მონაცემთა ანალიტიკა Python

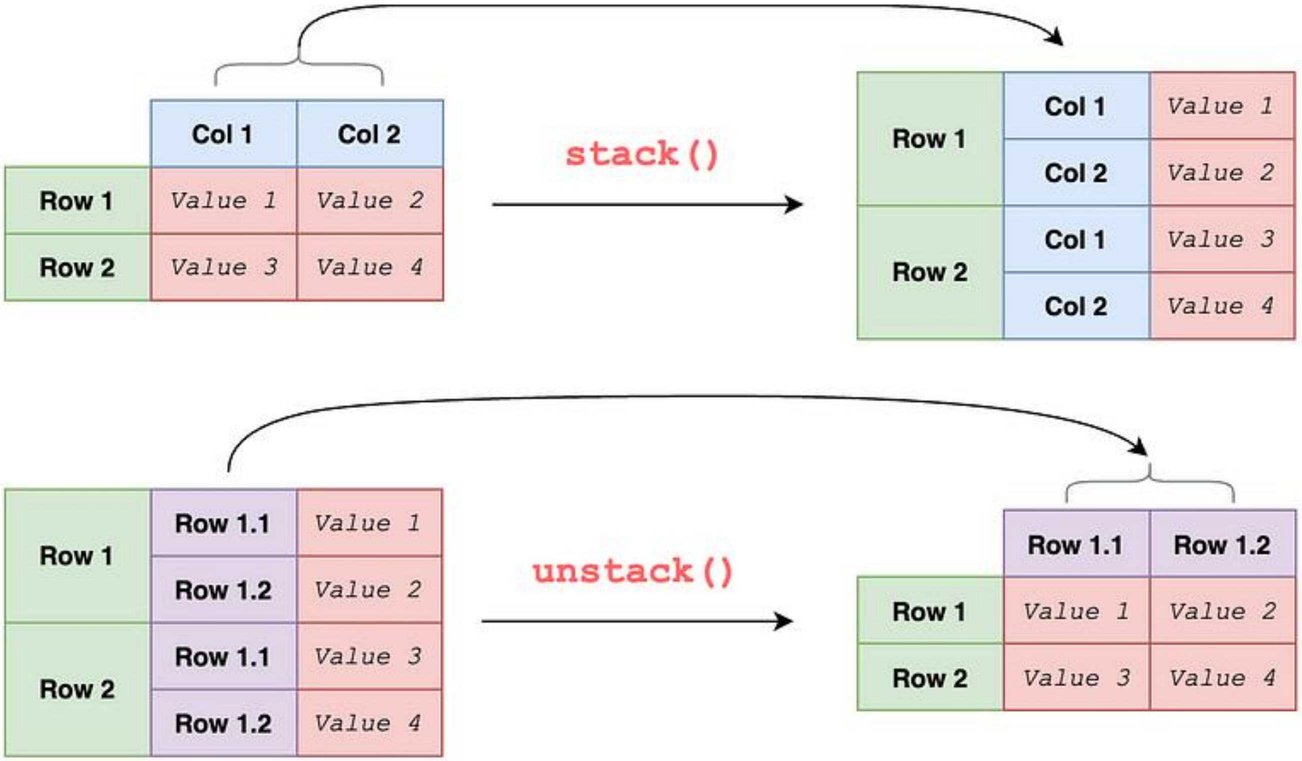
ლექცია 9: მონაცემების ფორმის ცვლილება. მონაცემების გახლეჩვა, დამუშავება და გაერთიანება. ჯვარედინა ტაბულაციის ცხრილები. მონაცემების ტრანსფორმაცია გრძელიდან განიერ და განიერიდან გრძელ ფორმატში.

ლიკა სვანაძე
lika.svanadze@btu.edu.ge

Reshaping a DataFrame

- Reshaping is often needed when you work with datasets that contain variables with some kinds of sequences, say, time-series data.
- Pandas provides various built-in methods for reshaping DataFrame. Among them, stack() and unstack() are the 2 most popular methods for restructuring columns and rows (also known as index).
- **stack():** stack the prescribed level(s) from column to row.
- **unstack():** unstack the prescribed level(s) from row to column. The inverse operation from stack.

