Quantifying model fit

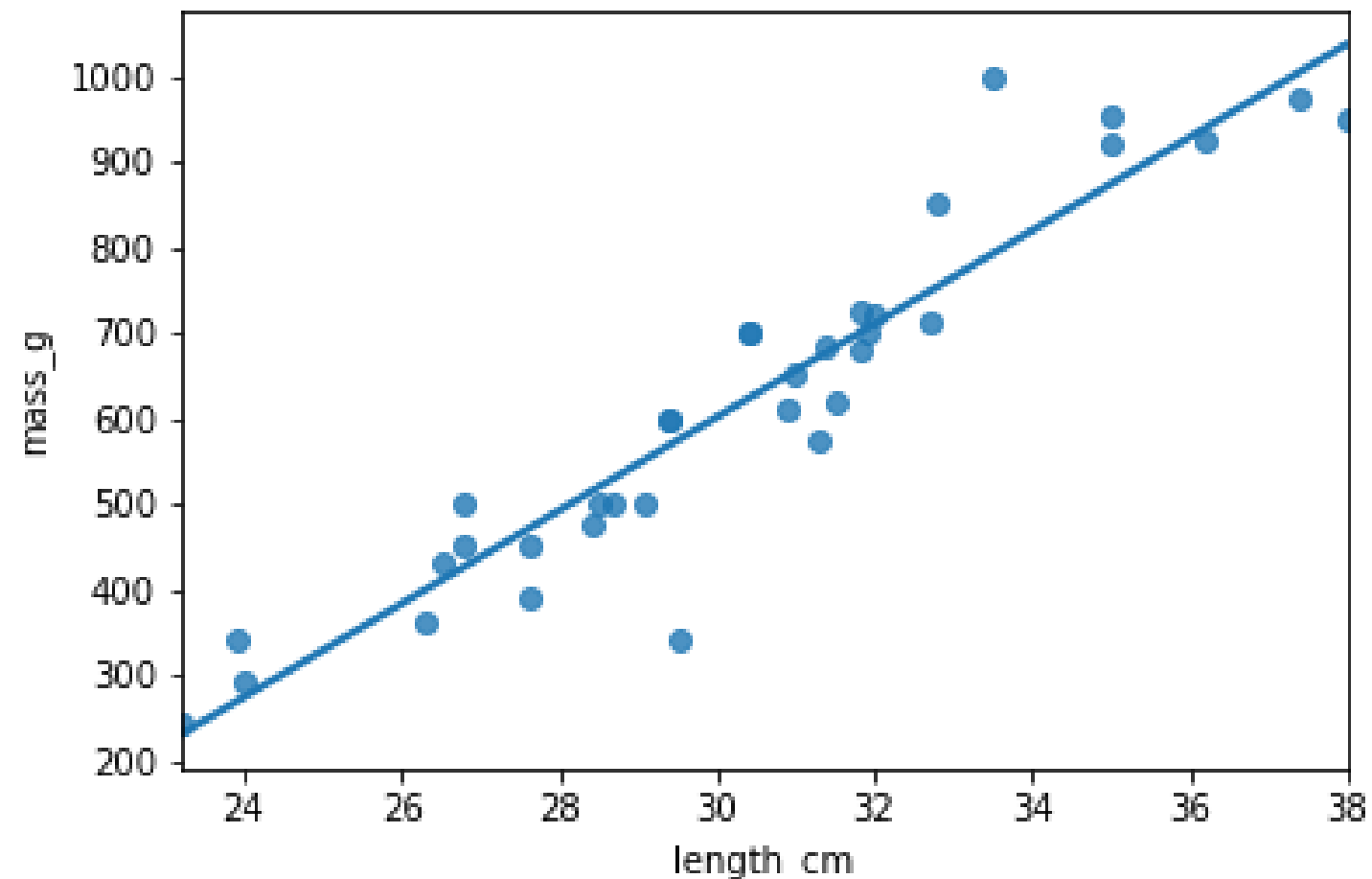## INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON
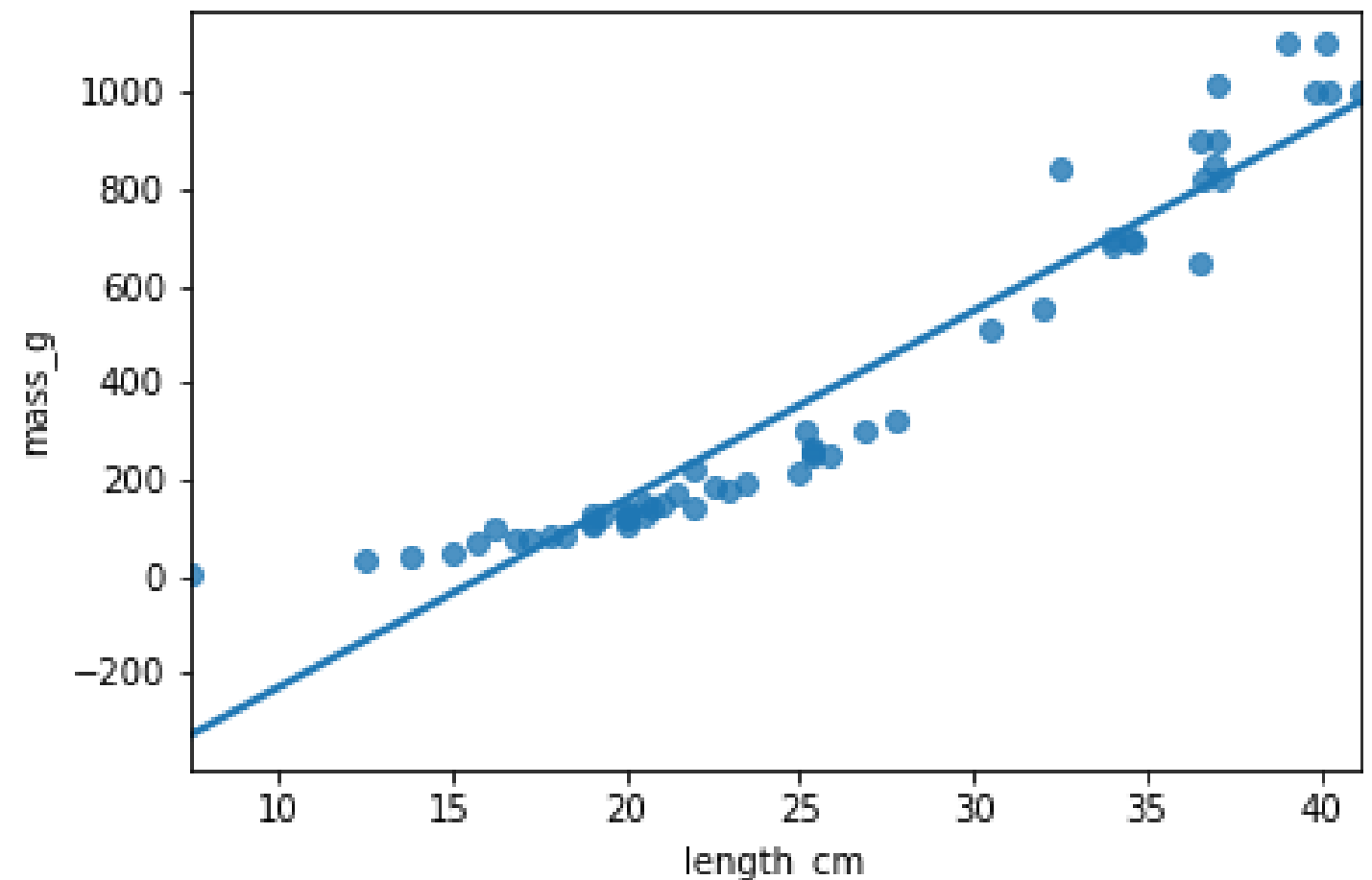
Maarten Van den Broeck
Content Developer at DataCamp

Bream and perch models

Bream

Perch

Coefficient of determination

Sometimes called "r-squared" or "R-squared".

> The proportion of the variance in the response variable that is predictable from the explanatory variable

- `1` means a perfect fit

- `0` means the worst possible fit

.summary()

