Week 2

#Data Science/2-Data Science Tools#

Jupyter Notebook and JupyterLab

Introduction to Jupyter Notebook

Welcome Introduction to Jupyter Notebooks. This tutorial walks you through what Jupyter is and some basic syntax. The following videos will contain tutorials and hands-on practice. A Jupyter notebook is a browser-based application that allows you to create and share documents that contain code, equations, visualizations, narrative text links, and so much more. It can be likened to a scientist's lab notebook where a scientist records all steps to perform their experiments and record the results so it can be reproduced in the future.

Similarly, a Jupyter Notebook allows a Data Scientist to record their data experiments, their results, and allows for the same experiment to be used and repeated by anyone. A Jupyter Notebook file allows Data Scientists to combine descriptive text, code blocks, and code output in a single file. When you run the code, it generates the outputs, including plots and tables, within the notebook file. And you can then export the notebook to a PDF or HTML file that can be then be shared with anyone.

Jupyter Notebooks originated as "iPython" originally developed for the Python Programming language. As it came to support additional languages it was renamed Jupyter which stands for: Julia - Python - R. But nowadays, the notebook supports many other languages as well. Jupyter Lab is a browser-based application that allows you to access multiple Jupyter Notebook files as well as other code and data files.

Jupyter Lab extends the functionalities of Jupyter notebooks by enabling you to work with multiple notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner. JupyterLab allows for: Interactive control of the notebook cells and output, real time editing markdowns, CSV etc. It is compatible with several file formats like CSV, JSON, PDF, Vega and more. And is open source.

Jupyter notebooks can be used with a cloud-based service like IBM and Google Collab. They do not require you to install anything on your local machine; they give you access to the Jupyter Notebook environment. They do allow you to import and export notebooks using the standard IPython Notebook file format. They support the Python language and other languages as well. Jupyter notebooks can be installed via the command line using the pip install function. Jupyter notebooks can also be downloaded locally to your laptop through the Anaconda Platform from Anaconda.com. Anaconda is one of the popular distributions which includes Jupyter and Jupyterlab.

For the purposes of learning in this course, you have access to a hosted version of JupyterLab in

Skills Network Labs, so you do not have download and install anything on your own device in order to complete the hands-on labs. You will see a screen like this that launches the Jupyter Lab in the virtual environment. Simply click on Open tool.
You should now be familiar with: How Jupyter notebooks are used in Data Science for recording experiments and projects. How Jupyter Lab is compatible with many files and many Data Science languages. And different ways to install and use Jupyter Notebooks.

Getting Started with Jupyter

In this video we will go through the basics of Getting started with Jupyter. In the lab session of this module, we can launch a notebook using the Skills Network virtual environment. Simply click on Open Tool and the environment will open up Jupyter Lab. The lab will open a notebook like this one.
When we start the notebook, we can change the name by clicking on File, then renaming the notebook to the preferred name. We can now start working with our new project. Let us print “hello world” to show that the environment is working properly. To get an output from a code cell, you must run the cell. To run the cell, place your cursor in the cell and click on Run on the top of your screen. Then click Run Selected Cells to run the current highlighted cells. To use a shortcut, press Shift + Enter. You can also run all the cells in the notebook by clicking on Run All Cells.
To add a code to a new cell, click on the plus symbol. You can work with multiple files. You can do that by clicking the plus button above the folders in your local machine, then choose the file you want to open; we will open another notebook. You can also use the file option to open a new launcher or open a new notebook. When we open the new file, we can move them around based on our preference. When working with multiple notebooks for the same problem, I like to put them side-by-side, so I can see both notebooks at the same time.
Now we will assign the variable one to the number 1, and the variable two to the number 2, and we will find the result of adding one and two. To delete a cell highlight the cell and click on Edit and choose Delete Cells. To use a shortcut press D twice on the highlighted cell. If you decide to move your cells around for better flow within the Notebook, you can either move the cell up or move the cell down depending on your preference. As data scientists, communicating our results is a big and important part of our jobs. Jupyter supports presenting results directly from the notebooks. You can create Markdown to add titles and text description to help with the flow of the presentation. To do that, click on the drop-down bar by code and select markdown.
We will create line plots and convert each cell and output into a slide or sub-slide depending on preference. And we will have a presentation.
The slides' functionality in Jupyter allows us to effectively deliver code, visualization, text, and outputs of the executed code as this is very important for presenting Data Science projects.
When you are done with your notebook or notebooks, you can shut them down, this helps free

up the memory that the notebook is using. To do this, click on the stop icon on the side bar, it is normally the second icon.
You can either terminate all sessions at once or you can terminate or shut them down individually. After you shut down the notebook session, you will see “no kernel” at the top right of the notebook confirming it is no longer active, and you can now close the tabs. You should now be familiar with: Run, delete, and insert a code cell; how to run multiple notebooks at the same time; Jupyter’s capabilities for presenting a notebook using a combination of Markdown and code cells; and how to shut down your sessions after you are done.

Jupyter Kernels

In this video, you will learn about kernels in the Jupyter environment. A notebook kernel is a computational engine that executes the code contained in a Notebook file. Jupyter Kernels for many other languages exist and we will explore some of the ones that are relevant in Data Science. When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed, the kernel performs the computation and produces the results.
Depending on the environment you are using, you may need to install other notebook languages in your Jupyter environment. In the skills network lab environment, a few of the languages have been pre-installed for you.
The first one is the Python kernel.When you launch a notebook, pick the language you are interested in for your Data Science project. The Python kernel allows you to run Python cells. You can run the Python script to produce results. On the top right corner, it tells you what kind of kernel it is, in this case it is a **Python kernel**.
You can also run other kernels, either by choosing the preferred kernel on the launch page or by clicking the top right to switch the kernel to the preferred kernel and picking the kernel from the drop-down menu. If you are running it on your local machine, you may need to manually install the languages through your command line interface.
The skills network virtual Jupyter environment has Apache, Julia, R and Swift. Depending on your preference and type of project you can use any of the languages that the Jupyter Notebook supports to execute your code. You should now be familiar with: What a kernel is, how the Jupyter Notebook supports different languages, and how to switch to a different kernel.

Jupyter Architecture

In this video, you will learn about the basic architectural design of the Jupyter ecosystem. Jupyter implements a two-process model, with a kernel and a client.

