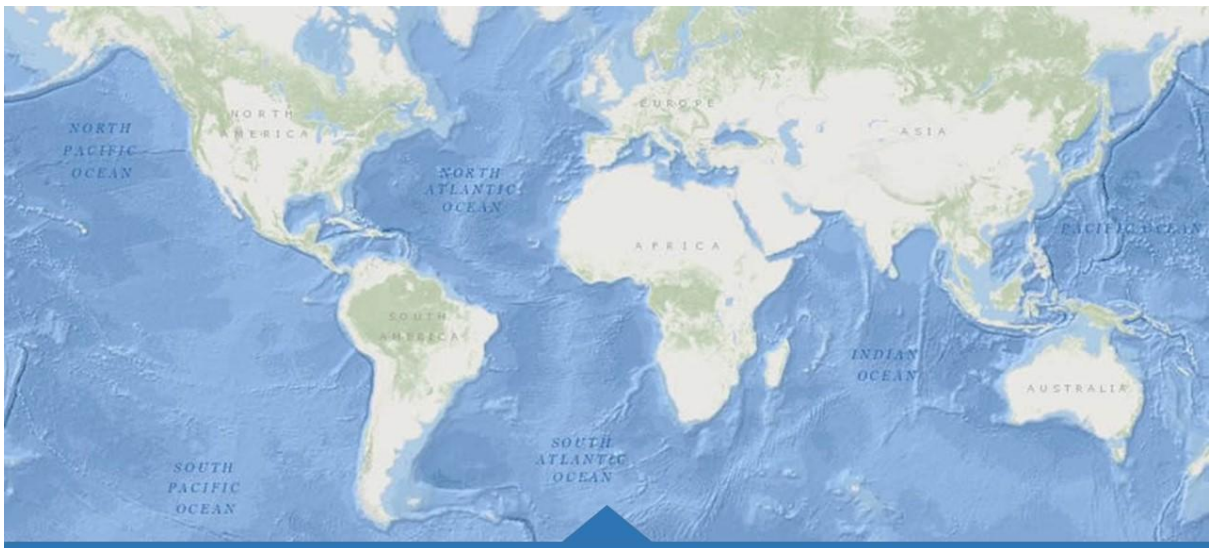


GIS and Data Visualisation Using QGIS and R Studio

Coursebook

Wanda Bodnar

(2017)

Table of Contents

Introduction

Introduction to R Studio

1. Installing R and R Studio

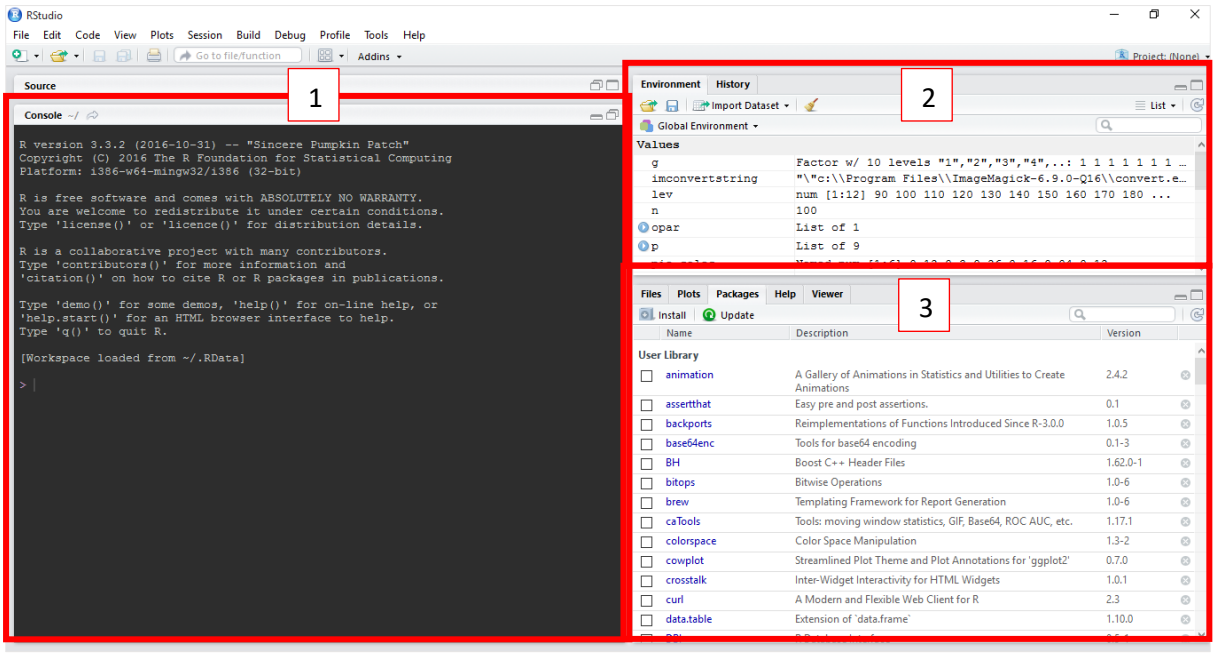
R and R Studio can be downloaded from:

https://cran.r-project.org/ and https://www.rstudio.com/products/rstudio/download/, respectively. (You will need R to work with R Studio.) After you have downloaded both software, open R Studio.

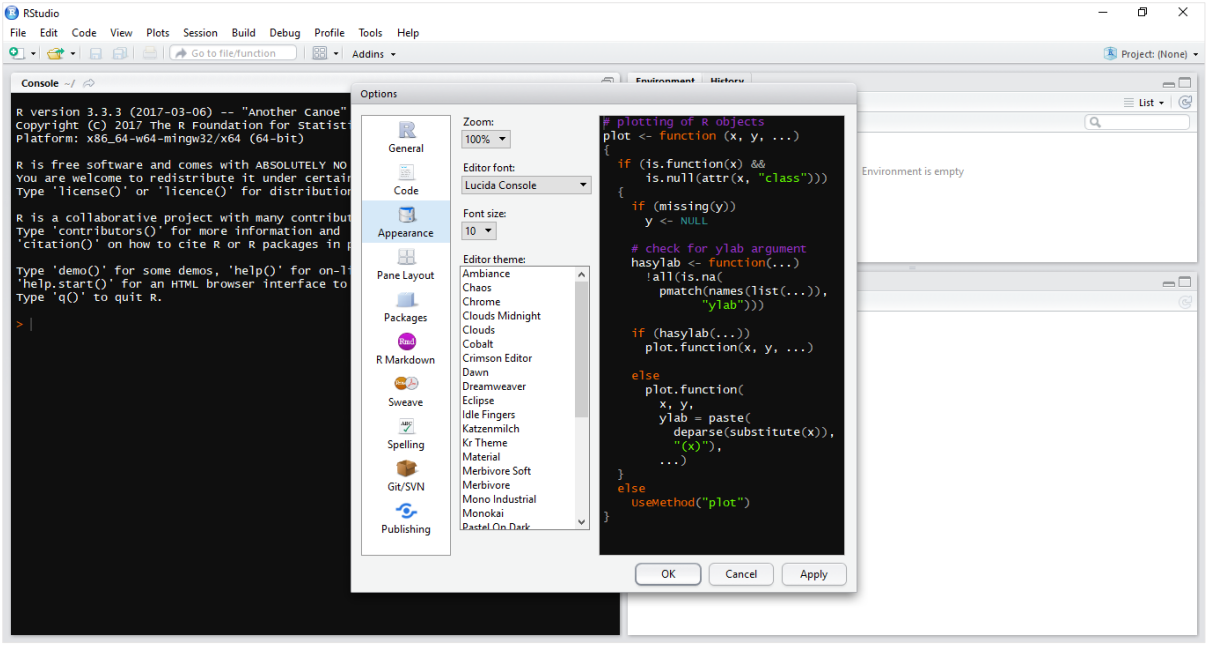
2. Layout

Once you have opened R Studio, you should have the screen open showing three different panes:

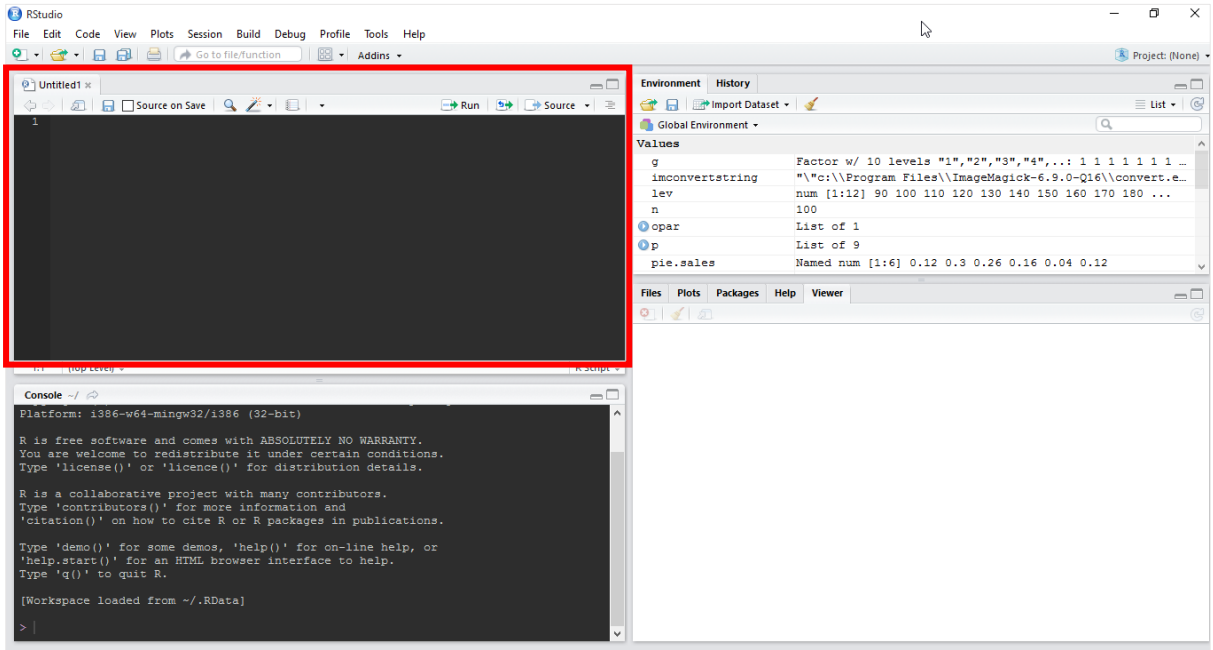
1. Console: for typing commands
2. Top right side with two tabs:
    a. Environment: showing all active objects
    b. History: showing list of commands
3. Bottom right side with five tabs:
    a. Files: your default workspace
    b. Plots: showing graphs that you will generate
    c. Packages: contain collections of R functions, data, and compiled code; packages are used to make graphs, maps and so on
    d. Help: additional info
    e. Viewer: interactive plots and graphs



By clicking on Tools → Options → Global Options you get many options: you can change Appearance, Panel Layout, manage the packages that you use, set Spelling and so on.

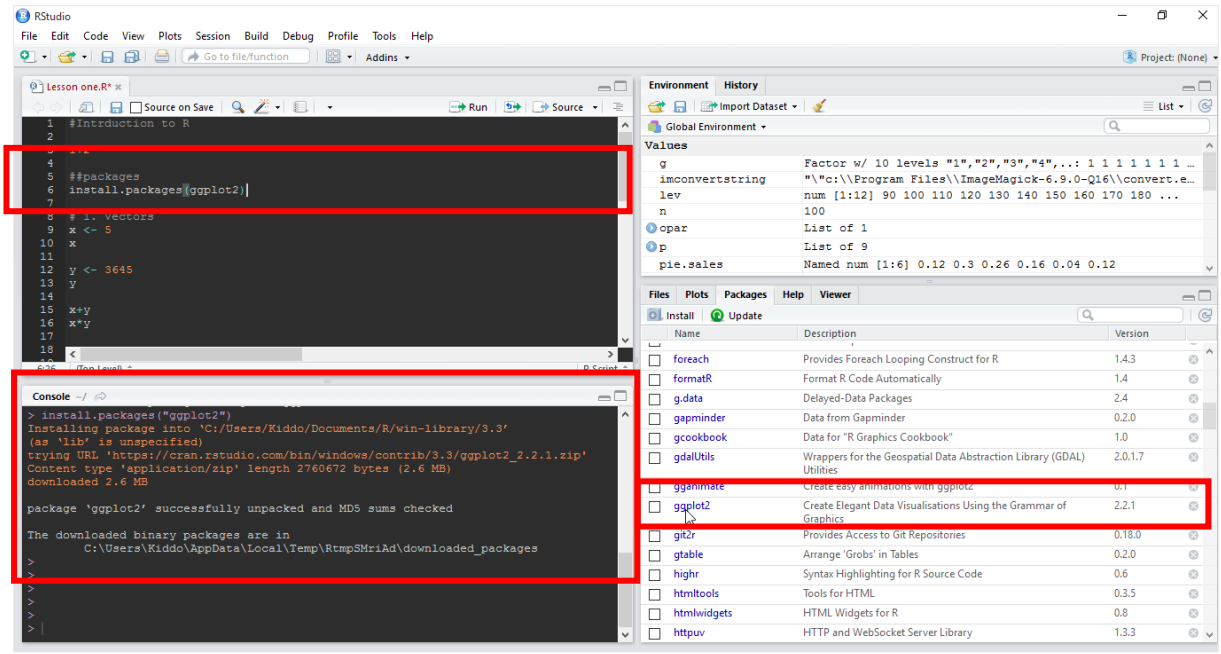


If you click File → New File → Script you will open a new script file that you can write your commands in. This is better to use as you can save your work as you go along.

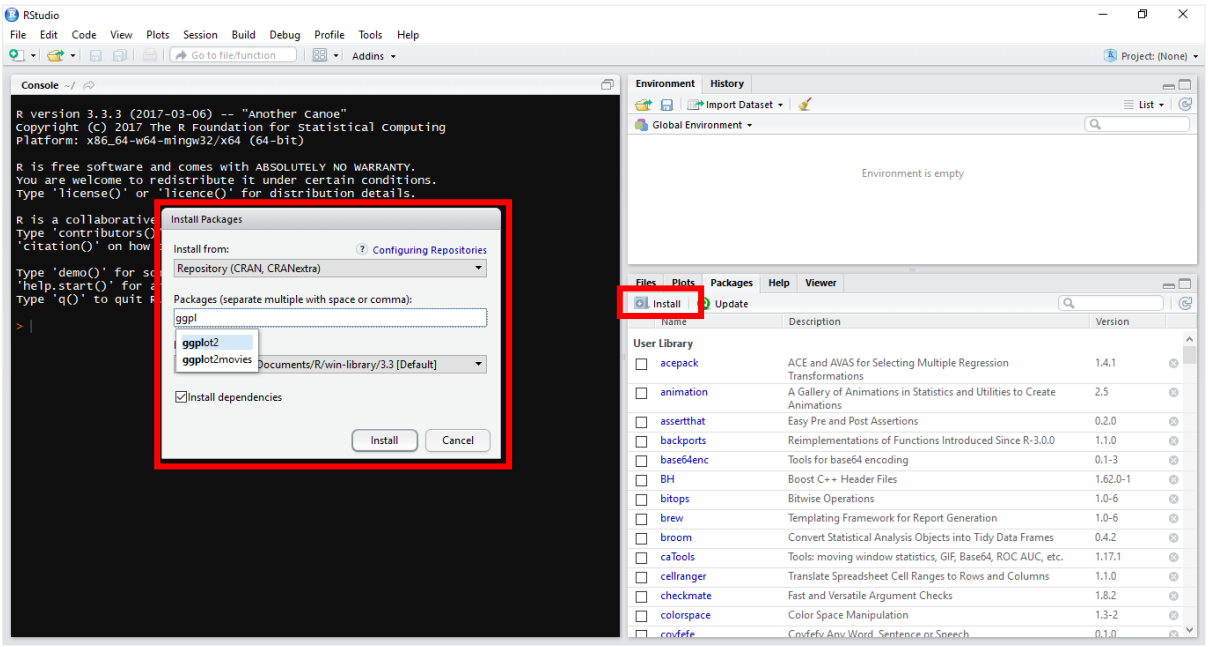


3. Packages

The Packages tab show all the packages that you already have installed. To install more just type: install.packages(name of the package). For example if you want to install the ggplot2 package just type: install.packages(ggplot). After installation it should appear in the Packages tab.



You can also use the Install button and type the name of the package.



If you want to load an already installed package you can either tick the box next to its name or type: library(ggplot2).

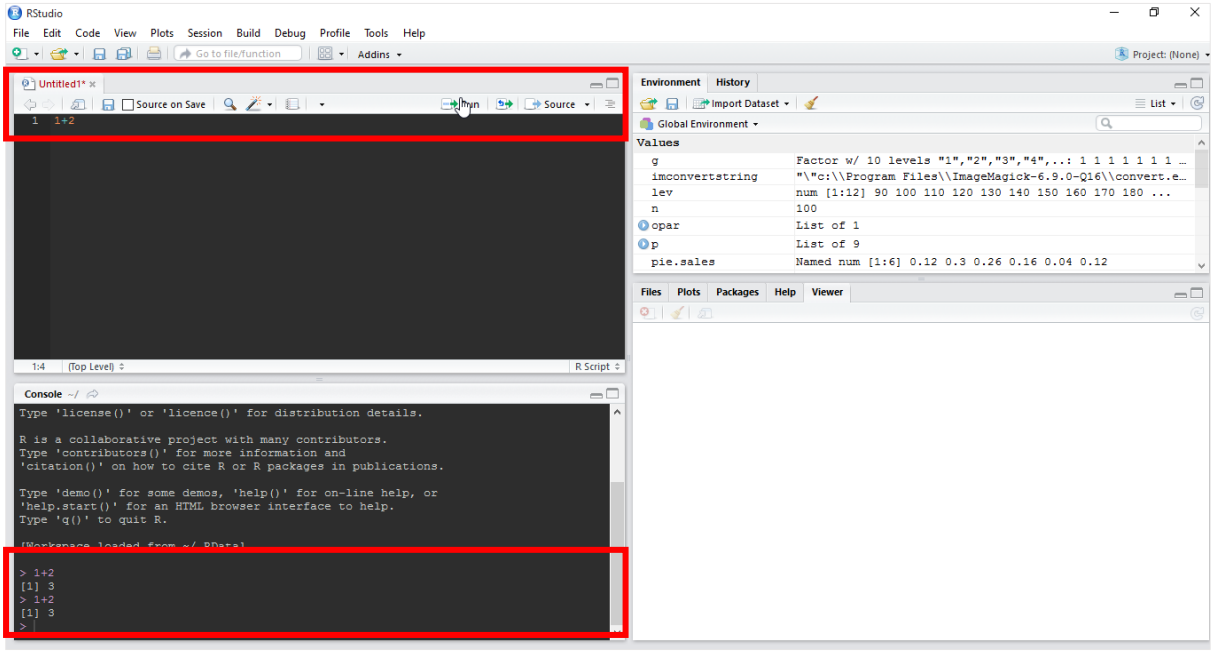
4. Basic functions

Expressions

First use the Console panel and type in 1+2 and click Enter.

```
> 1+2
[1] 3
```

Now use the top Script panel and type in 1+2 then click Run.



You should see the same result appear in the Console window.

Built-in functions

You can find all mathematical functions in R Studio. For example:

```
> log(10)
[1] 2.302585
> cos(5)
[1] 0.2836622
```

Vectors

Variables with one or more values are called vectors. You can create vectors, for example:

```
> x <- 5
> x
[1] 5
> y <- 3645
> y
[1] 3645
> z <- x + y
> z = x + y         #'=' works as well as '<-'
> z
[1] 3650
```

(Note that by using hashtag sign # you can make comment in a line.)

Using the concatenation 'c' function you can assign more than one value to a vector.

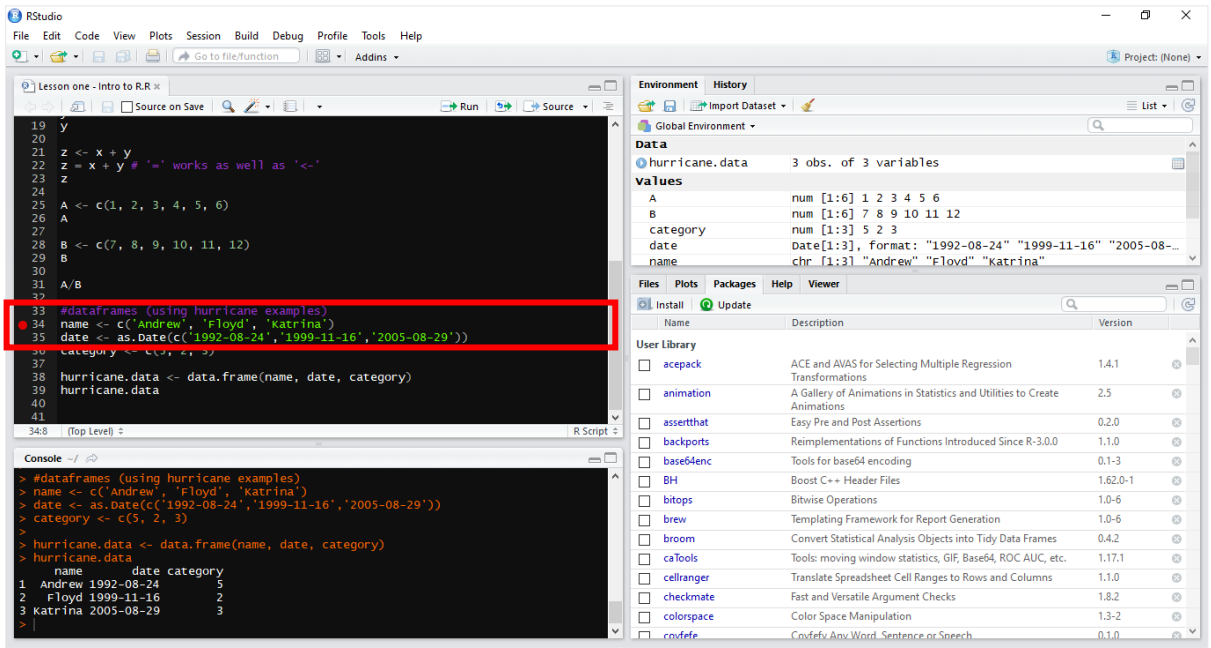
```
> A
[1] 1 2 3 4 5 6
> A <- c(1, 2, 3, 4, 5, 6)
> A
[1] 1 2 3 4 5 6
> B <- c(7, 8, 9, 10, 11, 12)
> B
[1]  7  8  9 10 11 12
> A/B
[1] 0.1428571 0.2500000 0.3333333 0.4000000 0.4545455 0.5000000
```

Dataframes

You can create dataframes from character, numeric and date vectors. These three vectors then can be combined into a dataframe. For example, you can create a dataframe listing three hurricane names, the date and their category:

```
> name <- c('Andrew', 'Floyd', 'Katrina')
> date <- as.Date(c('1992-08-24','1999-11-16','2005-08-29'))
> category <- c(5, 2, 3)
> hurricane.data <- data.frame(name, date, category)
> hurricane.data
     name       date category
1  Andrew 1992-08-24        5
2   Floyd 1999-11-16        2
3 Katrina 2005-08-29        3
```

If you end up having many lines in your R script and you want to find something back, a way to do that if you place a red dot next to the line number.



Making Plots with R Studio

1. Reading your file

For this exercise we will use an openly available global sea surface temperatures data set from the Met Office website. You can access the HadSST.3.1.1.0_annual_globe_ts.txt file here: http://www.metoffice.gov.uk/hadobs/hadsst3/data/download.html.

There are two ways to load the file into R Studio:

a. By accessing from the website directly:

```
data <-
read.table(url("http://www.metoffice.gov.uk/hadobs/hadsst3/data/HadSST.3.1.
1.0/diagnostics/HadSST.3.1.1.0_annual_globe_ts.txt"))
```

b. By downloading the .txt file directly to your computer:

Once you have the file you will have to set the working directory. You can do that by clicking Session → Set Working Directory → Choose Directory. Now you will be able to select the folder where you saved the HadSST.3.1.1.0_annual_globe_ts.txt file.

Once you have your working directory set you can load the file into R Studio:

```
#reading a txt file
temp1 <- read.table(file="HadSST.3.1.1.0_annual_globe_ts.txt")
head(temp1)   #head() will show the first 6 rows
```

If you want to load a .csv you can also do that:

```
#reading a csv file
Temp2 <- read.csv(file="HadSST.3.1.1.0_annual_globe_ts.csv")
Temp2
```

(To load the original txt file into an Excel sheet, open a blank Excel file, then Data → From Text and click on the txt file.)

We will carry on using the txt file (temp1), however, notice that the columns do not have names. We know that the file contains 12 columns, but we are only interested in the first two which are the Year and the Median values. So we will name the first two columns accordingly, and remove the rest:

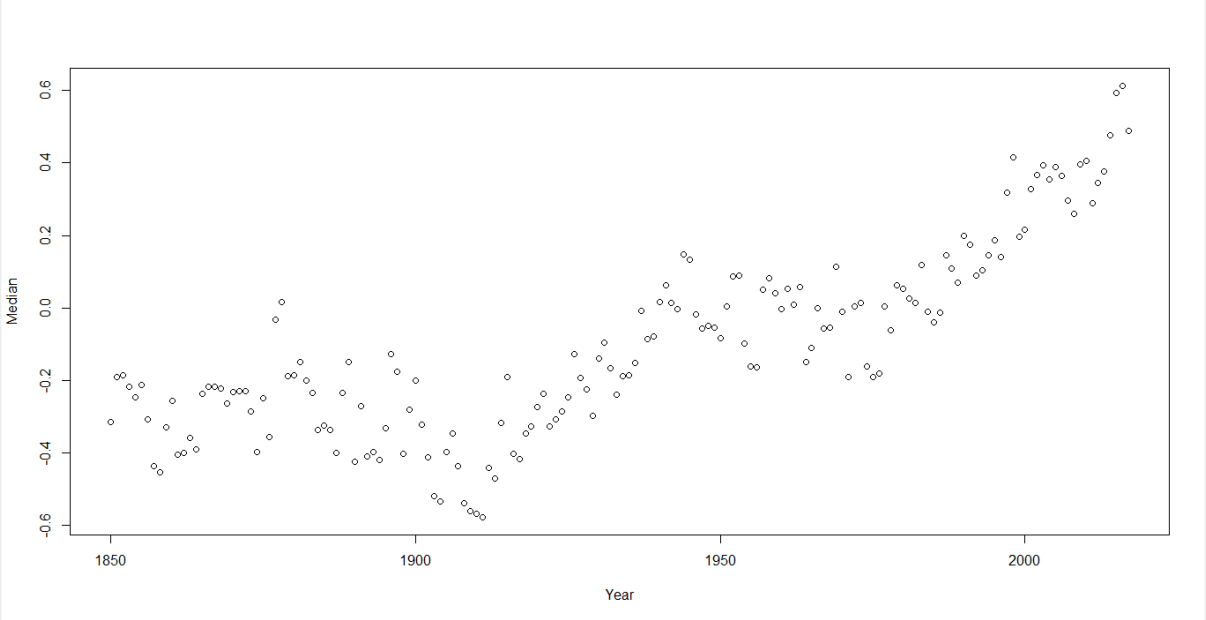
```
#naming the columns
names(temp1) <- c("Year", "Median")
head(temp1)   #names were given to the first two columns

#removing unwanted columns
temp1[3:12] <- NULL
temp1
```

Now you should have your temp1 data ready with the 'Year' and 'Median' columns.

2. Basic plot

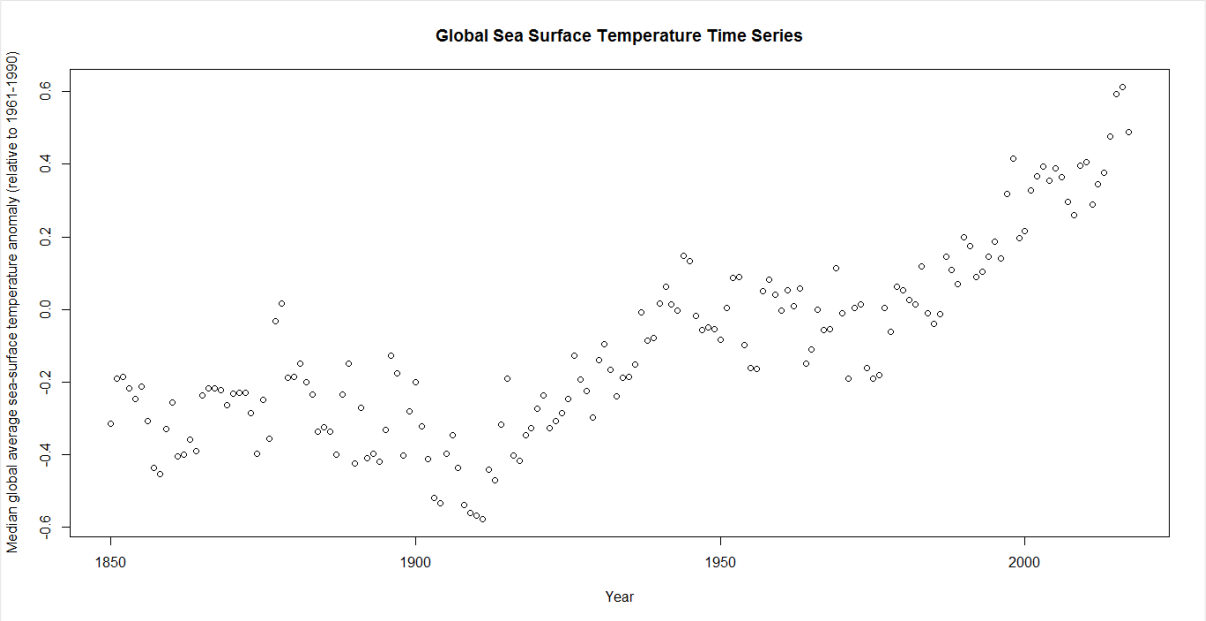
Let's make our first plot.

```
plot(temp1)
```



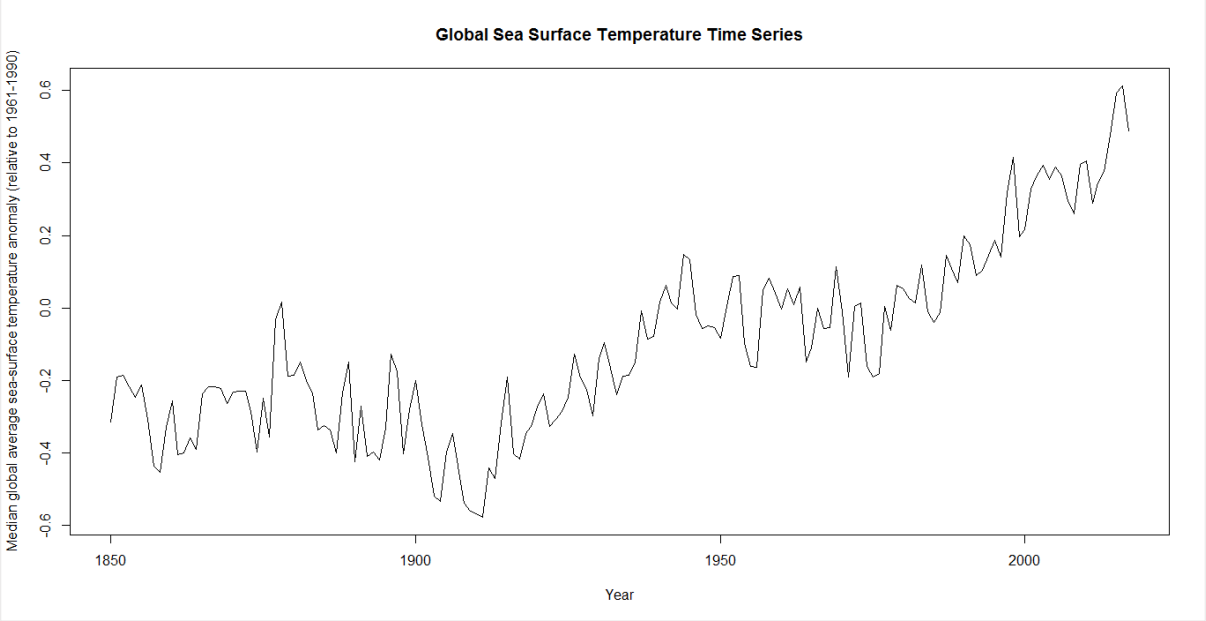
Let's improve on this plot by adding a main title, x-axis and y-axis labels:

```
plot(temp1,
     title("Global Sea Surface Temperature Time Series"),
     ylab = "Median global average sea-surface temperature anomaly (relative
to 1961-1990)",
     xlab = "Year")
```



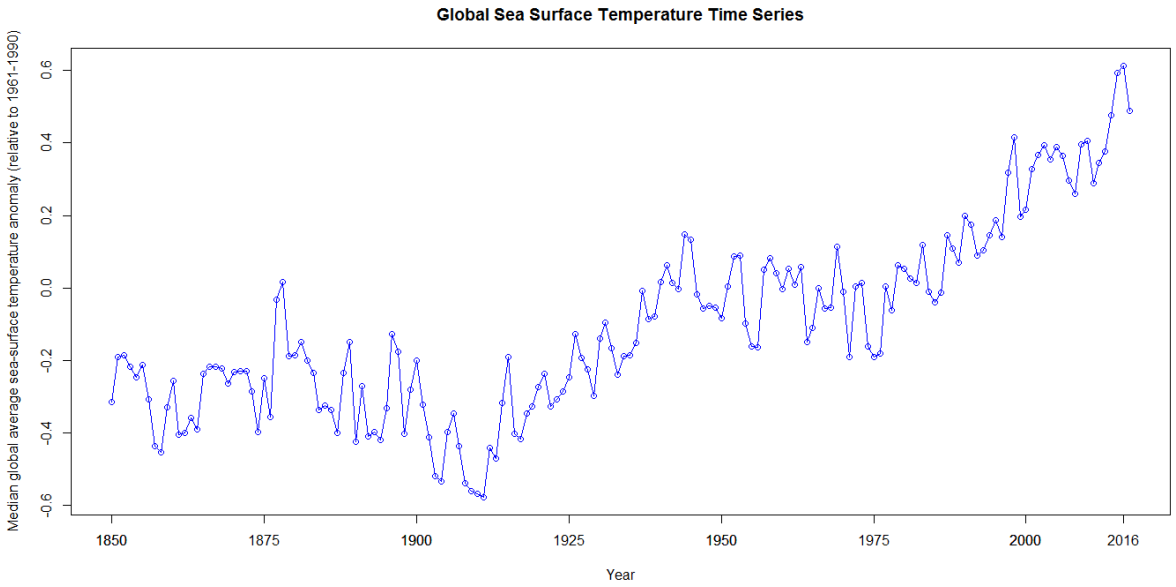
Let's change the points to line:

```
plot(temp1,
     ylab = "Median global average sea-surface temperature anomaly (relative
to 1961-1990)",
     xlab = "Year",
     type = "l")      #type can be l = line, p = point or o = overlaid
title("Global Sea Surface Temperature Time Series")
```



Let's change the colour and also add more ticks with years on the x-axis:

```
plot(temp1,
     ylab = "Median global average sea-surface temperature anomaly (relative
to 1961-1990)",
     xlab = "Year",
     type = "o",       #type can be l = line, p = point or o = overlaid
     col = "blue")
xaxt = "n"
xlim = c(1850, 2016)
ylim = c(-0.6, 0.6)
axis(side = 1, at = c(1850, 1875, 1900, 1925, 1950, 1975, 2000, 2016))
title("Global Sea Surface Temperature Time Series")
```



You can save this graph by going to the Plot tab and clicking Export → Save As Image or Save As PDF.

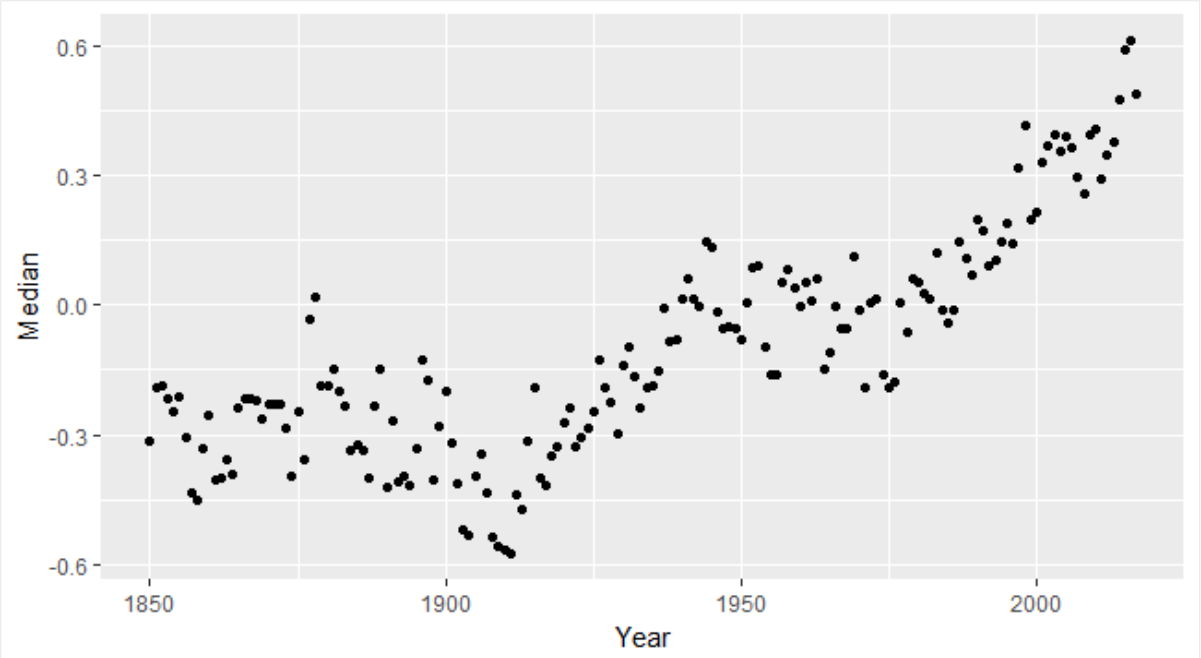
3. Using ggplot

With the package ggplot2 we can create more elegant and complex plots. ggplot2 should already be available in your Package tab. Remember, you can load it by either ticking the box next to it or by typing 'library(ggplot2)'.

ggplot (Grammar of Graphics) uses two principles: graphics, distinct layers of grammatical elements, and meaningful plots through aesthetic mapping. The three essential elements of ggplot are: data, aesthetics (the scales onto which data is being mapped), and geometries (the visual elements used for the data).

So, let's make our first ggplot graph:

```
ggplot(data = temp1, aes(Year, Median)) +
geom_point()
```



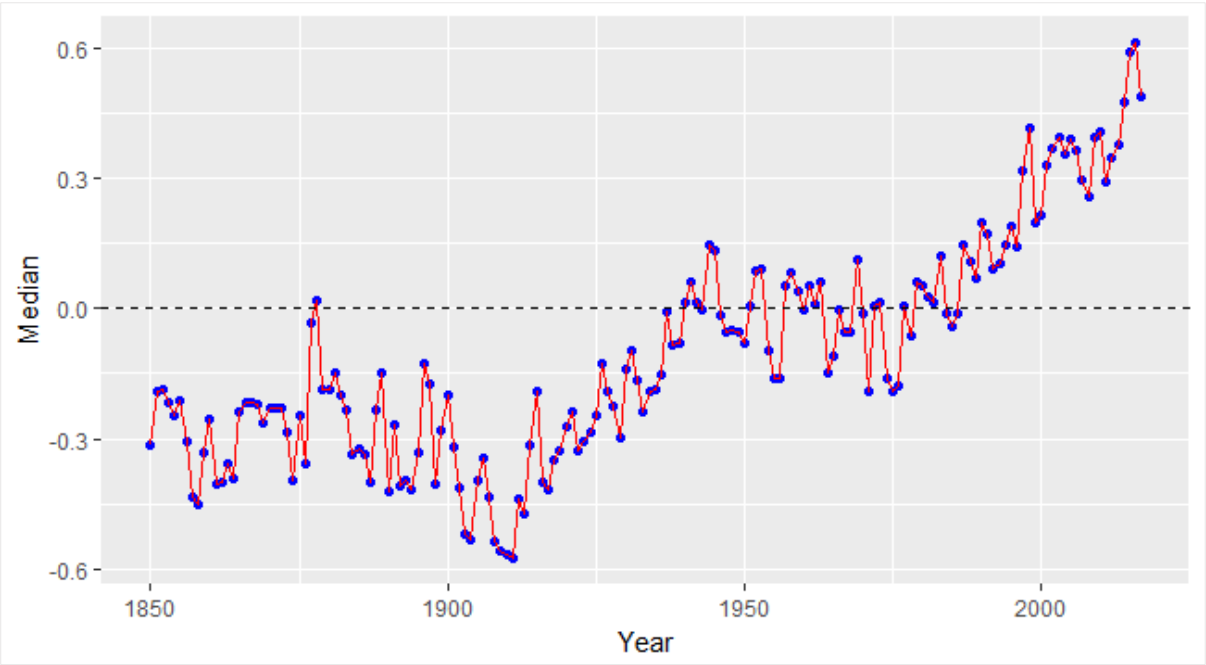
This is the default plot in ggplot2. You have now a number of options to improve your plot:

Let's add lines between the points:

```
ggplot(data = temp1, aes(Year, Median)) +
geom_point() +
geom_line()
```

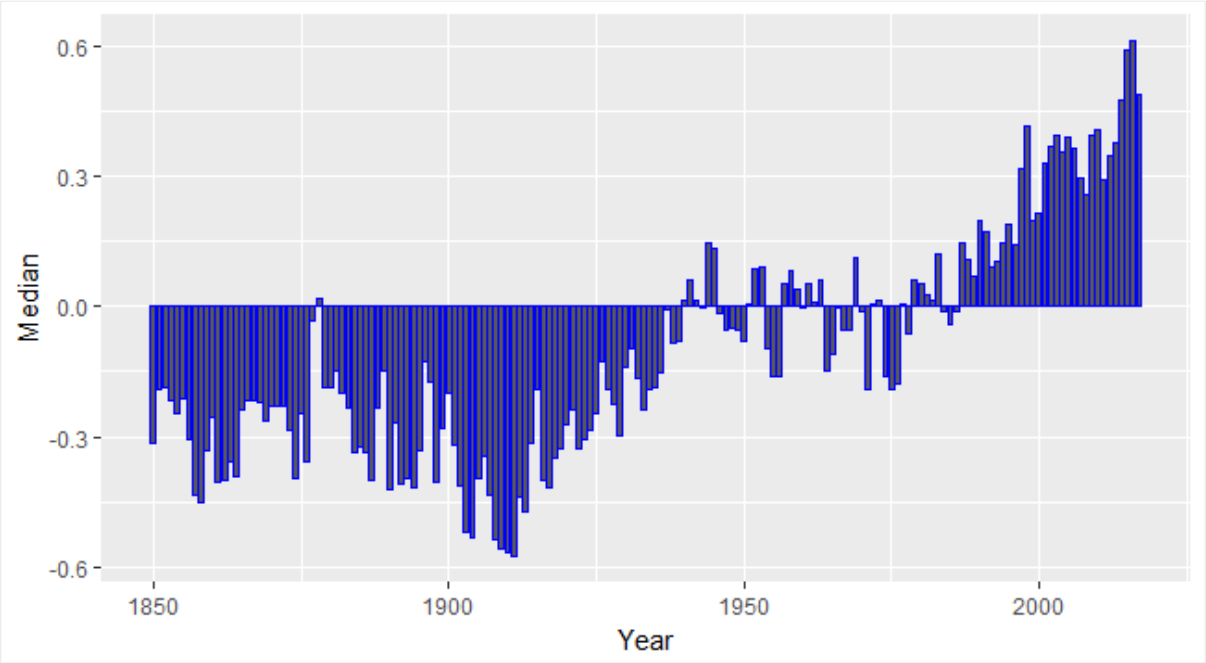
Let's change the colour of the lines and points, and add a horizontal line at 0:

```
ggplot(data = temp1, aes(Year, Median)) +
geom_point(colour="blue") +
geom_line(colour="red") +
geom_hline(yintercept = 0, linetype = "dashed")
```



You can also add bars instead of lines or points:

```
ggplot(data = temp1, aes(Year, Median)) +
geom_bar(stat="identity", colour = "blue")
```



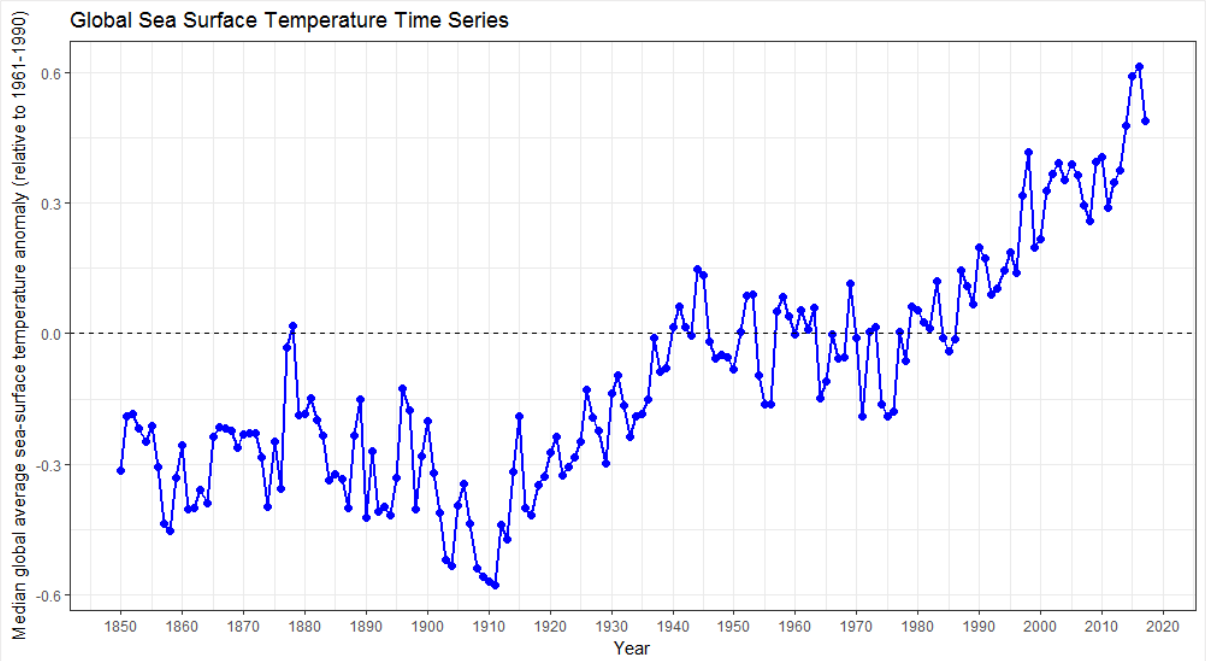
If you want to explore more geom options, using this code will give you all the choices:

```
grep("^geom", objects("package:ggplot2"), value = TRUE)
```

```
 [1] "geom_abline"     "geom_area"       "geom_bar"        "geom_bin2d"
 [5] "geom_blank"      "geom_boxplot"    "geom_col"        "geom_contour"
 [9] "geom_count"      "geom_crossbar"   "geom_curve"      "geom_density"
[13] "geom_density_2d" "geom_density2d"  "geom_dotplot"    "geom_errorbar"
[17] "geom_errorbarh"  "geom_freqpoly"   "geom_hex"        "geom_histogram"
[21] "geom_hline"      "geom_jitter"     "geom_label"      "geom_line"
[25] "geom_linerange"  "geom_map"        "geom_path"       "geom_point"
[29] "geom_pointrange" "geom_polygon"    "geom_qq"         "geom_quantile"
[33] "geom_raster"     "geom_rect"       "geom_ribbon"     "geom_rug"
[37] "geom_segment"    "geom_smooth"     "geom_spoke"      "geom_step"
[41] "geom_text"       "geom_tile"       "geom_violin"     "geom_vline"
```

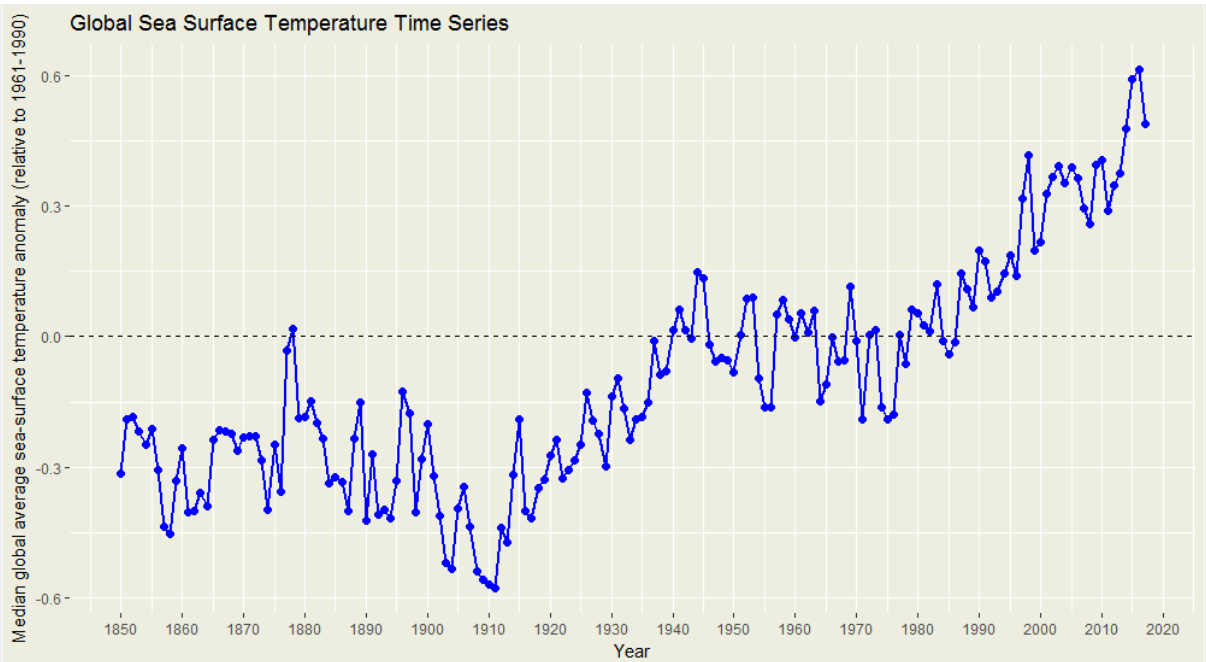
For now, let's carry on working with points and lines and improve our graph by adding a title, x and y labels, gridlines and displaying more years:

```
ggplot(data = temp1, aes(Year, Median)) +
geom_point(colour = "blue", size = 2) +
geom_line(colour = "blue", size = 1) +
geom_hline(yintercept = 0, linetype = "dashed") +
scale_x_continuous(breaks=c(1850, 1860, 1870, 1880, 1890,
                            1900, 1910, 1920, 1930, 1940,
                            1950, 1960, 1970, 1980, 1990,
                            2000, 2010, 2020)) +
scale_y_continuous(breaks=c(-0.6, -0.3, 0, 0.3, 0.6)) +
labs(title = "Global Sea Surface Temperature Time Series") +
labs(y = "Median global average sea-surface temperature anomaly (relative to
1961-1990)") +
theme_bw()      #removes background colour
```



Let's add a lighter colour both to the panel and the plot background:

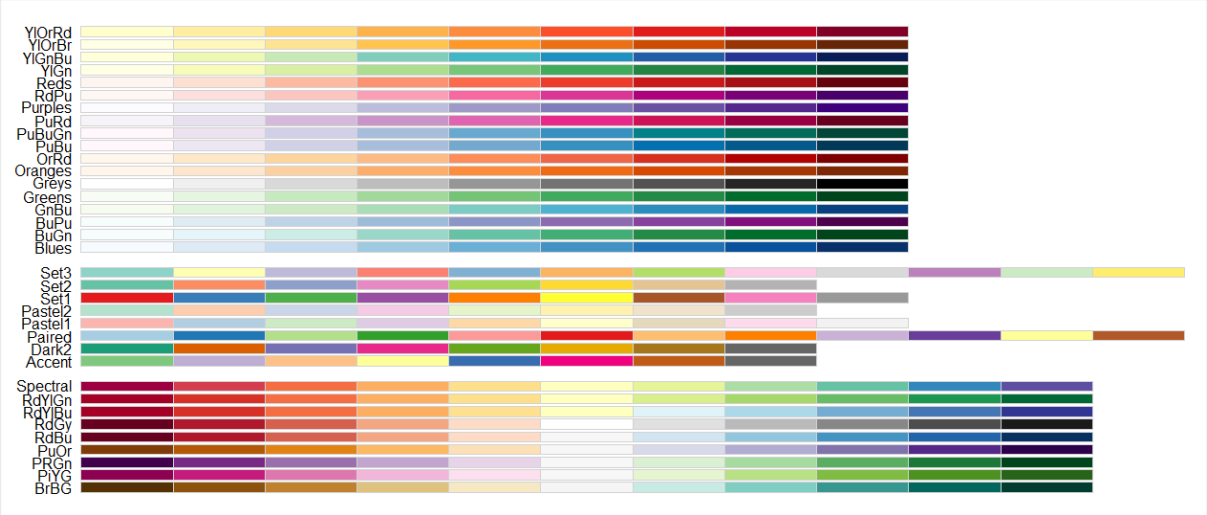
```
ggplot(data = temp1, aes(Year, Median)) +
geom_point(colour = "blue", size = 2) +
geom_line(colour = "blue", size = 1) +
geom_hline(yintercept = 0, linetype = "dashed") +
scale_x_continuous(breaks=c(1850, 1860, 1870, 1880, 1890,
                            1900, 1910, 1920, 1930, 1940,
                            1950, 1960, 1970, 1980, 1990,
                            2000, 2010, 2020)) +
scale_y_continuous(breaks=c(-0.6, -0.3, 0, 0.3, 0.6)) +
labs(title = "Global Sea Surface Temperature Time Series") +
labs(y = "Median global average sea-surface temperature anomaly (relative
to 1961-1990)") +
theme(panel.background = element_rect(fill = "ivory2")) +
theme(plot.background = element_rect(fill = "ivory2"))
```



Your graph is ready!