Look at the value titled "R-Squared"

```python
mdl_bream = ols("mass_g ~ length_cm", data=bream).fit()
```

```python
print(mdl_bream.summary())
```

```
# Some lines of output omitted

                        OLS Regression Results

Dep. Variable:                 mass_g   R-squared:                 0.878
Model:                            OLS   Adj. R-squared:            0.874
Method:                 Least Squares   F-statistic:               237.6
```
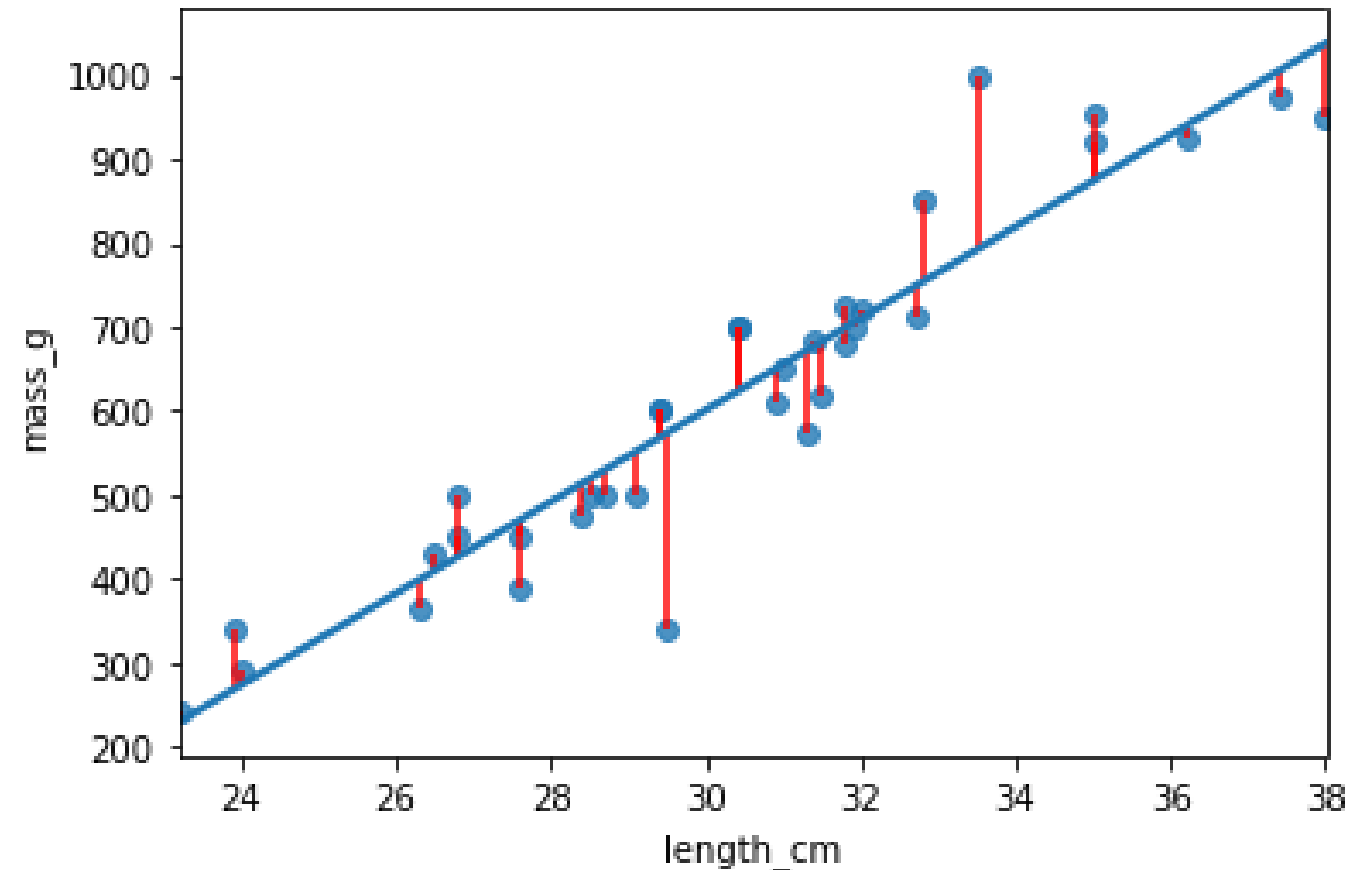
.rsquared attribute

```python
print(mdl_bream.rsquared)
```

```
0.8780627095147174
```

It's just correlation squared

```python
coeff_determination = bream["length_cm"].corr(bream["mass_g"]) ** 2
print(coeff_determination)
```

```
0.8780627095147173
```

Residual standard error (RSE)

- A "typical" difference between a prediction and an observed response

- It has the same unit as the response variable.

- MSE = RSE²

.mse_resid attribute

```
mse = mdl_bream.mse_resid
print('mse: ', mse)
```

```
mse:  5498.555084973521
```

```
rse = np.sqrt(mse)
print("rse: ", rse)
```

```
rse:  74.1522461594197
```

Calculating RSE: residuals squared

```python
residuals_sq = mdl_bream.resid ** 2

print("residuals sq: \n", residuals_sq)
```

```
residuals sq:
0          138.957118
1          260.758635
2         5126.992578
3         1318.919660
4          390.974309
        ...
30        2125.047026
31        6576.923291
32         206.259713
33         889.335096
34        7665.302003
Length: 35, dtype: float64
```

Calculating RSE: sum of residuals squared

```python
residuals_sq = mdl_bream.resid ** 2


resid_sum_of_sq = sum(residuals_sq)


print("resid sum of sq :",
      resid_sum_of_sq)
```

```
resid sum of sq : 181452.31780412616
```

Calculating RSE: degrees of freedom

```python
residuals_sq = mdl_bream.resid ** 2


resid_sum_of_sq = sum(residuals_sq)


deg_freedom = len(bream.index) - 2


print("deg freedom: ", deg_freedom)
```

```
deg freedom:  33
```

Degrees of freedom equals the number of observations minus the number of model coefficients.

Calculating RSE: square root of ratio

```python
residuals_sq = mdl_bream.resid ** 2

resid_sum_of_sq = sum(residuals_sq)

deg_freedom = len(bream.index) - 2

rse = np.sqrt(resid_sum_of_sq/deg_freedom)

print("rse :", rse)
```

```
rse : 74.15224261594197
```

Interpreting RSE

`mdl_bream` has an RSE of `74` .

> The difference between predicted bream masses and observed bream masses is typically about 74g.

Root-mean-square error (RMSE)

```python
residuals_sq = mdl_bream.resid ** 2

resid_sum_of_sq = sum(residuals_sq)

deg_freedom = len(bream.index) - 2

rse = np.sqrt(resid_sum_of_sq/deg_freedom)

print("rse :", rse)
```
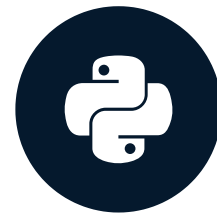
```
rse : 74.15224261594197
```

```python
residuals_sq = mdl_bream.resid ** 2

resid_sum_of_sq = sum(residuals_sq)

n_obs = len(bream.index)

rmse = np.sqrt(resid_sum_of_sq/n_obs)

print("rmse :", rmse)
```

```
rmse : 72.00244396727619
```

Let's practice!

datacamp

Visualizing model fit

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON

Maarten Van den Broeck
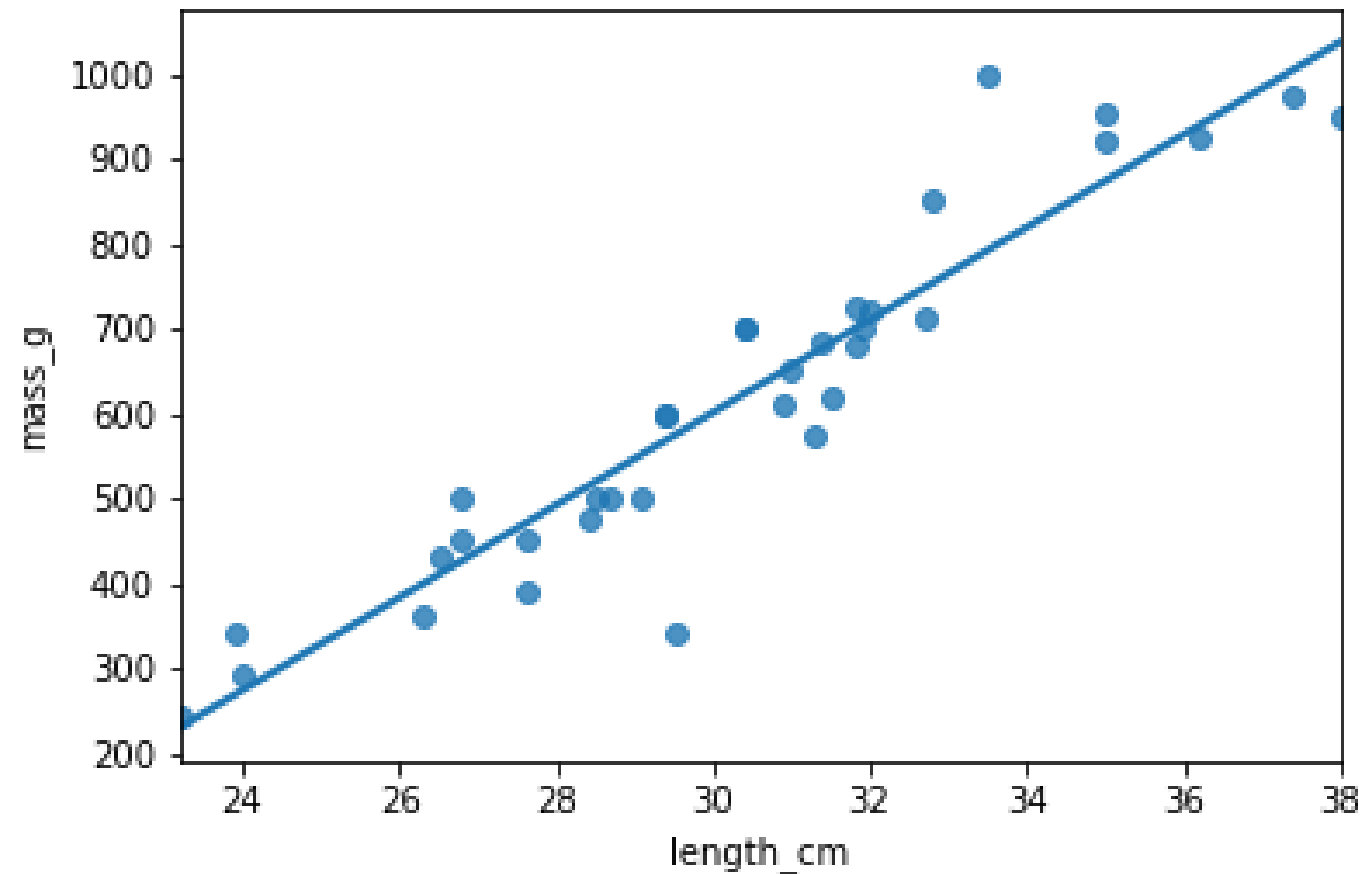Content Developer at DataCamp