- The client is the interface offering the user the ability to send the code to a kernel.
- The kernel executes the code and returns the result to the client for display.

The client is the browser when using a Jupyter notebook. Jupyter notebooks represent your code, metadata, contents, and outputs. When saved it uses a dot I Pi NB (.ipynb) extension and a JSON structure. When you, the user, saves it, it is sent from your browser to the Notebook server which saves the notebook file on a disk as a JSON file with a dot I PI NB (.ipynb) extension.
The Notebook server is responsible for saving and loading the notebooks. The kernel gets sent the cells of code when the user runs them. Jupyter also has an architecture of how it converts files to other formats. It uses a tool called NB convert. For example, if we want to convert a Notebook file into an HTML file, it will go through the following:

- The Notebook is modified by a preprocessor, an exporter converts the notebook to the new file format, and a postprocessor will work on the file produced by exporting it.
- After conversion, when you give the URL of the HTML file, it first fetches the notebook, converts it HTML, and displays the file to you in a HTML file.

You should now be familiar with: The two-process model implementation of Jupyter, how Notebook servers communicate with kernels and clients, and the architectural design of how notebook files are converted to other files.

Lab - Jupyter Notebook - The Basics

Objective

In this hands-on lab you will learn how to create a different markdown cells.

Special note about the lab environment:

These Jupyter notebooks labs now use the latest environment: **JupyterLab**, which uses the same .ipynb Jupyter notebook files.

Exercise 1 - Create a new notebook in Python

1 In the next section click on Open Tool to access the Skills Network virtual lab environment.

Exercise 2 - Write and execute code

1 In your new empty notebook (from Exercise 1), click within the gray code cell and write some code, like 1 + 1.

2 Execute the code, by either clicking the Play button in the menu above the notebook, or by pressing Shift+Enter on your notebook.

3 You should see in the output, 2.

4 Try executing other code (try simple math operations).

Great! Now you know how to write and run code in Jupyter Notebooks.

Exercise 3 - Create new cells

1 In your Jupyter notebook, first **click on any of the existing cells** to select the cell.

2 From the menu, click on *Insert*, then *Insert Cell Above* or *Insert Cell Below*.

3 Great! Now you know how to insert new cells in Jupyter Notebooks. Note you can use the keyboard shortcuts: [a] - Insert a Cell Above; [b] - Insert a Cell Below.

Exercise 4 - Create Markdown cells and add text

In your notebook, click on any code cell, and in the drop-down menu in the menu above, change the cell type from *Code* to *Markdown*. As you'll notice, you cannot create Markdown cells without first creating cells and converting them from *Code* to *Markdown*.

In the Markdown cell, write some text like *My Title*.

To render the Markdown text, make sure the cell is selected (by clicking within it), and press Play in the menu, or Shift+Enter.

Your Markdown cell should now be rendered!

To edit your Markdown cell, double-click anywhere within the cell. Note you can use the keyboard shortcut:

[m] - Convert Cell to Markdown

Lab - Jupyter Notebook - More Features

Objective

In this hands-on lab you will learn how to create a different markdown cells.

This lab is a continuation of the previous lab.

Note: If you have JupyterLab on Skills Network Labs open in a tab already, continue using that tab. Otherwise, click on Open Tool button in the next section to launch JupyterLab environment.

Exercise 5 - Rename your Notebook

1 In the list of notebooks in the right-hand panel, click on the File at the top to expand the list of options.

2 Click on Rename Notebook... to rename your notebook to something like *My_Notebook.ipynb*.

Exercise 6 - Save and Download your Jupyter Notebook from Skills Network Labs to your computer

Although your notebooks and data is preserved in your account after signing out, sometimes you may want to download your notebook to your computer.

- To save, click on the Save Notebook or Save Notebook as button in the File menu.
- To download the notebook, click on the File and scroll down to Download to download the notebook. You can also right-click on the notebook in the file browser pane and click Download

Exercise 7 - Upload a Jupyter Notebook to Skills Network Labs

- To upload a Jupyter Notebook from your computer to Skills Network Labs, drag and drop a .ipynb file from your computer directly into the browser. You can use the notebook you downloaded from Exercise 6.
- Your notebook should open up automatically once it has finished uploading.

Exercise 8 - Change your kernel (to Python 3, R) in JupyterLab

With a notebook open, click on the kernel name (e.g., Python 3) in the top-righthand corner of

the notebook to open up a pop-up window to change it to a different language.

Lab - Jupyter Notebook - Advanced Features

Effort: 20 mins

Objective

In this hands-on lab you will learn how to create a different markdown cells.

This lab is a continuation of the previous lab.

Note: If you have JupyterLab on Skills Network Labs open in a tab already, continue using that tab. Otherwise, click on Open Tool button in the next section to launch JupyterLab environment.

Exercise 9 - Advanced Markdown styling

- To get start using the Markdown feature, click on the drop-down menu at the top and click on Markdown.
- You can write HTML code in your Markdown cells. We can embed links, try executing the following HTML code in a Markdown cell: <a href= https://www.cognitiveclass.ai/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDS0105ENSkillsNetwork20083975-2021-01-01>Cognitive Class by right-clicking and opeing in a new tab. You should now see a hyperlink to https://www.cognitiveclass.ai that appears as **Cognitive Class**.
- You can also use Markdown formatting. For example, to use an H1 Header to write a title, try running the following in a Markdown cell: # Header 1. You can find more rules for Markdown formatting in this Markdown Cheatsheet: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
- You can also create HTML tables. To create a table like this, by doing the following:

header	header
cell	cell

Exercise 10 - Advanced Markdown styling: Create tables

- Using only Markdown, try creating a table with **three columns** and **four rows**.

Jupyter Notebooks on the Internet

Jupiter Notebooks on the Internet

RStudio IDE

Introduction to R and RStudio

Welcome to a gentle introduction to R and RStudio. This tutorial introduces you to the R programming language and its environment. The videos following will contain tutorials and hands-on practice. R is a statistical programming language. It is a powerful tool for data processing and manipulation, statistical inference, data analysis, and Machine Learning algorithms.

Based on 2017 Stack overflow visits, they found that R is most used by academics, healthcare, and the government. R has many functions; which support importing data from different sources. For example, flat files, databases, the web, and statistical software like SPSS and STATA. R is a preferred language for some Data Scientists because it is easy to use the functions within R. It is also known for producing great visualizations and readily available packages to handle data analysis without the need to install any libraries. To use R, we need an environment to help run the codes.

