Analytical Computing

Lecture 5: StatsModels for Python statistics

Radboud University

Agenda

- Statistische analyse
- StatsModels introductie
- Regressie
- Terug naar StatsModels
- Meer plots met StatsModels

Planning

20-4-2021 **Git**

22-4-2021 **Array Handling**

28-4-2021 **Pandas data manipulation**

30-4-2021 **Kick-off opdracht**

11-5-2021 **Visualization with MatplotLib**

12-5-2021 **StatsModels for Python statistics**

21-5-2021 **Sklearn for regression learning**

4-6-2021 **Eindpresentaties**

Statistische analyse

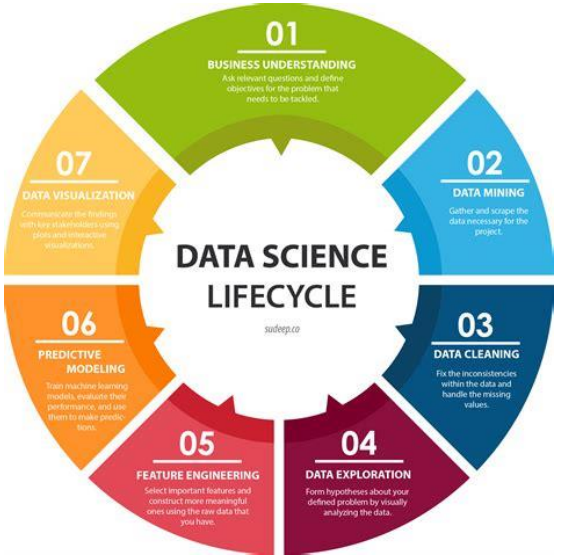
Waarom statistische analyses?

- Statistical data analysis:
  - collection and processing of a data source in an effort to identify trends within the data
- Difference between data analysis and statistical analysis
- Data analysis = check whether data works for our problem
- Statistical analysis = check if the data makes sense and draw inferences from it

Statistische analyses

- Key points for executing statistical analysis

1) Decide your goal and objective as soon as possible
2) Identify key latent variables (= variables that have the most effect on the potential outcome)
3) Conduct thourough data collection
4) Clean your data
5) Keep modeling your data
6) Optimize and repeat



StatsModels introductie

StatsModels

- Python module that provides classes and functions for the estimation of many different statistical models
- Conducting statistical tests and statistical data exploration
- Current version: 0.12.2
- Supports R-style programming formula styling and Pandas DataFrames
- Originally a model for `scipy.stats`

StatsModels

- As usual, import StatsModels by typing the following:
  - `import statsmodels.api as sm`
  - `import statsmodels.formula.api as smf`
- Why do we load in the API (Application Programming Interface) and not import statsmodels directly?
  - The API part of StatsModels is public, and thus more easily accessible
  - We can inspect what functions can be used from the statsmodels API package

```
dir(sm)
```

```
['BayesGaussMI',
 'BinomialBayesMixedGLM',
 'Factor',
 'GEE',
 'GLM',
 'GLMGam',
 'GLS',
 'GLSAR',
 'GeneralizedPoisson',
 'Logit',
 'MANOVA',
 'MI',
 'MICE',
 'MICEData',
 'MNLogit',
 'MixedLM',
 'NegativeBinomial',
 'NegativeBinomialP',
 'NominalGEE',
 'OLS',
 'OrdinalGEE',
 'PCA',
 'PHReg',
 'Poisson',
 'PoissonBayesMixedGLM',
 'ProbPlot',
 'Probit',
 'QuantReg',
 'RLM',
 'RecursiveLS',
 'SurvfuncRight',
 'WLS',
 'ZeroInflatedGeneralizedPoisson',
 'ZeroInflatedNegativeBinomialP',
 'ZeroInflatedPoisson',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '__version__',
 'add_constant',
 'categorical',
 'cov_struct',
 'datasets',
 'distributions',
 'duration',
 'emplike',
```