Residual properties of a good fit

- Residuals are normally distributed

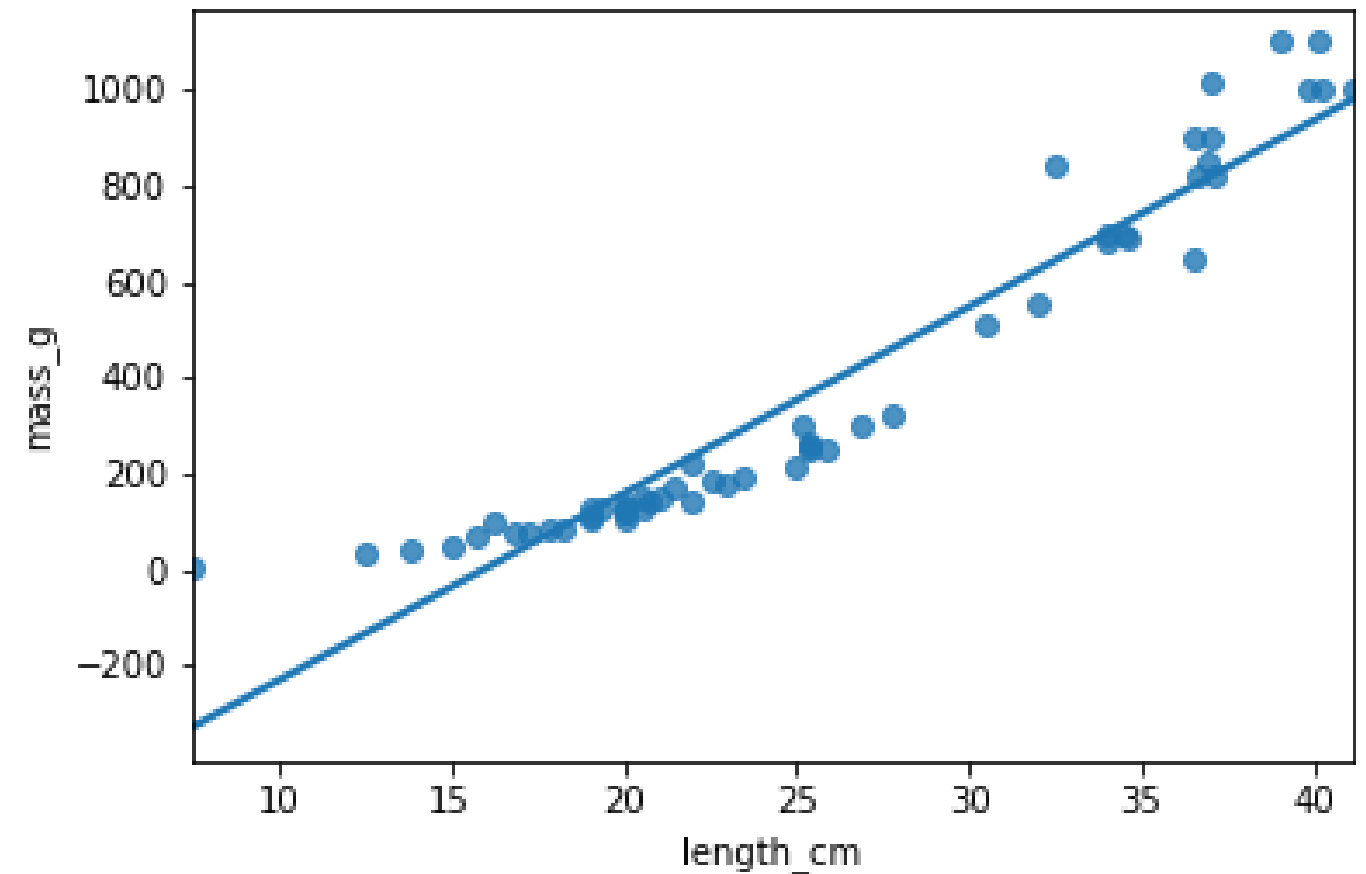- The mean of the residuals is zero

Bream and perch again

Bream: the "good" model

```
