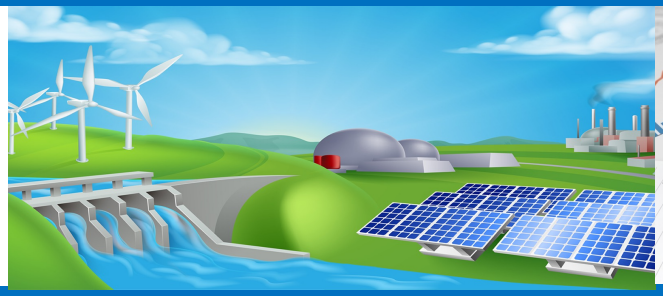



ENV790 – TIME SERIES ANALYSIS FOR ENERGY DATA

M1: Intro to TSA, R and RStudio

Luana Lima

Nicholas School of the Environment - Duke University

Learning Goals

- Introduction to Time Series Analysis (TSA)
  - What is TSA?
  - Examples
  - TSA Components (trend, cycle, seasonal, random)
- Introduction to R and RStudio
  - How to install packages?
  - How to create a scripts?
  - Importing and exporting Data
  - Graphs and plots

Introduction to Time Series Analysis

Meaning and definitions

Importance of TSA

Components of TSA

What is a Time Series?

- A set of observations on a variable collected over time
- Discrete and continuous time series
- Example: stock prices, interest rate, retail sales, electric power consumption, etc
- Mathematically representation: a time series is defined by the values $Y_1, Y_2, \ldots$ of a variable Y at times $t_1, t_2, \ldots$