StatsModels

- We could easily import distinct functions directly
- …however you would need to know the function location and with many functions your imports could get cluttered

```
sm.OLS
```

```
statsmodels.regression.linear_model.OLS
```

```
from statsmodels.regression.linear_model import OLS
```

```
OLS
```

```
statsmodels.regression.linear_model.OLS
```

Regressie

Regressie

- Simply said, we want to identify a “relationship” between to ‘things’ (variables in our data)
  - For example: when you climb a mountain, the temperature drops
  - As a result, we could conclude that height influences temperature
  - In statistics, this is termed “regression”
  - The temperature is dependent on the height, so the temperature is the ‘dependent variable’, whereas the height is the ‘independent’ variable (because it doesn’t change with the temperature)

Any equation, that is a function of the dependent variables and a set of weights is called a regression function.

Regressie

- In formal, mathematical terms: $y \sim f(x; w)$
  - y is our dependent variable (which we want to predict)
  - $f()$ is a function which takes as input a finite number of x independent variables (e.g. humidity or pressure) and w weights

- A possible equation would could be:
  - $y = 0.5 * x_1 + 2.15 * x_2 + 0.76 * x_3$
  - x_1, x_2 and x_3 are our, to be decided, independent variables
  - 0.5, 2.15 and 0.76 are our weights
  - …but how do we know what the weights are?

Regressie

- Weights in regression are *learnt* by studying the relationship between the dependent and independent variables
- We learn these weights by using **models** that can measure the relationship between the dependent and independent variables
- As a result, we can predict the dependent variable beforehand

- …but, to make predictions, we first need data

Height above sea level in meters	Pressure in hpa	Humidity in percentage	Temperature in Fahrenheit
0	1015	67	84
500	1000	60	73
700	850	40	65

Regressie

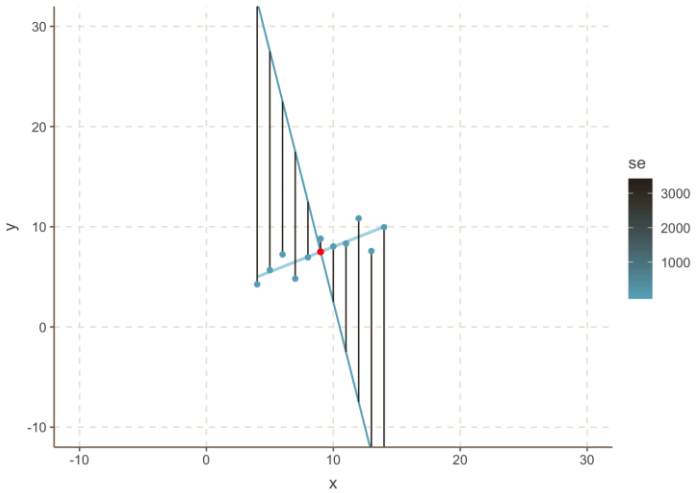
- In machine learning, your *independent* variables are **always** denoted as x and your **single** *dependent* variable (which you want to predict) is y
- As such, the temperature in Fahrenheit is our y variable, and all the others are our x variable

- From this point on, we can use different regression models to try and predict our temperature from our independent variables

Height above sea level in meters	Pressure in hpa	Humidity in percentage	Temperature in Fahrenheit
0	1015	67	84
500	1000	60	73
700	850	40	65