Stack() and unstack()



	Col 1	Col 2
Row 1	Value 1	Value 2
Row 2	Value 3	Value 4

stack()

Row 1	Col 1	Value 1
	Col 2	Value 2
Row 2	Col 1	Value 3
	Col 2	Value 4

Row 1	Row 1.1	Value 1
	Row 1.2	Value 2
Row 2	Row 1.1	Value 3
	Row 1.2	Value 4

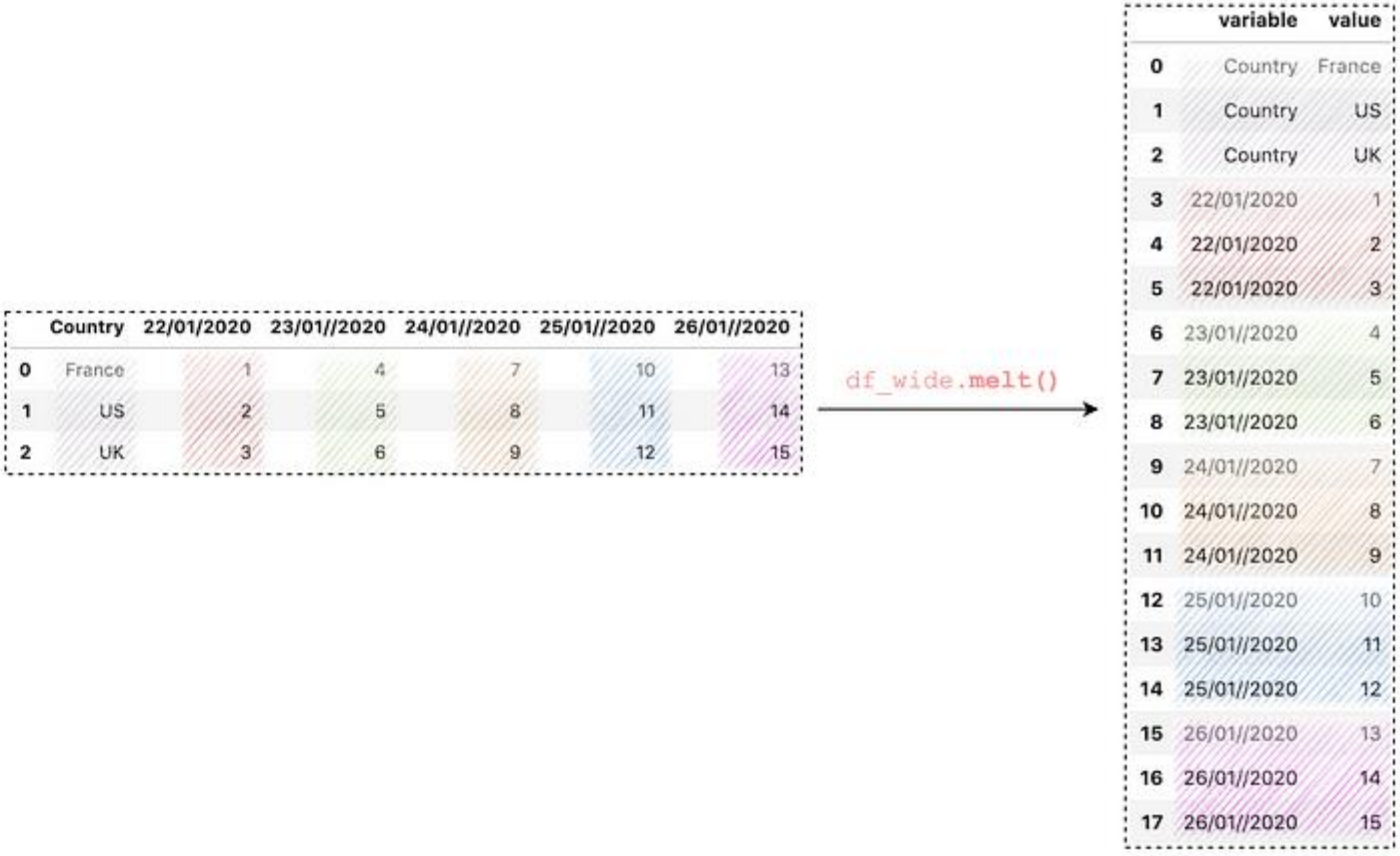
unstack()