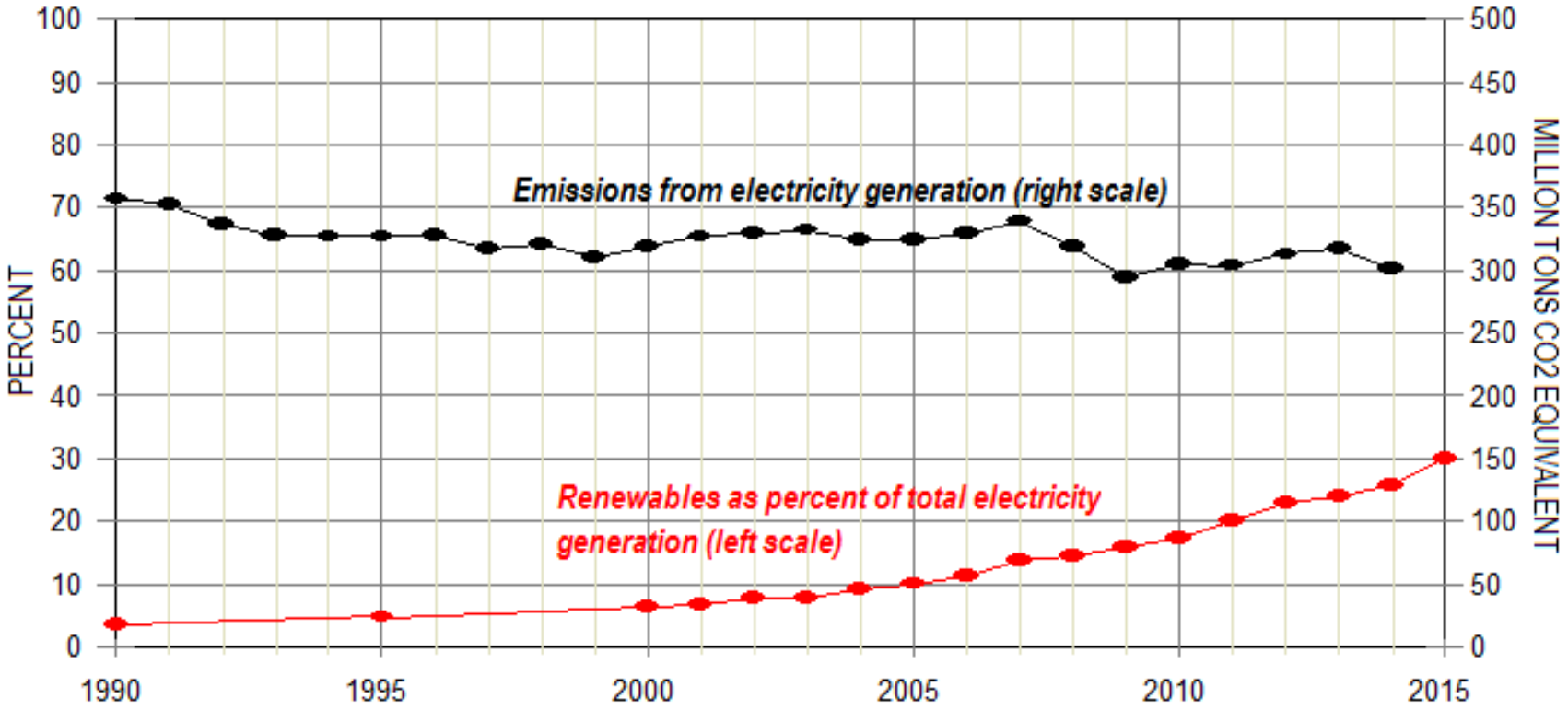
  Thus,

$$Y = F(t)$$

What is Time Series Analysis (TSA)?

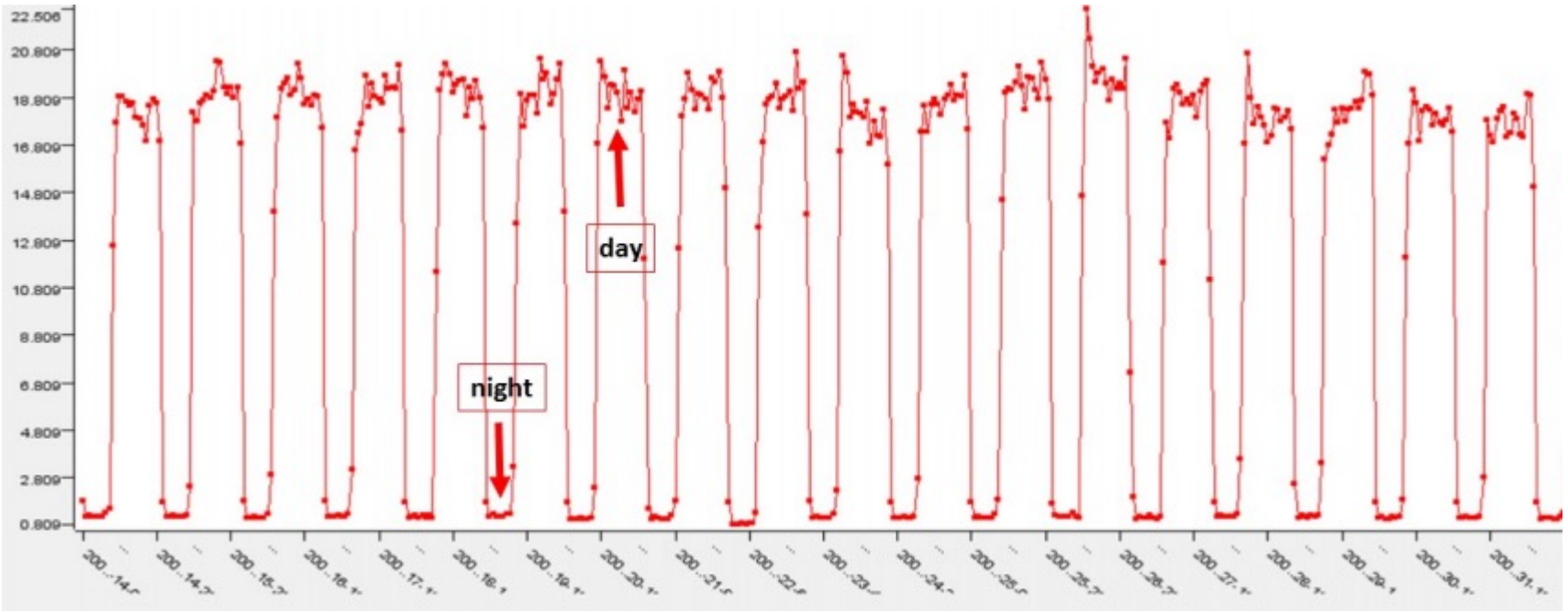
- In TSA, we analyze the past behavior of a variable in order to predict its future behavior
- Causes of variation of Time Series Data
  - Seasons, holidays, etc
  - Natural calamities: earthquake, epidemic, flood, drought, etc
  - Political movements or changes, war, etc

Example of Time Series Data



Percent renewables in Germany's electricity mix versus total greenhouse gas emissions, 1990-2015