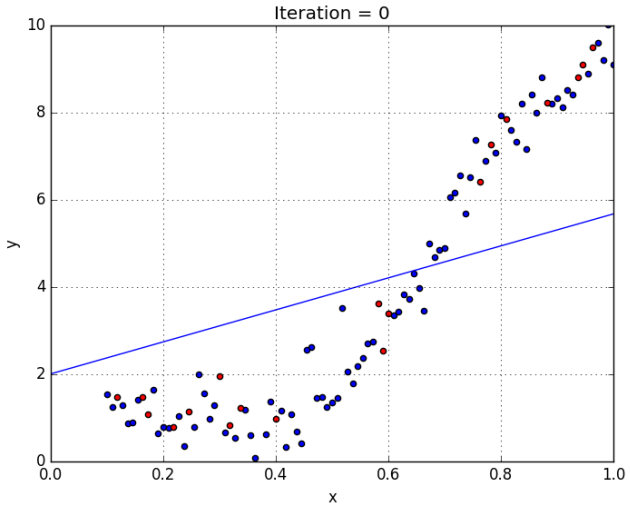
Regressie

- What do these models do?
- Depending on the model you choose (e.g. linear regression, ordinary least squares), your model will try and minimize the error it makes during training
- In summary: if you have a plot with data points, your model will try and fit a line through those points such that the margin between the line and each point is as small as possible



Regressie

- This training happens over iterations
  - Each iteration, the weights get updated according to a predefined loss function (e.g. higher errors get ‘punished’ more than lower errors)
  - There is no ‘optimum’ number of iterations



Pauze

Terug naar StatsModels

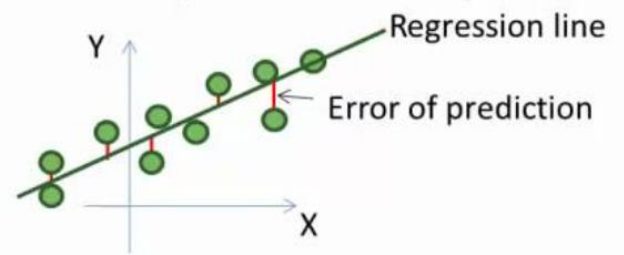
StatsModels

- We have learnt the basics of regression learning
- We have seen that regression and statistics is all about finding correlations between variables in the data
  - Specifically, correlations between dependent and independent variables
- How can we use this information in StatsModels?

Ordinary Least Squares

- Ordinary Least Squares (OLS) is a regression model which tries to minimize the error of a prediction based on the distance from the regression line to the data points

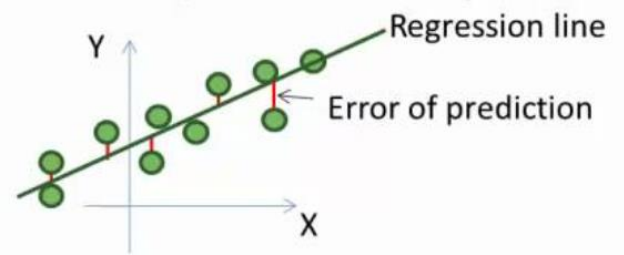
- Let's try an experiment with artificial data



Ordinary Least Squares

- Ordinary Least Squares (OLS) is a regression model which tries to minimize the error of a prediction based on the distance from the regression line to the data points

- Let's try an experiment



Ordinary Least Squares

- We use the Guerry dataset
  - 86 rows and 23 columns
  - Collection of historical data used in support of Andre-Michel Guerry's 1833 Essay on the Moral Statistics of France

```
df = sm.datasets.get_rdataset("Guerry", "HistData").data

print(df.shape)
df.head()
```

```
(86, 23)
```

	dept	Region	Department	Crime_pers	Crime_prop	Literacy	Donatio
0	1	E	Ain	28870	15890	37	50
1	2	N	Aisne	26226	5521	51	89
2	3	C	Allier	26747	7925	13	109
3	4	E	Basses-Alpes	12935	7289	46	27
4	5	E	Hautes-Alpes	17488	8174	69	69

5 rows × 23 columns

Ordinary Least Squares

- We select some variables of interest
  - Note that, in a real-life scenario, you would have to extract these ‘meaningful’ independent variables from your own research

```
vars = ['Department', 'Lottery', 'Literacy', 'Wealth', 'Region']
df = df[vars]

df.head()
```

