Basic techniques for investigating data objects

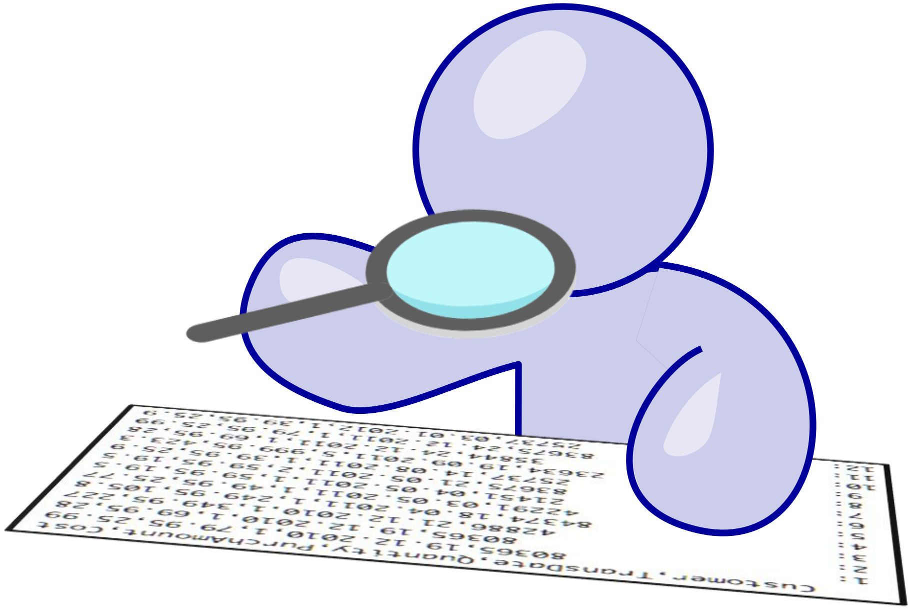
Observe and explore your data:
3 options to make sure the data is loaded correctly

Many mistakes can be made when loading data. Checking the data before working with it is always a good idea:

1. Look at the data

2. Look at the individual variables

3. Look at summary statistics



Step 1:
Look at your data

```
        Customer TransDate  Quantity PurchAmount   Cost   TransID
     0   149332 15.11.2005     1      199.95    107.00 27998739
     1   172951 29.08.2008     1      199.95    108.00 128888288
     2   120621 19.10.2007     1       99.95     49.00 125375247
     3   149236 14.11.2005     1       39.95     18.95 127996226
     4   149236 12.06.2007     1       79.95     35.00 128670302
...       ...       ...       ...       ...       ...       ...
223186   199997 17.09.2012     1       29.95     13.80 132481149
223187   199997 17.09.2012     1       29.95     13.80 132481149
223188   199998 17.09.2012     1       29.95     13.80 132481154
223189   199999 17.09.2012     1      179.95    109.99 132481165
223190   199542 17.09.2012     1       39.95     10.50 131973368

[223191 rows x 5 columns]
```

```
myData
```

Step 1:
Look at your data

Look at the first observations with the `head()` function:

```
myData.head(n=3)
```

```
     Customer  TransDate  Quantity PurchAmount   Cost    TransID
0    149332    15.11.2005     1       199.95    107.00  27998739
1    172951    29.08.2008     1       199.95    108.00  128888288
2    120621    19.10.2007     1        99.95     49.00  125375247
```

Do the same for the last observations with the `tail()` function:

```
myData.tail(n=3)
```

```
         Customer     TransDate  Quantity PurchAmount   Cost    TransID
223188    199998     17.09.2012     1        29.95     13.80  132481154
223189    199999     17.09.2012     1       179.95    109.99  132481165
223190    199542     17.09.2012     1        39.95     10.50  131973368
```

Step 1:
Look at your data

Look at the first observations with the `head()` function:

```
myData.head(n=3)
```

```
      Customer   TransDate  Quantity  PurchAmount    Cost    TransID
0     149332     15.11.2005     1        199.95     107.00  27998739
1     172951     29.08.2008     1        199.95     108.00  128888288
2     120621     19.10.2007     1         99.95      49.00  125375247
```