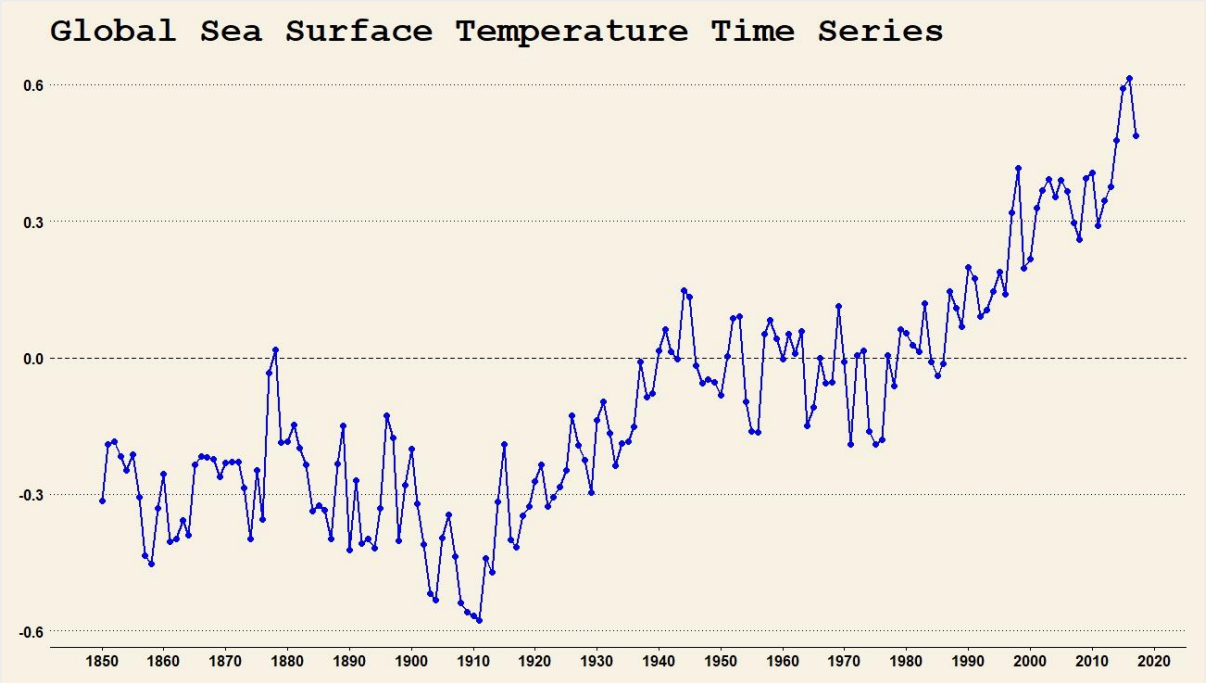
Of course you can make improvements and adjustments. For example further colour options available using:

```
library(RColorBrewer)
display.brewer.all()
```



Or you can choose from different background themes using the ggthemes package:

```
library(ggthemes)
theme_base()
theme_calc()
theme_economist()
theme_excel()
theme_map()
theme_solarized()
theme_stata()
theme_wsj()    #Wall Street Journal
```



4. XKCD styled plot

We will use this package as an example to show how you can change the design and font style of your plot. We will use packages called 'extrafont', 'ggplot2', and you will have to install 'xkcd'. For this download the font type from this website (the file is also available in your resource pack). Once you have the xkcd.ttf file, copy it into your Fonts folder (C:/WINDOWS/Fonts). It should install itself.

Then type font_import() into the Console. This will take some time as it will import all your fonts to the package extrafontdb in your R win-library.

```
font_import()
Importing fonts may take a few minutes, depending on the number of fonts an
d the speed of the system.
Continue? [y/n] y
Scanning ttf files in C:\WINDOWS\Fonts ...
Extracting .afm files from .ttf files...
C:\WINDOWS\Fonts\xkcd.ttf => C:/~/win-library/3.3/extrafontdb/metrics/xkcd
```

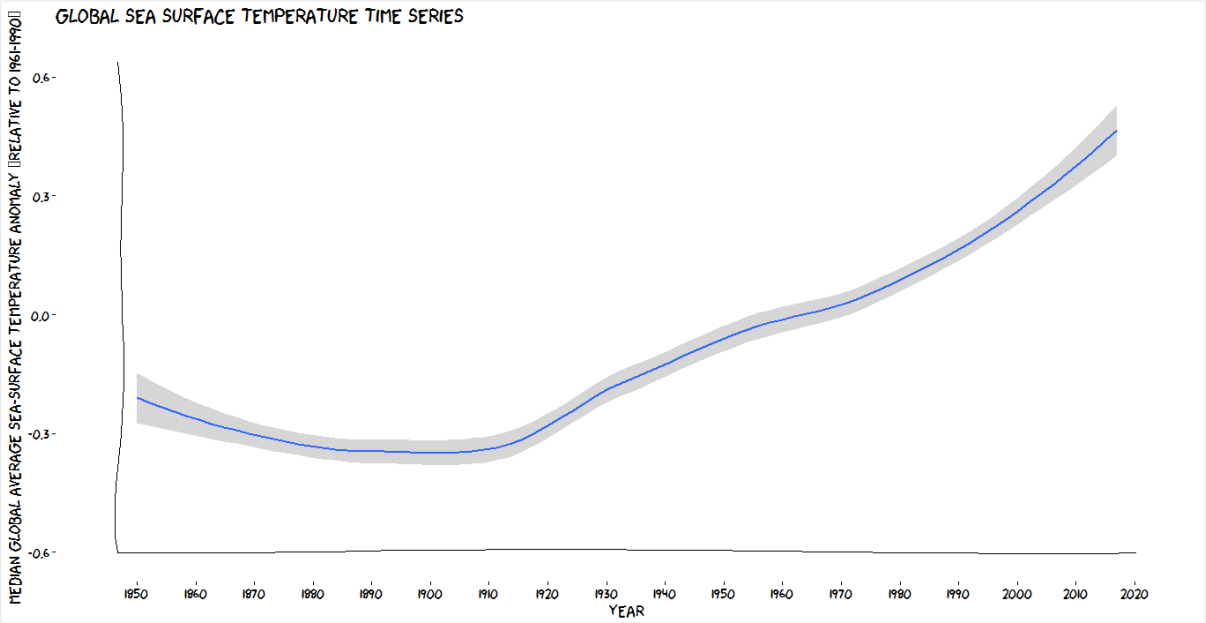
Then type loadfonts() also into the Console. This will register your fonts.

```
loadfonts()
Registering font with R using pdfFonts(): xkcd
```

After you have all this done, close R Studio and reopen it. Don't worry, you will not have to go through these steps anymore.

```
library(extrafont)
library(ggplot2)
library(xkcd)
loadfonts()

#create the XKCD theme
theme_xkcd <- theme(
```

```
  plot.background = element_rect(fill = "white"),
  panel.background = element_rect(fill = "white"),
  panel.grid = element_line(colour = "white"),
  axis.text.y = element_text(colour = "black"),
  axis.text.x = element_text(colour = "black"),
  text = element_text(size = 15, family = "xkcd"))

xrange <- range(temp1$Year)
yrange <- range(temp1$Median)

ggplot(temp1) +
  geom_smooth(mapping = aes(x = Year, y = Median)) +
  scale_x_continuous(breaks=c(1850, 1860, 1870, 1880, 1890,
                              1900, 1910, 1920, 1930, 1940,
                              1950, 1960, 1970, 1980, 1990,
                              2000, 2010, 2020)) +
  scale_y_continuous(breaks=c(-0.6, -0.3, 0, 0.3, 0.6)) +
  labs(title = "Global Sea Surface Temperature Time Series") +
  labs(y = "Median global average sea-surface temperature anomaly (relative
to 1961-1990)") +
  theme_xkcd +
  xkcdaxis(xrange, yrange)
```



5. Interactive plots

After exploring some of the options to make static plots, let's look into making interactive ones.

ggplotly

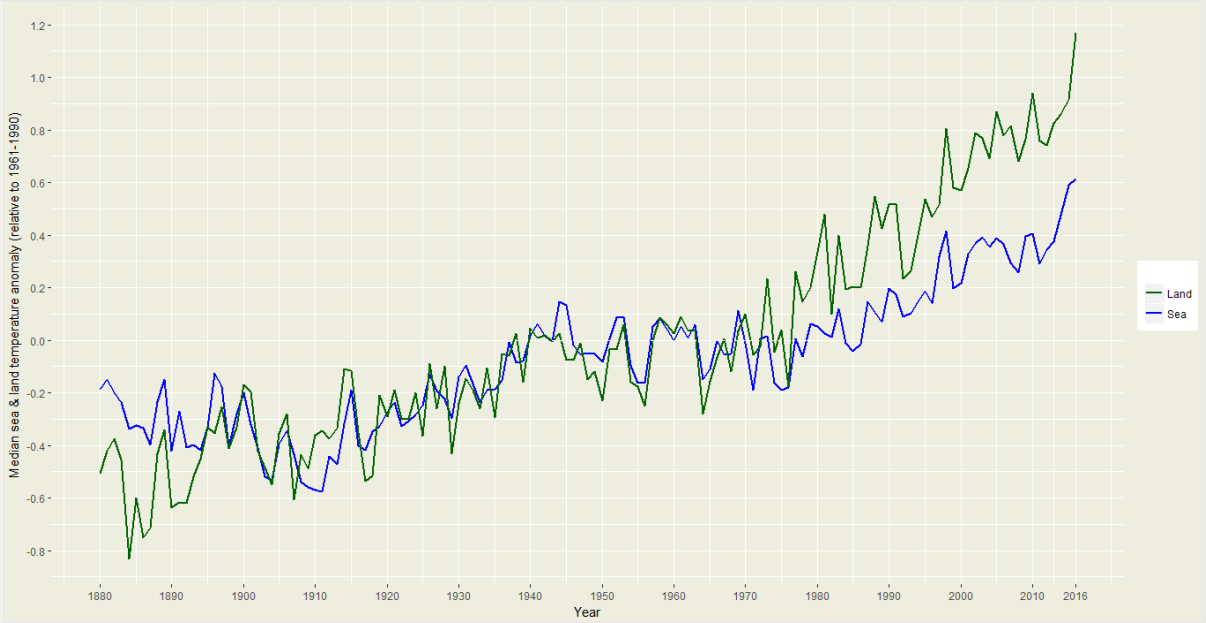
For this interactive line graph you will need the packages called 'tidyverse' and 'plotly', and we will add global land temperature time series to our plot as well. The data for this can be found on this website (and also in your resource pack). (Please note that an extra column was added to the .csv file calculating the median.)

First let's load the land dataset, remove the unwanted columns and merge it with the sea surface temperature dataset:

```
library(tidyverse)
library(plotly)

land <- read.csv(file = "GLB.Ts.csv")
land[2:19] <- NULL
land

t <- merge(temp1, land, by = "Year")
names(t) <- c("Year", "Sea", "Land")
t
```

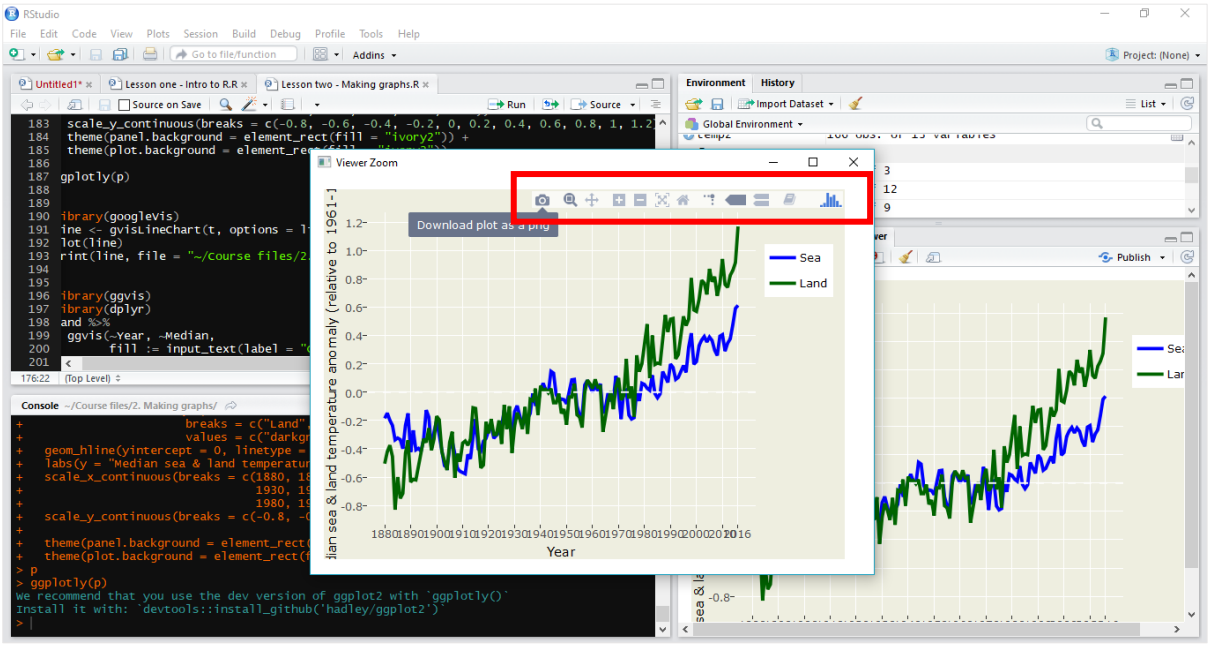
Now we can build the ggplot graph:

```
p <- ggplot(data = t, aes(x = Year)) +
  geom_line(aes(y = Sea, colour = "Sea"), size = 1) +
  geom_line(aes(y = Land, colour = "Land"), size = 1) +
  scale_colour_manual("",
                      breaks = c("Land", "Sea"),
                      values = c("darkgreen", "blue")) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "white") +
  labs(y = "Median sea & land temperature anomaly (relative to 1961-1990)")
+
  scale_x_continuous(breaks = c(1880, 1890, 1900, 1910, 1920,
                                1930, 1940, 1950, 1960, 1970,
                                1980, 1990, 2000, 2010, 2016)) +
  scale_y_continuous(breaks = c(-0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6,
0.8, 1, 1.2)) +
  theme(panel.background = element_rect(fill = "ivory2")) +
  theme(plot.background = element_rect(fill = "ivory2"))
p
```



Then use the function ggplotly to get your interactive plot:

```
ggplotly(p)
```

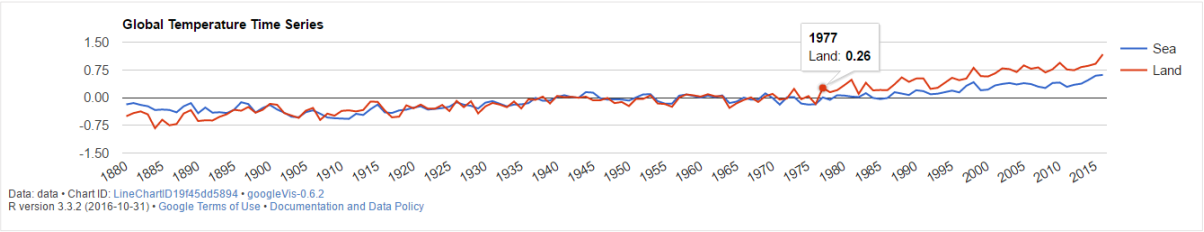


Here you have a number of options to manipulate the graph using the icons on the top right.

googleVis

Let's look at the package googleVis and using the same data set let's create a linechart using googleVis:

```
library(googleVis)
line <- gvisLineChart(t, options = list(title = "Global Temperature Time
Series"))
plot(line)
```



The same graph should have appeared in your default browser. If you run your cursor along the median line, you can see the value appearing in each year.

You can save the plot as .html by typing:

```
print(line, file = "C:/~/plot.htm")
```

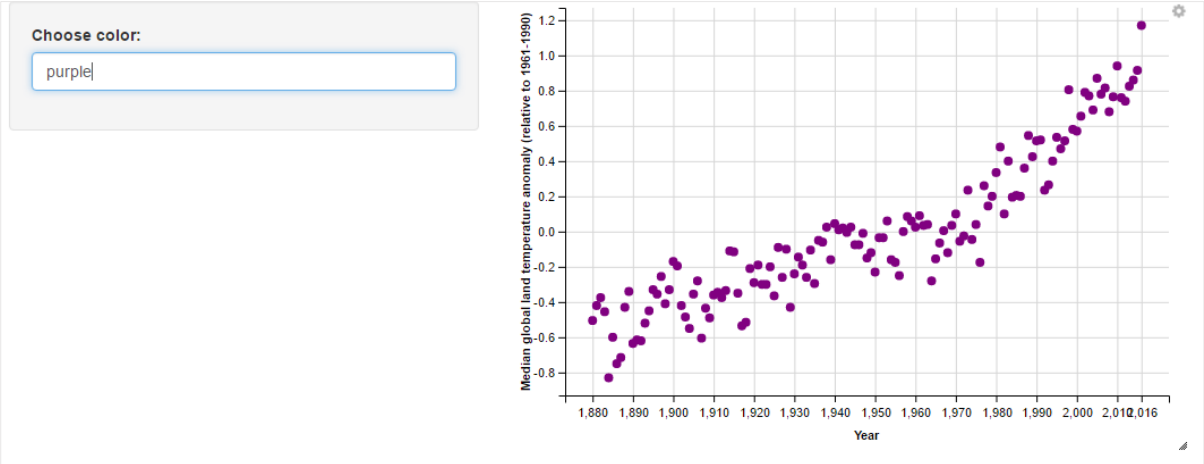
ggvis

In addition to the packages 'plotly' and 'googleVis', you can also use the package called 'ggvis'. For the next exercise we will only use the global land temperature time series to our plot.

```
library(ggvis)
land %>%
  ggvis(~Year, ~Median,
        fill := input_text(label = "Choose color:",
                           value = "black")) %>%
  layer_points() %>%
  add_axis("x",
           title = "Year",
           values = c(1880, 1890, 1900, 1910, 1920,
                      1930, 1940, 1950, 1960, 1970,
                      1980, 1990, 2000, 2010, 2016),
           orient = "bottom") %>%
  add_axis("y",
           title = "Median global land temperature anomaly (relative to
1961-1990)",
           values = c(-0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1,
1.2),
           orient = "left")
```

(ggvis also uses the pipe function, %>% to turn function composition into a series of imperative statements. Note that when you use ggvis, your R session will use dynamic visualisation and it will remain busy until you press escape.)

Using the function above a graph should appear that allows you to choose the colour of the points.

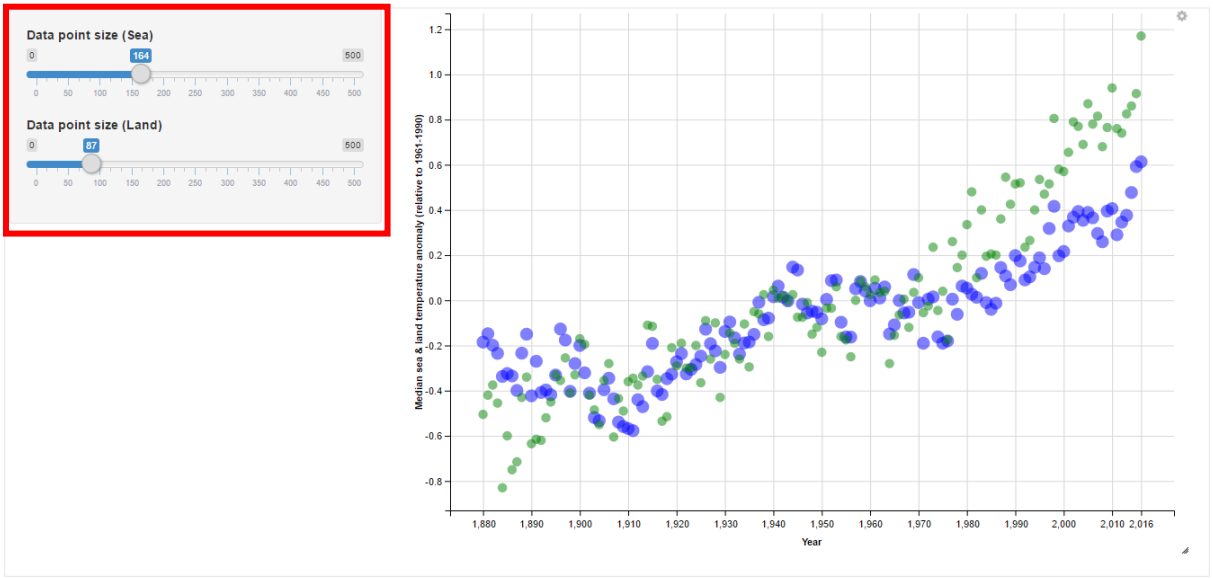


Using the `fill := input_text(label = "Choose color:", value = "black"))` function you can type any colour to display the points.

Now let's use the combined dataset (sea and land):

```
ggvis(t) %>%
  layer_points(x = ~Year, y= ~Sea, fill := "blue",
               size := input_slider(0, 500, label = "Data point size
(Sea)"),
               opacity := 0.5) %>%
  layer_points(x = ~Year, y= ~Land, fill := "green",
```

```
                size := input_slider(0, 500, label = "Data point size
(Land)"),
                opacity := 0.5) %>%
  add_axis("x",
           title = "Year",
           values = c(1880, 1890, 1900, 1910, 1920,
                      1930, 1940, 1950, 1960, 1970,
                      1980, 1990, 2000, 2010, 2016),
           orient = "bottom") %>%
  add_axis("y",
           title = "Median sea & land temperature anomaly (relative to
1961-1990)",
           values = c(-0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1,
1.2),
           orient = "left")
```



Note that for the points you can add the slider option using `size := input_slider(0, 500, label = "Data point size (Sea)")`

II. Discussing Climate Change

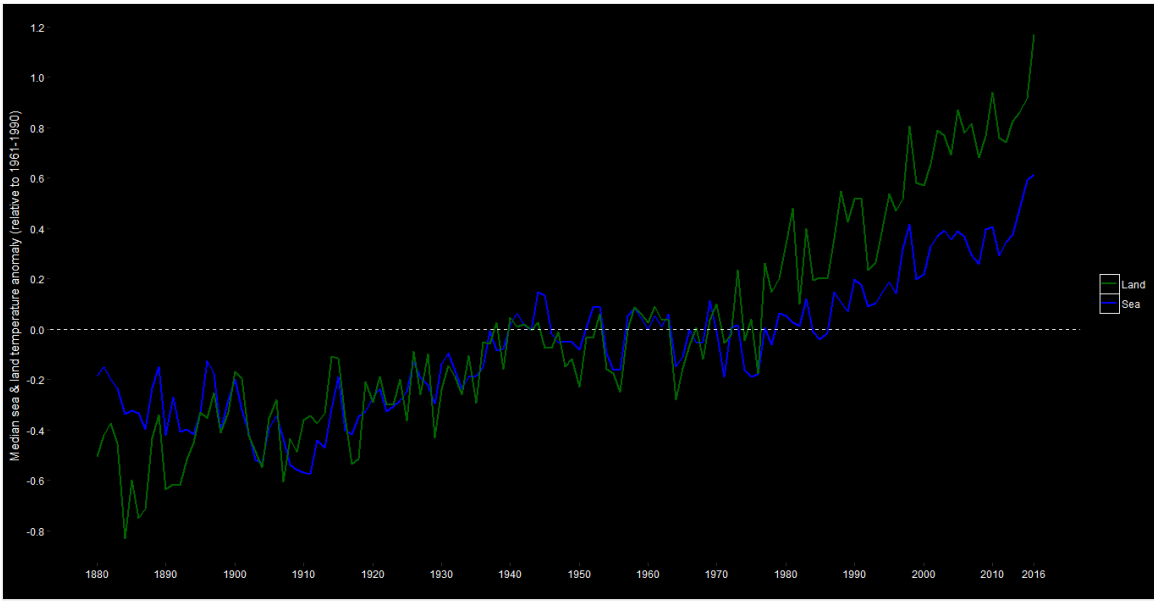
Global temperature rise

The planet's climate system is driven by heat energy arriving from the Sun. 70% of the solar energy arrives at the top of the atmosphere as visible radiation. (30% is reflected back into space by clouds, dust and ice.) About three-quarter of this radiation is absorbed by the surface of the Earth and one-quarter by the atmosphere. Apart from nitrogen and oxygen, the atmosphere contains greenhouse gases (carbon-dioxide, water vapour and methane) that absorb and trap longwave radiation. This is called the greenhouse effect. Because of the shape of the Earth, the incoming solar energy heats the surface unevenly: more heat is absorbed at lower than at higher latitudes.

Albedo or land surface reflectance has a key importance in the greenhouse effect. Dark surfaced areas like forests will absorb more heat, while open agricultural land or ice will reflect more. This doesn't, however, mean that deforestation causes cooling. While vegetation helps facilitate water evaporation into the atmosphere (through transpiration), this mechanism actually requires heat that is taken from the atmosphere.

Seas heat up less rapidly than land masses. This is because the seas diffuse the effects of a temperature change via vertical mixing, currents and convective movements.

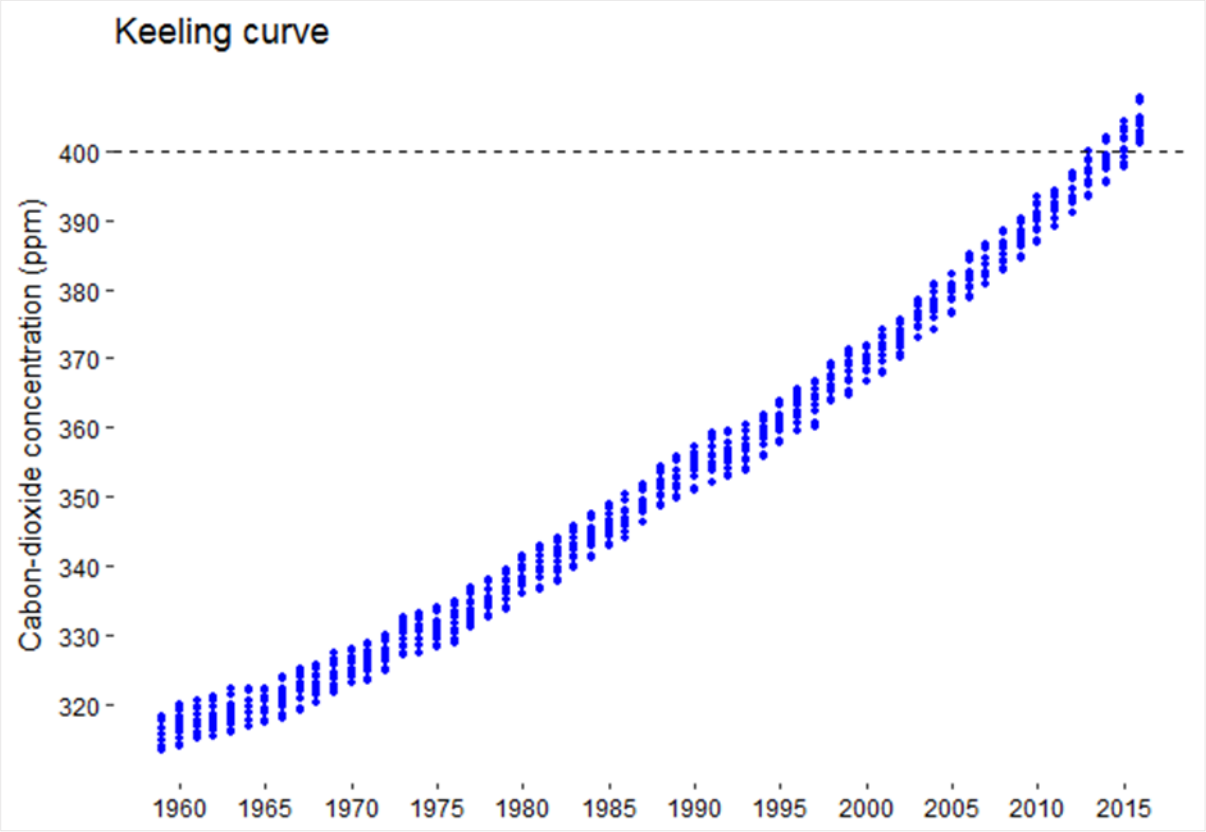
Both land and sea temperature data used in the previous exercise show a continuous increase since 1880, with land surface warming faster than the seas.



6. Animated or motion plot (GIF)

Let's look at another dataset, the well-known Keeling curve, which shows the rise of atmospheric carbon-dioxide concentration. The data can be downloaded from this website.

```
setwd("C:/~")
keeling <- read.csv("keeling.csv")
keeling

ggplot(data = keeling, aes(Year, CO2)) +
  geom_point(colour = "blue", size = 1) +
  labs(title = "Keeling curve") +
  geom_hline(yintercept = 400, linetype = "dashed") +
  labs(y = "Cabon-dioxide concentration (ppm)") +
  scale_x_continuous(breaks=c(1960, 1965, 1970, 1975, 1980,
                              1985, 1990, 1995, 2000, 2005,
                              2010, 2015)) +
  scale_y_continuous(breaks=c(320, 330, 340, 350,
                              360, 370, 380, 390,
                              400)) +
  theme(panel.background = element_rect(fill = "white"),
        plot.background = element_rect(fill = "white"),
        axis.text = element_text(colour = "black"),
        axis.title.x = element_blank())
```



The data points show monthly measurements starting in 1959 until December 2016. Let's take the yearly average and add a new column called Source to help create the animated graph:

```
library(plyr)
keeling2 <- ddply(keeling, .(Yr), summarize, Mn = mean(Mn), CO2 = mean(CO2))
keeling2$Source <- rep("ML", nrow(keeling2))
keeling2
```

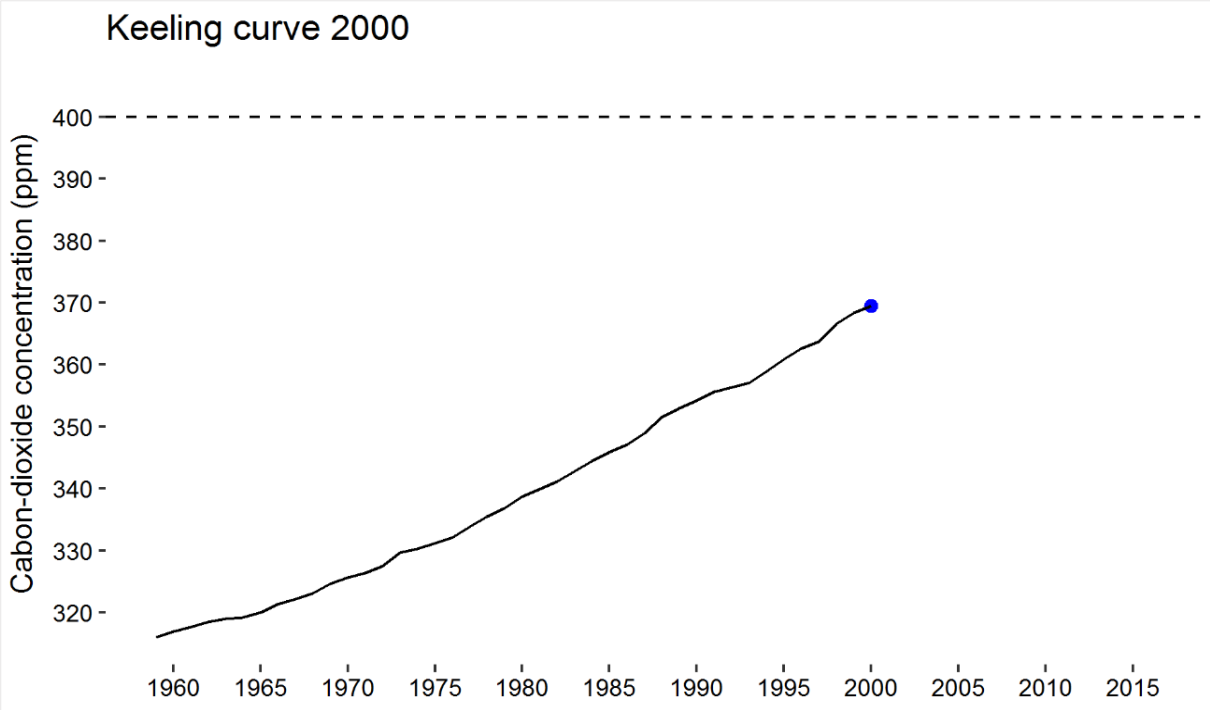
We will use the keeling2 dataset for the next exercise. But first, you have to install a software and two packages. You will need a specific version of the ImageMagick software as the package 'gganimate' only works properly with this.

The ImageMagick software can be downloaded from this website, and you need to look for version ImageMagick-6.9.3-9-Q16-x64-dll.exe. (Also available in your recourse pack.) This version of the ImageMagick software should contain a 'convert' file which R Studio needs to run.

Once you have ImageMagick, install the 'animate' and the 'gganimate' package. The latter can be accessed by typing: devtools::install_github("dgrtwo/gganimate") *(For this you will probably also need the 'devtools' package)*.

```
library(devtools)
library(animation)
devtools::install_github("dgrtwo/gganimate")
library(gganimate)

ggplot(data = keeling2, aes(Year, CO2, frame = Year)) +
  geom_point(colour = "blue", size = 2) +
  geom_line(colour = "blue", size = 1) +
  labs(title = "Keeling curve") +
  geom_hline(yintercept = 400, linetype = "dashed") +
  labs(y = "Cabon-dioxide concentration (ppm)") +
  geom_path(aes(cumulative = TRUE, group = Source)) +
  scale_x_continuous(breaks=c(1960, 1965, 1970, 1975, 1980,
                              1985, 1990, 1995, 2000, 2005,
                              2010, 2015)) +
  scale_y_continuous(breaks=c(320, 330, 340, 350,
                              360, 370, 380, 390,
                              400)) +
  theme(panel.background = element_rect(fill = "white"),
        plot.background = element_rect(fill = "white"),
        axis.text = element_text(colour = "black"),
        axis.title.x = element_blank())

ani.options(interval = .2) #adjusting speed

gganimate(t)
```

Your animated graph should have appeared in the Viewer tab, and it also should have appeared as a picture in your default image viewer.



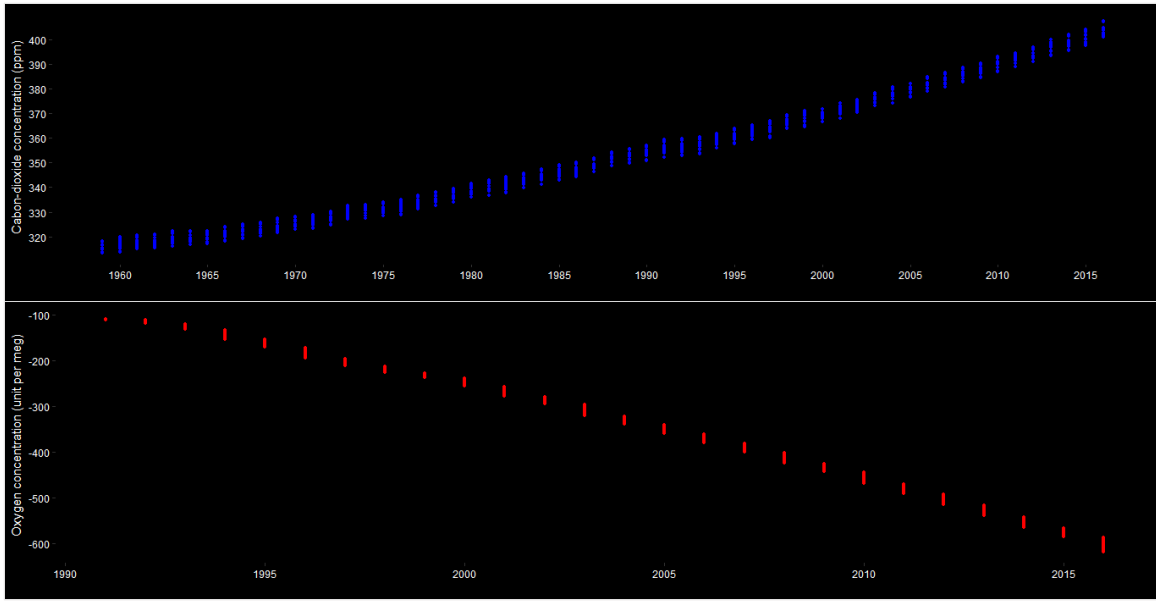
Notice that here the `frame = Year argument`, and an extra line `geom_path(aes(cumulative = TRUE, group = Source))` were added to the code. In essence R Studio creates a picture file (.png) and by adding the frame argument it generates the pictures by year that are stitched together at the end. Also, by determining the `geom_path` to be cumulative, the line between each point will build up.

III. Discussing Climate Change

Keeling curve

The Keeling curve is the global representation of atmospheric carbon-dioxide (CO_2) levels. CO_2 is a greenhouse gas that naturally occurs in the atmosphere. However, with the start of the Industrial Revolution and the burning of fossil fuels, CO_2 levels have constantly been increasing.

In 1958, Charles Keeling started to take daily CO_2 measurements atop Hawaii's Mauna Loa. On the first day his reading was 313 ppm. By 1960 it was established that atmospheric CO_2 concentrations show a seasonal pattern: due to photosynthesis, doing the summer month CO_2 levels drop, whilst during the winter month the levels increase. (This is why the zigzag pattern appears on the graph.) Charles Keeling continued to take daily measurements until his death in 2005. Since then, the measurements have been continued by his son Ralph Keeling, who is also monitoring atmospheric oxygen (O_2) levels. His results show that whilst atmospheric CO_2 have been increasing, O_2 levels have been decreasing.



(Oxygen measurements, reported unit per meg, represents the changes in the O_2/N_2 ratio of air relative to a reference. The observed downward trend of O_2 levels corresponds to losing 19 per meg O_2 molecules out of every 1 million O_2 molecules in the atmosphere each year.)*

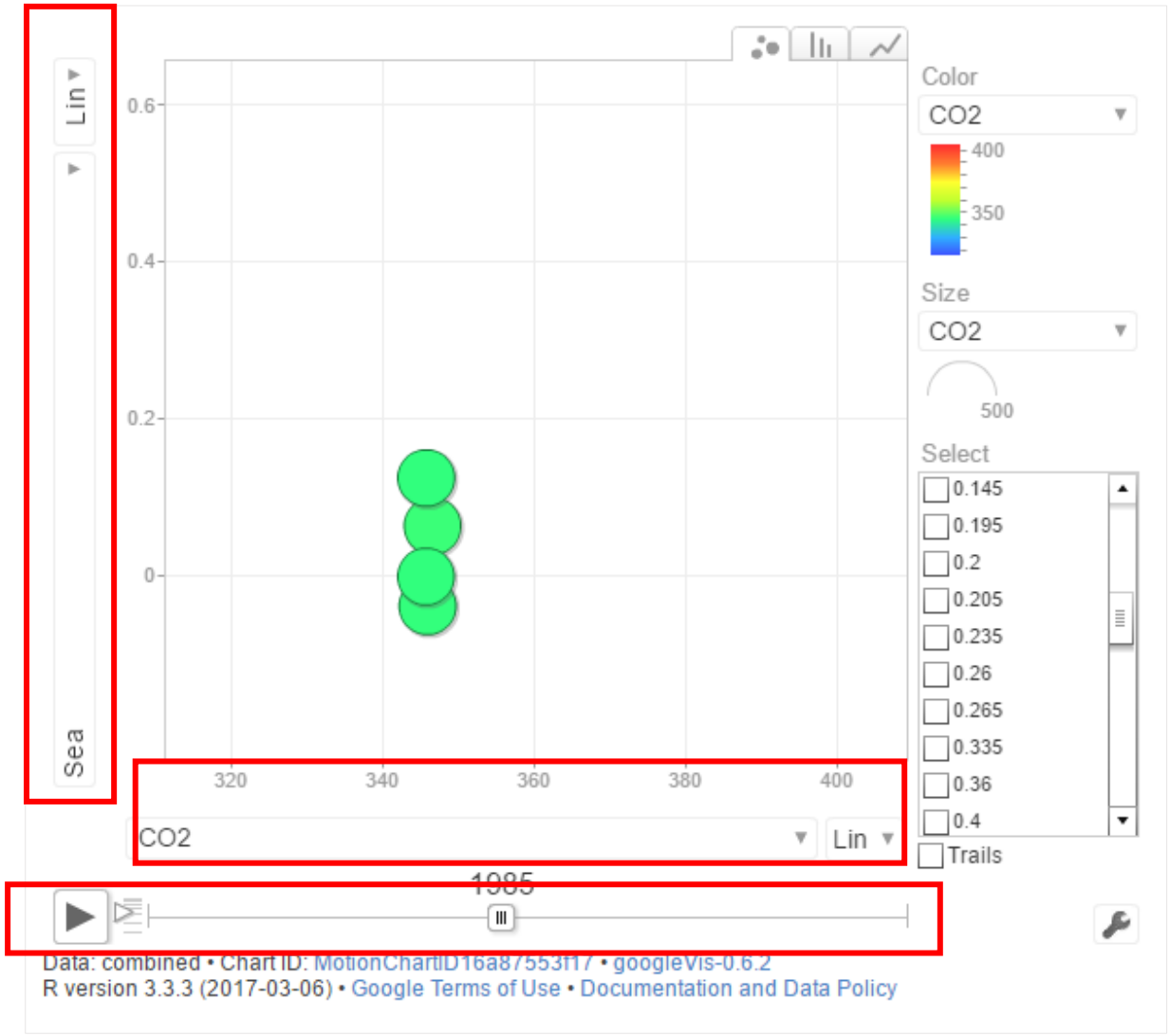
** meg equals 0.001 permil or 0.0001 percent or parts per million [[ppm]]*

7. Motion chart

For our motion chart we will use a combined data set showing the Keeling curve, land and sea temperature from 1959.

```
library(googleVis)
combined <- read.csv(file="combined.csv")
c <- gvisMotionChart(combined, idvar = "Land", timevar = "Year",
                     xvar = "Sea", yvar = "CO2")
plot(c)
```

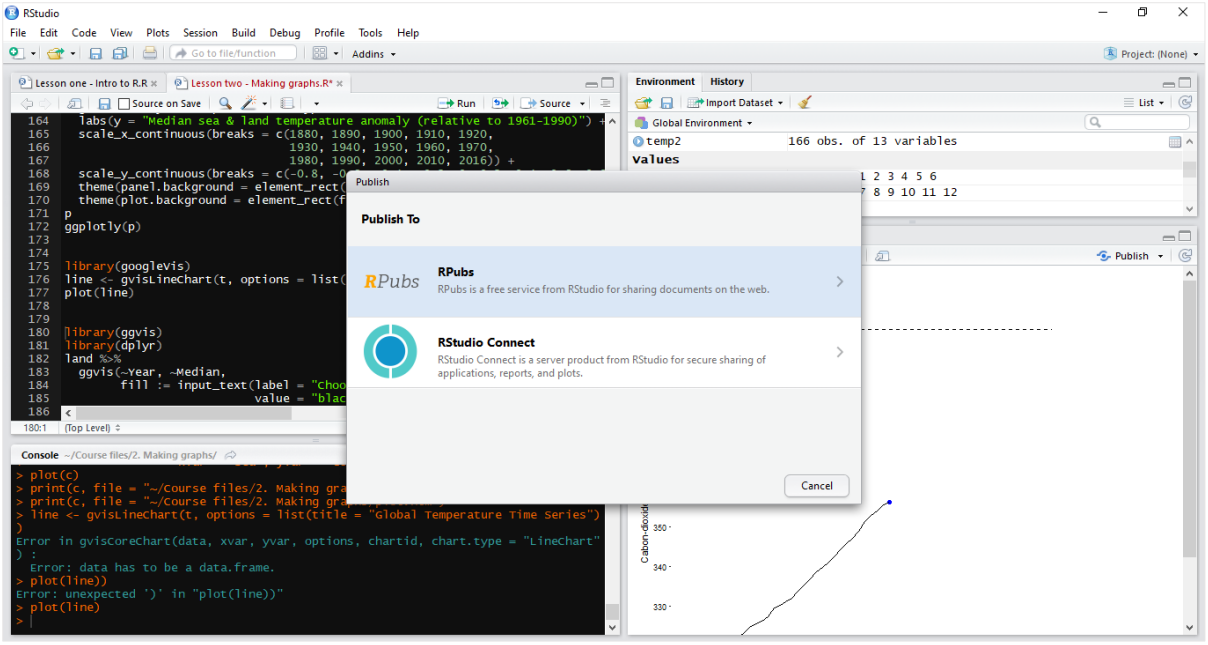
Similarly to the line chart, your motion chart should appear in your browser.