	Department	Lottery	Literacy	Wealth	Region
0	Ain	41	37	73	E
1	Aisne	38	51	22	N
2	Allier	66	13	61	C
3	Basses-Alpes	80	46	76	E
4	Hautes-Alpes	79	69	83	E

Ordinary Least Squares

- Next, we need to create a ‘design matrix’

- This design matrix must be saved in two variables: X and y
- Tells StatsModels which variable is dependent, and which ones are independent
  - Shown by `Lottery ~ Literacy + Wealth + Region`
  - In this case, Lottery is our dependent variable and Literacy, Wealth and Region are our independent variables
  - Note that ‘Region’ was categorical and is now split into multiple independent variables

```
y, X = dmatrices('Lottery ~ Literacy + Wealth + Region', data=df, return_type='dataframe')

print(X)
print("-"*100)
print(y)
```

```
    Intercept  Region[T.E]  Region[T.N]  Region[T.S]  Region[T.W]  Literacy  \
0         1.0          1.0          0.0          0.0          0.0      37.0
1         1.0          0.0          1.0          0.0          0.0      51.0
2         1.0          0.0          0.0          0.0          0.0      13.0
3         1.0          1.0          0.0          0.0          0.0      46.0
4         1.0          1.0          0.0          0.0          0.0      69.0
..        ...          ...          ...          ...          ...       ...
80        1.0          0.0          0.0          0.0          1.0      28.0
81        1.0          0.0          0.0          0.0          1.0      25.0
82        1.0          0.0          0.0          0.0          0.0      13.0
83        1.0          1.0          0.0          0.0          0.0      62.0
84        1.0          0.0          0.0          0.0          0.0      47.0

    Wealth
0     73.0
1     22.0
2     61.0
3     76.0
4     83.0
..     ...
80    56.0
81    68.0
82    67.0
83    82.0
84    30.0

[85 rows x 7 columns]
----------------------------------------------------------------------------------------------------
    Lottery
0      41.0
1      38.0
2      66.0
3      80.0
4      79.0
..      ...
80     68.0
81     40.0
82     55.0
83     14.0
84     51.0

[85 rows x 1 columns]
```

Ordinary Least Squares

- Now that we have our data ready we can call and fit our model
- `sm.OLS(y,X)`
  - Tells StatsModels that we want to use the Ordinary Least Squares (OLS) model
- `model.fit()`
  - Fit the model with the data specified in OLS()
- `results.summary()`
  - Print the results of the fit

```
model = sm.OLS(y, X)    # Describe model
result = model.fit()      # Fit model
print(result.summary())   # Summarize model
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Lottery   R-squared:                       0.338
Model:                            OLS   Adj. R-squared:                  0.287
Method:                 Least Squares   F-statistic:                     6.636
Date:                Wed, 12 May 2021   Prob (F-statistic):           1.07e-05
Time:                        12:06:49   Log-Likelihood:                -375.30
No. Observations:                  85   AIC:                             764.6
Df Residuals:                      78   BIC:                             781.7
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      38.6517      9.456      4.087      0.000      19.826      57.478
Region[T.E]   -15.4278      9.727     -1.586      0.117     -34.793       3.938
Region[T.N]   -10.0170      9.260     -1.082      0.283     -28.453       8.419
Region[T.S]    -4.5483      7.279     -0.625      0.534     -19.039       9.943
Region[T.W]   -10.0913      7.196     -1.402      0.165     -24.418       4.235
Literacy       -0.1858      0.210     -0.886      0.378      -0.603       0.232
Wealth          0.4515      0.103      4.390      0.000       0.247       0.656
==============================================================================
Omnibus:                        3.049   Durbin-Watson:                   1.785
Prob(Omnibus):                  0.218   Jarque-Bera (JB):                2.694
Skew:                          -0.340   Prob(JB):                        0.260
Kurtosis:                       2.454   Cond. No.                         371.
==============================================================================

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

Ordinary Least Squares

- The results may look intimidating
- …but some statistics are useful and worth noting

- For example:
- **R-squared** shows the 'strength' or 'effect' that the independent variables have on the dependent variable
  - R-squared is a value between 0% and 100%, or, between 0 and 1
  - 0 suggests that the dependent variable can definately not be explained by the present independent variables
  - 1 suggests that all of the independent variables explain the movement of the dependent variable (there is a trend)