One of the most popular environments for developing and running R language source code and programs is Rstudio. Rstudio is an integrated development environment that helps improve and increase productivity with the R language. R studio includes:

- A syntax-highlighting editor that supports direct code execution and where you keep a record of your work; a console for typing R commands;
- a workspace and history tab that shows the list of R objects you created during your R session and shows the history of all previous commands;
- a Plots, Files, Packages, and Help tab for showing files in your working directory; history of plots you have created and allow for exporting plots to PDF or images files;

External R packages available on your local computer and help on R resources, RStudio support, packages, and more. If R is your tool choice for Data Science, here are some popular Libraries in the Data Science community:

- dplyr for manipulating data,
- stringer for manipulating strings,
- ggplot for visualizing data,
- caret for Machine Learning To get you up and learning quickly,

we have provided you with an R Studio virtual environment as part of the Skills Network Labs. This virtual lab environment is designed to assist you to easily practice what you learn in the course and skip the need to create an account or download or install anything. You should now be familiar with: The capabilities of R and its uses in Data Science The RStudio interface for running R codes and Popular R packages for Data Science

Plotting within RStudio

With the increasing amount of data, one of our many jobs as data scientists is to produce

insights using visualizations. R is a very great tool for data visualization and has different packages. Some of the popular and top data visualization tools include:

- ggplot which used for data visualizations such as histograms, bar charts, scatterplots etc. It allows adding layers and components on a single visualization.
- Plotly is an R package can be used to create web-based data visualizations that can be displayed or saved as individual HTML files.
- Lattice is a data visualization tool that is used to implement complex, multi-variable data sets. Lattice is a high-level data visualization library; it can handle many of the typical graphics without needing many customizations.
- Leaflet is very useful in creating interactive plots.

Depending on your need and data science project, most of these libraries and packages can come in handy. To install these packages in your R environment, use the `install.packages("Package name")` command.

R also has its own inbuilt functions to create plots and visualization, it is mostly used using the plot function. For example, the function above is the simplest form of the plot function and it returns a scatterplot of the values vs the index as shown here. You can also add lines to the function and add a title to make it easier to read and understand. Here we add a line to the plot and to add a title. We specify the title function, and we can see we have added a line, and a title.

We can create more beautiful visualizations using the ggplot libraries. ggplot is a data visualization package for R. It can handle complex requests allowing for you to add layers into plots by tweaking the functions and arguments.

To create a scatter plot, we will use the inbuilt dataset Mtcars. We will first read ggplot library into memory by using the library function and then call ggplot on the dataframe MTcars, specify the X axis as miles per gallon and Y axis as weight, and then add the geom point function to specify to ggplot that we want a scatter plot, otherwise it will return an empty plot. It will produce a more beautiful and easier to read the plot like this You can also add titles and tweak the axis name by using the ggtitle argument to add a title. And the labs argument to change the names of the axis by specifying the new names we want to see. We will see an easier to read and understand graph.

In our lab we will be recreating the above graphics with ggplot and the extension library called GGally. GGally extends ggplot2 by adding several functions to reduce the complexity of combining geometric objects with transformed data. Please type and execute the following command in the console window to install both packages. You will not need any coding experience as the code and commands will be given to you.

You are now familiar with: Popular data visualization packages in R, plotting with Base R plot function, plotting with ggplot, adding a title, and changing the axis name using the ggtitle and labs function.

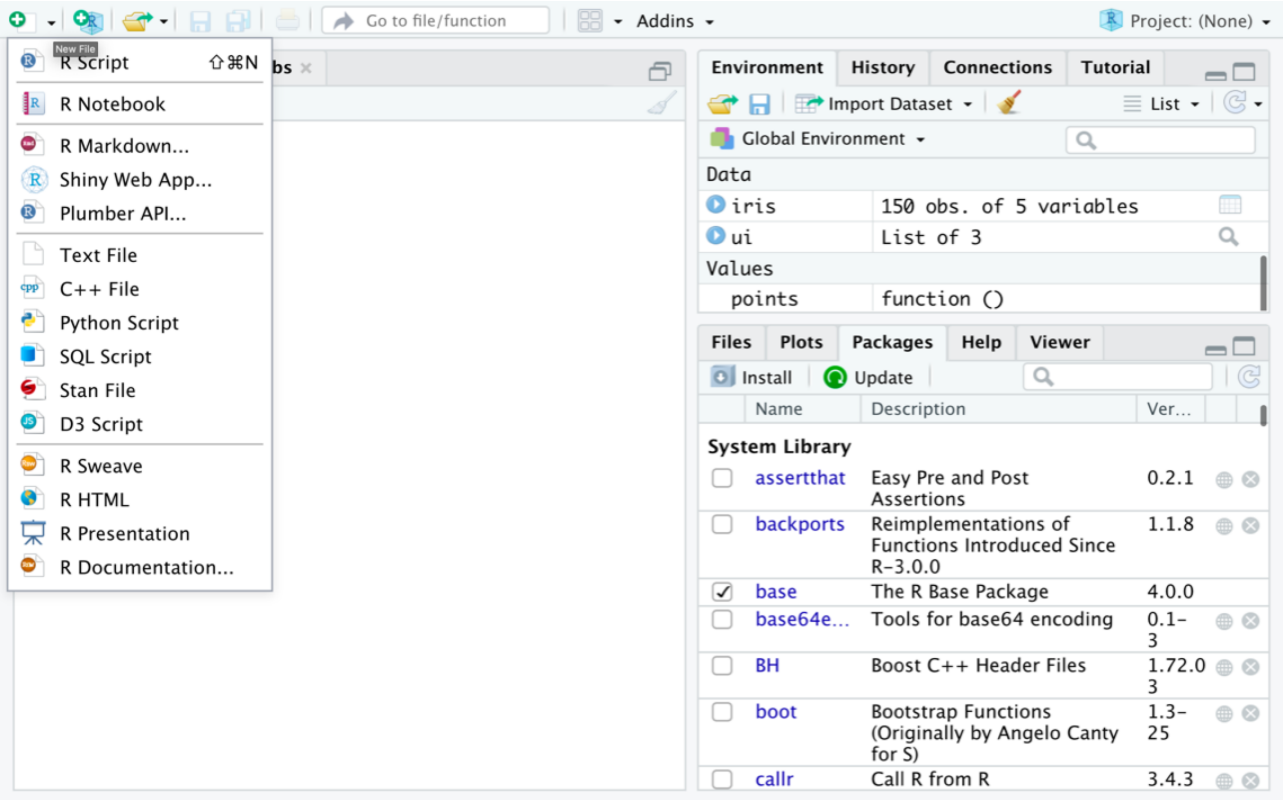
Getting started with RStudio and installing packages