Example of Time Series Data



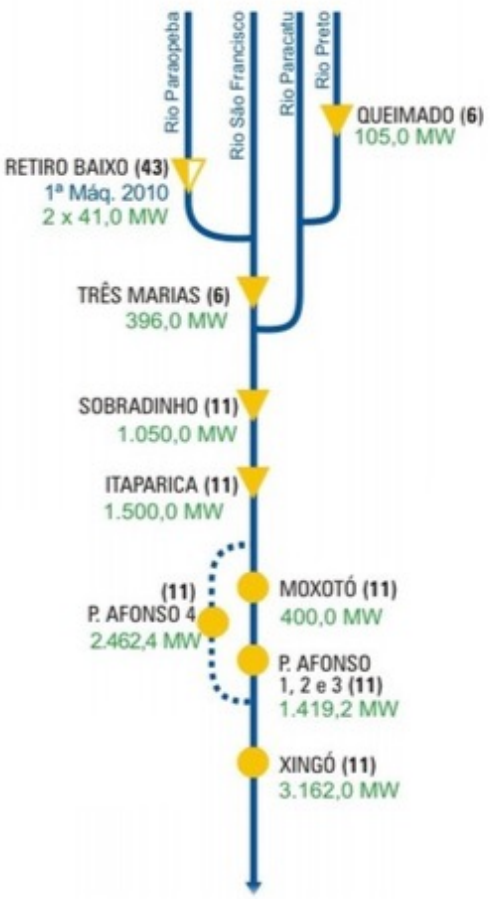
Energy consumption time-series for meter ID 1038. Notice the day/night rhythm.

Source: https://dzone.com/articles/data-chef-etl-battles-energy-consumption-time-seri

TSA Definitions

- An analysis of a single sequence of data is called univariate time-series analysis

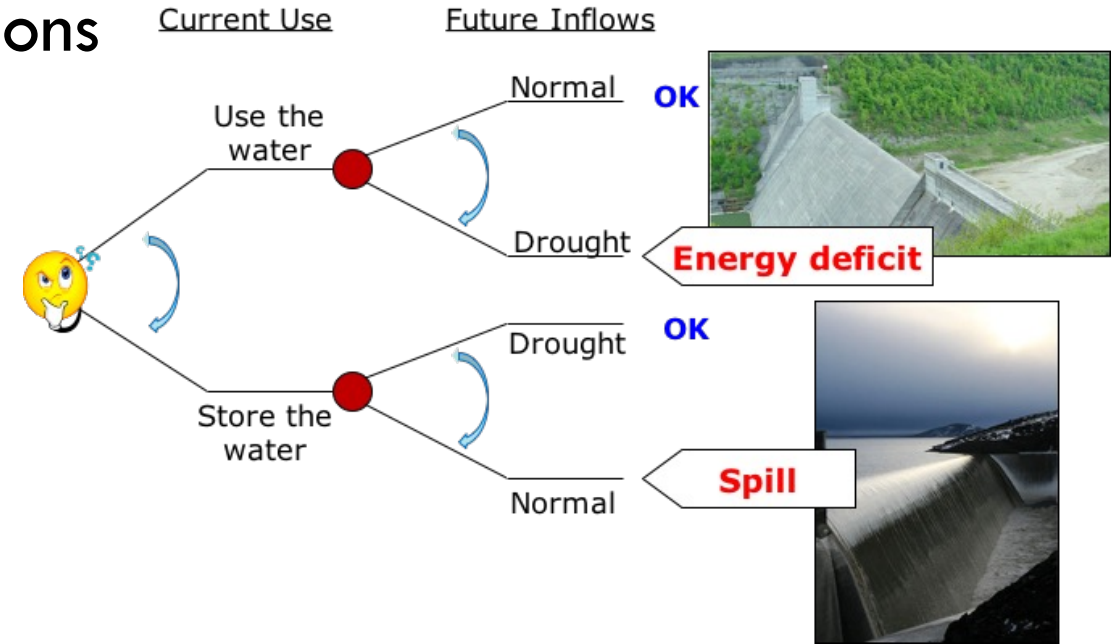
- An analysis of several sets of data for the same sequence of time periods is called multivariate time-series analysis



