# COMP3430/COMP8430 – Data Wrangling – 2019

# Lab 1: Data Exploration using Rattle and Pandas Week 3

---

## Overview and Objectives

The objectives of this first lab are to get familiar with the graphical user interface of the open source data wrangling and mining tool **Rattle**, and the Python **Pandas** library, and to conduct data exploration on smaller example data sets.

Rattle is a freely available software tool that provides a graphical user interface on top of the R statistical programming language. Rattle provides access to many of the data wrangling, data mining and statistical functionalities in R. Pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas enables you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R. For more on R and how to download Rattle and Pandas please see the **Course Resources** link on the COMP8430 Wattle page.

**This is a data exploration lab, so go and explore!** We don't give you a very strict list of tasks to do, but rather some possibilities and questions we encourage you to investigate. Also think about what the advantages and disadvantages, and strengths and limits, of tools like Rattle and Pandas are. Ask your tutor if you need help!

## Preliminaries

If you have never logged into a computer in the CSIT labs and you wish to do the labs on the CSIT machines, **you must login to https://cs.anu.edu.au/streams/** with your ANU ID and password – then log out. This will distribute your ID and password to the local lab machines. Please note this can take a few minutes.

Alternatively, if you plan to use your own laptop, try to install Rattle and Pandas before the lab. Installation instructions for Rattle are provided in the Rattle Data Miner documentation, and for Pandas installation instructions and extensive documentation can be found at https://pandas.pydata.org.

For this lab we will be using Pandas installed as part of the open source data science package **anaconda** (see the course resource page for more details).

You should have a look at the Rattle documentation at http://datamining.togaware.com/survivor/Rattle_Data.html. Note that both Rattle and its documentation are under development, and currently not all functionality and chapters are complete. Any feedback on errors, typos, and other issues is much appreciated (you can tell us and we will contact the Rattle developer Graham Williams).

## Starting and Quitting Rattle

**Starting Rattle**: After logging into a lab machine using your ANU ID and password, start a Terminal (Konsole). This can be done by clicking on the Main Menu (round orange-red icon on the left top menu panel), select Accessories, then Terminal. If you are using the alternative interface to Ubuntu, it can be found by clicking on Applications, then System Tools, then Terminal.

A new window will pop-up with a cursor that allows text input. Type the upper-case letter `R` followed by 'Enter'. This will start the R statistical language. The prompt (character at the beginning of the line where the cursor is) should have changed to '>'. Now type the following into the terminal window:

```
library(rattle)
```

followed by 'Enter'. Some text on Rattle will be shown. Finally, to start Rattle type:

```
rattle()
```

followed by 'Enter'. Make sure that you type the opening and closing bracket! A new window will appear, which is the main Rattle graphical user interface.

**Quitting Rattle**: Click on the 'Quit' button in the main Rattle window, and confirm with a click on Yes if you really want to quit. Exit R by typing 'Ctrl-D' (holding both the 'Ctrl' and the 'D' keys at the same time). When asked to save the workspace image you can type 'n' followed by 'Enter'.

**Before you leave the lab make sure you log out of your machine!**

## Part 1: Lab Questions on Rattle (spend about 1 hour on this part)

This section basically consists of working with Rattle and load and explore a small example data set. To load this example data set conduct the following two steps:

- Click on the `Execute` button.
- Click on Yes when asked: "Would you like to use the example **weather** dataset?"

The **weather** data set comes from the Australian Bureau of Meteorology, and it contains 366 records (observations) and 24 attributes (or variables). Note: The full directory path where the Rattle example data sets are located is:
**/usr/local/lib/R/site-library/rattle/csv/**

You can now see detailed information about this data set. Please read through the `Help` menu (`Data` sub-menu) or the Data section in the online Rattle documentation about the different role types and other data related settings. Make sure you understand the different role types before you continue.

If you would like to look at other data sets, then click on the `Library` radio button, and select one of the many available data sets from the menu available next to `Data Name`. You always need to confirm your selection with a click on the `Execute` button.

Before you continue make sure you have selected the **weather** sample data set (and have confirmed your selection with a click on `Execute`).

Make sure you untick the `Partition` box! Partitioning is only required for supervised machine learning algorithms which we do not cover in this course.

Now go to the `Explore` tab, which offers a large number of possibilities to explore the loaded data set.

1. Explore the content of the **weather** data set using the various options provided. Specifically, you should learn about the distribution of values in the variables (attributes), as well as the number of missing values (use `Summary` / `Describe` / `Correlation` / `Show Missing` etc. for this).