Objective for Exercise:

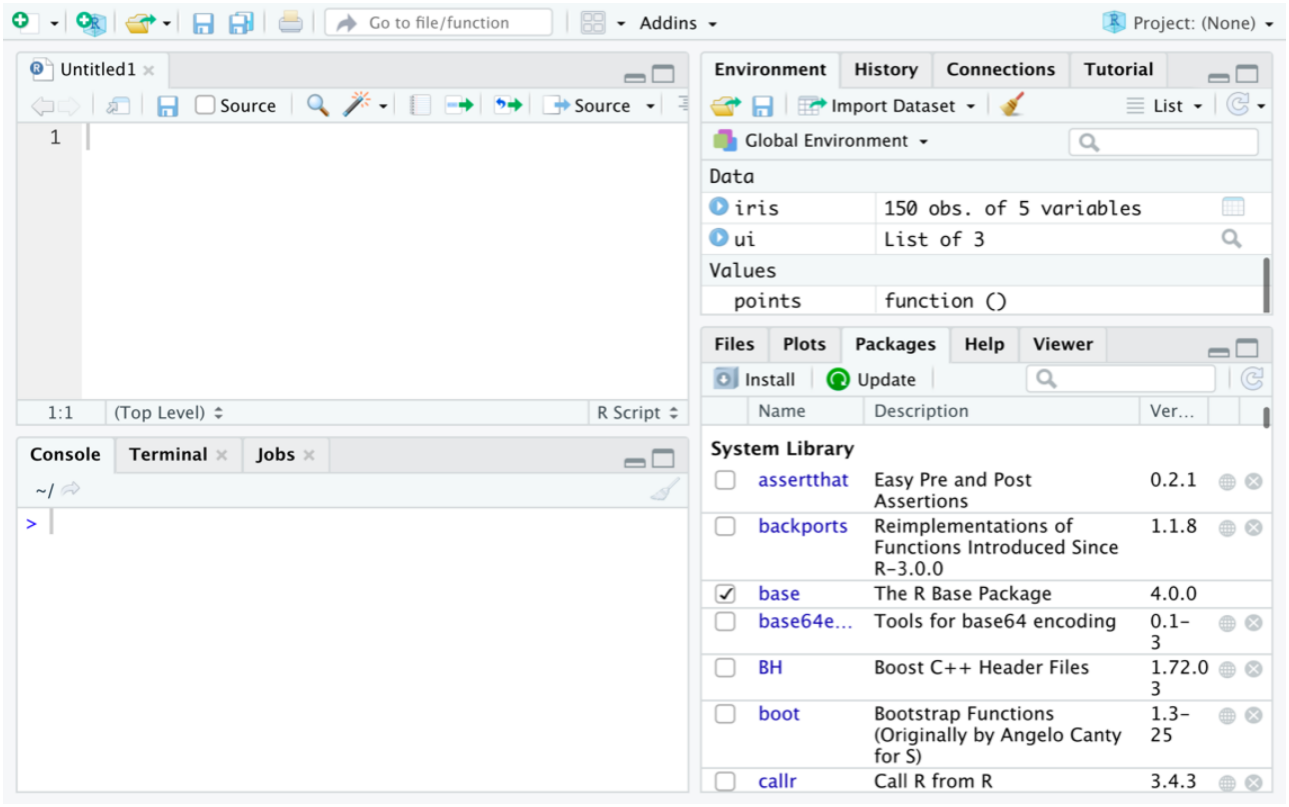
This lab introduces you to R and RStudio

Exercise:

1. Click in the tiny plus symbol top left and select R Script

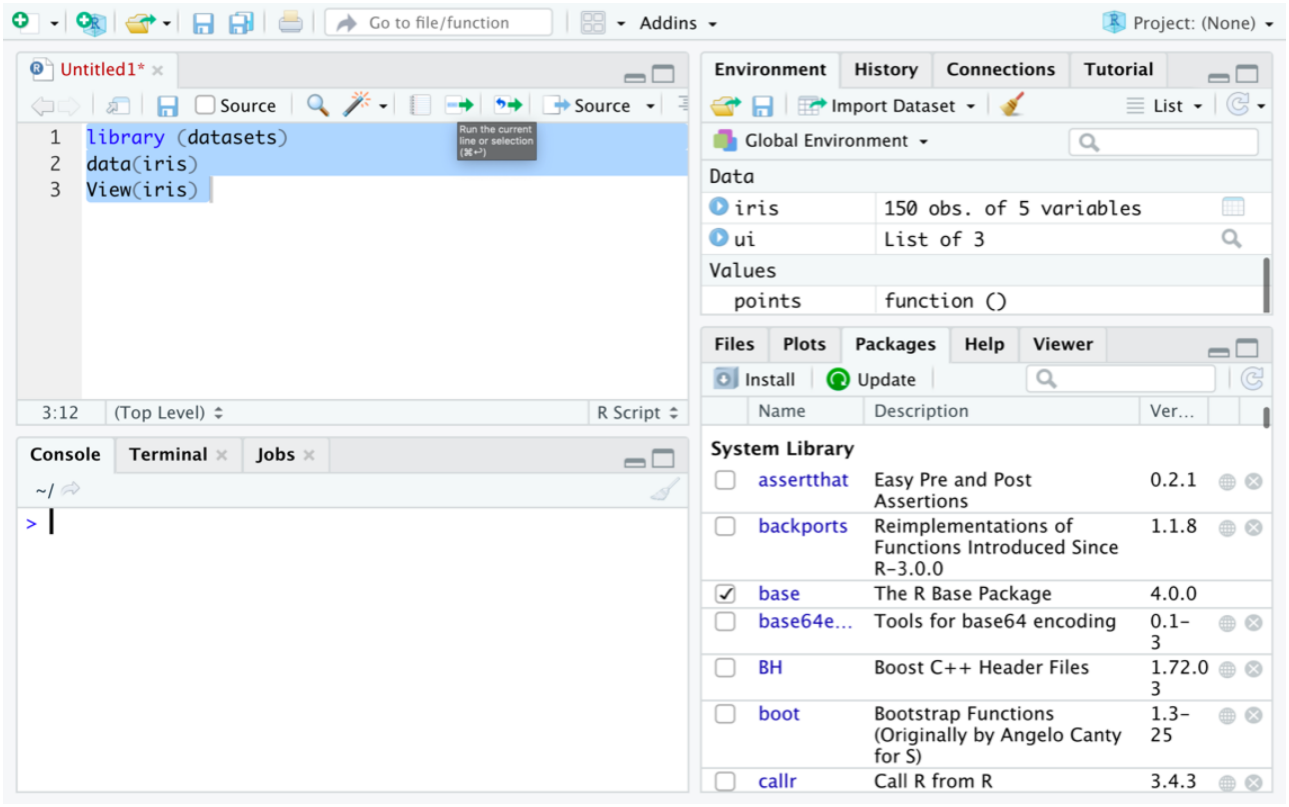


2. An untitled R Script panel opens. It would look like this.

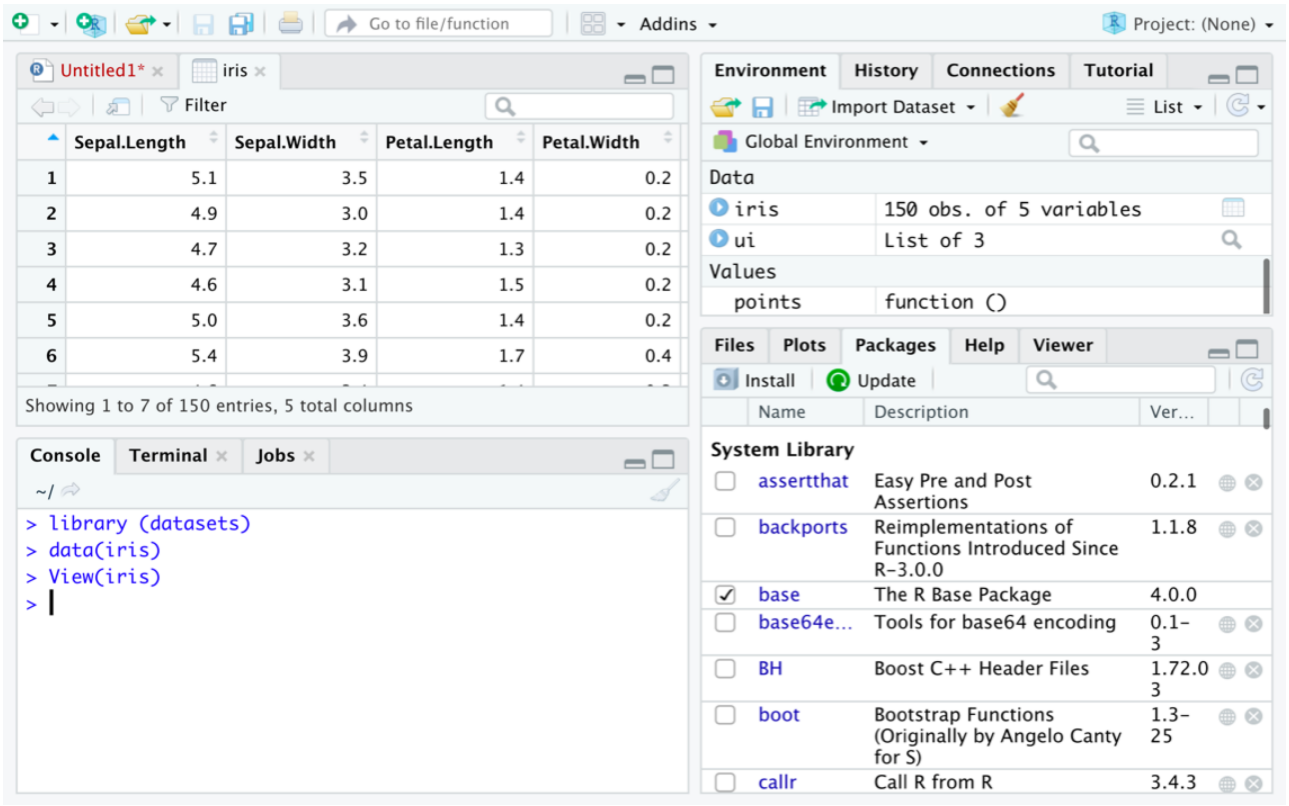


3. Now we load the iris dataset. Please enter the following lines into the editor window which just appeared. Then select them all such that they turn blue. Then click on the tiny Run icon just above the editor window.

```
library (datasets)
data(iris)
View(iris)
```