8. Publishing your plots online

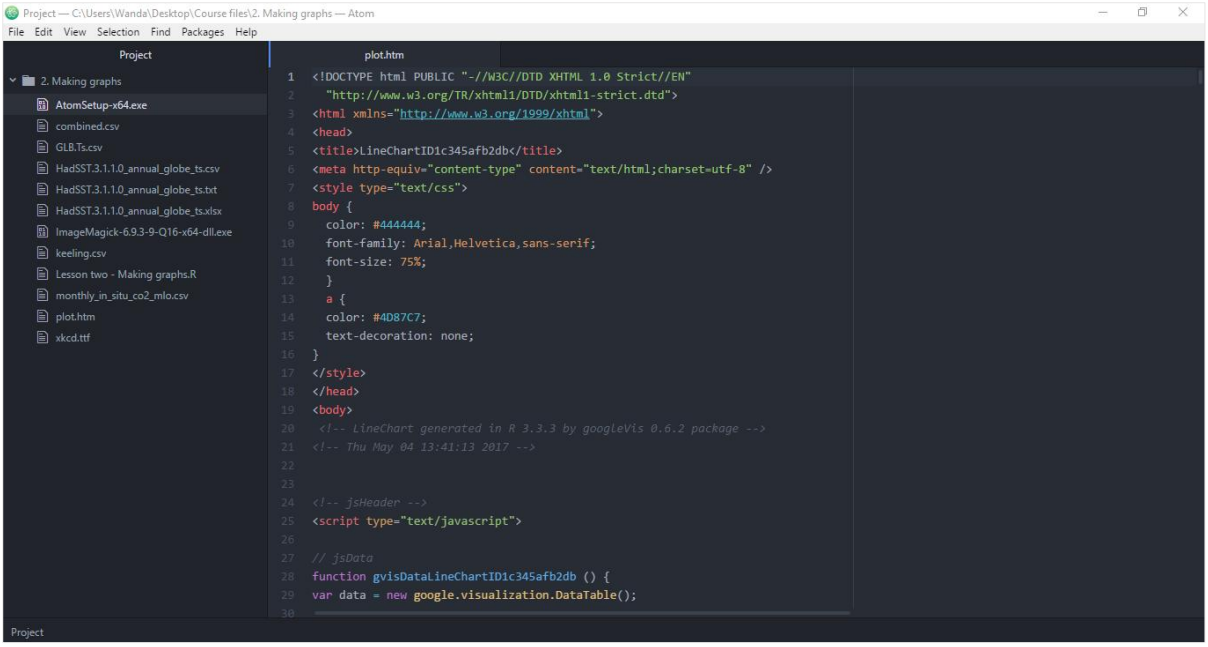
There are two ways you can publish your graphs online:

1) RPubs is a free service that you can use once you created an account. When your plot is ready just click Publish and choose RPubs.

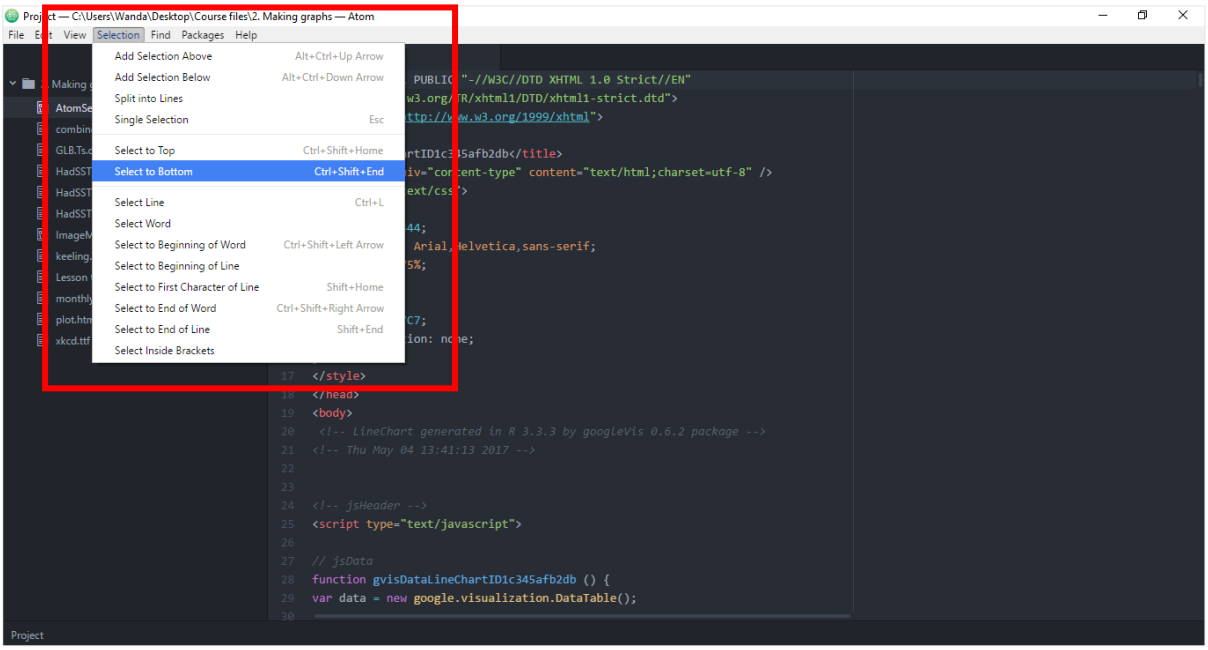


One you press Publish it will take you to the website where you can log in. You can give a title to your plot and when you hit Continue it will appear in your browser.

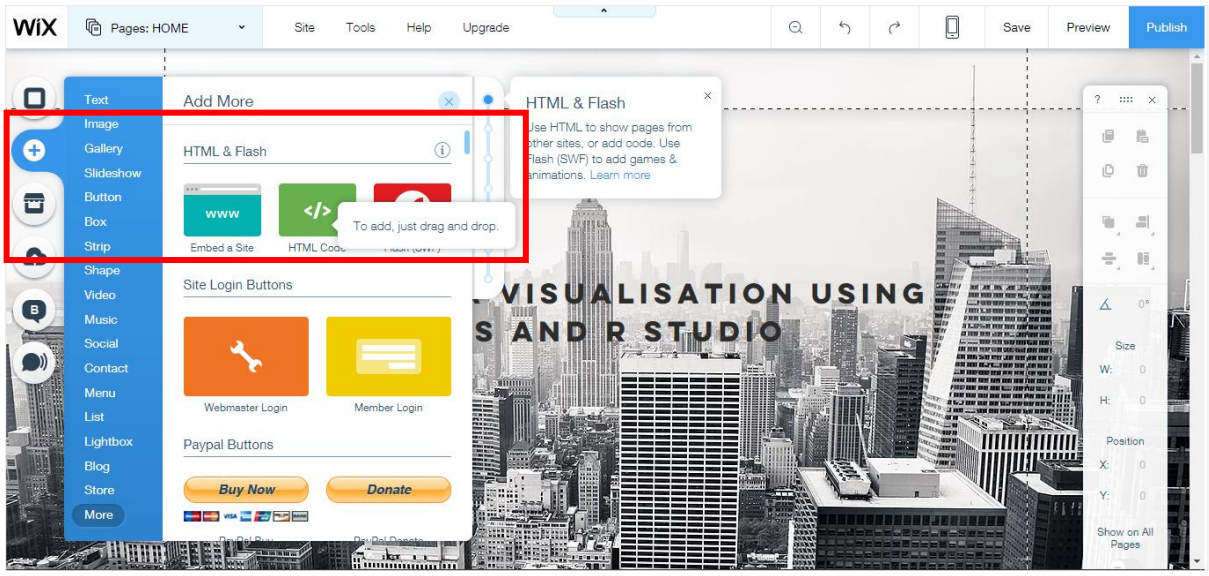
2) Another option is using the HTTP code that you can add to a website for example on WIX. To generate the HTTP code you need to install Atom (the .exe file is also available in the resource pack). One you installed Atom, just open the .html file that you want to use. For example, let's look at the googleVis plot we created in a previous exercise. Open the plot.html with Atom.



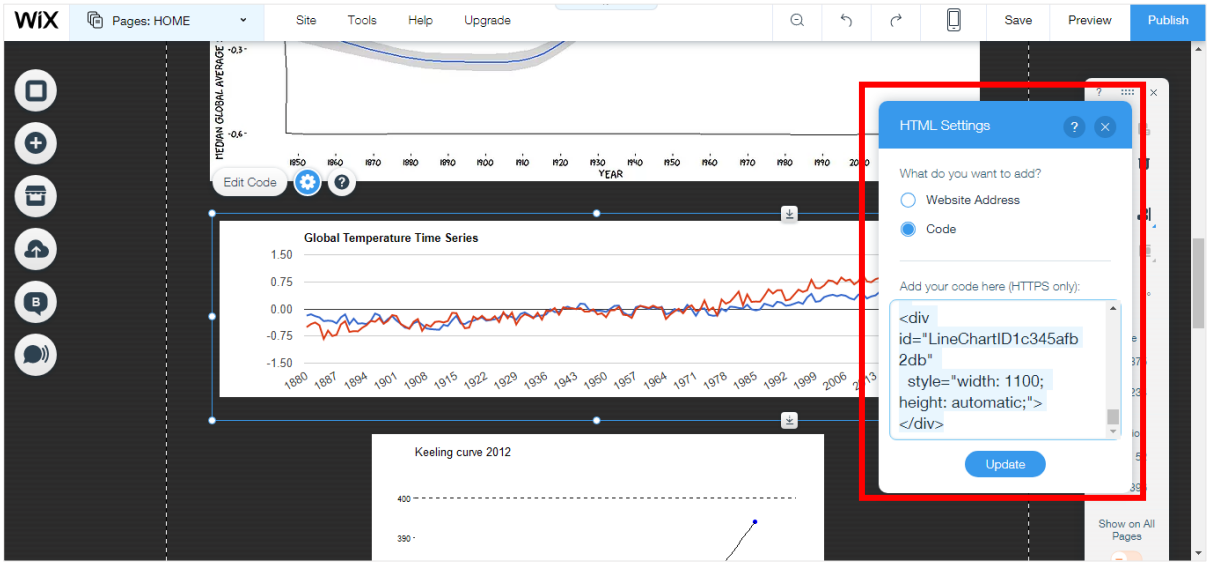
Using the Selection button click Select to Bottom. This will highlight the whole code. Copy it.



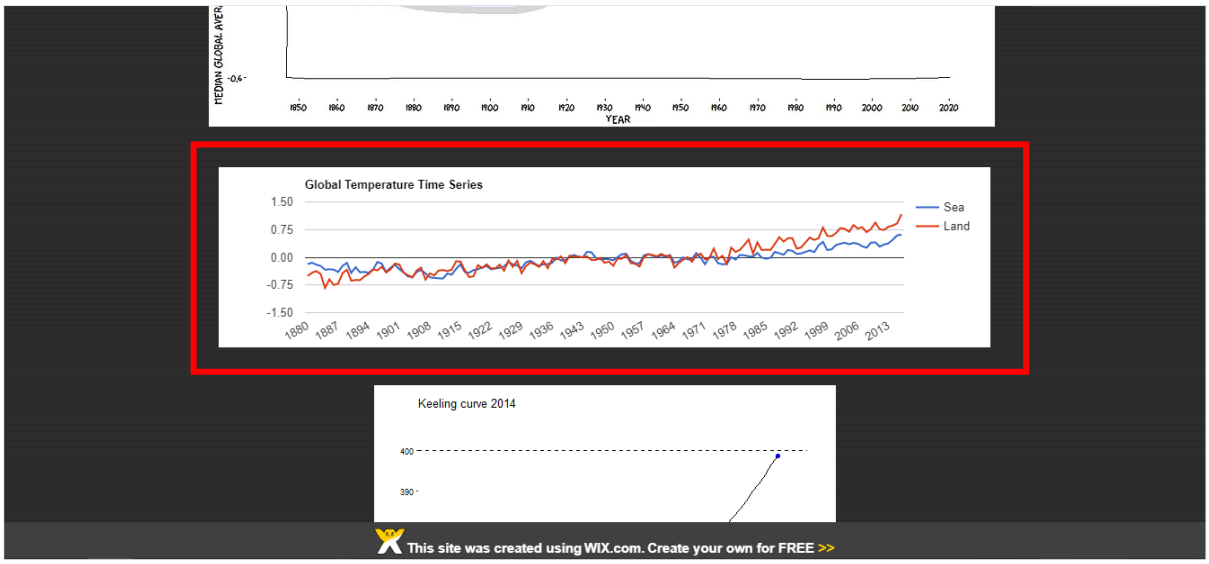
If you have a WIX account, go the Edit Site. Here you can add a HTML code window to the site.



Use Edit Code, then copy paste your code into the window provided.



Press Update and once you publish your website, the interactive plot should appear.



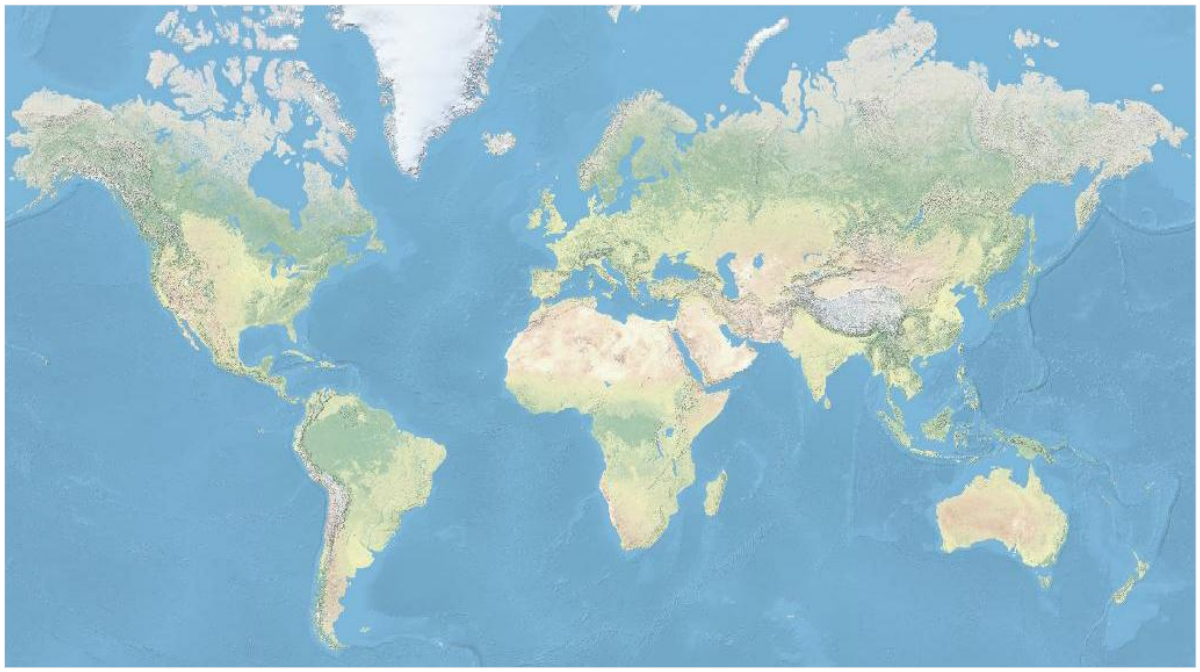
Introduction to Cartography and Mapping

Cartography, used as noun, is the creation of charts and maps based on geographical layout, and map is the visual representation of space. Map, used as a verb, means to create a visual representation of an area via cartography.

Maps store information about spatial patterns and relationships, and help us to understand environmental complexity.

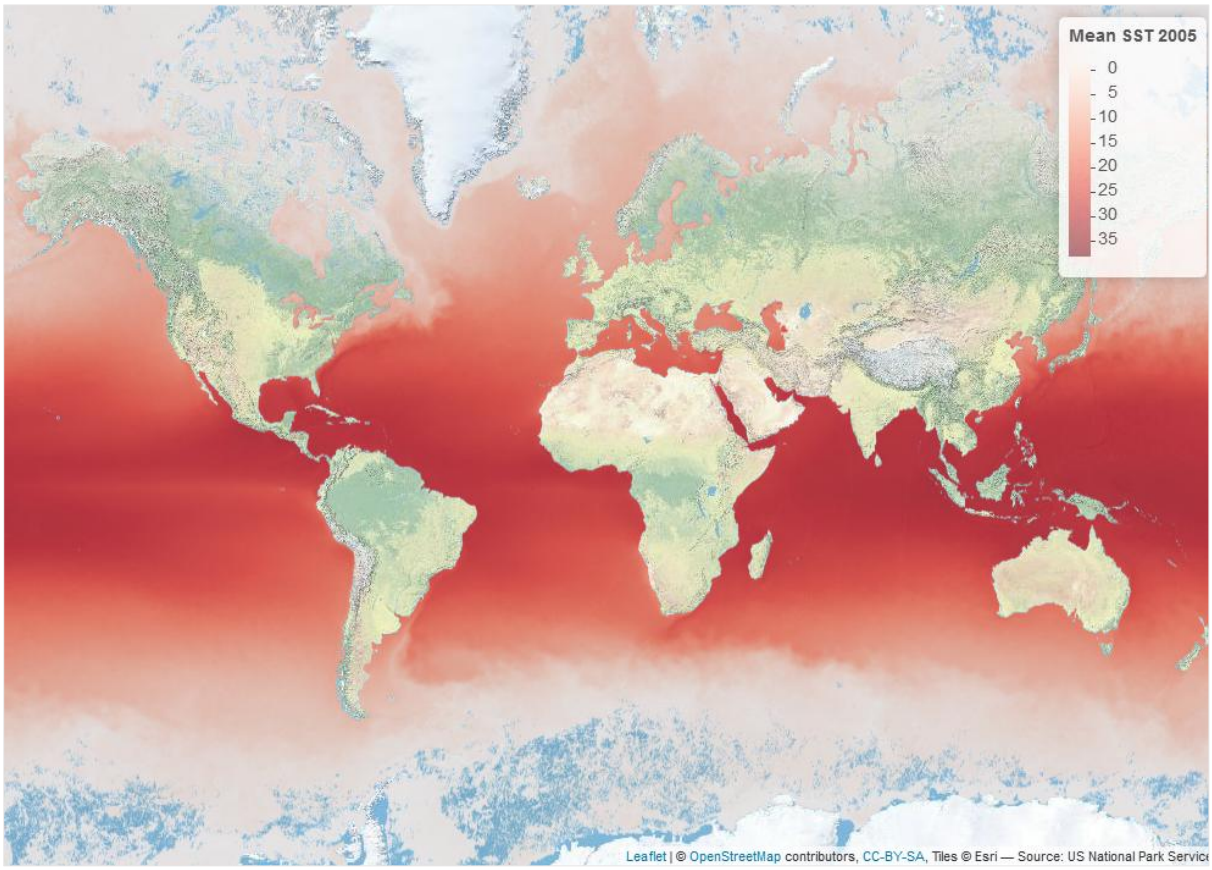
1. Map categories

Reference maps are used for general purpose with the main emphasis on location. It displays boundaries, names and unique identifiers of standard geographic areas, and can also show major cultural and physical features like roads, coastlines, rivers and lakes, just like the world topography map below.



```
library(rgdal)
library(raster)
library(sp)
library(leaflet)
library(ncdf4)

#topography
leaflet() %>% addTiles() %>% addProviderTiles("Esri.WorldPhysical")
```

Thematic maps can be both qualitative and quantitative, displaying a single topic with colours or symbols. For example, below the average sea surface temperature measured in 2005 is overlaid on a topography map.

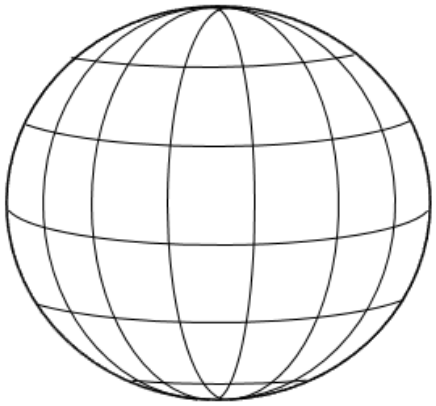



```
#sea surface temperature
#data source: https://giovanni.gsfc.nasa.gov/giovanni/
setwd("C:/Users/Kiddo/Desktop/GIS and Data Visualisation using R and
QGIS/Online course/Map types")
r <- raster("g4.timeAvgMap.MODISA_L3m_SST_2014_sst.20050101-
20051231.180W_90S_180E_90N.nc")
pal <- colorNumeric(c("Reds"), values(r), na.color = "transparent")
leaflet() %>% addTiles() %>%
  addProviderTiles("Esri.WorldPhysical") %>%
  addRasterImage(r, colors = pal, opacity = 0.8) %>%
  addLegend(pal = pal, values = values(r),
            title = "Mean SST 2005")
```

2. Graticules, distortion, and projections

Graticules and distortion

Before we talk about how map projections work, let's take the globe as an evenly spaced network of lines that run on the surface. These lines are called **graticules** and they represent the latitudes and longitudes.



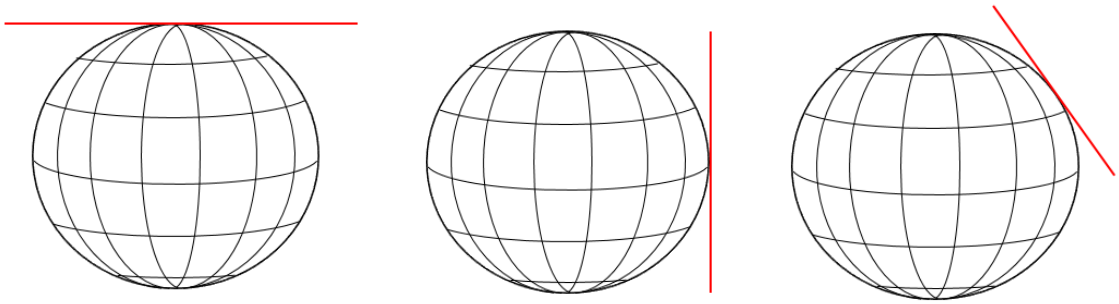
Now wrap a piece of paper around the globe and let's imagine a light shining from the middle of the sphere. You will see the light casting the shadow of the graticules on the paper. When you unwrap the paper you will see that the shape of the graticules will be different on the paper than on the globe.

Distortion occurs because of the curved surface becoming flat, and it affects shape, area and distance on a flat map. The different projections will preserve and/or distort different attributes. These are:
Conformal: preserves the local shape of small areas.
Equal area: preserves the area of displayed features.
Equidistant: preserves the distance between certain points.
Azimuthal: shows the shortest route between two points (straight line).

Projections

Map *projection* is the process of converting the spherical Earth to a flat surface. The three main type of projections are:

1) Planar/azimuthal: a plane is placed over the sphere. Depending on the position of the plane this can be either a) *polar*, b) *equatorial* or c) *oblique*.



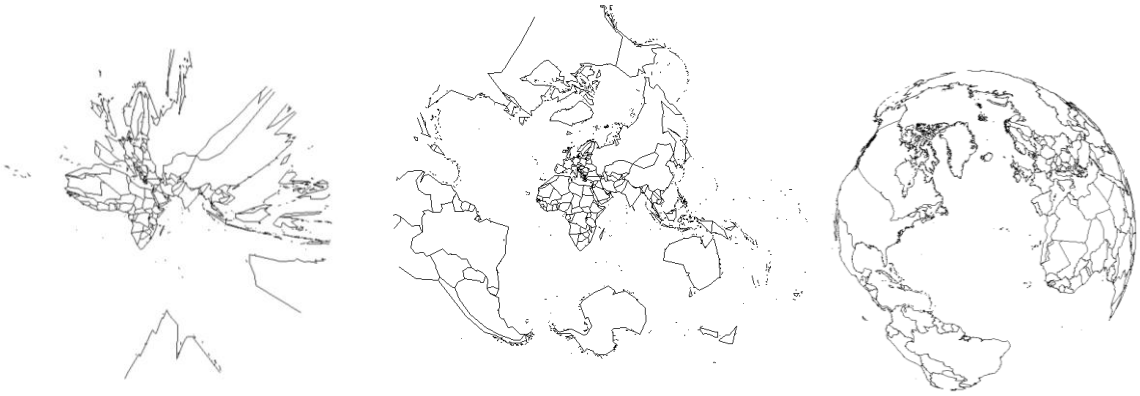
Polar Equatorial Oblique

The polar projection has three different perspectives.

i) Gnomonic: projection point from the centre of the globe.

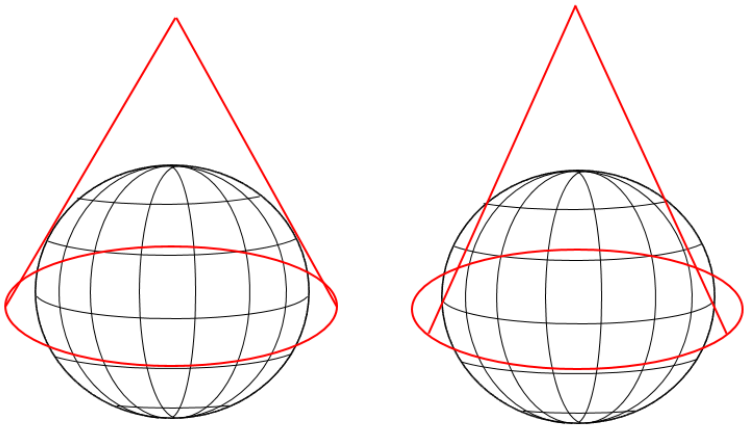
ii) Stereographic: projection point from the opposite side of the globe.

iii) Orthographic: projection from an infinite point.



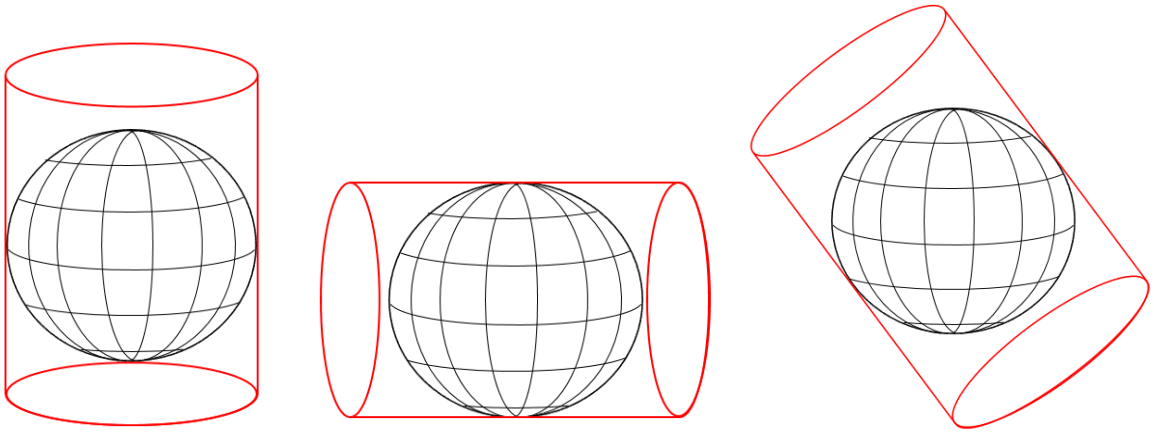
Gnomonic Stereographic Orthographic

2) Conic: a cone is placed over the globe. Conic projection can also be either a) tangent or b) secant. During conic tangent projection the cone is only touching the surface along a latitude line, while during conic secant projection the cone is cutting into the surface along two latitude lines.



Tangent Secant

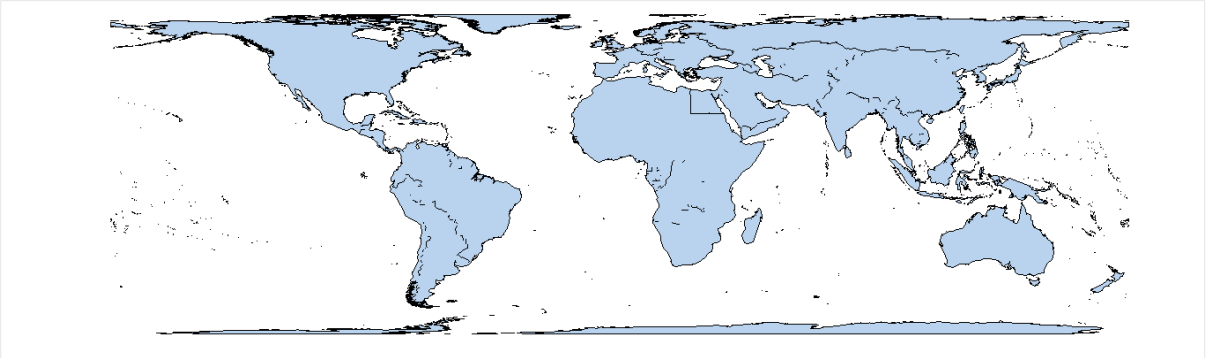
3) Cylindrical: a cylinder is placed over the globe that can touch either along a) a latitude line (normal), b) a longitude line (transverse) or c) another line (oblique).



Normal Transverse Oblique

Few examples of map projection types with R codes below them:

Equal Area Cylindrical



Undistorted along the Equator but distortion occurs towards the Poles.

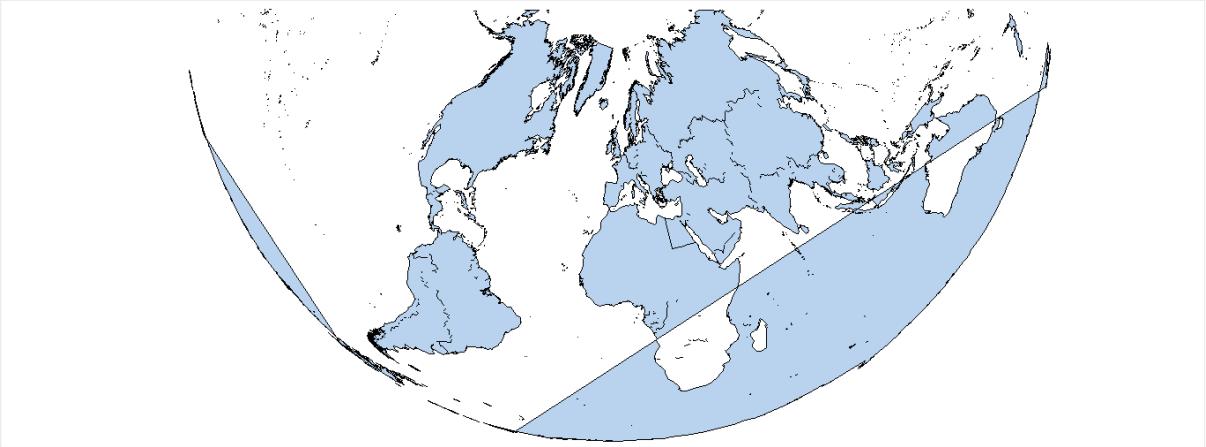
```
#equal area cylindrical
data(wrld_simpl)
poly <- Polygon(cbind(c(-180, 180, 180, -180, -180), c(-80, -80, 80, 80, -
80)))
clipPoly <- SpatialPolygons(list(Polygons(list(poly), ID = "1")),
proj4string = CRS(proj4string(wrld_simpl)))
wrld.clip <- gIntersection(clipPoly, wrld_simpl)
wrld.cea <- spTransform(wrld.clip, CRS("+proj=cea"))
plot(wrld.cea, col = "slategray2")
```

Lambert Conformal Conic

Portrays shape more accurately than area.

```
#lambert conformal conic
wrld.lamberconconic <- spTransform(wrld.clip, CRS("+proj=lcc"))
plot(wrld.lamberconconic, col = "slategray2")
```

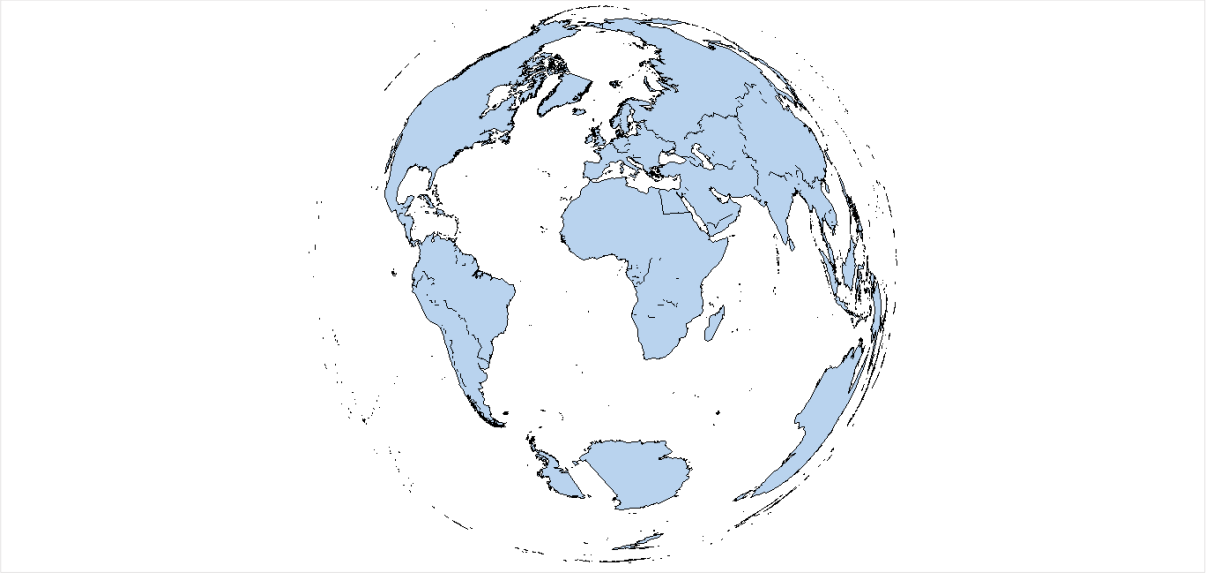
Lambert Equal Area Conic



Scale and shape are not preserved, distortion is minimal between the two standard parallels 20°N and 50°N.

```
#lambert equal area conic
wrld.lambereqcon <- spTransform(wrld.clip, CRS("+proj=leac"))
plot(wrld.lambereqcon, col = "slategray2”
```

Lambert Azimuthal Equal Area



Shows all areas but angles aren’t represented accurately.

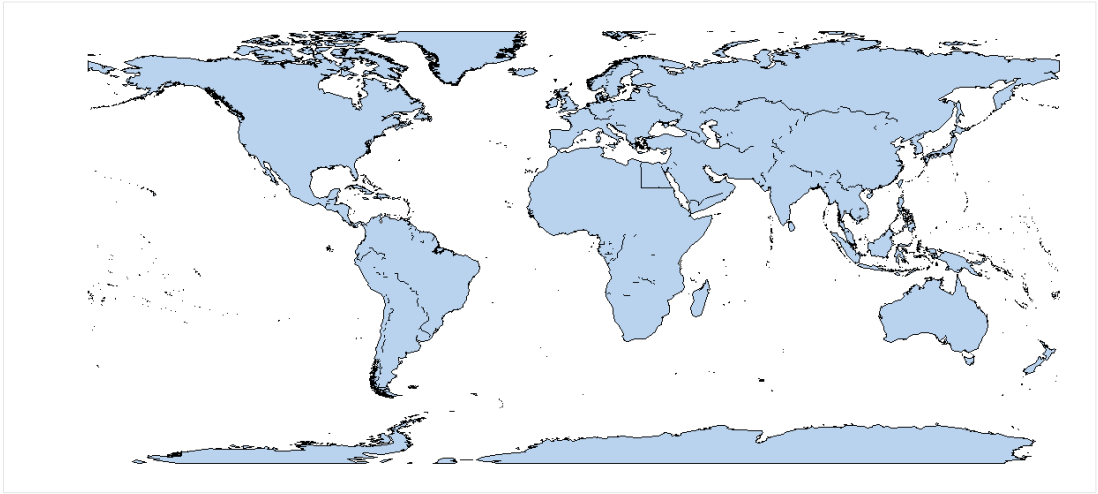
```
#lambert azimuthal equal area
wrld.lambereqcon <- spTransform(wrld.clip, CRS("+proj=laea"))
plot(wrld.lambereqcon, col = "slategray2")
```

Mercator

Created by Gerardus Mercator in 1569. It was a standard map used for nautical purposes. It preserves angles and shapes, but distorts the size of the objects from the Equator to the Poles.

```
#mercator
wrld.merc <- spTransform(wrld.clip, CRS("+proj=merc"))
plot(wrld.merc, col = "slategray2")
```

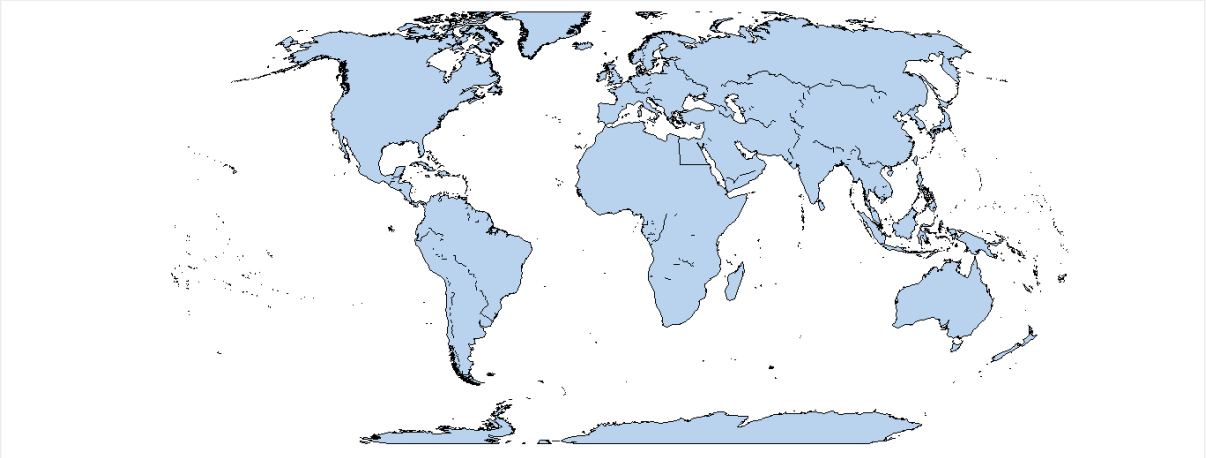
Universal Transverse Mercator



Distortion becomes worse as you move away from the central meridian.

```
#utm
wrld.utm <- spTransform(wrld.clip, CRS("+proj=longlat +datum=WGS84"))
plot(wrld.utm, col = "slategray2")
```

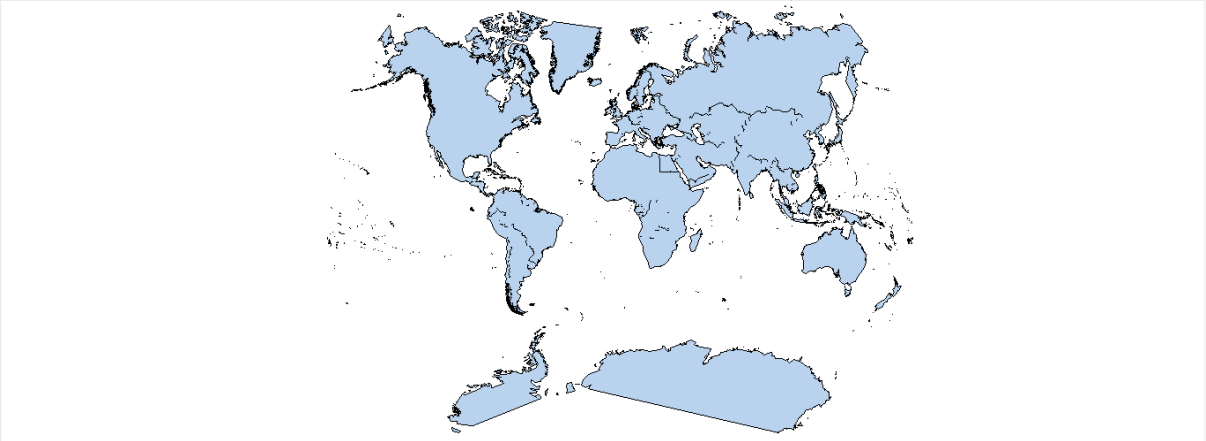
Robinson



It was created to show the whole globe on a flat image.

```
#robinson
wrld.robin <- spTransform(wrld.clip, CRS("+proj=robin"))
plot(wrld.robin, col = "slategray2")
```

Van der Grinten



This projection is neither equal area nor conformal; it shows the globe on a circle with distorted Polar regions.

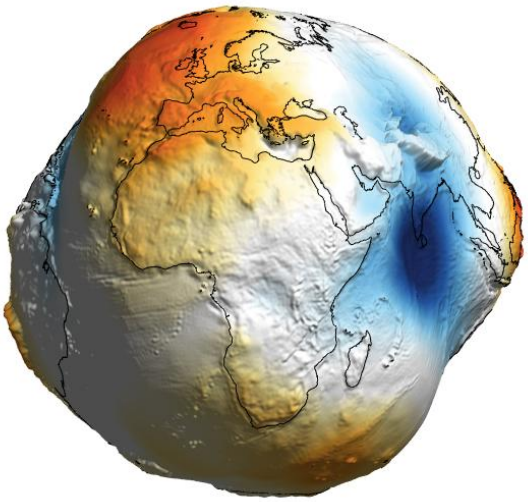
```
#van der grinten
wrld.vandg <- spTransform(wrld.clip, CRS("+proj=vandg"))
plot(wrld.vandg, col = "slategray2")
```

3. Geoid, ellipsoid & datum

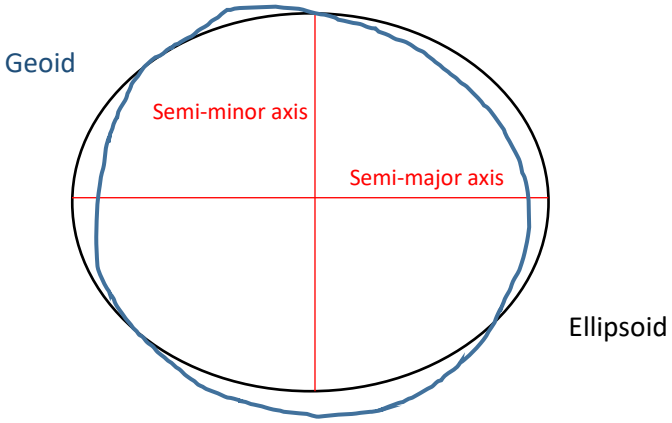
Although the Earth is round in shape, due to rotation, geology and gravitational pull, the surface of the Earth is actually irregular.

Round shape

Irregular shape

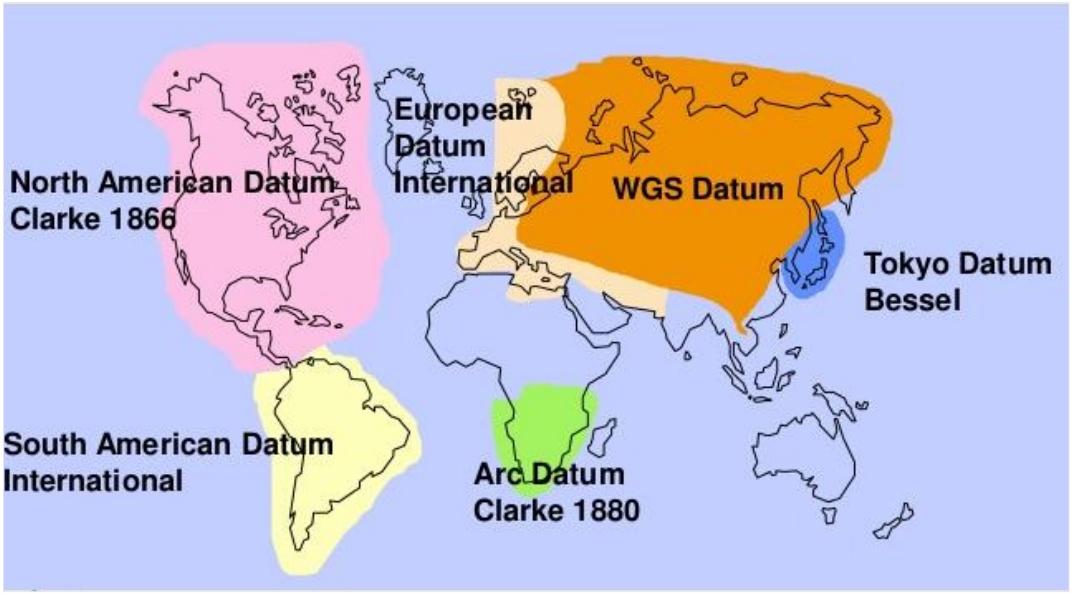


In order to model the surface of the Earth scientists use the *geoid*. The geoid accounts for the reality of Earth's surface (i.e. topographic changes) and it uses the global mean sea level to approximate the surface. Since the geoid is not a smooth surface, the *ellipsoid*, which tries to approximate the geoid, is used. If you imagine using a semi-minor and a semi-major axis to adjust the curvature of the Earth, you can model surface depending on your geographical location.



Datum defines the position of the ellipsoid relative to the centre of the earth and it uses local variations in elevation. It is based on a network of benchmarks with known latitude and longitude coordinates. Different countries use different datum. For example, North American countries started to use NAD27 (North American Datum 1927) which was followed by the NAD83 (North American

Datum 1983). The common datum used in almost all handheld GPS, phone applications and by the NASA is the WGS84 (World Geodetic System 1984).



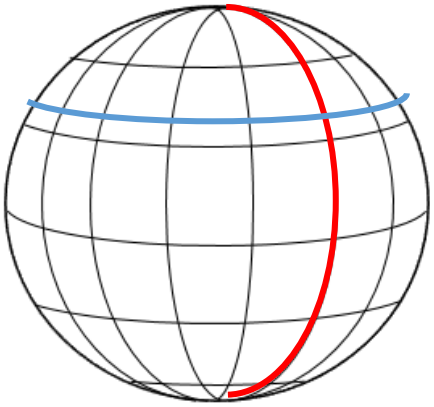
(Modified after Neuralog)

4. Coordinate reference systems

Whenever a map is created two types of coordinate reference systems (CRS) are used: geographic and projected.

In the **geographic coordinate system (GCS)**, the Earth has an ellipsoid shape (bulges along the equator and flattens at the Poles). GCS references a datum, prime meridian and an angular unit of measure.

The **projected coordinate system (PCS)** is the systematic transformation of locations on the earth to a flat surface. It uses x and y coordinates to identify a location.



Longitude: angular distance running parallel the prime meridian from north to south. The x-coordinate values range between -180 to 180 degrees. Lines of longitudes are also referred to as meridians.
Latitude: angular distance running parallel with the equator from east to west. Measured from the centre of the Earth. The y-coordinate values range between -90 to 90 degrees.