Do the same for the last observations with the `tail()` function:

```
myData.tail(n=3)
```

```
        Customer    TransDate  Quantity  PurchAmount    Cost    TransID
223188   199998     17.09.2012     1         29.95      13.80  132481154
223189   199999     17.09.2012     1        179.95     109.99  132481165
223190   199542     17.09.2012     1         39.95      10.50  131973368
```

Step 1:
Look at your data

Look at the first observations with the `head()` function:

```
myData.head(n=3)
```

```
     Customer  TransDate  Quantity PurchAmount   Cost   TransID
0    149332    15.11.2005     1       199.95    107.00  27998739
1    172951    29.08.2008     1       199.95    108.00  128888288
2    120621    19.10.2007     1        99.95     49.00  125375247
```

Do the same for the last observations with the `tail()` function:

```
myData.tail(n=3)
```

```
         Customer      TransDate  Quantity PurchAmount   Cost   TransID
223188    199998     17.09.2012     1        29.95     13.80  132481154
223189    199999     17.09.2012     1       179.95    109.99  132481165
223190    199542     17.09.2012     1        39.95     10.50  131973368
```

Step 2:
Look at individual variables

Before processing any data, you always have to ensure that your data is formatted properly and that the right data types are assigned to your variables. This will save a lot of time and you can avoid common mistakes.

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15/11/05	1	199.95	107.00	127998739
172951	29/08/08	1	199.95	108.00	128888288
120621	19/10/07	1	99.95	49.00	125375247
149236	14/11/05	1	39.95	18.95	127996226
149236	12/06/07	1	79.95	35.00	128670302
…	…	…	…	…	…

```
Customer        int64
TransDate       object
Quantity        int64
PurchAmount     float64
Cost            float64
dtype: object
```

Check if the type of the variables is correct.

```
mydata.dtypes
```

Sidenote: Built-in data types in Python

Python distinguishes between several data types. The most common are:

Data type		Description	Sign	Example
Logical		Variable is a logical value which can either be *True* or *False.*	bool	*True, False*
Numeric	integer	Variable is a number which can be written without a fractional component (whole-number).	int	*-3, 0, 1, 2, 3,…*
	float	Variable is a computational approximation of any real-valued number.	float	*-2.6, 1.0, 1.1, 1.329*
Text	string	Variable is interpreted as "text".	str	*"a", "Z", "Hello", "Anna"*
Categorical	pandas. categorical	Variable, which can take on only a limited and usually fixed, number of possible values on a nominal scale.	category	*pd.Series(["a","b","c"], dtype="category")*
Dates and time	datetime	Variable is a data or time and special functionalities for manipulation are provided.	date / time	*d = datetime.datetime (2009, 10, 5)*

Sidenote: Built-in data types in Python

Python distinguishes between several data types. The most common are:

Data type		Description	Sign	Example
Logical		Variable is a logical value which can either be *True* or *False.*	bool	*True, False*
Numeric	integer	Variable is a number which can be written without a fractional component (whole-number).	int	*-3, 0, 1, 2, 3,…*
	float	Variable is a computational approximation of any real-valued number.	float	*-2.6, 1.0, 1.1, 1.329*
Text	string	Variable is interpreted as "text".	str	*"a", "Z", "Hello", "Anna"*
Categorical	pandas. categorical	Variable, which can take on only a limited and usually fixed, number of possible values on a nominal scale.	category	*pd.Series(["a","b","c"], dtype="category")*
Dates and time	datetime	Variable is a data or time and special functionalities for manipulation are provided.	date / time	*d = datetime.datetime (2009, 10, 5)*

Sidenote: Built-in data types in Python

Python distinguishes between several data types. The most common are:

Data type		Description	Sign	Example
Logical		Variable is a logical value which can either be *True* or *False.*	bool	*True, False*
Numeric	integer	Variable is a number which can be written without a fractional component (whole-number).	int	*-3, 0, 1, 2, 3,…*
	float	Variable is a computational approximation of any real-valued number.	float	*-2.6, 1.0, 1.1, 1.329*
Text	string	Variable is interpreted as "text".	str	*"a", "Z", "Hello", "Anna"*
Categorical	pandas. categorical	Variable, which can take on only a limited and usually fixed, number of possible values on a nominal scale.	category	*pd.Series([“London", “Berlin",“Paris"], dtype="category")*
Dates and time	datetime	Variable is a data or time and special functionalities for manipulation are provided.	date / time	*d = datetime.datetime (2009, 10, 5)*

Sidenote: Module "datetime"

A **module** is a single file which can be imported in Python. A **package** refers to a collection of modules. 1

If working with dates and times, many mistakes occur when dates and times are not identified and/or formatted properly (especially wrt international settings).

