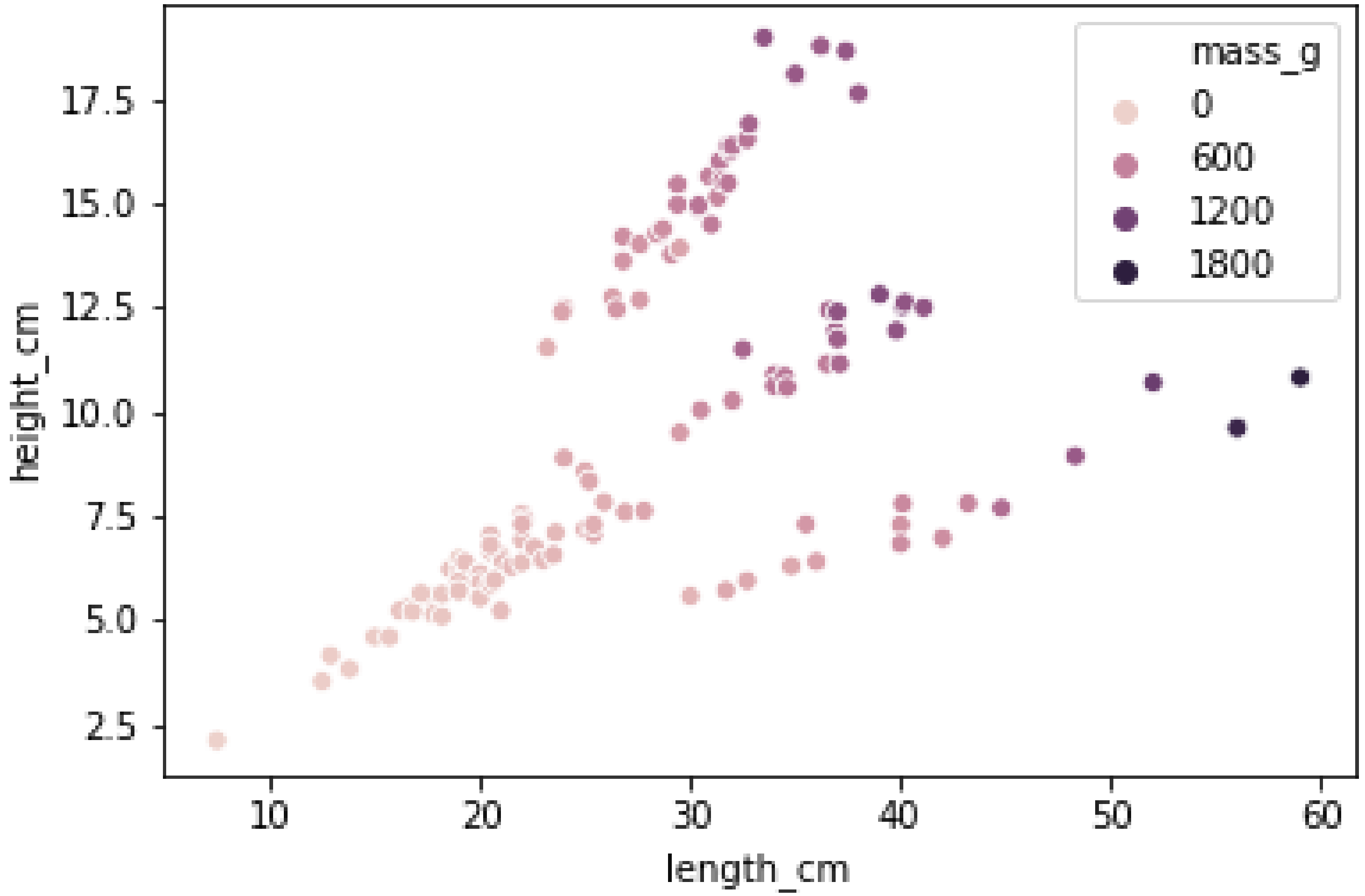
# More than two explanatory variables

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

**Maarten Van den Broeck**
Content Developer at DataCamp

# From last time

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