Importance of TSA

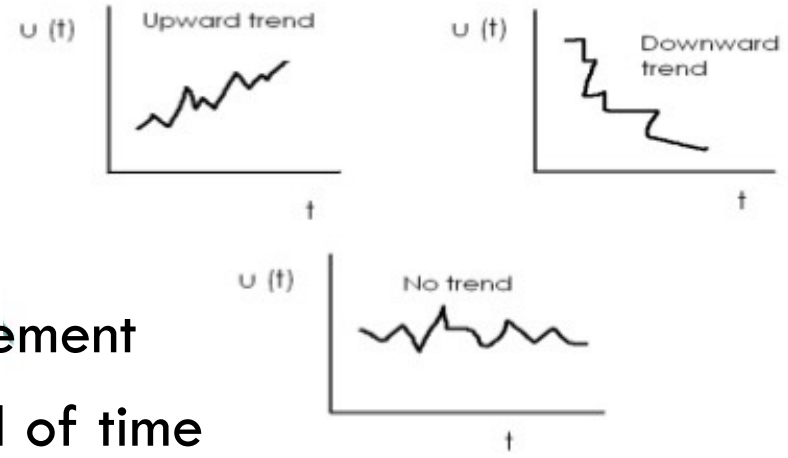
- Very popular tool for business forecasting
- Basis for understanding past behavior

- Can forecast future activities/planning for future operations



Components of TSA

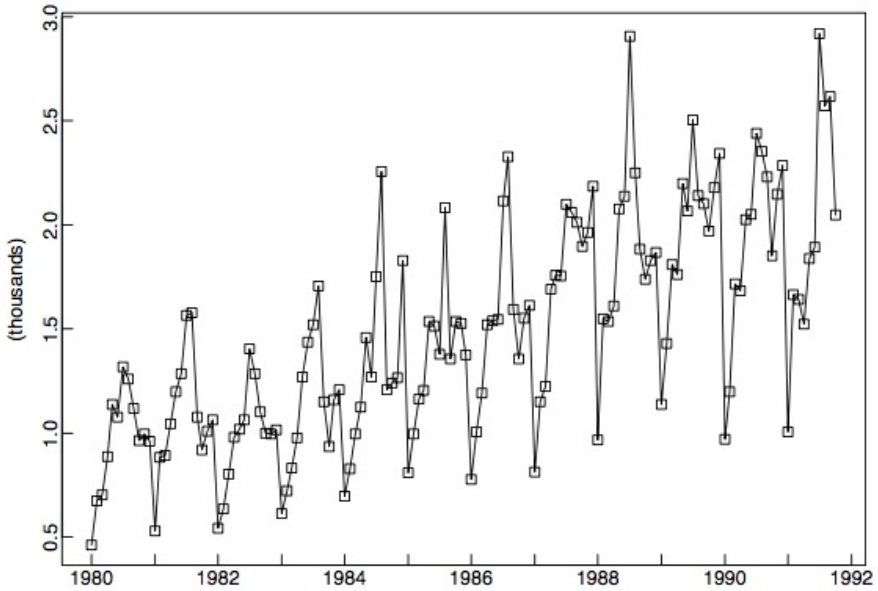
- Time frame: short, medium and long-term
  - How far can we predict?
- Trend
  - General tendency to grow or decline over a long period
  - Easiest to detect
  - Maybe linear or non-linear
- Cycle
  - An up and down repetitive movement
  - Repeat itself over a long period of time
  - Example: business cycle (prosperity, decline, depressions, recovery)



Components of TSA (cont'd)

- Seasonal Variation
  - An up and down repetitive movement occurring periodically (short duration)
  - Factor that cause seasonal variations: climate and weather condition or custom traditions and habits