```
model = sm.OLS(y, X)    # Describe model
result = model.fit()      # Fit model
print(result.summary())   # Summarize model
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Lottery   R-squared:                       0.338
Model:                            OLS   Adj. R-squared:                  0.287
Method:                 Least Squares   F-statistic:                     6.636
Date:                Wed, 12 May 2021   Prob (F-statistic):           1.07e-05
Time:                        12:06:49   Log-Likelihood:                -375.30
No. Observations:                  85   AIC:                             764.6
Df Residuals:                      78   BIC:                             781.7
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      38.6517      9.456      4.087      0.000      19.826      57.478
Region[T.E]   -15.4278      9.727     -1.586      0.117     -34.793       3.938
Region[T.N]   -10.0170      9.260     -1.082      0.283     -28.453       8.419
Region[T.S]    -4.5483      7.279     -0.625      0.534     -19.039       9.943
Region[T.W]   -10.0913      7.196     -1.402      0.165     -24.418       4.235
Literacy       -0.1858      0.210     -0.886      0.378      -0.603       0.232
Wealth          0.4515      0.103      4.390      0.000       0.247       0.656
==============================================================================
Omnibus:                        3.049   Durbin-Watson:                   1.785
Prob(Omnibus):                  0.218   Jarque-Bera (JB):                2.694
Skew:                          -0.340   Prob(JB):                        0.260
Kurtosis:                       2.454   Cond. No.                         371.
==============================================================================

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

Ordinary Least Squares

- **Parameter coefficients**
  - "Parameter estimates (also called coefficients) are the change in the response associated with a one-unit change of the predictor, all other predictors being held constant."
  - Can be either positive or negative
  - Describes the size of the contribution of that independent variable to the prediction of the dependent variable
  - Near-zero coefficient indicates that variable has little influence on the response
  - Large positive or negative coefficients denote a large positive or negative correlation between that variable and the dependent variable

```
model = sm.OLS(y, X)    # Describe model
result = model.fit()      # Fit model
print(result.summary())   # Summarize model
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Lottery   R-squared:                       0.338
Model:                            OLS   Adj. R-squared:                  0.287
Method:                 Least Squares   F-statistic:                     6.636
Date:                Wed, 12 May 2021   Prob (F-statistic):           1.07e-05
Time:                        12:06:49   Log-Likelihood:                -375.30
No. Observations:                  85   AIC:                             764.6
Df Residuals:                      78   BIC:                             781.7
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      38.6517      9.456      4.087      0.000      19.826      57.478
Region[T.E]   -15.4278      9.727     -1.586      0.117     -34.793       3.938
Region[T.N]   -10.0170      9.260     -1.082      0.283     -28.453       8.419
Region[T.S]    -4.5483      7.279     -0.625      0.534     -19.039       9.943
Region[T.W]   -10.0913      7.196     -1.402      0.165     -24.418       4.235
Literacy       -0.1858      0.210     -0.886      0.378      -0.603       0.232
Wealth          0.4515      0.103      4.390      0.000       0.247       0.656
==============================================================================
Omnibus:                        3.049   Durbin-Watson:                   1.785
Prob(Omnibus):                  0.218   Jarque-Bera (JB):                2.694
Skew:                          -0.340   Prob(JB):                        0.260
Kurtosis:                       2.454   Cond. No.                         371.
==============================================================================

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

Ordinary Least Squares

- Finally, we can use the `predict()` function to show new predictions of the Ordinary Least Squares model