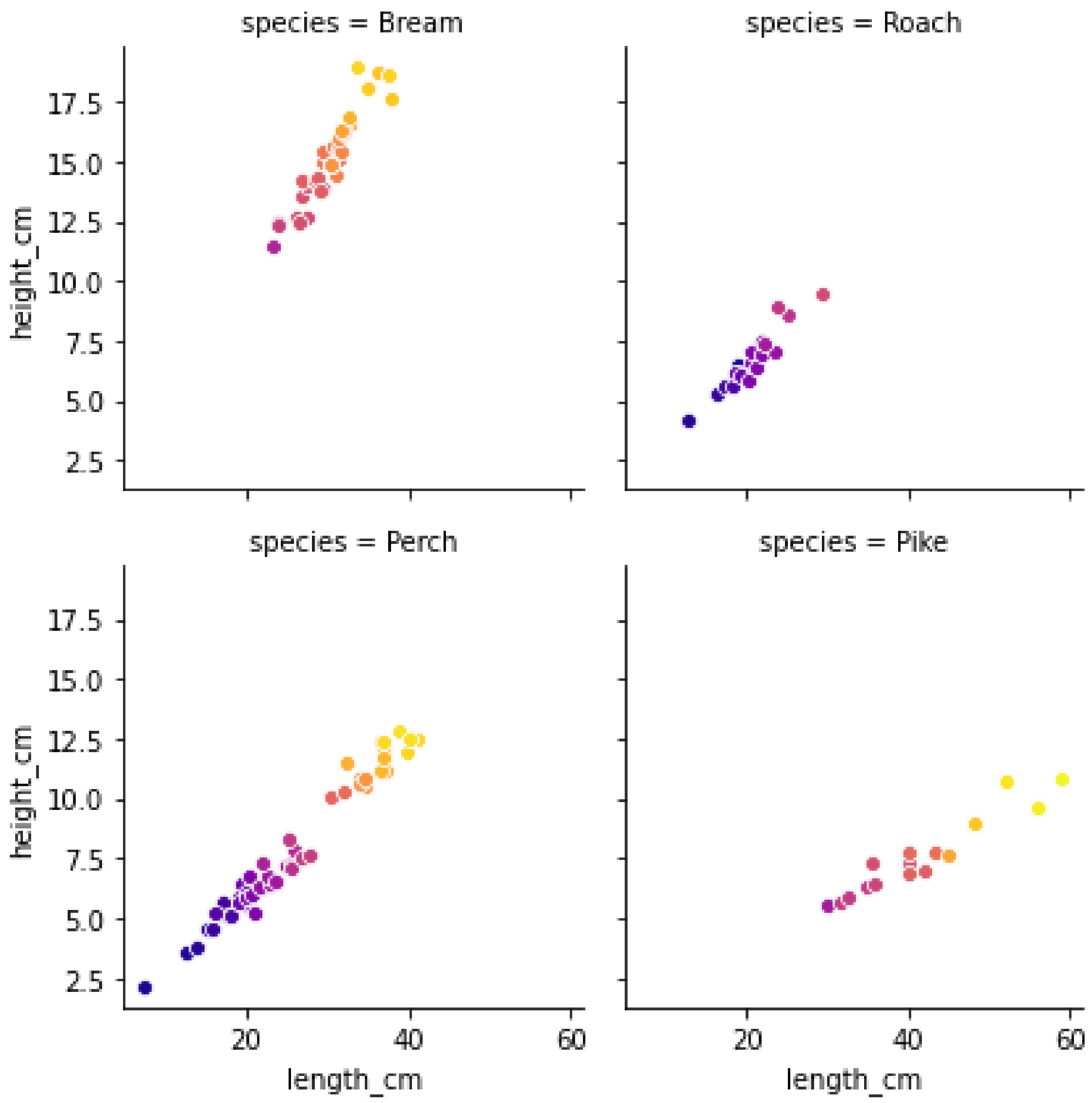


# Faceting by species

```
grid = sns.FacetGrid(data=fish,
                     col="species",
                     hue="mass_g",
                     col_wrap=2,
                     palette="plasma")
```