mdl_bream = ols("mass_g ~ length_cm", data=bream).fit()
```
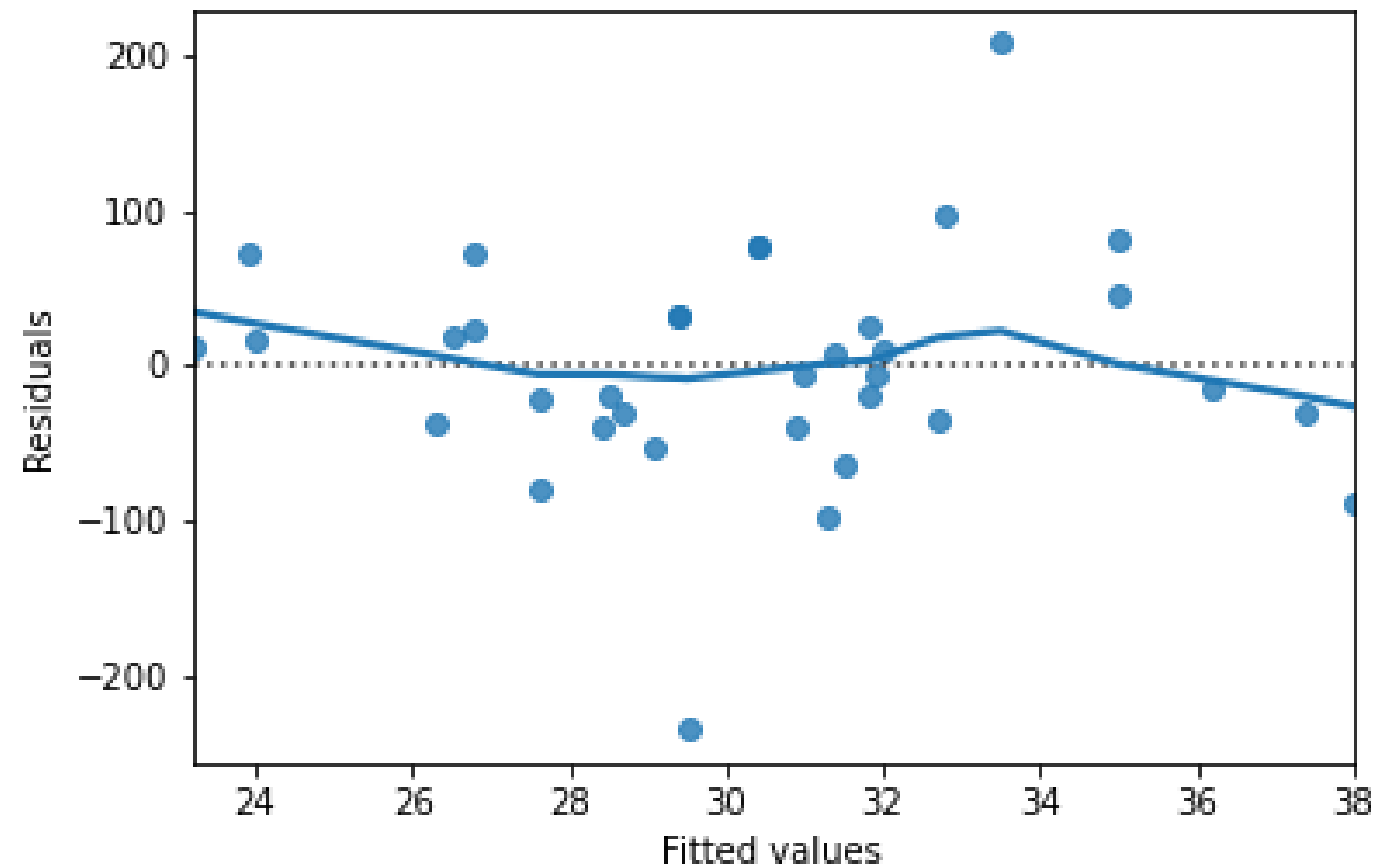
Perch: the "bad" model

```
mdl_perch = ols("mass_g ~ length_cm", data=perch).fit()
```