Latitude and longitude coordinates can be expressed as degrees, minutes and seconds (DMS), decimal degrees (DD) and decimal minutes (DDM). (1 degree = 60 minutes and 1 minute = 60 seconds.) You can convert DMS coordinates to DD coordinates. Let's use London as an example:

DMS Latitude and Longitude	DM Latitude and Longitude
51° 30' 35''	51.50972
0° 7' 5''	-0.1180

1: convert DMS seconds to minutes (51° 30' **35''**)

35'' / 60 = 0.583

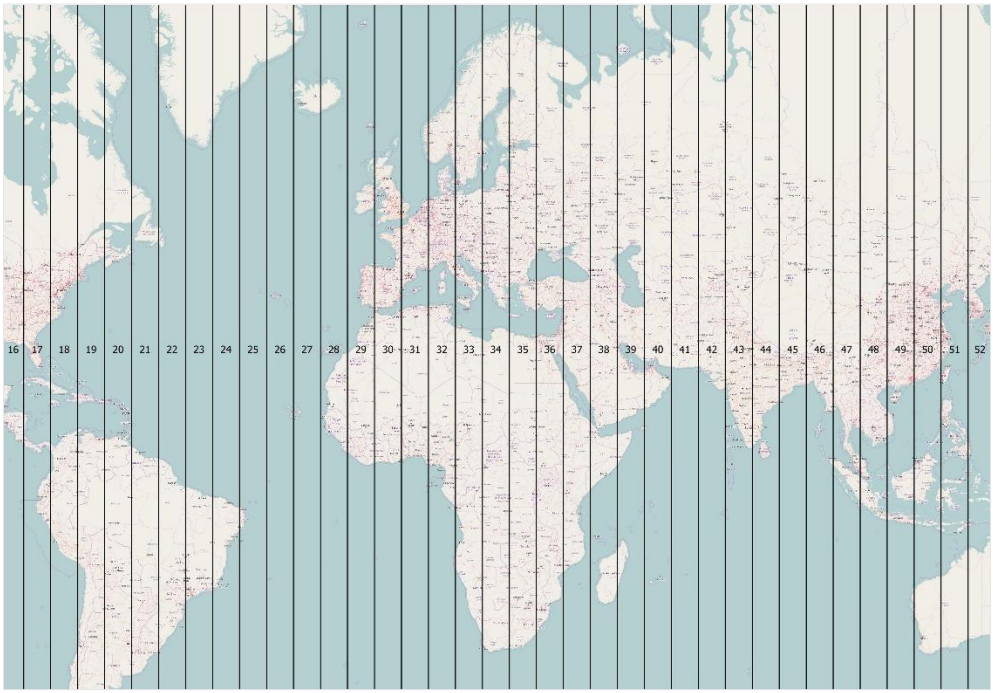
2: Convert DMS minutes to degrees (51° **30'** 35'')

30.583' / 60 = 0.50972

3: Combine the result with the original degree (**51°** 30' 35'')

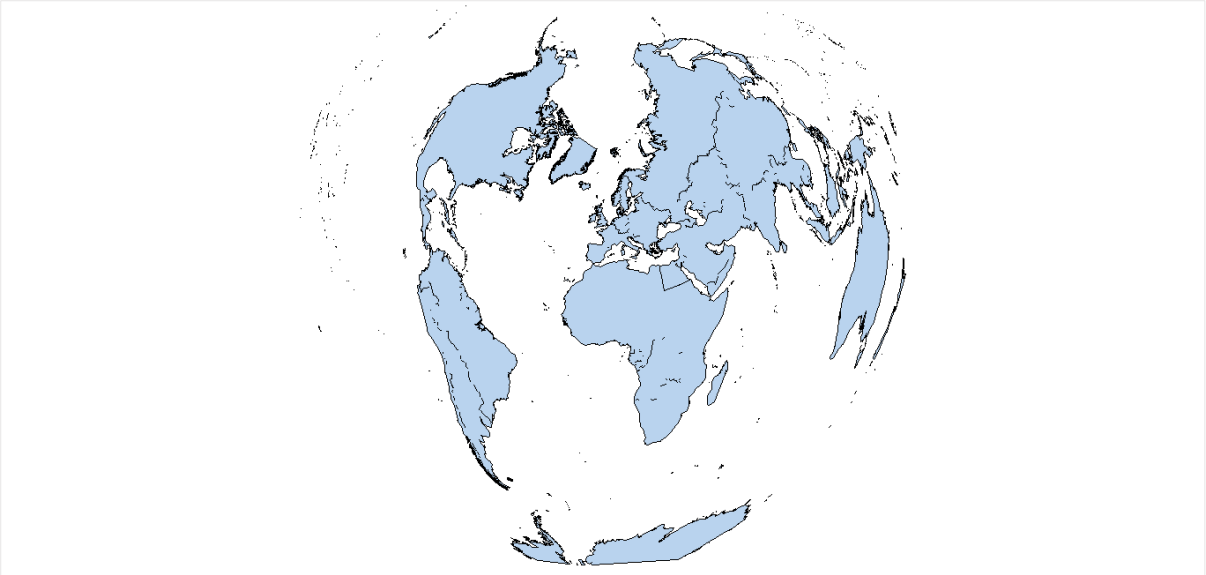
51.50972

In addition, coordinates can also be expressed in UTM (Universal Transverse Mercator) values. Instead of latitude and longitudes, UTM uses the transverse Mercator projection and it divides the Earth into sixty zones.



A position is given using the UTM zone number and the easting (x) and northing (y) coordinate pair of that zone. For example, the City of London is represented by 30U 699978E 5710464N. (30U is the zone number.)

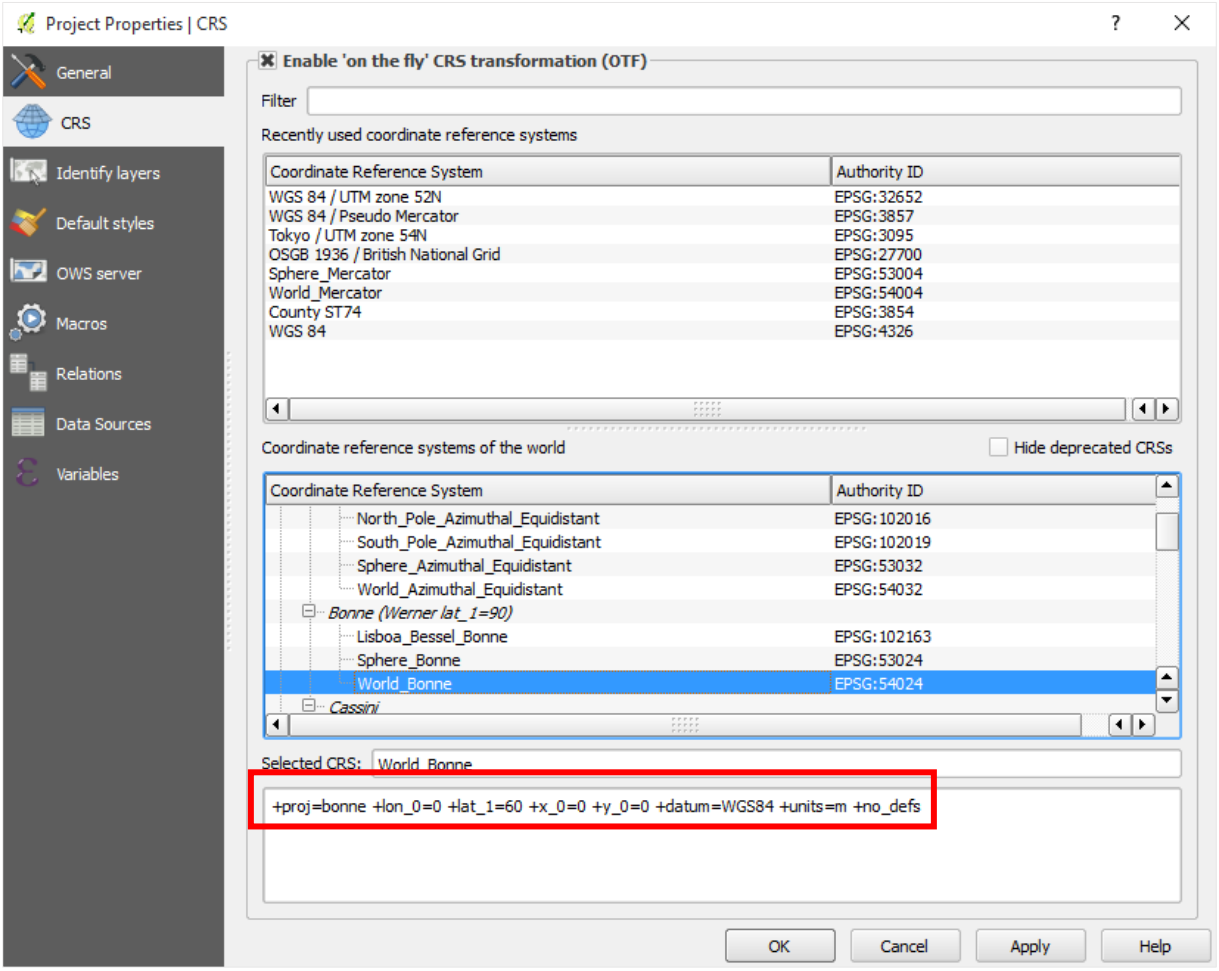
Both in R Studio and QGIS you can use the CRS IDs. See the example below for the World Bonne projection:



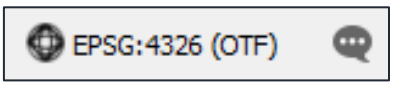
In R Studio:

```
data(wrld_simpl)
poly <- Polygon(cbind(c(-180, 180, 180, -180, -180), c(-80, -80, 80, 80, -
80)))
clipPoly <- SpatialPolygons(list(Polygons(list(poly), ID = "1")),
proj4string = CRS(proj4string(wrld_simpl)))
wrld.clip <- gIntersection(clipPoly, wrld_simpl)
wrld.bonne <- spTransform(wrld.clip, CRS("+proj=bonne +lon_0=0 +lat_1=60
+x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs"))
plot(wrld.bonne, col = "slategray2")
```

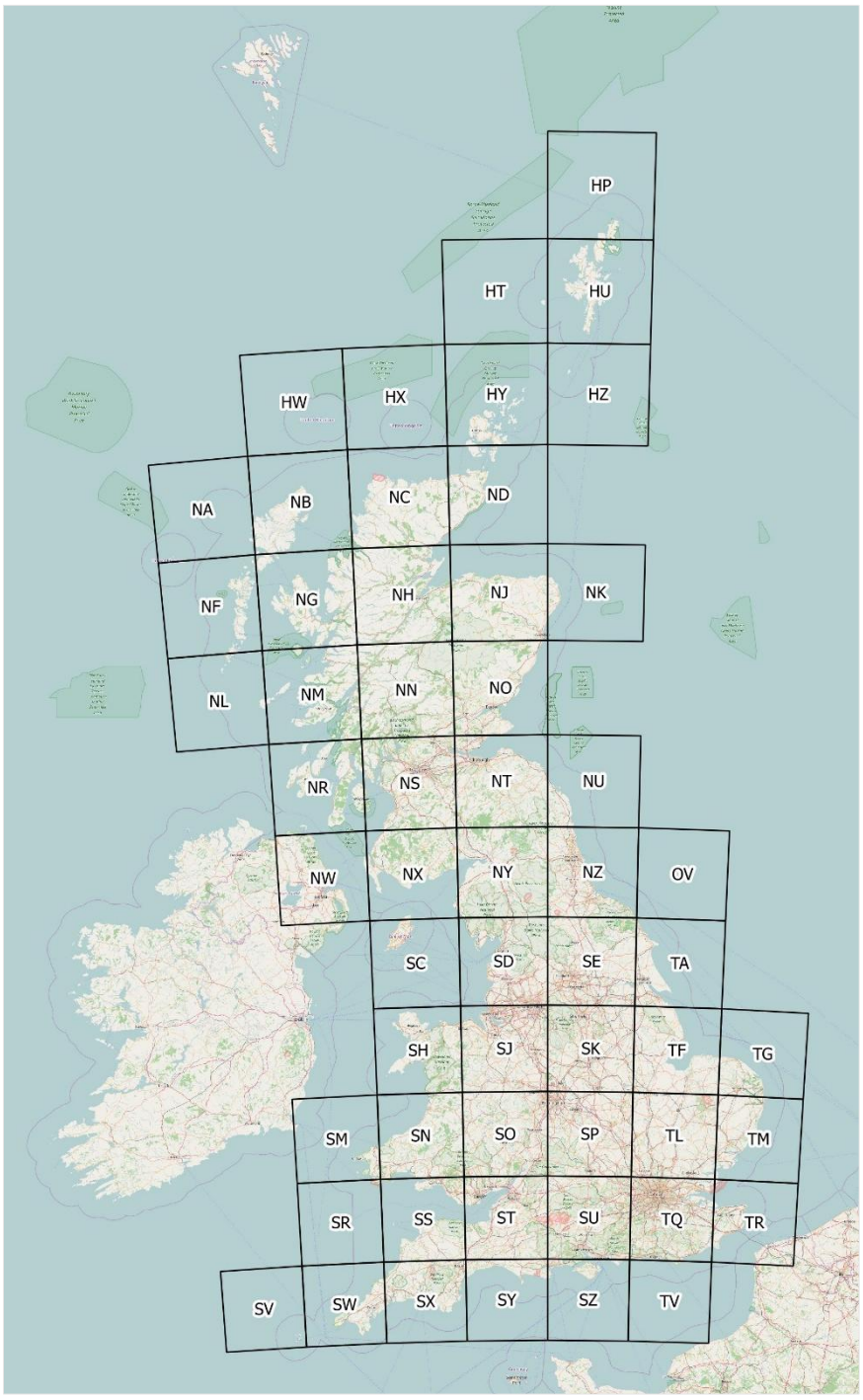
In QGIS:



You can choose the coordinate reference system by clicking Project → Project Properties, or on this icon at the bottom right on the main page:



Grid reference system: instead of latitudes and longitudes, grid references can also be used. The British National Grid is based on the OSGB36 datum (Ordnance Survey Great Britain 1936). This system breaks Great Britain down into progressively smaller squares: the largest unit of the grid is 500km squares, which is then further broken down into 100km, 10km and 1km squares. Each square is designated by a prefix letter: H, N, S and T and digits (easting and northings), and it uses the south-west corner of the square as a reference point



This is an excellent website with an openly available software that can help convert grid references to UTM coordinates. Just download the NGConv.exe file (also available in the resource pack), and drag and drop your .txt file with the grid references onto the ngconv icon. It will give you a .csv file with the UTM coordinates.

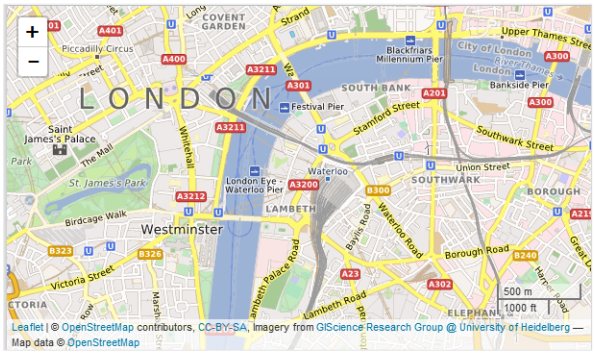
Introduction to GIS and QGIS

1. Geographical Identification System (GIS)

Geographical Identification System: capturing, assembling, manipulating, analysing and displaying geographically referenced information. GIS looks into locations, conditions, and it is used to find trends and analyse patterns.

The two main scales that GIS uses are:

Local scale



Planetary scale

Spatial and attribute data

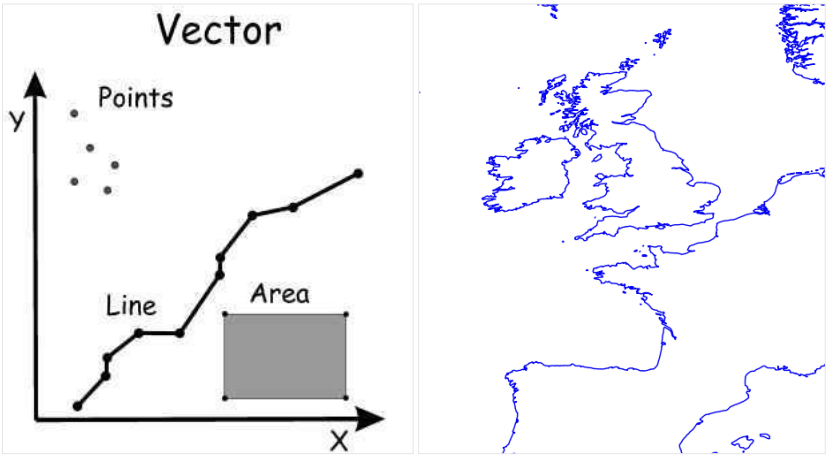
GIS uses two types of data:

1, *Spatial data*: answers the question 'Where?' and stores the information as a shapefile.

2, *Attribute data*: answers the questions 'What?', 'When?' and 'How much?', and it displays specific characteristics about the spatial data.

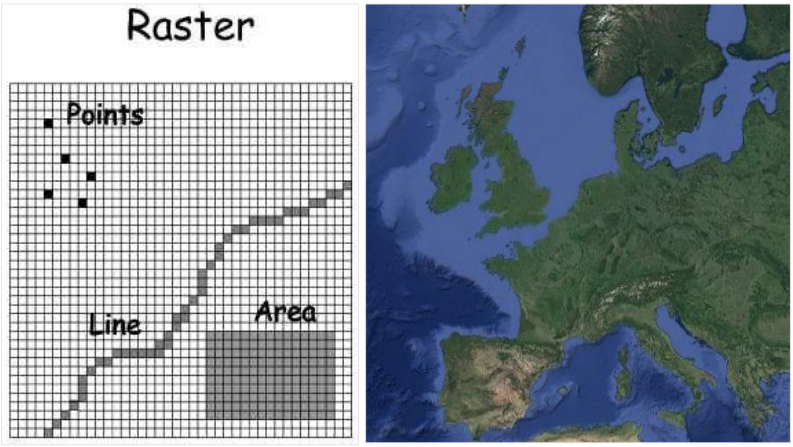
Vector and raster data

Vector data: shows positional information (latitude/longitude or Universal Transverse Mercator grid coordinates) with distinct boundaries defined by points, lines and polygons.



Vector data model in GIS (Bolstad, 2005)

Raster data: is a graphical representation of features. It shows continuous data, and it builds up from equal-size cells (grids), arranged in rows and columns, each with a value. For example, images used for satellites can be viewed in GIS programs.



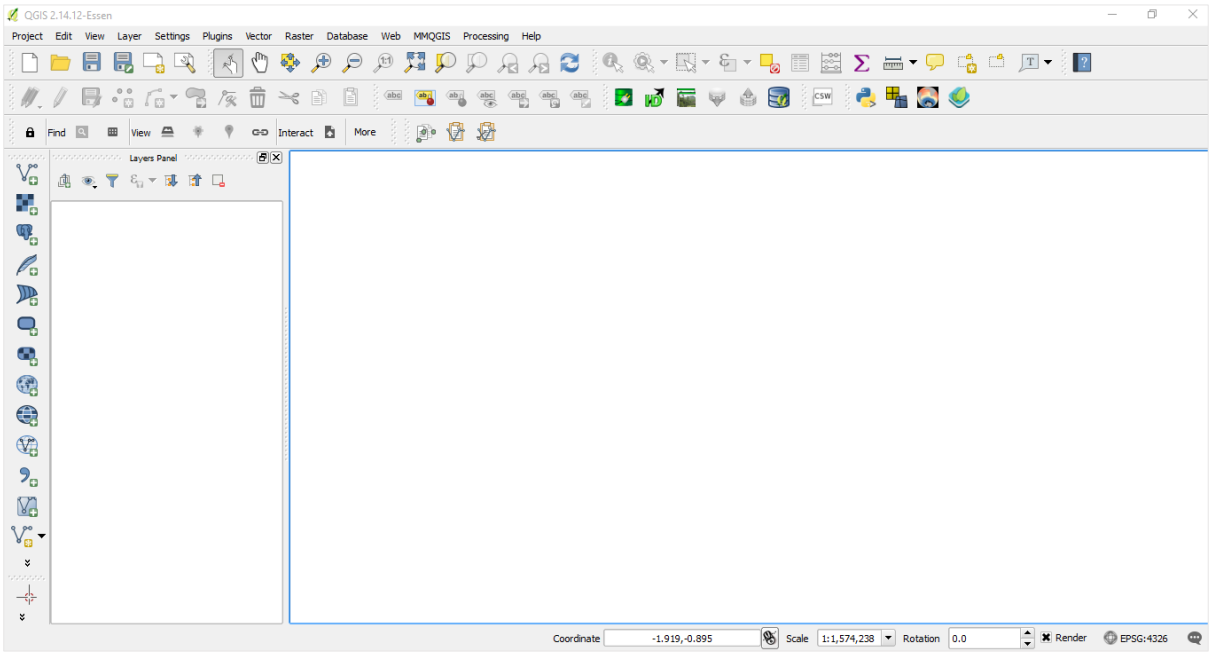
Raster data models in GIS (Bolstad, 2005)

2. Quantum GIS (QGIS)

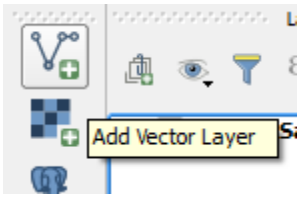
Installing QGIS and its layout

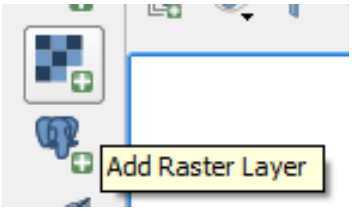
For this course we will be using an open source GIS software called Quantum GIS or QGIS. We will be using the 2.14 (64bit) version which is available to download using this link: http://www.qgis.org/en/site/forusers/download.html#.

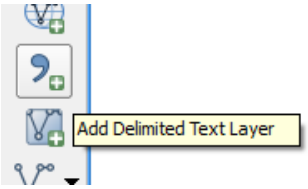
After you have downloaded and installed the software, look for the QGIS Desktop 2.14.12 and open it.



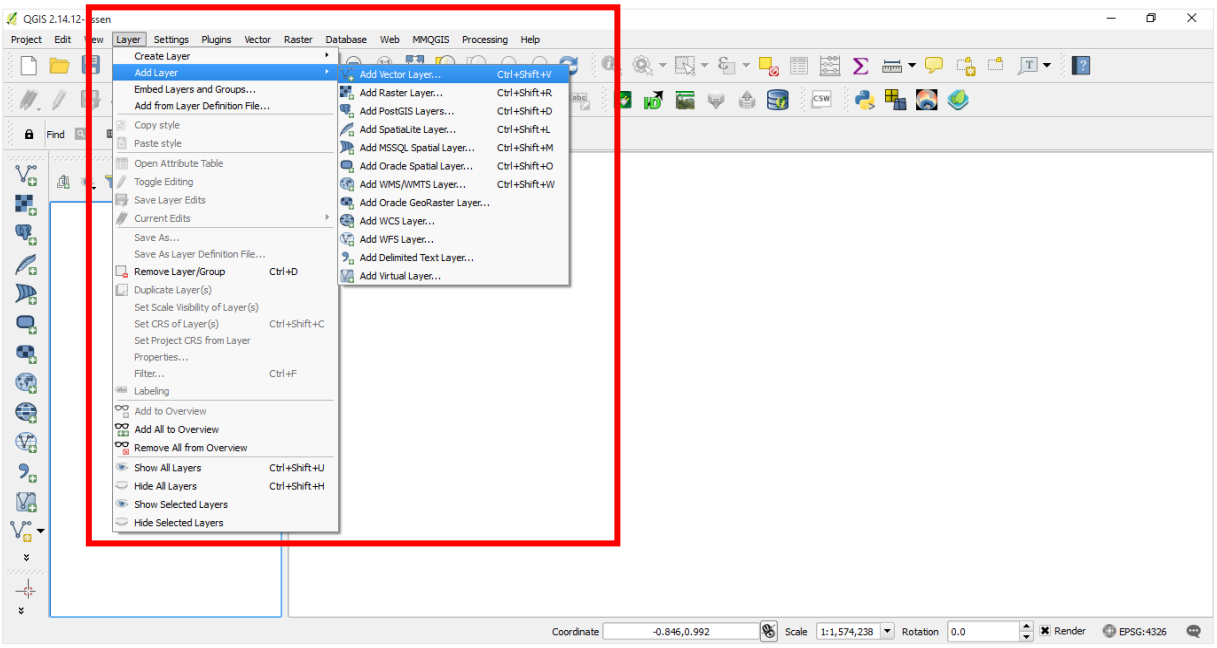
You will see many icons on the top and on the side. If you hover over these with your cursor, a text will appear explaining what you can do with them. For example:





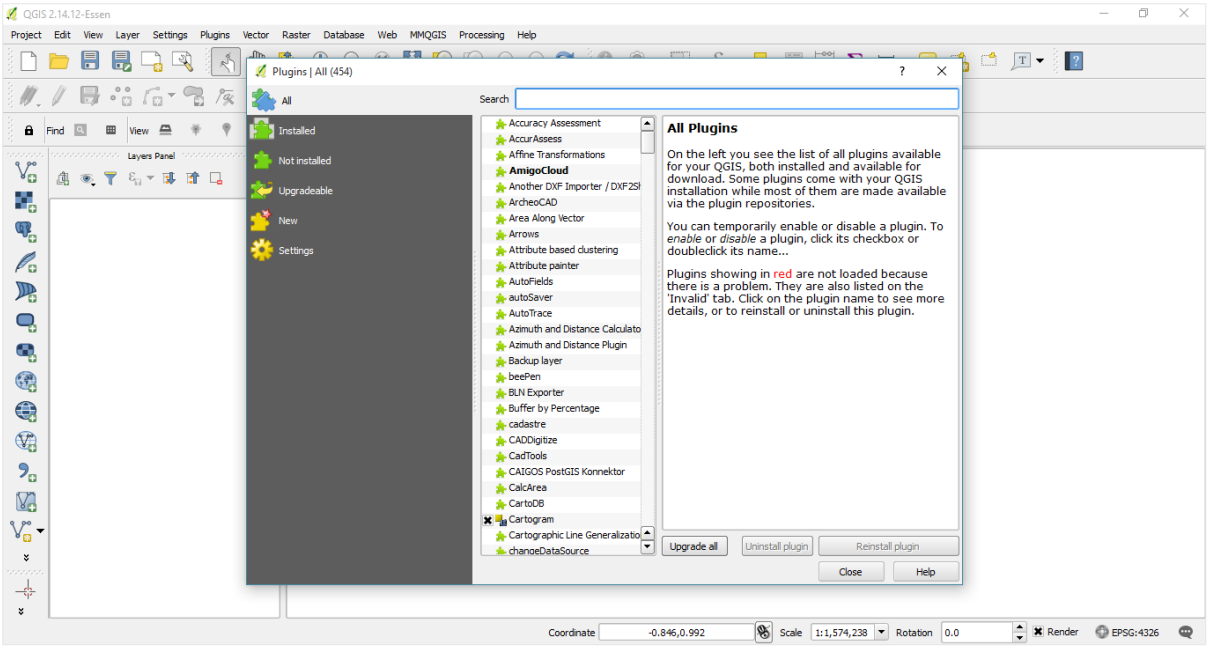


You can also use the menu points on the top to add the various layers.

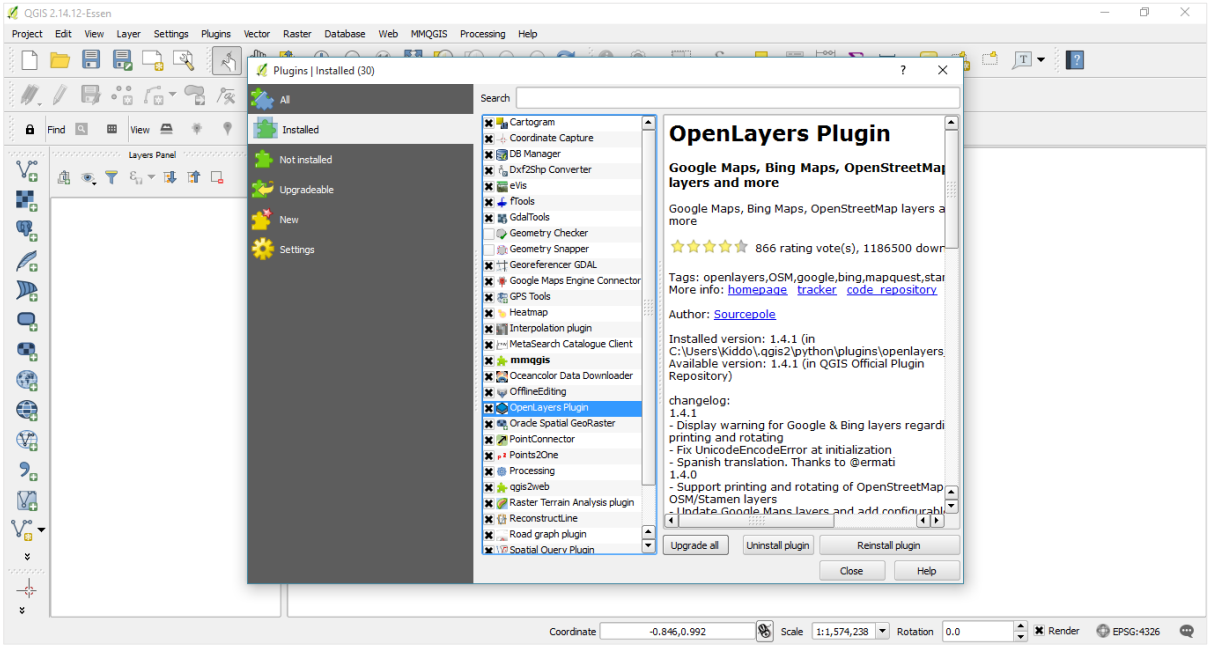


Adding Plugins

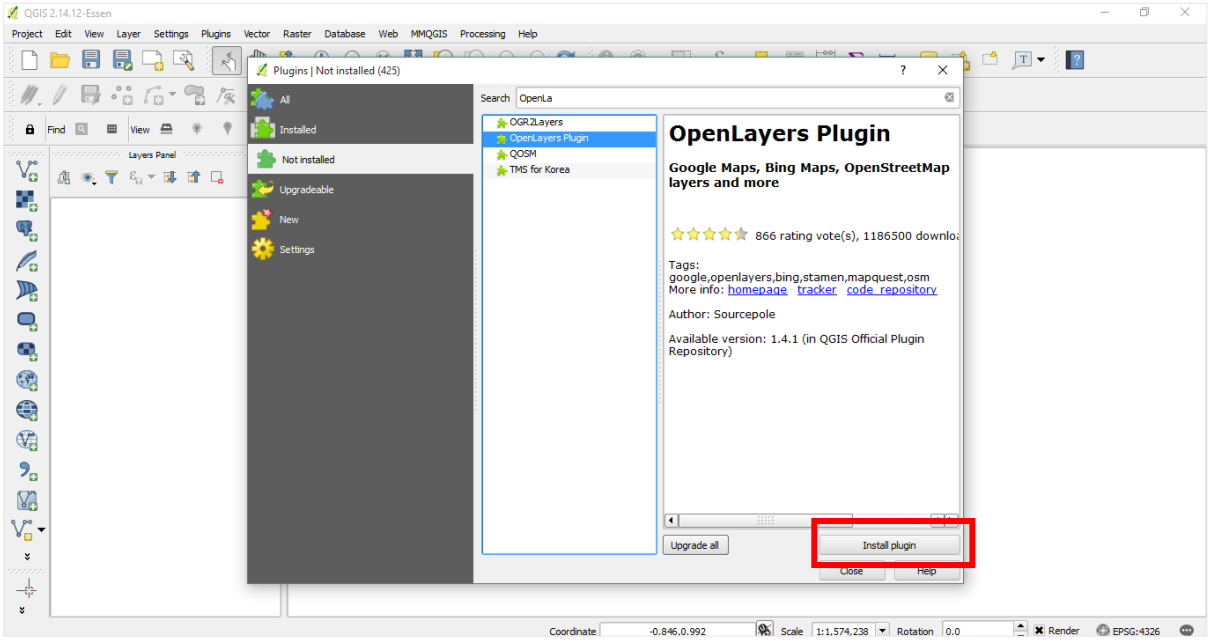
There are various plugins available that you can add to QGIS. Go to Plugins → Manage and Install Plugins. You will see a new pane appear with a list of choices.



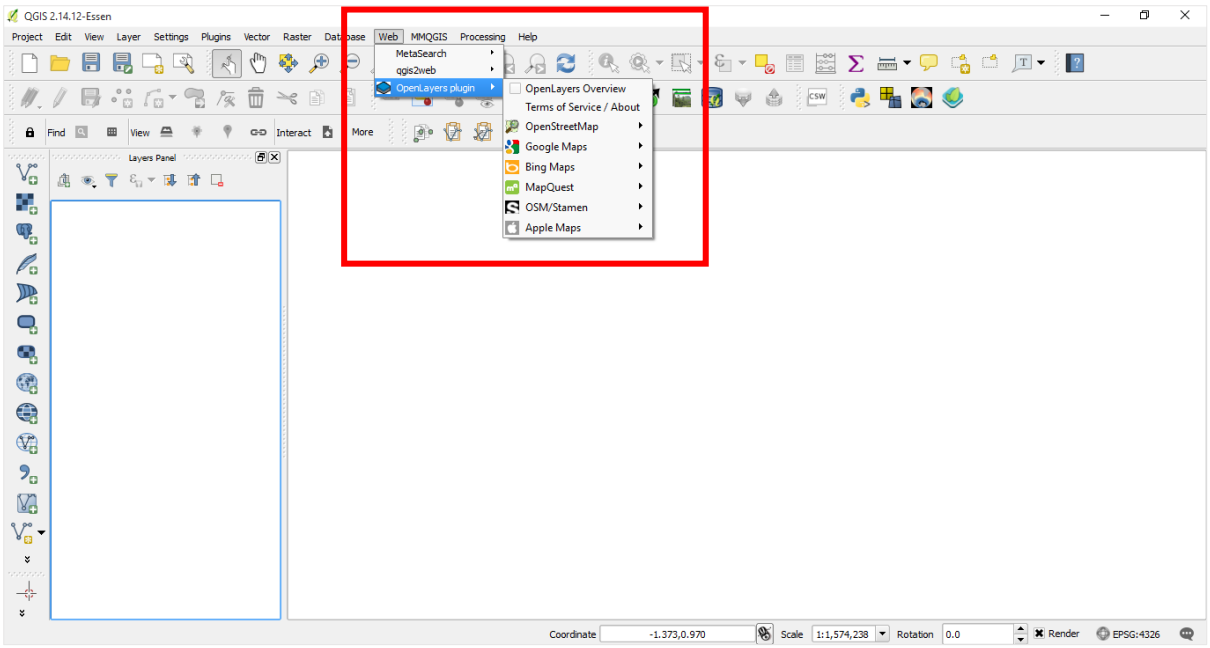
If you click on Installed, you will see the list of plugins that you have. Here you already have the OpenLayers Plugin installed.



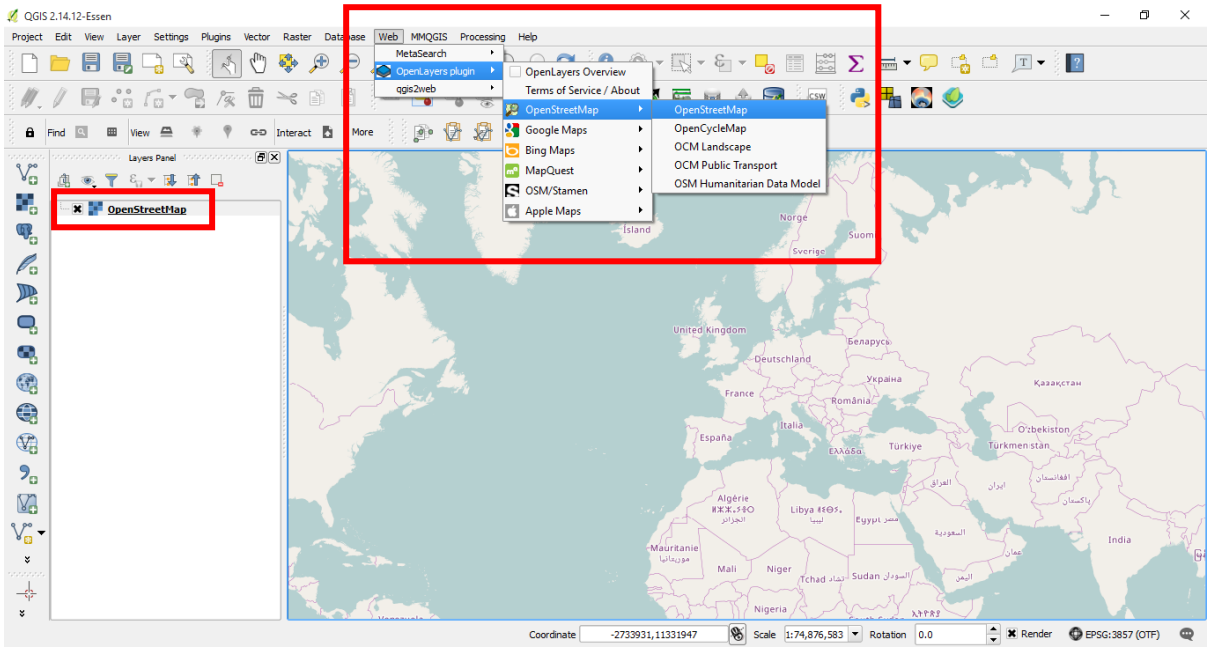
If you can't find it, you can easily install it by going to Not Installed and in Search, start to type OpenLayer and the plugin should appear.



Click on Install Plugin. QGIS Python Installer should have installed and you can check it by clicking on the Installed icon. You can also reach it by clicking on Web → OpenLayers Plugin.



If you click on OpenStreetMap from the dropdown menu, you will see it appearing in your main window. You can zoom in and out using your mouse, and you can turn the layer on and off by ticking the box next to OpenStreetMap in the Layers Panel.



Now let's create a new project by clicking on Project → New or on this icon:

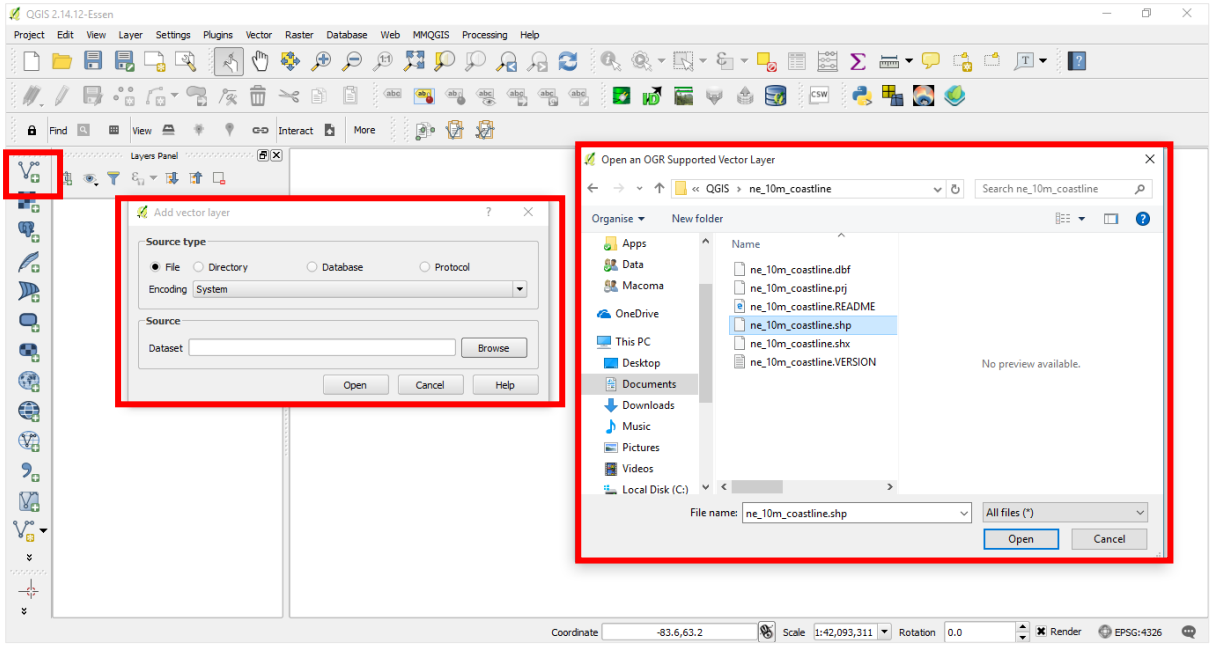
Working with shapefiles

For the next exercise, we will use data from the Natural Earth Data website. (These shapefiles are also available in the course file folder.) Download the following 1:10m shapefiles: coastline, land and reefs. Once you have the data, extract the zipped folders. You will see that each folder contains the following files: .dbf, .prj, .shp, and .shx. The table below explains what the different extension formats mean:

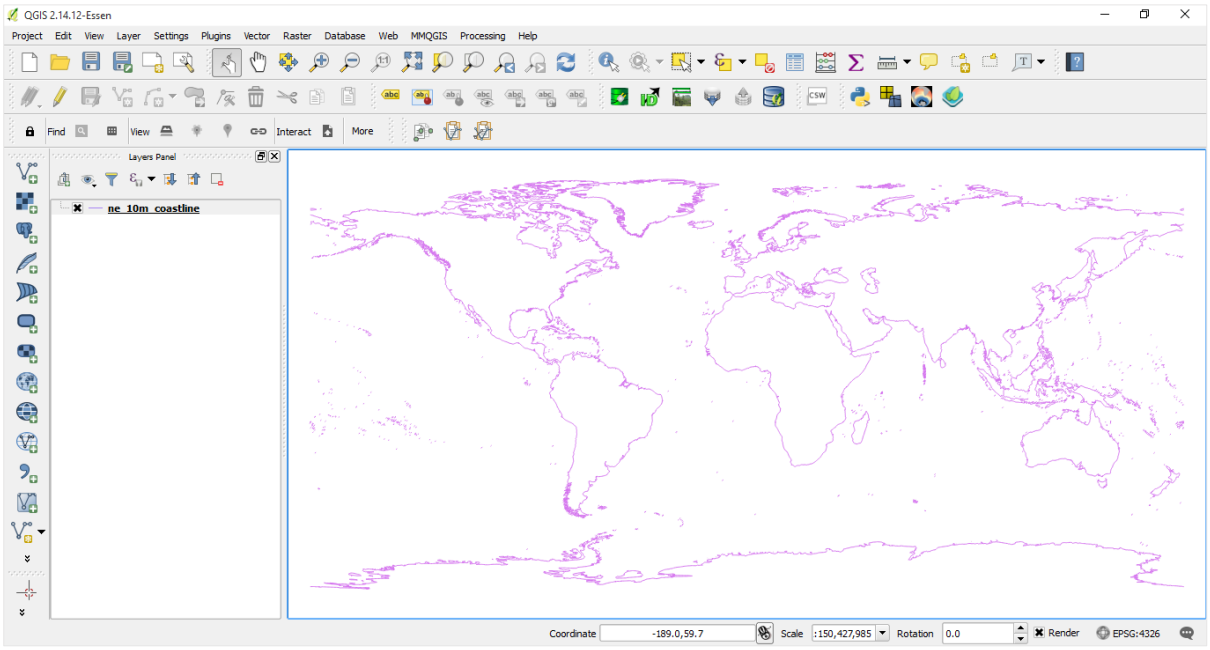
.shp	shape format; the feature geometry itself
.shx	shape index format; a positional index of the feature geometry to allow seeking forwards and backwards quickly
.dbf	attribute format; columnar attributes for each shape, in dBase IV format
.prj	projection format; the coordinate system and projection information, a plain text file describing the projection using well-known text format

These are the different shapefile formats that store geometric location and associated attribute information. Although you normally will only use the files with the .shp extension, you will always need the other three types in the same folder as well.

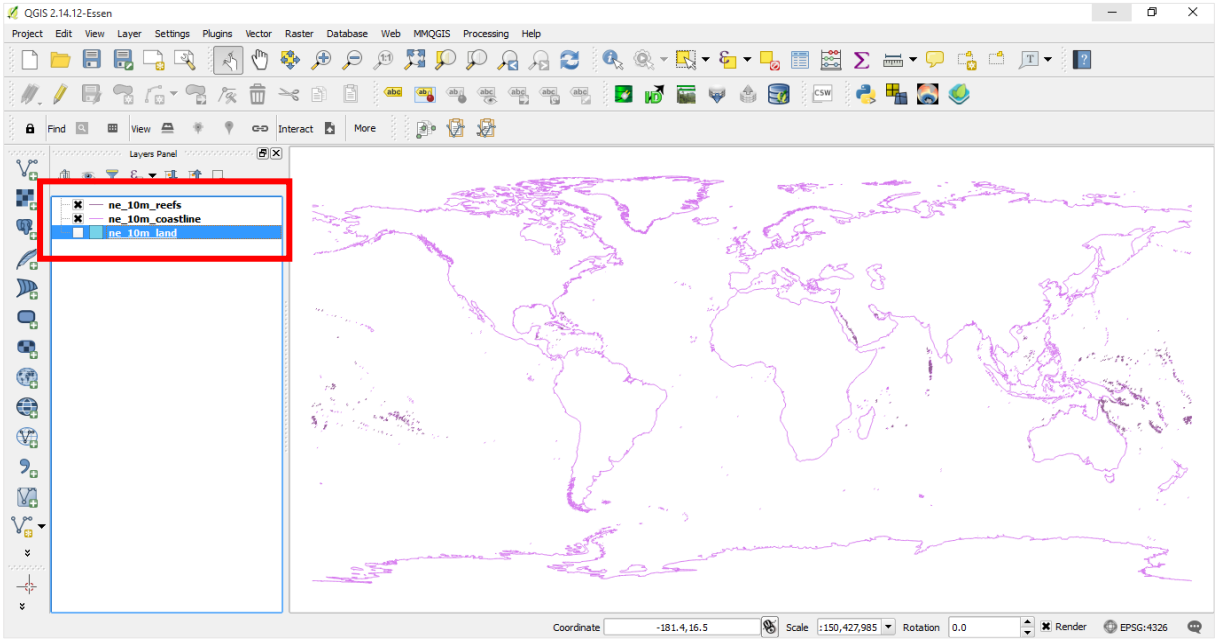
Now let's add these layers to the map. Go to the add vector layer icon and click on it. You will see a dialog box appear. Look for the folder where you have these shapefiles saved, and open the ne_10m_coastline.shp.



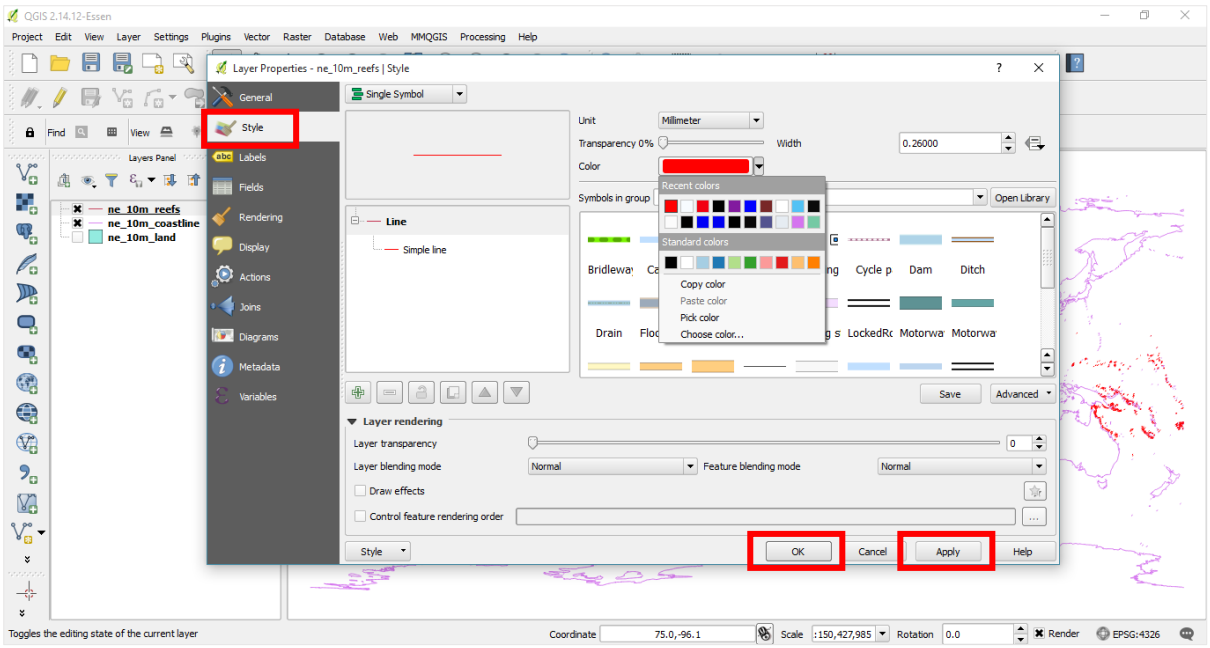
You will see the world's coastline appear.



Now let's add the land and reefs shapefiles in the same way. You will have your land and reefs appear on your map now as well.