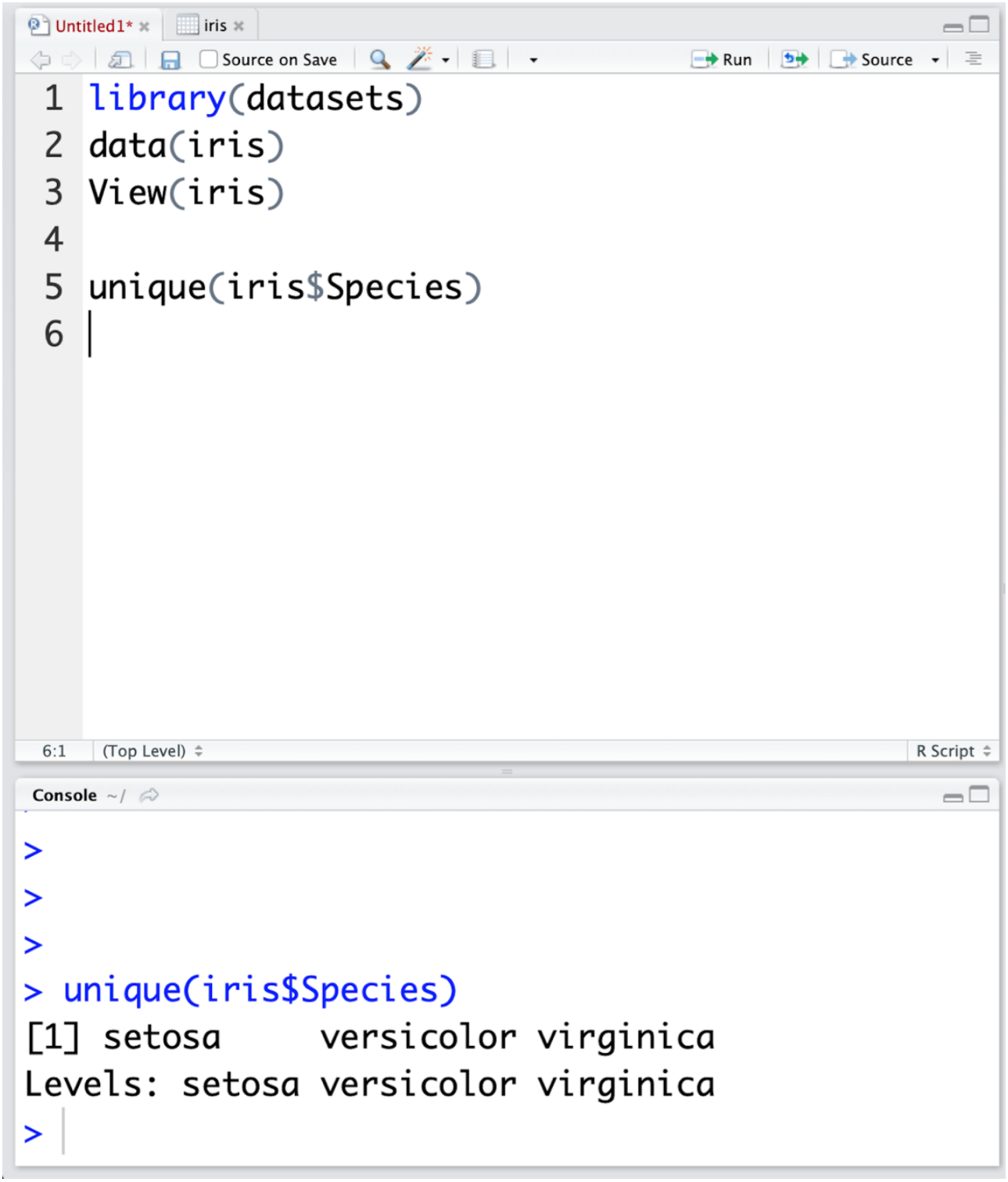
4. You are directly taken to the data view tab to inspect your dataset. We can see that there are five columns in this data set and that the first four are floating point and the last one is label of data type string which contains the category value of our data set. We also see that we have 150 entries in total of which we are seeing the first 19.



5. Now let's find how many different species there are present in the data set. Type the

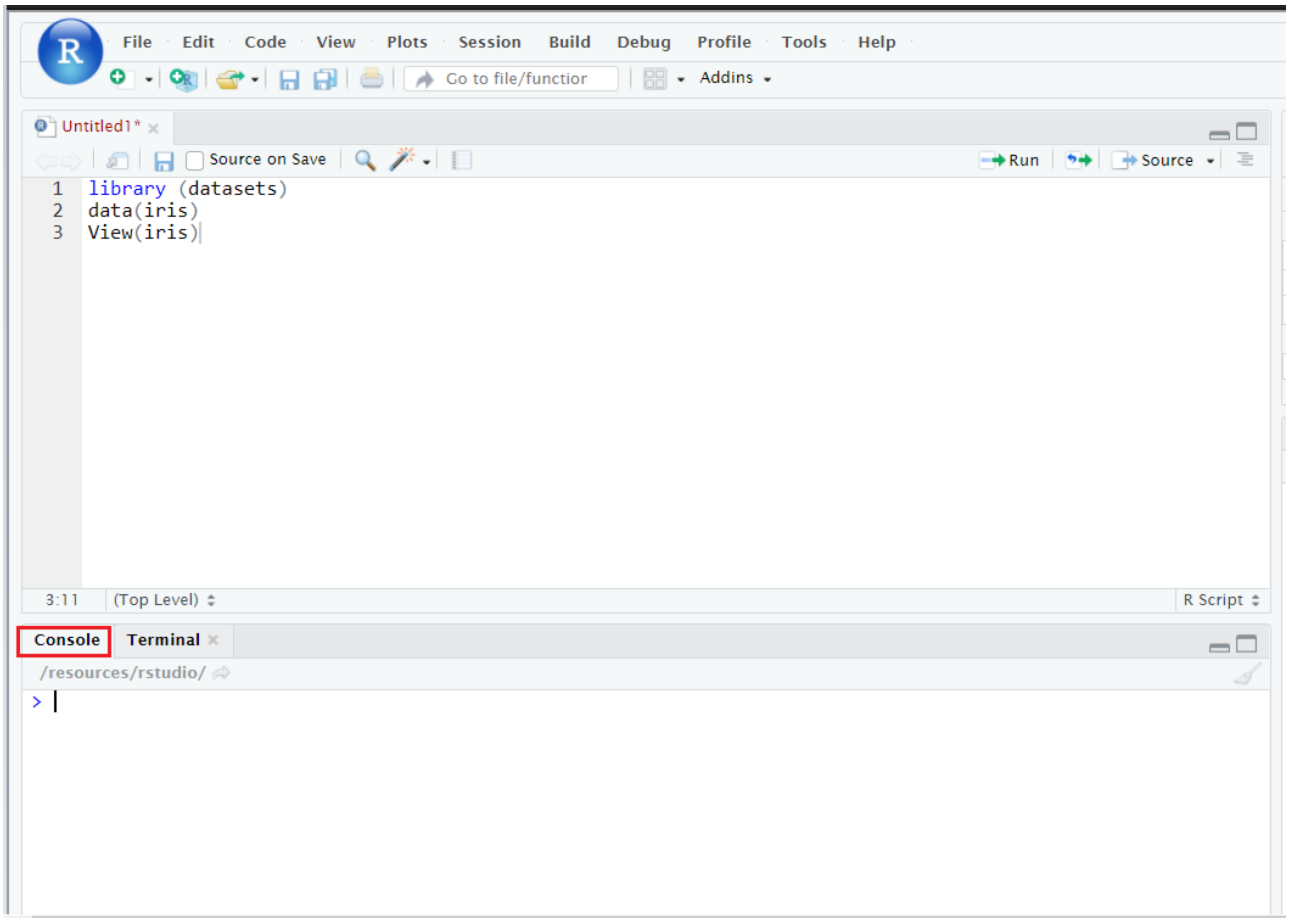
following command into the editor window and click the run.