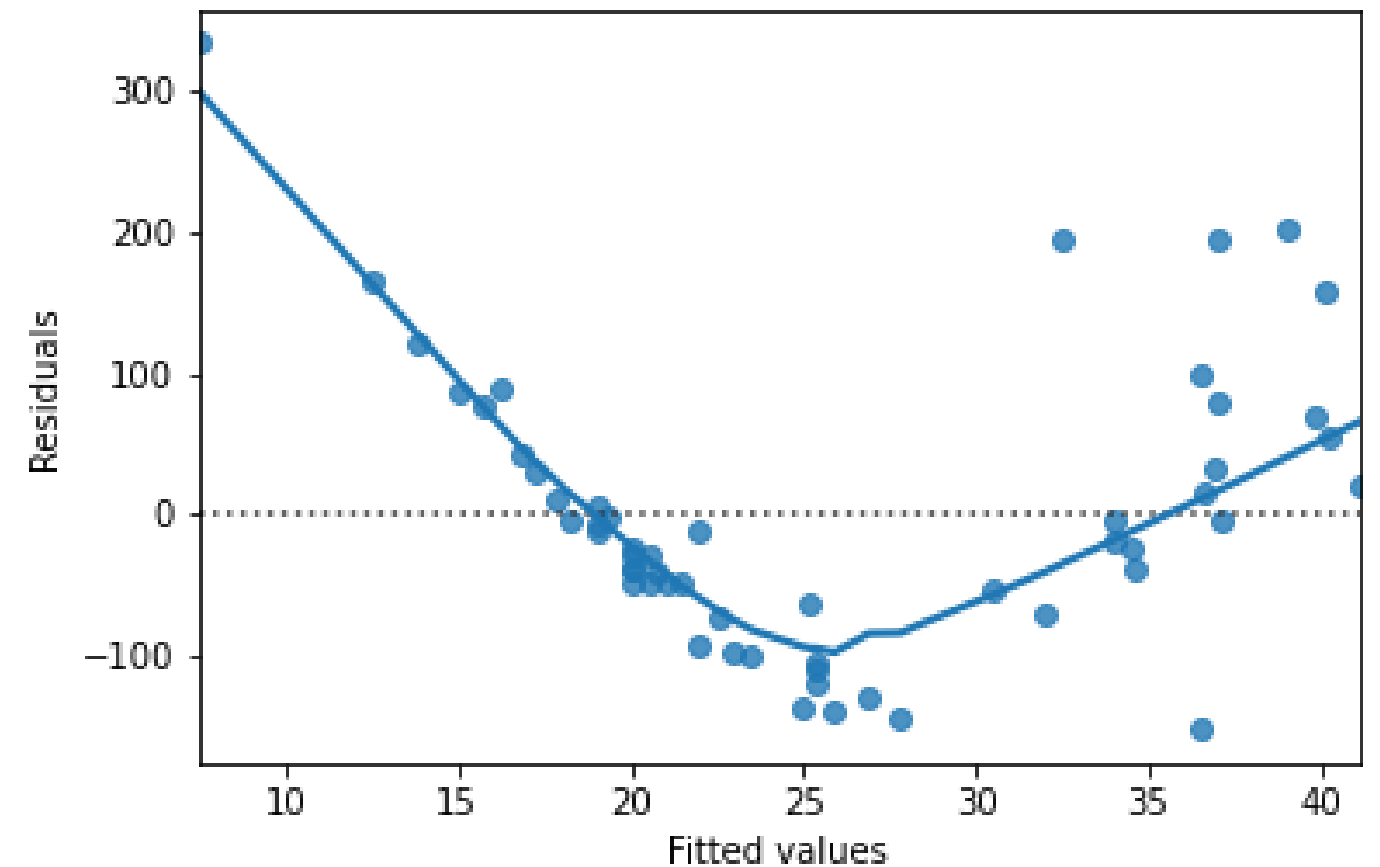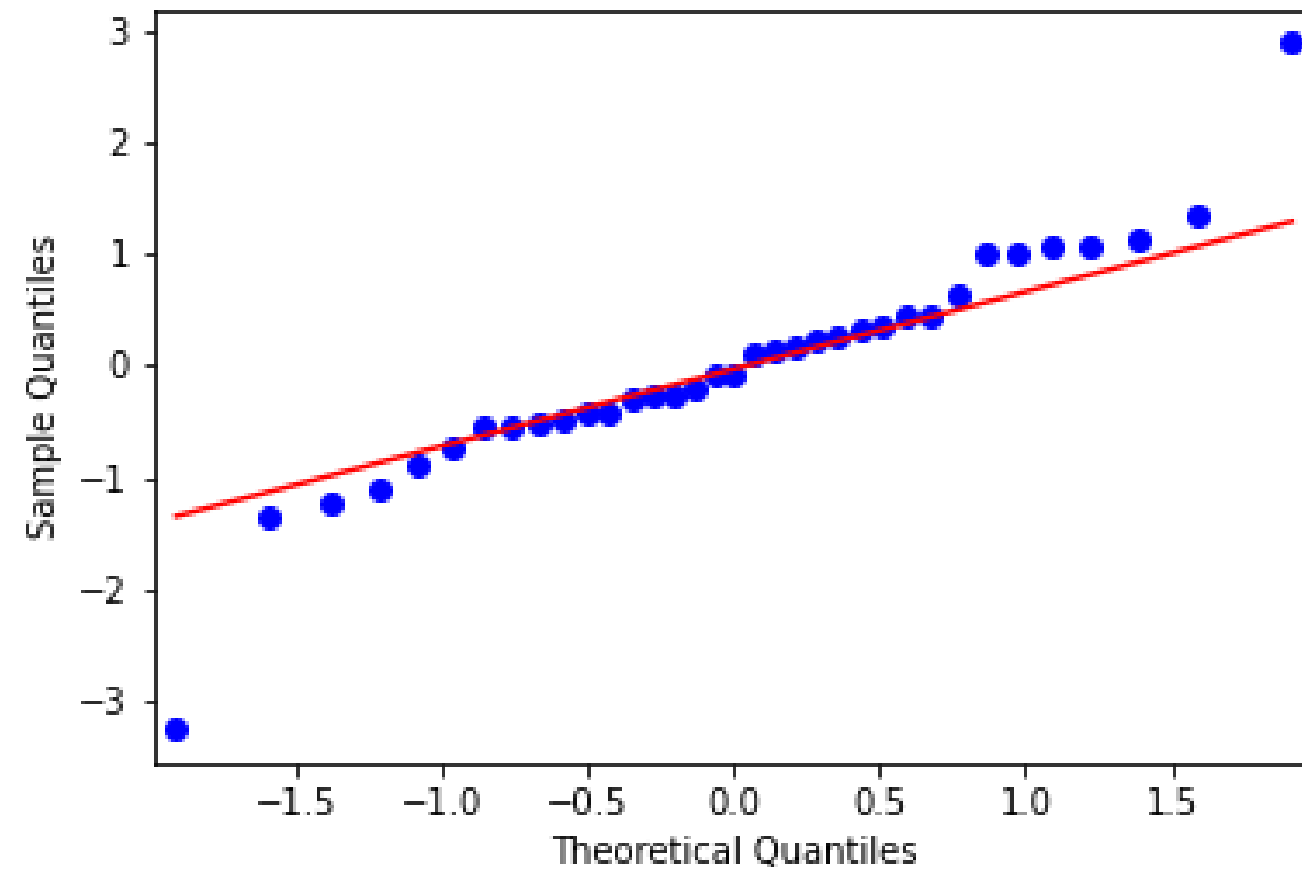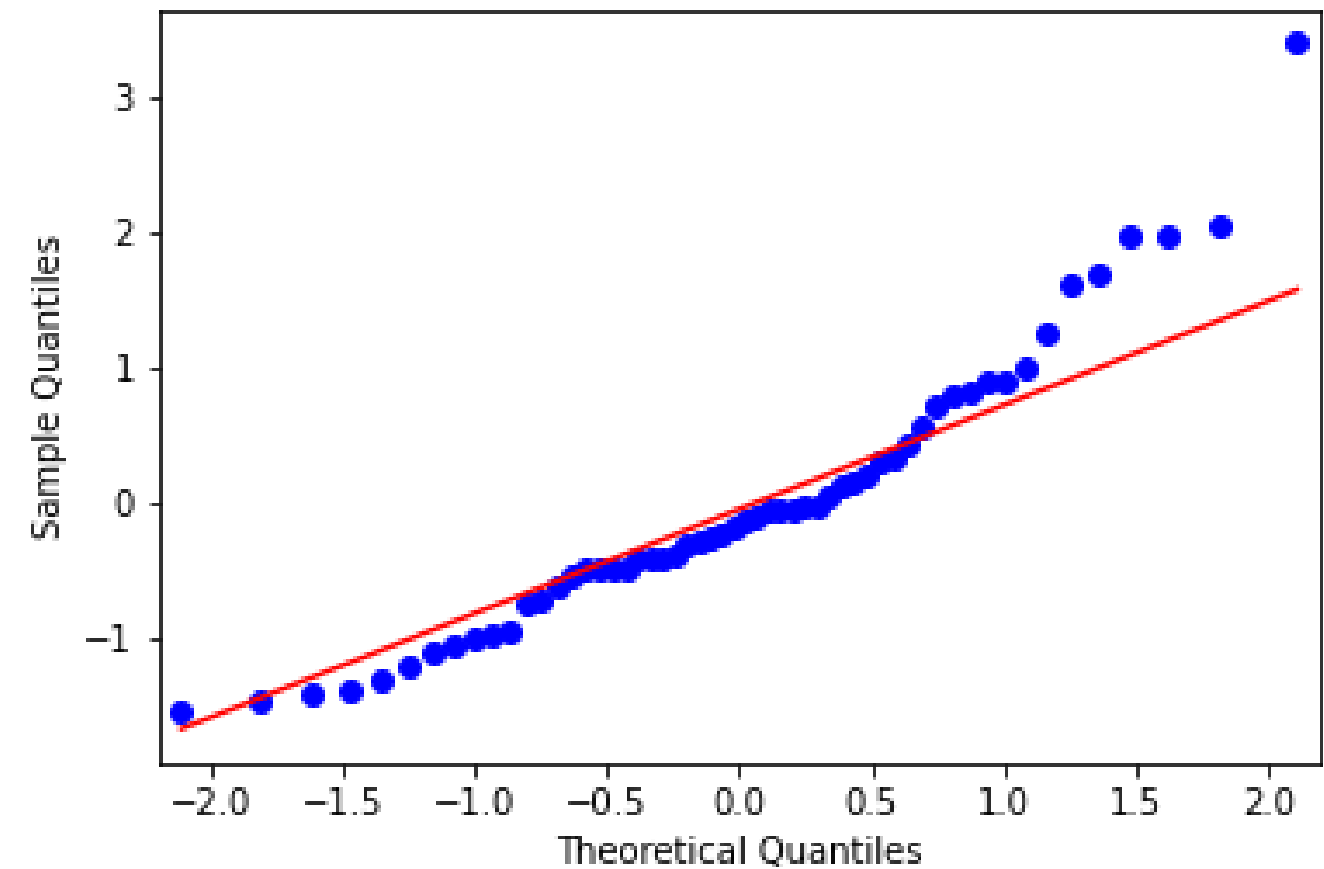
Residuals vs. fitted
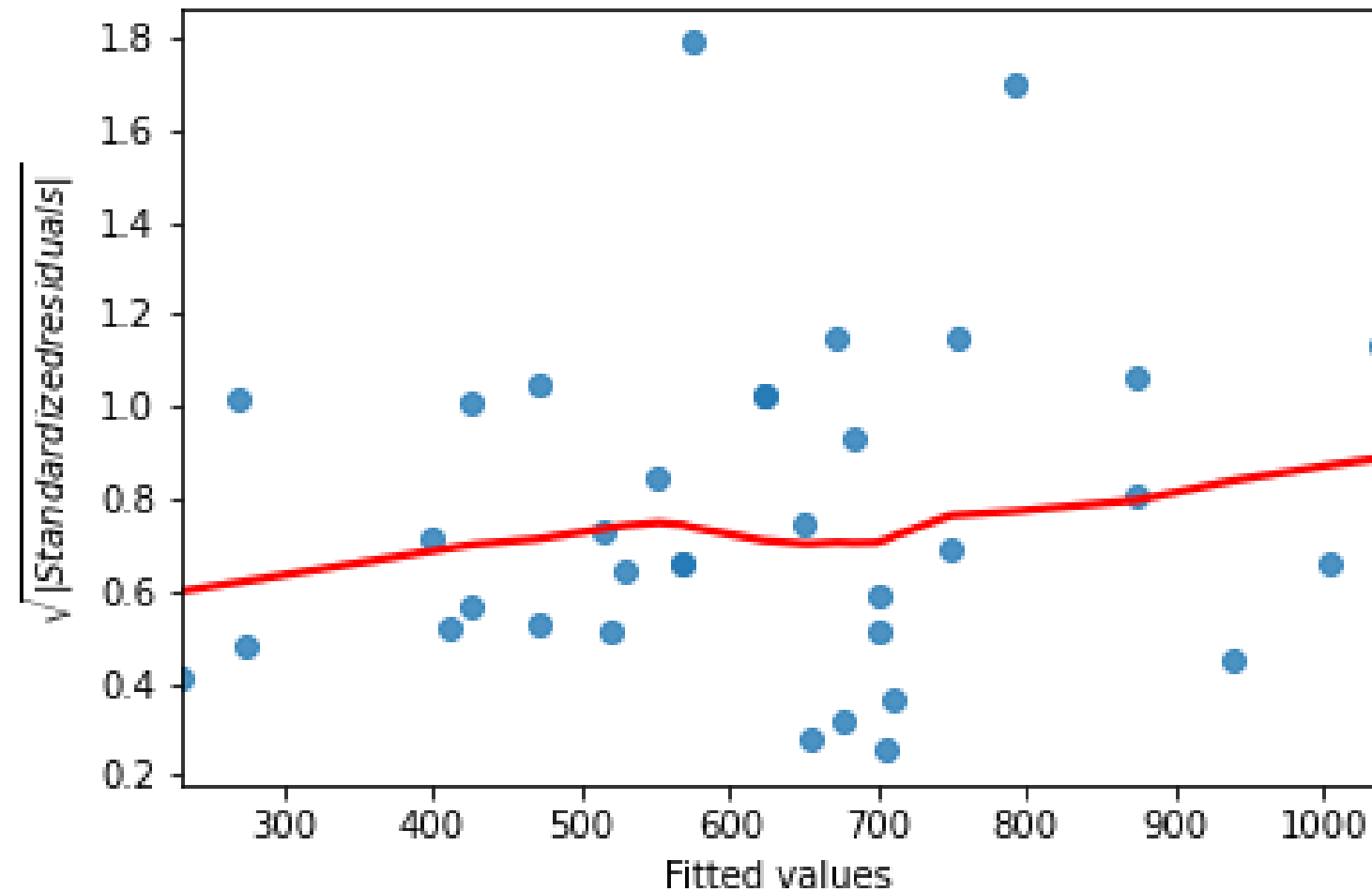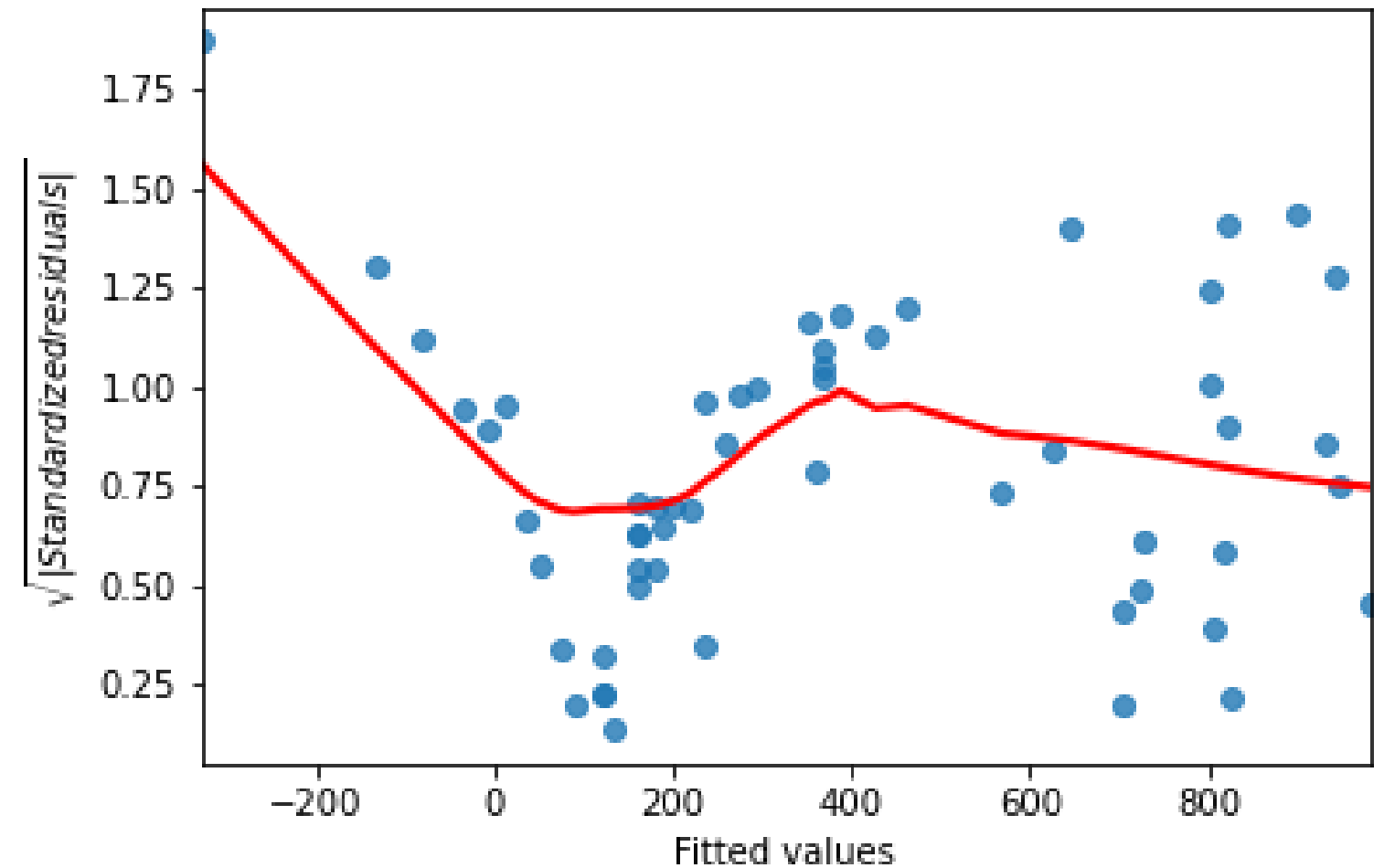
Bream