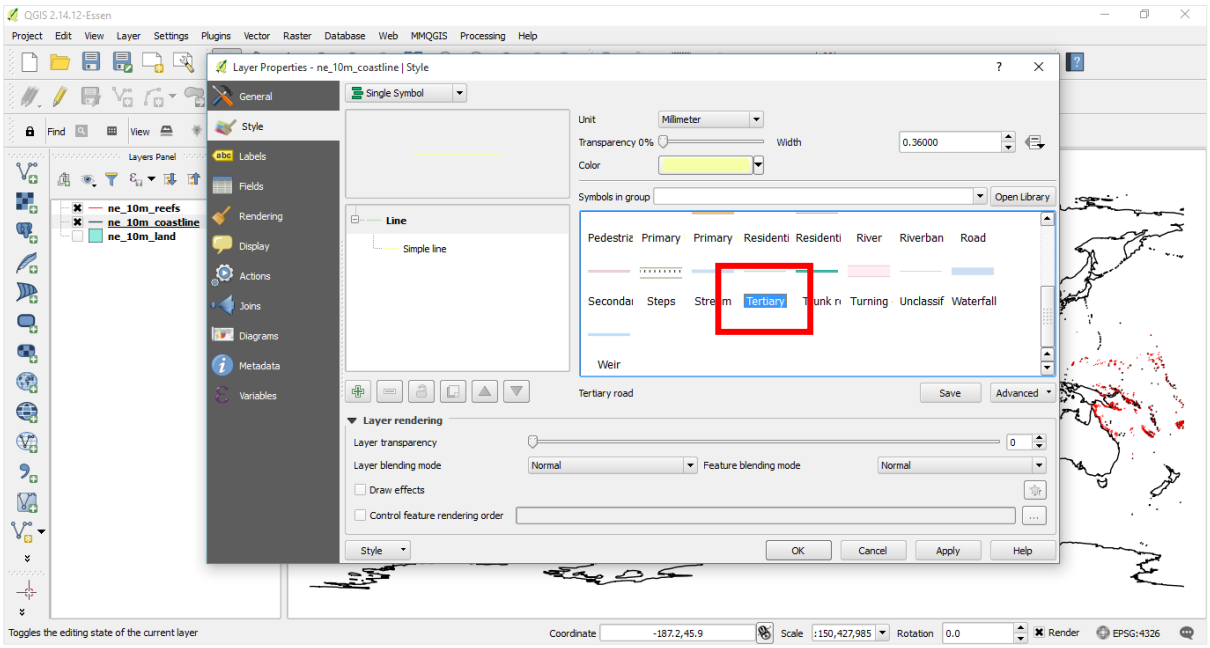


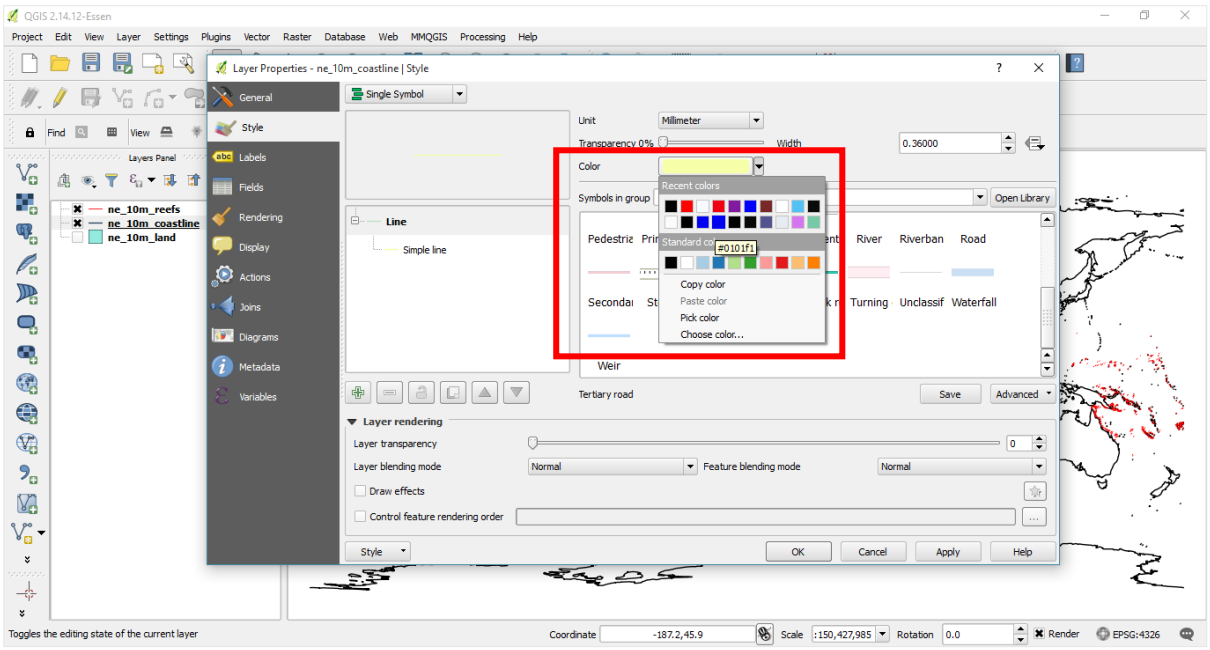
Remember, you can turn the layers on and off, and you can also changes the appearance, (for example colour) of the layers. Click on the layer name that you want to change and right-click then choose Properties. You will have a new window appearing.



Here you can look at information about this shapefile by clicking on General, or add labels in Labels, or change the Style. So let's click on Style and change the colour to red. If you click Apply and then OK, you will see your map with the coral reefs highlighted with red colour.

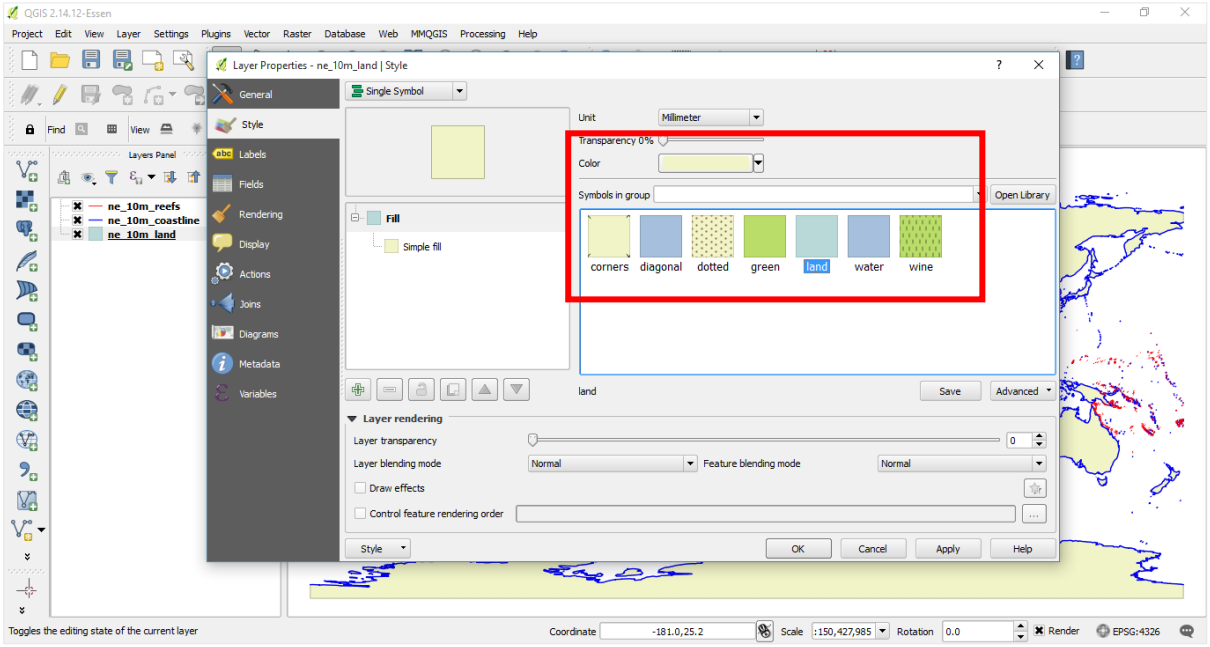
Now let's change the colour of the coastline. Click on the layer, right-click, choose Properties, Style and use Tertiary with blue colour.

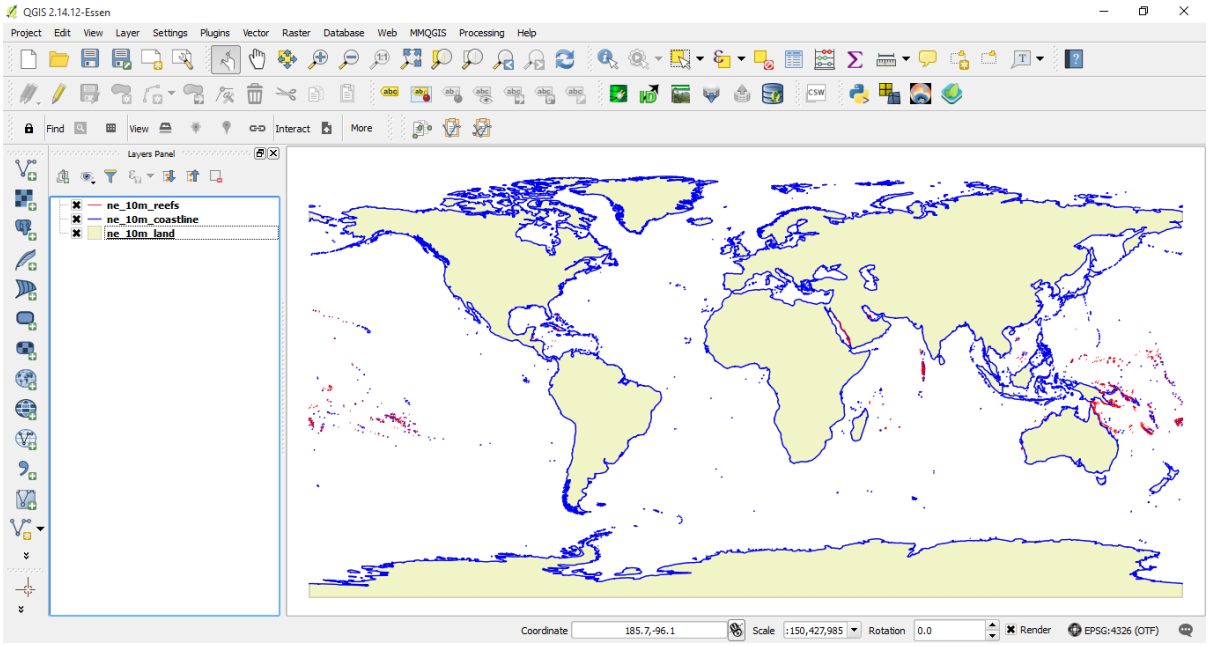




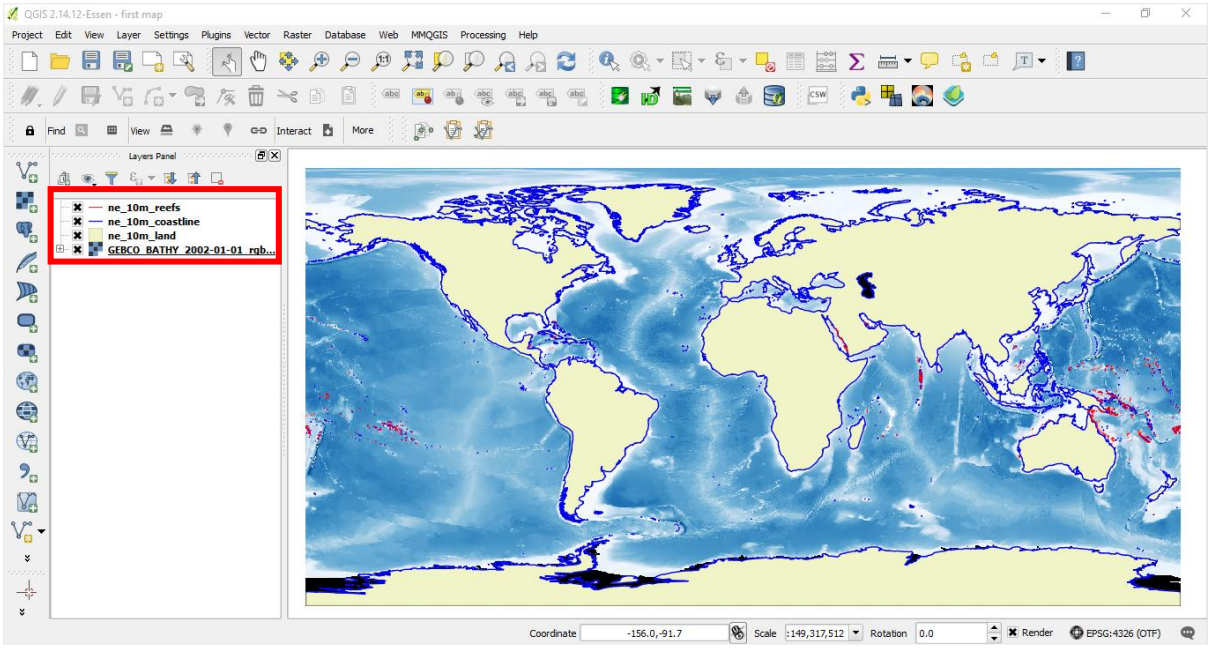
Click Apply and then OK.

Now let's change the colour of the land shapefile. Again, right-click and choose Properties → Style and you will have the land option. Click on it, then Apply and OK.





Now let's add ocean bathymetry using the raster layer option. You can download the bathymetry data from this website. As a file type choose GeoTIFF (raster) and use the image size 3600 x 1800. Now add as raster layer to the map: Layer → Add Layer → Add Raster Layer. You should have the new layer appear in the Layer Panel.



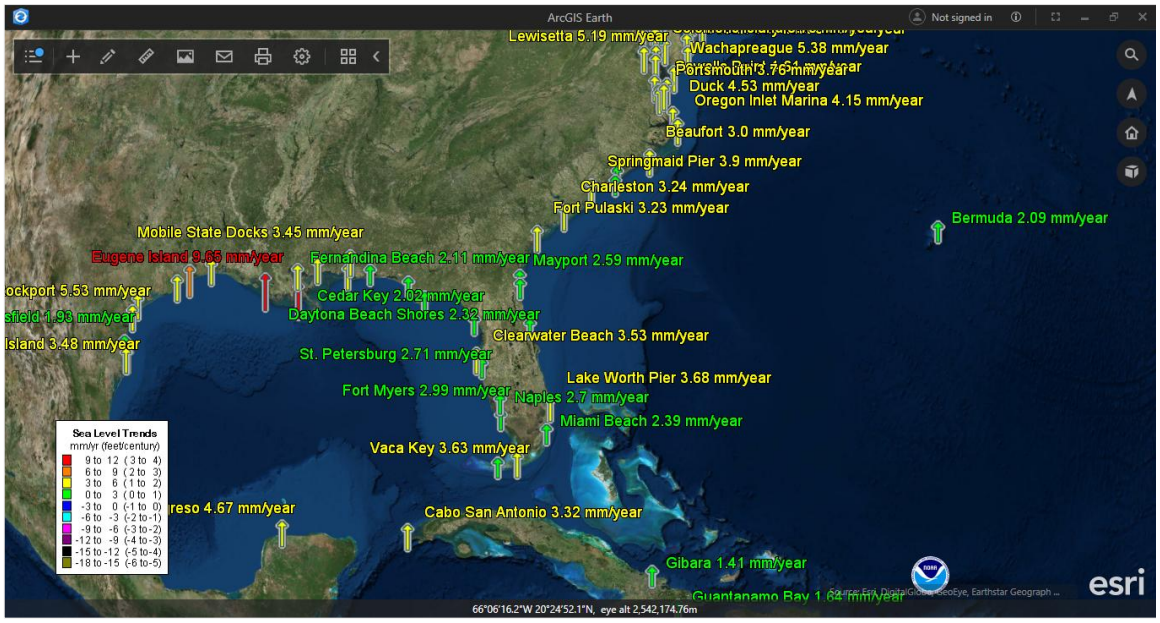
You have your first map!

III. Discussing Climate Change

Sea-Level Rise

Sea-level rise is already affecting many part of the world such as Florida, New Orleans or low-lying islands like Solomon Islands, Maldives or Kiribati. The main driver of sea-level rise is the warming sea temperatures and the subsequent thermal expansion. Measurements taken since the 19th century show that the average rise was 1.7mm/year since the 1950s, and since the early 1990s the average rise has increased to 3.3mm/year (Nicholls *et al*. 2010).

Sea-level rise, however, doesn't uniformly affect all coastlines. Since the mass of the earth is not uniformly distributed, the Earth's gravity field will vary, thus the rate of sea-level rise will vary by region. If you want to find out more, this sea level trends website has data available, showing which areas are affected and at what rate. You can open the coops-slt.kmz file using either GoogleEarth or ArcGIS Earth and look at each data point.

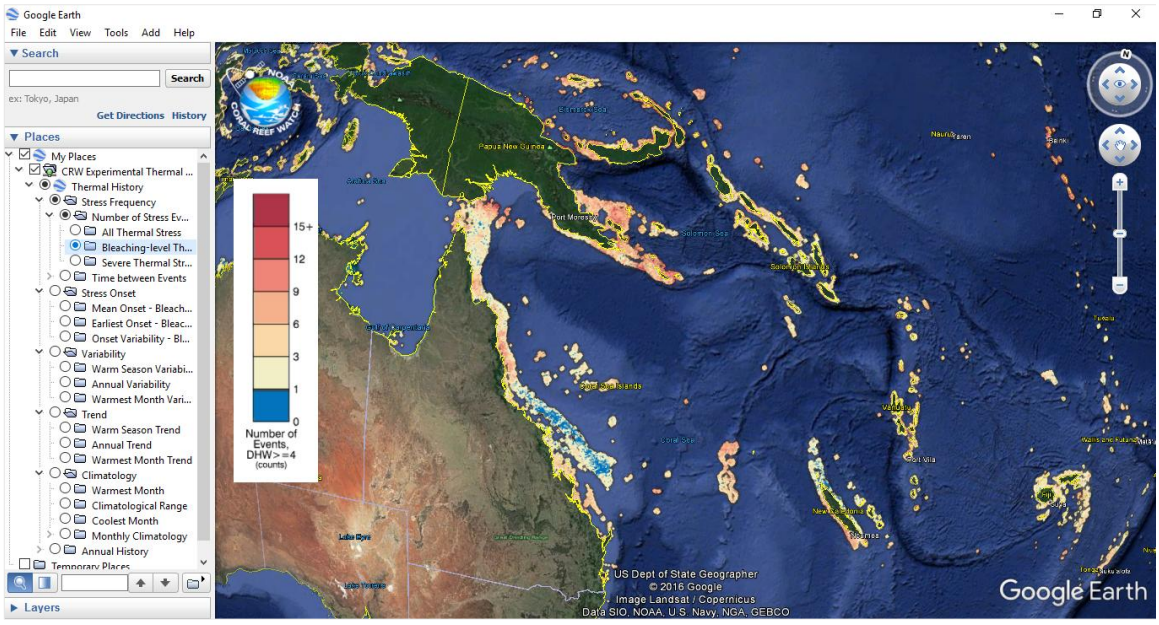


Coral Reefs

Coral reefs are important marine ecosystems both biologically and economically providing home for at least 25% of all marine species. Shallow-water reefs form in a zone between 30° N to 30° S of the equator, in an optimum temperature range of 26–27 °C.

Global climate change poses various threats to corals. Increasing carbon-dioxide levels are altering ocean chemistry causing acidification. Rising sea temperatures are driving an increased frequency of coral bleaching and mortality. It is estimated that 30% of the coral reefs are already severely damaged, and close to 60% may be lost by 2030 (Hughes *et al*. 2003).

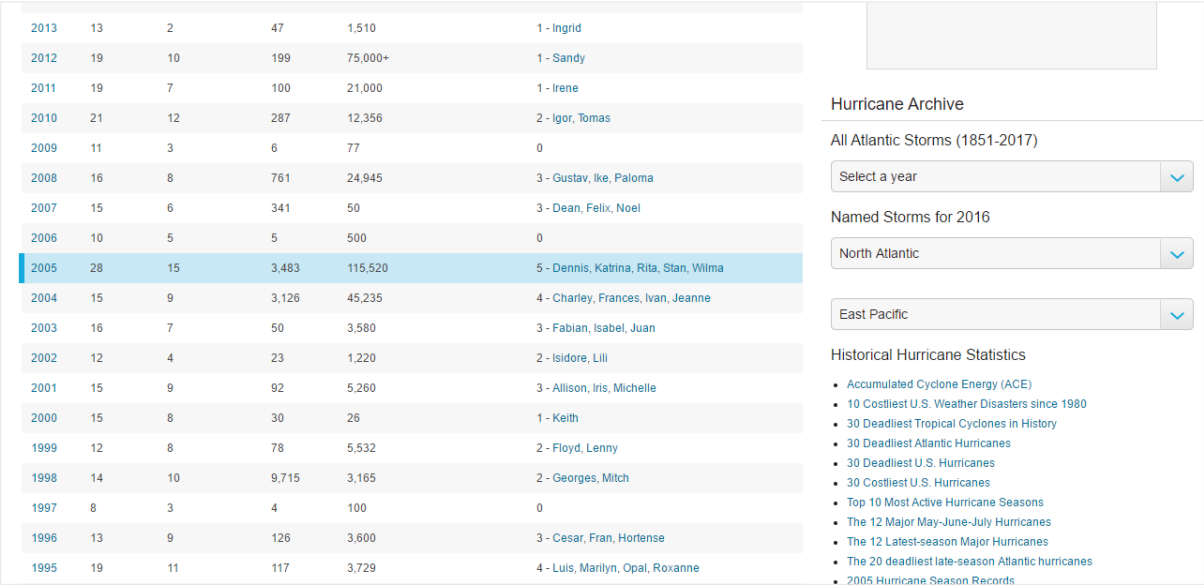
On NOAA's Coral Reef Watch website data is available on coral reef thermal history, and can be viewed on GoogleEarth.



Mapping with R Studio and QGIS

In the following two chapters we will explore R Studio and QGIS further, and for the exercises we will use the tracks of the 5 major hurricanes that occurred in the North Atlantic basin in 2005, and sea surface temperature.

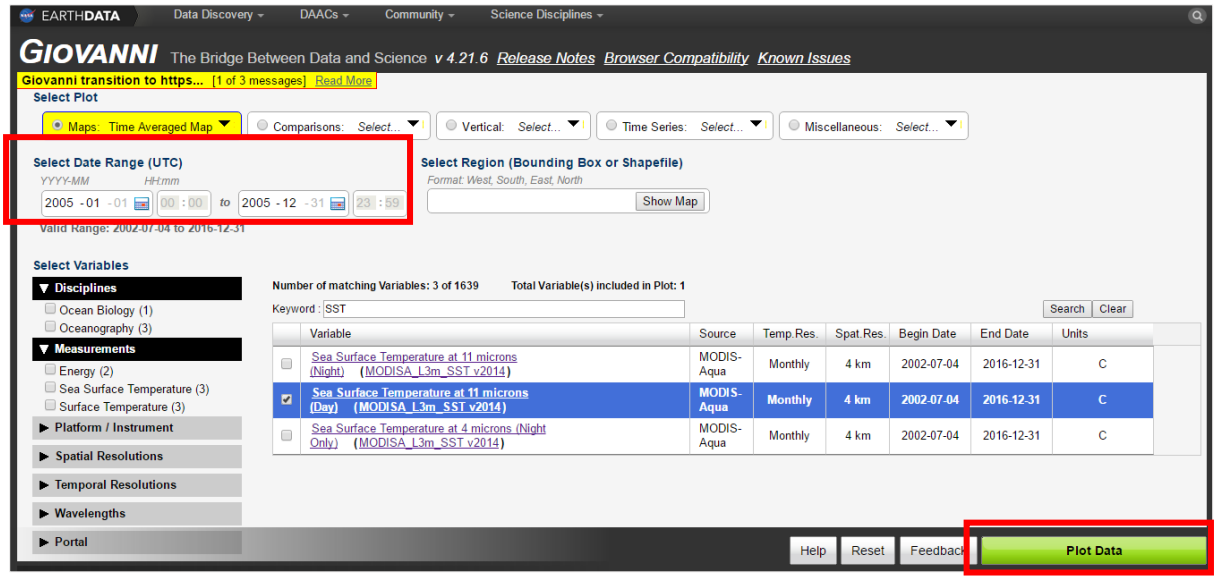
Data on hurricanes is available on this website (also in the resource pack). Look for 2005 and you will see the names of the five major ones (Retired names).

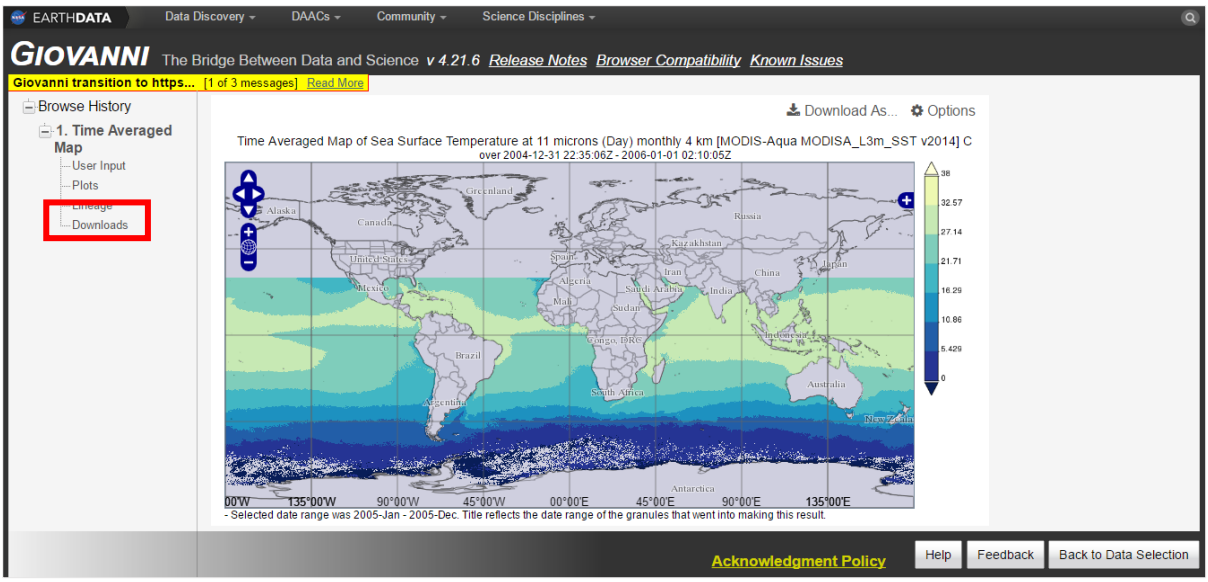


2013	13	2	47	1,510	1 - Ingrid
2012	19	10	199	75,000+	1 - Sandy
2011	19	7	100	21,000	1 - Irene
2010	21	12	287	12,356	2 - Igor, Tomas
2009	11	3	6	77	0
2008	16	8	761	24,945	3 - Gustav, Ike, Paloma
2007	15	6	341	50	3 - Dean, Felix, Noel
2006	10	5	5	500	0
2005	28	15	3,483	115,520	5 - Dennis, Katrina, Rita, Stan, Wilma
2004	15	9	3,126	45,235	4 - Charley, Frances, Ivan, Jeanne
2003	16	7	50	3,580	3 - Fabian, Isabel, Juan
2002	12	4	23	1,220	2 - Isidore, Lili
2001	15	9	92	5,260	3 - Allison, Iris, Michelle
2000	15	8	30	26	1 - Keith
1999	12	8	78	5,532	2 - Floyd, Lenny
1998	14	10	9,715	3,165	2 - Georges, Mitch
1997	8	3	4	100	0
1996	13	9	126	3,600	3 - Cesar, Fran, Hortense
1995	19	11	117	3,729	4 - Luis, Marilyn, Opal, Roxanne

We will use all five of them, so copy and paste them separately into Excel sheets and save them as separate .csv files. (You will also find them in your resource pack).

To access sea surface temperature data, go to this website. Make sure that the date range is correct and the keyword is 'SST'. Tick 'Sea Surface Temperature at 11 microns (Day)', then click 'Plot Data'.





Click Download and you will have four file options (.nc, .png, .geotiff and .kmz). Download them all. Make sure that the five hurricane .csv data and the SST data is saved and accessible easily. (The data is also available in the resoure pack).

1. R Studio

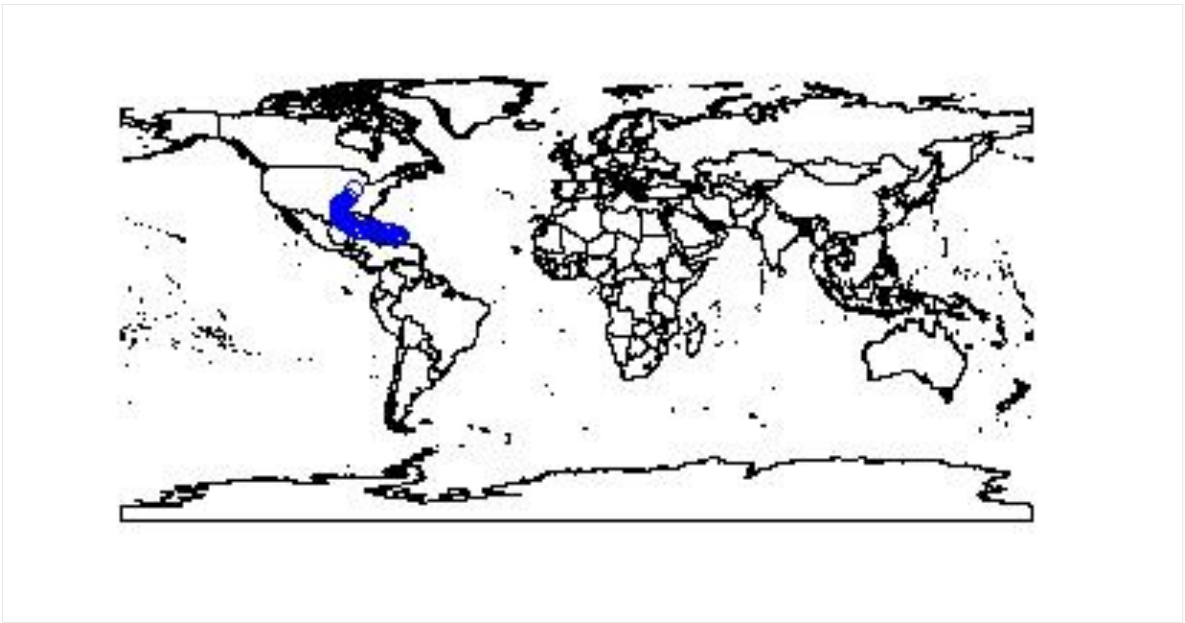
Before we explore the various R packages, let's open R Studio and open a new R script. First, let's look at the track and direction of Hurricane Dennis.

```
setwd("~/Course files/5. Mapping with R Studio and QGIS/Hurricanes")

d <- read.csv(file = "dennis.csv")
require(maps)
map("world", ylim = c(10, 60), xlim = c(-120, -40))

Lo <- d$Long
La <- d$Lat
n <- length(Lo)
lines (Lo, La, lwd = 2.5, col = "blue")
arrows(Lo[n - 1], La[n - 1], Lo [n], La[n], lwd = 2.5, lentgh = .1, col =
"blue")
```

rworldmap

First we will use the packages ‘rworldmap’ and ‘rworldxtra’.

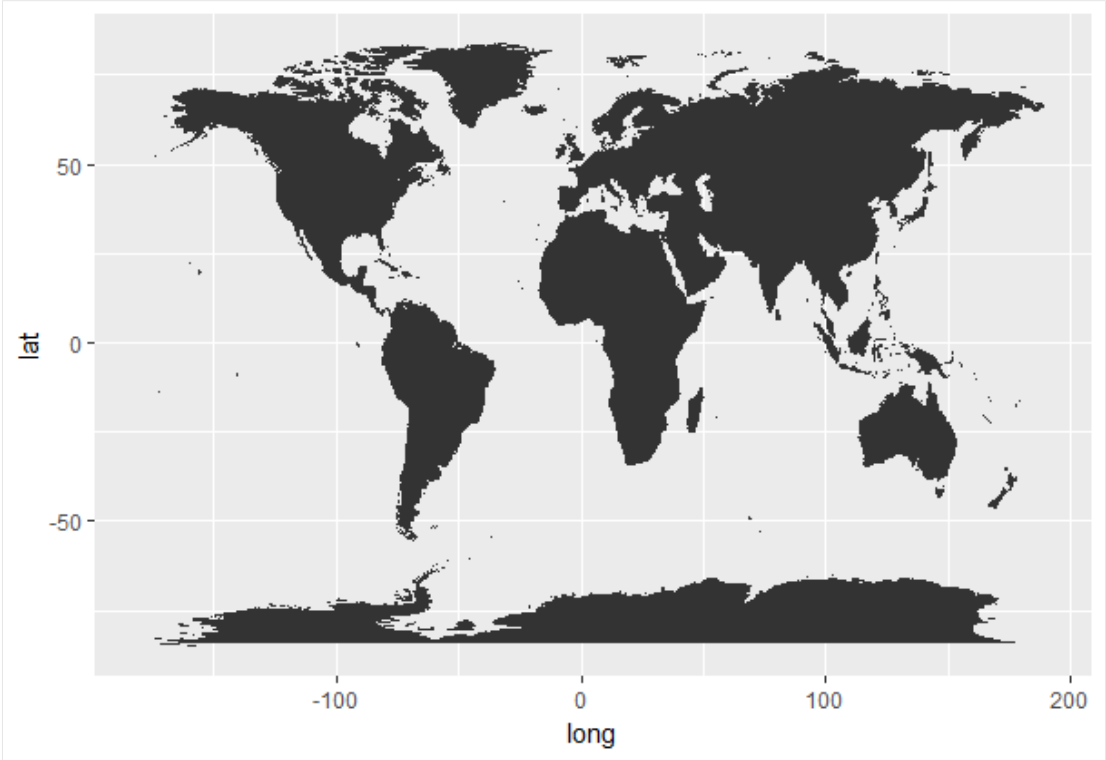
```
setwd("C:/~")
library(rworldmap)
library(rworldxtra)
r <- read.csv(file = "rita.csv")
newmap <- getMap(resolution = "high") #you can also change it to low res
plot(newmap)
points(r$Long, r$Lat, col = "blue")
```



ggmap

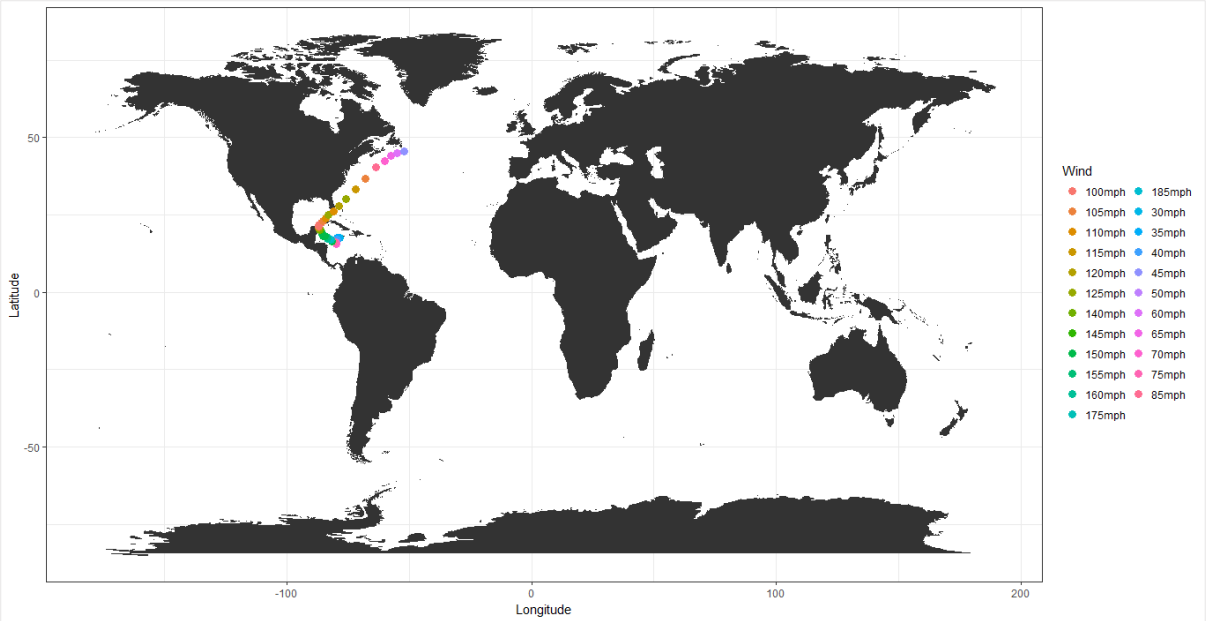
Now let's look at ggmap and ggplot packages. First, we will make a world map:

```
library(ggmap)
map <- map_data("world")
ggplot() + geom_polygon(data = map, aes(x = long, y = lat, group = group))
```



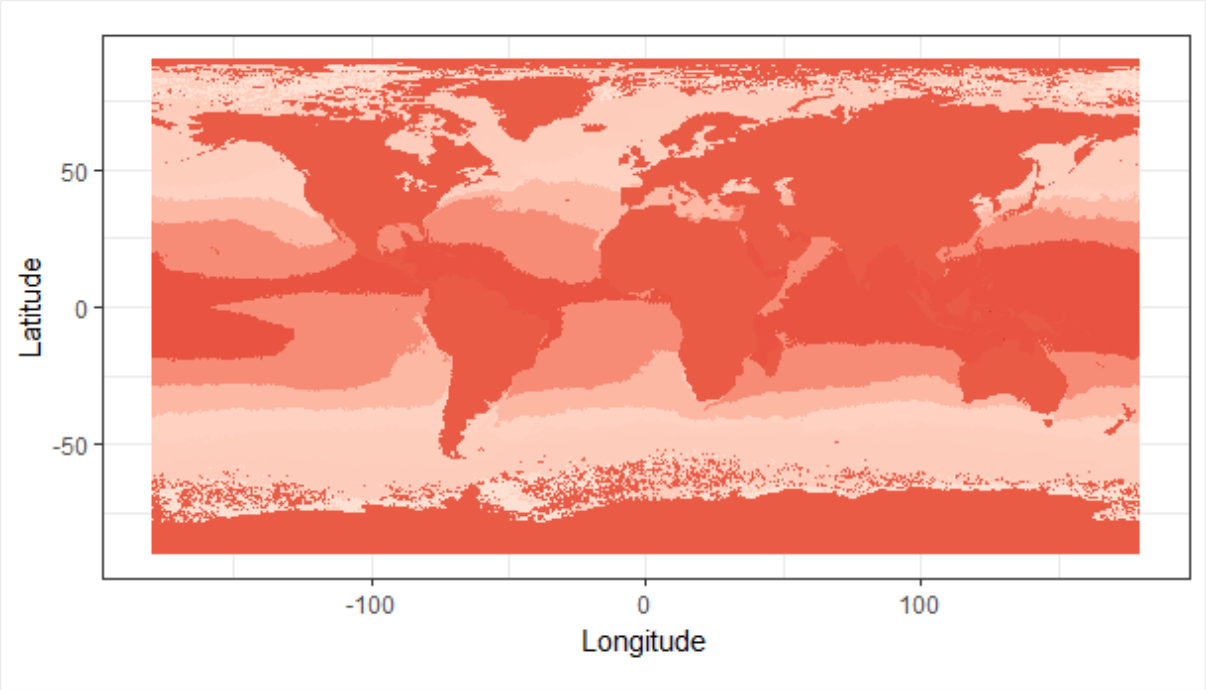
Then add the points of Hurricane Wilma, name the axis accordingly, remove the grey background and add legend showing the wind force of each points:

```
w <- read.csv(file = "wilma.csv")
ggplot() +
  geom_polygon(data = map, aes(x = long, y = lat, group = group)) +
  geom_point(data = w, aes(Long, Lat, colour = Wind), size = 3) +
  theme_bw() +
  ylab("Latitude") +
  xlab("Longitude")
```



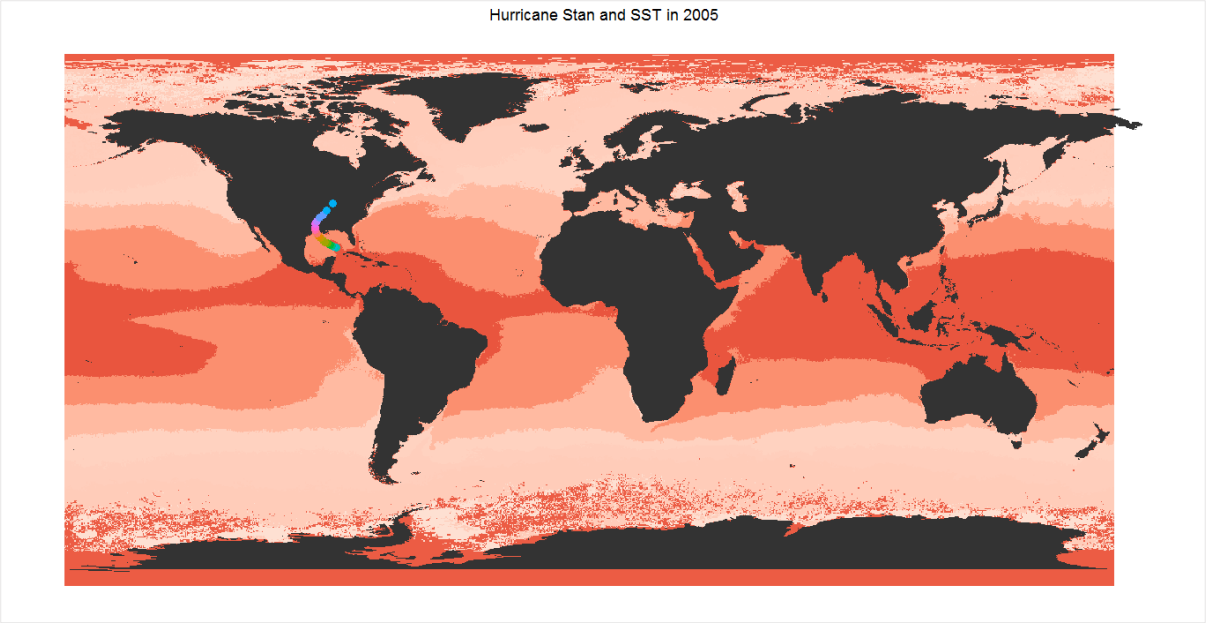
We will now add the sea surface temperature as a raster. After adding the raster, it needs to be converted to spatial pixel dataframe. (You can use this website to choose the fill gradient – colour the will appear to indicate sea surface temperature.)

```
library(raster)
library(rasterVis)
library(rgdal)

sst <- raster("sst.tif")
sst.spdf <- as(sst, "SpatialPixelsDataFrame")
sst.df <- as.data.frame(sst.spdf)

ggplot(sst.df, aes(x = x, y = y)) +
  geom_tile(aes(fill = sst)) +
  scale_fill_gradient(low = "#fee0d2", high = "#de2d26") +
  coord_equal() +
  ylab("Latitude") +
  xlab("Longitude") +
  theme_bw() +
  theme(legend.position = "none")
```



Now add Hurricane Stan to this map:

```
s <- read.csv(file = "stan.csv")
ggplot() +
  geom_tile(data = sst.df, aes(x = x, y = y, fill = sst)) +
  scale_fill_gradientn(colours = c("#fee0d2","#fc9272","#de2d26")) +
  geom_polygon(data = map, aes(x = long, y = lat, group = group)) +
  geom_point(data = s, aes(Long, Lat, colour = Wind), size = 3) +
  ggtitle("Hurricane Stan and SST in 2005") +
  ylab("Latitude") +
  xlab("Longitude") +
  theme_bw() +
  theme(
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        plot.title = element_text(hjust = 0.5),
        legend.position = "none")
```



googleVis

This is the same package that we used before for the land and sea surface temperature plot. We will use it to make an interactive map.

```
library(googleVis)

setwd("~/Course files/5. Mapping with R Studio and QGIS/Katrina")

k <- read.csv(file = "katrina.csv")

KatrinaGeoMap <- gvisGeoMap(k, locationvar = 'LatLong', numvar =
'Pressure',
                            hovervar = 'Type',
                            options = list(width = 800, height = 400,
                                           region = 'US', dataMode =
'Markers'))
plot(KatrinaGeoMap)

KatrinaMap <- gvisMap(k, 'LatLong' ,'Pressure',
                      options = list(showTip = TRUE, showLine = TRUE,
                                     enableScrollWheel = TRUE,
                                     mapType = 'hybrid', useMapTypeControl
= TRUE,
                                     width = 800, height = 400))
plot(KatrinaMap)

KatrinaVis <- gvisMerge(KatrinaGeoMap, KatrinaMap)
plot(KatrinaVis)
```

3D Globe

For the 3D Globe, first we will create a static globe. We will use the Hurricane Katrina data and convert it into spatial points data frame:

```
k_df <- data.frame(KLong = k$Long, KLat = k$Lat)
k_sp <- SpatialPoints(k_df, proj4string = CRS("+proj=longlat"))
k_spdf <- SpatialPointsDataFrame(k_sp, data = k_df)
```

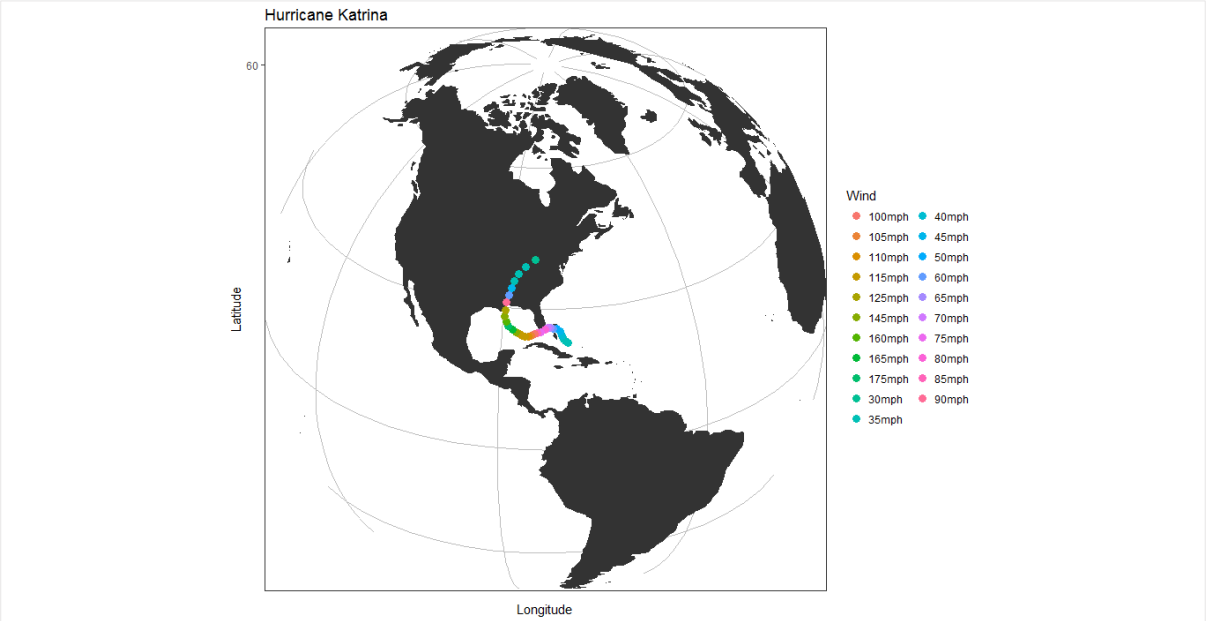
Then use an inbuilt world data:

```
worldMap <- getMap()
world.points <- fortify(worldMap)
world.points$region <- world.points$id
world.df <- world.points[,c("long","lat","group", "region")]
```

And create a map using the orthographic projection:

```
worldmap <- ggplot() +
  geom_polygon(data = world.df, aes(x = long, y = lat, group = group)) +
  geom_point(aes(x = k$Long, y = k$Lat), color = "blue", size = 1) +
  scale_y_continuous(breaks = (-2:2) * 30) +
  scale_x_continuous(breaks = (-4:4) * 45) +
  theme_bw() +
  theme(axis.line = element_line(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank())
```

```
worldmap +
  coord_map("ortho", orientation = c(30, -80, 0))+
  ylab("Latitude") +
  xlab("Longitude") +
  ggtitle("Hurricane Katrina") +
  theme_bw() +
  theme(panel.grid.major = element_line(color = "grey"),
        panel.grid.minor = element_line(color = "grey"))
```

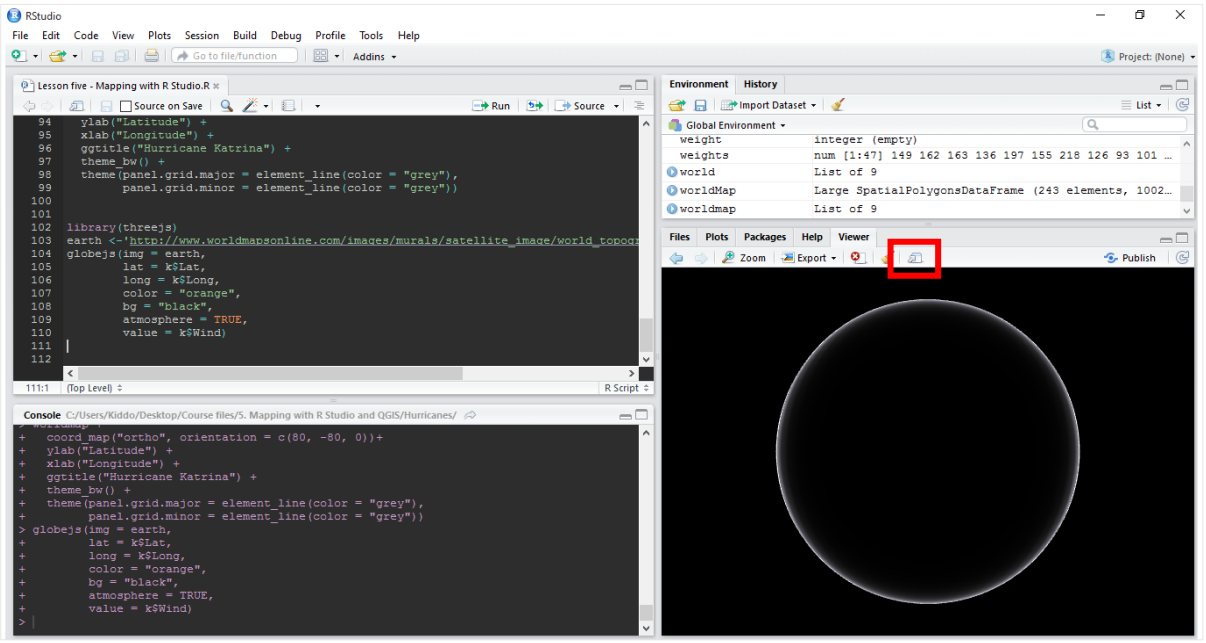


The orientation helps you set the centre view: `orientation = c(30, -80, 0)`. The two numbers are the longitude and latitude, respectively, and the third is the tilt.