"with respect to" 2

The "**datetime**" module makes it easier to work with dates and times:

- Identify and parse time
- Extract and modify years, months, days, hours, …
- Perform accurate math with date-times

Format the data

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15/11/05	1	199.95	107.00	127998739
172951	29/08/08	1	199.95	108.00	128888288
120621	19/10/07	1	99.95	49.00	125375247
149236	14/11/05	1	39.95	18.95	127996226
149236	12/06/07	1	79.95	35.00	128670302
	...	...	...	...	...

Customer	TransDate	...
149332	2005-11-15	...
172951	2008-08-29	...
120621	2007-10-19	...
149236	2005-11-14	...
149236	2007-06-12	...
...	...	...

Object ❶

Pandas object to modify ❷

Column to modify ❸

```
myData["TransDate"]=
        pd.to_datetime (myData["TransDate"],
                      format="%d.%m.%Y",
                      utc=True, dayfirst=True)
```

Function is part of the pandas library ❹

Pandas recognizes often used separators as "-" and "." automatically, but it is safer to specify the format explicitly. ❺

Ensure the right timezone ❻

Ensure correct month and day ordering ❼

Recognized as date ❽

Format the data

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15/11/05	1	199.95	107.00	127998739
172951	29/08/08	1	199.95	108.00	128888288
120621	19/10/07	1	99.95	49.00	125375247
149236	14/11/05	1	39.95	18.95	127996226
149236	12/06/07	1	79.95	35.00	128670302
	...	...	...	...	...

Customer	TransDate	...
149332	2005-11-15	...
172951	2008-08-29	...
120621	2007-10-19	...
149236	2005-11-14	...
149236	2007-06-12	...
...	...	...

Object 1

Recognized as date 8

Pandas object to modify 2

Column to modify 3

```
myData["TransDate"]=
        pd.to_datetime (myData["TransDate"],
                         format="%d.%m.%Y",
                         utc=True, dayfirst=True)
```

Pandas recognizes often used separators as "-" and "." automatically, but it is safer to specify the format explicitly. 5

Function is part of the pandas library 4

Ensure the right timezone 6

Ensure correct month and day ordering 7

Format the data

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15/11/05	1	199.95	107.00	127998739
172951	29/08/08	1	199.95	108.00	128888288
120621	19/10/07	1	99.95	49.00	125375247
149236	14/11/05	1	39.95	18.95	127996226
149236	12/06/07	1	79.95	35.00	128670302
	...	...	...	...	...

Object 1

Customer	TransDate	...
149332	2005-11-15	...
172951	2008-08-29	...
120621	2007-10-19	...
149236	2005-11-14	...
149236	2007-06-12	...
...	...	...

Recognized as date 8

Pandas object to modify 2

Column to modify 3

```
myData["TransDate"]=
        pd.to_datetime (myData["TransDate"],
                        format="%d.%m.%Y",
                        utc=True, dayfirst=True)
```

Pandas recognizes often used separators as "-" and "." automatically, but it is safer to specify the format explicitly. 5

Function is part of the pandas library 4

Ensure the right timezone 6

Ensure correct month and day ordering 7

Format the data

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15/11/05	1	199.95	107.00	127998739
172951	29/08/08	1	199.95	108.00	128888288
120621	19/10/07	1	99.95	49.00	125375247
149236	14/11/05	1	39.95	18.95	127996226
149236	12/06/07	1	79.95	35.00	128670302
	...	...	...	...	...

Customer	TransDate	...
149332	2005-11-15	...
172951	2008-08-29	...
120621	2007-10-19	...
149236	2005-11-14	...
149236	2007-06-12	...
...	...	...

