Introduction to Python

Tom Paskhalis

RECSM Summer School 2021, Linear Regression & Communicating Results, Part 5, Day 2

Anscombe's quartet

- 4 artificial datasets constructed by [Anscombe (1973)]
- All of them have nearly identical summary statistics
- But show dramatically different relationships between variables
- Designed to illustrate the importance of data visualization

Data for Anscombe’s quartet

Data for Anscombe’s quartet

In [1]:
```
import pandas as pd
anscombe_quartet = pd.read_csv('../data/anscombes_quartet.csv')
```

Data for Anscombe's quartet

In [1]:
```
import pandas as pd
anscombe_quartet = pd.read_csv('../data/anscombes_quartet.csv')
```

In [2]:
```
anscombe_quartet.head()
```

Out[2]:

	dataset	x	y
0	I	10	8.04
1	I	8	6.95
2	I	13	7.58
3	I	9	8.81
4	I	11	8.33

Summary statistics for Anscombe’s quartet

Summary statistics for Anscombe's quartet

In [3]:

```python
# Here we use `groupby` method to create summary by a variable ('dataset')
anscombe_quartet.groupby(['dataset']).describe()
```

Out[3]:

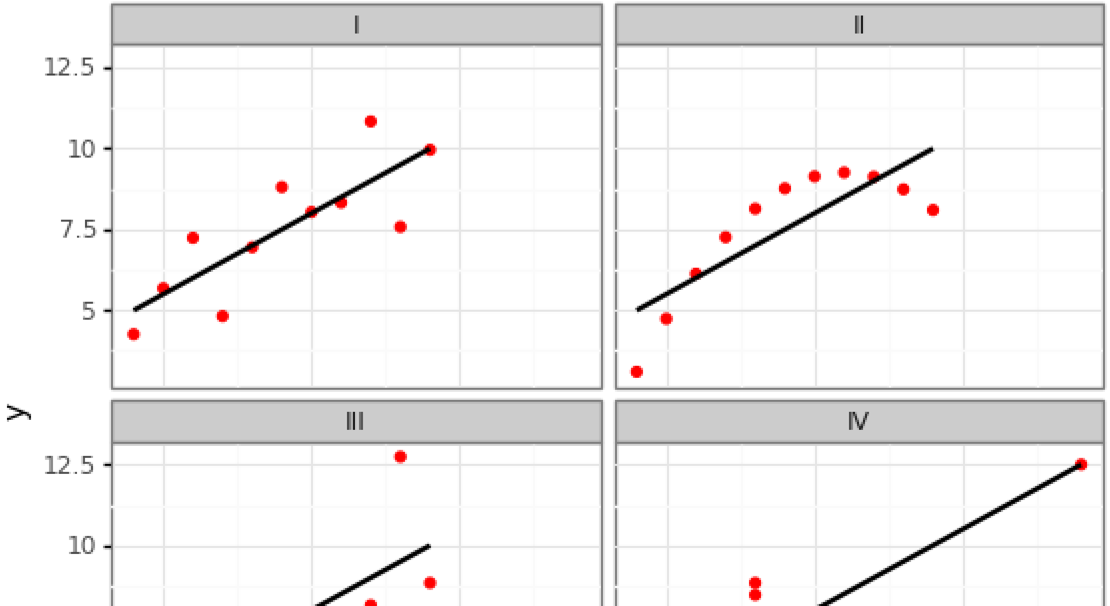
	x								y						
	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**	**count**	**mean**	**std**	**min**	**25%**	**50%**	**75**
dataset															
I	11.0	9.0	3.316625	4.0	6.5	9.0	11.5	14.0	11.0	7.500909	2.031568	4.26	6.315	7.58	8.5
II	11.0	9.0	3.316625	4.0	6.5	9.0	11.5	14.0	11.0	7.500909	2.031657	3.10	6.695	8.14	8.9
III	11.0	9.0	3.316625	4.0	6.5	9.0	11.5	14.0	11.0	7.500000	2.030424	5.39	6.250	7.11	7.9
IV	11.0	9.0	3.316625	8.0	8.0	8.0	8.0	19.0	11.0	7.500909	2.030579	5.25	6.170	7.04	8.1

Plotting Anscombe's quartet

Plotting Anscombe's quartet

In [4]:
```
from plotnine import *

ggplot(anscombe_quartet, aes(x = 'x', y = 'y')) +\
    geom_point(colour = 'red') +\
    geom_smooth(method = 'lm', se = False) +\
    facet_wrap('dataset') +\
    theme_bw()
```



Linear regression

- Linear regression is the classical tool of statistical analysis
- It allows to estimate the degree of association between variables
- Typically, it is the association between one or more independent variables (IV) and one dependent variable (DV)
- The main quantities of interest usually are direction, magnitude of association and its statistical significance