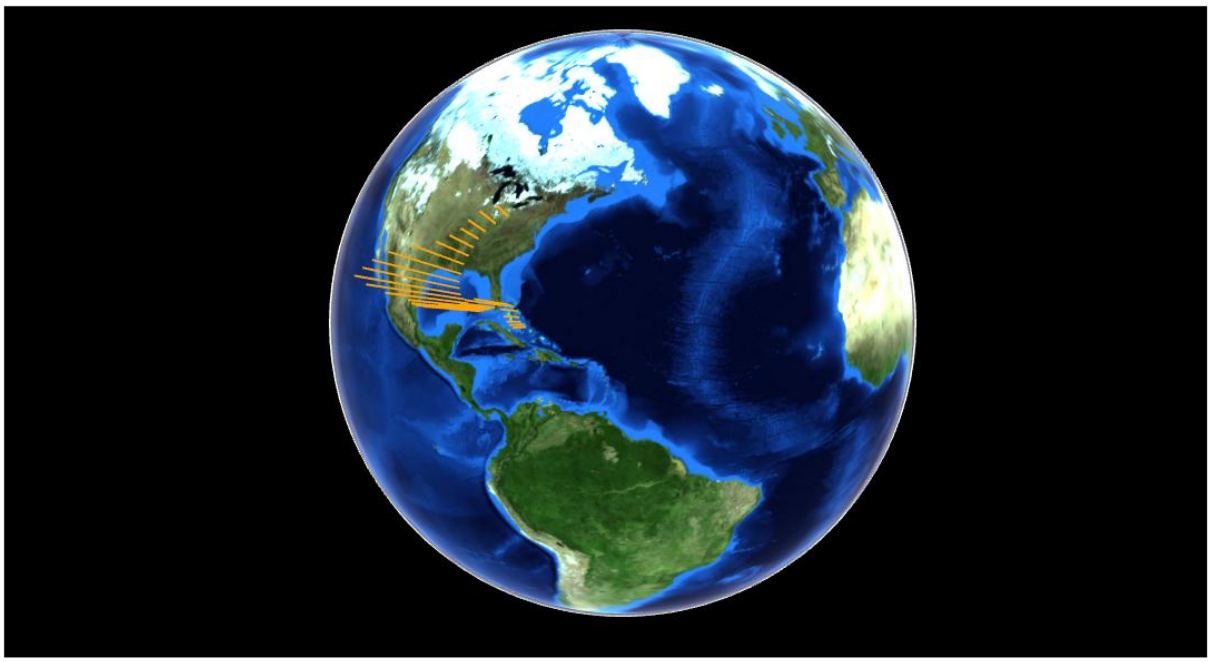
Now we will use the package called 'threejs' to create a 3D globe that you can rotate. (For the background topography and bathymetry image we are using an external .jpg file.)

```
#devtools::install_github("bwlewis/rthreejs")
library(threejs)
earth <-
"http://eoimages.gsfc.nasa.gov/images/imagerecords/73000/73909/world.topo.bathy.200412.3x5400x2700.jpg"
globejs(img = earth,
        lat = k$Lat,
        long = k$Long,
        color = "orange",
        bg = "black",
        atmosphere = TRUE,
        value = k$Wind)
```

It is possible that your globe will not appear straight away. In this case click on the icon 'Show in a new window'.



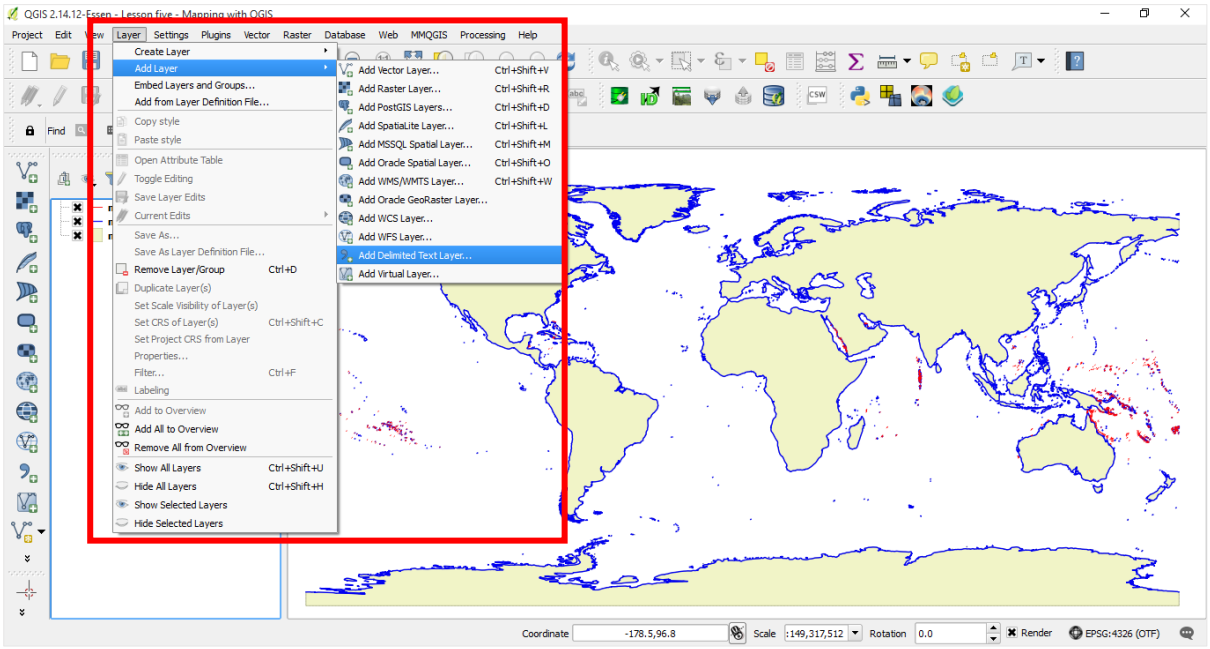
This will open up your default browser showing the Earth with the hurricane tracks. The height of each point track represents the wind force.



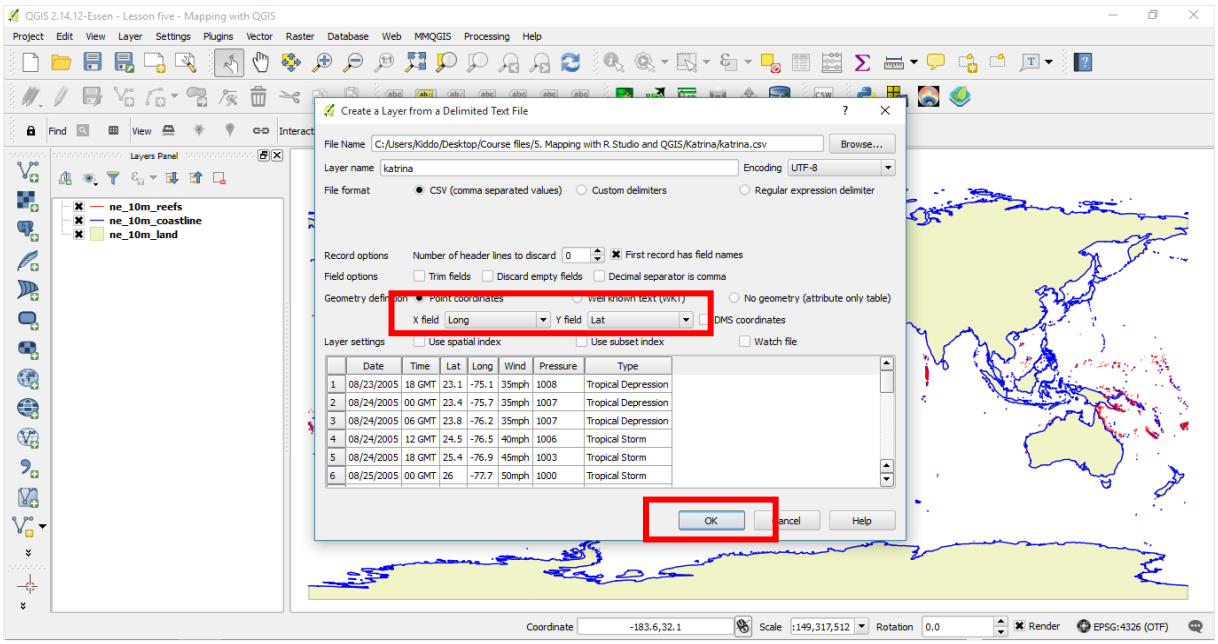
2. QGIS

We now will look into creating a map showing Hurricane Katrina and the sea surface temperature using QGIS.

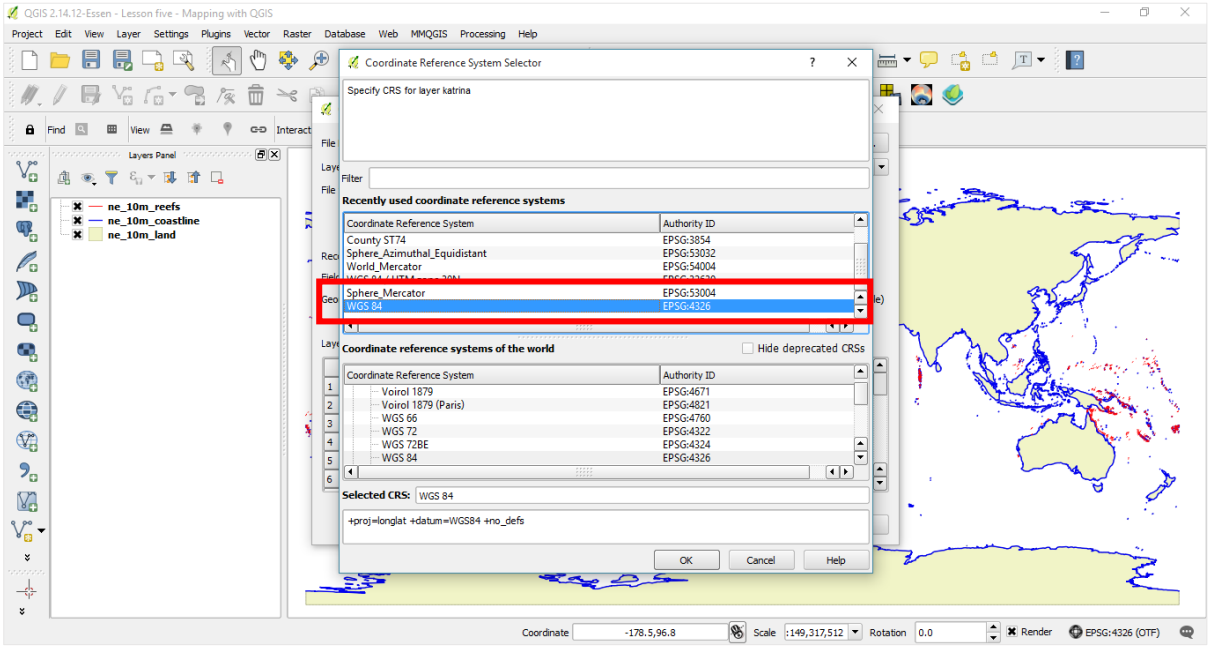
Let's open the project used in chapter 4 (QGIS), remove the bathymetry data and let's add the Hurricane Katrina tracks. Go to Layer → Add Layer → Add Delimited Text Layer:



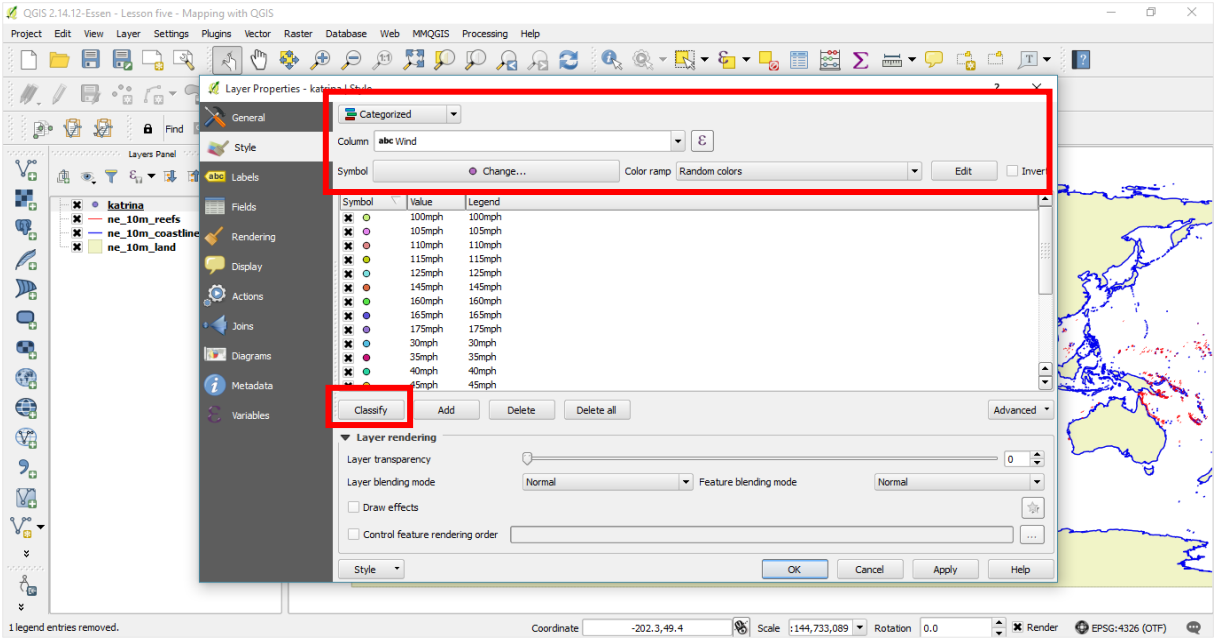
Choose the latitude and longitude and press OK:



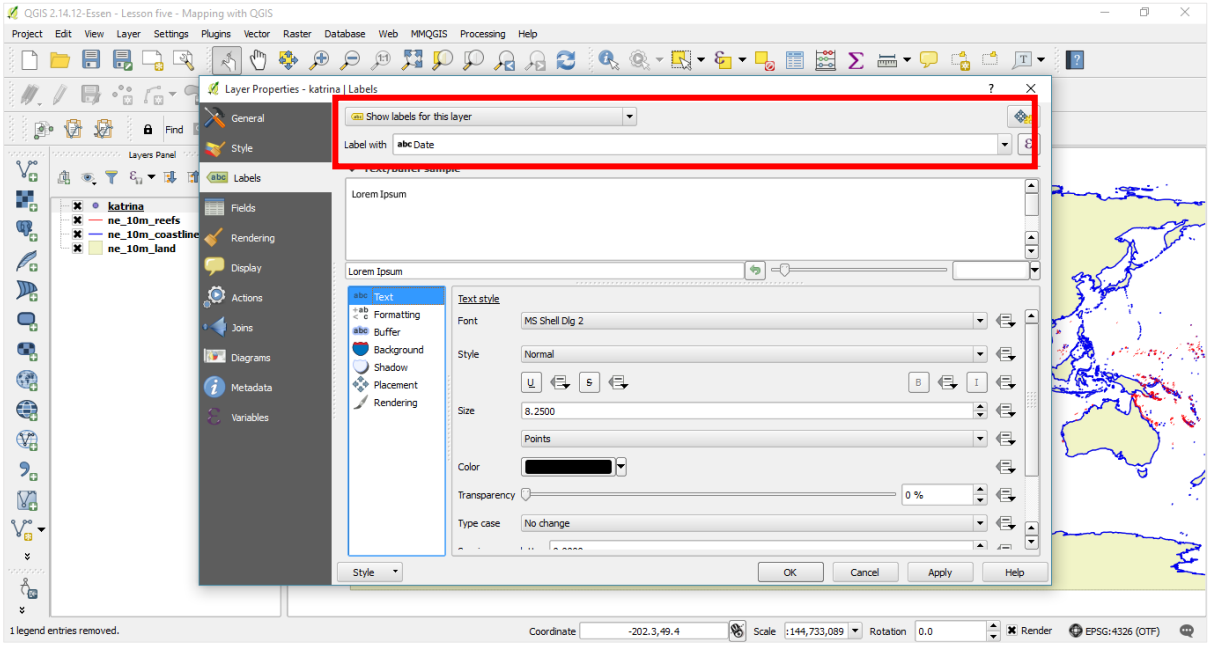
Choose WGS 84 as CFS and click OK:



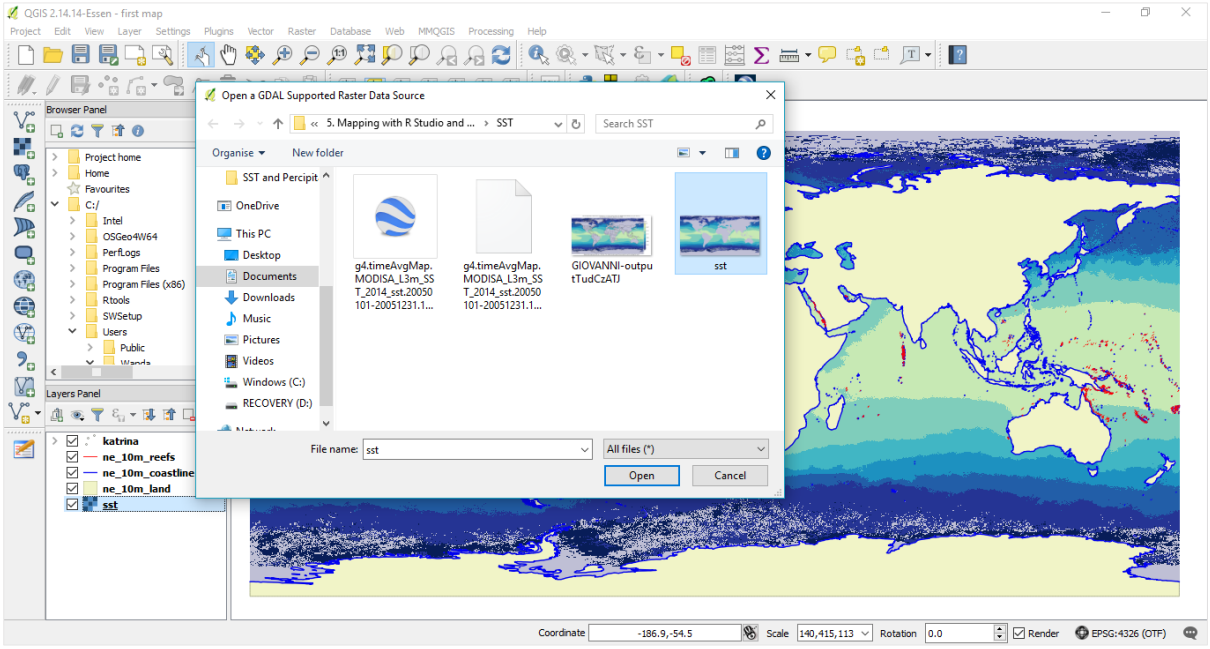
Then right click on the katrina layer, then choose Properties. Choose Categorized as style with Random colours to display Wind, and click Classify:



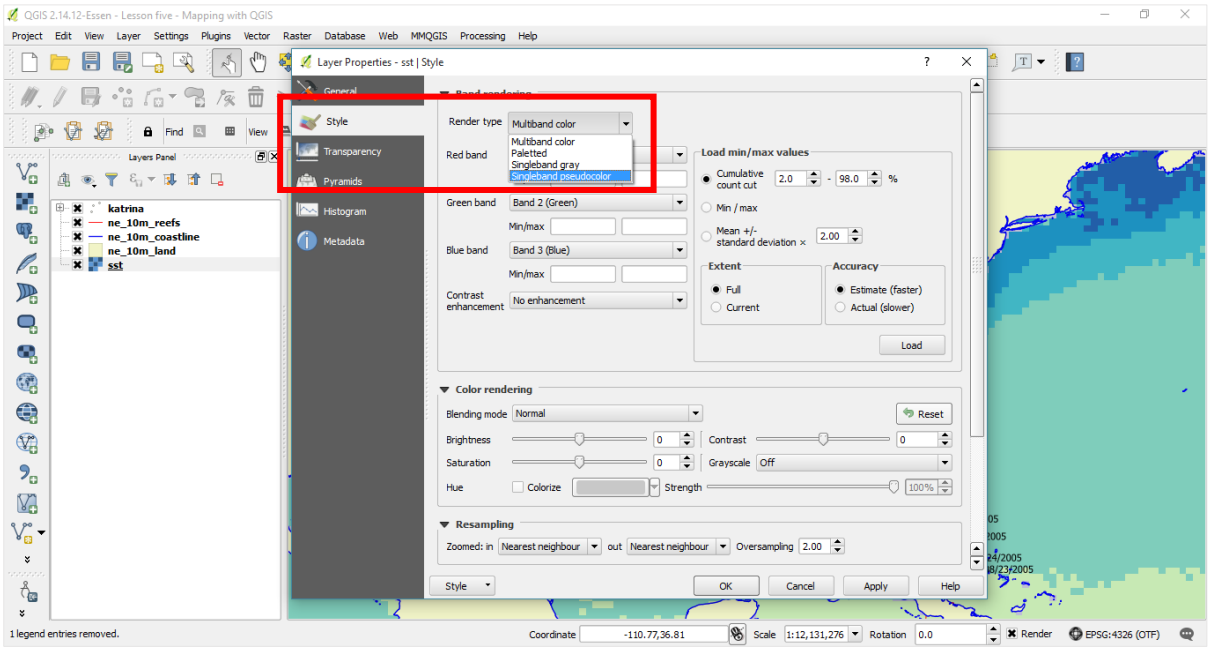
Now from the Labels choose Show labels for this layer and Label with Date, then click OK:



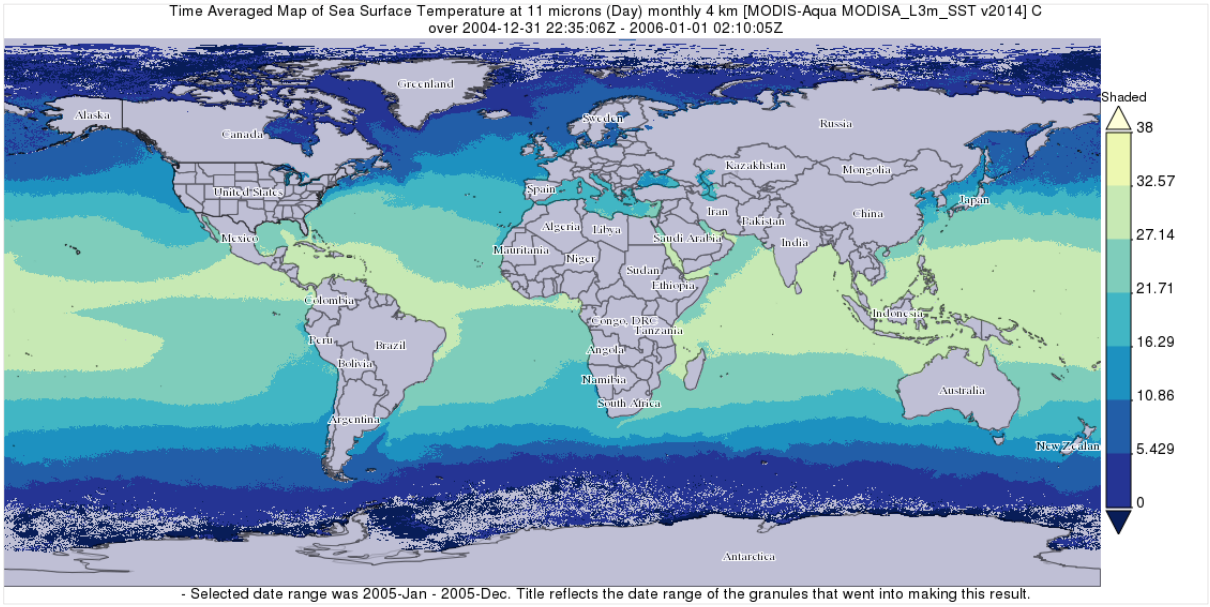
Now add the sst raster layer:



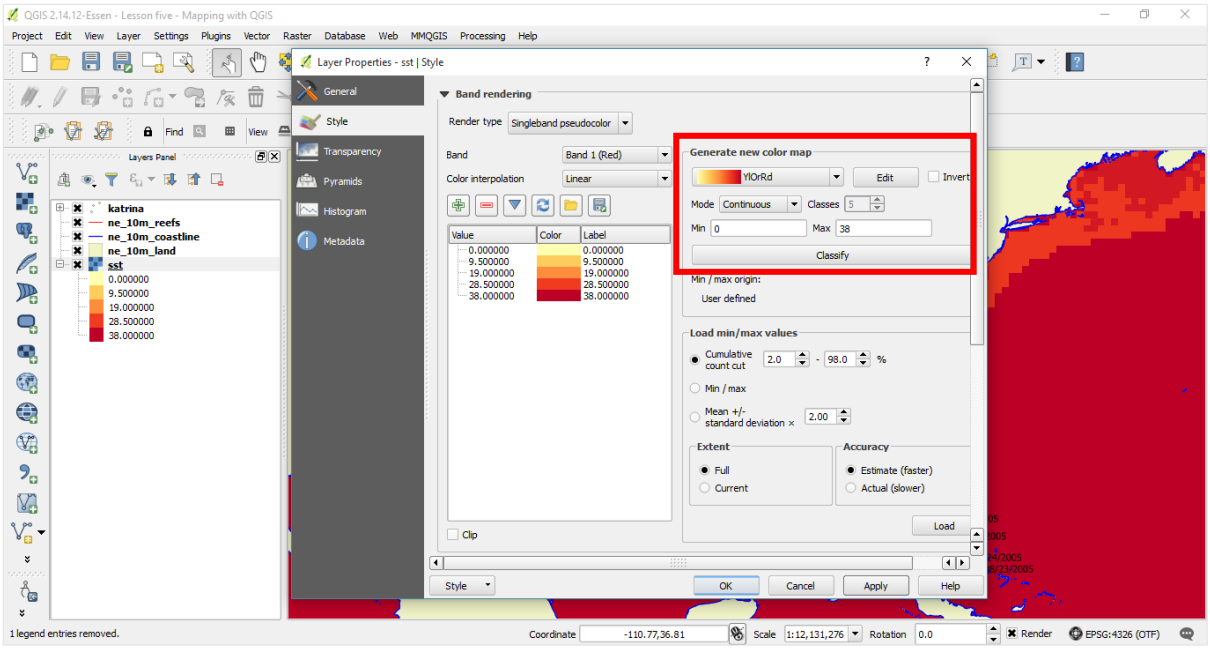
For the colours, right click on the layer → Style → Render → Type and choose Singleband pseudocolor:



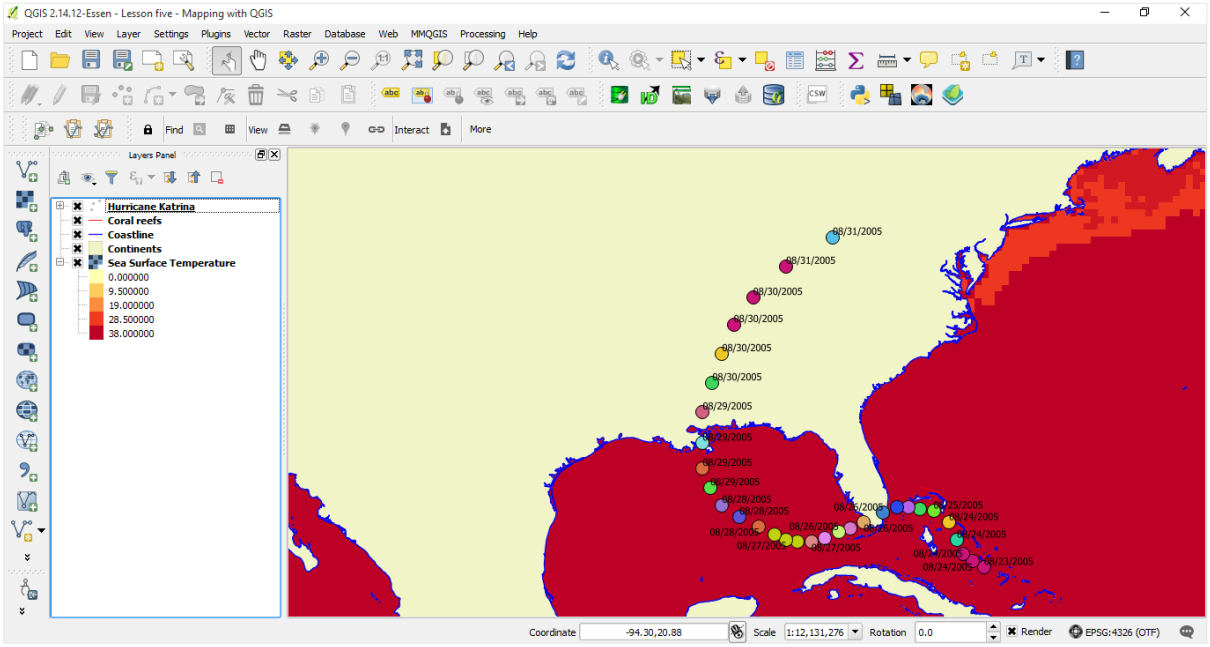
Using the downloaded .png file, set the temperature range:



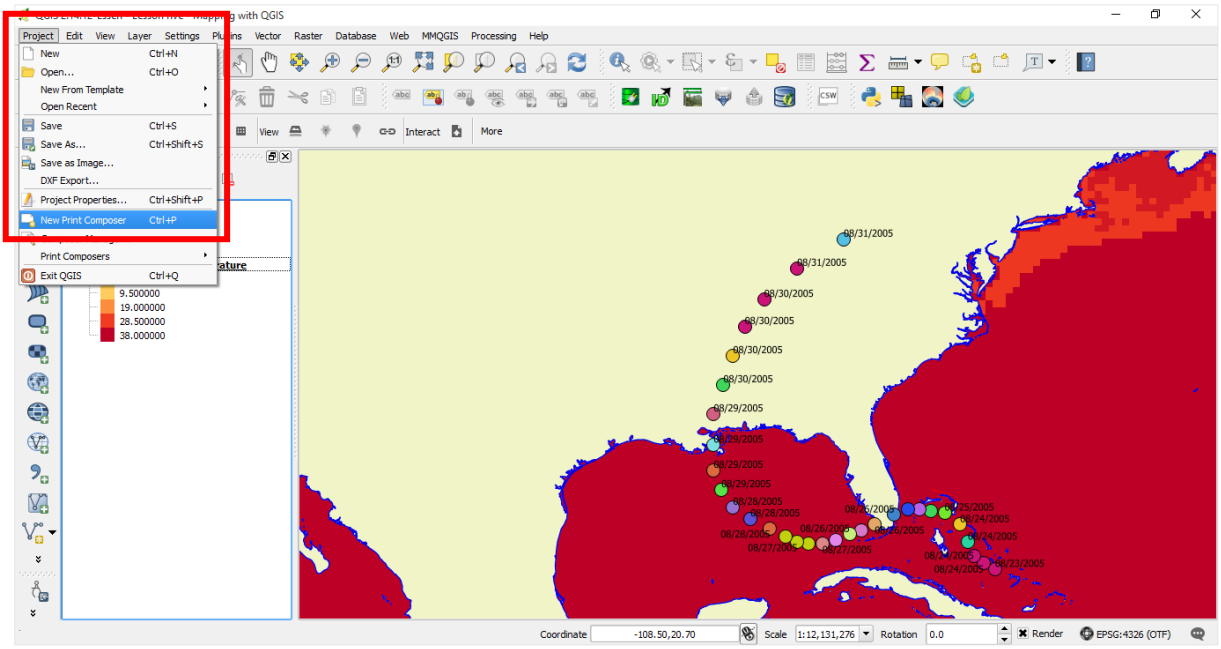
Add 0 to min and 38 to max and click Classify:



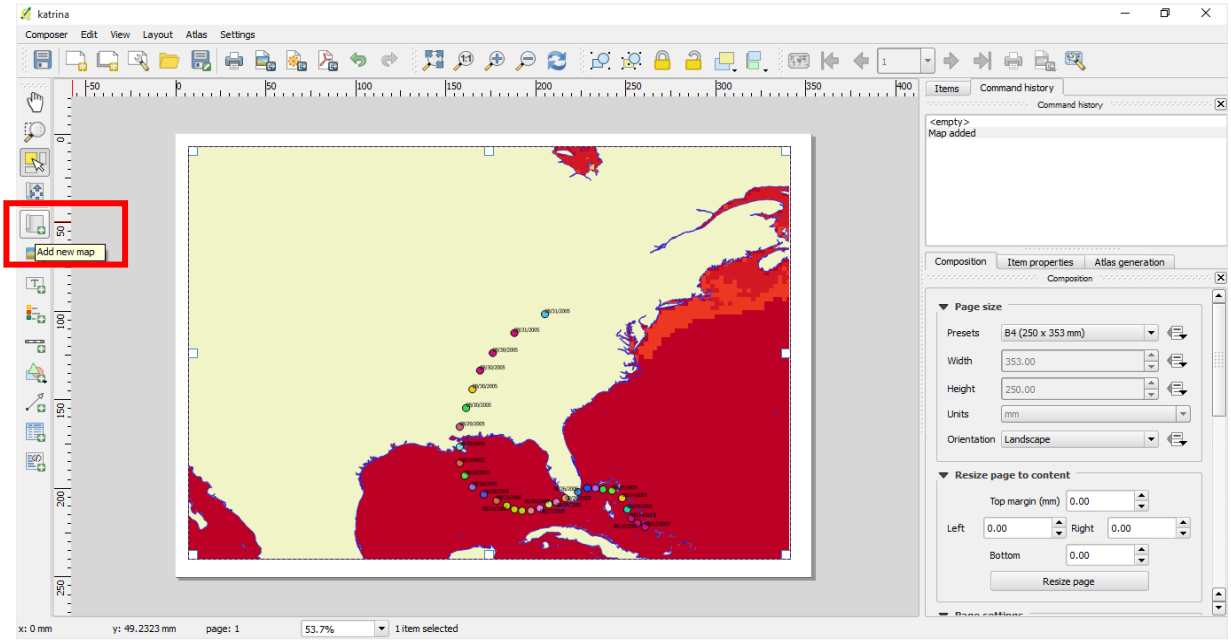
Now you have your map. You can change the layer names by right click → Rename.



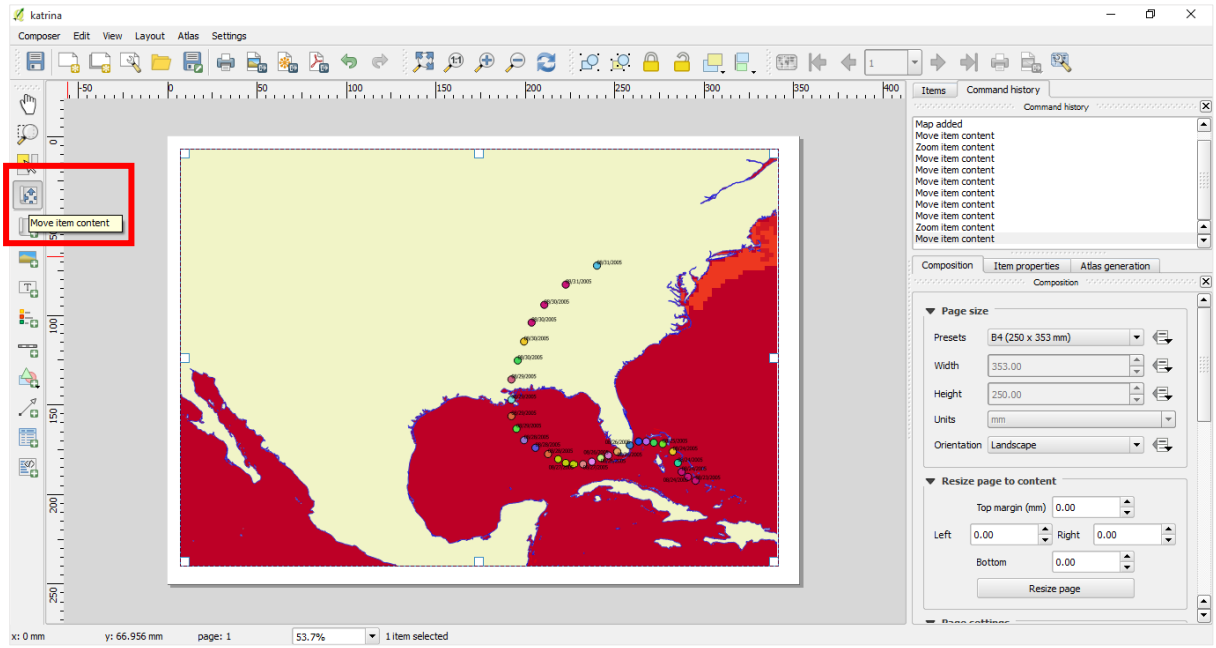
Now customise the map so you can save it as jpeg or .png file. Click Project → New Print Composer:



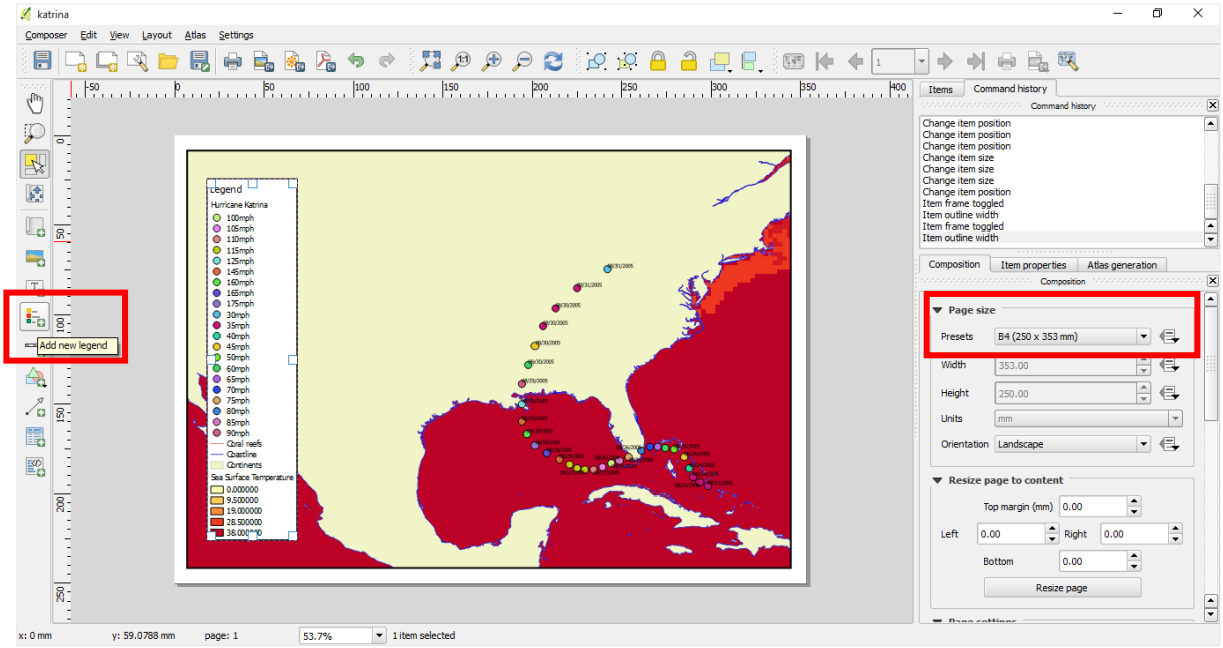
Now let's add the map using Add new map:



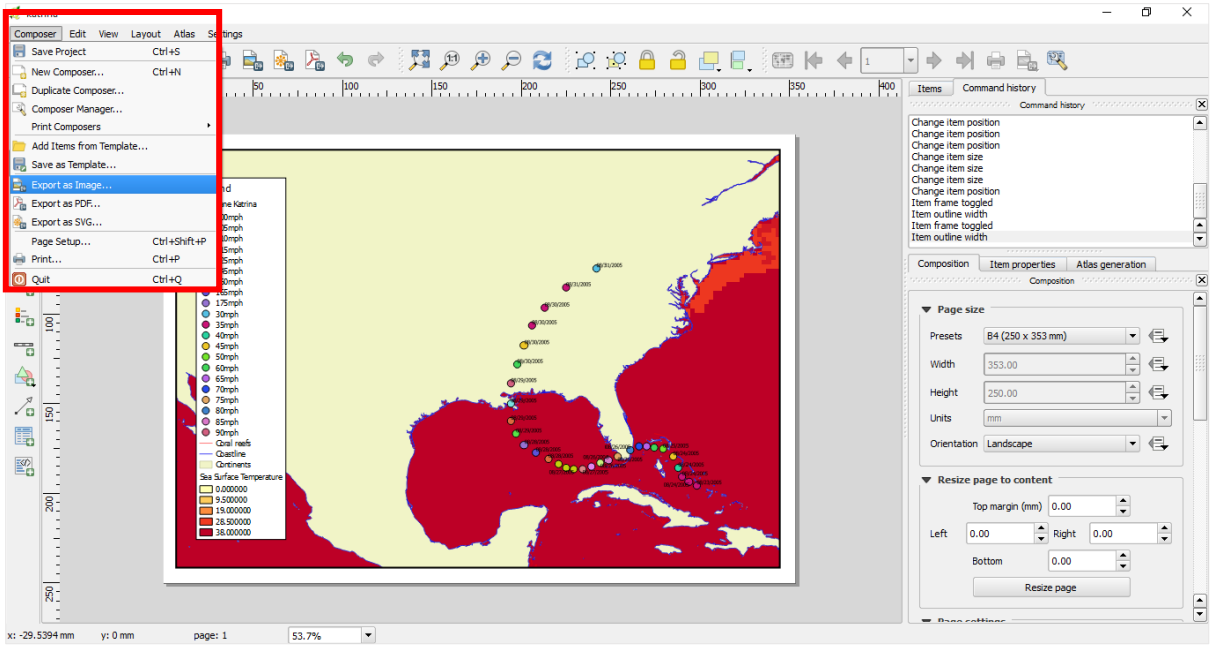
You can easily move the map or zoom as you like using Move item content:

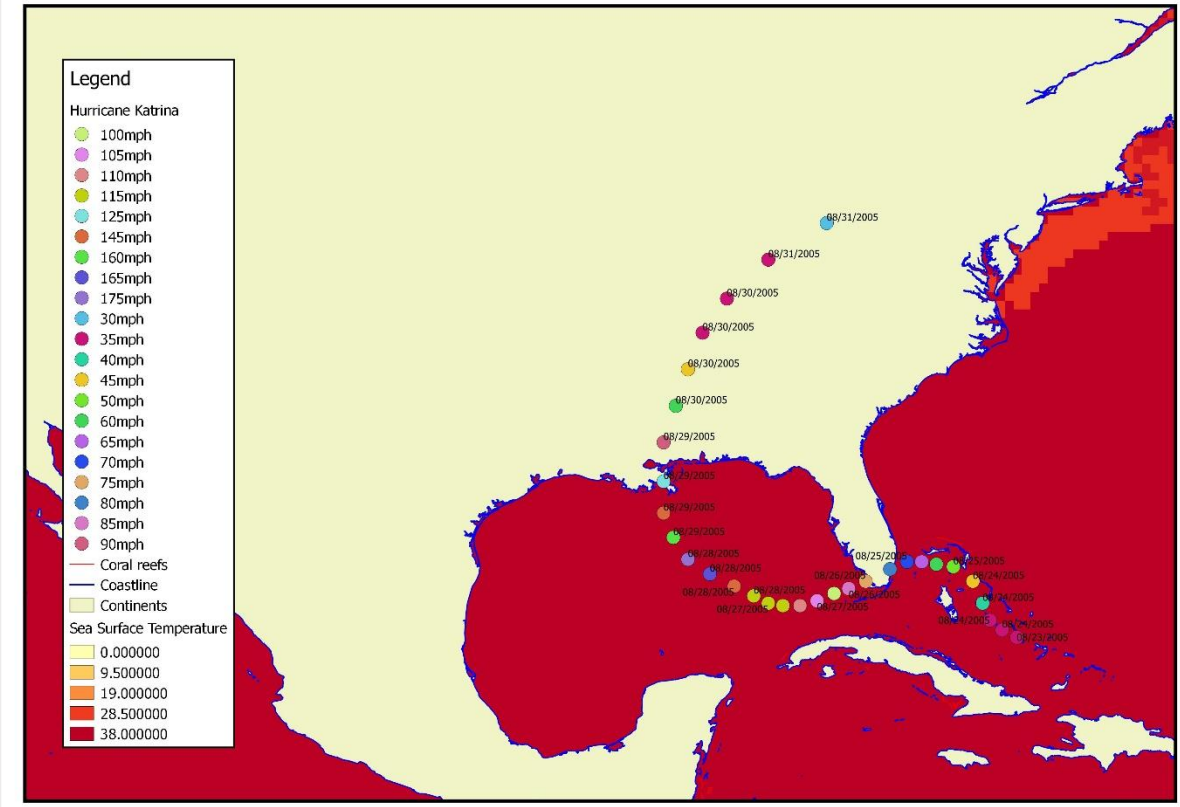


Let's add the legend. Make sure you have the right Page size:



Once your map is ready you can save it as Image, PDF or SVG:

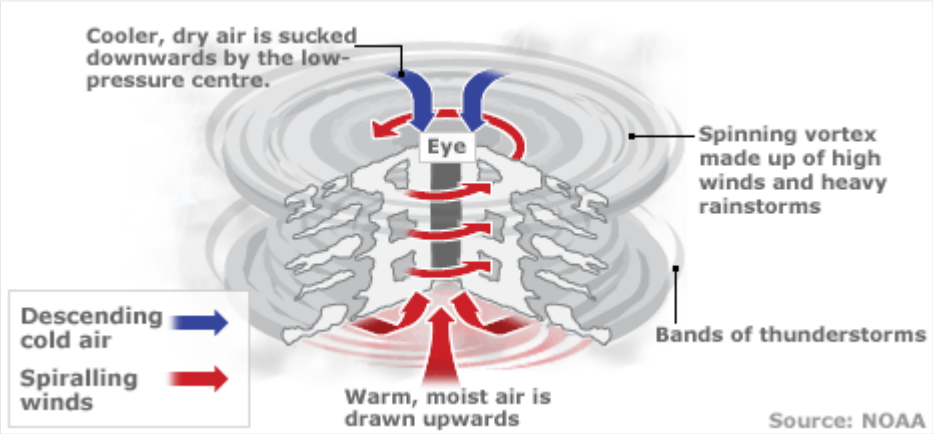




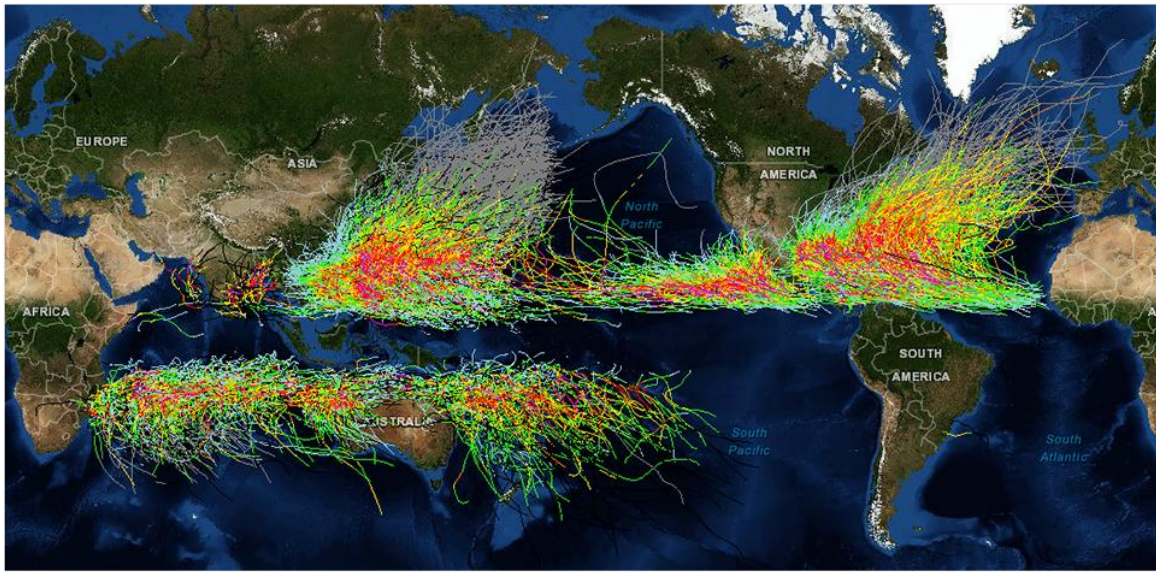
IV. Discussing Climate Change

Hurricanes / Cyclones / Typhoons

Hurricanes (or cyclones in the South Pacific and the Indian Ocean, and typhoons in the Northwestern Pacific Ocean) form over warm tropical (min. 27°C) waters when rotating (low pressure) winds exceed 33m s^{-1}. A few degrees above normal sea surface temperatures is enough for a hurricane to be formed (Hidore and Oliver, 1993). In the place of the rising hot temperature (high pressure) cold air will be sucked in (low pressure), and as the air rubs on the water and spins due to the rotation of the earth, it will spiral clockwise (S) or anticlockwise (N), depending on its position from the Equator (Elsner and Jagger, 2013). For a storm to become classified as hurricane its winds need to exceed the 125km/h or 75mph speed.