Object 1

Pandas object to modify 2

Column to modify 3

```
myData["TransDate"]=
        pd.to_datetime (myData["TransDate"],
                        format="%d.%m.%Y",
                        utc=True, dayfirst=True)
```

Function is part of the pandas library 4

Pandas recognizes often used separators as "-" and "." automatically, but it is safer to specify the format explicitly. 5

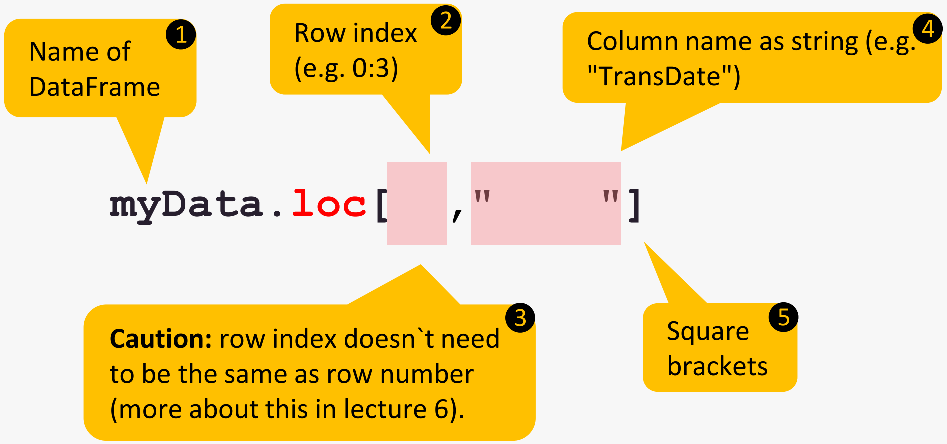
Ensure the right timezone 6

Ensure correct month and day ordering 7

Recognized as date 8

Sidenote: General command structure for addressing columns in a Pandas DataFrame

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...



```
myData.loc[ ,"     "]
```

1 Name of DataFrame

2 Row index (e.g. 0:3)

3 **Caution:** row index doesn`t need to be the same as row number (more about this in lecture 6).

4 Column name as string (e.g. "TransDate")

5 Square brackets

Step 3:
Look at summary statistics

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

```
myData.describe()
```

```
            Customer       Quantity    PurchAmount           Cost
count  223191.000000  223191.000000  223191.000000  223191.000000
mean   148366.708384       1.037009      84.164615      39.013295
std     28657.866956       0.336899     105.864308      57.145100
min    100001.000000       1.000000       0.000000       0.000000
25%    123563.000000       1.000000      34.950000      14.030000
50%    148635.000000       1.000000      59.950000      24.000000
75%    172467.000000       1.000000      99.950000      45.000000
max    199999.000000      70.000000    5000.000000    3100.000000
```

Are the summary statistics as you expect them to be?

Step 3:
Look at summary statistics

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

```
myData.describe()
```

```
           Customer       Quantity    PurchAmount          Cost
count  223191.000000  223191.000000  223191.000000  223191.000000
mean   148366.708384       1.037009      84.164615      39.013295
std     28657.866956       0.336899     105.864308      57.145100
min    100001.000000       1.000000       0.000000       0.000000
25%    123563.000000       1.000000      34.950000      14.030000
50%    148635.000000       1.000000      59.950000      24.000000
75%    172467.000000       1.000000      99.950000      45.000000
max    199999.000000      70.000000    5000.000000    3100.000000
```

Are the summary statistics as you expect them to be?

Write data as CSV

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

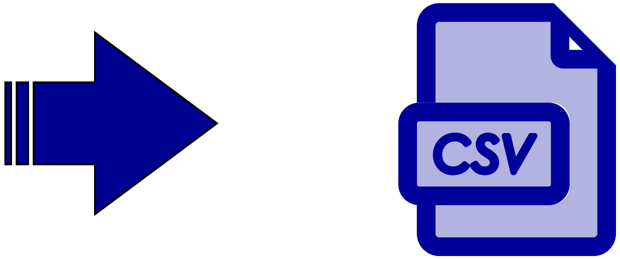
Object to save 1

```
myData.to_csv(path_or_buf="file.csv",
                 sep="," ,...)
```

Value separation 3

Write data as CSV

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...



1 Object to save

2 Location and name of output CSV file

```
myData.to_csv(path_or_buf="file.csv",
                sep="," ,...)
```

3 Value separation

Sidenote: Remove objects from your workspace

When you are finished with an object it is good practice (but not obligatory) to remove it from your workspace. Thus, you save storage and keep your programming environment clean:

```
del DataFrame
```

`del` removes objects from your environment

Basic techniques for investigating data objects