Linear regression in Python

- As for tabular data and visualization we need external libraries for running regression
- `statsmodels` library provides tools for estimating many statistical models
- Another useful library is `scikit-learn`
- It is more focussed on machine-learning applications

Linear regression in Python

- As for tabular data and visualization we need external libraries for running regression
- `statsmodels` library provides tools for estimating many statistical models
- Another useful library is `scikit-learn`
- It is more focussed on machine-learning applications

In [5]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf # Formula API provides R-style formula specif
```

Data transformation

Data transformation

```
In [6]: kaggle2020 = pd.read_csv('../data/kaggle_survey_2020_responses.csv', skiprows = [1]
```

Data transformation

In [6]:
```
kaggle2020 = pd.read_csv('../data/kaggle_survey_2020_responses.csv', skiprows = [1])
```

In [7]:
```
# Let's give more intuitive names to out variables
kaggle2020 = kaggle2020.rename(columns = {
                'Q1': 'age',
                'Q2': 'gender',
                'Q3': 'country',
                'Q4': 'education',
                'Q24': 'compensation'})
```

Data transformation

In [6]:
```
kaggle2020 = pd.read_csv('../data/kaggle_survey_2020_responses.csv', skiprows = [1])
```

In [7]:
```
# Let's give more intuitive names to out variables
kaggle2020 = kaggle2020.rename(columns = {
                'Q1': 'age',
                'Q2': 'gender',
                'Q3': 'country',
                'Q4': 'education',
                'Q24': 'compensation'})
```

In [8]:
```
kaggle2020['compensation'].head(n = 2)
```

Out[8]:
```
0                  NaN
1      100,000-124,999
Name: compensation, dtype: object
```

Data transformation

In [6]:
```
kaggle2020 = pd.read_csv('../data/kaggle_survey_2020_responses.csv', skiprows = [1]
```

In [7]:
```
# Let's give more intuitive names to out variables
kaggle2020 = kaggle2020.rename(columns = {
                'Q1': 'age',
                'Q2': 'gender',
                'Q3': 'country',
                'Q4': 'education',
                'Q24': 'compensation'})
```

In [8]:
```
kaggle2020['compensation'].head(n = 2)
```

Out[8]:
```
0                  NaN
1     100,000-124,999
Name: compensation, dtype: object
```

In [9]:
```
from statistics import mean
# Here we are replacing the compensation range by its midpoint (i.e. 112499.5 for $
# This variable requires substantial cleaning before transformation
```

Pandas and linear regression

Pandas and linear regression

```
In [10]: kaggle2020['compensation'] # Level of compensation (in USD) - our DV

Out[10]: 0             NaN
         1        112499.5
         2         17499.5
         3        137499.5
         4             NaN
                    ...
         20031         NaN
         20032         NaN
         20033       499.5
         20034       499.5
         20035       499.5
         Name: compensation, Length: 20036, dtype: float64
```

Pandas and linear regression

```
In [10]: kaggle2020['compensation'] # Level of compensation (in USD) - our DV

Out[10]: 0               NaN
         1          112499.5
         2           17499.5
         3          137499.5
         4               NaN
                      ...
         20031           NaN
         20032           NaN
         20033         499.5
         20034         499.5
         20035         499.5
         Name: compensation, Length: 20036, dtype: float64
```

```
In [11]: kaggle2020['gender'].value_counts()  # Frequencies of gender categories - our IV

Out[11]: Man                          15789
         Woman                         3878
         Prefer not to say              263
         Prefer to self-describe         54
         Nonbinary                       52
```

Pandas and linear regression

In [10]:
```
kaggle2020['compensation'] # Level of compensation (in USD) - our DV
```

Out[10]:
```
0              NaN
1         112499.5
2          17499.5
3         137499.5
4              NaN
            ...
20031          NaN
20032          NaN
20033        499.5
20034        499.5
20035        499.5
Name: compensation, Length: 20036, dtype: float64
```

In [11]:
```
kaggle2020['gender'].value_counts() # Frequencies of gender categories - our IV
```

Out[11]:
```
Man                        15789
Woman                       3878
Prefer not to say            263
Prefer to self-describe       54
Nonbinary                     52
```

Model summary

Model summary

In [13]:
```
fit1.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	compensation	**R-squared:**	0.007
Model:	OLS	**Adj. R-squared:**	0.007
Method:	Least Squares	**F-statistic:**	20.24
Date:	Tue, 29 Jun 2021	**Prob (F-statistic):**	1.25e-16
Time:	11:57:36	**Log-Likelihood:**	-1.3415e+05
No. Observations:	10729	**AIC:**	2.683e+05
Df Residuals:	10724	**BIC:**	2.683e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.63e+04	691.650	66.938	0.000	4.49e+04	4.77e+04
gender[T.Nonbinary]	5.538e+04	1.46e+04	3.797	0.000	2.68e+04	8.4e+04
gender[T.Prefer not to say]	1.294e+04	5733.824	2.256	0.024	1698.795	2.42e+04
gender[T.Prefer to self-describe]	1331.9561	1.36e+04	0.098	0.922	-2.53e+04	2.8e+04
gender[T.Woman]	-1.326e+04	1732.102	-7.653	0.000	-1.67e+04	-9859.882