Perch

Q-Q plot

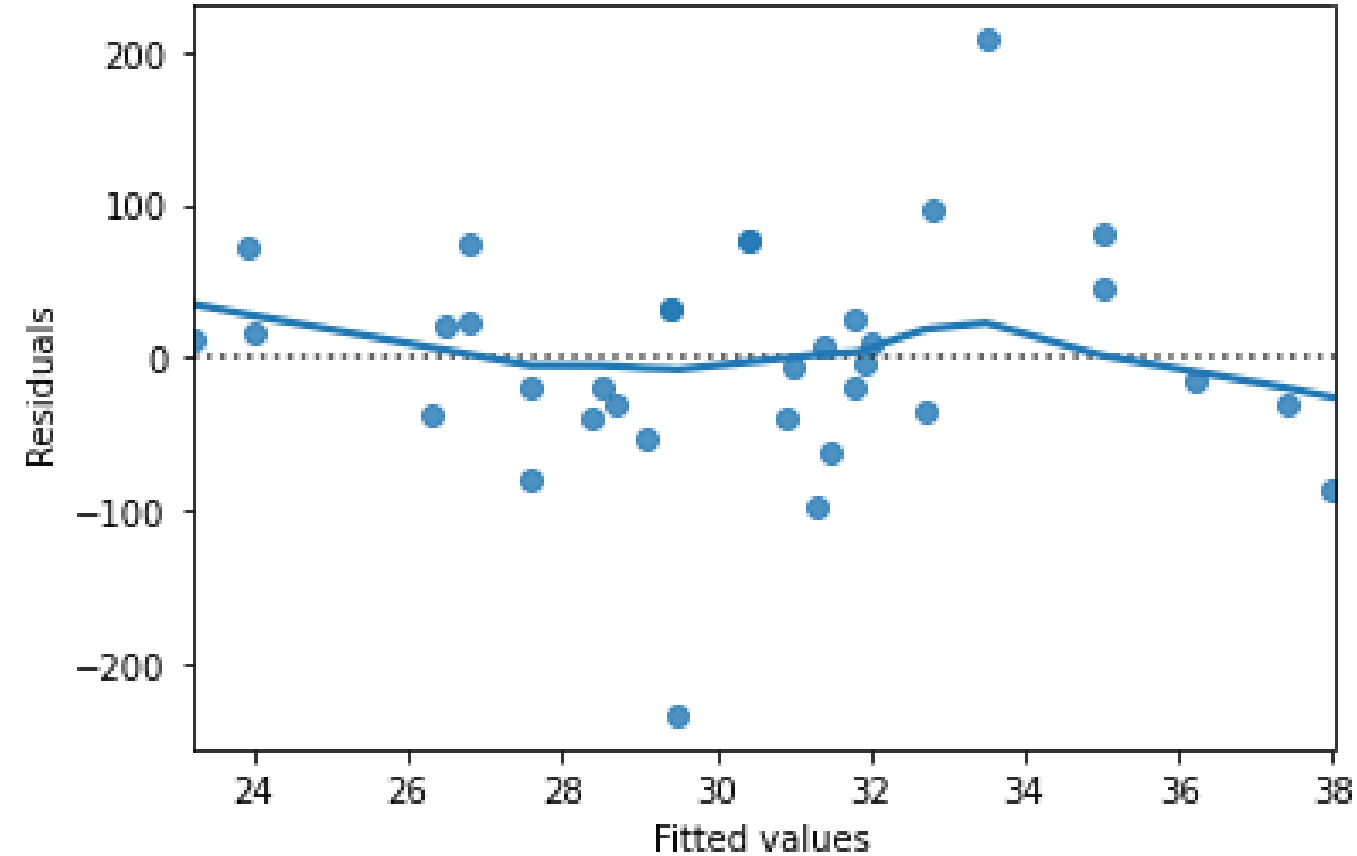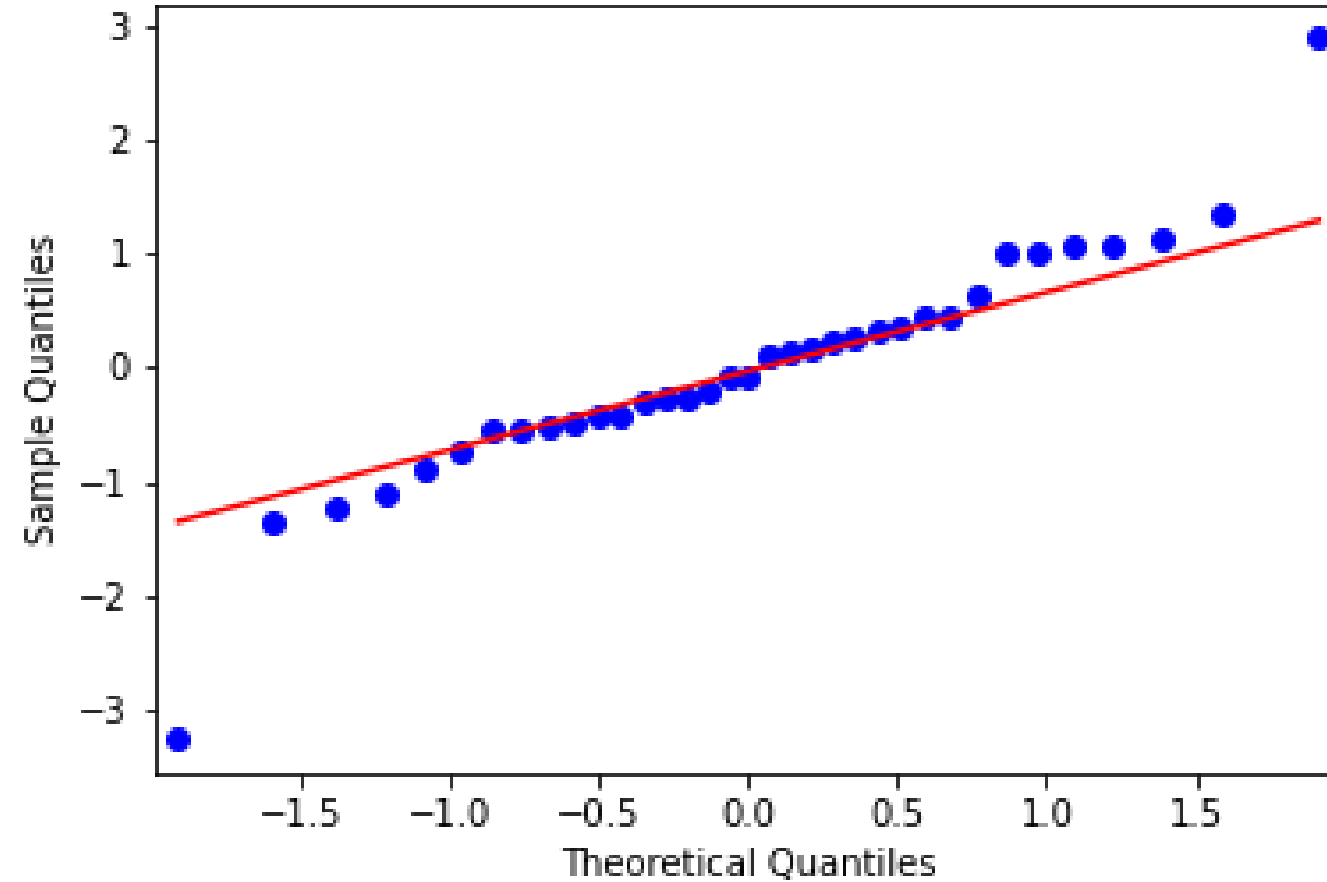Bream

Perch

Scale-location plot

Bream

Perch

residplot()

```python
sns.residplot(x="length_cm", y="mass_g", data=bream, lowess=True)
plt.xlabel("Fitted values")
plt.ylabel("Residuals")
```

qqplot()

```python
from statsmodels.api import qqplot
qqplot(data=mdl_bream.resid, fit=True, line="45")
```

Scale-location plot

```python
model_norm_residuals_bream = mdl_bream.get_influence().resid_studentized_internal
model_norm_residuals_abs_sqrt_bream = np.sqrt(np.abs(model_norm_residuals_bream))
sns.regplot(x=mdl_bream.fittedvalues, y=model_norm_residuals_abs_sqrt_bream, ci=None, lowess=True)
plt.xlabel("Fitted values")
plt.ylabel("Sqrt of abs val of stdized residuals")
```
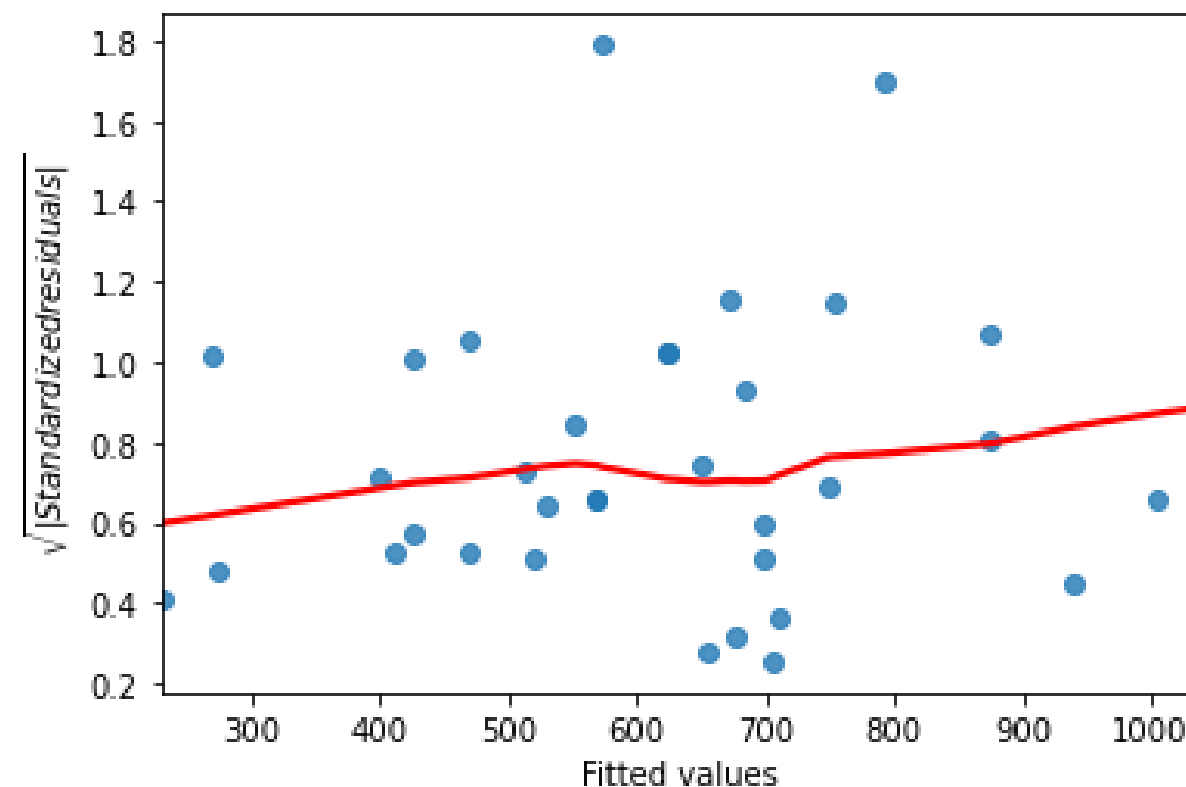
Let's practice!