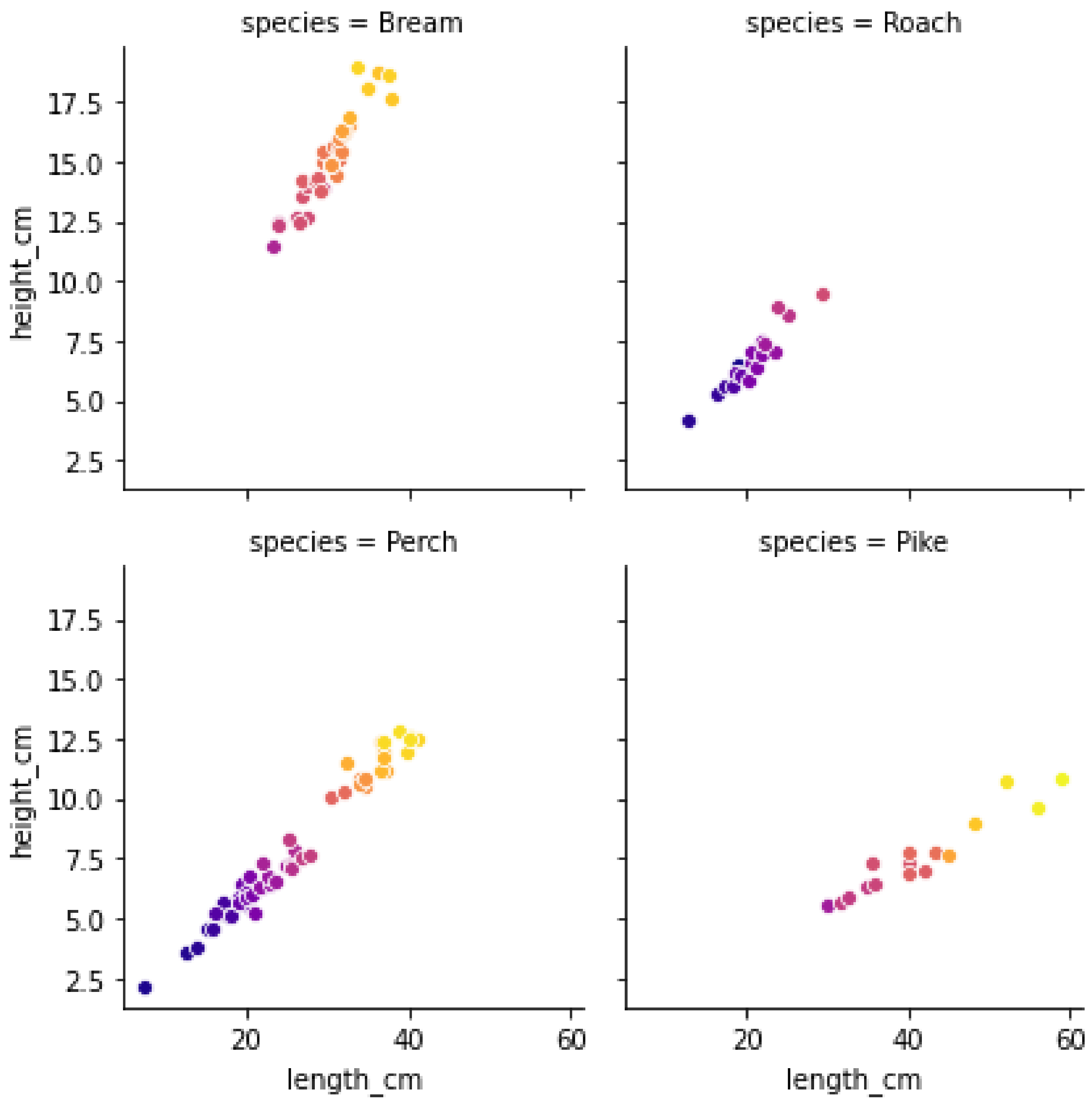
```
grid.map(sns.scatterplot,
         "length_cm",
         "height_cm")
```

```
plt.show()
```