	Row 1.1	Row 1.2
Row 1	Value 1	Value 2
Row 2	Value 3	Value 4

Reshaping a DataFrame using Pandas melt()

- **melt()** function is used to reshape the data from a wide format to a long format to facilitate further data analysis
- The simplest melt() doesn’t require any argument and it will turn all columns into rows (shown as a column variable) and list all associated values in a new column value.
- However, this output often doesn’t make much sense, so the general use case at least specifies the id_vars argument. For example, id_vars='Country' will tell pandas to keep Country as a column, and turn all the other columns into rows.

Reshaping a DataFrame using Pandas melt()

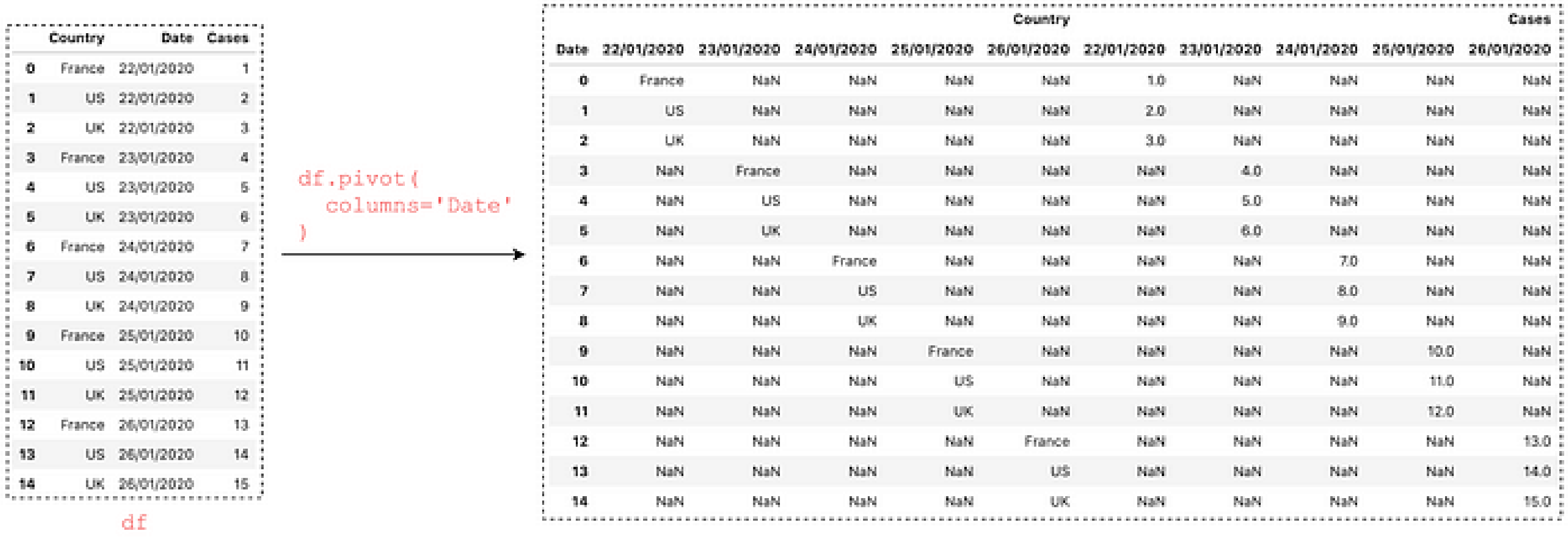


	Country	22/01/2020	23/01//2020	24/01//2020	25/01//2020	26/01//2020
0	France	1	4	7	10	13
1	US	2	5	8	11	14
2	UK	3	6	9	12	15

	variable	value
0	Country	France
1	Country	US
2	Country	UK
3	22/01/2020	1
4	22/01/2020	2
5	22/01/2020	3
6	23/01//2020	4
7	23/01//2020	5
8	23/01//2020	6
9	24/01//2020	7
10	24/01//2020	8
11	24/01//2020	9
12	25/01//2020	10
13	25/01//2020	11
14	25/01//2020	12
15	26/01//2020	13
16	26/01//2020	14
17	26/01//2020	15