On NOAA's website an interactive map is available showing all major storms since 1857.

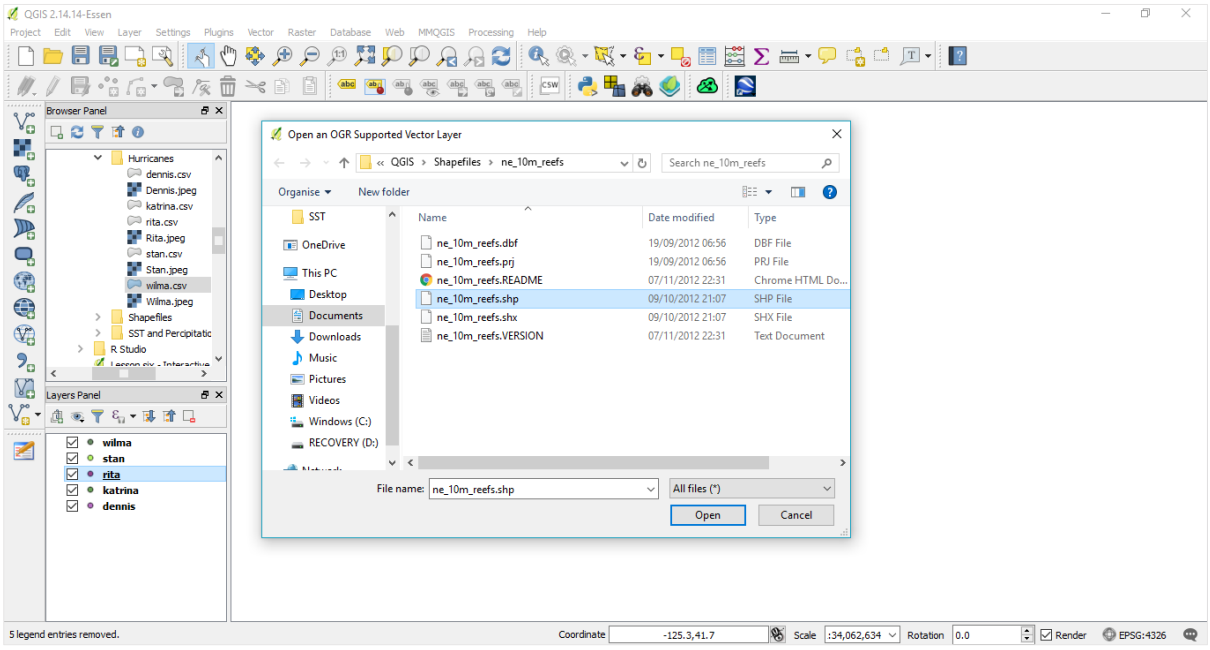


Creating Interactive Maps with R Studio and QGIS

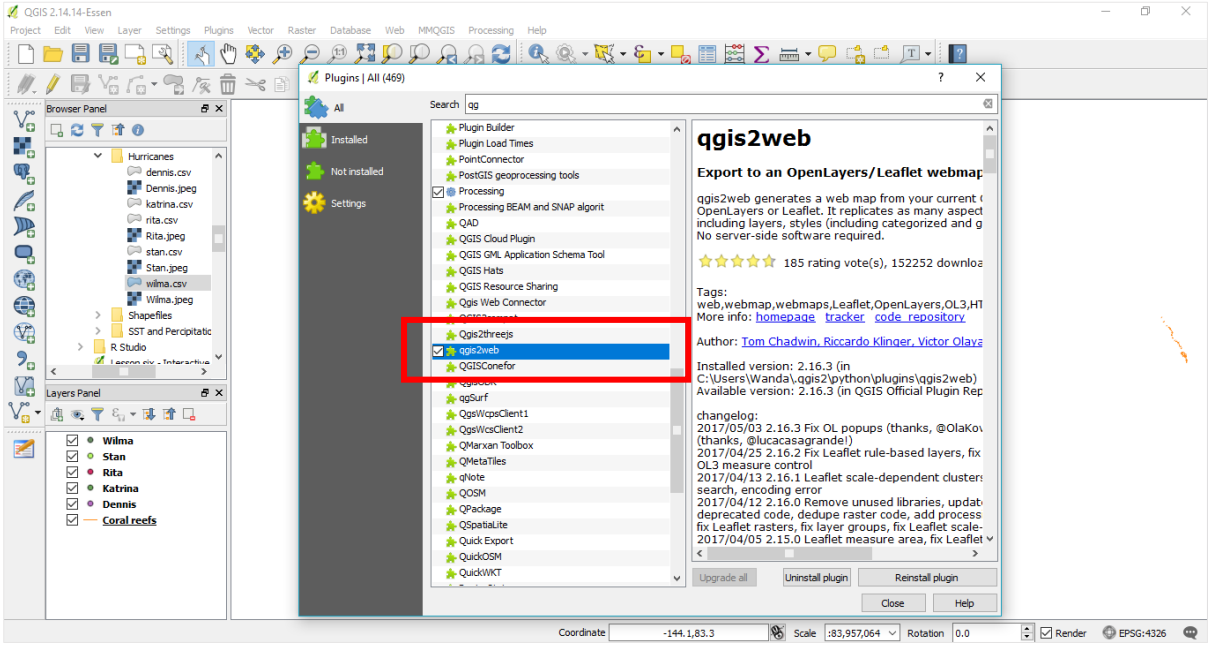
In this chapter we will look into how to create interactive maps that you can save as .html files using both QGIS and R Studio.

1. QGIS

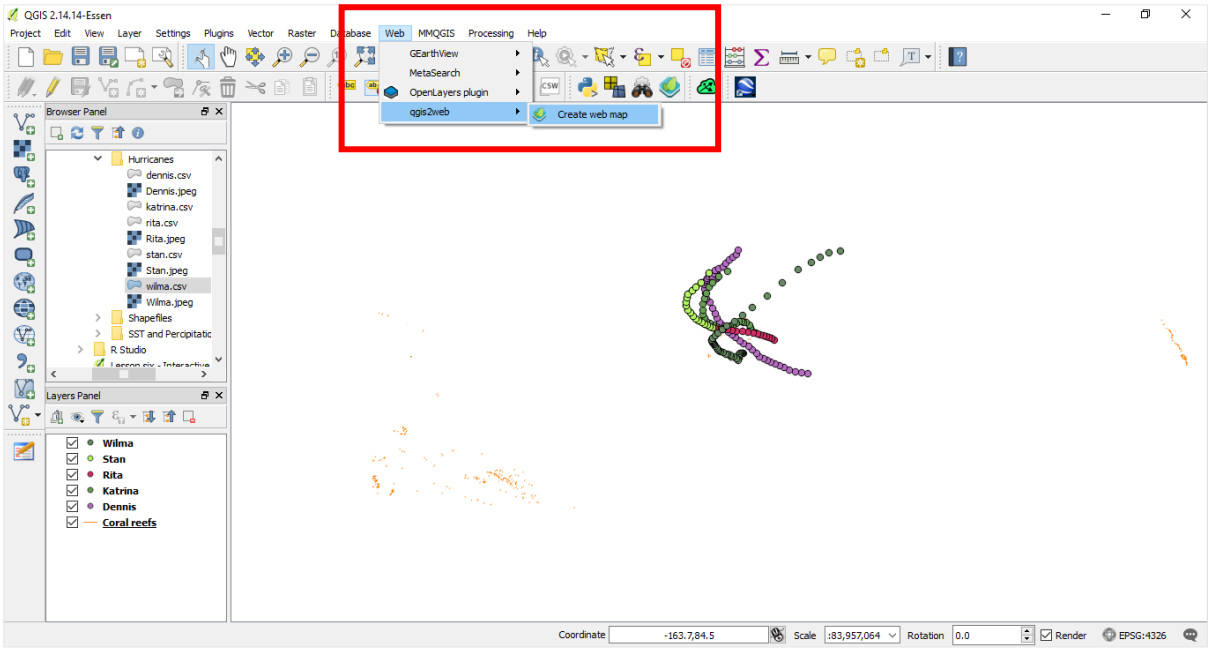
Let's start by opening a new project in QGIS. Now add all 5 major hurricanes and the coral reefs shapefile.



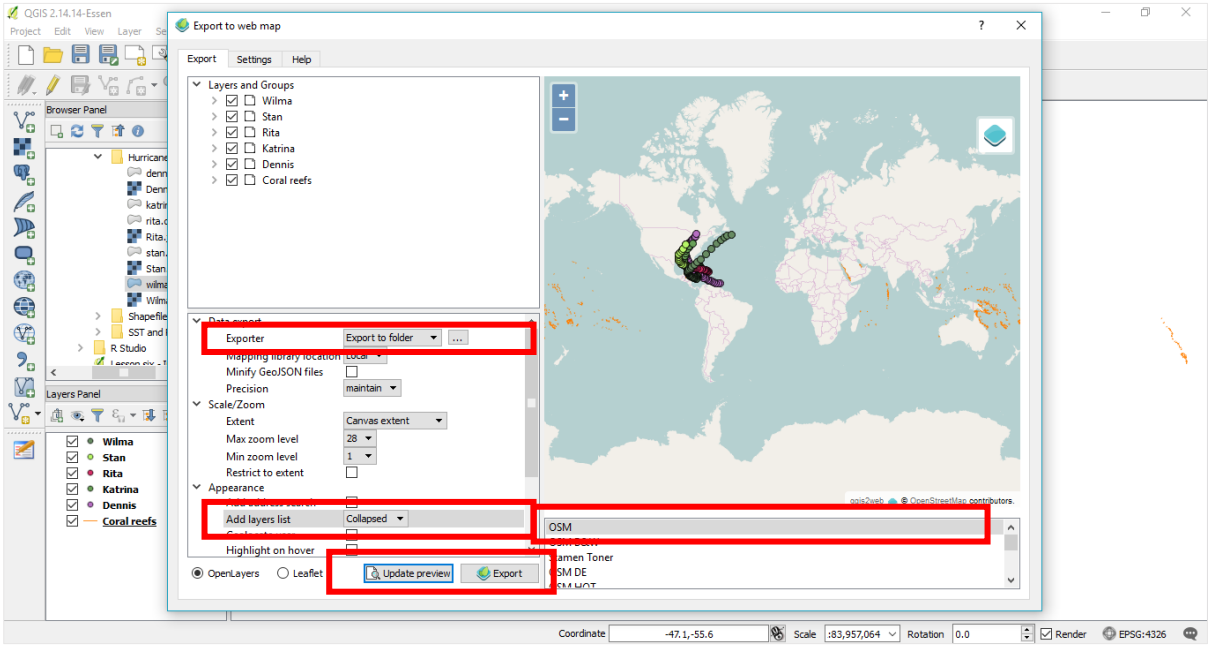
Using the Manage and Install Plugins… menu install qgis2web.



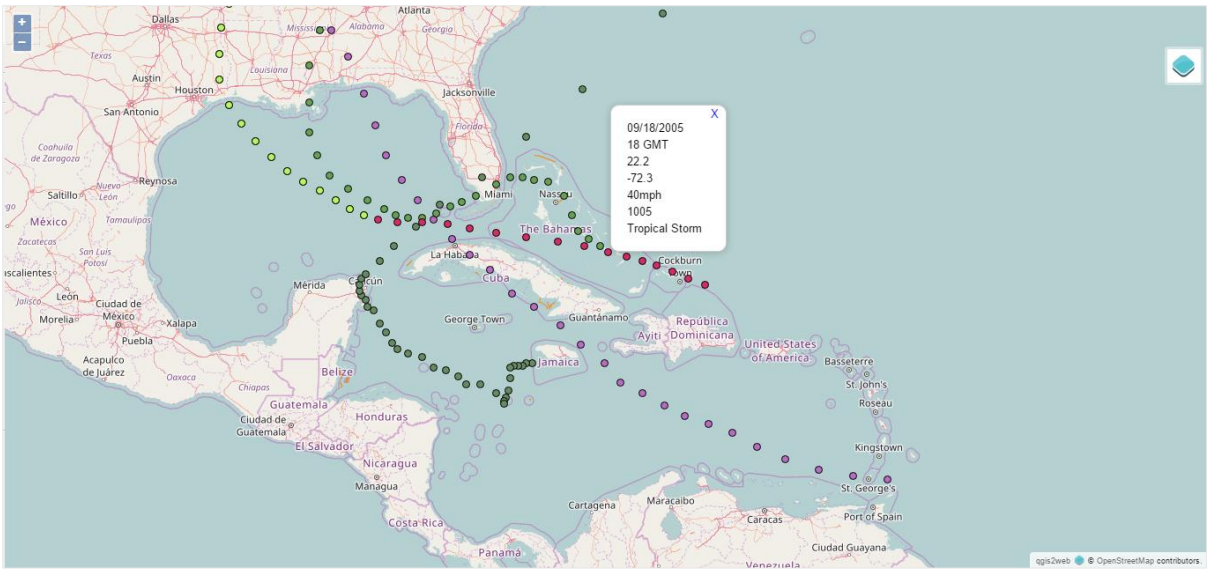
Now go to Web → qgis2web → Create web map.



Choose an Export folder, layers list is 'Collapsed' and the background map is 'OSM'.



Click Export.



The map should appear in your default browser and also in your chosen folder as index.html.

2. R Studio

For the last part of the course we will use R Studio and the leaflet package to build a map that shows sea surface temperature, hurricanes, coral reefs and marine ecoregions. We will also add a GIF image, a choropleth layer which will indicate damages caused by Hurricane Katrina by state, and a WMS layer that shows the population density.

Before we start, let's make sure all the data is available. We will need: SST, the 5 major hurricanes, coral reefs (these you already have from the previous exercises). Marine ecoregions shapefile is available here, the GIF file is here, and the geographic lines shapefile is here.

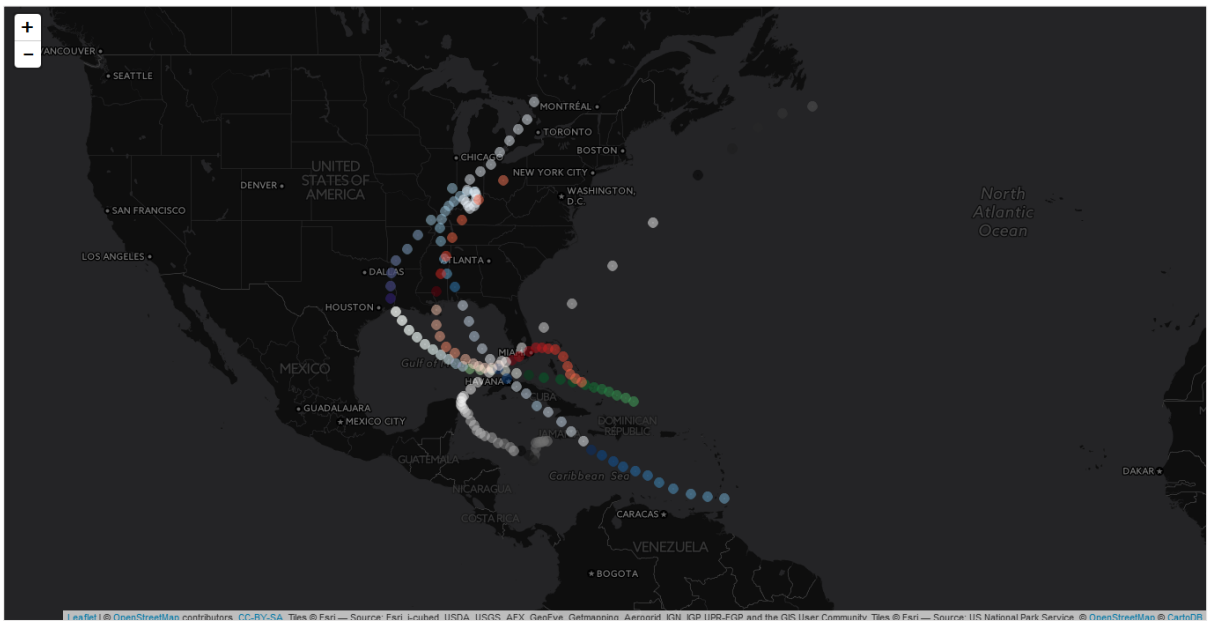
Let's open R Studio and load the necessary packages.

```
library(leaflet)
library(leaflet.extras)
library(raster)
library(RColorBrewer)
library(rgdal)
library(mapview)
```

Now add the base maps and hurricane points. For each point we will determine the colour to be used and the 'Group' so the appearance can be controlled with the LayerControl function.

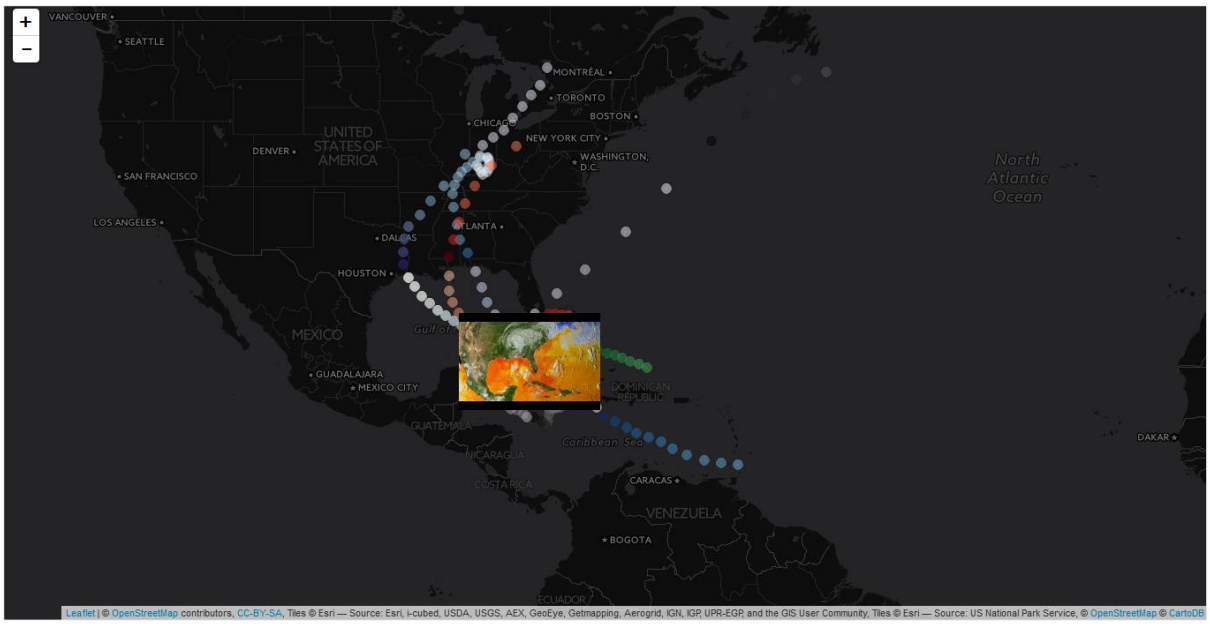
```
m <- leaflet()
m <- setView(m, lng = -60, lat = 17, zoom = 3)
m <- addTiles(m)
m <- addProviderTiles(m, "Esri.WorldImagery", group = "Esri.WorldImagery")
m <- addProviderTiles(m, "Esri.WorldPhysical", group =
"Esri.WorldPhysical")
m <- addProviderTiles(m, "CartoDB.DarkMatter", group =
"CartoDB.DarkMatter")
m
```

```
dennis <- read.csv("dennis.csv")
pal1 <- colorFactor(palette = "Blues", domain = dennis$Wind)
kat <- read.csv("katrina.csv")
pal2 <- colorFactor(palette = "Reds", domain = kat$Wind)
rita <- read.csv("rita.csv")
pal3 <- colorFactor(palette = "Greens", domain = rita$Wind)
stan <- read.csv("stan.csv")
pal4 <- colorFactor(palette = "Purples", domain = stan$Wind)
wilma <- read.csv("wilma.csv")
pal5 <- colorFactor(palette = "Greys", domain = wilma$Wind)

m <- addCircles(m, data = rita, lng = ~Lon, lat = ~Lat, weight = 10, color
= ~pal3(Wind), popup = ~as.character(Type), group = "Hurricane Rita",
label=~as.character(Wind))
m <- addCircles(m, data = stan, lng = ~Lon, lat = ~Lat, weight = 10, color
= ~pal4(Wind), popup = ~as.character(Type), group = "Hurricane Stan",
label=~as.character(Wind))
m <- addCircles(m, data = wilma, lng = ~Lon, lat = ~Lat, weight = 10, color
= ~pal5(Wind), popup = ~as.character(Type), group = "Hurricane Wilma",
label=~as.character(Wind))
m <- addCircles(m, data = dennis, lng = ~Long, lat = ~Lat, weight = 10,
color = ~pal1(Wind), popup = ~as.character(Type), group = "Hurricane
Dennis", label=~as.character(Wind))
m <- addCircles(m, data = kat, lng = ~Long, lat = ~Lat, weight = 10, color
= ~pal2(Wind), popup = ~as.character(Type), group = "Hurricane Katrina",
label = ~as.character(Wind))
m
```



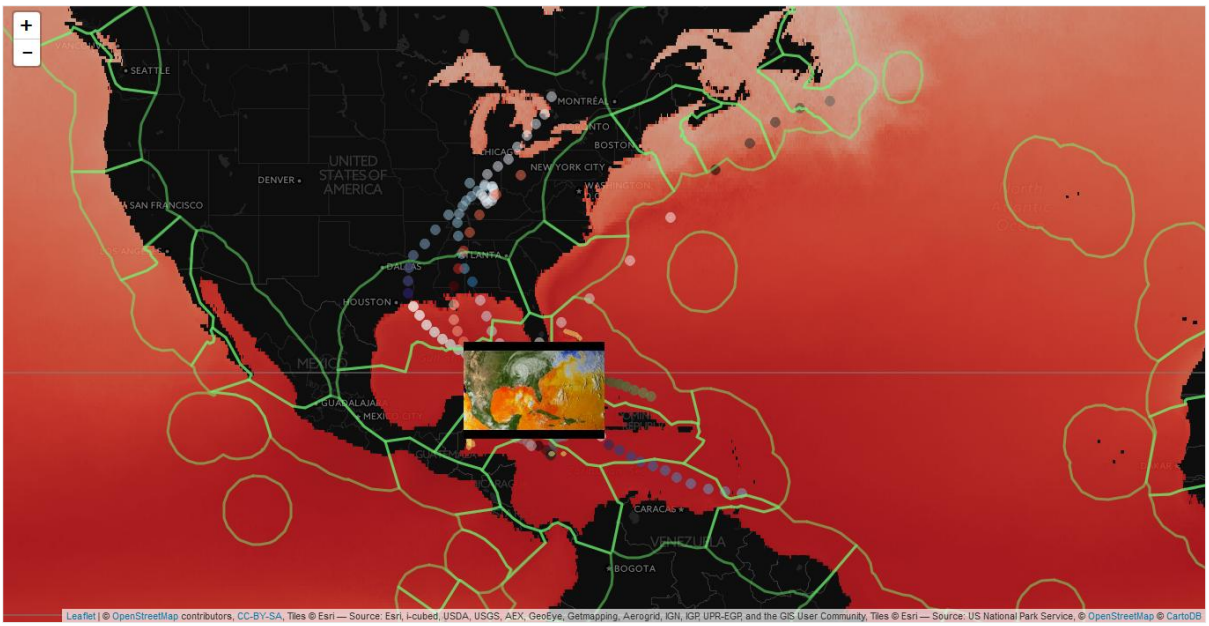
Now here is the Hurricane Katrina GIF added to the point where the wind speed was maximum.

```
k <- iconList(k = makeIcon("katrina_sstHD.gif"))
m <- addMarkers(m, lng = -88.6, lat = 26.3, icon = k, group = 'GIF')
m
```



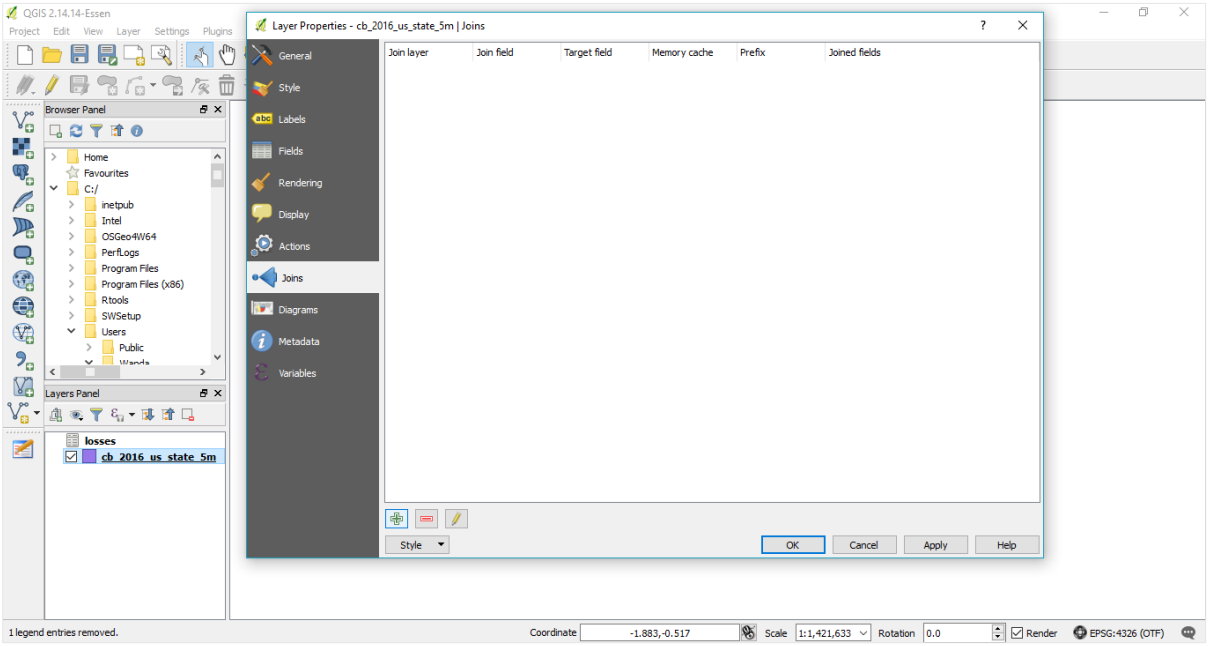
Let's add the SST data, the coral reef, marine ecoregions, and geographic lines shapefiles.

```
sst <- raster("g4.timeAvgMap.MODISA_L3m_SST_2014_sst.20050101-
20051231.180W_90S_180E_90N.nc")
pal6 <- colorNumeric(c("Reds"), values(sst), na.color = "transparent")
m <- addRasterImage(m, sst, colors = pal6, opacity = 0.8, group = "Sea
Surface Temperature")

coral <- shapefile("ne_10m_reefs.shp")
m <- addPolygons(m, data = coral, fill = F, weight = 5, color = "#F6FA6E",
group = "Coral Reefs")

gl <- shapefile("ne_10m_geographic_lines.shp")
m <- addPolygons(m, data = gl, fill = F, weight = 2, color = "Grey",
label=~as.character(name))

marineecoz <- shapefile("meow_ecos.shp")
m <- addPolygons(m, data = marineecoz, fill = F, weight = 3, color =
"#6EFA73", label=~as.character(ECOREGION), group = "Marine Ecoregions")

m
```

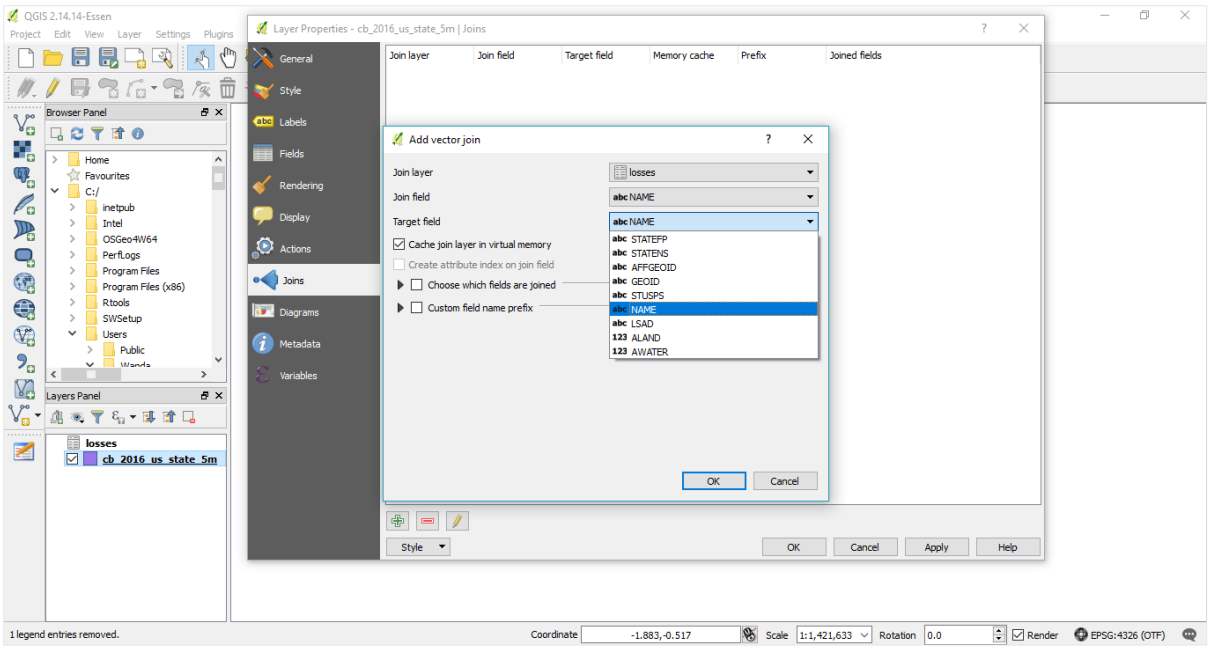


To add the choropleth map we will use data from The Insurance Information Institute on insured losses by state. Copy paste data on the six states (Mississippi, Alabama, Tennessee, Florida, Louisiana and Georgia) into .csv files and save it along with the States shapefile (5m) which is available here. Make sure that the state name column is labelled as NAME.

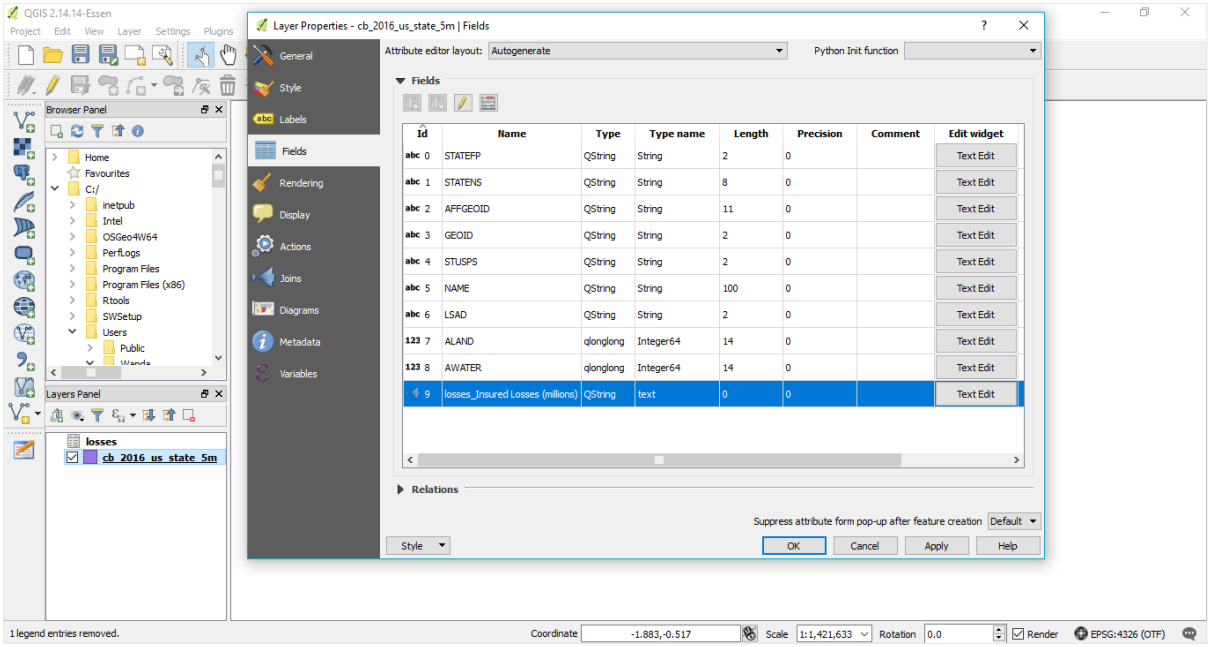
First open QGIS and add both the losses.csv and cb_2016_us_state_5m.shp files. Then right click on the shapefile, then Properties and Joins.



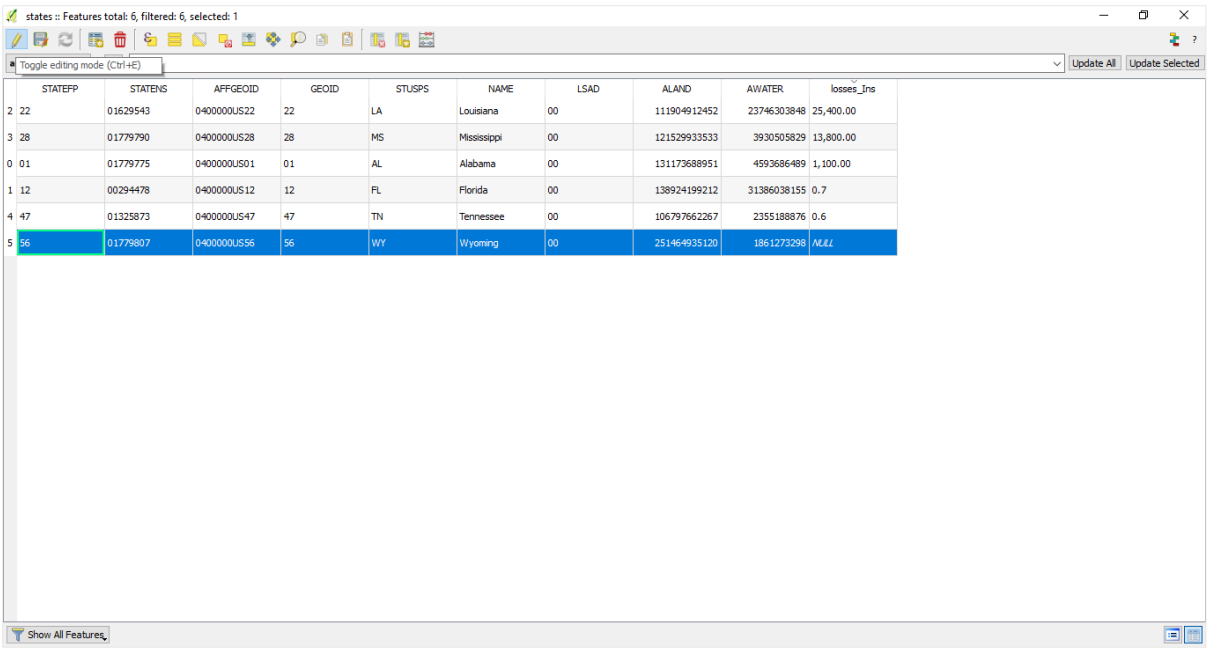
Click on the + icon. The Join layer is the losses.csv file, the Join field is the NAME from the .csv file and the Target field is the shapefile.



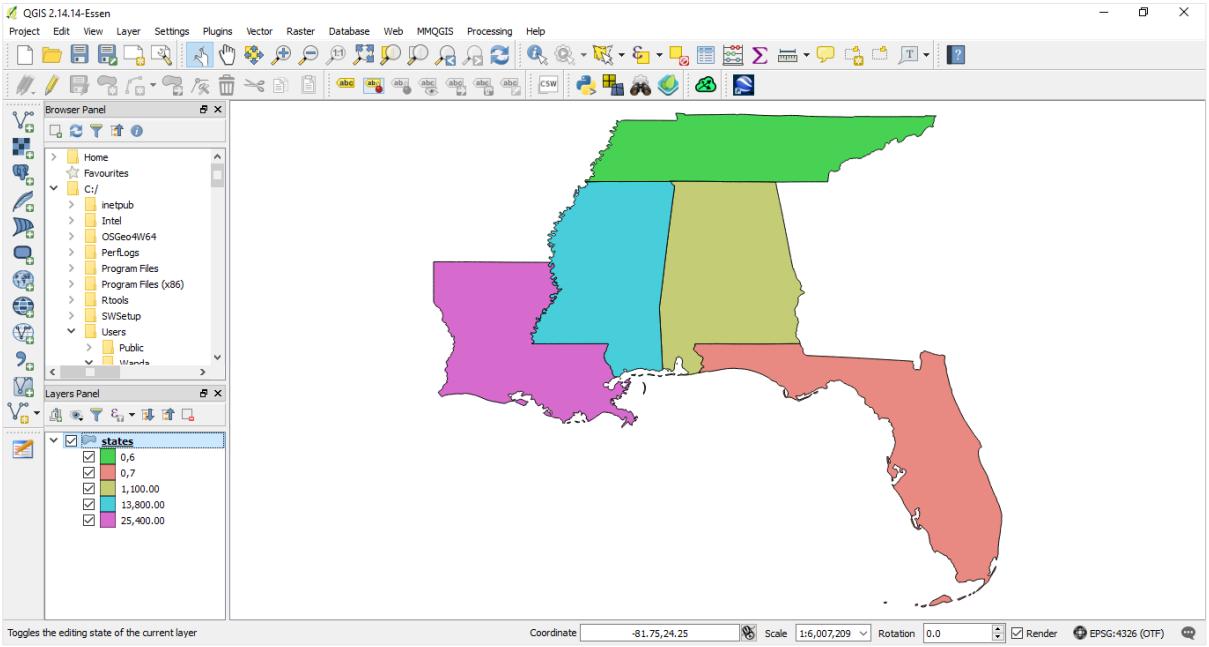
Click Apply then to go Field. You will see the extra row added.



Save the shapefile layer as an ESRI shapefile, then remove the .csv file and the original shapefile. Open the Attribute table of the newly created shapefile. You will see the information that we need in the last column. After clicking on the Toggle editing icon, highlight and delete the unwanted rows.

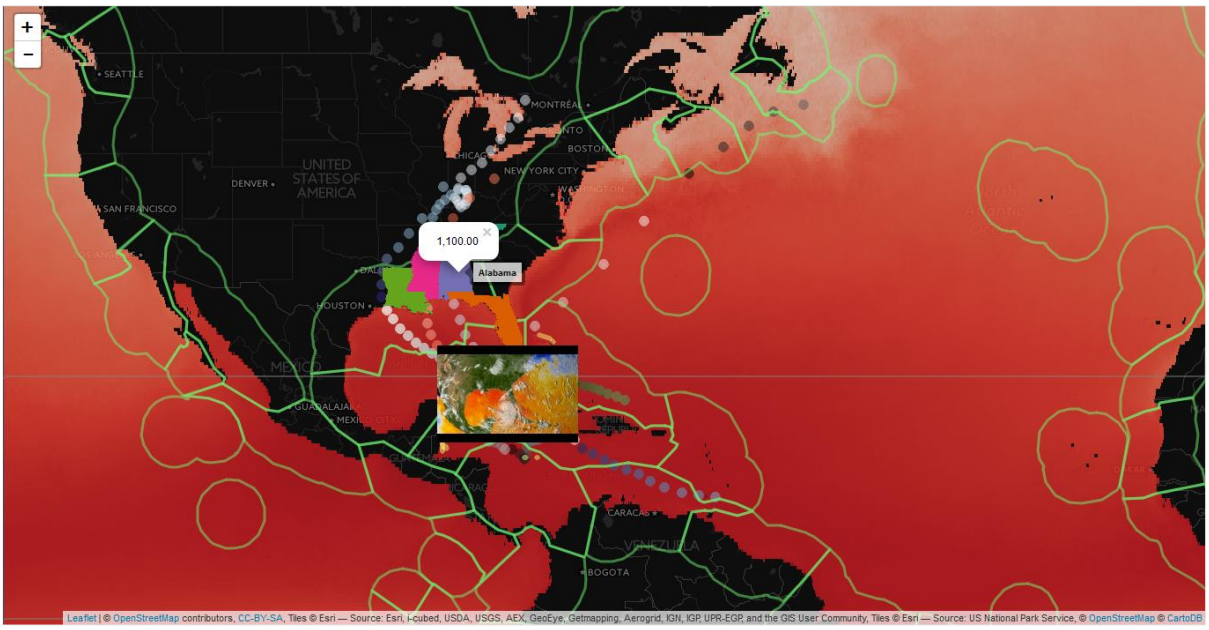


You will have the 5 states appear.



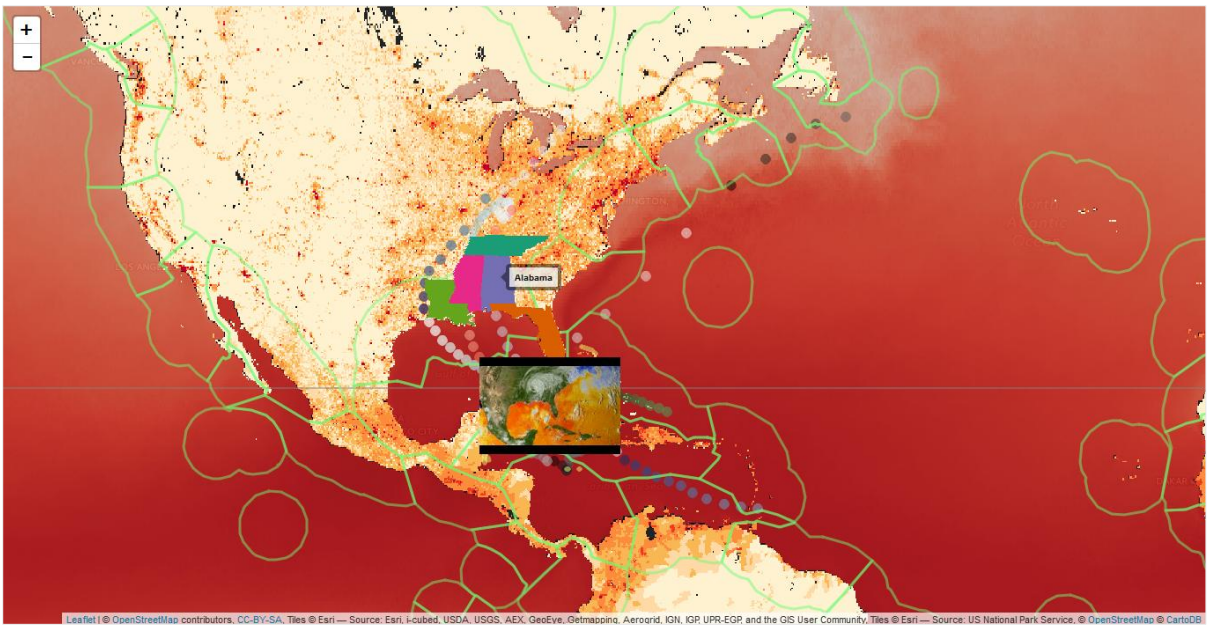
Now let's add this states shapefile to our leaflet map.

```
state <- readOGR("states.shp")
pal7 <- colorFactor("Dark2", domain = state$losses_Ins)
m <- addPolygons(m, data = state, stroke = FALSE, smoothFactor = 0.2,
fillOpacity = 1,
                 color = ~pal7(losses_Ins),
                 label= ~as.character(NAME),
                 popup = ~as.character(losses_Ins),
                 group = "Hurricane Katrina damage")
m
```



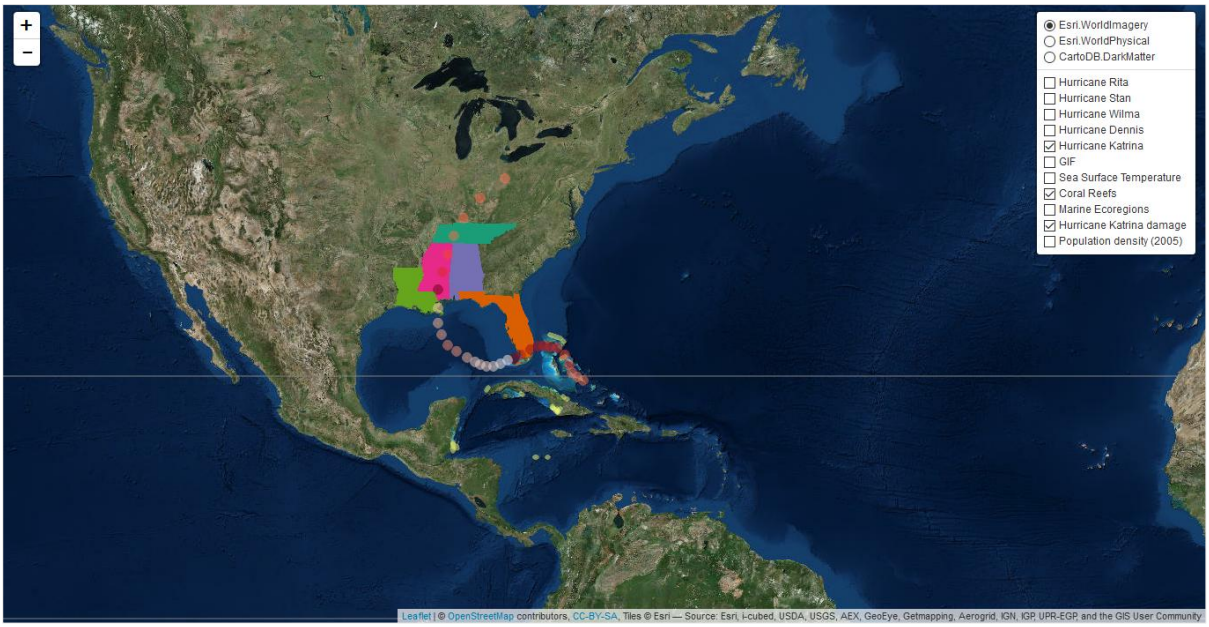
We will add the WMS (Web Mapping Services) layer from this website.

```
m <- addWMSTiles(m, "http://sedac.ciesin.columbia.edu/geoserver/wms",
                 layers = "gpw-v4:gpw-v4-population-density_2005",
                 options = WMSTileOptions(format = "image/png", transparent
= TRUE),
                 group = "Population density (2005)")
m
```



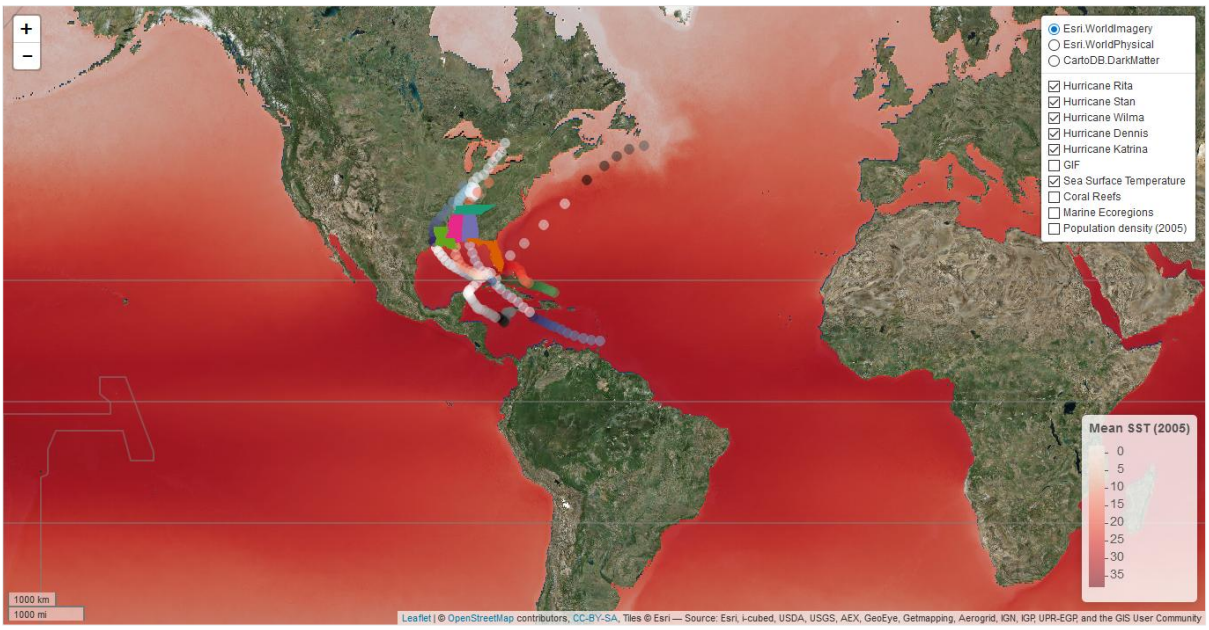
Now let's add the layer control.

```
m <- addLayersControl(m,
                      baseGroups = c("Esri.WorldImagery",
"Esri.WorldPhysical", "CartoDB.DarkMatter"),
                      overlayGroups = c("Hurricane Rita", "Hurricane Stan",
"Hurricane Wilma", "Hurricane Dennis", "Hurricane Katrina", "Coral Reefs",
"Marine Ecoregions", "Sea Surface Temperature", "Hurricane Katrina damage",
"Population density (2005)", "GIF"),
                      options = layersControlOptions(collapsed = TRUE),
                      position = 'topright')
m
```



Finally, the legend for the SST values and a scalebar.