# Faceting by species

- It's possible to use more than one categorical variable for faceting
- Beware of faceting overuse
- Plotting becomes harder with increasing number of variables



# Different levels of interaction

No interactions

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

two-way interactions between pairs of variables

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

three-way interaction between all three variables

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

# All the interactions

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

same as

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

# Only two-way interactions

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

same as

```
ols(
  "mass_g ~ (length_cm + height_cm + species) ** 2 + 0",
  data=fish).fit()
```

# The prediction flow

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

length_cm = np.arange(5, 61, 5)
height_cm = np.arange(2, 21, 2)
species = fish["species"].unique()

p = product(length_cm, height_cm, species)

explanatory_data = pd.DataFrame(p,
                                columns=["length_cm",
                                         "height_cm",
                                         "species"])


prediction_data = explanatory_data.assign(
  mass_g = mdl_mass_vs_all.predict(explanatory_data))
```

```
     length_cm  height_cm species       mass_g
0            5          2   Bream  -570.656437
1            5          2   Roach    31.449145
2            5          2   Perch    43.789984
3            5          2    Pike   271.270093
4            5          4   Bream  -451.127405
..         ...        ...     ...          ...
475         60         18    Pike  2690.346384
476         60         20   Bream  1531.618475
477         60         20   Roach  2621.797668
478         60         20   Perch  3041.931709
479         60         20    Pike  2926.352397