Reshaping a DataFrame using Pandas pivot()

- Reshaping a DataFrame from long to wide format using
- In practice, we often need the complete opposite operation as well — to reshape the data from a long to a wide format. That’s where the Panda pivot() comes to help. In short, the Pandas pivot() is the complete opposite of melt().



	Country	Date	Cases
0	France	22/01/2020	1
1	US	22/01/2020	2
2	UK	22/01/2020	3
3	France	23/01/2020	4
4	US	23/01/2020	5
5	UK	23/01/2020	6
6	France	24/01/2020	7
7	US	24/01/2020	8
8	UK	24/01/2020	9
9	France	25/01/2020	10
10	US	25/01/2020	11
11	UK	25/01/2020	12
12	France	26/01/2020	13
13	US	26/01/2020	14
14	UK	26/01/2020	15

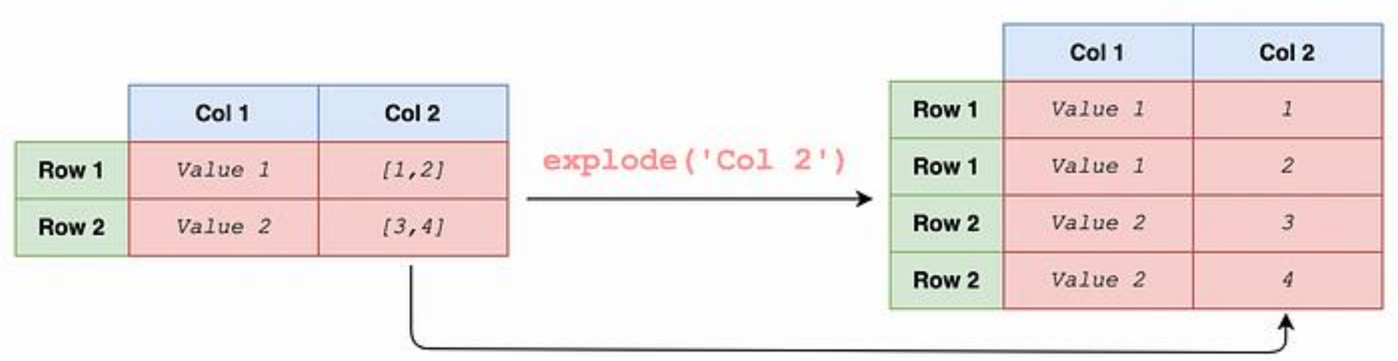
df

```
df.pivot(
   columns='Date'
)
```

	Country					Cases				
Date	22/01/2020	23/01/2020	24/01/2020	25/01/2020	26/01/2020	22/01/2020	23/01/2020	24/01/2020	25/01/2020	26/01/2020
0	France	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
1	US	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN
2	UK	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN
3	NaN	France	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN
4	NaN	US	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN
5	NaN	UK	NaN	NaN	NaN	NaN	6.0	NaN	NaN	NaN
6	NaN	NaN	France	NaN	NaN	NaN	NaN	7.0	NaN	NaN
7	NaN	NaN	US	NaN	NaN	NaN	NaN	8.0	NaN	NaN
8	NaN	NaN	UK	NaN	NaN	NaN	NaN	9.0	NaN	NaN
9	NaN	NaN	NaN	France	NaN	NaN	NaN	NaN	10.0	NaN
10	NaN	NaN	NaN	US	NaN	NaN	NaN	NaN	11.0	NaN
11	NaN	NaN	NaN	UK	NaN	NaN	NaN	NaN	12.0	NaN
12	NaN	NaN	NaN	NaN	France	NaN	NaN	NaN	NaN	13.0
13	NaN	NaN	NaN	NaN	US	NaN	NaN	NaN	NaN	14.0
14	NaN	NaN	NaN	NaN	UK	NaN	NaN	NaN	NaN	15.0

