# Learning R for Data Visualization

**Section 1: Introducing Scientific Plotting in R**

**1.1 The Course Overview**

This video provides an overview of the entire course.

**1.2 Preview of R Plotting Functionalities**

Creating professional looking plots, both static and interactive, may seem hard; however, with R we can create and fully customize plots with a few lines of code.

* Use ggplot2 that can produce beautiful plots with a few lines of code
* Plots are fully customizable; to obtain perfect plots every time, use customizable plots.
* Create complex results using interactive plots

**1.3 Introducing the Dataset**

Often, beginners fail to properly understand their dataset before analyzing it. However, a good understanding of the origin and structure of the data is of primary importance.

* Introduce the data provider, that is, EPA.
* Understand the EPA network of stations
* A detailed description of the data structure

**1.4 Loading Tables and CSV Files**

It is not always good to import data in R using the default settings. For doing it successfully, several parameters need to be set.

* Set the working directory
* Understand the important setting of the function read.table
* Import the data and check the structure

**1.5 Loading Excel Files**

Importing Excel tables in R may be tricky. However, with the right explanation the proper package can be installed and everything should work out fine.

* Install the package xlsx
* Understand the format of the code to import Excel files
* Import and check the data

**1.6 Exporting Data**

Exporting data in R may seem difficult, since we have many options to choose from. However, R has powerful exporting functions that with few options can do the job successfully.

* Firstly, we need to subset our data to have something to export
* Then, we can learn how to export data in R
* The final step would be exporting data into multiple Excel sheets

**Section 2: Scientific Plotting in ggplot2**

**2.1 Creating Histograms**

Producing elegant plots in ggplot2 may seem difficult but it is actually quite easy to do. In fact, ggplot2 takes care, by default, of most of the graphical design of the plot, meaning that we can produce beautiful histograms with just a few lines of code.

* Load ggplot2 and then import the dataset
* Plot a simple histogram, using the default settings.
* Plot multiple distributions with faceting

**2.2 The Importance of Box Plots**

Histograms are useful for certain tasks, but for comparing several variables at once they are not the best. Box plots can be used instead, since they allow the comparison of the distribution of multiple variables side by side.

* Explain what a box plot is and what does it represents
* Create multiple box plots with just two lines of code
* Order the plot to achieve better results

**2.3 Plotting Bar Charts**

Categorical variables are invariably difficult to visualize in meaningful ways. Bar charts are important for plotting categorical variables and defining their characteristics.

* Learn bar charts
* Create simple bar charts in ggplot2
* Learn how to automatically order a data.frame and plot ordered bar charts

**2.4 Plotting Multiple Variables – Scatterplots**

In many cases, we are interested in comparing multiple variables at once and checking their correlation. Scatterplots allow us to do just that and are an important tool in a data analyst's toolbox.

* Describe the importance of scatterplots
* Create simple scatterplots in ggplot2
* Create more complex visualization by tweaking some basic options

**2.5 Dealing with Time – Time-series Plots**

In many cases, the variable time is underestimated. However, time-series are extremely useful to determine the temporal pattern of a variable.

* Understand the structure of time-series plots
* Plot a simple time-series plot in ggplot2
* Customize the plots with color and size

**2.6 Handling Uncertainty**

Many datasets are affected by uncertainty and people not always know how to show this in plots. This video will present ways to solve this and take uncertainty into account.

* Understand how to handle uncertainty
* Present simple ways to include uncertainty in bar-charts
* Present the scatterplots with double error bars

**Section 3: Customizing Plots**

**3.1 Changing Theme**

By default, ggplot2 creates plots with a grayish background, and without axes lines and white gridlines. This is not the standard look you normally find in scientific manuscripts.

* Explain the graphical elements of the standard theme
* Change the default theme
* Explore the differences between the default theme and the others

**3.2 Changing Colors**

The default color scale is not always appropriate to spot all the differences in the data we are trying to plot. In many cases, we have to change it so that our plots can become more informative.

* Change the default two colors for plotting continuous variables
* Explore ways to include more colors in the color scale
* Present discrete color scale for categorical variables

**3.3 Modifying Axis and Labels**

ggplot2 uses the names of the columns as labels, meaning that if these are not self-explanatory, the plot will not provide a good framework to understand its meaning. By adding some lines of code, we can customize the plot in order to change the labels and make it clearer.

* Add a title for the plot
* Change the title of the legend
* Change the axes labels

**3.4 Adding Supplementary Elements**

The default plots created by ggplot2 lack several elements that in many cases are useful to provide additional information to viewers. However, there are simple functions that can be used to add supplementary elements to the plot.