[480 rows x 4 columns]
```

# Let's practice!

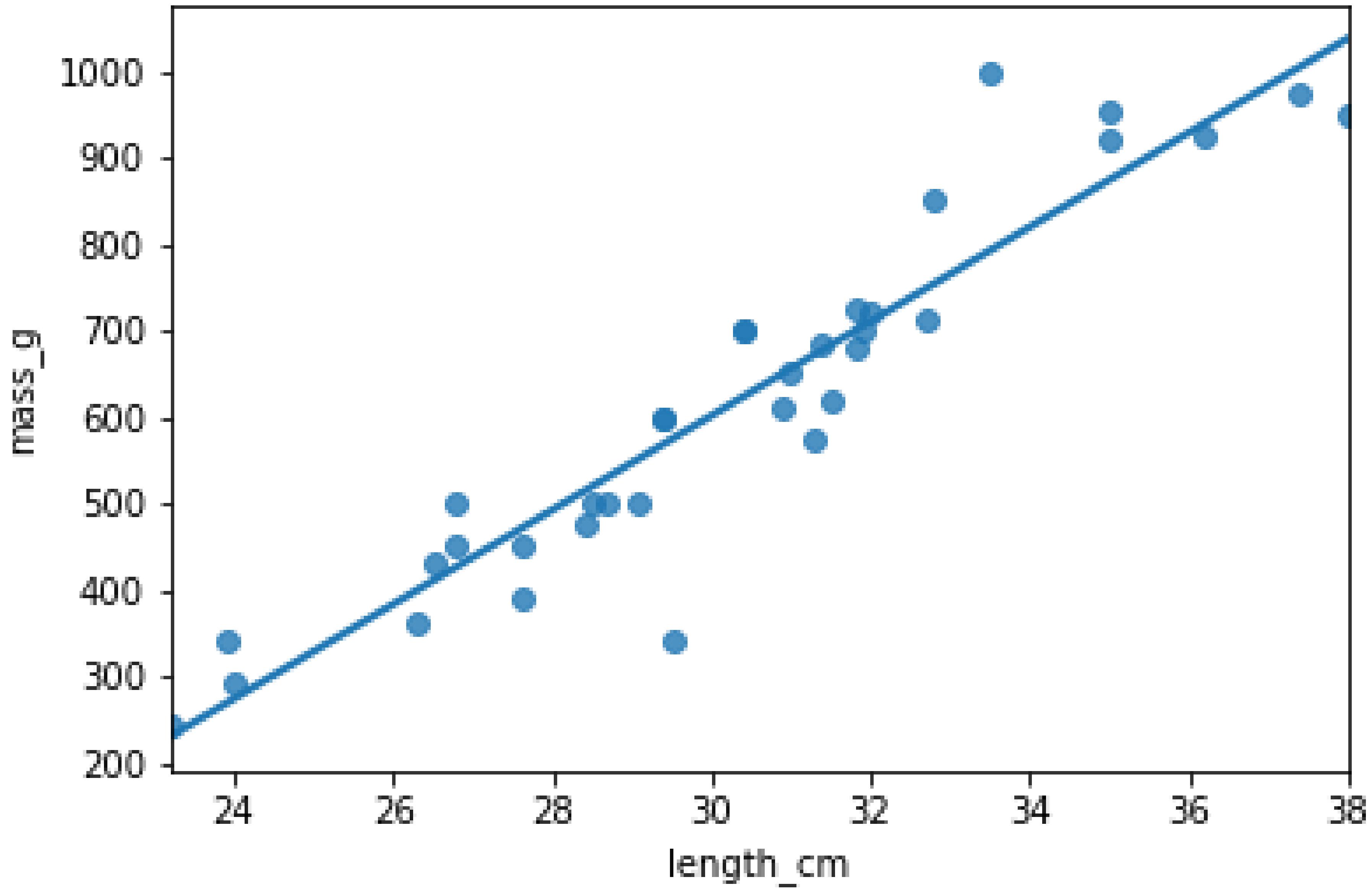
INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

# How linear regression works

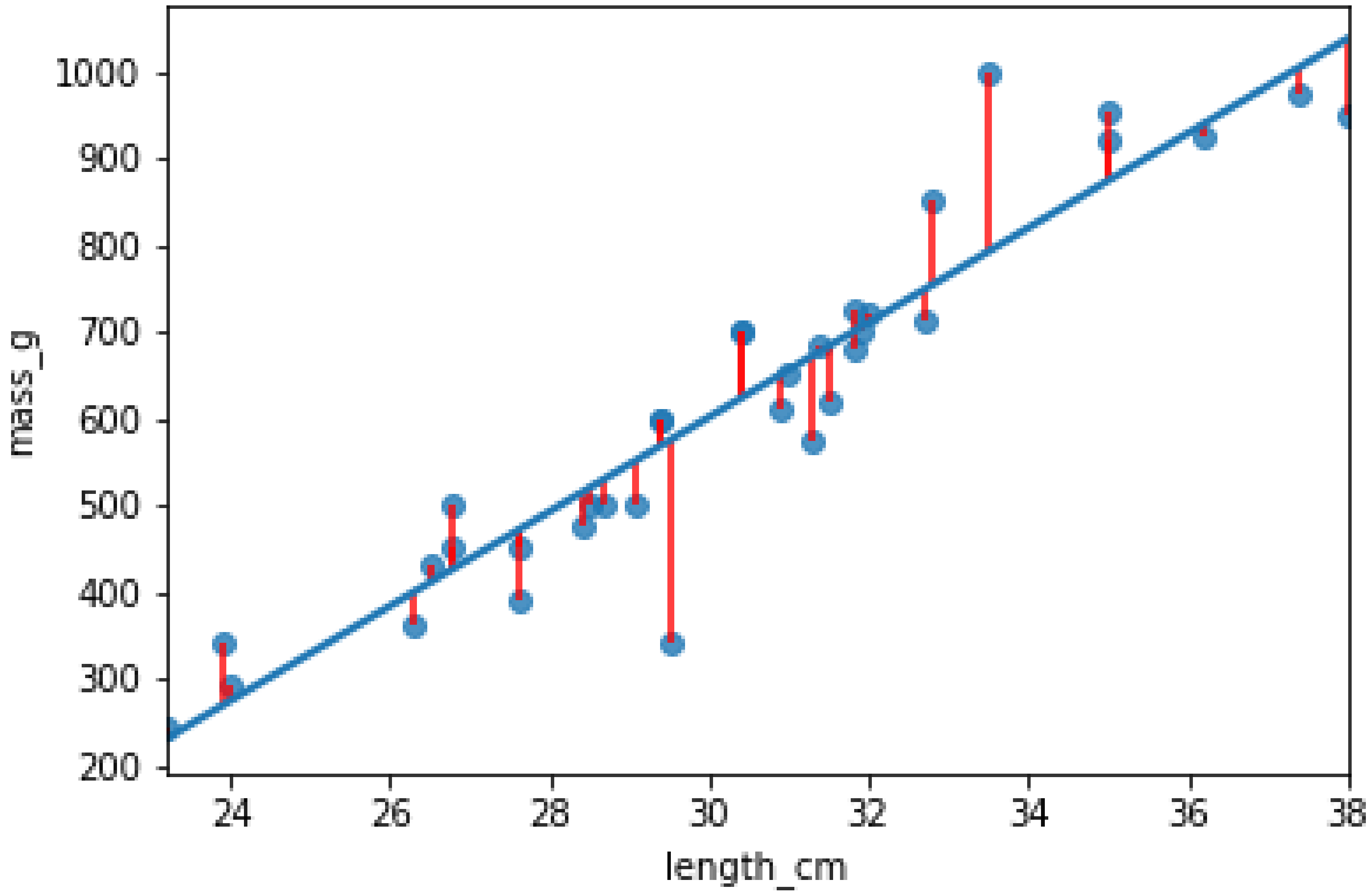
INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON

**Maarten Van den Broeck**
Content Developer at DataCamp

# The standard simple linear regression plot



# Visualizing residuals



# A metric for the best fit

## The simplest idea (which doesn't work)

- Take the sum of all the residuals.
- Some residuals are negative.

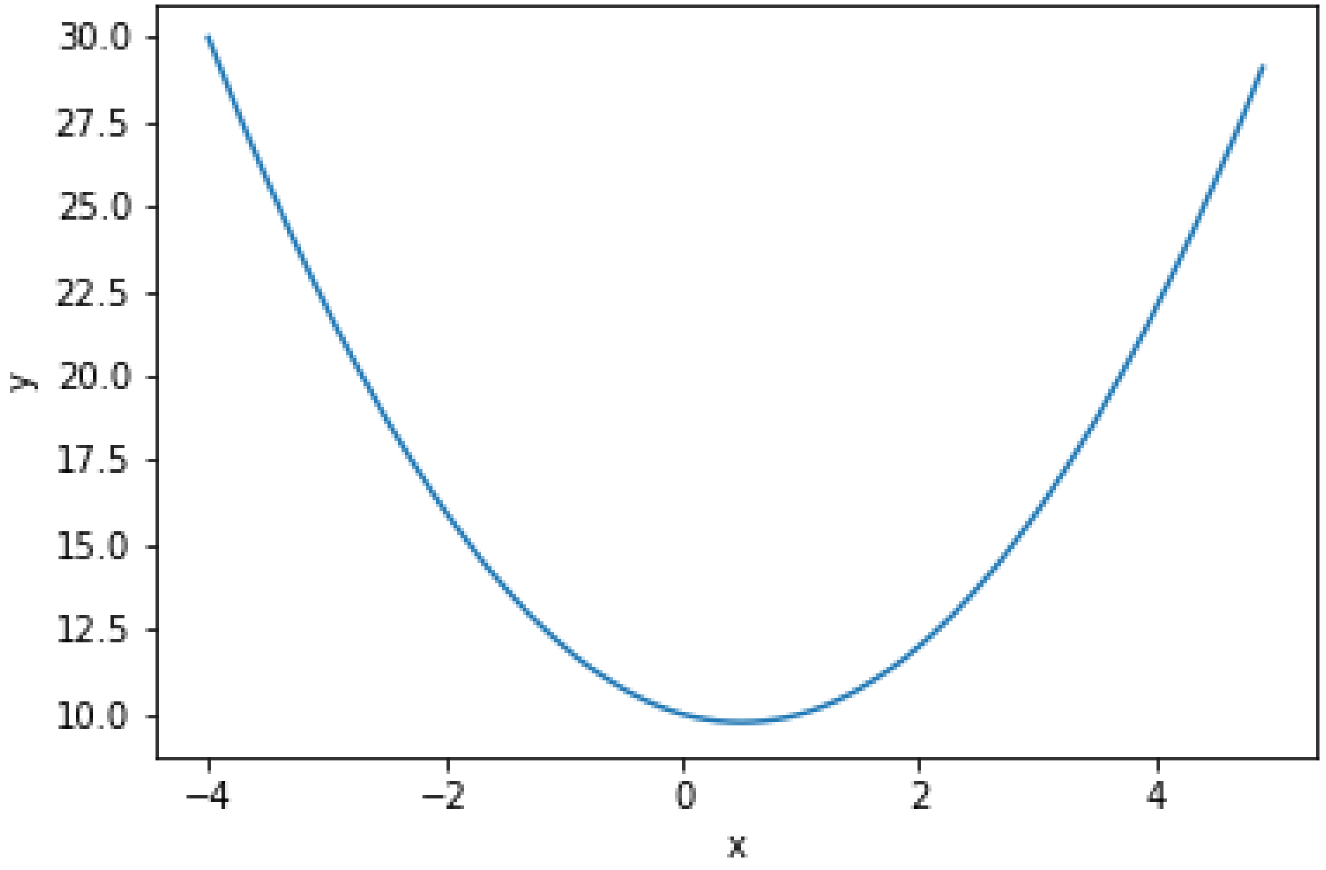
## The next simplest idea (which does work)

- Take the square of each residual, and add up those squares.
- This is called the *sum of squares*.

# A detour into numerical optimization

A line plot of a quadratic equation