Reshaping a DataFrame using Pandas explode ()

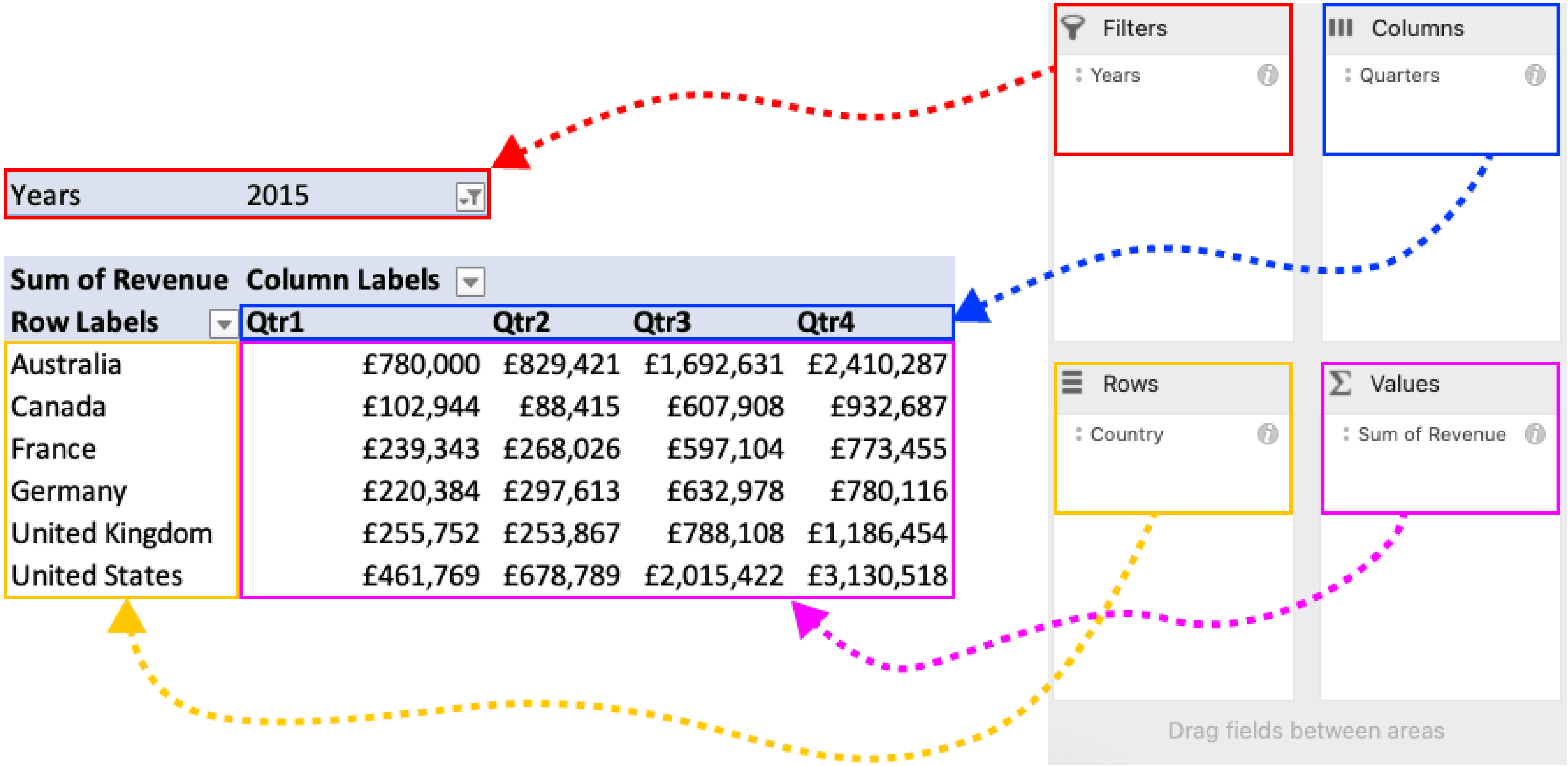
In the step of data pre-processing, we often need to prepare our data in specific ways before feeding it into a machine learning model. One of the examples is to transform list-like columns into rows. Pandas provides various methods for that, among them apply() and explode() are the two most popular methods.