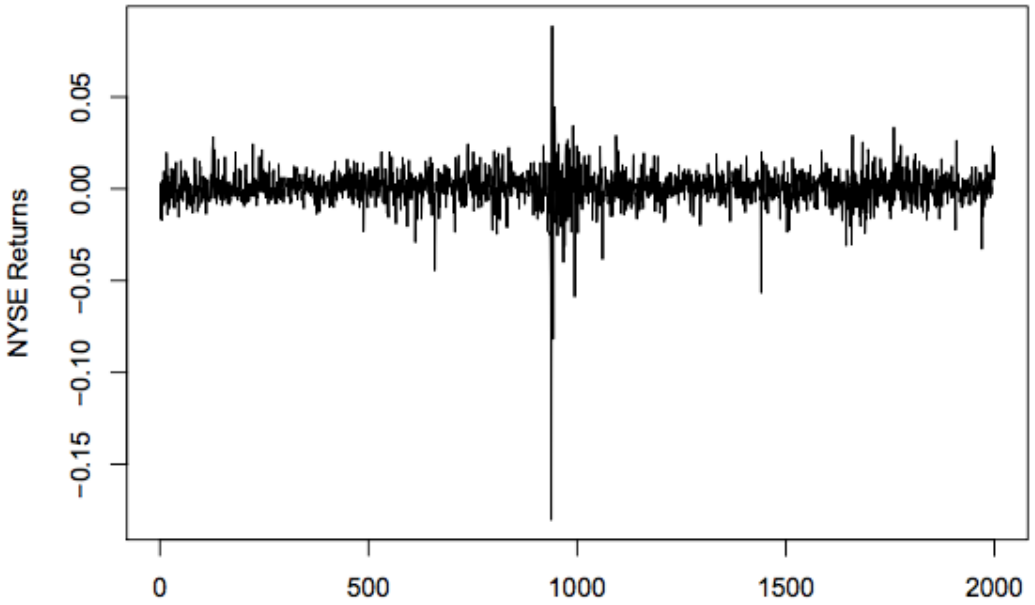
Australian Red Wine Sales



Source: Brockwell and Davis, Introduction to Time Series and Forecasting

Components of TSA (cont'd)

- Random Variations
  - Erratic movements that are not predictable because they don't follow a pattern
  - Example: strike, fire, war, flood, earthquake, etc..



Source: Brockwell and Davis, Introduction to Time Series and Forecasting

TSA Terms

- Stationary Data - a time series variable exhibiting no significant upward or downward trend over time
- Nonstationary Data - a time series variable exhibiting a significant upward or downward trend over time
- Seasonal Data - a time series variable exhibiting a repeating patterns at regular intervals over time

TSA Techniques Overview

- There are many, many different time series techniques
- It is usually impossible to know which technique will be best for a particular data set
- It is customary to try out several different techniques and select the one that seems to work best
- To be an effective time series modeler, you need to keep several time series techniques in your "tool box"

Intro to R and RStudio

Please go over the software installation guide on the website

Always remember…

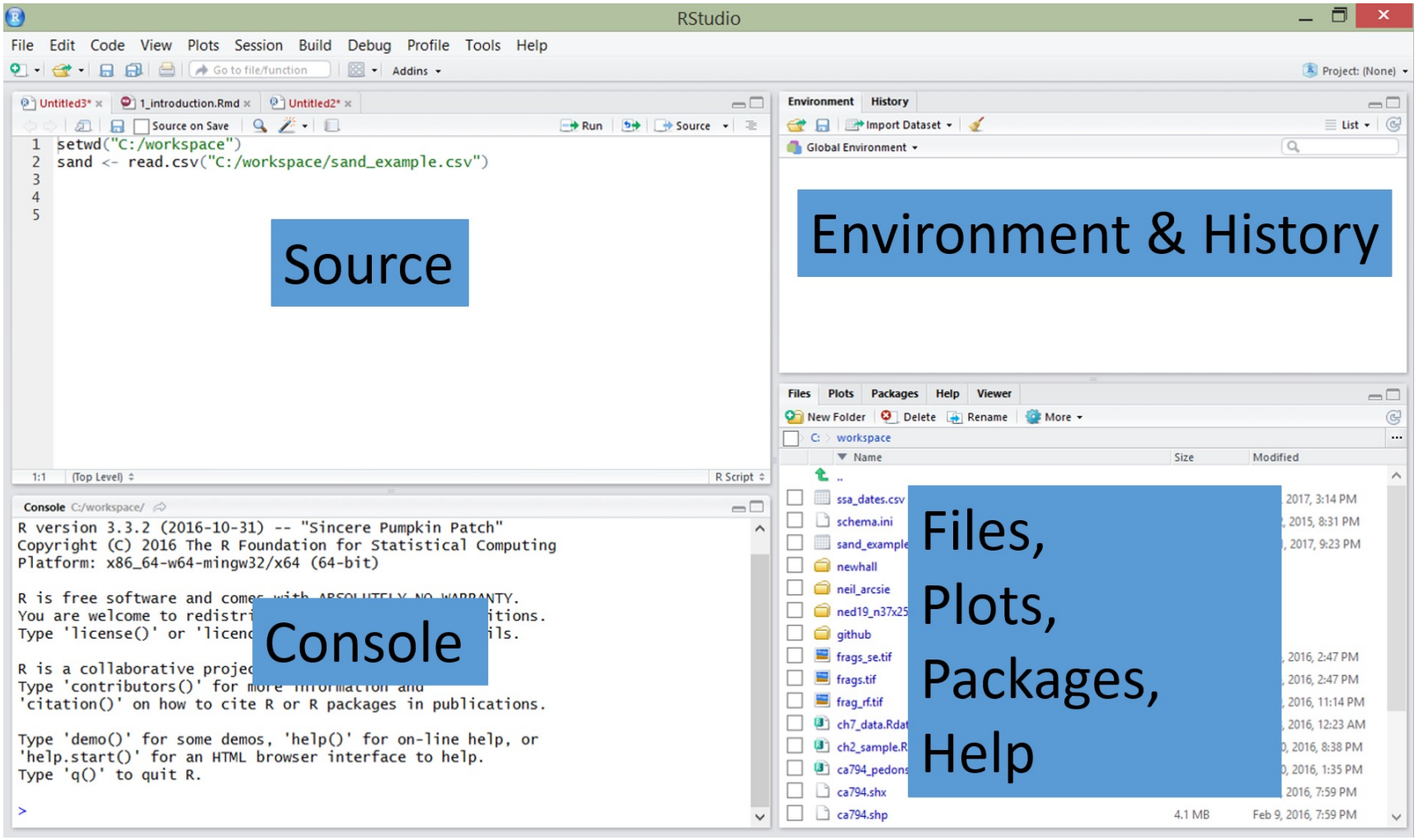
- R is the name of the programming language