* Add the trend lines to scatterplots
* Learn how to add vertical and horizontal lines to plots
* Customize the lines

**3.5 Adding Text Inside and Outside of the Plot**

In many cases, it is crucial to be able to include textual labels on plots to provide viewers with additional information. This can be done in ggplot2 in both static and dynamic ways.

* Add fixed text labels
* Add dynamic textual labels
* Add text outside the plot and change the axis labels

**3.6 Multi-plots**

With the function facet\_wrap, it is only possible to create a grid of plots of the same type. However, in some cases, it is necessary to create side-by-side graphs with diverse plots. This can be done in the package gridExtra.

* Review the facet\_wrap function
* Install the gridExtra package
* Create the multi plots

**Section 4: Exporting Plots**

**4.1 Exporting Plots as Images**

We could easily save our plots as images directly from R Studio. This way of saving however, does not provide much flexibility. If we want to customize our images, we need to learn how to export plots from the R code.

* Create an object with the plot we want to save
* Learn the basics of the ggsave function
* Change the size of the image

**4.2 Adjusting the Page Size**

The default size that ggplot2 uses to save plots is ideal for most of our needs, such as embedding plots in Word documents. However, in some cases, we may need to specify a particular page size for our plots, which can be easily done with the option paper.

* Specify the page size
* Rotate the page
* Specify other options

**Section 5: Interactive Plots in rCharts**

**5.1 Getting Started with Interactive Plotting**

Static plots are the standard for publishing in traditional media, such as journal papers. However, the world is moving towards an internet-based presentation of results and even scientific journals are quickly adapting it. Many now offer the possibility of including interactive plots. In R, we can create plots for the Web with the rCharts package, which is a bit more difficult to install than ggplot2.

* Explain the rCharts package
* Install devtools
* Install rCharts from GitHub

**5.2 Creating Interactive Histograms and Box Plots**

rCharts features a syntax more similar to standard plotting in R than what we saw with ggplot2. However, it is easy to pick up by showing simple examples and then including additional details.

* Explain the syntax of rCharts
* Include more details
* Add JavaScript functions for more flexibility

**5.3 Plotting Interactive Bar Charts**

Even though we know nothing about HTML and CSS, we can still obtain beautiful bar-charts using templates created by other users.

* Plot basic interactive bar charts
* Add axis labels
* Use a template for an elegant finish

**5.4 Creating Interactive Scatterplots**

If too many data points are present in our dataset, scatterplot visualization may become very confusing in static plots. However, in interactive plots this limitation no longer applies, since we can select to visualize only some datasets.

* Create basic interactive scatterplots
* Understand the interactivity
* Add elements and controls

**5.5 Developing Interactive Time-series Plots and Saving**

Time-series plots are a great way to visualize the temporal pattern of a variable. However, sometimes we cannot fully understand the exact date of each point based only on the values on the x axis. Interactive visualization can solve this problem by adding tooltips in which we can take a look at the raw data.

* Set the data in the correct format
* Plot a basic time-series plot
* Add elements

**Section 6: Creating a Website with Shiny**

**6.1 Getting Started with Shiny**

Shiny is a package to build fully featured websites from scratch in R. The way it communicates between the user interface and the server may seem difficult to understand. However, with some explanation, understanding Shiny becomes very easy and intuitive.

* Introduce the Shiny package
* Explain the tutorial and examples
* Understand the basic structure of a Shiny website

**6.2 Creating a Simple Website**

Understanding the structure of a Shiny website is very important. However, presenting it from a website is not enough for the viewers to replicate it. Therefore, in this video, we are going to create a simple website with data and plots we already used, to further help viewers.

* Understand the basic structure of Shiny
* Add elements to UI and Server
* Test the website

**6.3 File Input**

If we plan to upload our Shiny website on-line, we need to implement a way for users to upload their own data. In this video, we are going to show how to do just that.

* Importing files in Shiny
* Simple code to do it
* Add a separator for more flexibility

**6.4 Conditional Panels – UI**

One of the key components of a successful website is the ability to respond to users’ interactions. This can be achieved with conditional panels, which change the UI based on users’ interactions.

* Explain conditional panels
* Understand UI modifications
* Apply server modifications

**6.5 Conditional Panels – Servers**

One of the key components of a successful website is the ability to respond to users’ interactions. This can be achieved with conditional panels, which change the UI based on users’ interactions.

* Modify server side modifications
* Keep track of the IDs
* Recognize variables automatically

**6.6 Deploying the Site**

So far, we have looked at ways to create and add elements to a Shiny website. However, sooner or later, this website needs to be deployed on the Internet so that everybody can use it. Here, you will learn how to do it using a free account on shinyapps.io.

* Separate ui.r and server.r
* Add plots to the script
* Finally, we deploy the site.