	Col 1	Col 2
Row 1	Value 1	[1,2]
Row 2	Value 2	[3,4]

explode('Col 2')

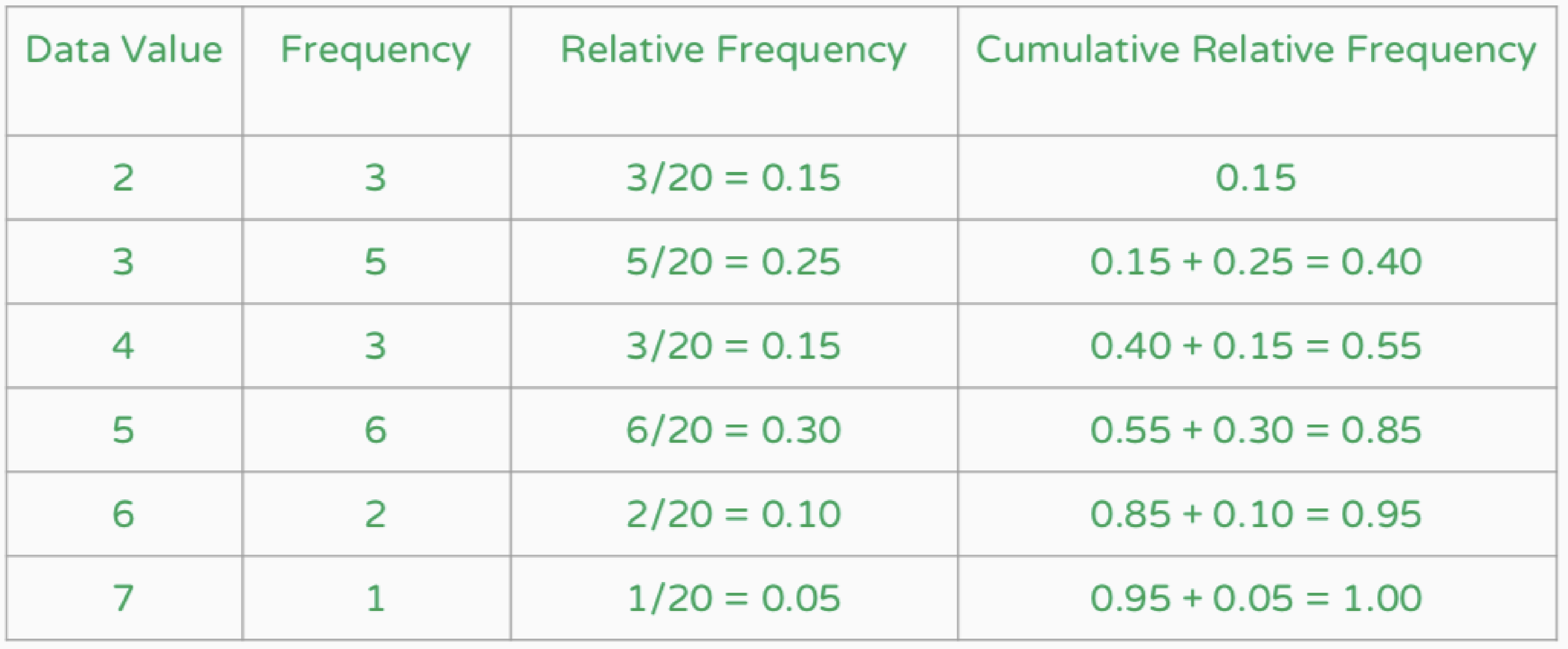
	Col 1	Col 2
Row 1	Value 1	1
Row 1	Value 1	2
Row 2	Value 2	3
Row 2	Value 2	4