- RStudio is a convenient interface

RStudio Windows

RStudio Windows / Tabs	Location	Description
Console Window	lower-left	location were commands are entered and the output is printed
Source Tabs	upper-left	built-in text editor
Environment Tab	upper-right	interactive list of loaded R objects
History Tab	upper-right	list of key strokes entered into the Console
Files Tab	lower-right	file explorer to navigate C drive folders
Plots Tab	lower-right	output location for plots
Packages Tab	lower-right	list of installed packages
Help Tab	lower-right	output location for help commands and help search window
Viewer Tab	lower-right	advanced tab for local web content

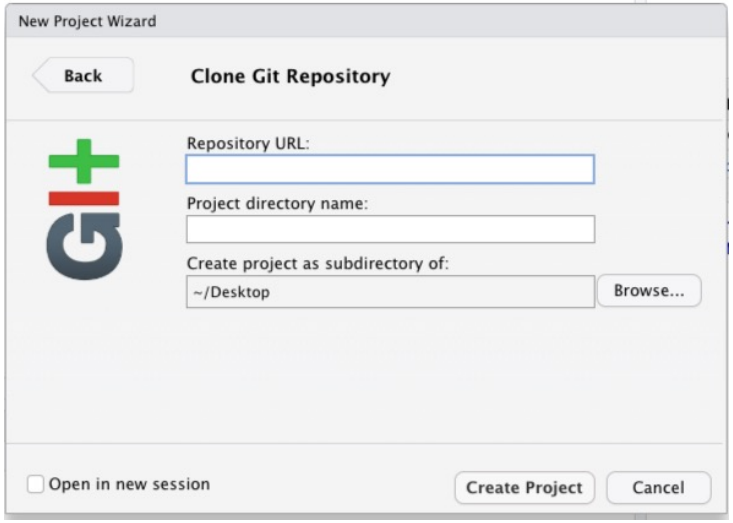
RStudio Default Layout



*Source: Chapter 1 Introduction to R and Rstudio by Katey Yoast, Skye Wills, Stephen Roecker and Tom D'Avello (2017-05-05)

Before you start…

- Create a new RStudio Project
  - New Project > Version Control > Git >



- Refer to your forked repository
- Choose were you want to store on your machine

Install Packages

- A package is a bundle of commands that can be loaded into R, to provide extra functionality.
- To install a package, type on the console window

```
> install.packages("package-name")
```

- There is also user interface to install packages
- Note: You only need to install packages once, afterwards you just need to load them using

```
> library(package-name)
```

Install Packages (cont'd)

- I will point you out to the packages you will need throughout the course
- But I encourage you to simply google what you need and you will be able to find a package for that in CRAN or GitHub. Example:

  type: correlation in R

- It is very easy to figure out how to do things in R because tons of people are using it
- Here is a useful link with a list of common packages in R

https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages

Install Packages (cont'd)

- To see what packages you have already installed, type on the console:

  > *installed.packages()*

These are the default packages that are installed with R

Package	Version	Priority
"base"	"3.4.3"	"base"
"boot"	"1.3-20"	"recommended"
"class"	"7.3-14"	"recommended"
"cluster"	"2.0.6"	"recommended"
"codetools"	"0.2-15"	"recommended"
"compiler"	"3.4.3"	"base"
"datasets"	"3.4.3"	"base"
"foreign"	"0.8-69"	"recommended"
"graphics"	"3.4.3"	"base"
"grDevices"	"3.4.3"	"base"
"grid"	"3.4.3"	"base"
"KernSmooth"	"2.23-15"	"recommended"
"lattice"	"0.20-35"	"recommended"
"MASS"	"7.3-47"	"recommended"
"Matrix"	"1.2-12"	"recommended"
"methods"	"3.4.3"	"base"
"mgcv"	"1.8-22"	"recommended"
"nlme"	"3.1-131"	"recommended"
"nnet"	"7.3-12"	"recommended"
"parallel"	"3.4.3"	"base"
"rpart"	"4.1-11"	"recommended"
"spatial"	"7.3-11"	"recommended"
"splines"	"3.4.3"	"base"
"stats"	"3.4.3"	"base"
"stats4"	"3.4.3"	"base"
"survival"	"2.41-3"	"recommended"
"tcltk"	"3.4.3"	"base"
"tools"	"3.4.3"	"base"
"utils"	"3.4.3"	"base"

Writing Scripts

- To create a new script click on *File/New File/R script or Rmd script*
- When writing a script, add comments to describe your analysis by inserting a # in front of a line of text.
- Begin your script by specifying who is writing the script, the date, and the main goal. For example:

  #Time Series Analysis – ENV790.30

  #Introduction to R and RStudio

  #Written by Luana Lima
- The next few lines of code usually load the packages you will use
- Finally, you may start writing your code…

Importing Data

- Common function to read data and most relevant default arguments

  > library("utils")

  > read.table(file="C:/workspace/mydata.csv", header = FALSE, sep=" ", dec=".",…)

- This function supports common data files as .txt, .xlsx and .csv
- For further information on this function write

  > help("read.table")

- You can also use its variations for csv files

  > read.csv()

- Note: with this function data is saved on a data frame format (more on that later)

Importing data (cont'd)

- For excel .xlsx file we also have

  > library(xlsx)

  > read.xlsx()

- With this function you have an argument where you can specify your worksheet

  > read.xlsx(file="C:/workspace/mydata.csv", sheetName="myworksheet")

- Note: Sometimes you don't need to specify the argument name. R will still understand it. But I encourage you to do so. You will thank me later!

Import data (cont'd)

- You can import files from Stata, SAS, Minitab and others
- Look up for specific packages for your data file
- After you import data, make sure it loaded as expected
  - You may have missing rows or columns
  - Columns may have different formats (int, real, char, …)
- Functions that might help you look up
  - head(), str() (package "utils")
  - dim(), names(), summary() (package "base")

Data Types

Vectors

- Sequence of data elements of the same mode (num, char)

Matrices

- All columns must have same mode and same length

Arrays

- Similar to matrices but can have mora than one dimension

Data Frame

- More general than a matrix, list of vector of equal length but can have different modes

Lists

- Ordered collection of objects

A **data frame** is used for storing data tables.

Time Series Objects

- Once you have read the data into R, store the data in a time series object
- Then, you will be able to use R's functions for analyzing time series data
- The function ts() from package "stats" is used to create time series objects

  > ts(data = NA, start = 1, end = numeric(), frequency = 1,...)

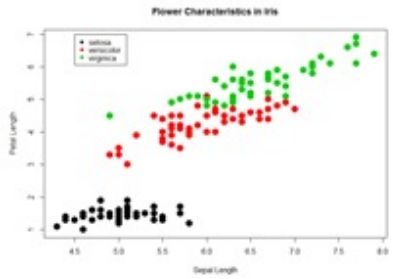
- Function as.ts(x) converts an object to a time series
- Function is.ts(x) tests if an object is a time series

Graphs and Plots

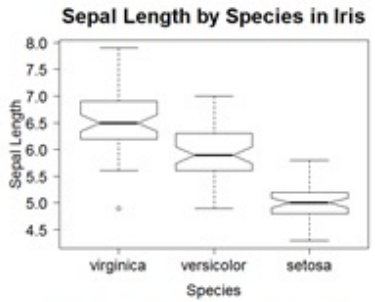
- Create basic graphs like histograms, bar charts, line charts, pie charts, boxplots and scatterplots