Multiple linear regression

Multiple linear regression

```
In [14]: # Let's now also control for age and education
         fit2 = smf.ols('compensation ~ gender + age + education', data = kaggle2020).fit()
```

Multiple linear regression

In [14]:
```
# Let's now also control for age and education
fit2 = smf.ols('compensation ~ gender + age + education', data = kaggle2020).fit()
```

In [15]:
```
fit2.summary()
```

Out[15]:

OLS Regression Results

Dep. Variable:	compensation	**R-squared:**	0.116
Model:	OLS	**Adj. R-squared:**	0.115
Method:	Least Squares	**F-statistic:**	70.41
Date:	Tue, 29 Jun 2021	**Prob (F-statistic):**	1.98e-268
Time:	11:57:36	**Log-Likelihood:**	-1.3352e+05
No. Observations:	10729	**AIC:**	2.671e+05
Df Residuals:	10708	**BIC:**	2.672e+05
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.536e+04	2847.115	5.396	0.000	9781.101	2.09e+04
gender[T.Nonbinary]	5.813e+04	1.38e+04	4.220	0.000	3.11e+04	8.51e+04

Markdown - a language of reports

- Markdown is a markup language for formatting text with simple syntax
- The key goal of Markdown is readability
- Only a limited set of formatting options is supported
- Markdown is used in online documentation, blogging and instant messaging

Formatting text in Markdown

- For *italics* `*one star on each side*`
- For **bold** `**two stars on each side**`
- For ~~strikethrough~~ `~~two tildes on each side~~`

Lists in Markdown

For bulleted or unordered list of items:

```
- Just add a dash first and then write a text.
- If you add another dash in the following line, you wi
l have another item in the list.
  - If you add four spaces or use a tab key, you will c
eate an indented list.
```

For numbered or ordered list of items:

```
1. Just type a number and then write a text.
2. If you want to add a second item, just type in anoth
r number.
1. If you make a mistake when typing numbers, fear not,
```

Headers in Markdown

Headers or section titles are created with hashes(`#`)

```
# This is a first-tier header
```

```
## This is a second-tier header
```

```
### This is a third-tier header
```

Images and links in Markdown

- To add an image you can write `![some text](image_path)`
- To add a link you can write `[some text](URL)`
- For more complex cases HTML code can be used

Tables in Markdown

- Tables in Markdown can be created using the following syntax (there are a few variants)

```
| Header1 | Header2 |
|:--------|:--------|
| content | content |
```

- `:---` produces left-aligned text in cells
- `---:` produces right-aligned text in cells
- `:--:` produces centered text in cells

Markdown tables in pandas

- Pandas can generate Markdown tables from DataFrame

Markdown tables in pandas

- Pandas can generate Markdown tables from DataFrame

In [16]:
```python
# Let's revisit the summary statistics of Anscombe's quartet
anscombe_quartet.groupby(['dataset']).describe().iloc[:,0:3]
```

Out[16]:

	x		
	count	mean	std
dataset			
I	11.0	9.0	3.316625
II	11.0	9.0	3.316625
III	11.0	9.0	3.316625
IV	11.0	9.0	3.316625

Markdown tables in pandas

- Pandas can generate Markdown tables from DataFrame

In [16]:
```
# Let's revisit the summary statistics of Anscombe's quartet
anscombe_quartet.groupby(['dataset']).describe().iloc[:,0:3]
```

Out[16]:

	x		
	count	mean	std
dataset			
I	11.0	9.0	3.316625
II	11.0	9.0	3.316625
III	11.0	9.0	3.316625
IV	11.0	9.0	3.316625

In [17]:
```
print(anscombe_quartet.groupby(['dataset']).describe().iloc[:,0:3].to_markdown(inde
```

```
|    ('x', 'count') |    ('x', 'mean') |    ('x', 'std') |
|-----------------:|----------------:|---------------:|
|               11 |               9 |       3.31662 |
|               11 |               9 |       3.31662 |
```

The end