Outliers, leverage, and influence

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON

Maarten Van den Broeck
Content Developer at DataCamp

Roach dataset

```python
roach = fish[fish['species'] == "Roach"]
print(roach.head())
```
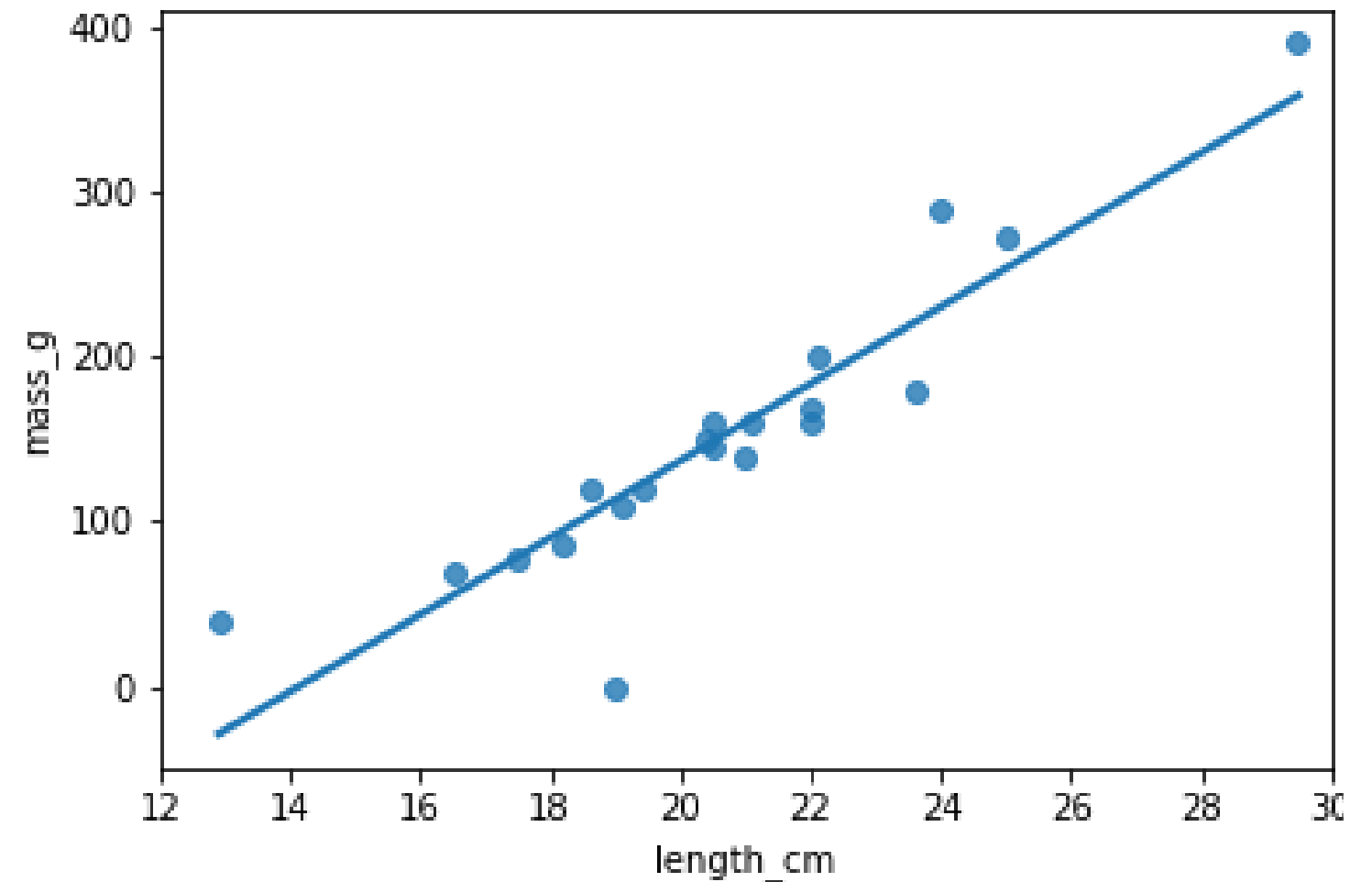
```
    species    mass_g   length_cm
35    Roach     40.0        12.9
36    Roach     69.0        16.5
37    Roach     78.0        17.5
38    Roach     87.0        18.2
39    Roach    120.0        18.6
```

Which points are outliers?

```
sns.regplot(x="length_cm",
            y="mass_g",
            data=roach,
            ci=None)
plt.show()
```

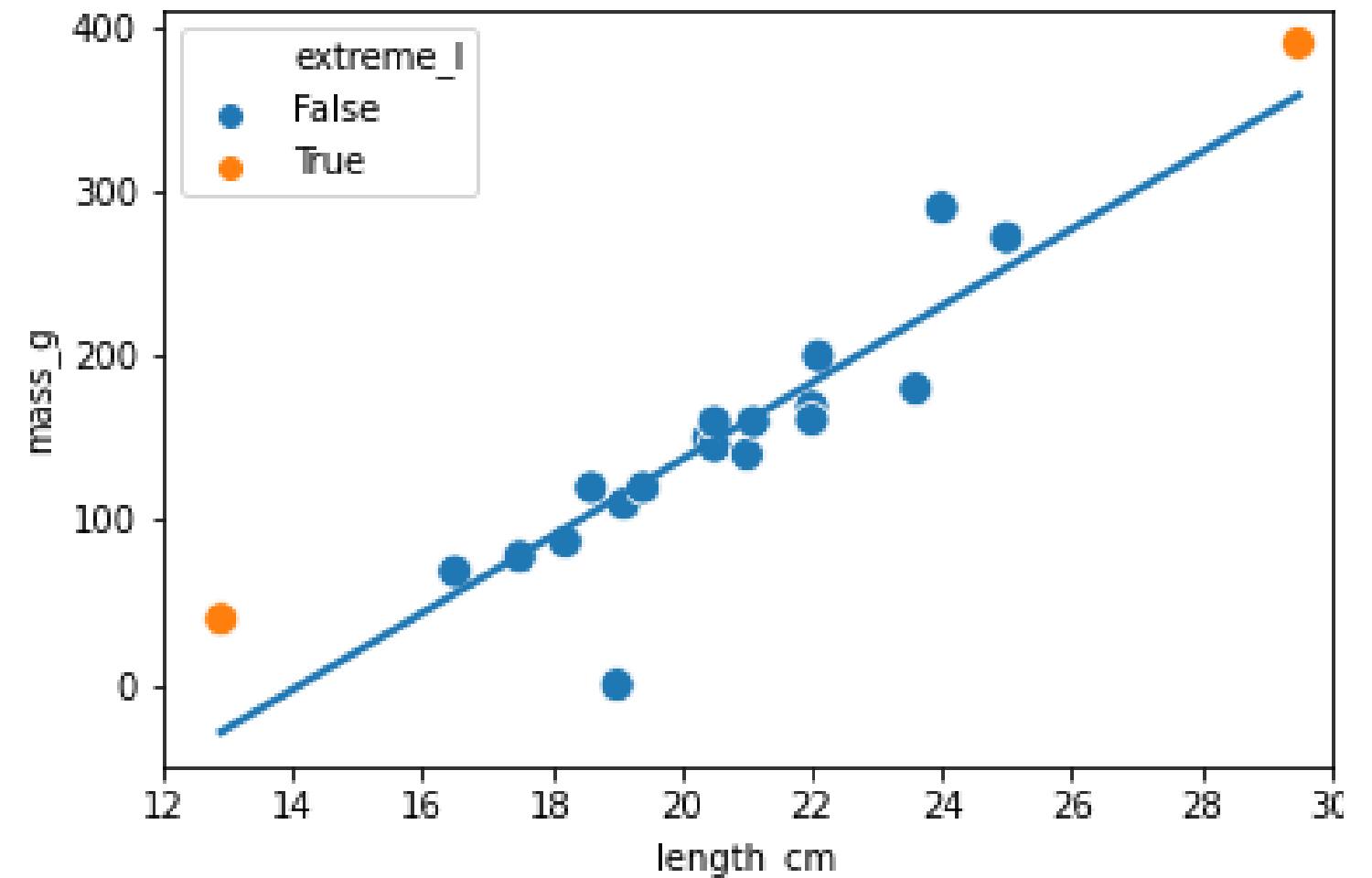# Extreme explanatory values

```python
roach["extreme_l"] = ((roach["length_cm"] < 15) |
                      (roach["length_cm"] > 26))


fig = plt.figure()
sns.regplot(x="length_cm",
            y="mass_g",
            data=roach,
            ci=None)

sns.scatterplot(x="length_cm",
                y="mass_g",
                hue="extreme_l",
                data=roach)
```