```
print('Predicted values: ', result.predict())
```

```
Predicted values:  [49.30622039 29.09035627 63.77597091 48.98827119 47.8747511  67.01016548
 31.08347079 63.26484093 18.58117879 35.46061463 50.91674297 27.94521448
 23.48683863 59.52827361 60.69772238 29.43997391 64.67892063 69.83096341
 19.2983092  57.19051079 69.59285045 52.30412357 23.65478109 45.21679982
 28.18740655 33.58363617 42.02618483 38.40897252 38.72692172 45.10125919
 24.73940599 31.15922792 48.74601409 59.87236964 45.8243553  47.18097598
 37.65050178 59.92690255 42.21255641 44.0982921  68.61006448 60.21870493
 34.91051816 48.60567925 25.96030491 63.84984145 39.63668063 35.09165936
 22.79725081 21.13175123 42.18585732 36.32222402 38.36075798 47.62970247
 41.9711611  54.8001631  32.91412129 25.37257432 36.07444532 39.84591669
 58.14630644 55.49338595 62.63033829 60.39771505 32.47091641 36.21624095
 17.57085023 43.41229614 39.39992028 51.58754543 15.892998   21.99888839
 20.40635084 20.48618709 38.54930737 29.93961658 51.60633079 42.55068675
 49.24297342 56.12247516 48.64003103 54.61518729 66.48482007 48.7240114
 43.46239369]
```

Ordinary Least Squares

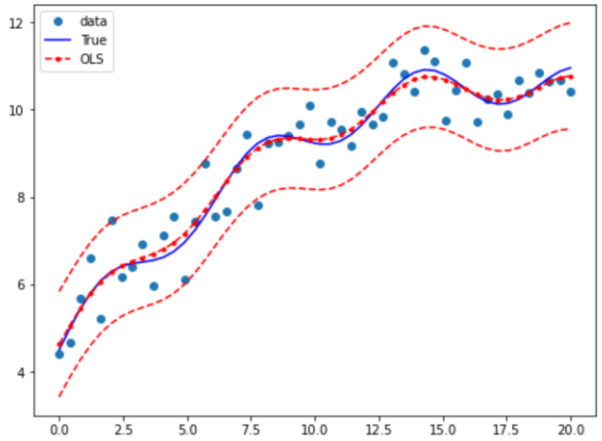
- When you have the ‘true’ data labels, you can check if your predictions are correct
- The dots show the data points of the variables in our data set we used for training
- The blue line shows the distribution of the labels that *should have* been predicted
- The red line in the center shows the distribution of labels that *were* predicted
- What you want is that blue line = red line

```
prstd, iv_l, iv_u = wls_prediction_std(res)

fig, ax = plt.subplots(figsize=(8,6))

ax.plot(x, y, 'o', label="data")
ax.plot(x, y_true, 'b-', label="True")
ax.plot(x, res.fittedvalues, 'r--.', label="OLS")
ax.plot(x, iv_u, 'r--')
ax.plot(x, iv_l, 'r--')
ax.legend(loc='best')
```

```
<matplotlib.legend.Legend at 0x24ef7487430>
```



Meer plots met StatsModels

Regressie plots

- Statewide Crime 2009 Dataset
- 51 rows, 7 columns
- Load data and make OLS summary

Question: how well do the independent variables describe the dependent variable?