   Some specific questions you might want to explore could include:

   - Which variables are most strongly positively correlated? Does this make sense?
   - Which variables are most strongly negatively correlated? Again, does this make sense?
   - Which variable is most positively correlated with variable `Evaporation`? Which one is most negatively correlated with `Evaporation`?
   - Which variable has the most skewed distribution?
   - Which variable has the largest number of missing values?
   - Which variable combinations have the largest number of missing values?
   - What is the meaning of the Missing Value Summary table?

   To analyse the numerical values of correlations, you can also go to the `Test` tab, select `correlation` and choose two variables. More information about tests can be found in the `Test` chapter of the Rattle documentation. Some specific questions you might want to explore could include:

   - Which two numerical variables, and which two categorical variables are the most correlated?
   - Can you calculate the correlation between numerical and categorical variables? If so how, if not why?

   If some of the result plots are too clustered, then you might want to go back to the `Data` page and set the role of certain variables to `Ignore` (make sure you click `Execute` to confirm the new selection).

2. Now select to the `Distributions` option and select plot types appropriate to the content of the variables of the `weather` data set. Specifically, create both `box plots` and `histograms` for different variables and examine how they differ.

   Then explore the different plotting types – which ones are more useful than others for certain types of variables?

   **Note:** If the graphics output does not work (an empty window appears and an error message is printed in the terminal window), please go to the Rattle `Settings` menu and untick the `Advanced Graphics` and the `Use Cairo Graphics Device` options.

3. Save some of your plots into files using different formats, such as PDF or PNG, and then open these files and zoom in. Which shows the best quality when zoomed in? For your assignments use plots of highest quality possible.

## Starting and Quitting Pandas

**Starting Pandas**: After logging into a lab machine using your ANU ID and password, start a Terminal (Konsole) as described above under starting Rattle.

A new window will pop-up with a cursor that allows text input. Type `anaconda2` followed by 'Enter' to start the Anaconda distribution that uses Python 2.7. The prompt (character at the beginning of the line where the cursor is) should have changed to 'your ANU ID@anaconda2:/$'. Now type the following into the terminal window to start Python:

```
python
```

followed by 'Enter'. The prompt (character at the beginning of the line where the cursor is) should have changed to '>>>'.

**Quitting Python**: To exit from Python console type 'quit()' or holding both the 'Ctrl' and the 'D' keys at the same time. Once you successfully exit from the Python console you should be return back to Anaconda. Exit Anaconda by typing 'exit' followed by 'Enter'.

**Before you leave the lab make sure you log out of your machine!**

## Part 2: Lab Questions on Pandas

This section basically consists of working through Pandas, its DataFrame data structure, and exploring and visualising data using Pandas.

We will use the same simple example data set used which we have used in the previous exercises using Rattle. First we need to import Pandas into Python. To import the Pandas library in the Python console type:

```
import pandas as pd
```

followed by 'Enter'. Once pandas is successfully imported into Python the prompt (character at the beginning of the line where the cursor is) should have changed back to '>>>'. Note the keyword 'as' is used to modify the names of modules and their functions within Python. So we now only have to write `pd` instead of `pandas`.

Next we load the weather data set into a 'DataFrame' data structure in pandas. A DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labelled axes (rows and columns) which is used to load data in Pandas.

To read the weather data set into a DataFrame named `df` we use the function *read_csv()* in Pandas:

```
df = pd.read_csv("/usr/local/lib/R/site-library/rattle/csv/weather.csv")
```

Note: You may copy the weather data set into your working folder and read that file from your working folder rather reading it from its original location.

You can now see detailed information about this data set. To read the first 5 records in the DataFrame we can use the *head()* function.

```
df.head()
```

- Try to read the first 10, 20, and 50 records. You can pass the number of records to view as a variable to the *head()* function.
- Similarly we can use the function *tail()* to view the last records in a DataFrame. Using the *tail()* function try to read the last 10, 20, and 50 records in df.

If you need to select a range of rows we can specify the range using ':' as in: `df[10:20]`

Note that in this example the rows (records) that start from index 10 up-to 19 are returned as a result. So for the 0:10 range the first 10 rows are returned with the positions starting with 0 and ending with 9. Try viewing rows with different ranges. See what happens if you use negative values as indexes.