unique(iris$Species)



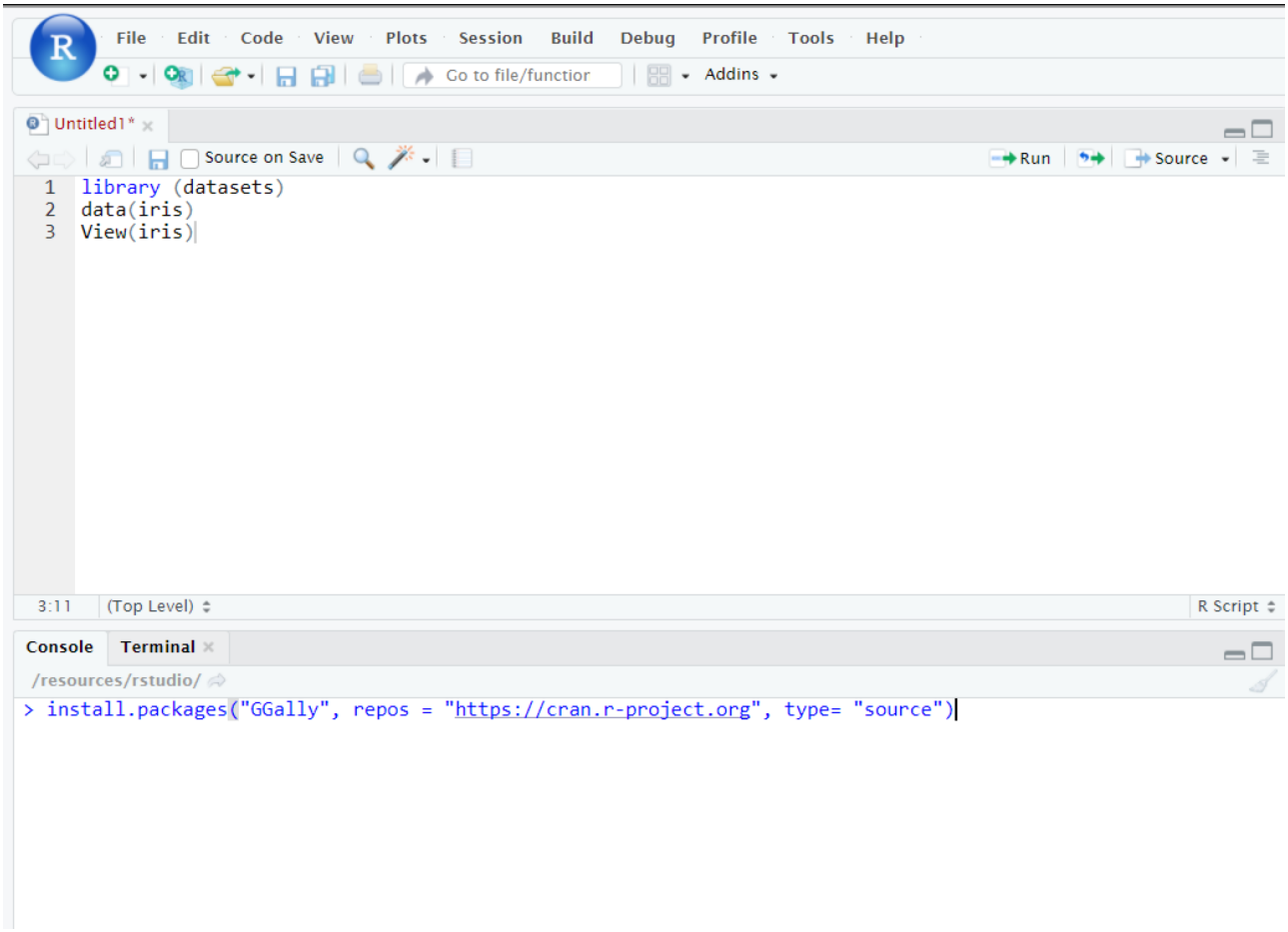
In the Console window at the bottom you'll see the result of the executed command and will know that there are only three different species present in the data set.

6. Now it's time to look into the data set in more detail.
7. Open a Console.



8. Run the following command in the console.

```
install.packages("GGally", repos = "https://cran.r-project.org", type= "source")
```