```
dta = sm.datasets.statecrime.load_pandas().data

print(dta.shape)
dta.head()
```

```
(51, 7)
```

state	violent	murder	hs_grad	poverty	single	white	urban
Alabama	459.9	7.1	82.1	17.5	29.0	70.0	48.65
Alaska	632.6	3.2	91.4	9.0	25.5	68.3	44.46
Arizona	423.2	5.5	84.2	16.5	25.7	80.0	80.07
Arkansas	530.3	6.3	82.4	18.8	26.3	78.4	39.54
California	473.4	5.4	80.6	14.2	27.8	62.7	89.73

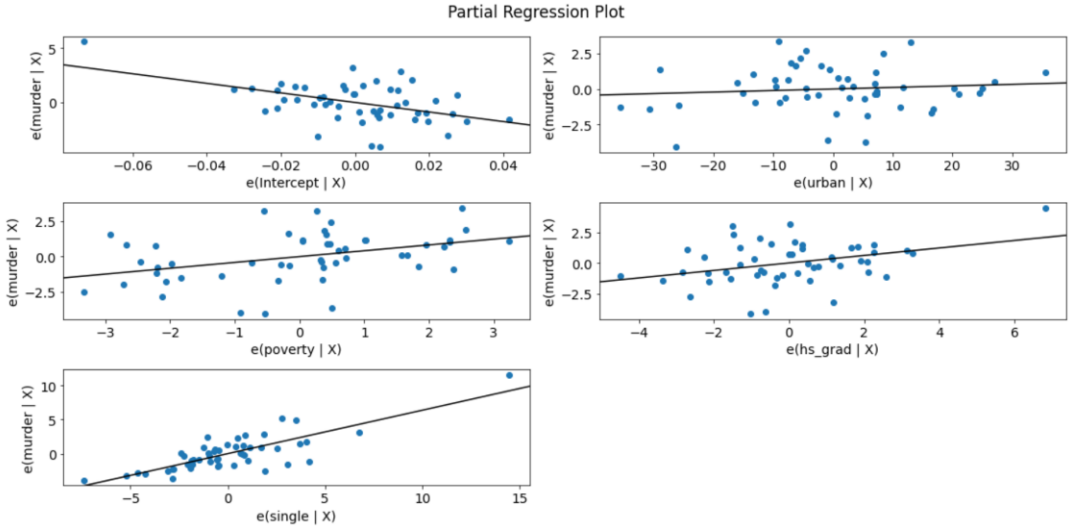
```
crime_model = ols("murder ~ urban + poverty + hs_grad + single", data=dta).fit()
print(crime_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 murder   R-squared:                       0.813
Model:                            OLS   Adj. R-squared:                  0.797
Method:                 Least Squares   F-statistic:                     50.08
Date:                Wed, 12 May 2021   Prob (F-statistic):           3.42e-16
Time:                        12:29:41   Log-Likelihood:                -95.050
No. Observations:                  51   AIC:                             200.1
Df Residuals:                      46   BIC:                             209.8
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -44.1024     12.086     -3.649      0.001     -68.430     -19.774
urban          0.0109      0.015      0.707      0.483      -0.020       0.042
poverty        0.4121      0.140      2.939      0.005       0.130       0.694
hs_grad        0.3059      0.117      2.611      0.012       0.070       0.542
single         0.6374      0.070      9.065      0.000       0.496       0.779
==============================================================================
Omnibus:                        1.618   Durbin-Watson:                   2.507
Prob(Omnibus):                  0.445   Jarque-Bera (JB):                0.831
Skew:                          -0.220   Prob(JB):                        0.660
Kurtosis:                       3.445   Cond. No.                     5.80e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.8e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