Years 2015

Sum of Revenue	Column Labels			
Row Labels	Qtr1	Qtr2	Qtr3	Qtr4
Australia	£780,000	£829,421	£1,692,631	£2,410,287
Canada	£102,944	£88,415	£607,908	£932,687
France	£239,343	£268,026	£597,104	£773,455
Germany	£220,384	£297,613	£632,978	£780,116
United Kingdom	£255,752	£253,867	£788,108	£1,186,454
United States	£461,769	£678,789	£2,015,422	£3,130,518

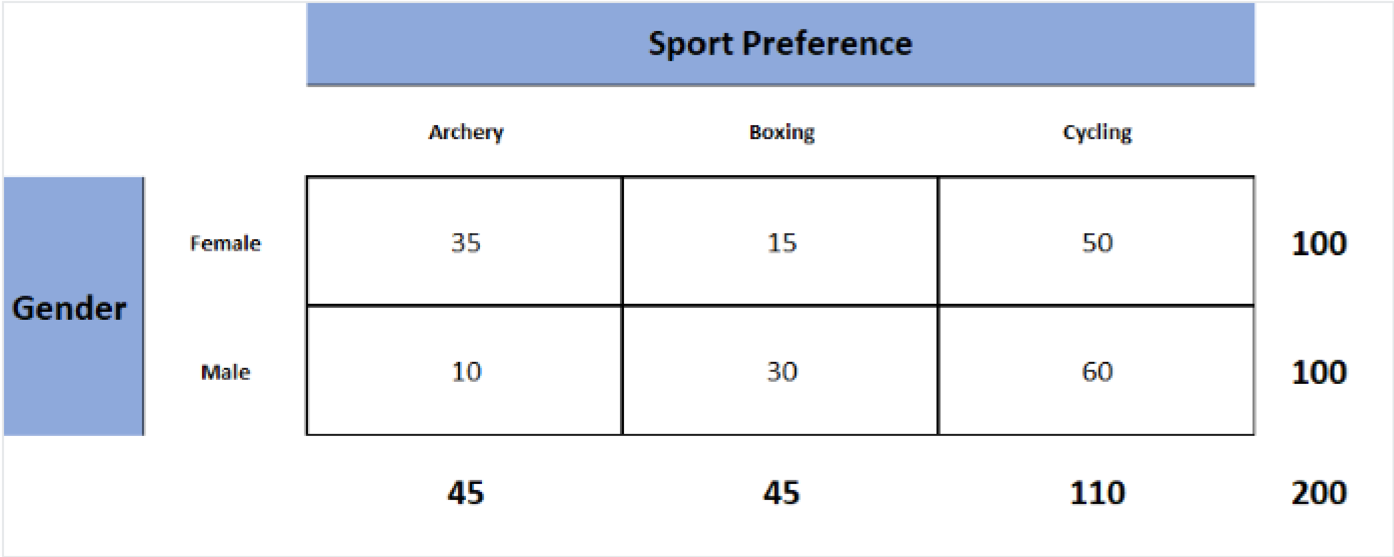
Frequency, Relative Frequency and CRF



Data Value	Frequency	Relative Frequency	Cumulative Relative Frequency
2	3	3/20 = 0.15	0.15
3	5	5/20 = 0.25	0.15 + 0.25 = 0.40
4	3	3/20 = 0.15	0.40 + 0.15 = 0.55
5	6	6/20 = 0.30	0.55 + 0.30 = 0.85
6	2	2/20 = 0.10	0.85 + 0.10 = 0.95
7	1	1/20 = 0.05	0.95 + 0.05 = 1.00

Contingency Table - crosstab

- A contingency table, sometimes called a two-way frequency table, is a tabular mechanism with at least two rows and two columns used in statistics to present categorical data in terms of frequency counts.
- To know the relationship between two ordinal or nominal variables then look for contingency table which displays this relationship.



		Sport Preference			
		Archery	Boxing	Cycling	
Gender	Female	35	15	50	**100**
	Male	10	30	60	**100**
		45	**45**	**110**	**200**