```
x = np.arange(-4, 5, 0.1)
y = x ** 2 - x + 10

xy_data = pd.DataFrame({"x": x,
                        "y": y})


sns.lineplot(x="x",
             y="y",
             data=xy_data)
```



# Using calculus to solve the equation

$$y = x^2 - x + 10$$

$$\frac{\partial y}{\partial x} = 2x - 1$$

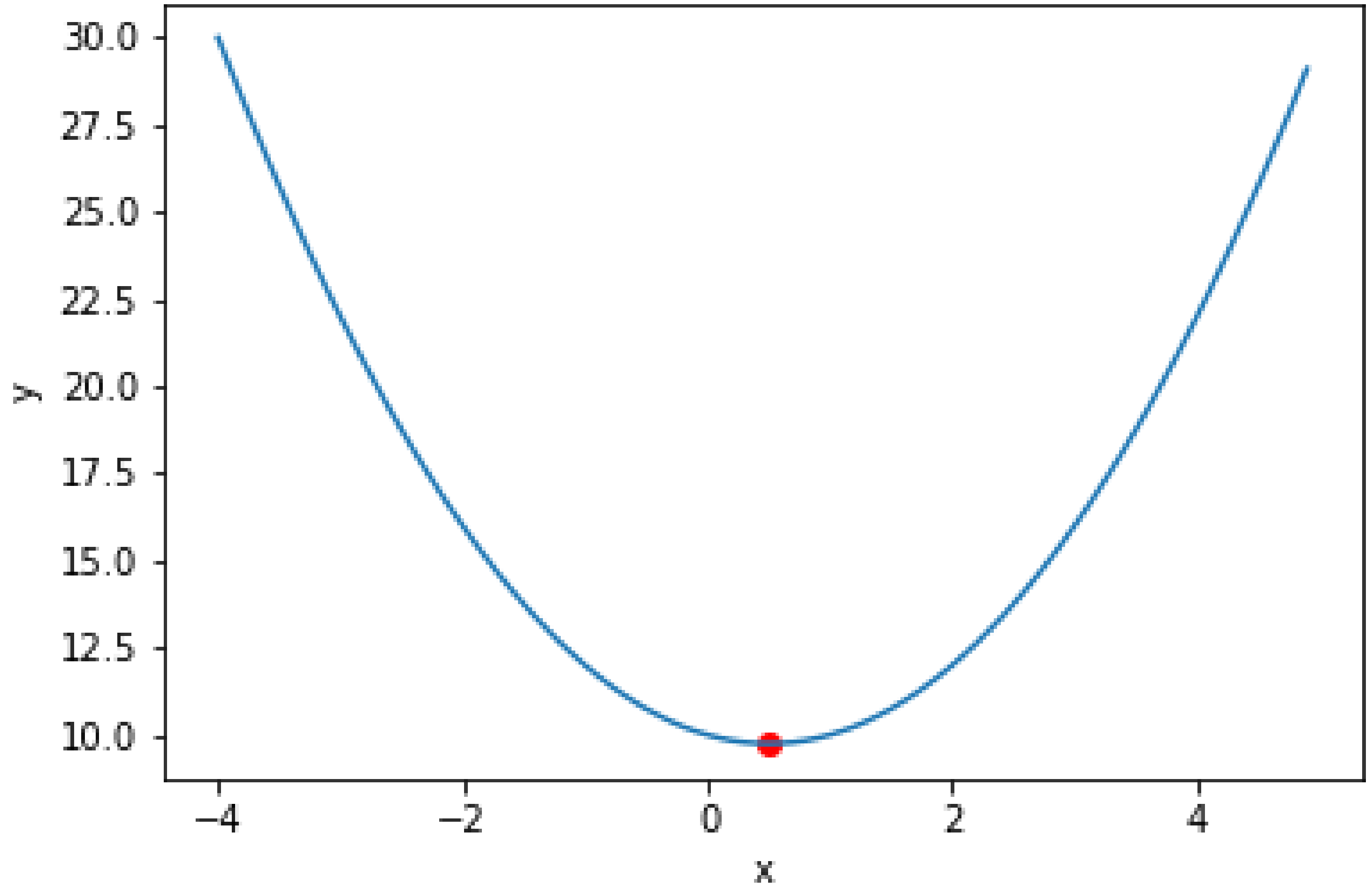
$$0 = 2x - 1$$

$$x = 0.5$$

$$y = 0.5^2 - 0.5 + 10 = 9.75$$

- Not all equations can be solved like this.
- You can let Python figure it out.

Don't worry if this doesn't make sense, you won't need it for the exercises.



# minimize()

```python
from scipy.optimize import minimize
```

```python
def calc_quadratic(x):
  y = x ** 2 - x + 10
  return y
```

```python
minimize(fun=calc_quadratic,
         x0=3)
```

```
      fun: 9.75
 hess_inv: array([[0.5]])
      jac: array([0.])
  message: 'Optimization terminated successfully.'
     nfev: 6
      nit: 2
     njev: 3
   status: 0
  success: True
        x: array([0.49999998])
```

# A linear regression algorithm

Define a function to calculate the sum of squares metric.

```
def calc_sum_of_squares(coeffs):
  intercept, slope = coeffs
  # More calculation!
```

Call `minimize()` to find coefficients that minimize this function.

```
minimize(
  fun=calc_sum_of_squares,
  x0=0
)
```

# Let's practice!

INTERMEDIATE REGRESSION WITH STATSMODELS IN PYTHON