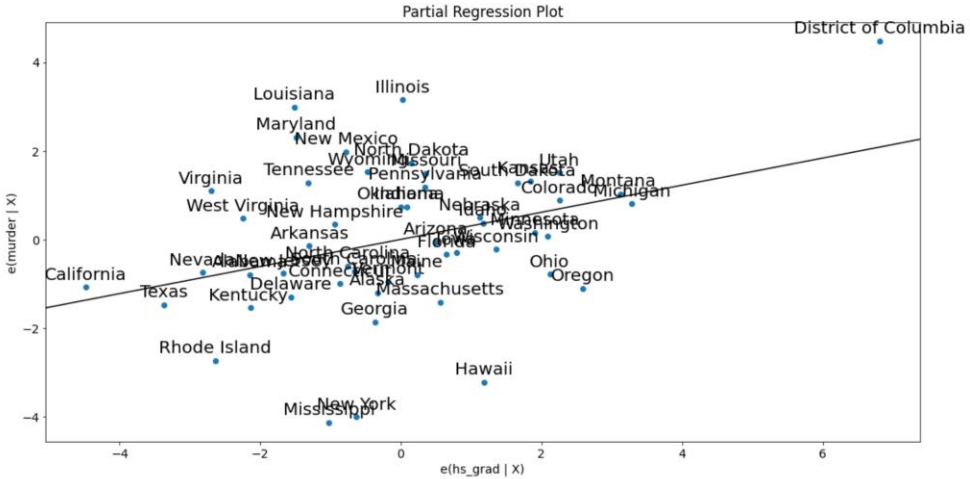
Regressie plots

- We can plot dataset statistics directly from StatsModels
- Using this we can see, for example, data distribution, correlations, trends, etc.



Regressie plots

```
fig = sm.graphics.plot_partregress("murder", "hs_grad", ["urban", "poverty", "single"], data=dta)
fig.tight_layout(pad=1.0)
```



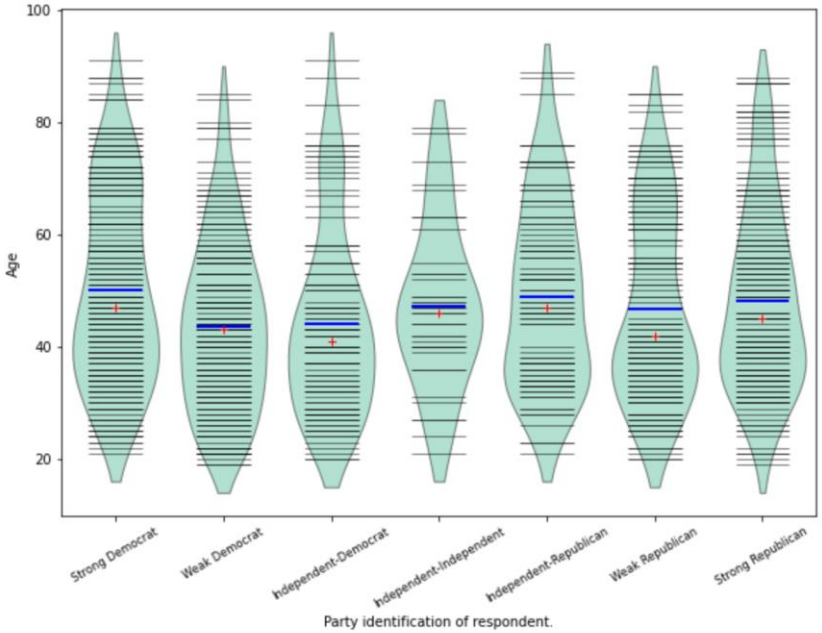
Box plots

- One more example…
- On the right shows results of a survey in the USA and with which party one identifies with

Can you draw a conclusion between the party identification of respondents and their age?

```
data = sm.datasets.anes96.load_pandas()
party_ID = np.arange(7)
labels = ["Strong Democrat", "Weak Democrat", "Independent-Democrat",
          "Independent-Independent", "Independent-Republican",
          "Weak Republican", "Strong Republican"]
```

```
plt.rcParams['figure.subplot.bottom'] = 0.23  # keep labels visible
plt.rcParams['figure.figsize'] = (10.0, 8.0)  # make plot larger in notebook
age = [data.exog['age'][data.endog == id] for id in party_ID]
fig = plt.figure()
ax = fig.add_subplot(111)
plot_opts={'cutoff_val':5, 'cutoff_type':'abs',
                           'label_fontsize':'small',
                           'label_rotation':30}
sm.graphics.beanplot(age, ax=ax, labels=labels,
                     plot_opts=plot_opts)
ax.set_xlabel("Party identification of respondent.")
ax.set_ylabel("Age")
plt.show()
```



Vragen?

hogeschool Windesheim

Radboud University