1. Basic Histogram

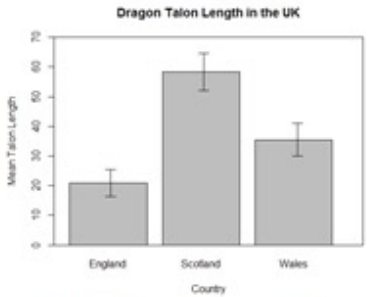
2. Line Graph with Regression



3. Scatterplot with Legend



4. Boxplot with reordered/ formatted axes



5. Boxplot with Error Bars

Graphs and Plots (cont'd)

- Common packages "graphics" and "ggplot"
- Function plot(), so many possible arguments

  > help("plot")
- Trust me, R can do anything you want in a graph! Look up for online tutorials and examples for specific cases

Usage

```
plot(x, y, ...)
```

Arguments

`x`	the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any* `R` *object with a* `plot` *method* can be provided.
`y`	the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type`
what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

All other `type`s give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g. `plot.factor`, do not accept this.

`main`
an overall title for the plot: see `title`.

`sub`
a sub title for the plot: see `title`.

`xlab`
a title for the x axis: see `title`.

`ylab`
a title for the y axis: see `title`.

`asp`
the *y/x* aspect ratio, see `plot.window`.

Graphs and Plots (cont'd)

- The par() function is used to set graphical parameters
  - I use the arguments mfcol or mfrow quite often

    > par(mfrow=c(nr,nc))
  - This command allows you to plot more than one graph in your screen, very useful for comparing time series
- The function par() is often used before plot()
- Other useful functions that can be used after plot()
  - legend() – add legends to plots
  - axis() – add and axis to a plot
  - title() – add labels to a plot (main, xlab, ylab, …)

Graphs and Plots (cont'd)

- Other useful functions
  - lines() – add connected line segments to a plot (useful to plot two time series in the same graph)
  - points() – add points to a plot
  - abline() – add straight lines to a plot (improve graph visualization)

- Use R help to learn more about these functions

Exporting Graphs to a File

- Saving plots as pdf files

  pdf(file="output.pdf", pointsize=16)

  plot()

  dev.off()

  Note: function pdf() uses default package "grDevices"
- You may save several plots in one file by using the pdf() prior to all plot() commands
- Or you may save one plot per file by using pdf() before each plot() command
- Don't forget to close the file after you finish!
- If your code stops due to an error prior to the command dev.off(), you need to close the file manually by typing the command in the console before running the code again

Write Data to a File

- Use the function formatC() to get your data in the format you want

  > x1.fmt=formatC(x1, format="f", digits =3, width=4)

  The width argument is very useful for writing text files with data table because it keeps all columns perfectly aligned

- If you to display a vector in your console, type:

  > cat("Whatever you want to write", var1)

- If you want to write your data in a specific file, call function write() or write.table() or write.csv()

Write Data to a File (cont'd)

write {base} R Documentation

Write Data to a File

Description

The data (usually a matrix) `x` are written to file `file`. If `x` is a two-dimensional matrix you need to transpose it to get the columns in `file` the same as those in the internal representation.

Usage

```
write(x, file = "data",
      ncolumns = if(is.character(x)) 1 else 5,
      append = FALSE, sep = " ")
```

Arguments

`x`	the data to be written out, usually an atomic vector.
`file`	A connection, or a character string naming the file to write to. If `""`, print to the standard output connection. If it is `"|cmd"`, the output is piped to the command given by 'cmd'.
`ncolumns`	the number of columns to write the data in.
`append`	if `TRUE` the data `x` are appended to the connection.
`sep`	a string used to separate columns. Using `sep = "\t"` gives tab delimited output; default is `" "`.

- You may want to declare your file name and location before calling write()

  > outputfile="results.txt"

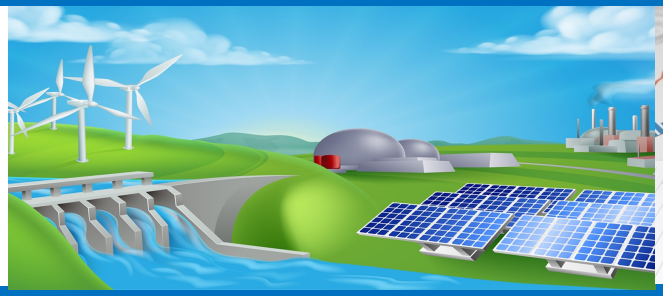
- This will create a file in your working space but you can point to any path in your machine

>outputfile="Users/lmm89/Documents/results.txt"

One condition: Must be an existing folder.

Group discussion

Applications of TSA in the energy/environment field?




THANK YOU !

luana.marangon.lima@duke.edu