Response values away from the regression line

```python
roach["extreme_m"] = roach["mass_g"] < 1

fig = plt.figure()
sns.regplot(x="length_cm",
            y="mass_g",
            data=roach,
            ci=None)


sns.scatterplot(x="length_cm",
                y="mass_g",
                hue="extreme_l",
                style="extreme_m",
                data=roach)
```
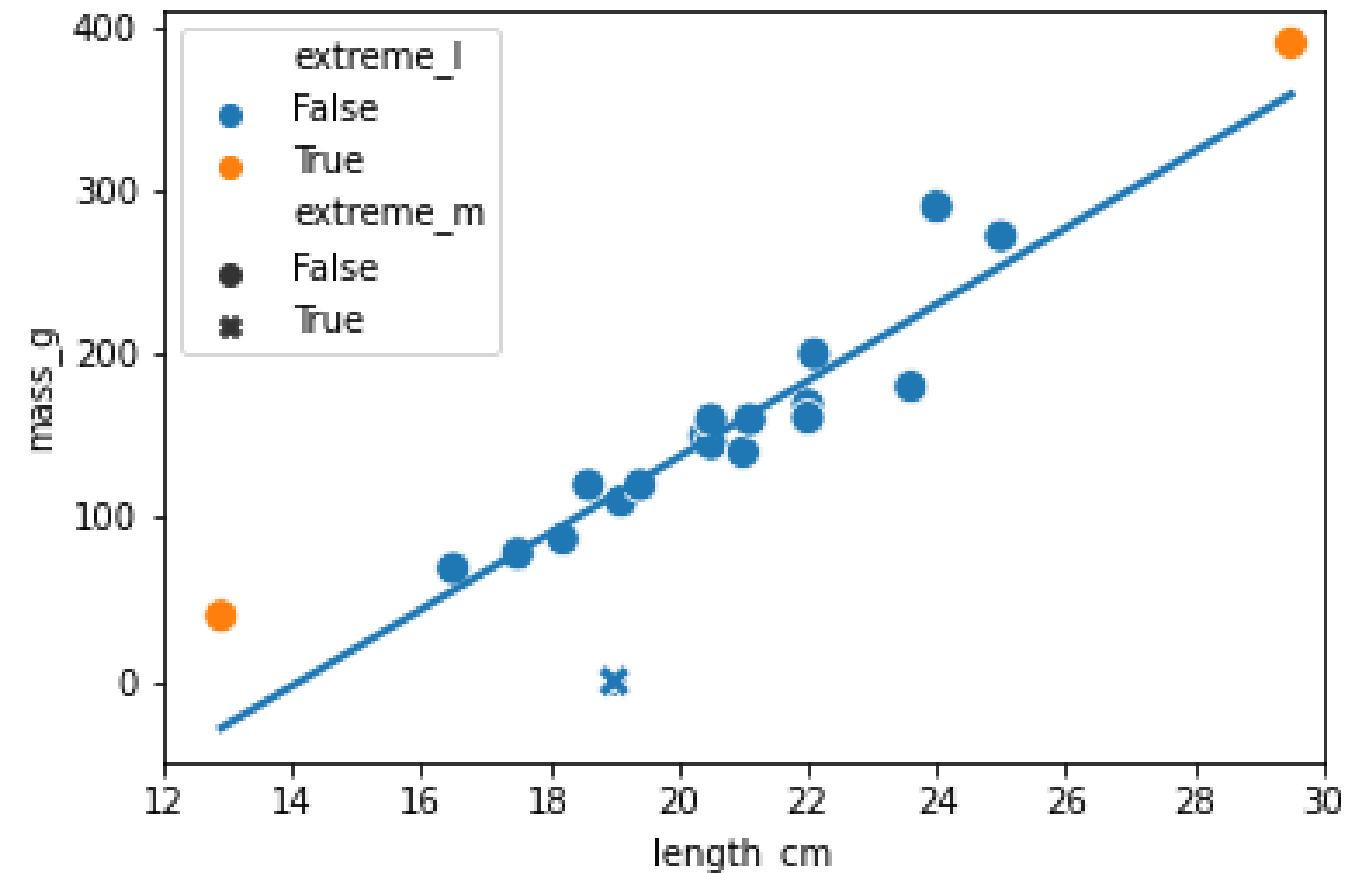
Leverage and influence

Leverage is a measure of how extreme the explanatory variable values are.

Influence measures how much the model would change if you left the observation out of the dataset when modeling.

.get_influence() and .summary_frame()

```python
mdl_roach = ols("mass_g ~ length_cm", data=roach).fit()
summary_roach = mdl_roach.get_influence().summary_frame()
roach["leverage"] = summary_roach["hat_diag"]


print(roach.head())
```

```
    species  mass_g  length_cm  leverage
35    Roach    40.0       12.9  0.313729
36    Roach    69.0       16.5  0.125538
37    Roach    78.0       17.5  0.093487
38    Roach    87.0       18.2  0.076283
39    Roach   120.0       18.6  0.068387
```

Cook's distance

Cook's distance is the most common measure of influence.

```python
roach["cooks_dist"] = summary_roach["cooks_d"]
print(roach.head())
```

```
    species  mass_g  length_cm  leverage  cooks_dist
35    Roach    40.0       12.9  0.313729    1.074015
36    Roach    69.0       16.5  0.125538    0.010429
37    Roach    78.0       17.5  0.093487    0.000020
38    Roach    87.0       18.2  0.076283    0.001980
39    Roach   120.0       18.6  0.068387    0.006610
```

Most influential roaches

```
print(roach.sort_values("cooks_dist", ascending = False))
```

```
    species  mass_g  length_cm  leverage  cooks_dist
35    Roach    40.0       12.9  0.313729    1.074015 # really short roach
54    Roach   390.0       29.5  0.394740    0.365782 # really long roach
40    Roach     0.0       19.0  0.061897    0.311852 # roach with zero mass
52    Roach   290.0       24.0  0.099488    0.150064
51    Roach   180.0       23.6  0.088391    0.061209
..      ...     ...        ...       ...         ...
43    Roach   150.0       20.4  0.050264    0.000257
44    Roach   145.0       20.5  0.050092    0.000256
42    Roach   120.0       19.4  0.056815    0.000199
47    Roach   160.0       21.1  0.050910    0.000137
37    Roach    78.0       17.5  0.093487    0.000020
```

Removing the most influential roach

```python
roach_not_short = roach[roach["length_cm"] != 12.9]

sns.regplot(x="length_cm",
            y="mass_g",
            data=roach,
            ci=None,
            line_kws={"color": "green"})


sns.regplot(x="length_cm",
            y="mass_g",
            data=roach_not_short,
            ci=None,
            line_kws={"color": "red"})
```
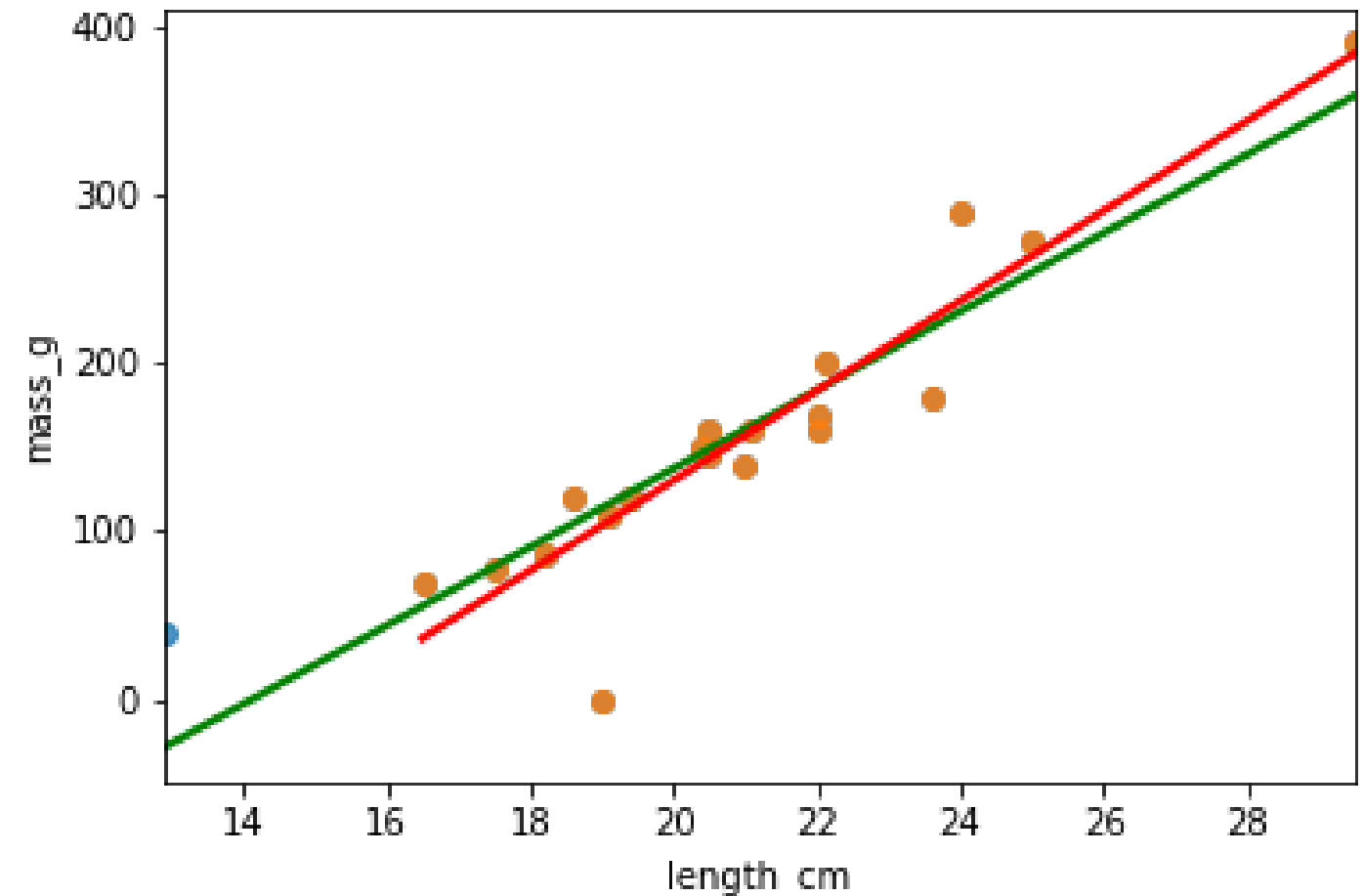
Let's practice!

INTRODUCTION TO REGRESSION WITH STATSMODELS IN PYTHON