9. Click *Enter* to install the packages.

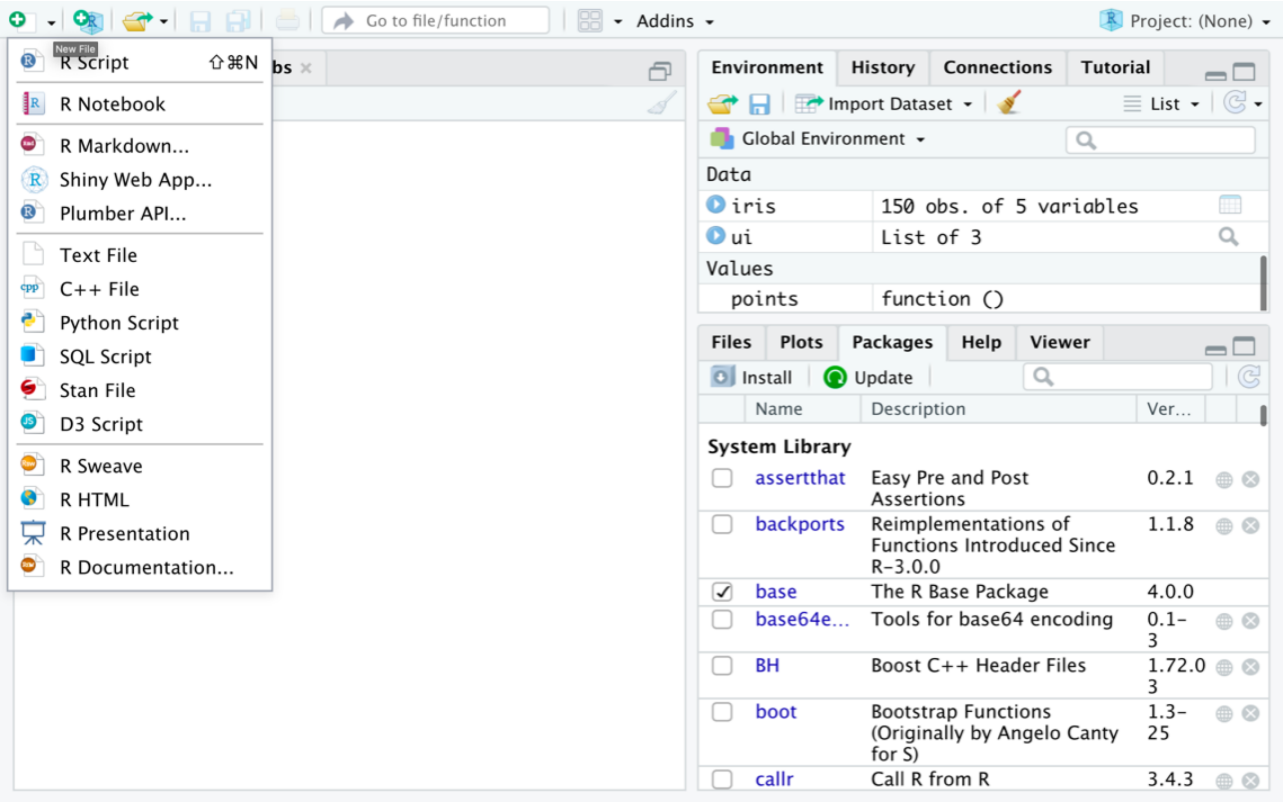
Plotting with RStudio

Objective for Exercise:

This lab introduces you to plotting in R with ggplot and GGally. GGally is an extension of ggplot2

Exercise:

1. Click in the tiny plus symbol top left and select R Script to create a new R script if you dont have one open already.



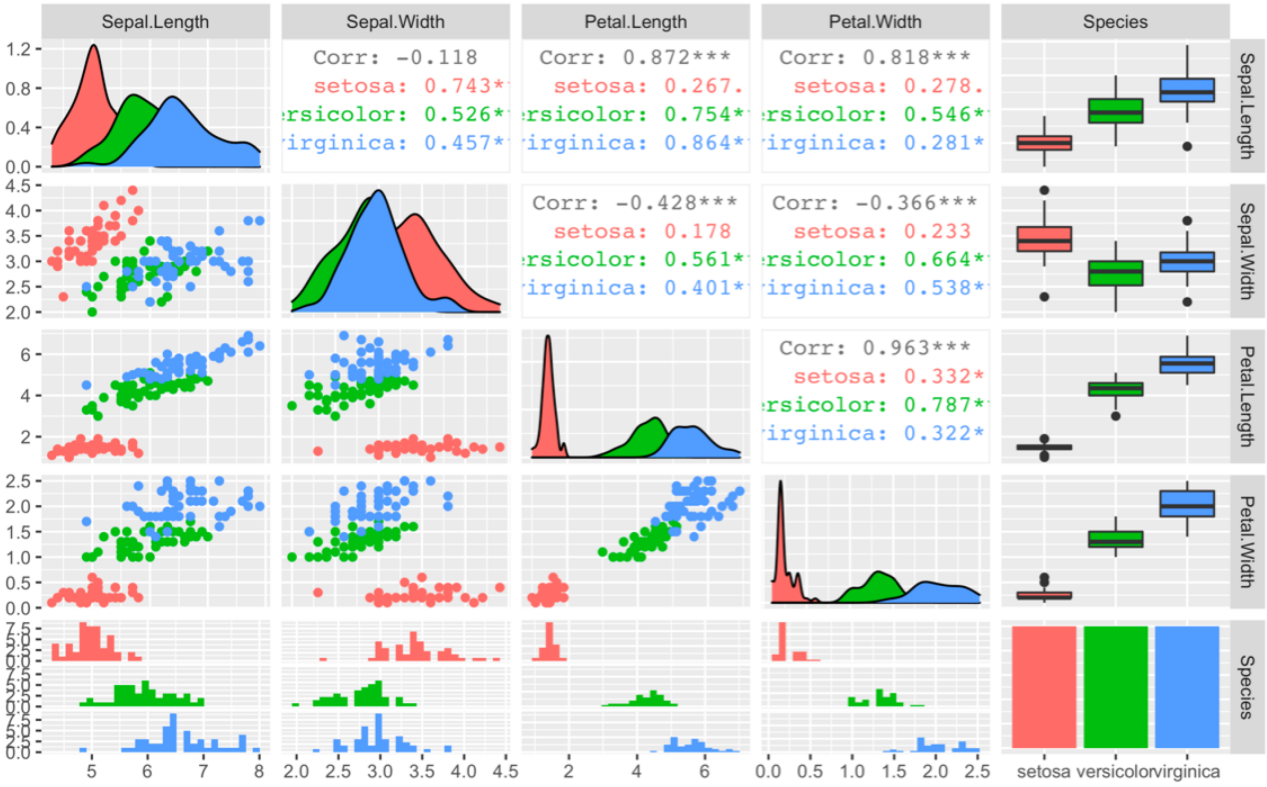
2. We will use the iris dataset. If you don't have it loaded, please copy and paste the following into your R script file.

library(datasets)
data(iris)

3. In the previous lab, you installed the libraries necessary to create some nice plots let's execute the following commands:

library(GGally)
ggpairs(iris, mapping=ggplot2::aes(colour = Species))

4. Select the commands and click on run on the top. You'll now see the following plot in the **plots** window:
5. Click on **Zoom** icon on the plot window to zoom and see the plot.



6. This gives us a lot of information for a single line of code. First, we see the data distributions per column and species on the diagonal. Then we see all pair-wise scatter plots on the tiles left to the diagonal, again broken down by color. It is, for example, obvious to see that a line can be drawn to separate **setosa** against **versicolor** and **virginica**. In later courses, we'll of course teach how the overlapping species can be separated as well. This is called supervised machine learning using non-linear classifiers by the way. Then you see the correlation between individual columns in the tiles right to the diagonal which confirms our thoughts that **setose** is more different, hence more easy to distinguish, than **versicolor** and **virginica** since a correlation value close to one signifies high similarity whereas a value closer to zero signifies less similarity. The remaining plots on the right are called **box-plots** and the ones at the bottom are called **histograms** bit we won't go into detail here and save this for a more advanced course in this series.