To view the variables (column names) in the loaded data set we can use the attribute **columns** of the DataFrame: `df.columns`

To view the values of a column we can use the column name as an index: `df['column1']` or `df.column1`

To access multiple columns we can use list of column names as an index: `df[['column1','column2']]`

Following is a list of attributes of a data frame that can be used to get information about different variables in the data set.

| DataFrame.attribute | Description |
|---|---|
| dtypes | List the types of the columns |
| columns | List the columns names |
| axes | List the row labels and column names |
| size | Number of elements |

Using these attributes try to:

- Find how many records (rows) this data frame has.
- Find how many elements are there (and think about what an element is).
- Find the column names.
- Find what types of columns (object, int, float, or datetime) we have in this data frame.

Also, there are many other ways to subset (slice) the data frame. We encourage you to explore these slicing techniques by reading the corresponding Pandas documentation.

To view the missing values in a data frame we can used the function *isnull()*. To select the rows that have at least one missing value we can use `df[df.isnull().any(axis=1)]` or `df[df.isnull().all(axis=1)]` to select rows that have missing values in all columns. Missing values will be marked as NaN (not-a-number) in the result.

Similar to attributes, a data frame contains many methods that can be used to view statistics of a data set. The table on the next page shows some of the methods that can be used view basic descriptive statistics.

You can use each of these functions on a single or a set of columns to generate statistics. For example to generate basic descriptive statistics for a single column we can use `df['column1'].describe()` and `df[['column1', 'column2']].describe()` for multiple columns. To compute the correlation between two columns you can use `df['column1'].corr(df['column2'])`.

Also, if you want to view all the unique values of a column you can use the function *unique()*.

```
df['column1'].unique()
```

Now using the above methods explore the content of the **weather** data set. Specifically, you should learn about the basic statistics of the values in the variables, as well as the number of missing values. Similar as in the Rattle exercises some specific questions you might want to explore could include:

| DataFrame.method | description |
|---|---|
| describe() | Generate basic descriptive statistics such as count, mean, std, min, max, and quantiles (with different statistics provided for non-numerical columns) |
| min(), max() | Gives minimum and maximum values |
| count() | Gives the number of values |
| unique() | Gives the number of unique values |
| mean() | Gives the arithmetic average |
| median() | Gives the median value |
| mode() | Gives the mode value |
| mad() | Gives the median absolute deviation value |
| var(), std() | Gives the variance and standard deviation values |
| skew() | Summarises the skewness of data |
| kurt() | Summarises the kurtosis of data |
| corr() | Gives the correlation between suitable attributes |
| crosstab() | Computes the simple cross tabulation of two columns e.g. crosstab(df.column1, df.column2) |

- Which variables are most strongly positively correlated? Does this make sense?
- Which variables are most strongly negatively correlated? Again, does this make sense?
- Which variable is most positively correlated with variable `Evaporation`? Which one is most negatively correlated with `Evaporation`?
- Which variable has the most skewed distribution?
- Which variable has the largest number of missing values?
- Which variable combinations have the largest number of missing values?
- **Programming idea:** Can you implement the Missing Value Summary table from Rattle using Python (with or without Pandas)?

Similar to Rattle, visualisations with Pandas is easy when you want to view your data. To generate different plots we first need to import the `matplotlib` plotting library into Python.

```
import matplotlib.pyplot as plt
```

To plot a histogram of a data column we can use the function *plot()* or *hist()*.

`df['column1'].plot(kind='hist', bins=50)` or `df['column1'].plot.hist(bins=50)`

Similarly, you can show histograms of multiple columns in the same plots.

`df[['column1','column2']].plot(kind='hist', bins=50)` or `df[['column1','column2']].plot.hist(bins=50)`

Note, this generates a matplotlib plot object and to view this plot we can use,

```
plt.show()
```

To plot a box plot of a data column or set of columns we can use the function *plot()* or *hist()*.

`df['column1'].plot(kind='box')` or `df['column1'].plot.box()`

and for multiple attributes,

`df[['column1','column2']].plot(kind='box')` or `df[['column1','column2']].plot.box()`

To plot a density distribution plot of a data column or set of columns we can use the function *plot()* or *kde()*.

`df['column1'].plot(kind='kde')` or `df['column1'].plot.kde()`

and for multiple attributes,

`df[['column1','column2']].plot(kind='kde')` or `df[['column1','column2']].plot.kde()`

Create different plots for different variables and examine how they differ. Do these plots show the same as the plots generated by Rattle?

Again, save some of your plots into files using different formats, and then open these files and zoom in. Which shows the best quality when zoomed in? For your assignments use plots of highest quality possible.

The `matplotlib.pyplot.savefig()` allows you to save plots directly from a Python program, where the file name extension you pass to that function will determine the file type.