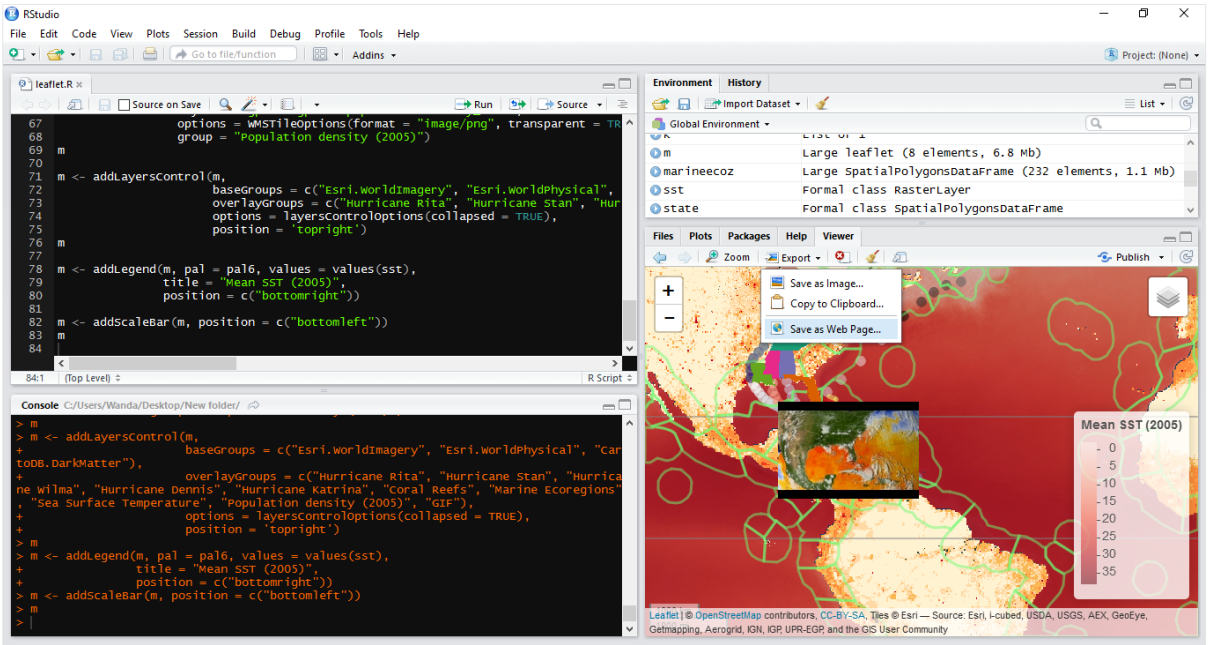
```
m <- addLegend(m, pal = pal, values = values(sst),
               title = "Mean SST (2005)",
               position = c("topleft"))

m <- addScaleBar(m, position = c("bottomleft"))
m
```



This is the final map.

3. Publishing your map online

This can either be saved as a .html file and use it as a HTTP code.



Or published online by clicking on ‘Publish’ using RPubs.

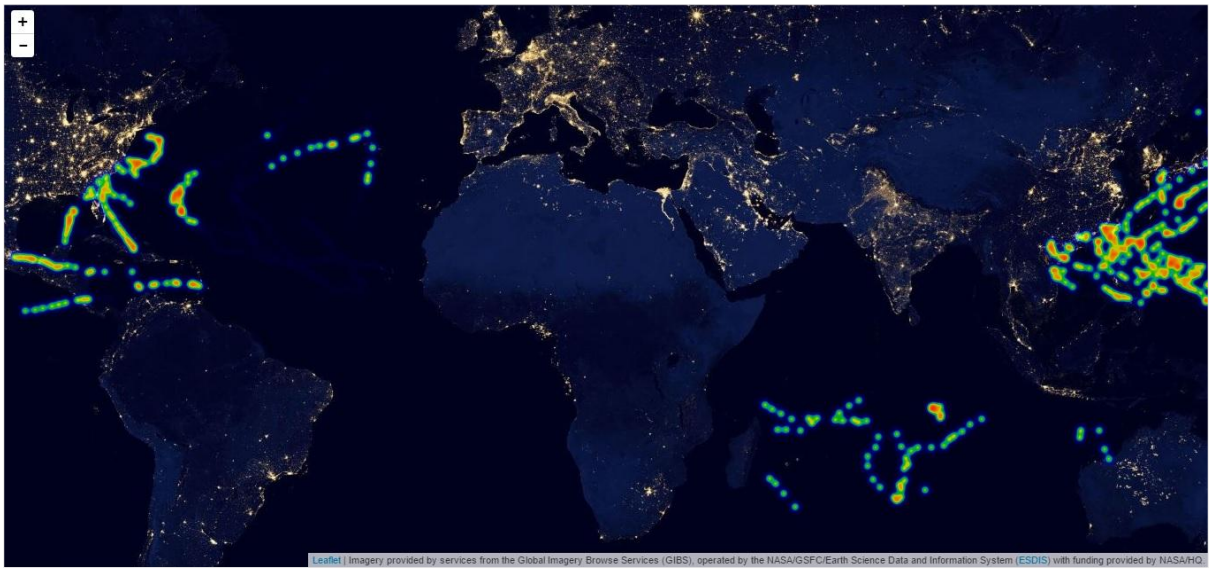
Additional data visualisation & mapping tips

1. Heatmap (leaflet.extras)

Hurricane and tropical cyclone data can be found on the Weather Underground website. Data is available from 1851 until present. For this exercise, we will use all hurricane and cyclone data from 2016. This exercise will require the installation of leaflet.extras.

The heatmap is specifically focusing on wind speed (mph):

```
devtools::install_github('rstudio/leaflet')
devtools::install_github('bhaskarvk/leaflet.extras')

#data source: https://www.wunderground.com/hurricane/at2016.asp

library(leaflet)
library(leaflet.extras)

storms <- read.csv("storms.csv")

leaflet(storms) %>% addProviderTiles(providers$
NASAGIBS.ViirsEarthAtNight2012) %>%
  addHeatmap(lng = ~Lon, lat = ~Lat, intensity = ~Wind,
             blur = 7, max = 0.2, radius = 5)
```



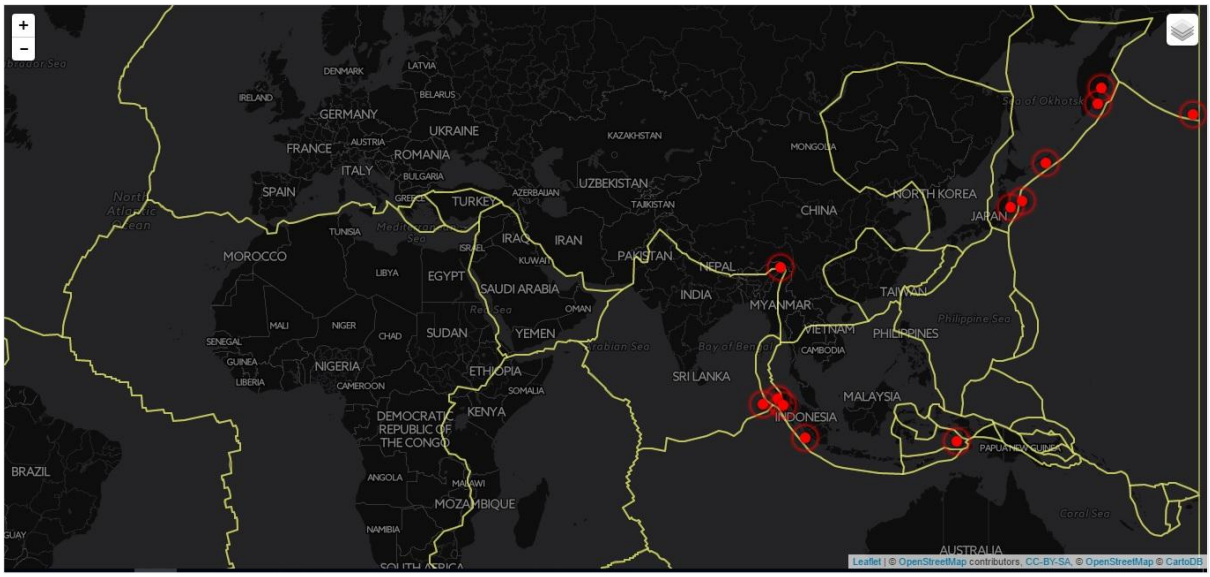
The resulting map shows all hurricanes and tropical cyclones that occurred in 2016.

2. Pulsemarkers (leaflet.extras)

Pulsemarkers are also in the leaflet.extras package, and for this exercise we will use data to display the top largest earthquakes in the world. We will also add the tectonic plates as shapefiles.

```
library(leaflet)
library(leaflet.extras)
library(raster)
library(rgdal)
library(mapview)
```

```
e <- read.csv("earthquakes.csv")
s <- shapefile("PB2002_plates.shp")

m <- leaflet()
m <- addTiles(m)
m <- setView(m, lng = 0, lat = 0, zoom = 3)
m <- addProviderTiles(m, "CartoDB.DarkMatter", group =
"CartoDB.DarkMatter")
m <- addProviderTiles(m, "Esri.WorldImagery", group = "Esri.WorldImagery")
m <- addProviderTiles(m, "Esri.WorldPhysical", group =
"Esri.WorldPhysical")
m <- addPolygons(m, data = s, fill = F, weight = 2,
                 color = "#F6FA6E", popup = ~as.character(PlateName),
                 group = "Tectonic Plates")
m <- addPulseMarkers(m, data = e, lng = ~Longitude, lat = ~Latitude,
                     popup = ~as.character(Alternative.Name),
                     label = ~as.character(Mag),
                     icon = makePulseIcon(heartbeat = 1),
                     group = "Locations")
m <- addLayersControl(m,
                      baseGroups = c("CartoDB.DarkMatter",
"Esri.WorldImagery", "Esri.WorldPhysical"),
                      overlayGroups = c("Tectonic Plates", "Locations"),
                      options = layersControlOptions(collapsed = TRUE),
                      position = 'topright')
m
```



The resulting map may not show the icons pulsing in R Studio's viewer pane, but if you click on 'Show in a new window' icon it will appear in your default browser. If you hover over the points, the magnitude will appear, and if you click on it, the name of the location will pop up. By clicking on the layer control you are also able to change the map provider.

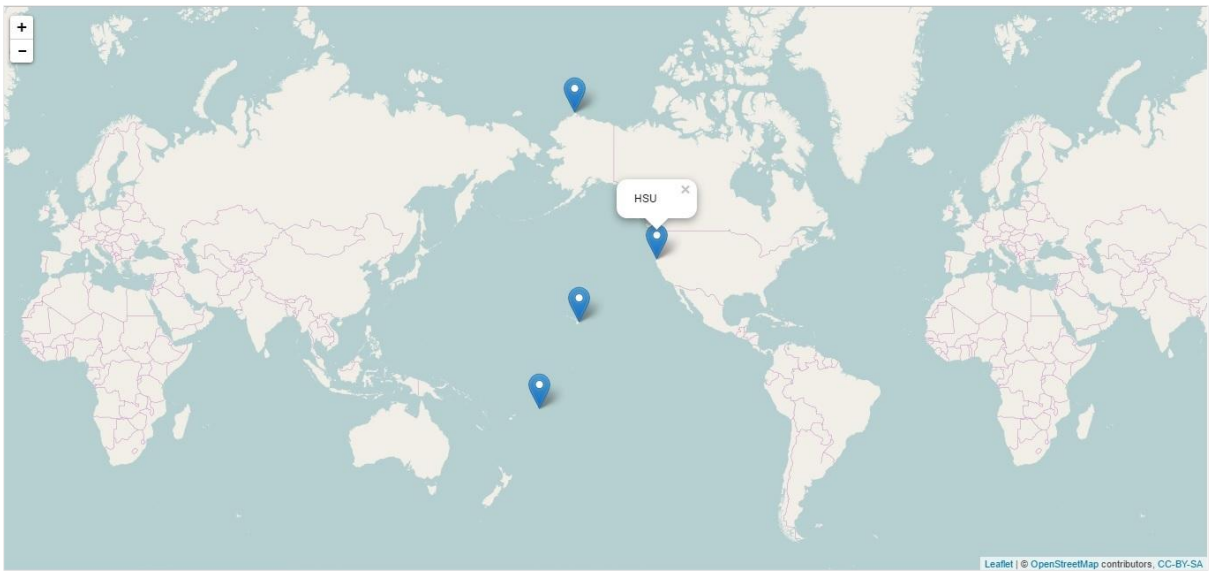
3. Geocoding / Georeferencing

If you know the address of a certain place, you can easily map them both In R Studio and QGIS. Let's look at the four main atmospheric carbon-dioxide baseline observatories around the globe:

Mauna Loa Observatory 1437 Kilauea Ave. 102 Hilo, Hawaii, 96720 United States	**Barrow Observatory** PO Box 888 Barrow, Alaska, 99723 United States
HSU Marine Lab 570 Ewing St Trinidad, California, 95570 United States	**Samoa Observatory** PO Box 2568 Pago Pago, Eastern, 96799 American Samoa

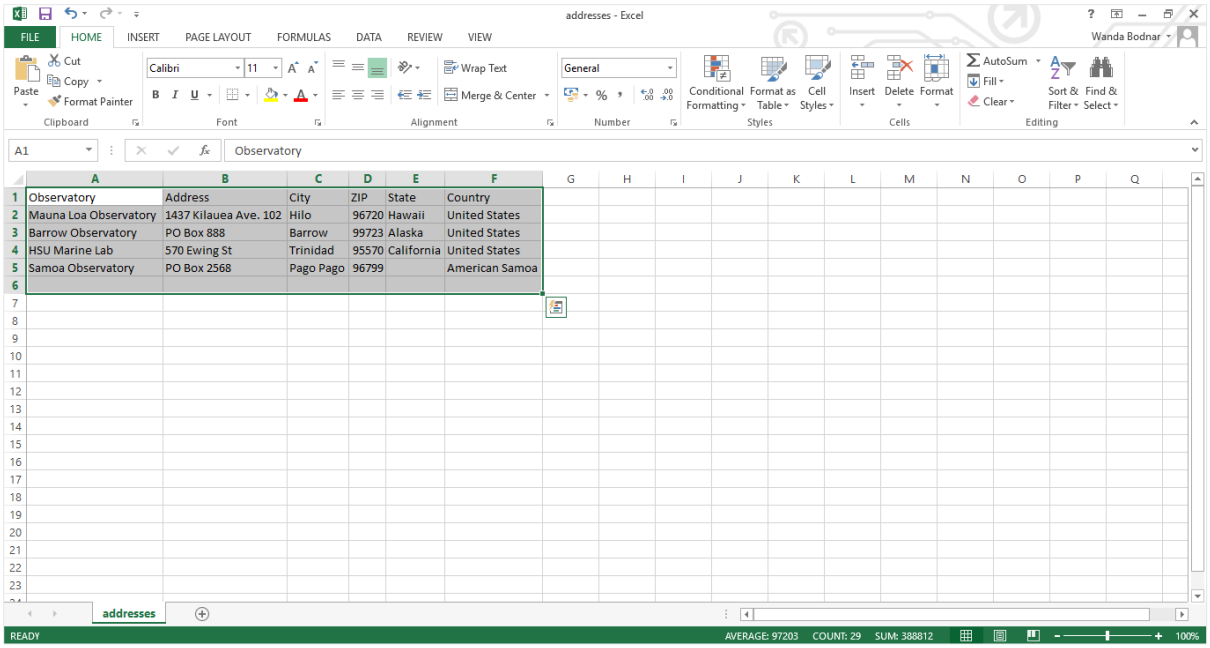
R Studio

```
library(leaflet)
library(ggmap)

MLO <- geocode("1437 Kilauea Ave. 102 Hilo, Hawaii, 96720, United States",
output = "latlon" , source = "google")
MLO

BO <- geocode("Barrow Observatory PO Box 888 Barrow, Alaska, 99723, United
States", output = "latlon" , source = "google")
BO


HSU <- geocode("HSU Marine Lab 570 Ewing St Trinidad, California, 95570,
United States", output = "latlon" , source = "google")
HSU


SMO <- geocode("Pago Pago, Eastern, 96799, American Samoa", output =
"latlon" , source = "google")
SMO


#dataframe
df <- as.data.frame(rbind(MLO, BO, HSU, SMO))
df

df$Locations <- c("MLO", "BO", "HSU", "SMO")   #column with Locations
df

leaflet(data = df) %>% addTiles() %>%
  addMarkers(~lon, ~lat, popup = ~as.character(Locations))
```



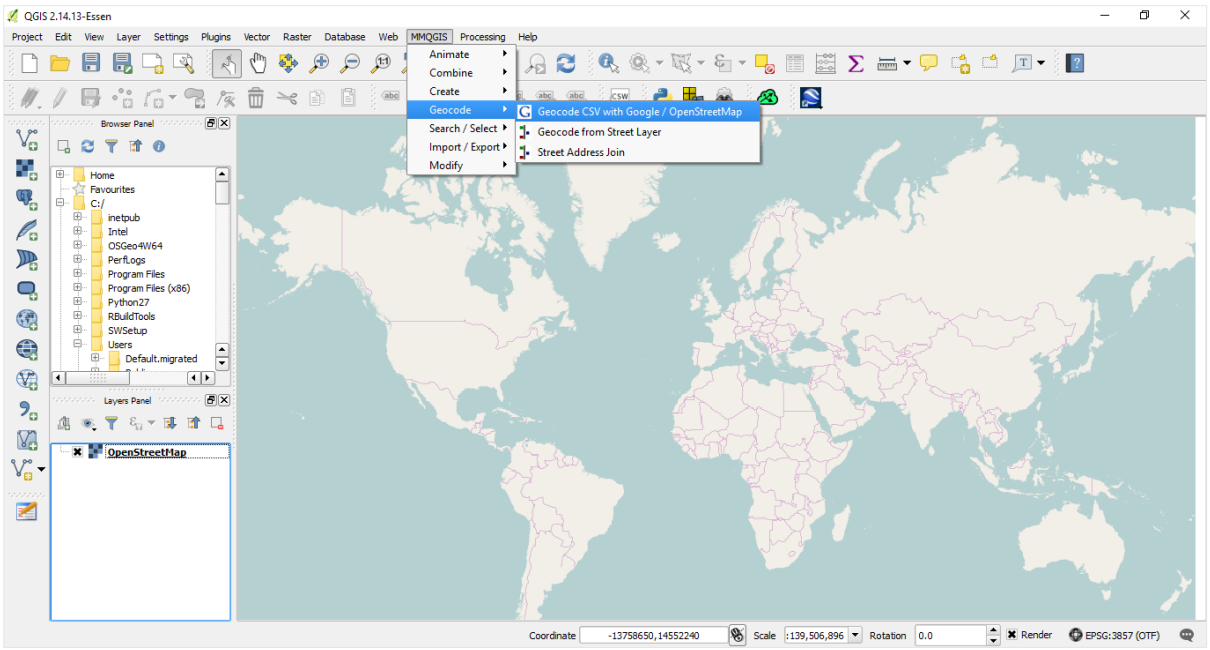
QGIS

Using the same address, make sure you have your Excel sheet neat and tidy, and saved as .csv.

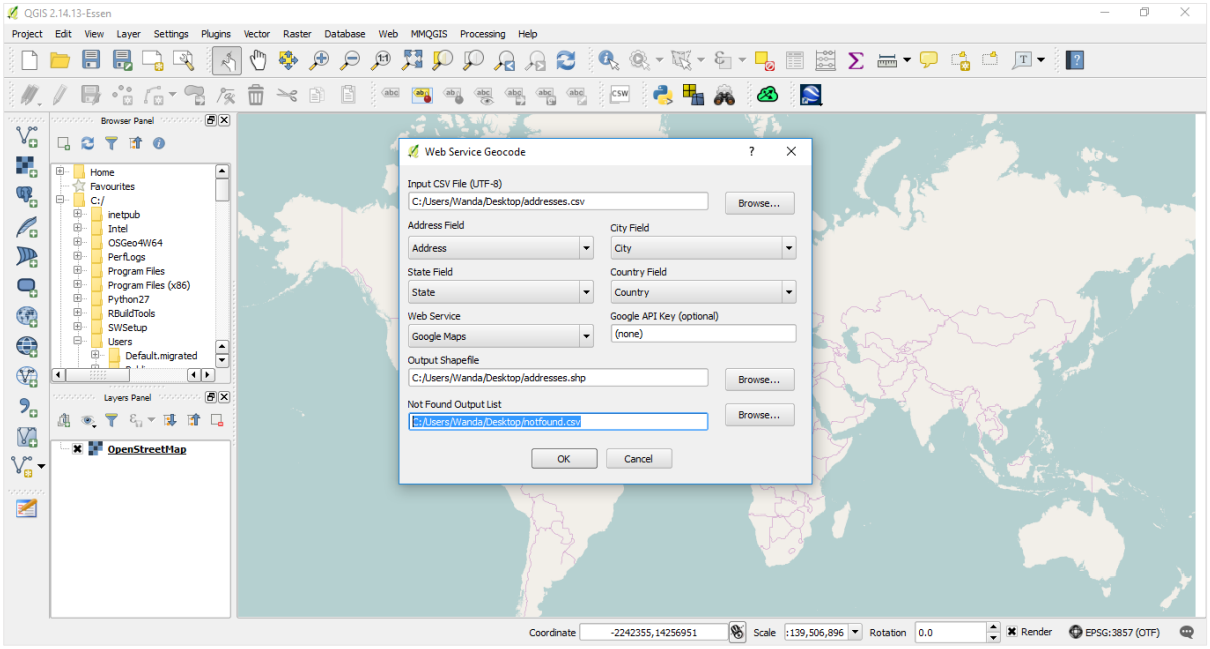


Observatory	Address	City	ZIP	State	Country
Mauna Loa Observatory	1437 Kilauea Ave. 102	Hilo	96720	Hawaii	United States
Barrow Observatory	PO Box 888	Barrow	99723	Alaska	United States
HSU Marine Lab	570 Ewing St	Trinidad	95570	California	United States
Samoa Observatory	PO Box 2568	Pago Pago	96799		American Samoa

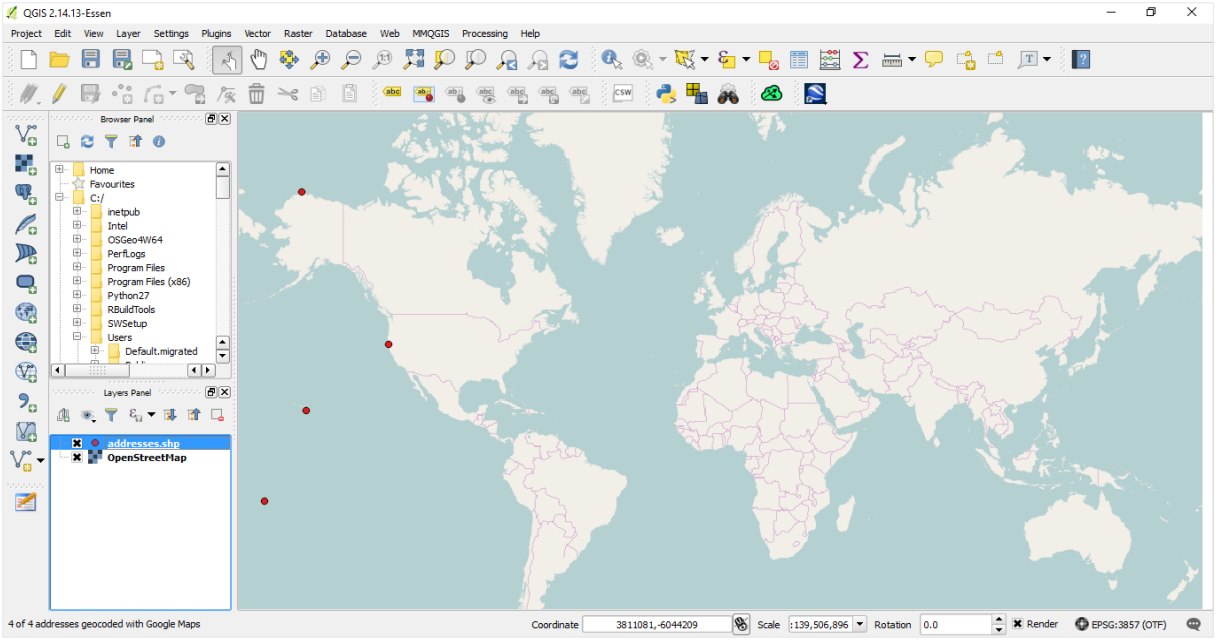
Now open QGIS and install the mmqgis plugin, then go to MMQGIS in the menu bar, then Geocode → Geocode with CSV with Google / OpenStreetMap.



Choose the address.csv, and make sure that the notfound.csv is set on the same directory. Click OK.



The points should appear on your map.



4. Adding geotagged pictures to leaflet map

Pictures that have GPS coordinates stored inside their EXIF data can be added to your map. Let's take the example of three pictures that were taken at three different places. I have added these to Flickr as it is one of the few web photo application sites that doesn't strip the EXIF data off a picture.

Now let's build up our code using the 'exifr' package.

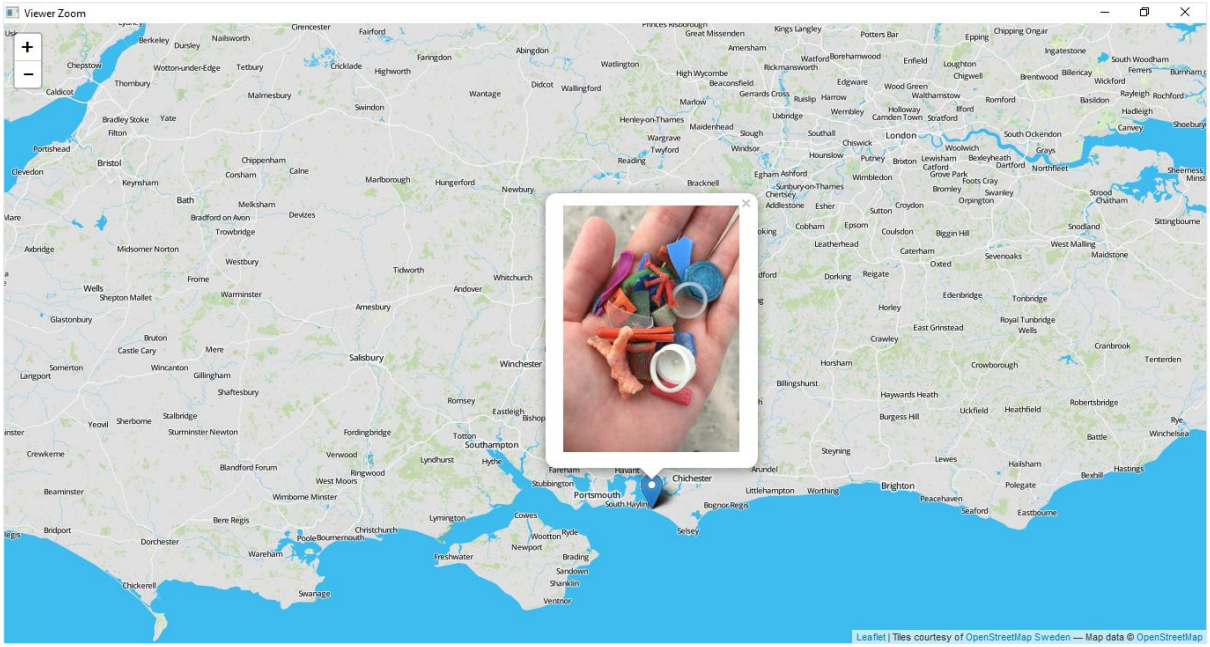
```
library(leaflet)
library(exifr)

content1 <- paste(sep = "<br/>",
                  "<img
src='https://c1.staticflickr.com/5/4288/35154951026_4ab981910a_b.jpg'
                  style='width:200px;height:280px;'>")
```

The content source is the picture on Flickr, and the picture data will come from the picture in your folder using the exifr function to extract the coordinates.

```
scene1 <- exifr('IMG_2965.jpg')
x1 <- scene1$GPSLongitude
y1 <- scene1$GPSLatitude
```

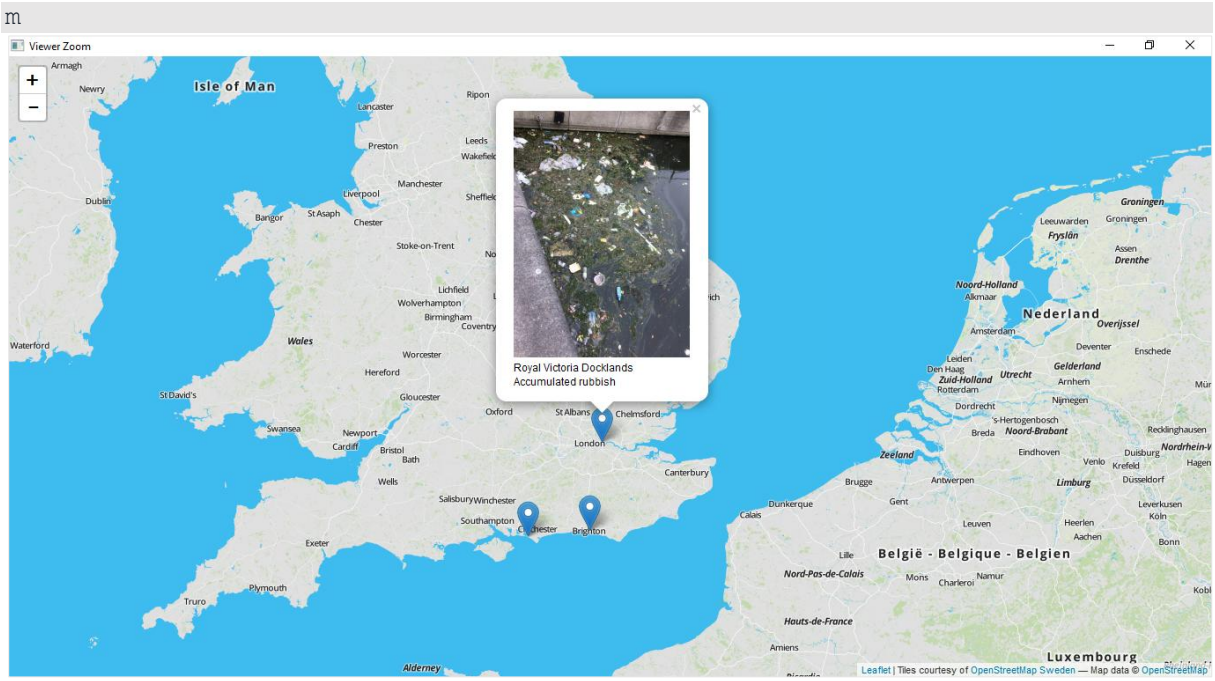
Now let's add it to the map.

```
m <- leaflet()
m <- addProviderTiles(m, "Hydda.Full")
m <- addMarkers(m, x1, y1, popup = content1)
m
```



Now let's add the other two pictures too.

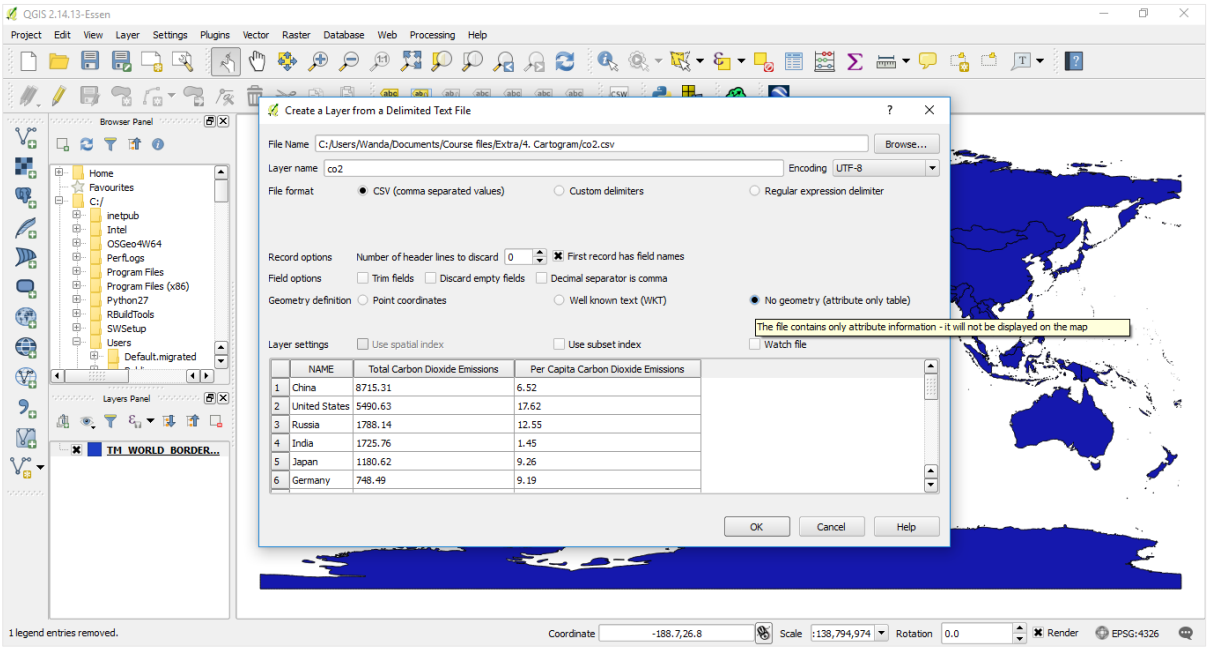
```
content1 <- paste(sep = "<br/>",
                  "<img
src='https://c1.staticflickr.com/5/4288/35154951026_4ab981910a_b.jpg'
                  style='width:200px;height:280px;'>",
                  "West Wittering",
                  "Plastic pieces collected ")
scene1 <- exifr('IMG_2965.jpg')
x1 <- scene1$GPSLongitude
y1 <- scene1$GPSLatitude


content2 <- paste(sep = "<br/>",
                  "<img
src='https://c1.staticflickr.com/5/4254/34808757650_1a64d48b02_b.jpg'
                  style='width:280px;height:200px;'>",
                  "Brighton",
                  "Polysteryn piece")
scene2 <- exifr('IMG_2982.jpg')
x2 <- scene2$GPSLongitude
y2 <- scene2$GPSLatitude

content3 <- paste(sep = "<br/>",
                  "<img
src='https://c1.staticflickr.com/5/4213/35195401115_907b7928b4_b.jpg'
                  style='width:200px;height:280px;'>",
                  "Royal Victoria Docklands",
                  "Accumulated rubbish")
scene3 <- exifr('IMG_2986.jpg')
x3 <- scene3$GPSLongitude
y3 <- scene3$GPSLatitude


m <- leaflet()
m <- addProviderTiles(m, "Hydda.Full")
m <- addMarkers(m, x1, y1, popup = content1)
m <- addMarkers(m, x2, y2, popup = content2)
m <- addMarkers(m, x3, y3, popup = content3)
```



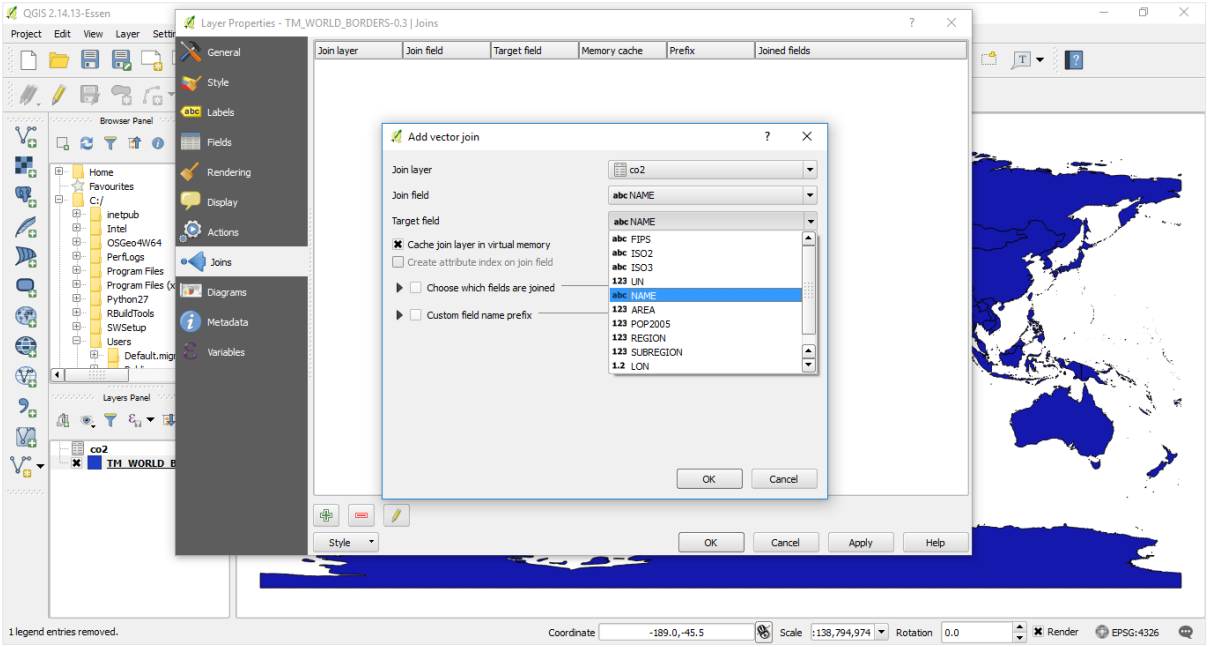
5. Creating cartograms

For the creation of a cartogram, which combines statistical information with geographic location, we will use both R Studio and QGIS. For this exercise we will use carbon-dioxide data from 2011.

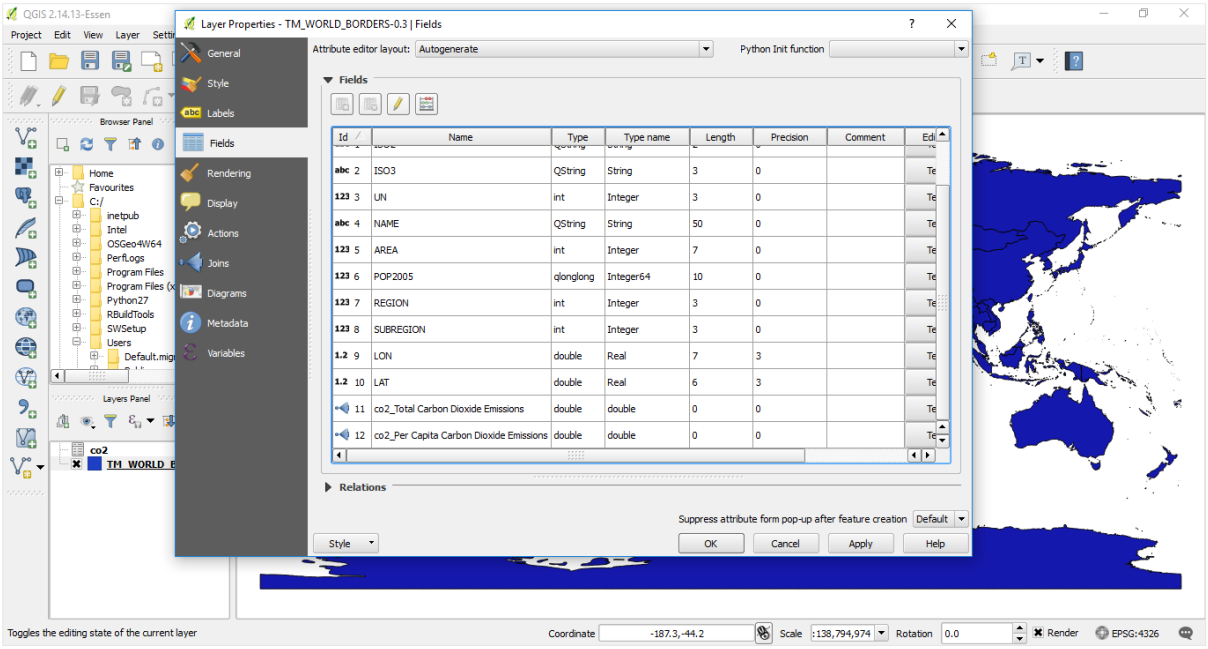
First, download the country shapefiles from here. Then open QGIS and load the TM_WORLD_BORDERS-0.3.shp file. Then open the .csv file which contains the CO_2 data. Make sure you tick the 'No geometry' option.



Then click on the shapefile layer and choose Properties. Click on Joins.

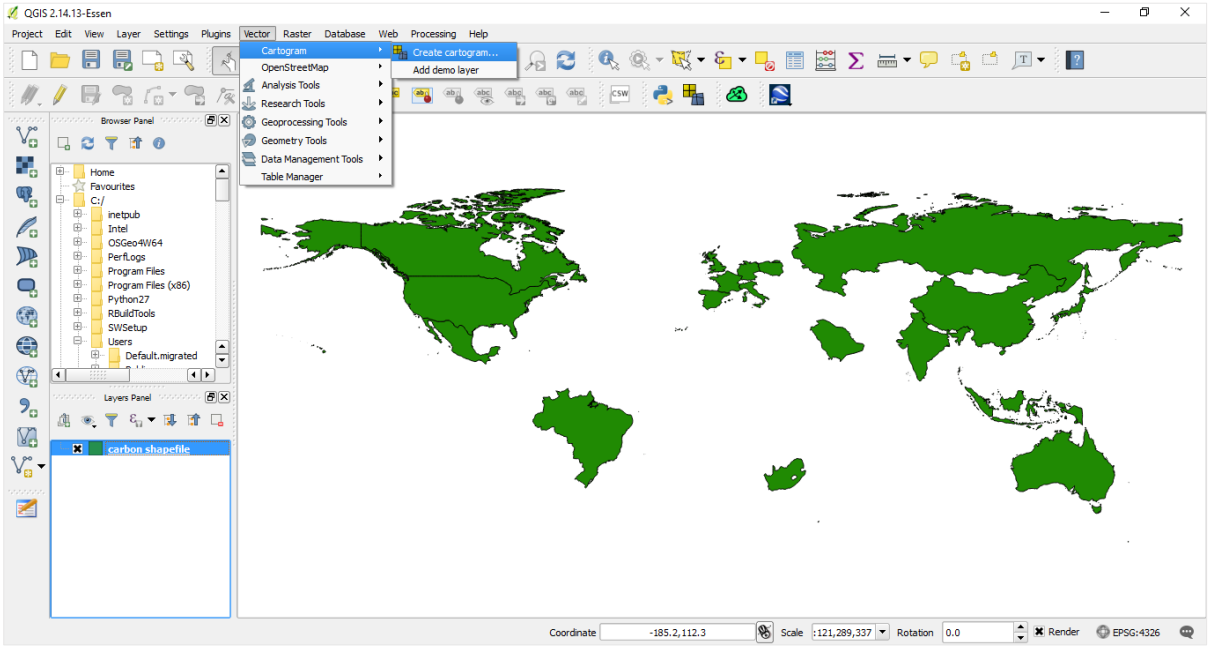


The Join layer is the .csv file, the Join layer is Name column from the .csv file and the Target field is also Name. This will allow the two files to be joined by the country names. Click Apply and now two extra columns should appear when you check the Fields.

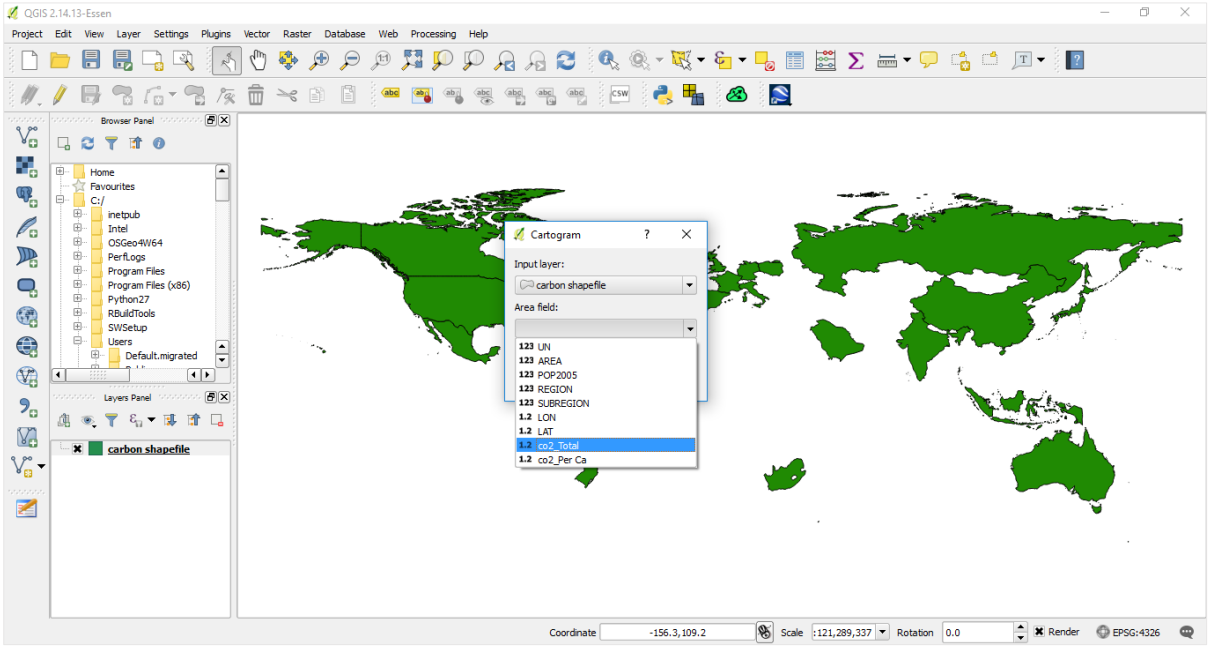


The next step will be a slightly tedious one, but the least complicated: Open the Attribute table and sort the data by either of the two new columns, so the 20 countries that we will use are on top. Now click on the Toggle Editing Mode (pencil icon) and using Ctrl and click highlight all the fields that we do not want to display. Once you have it all highlighted, press Delete (bin icon). You should be left with

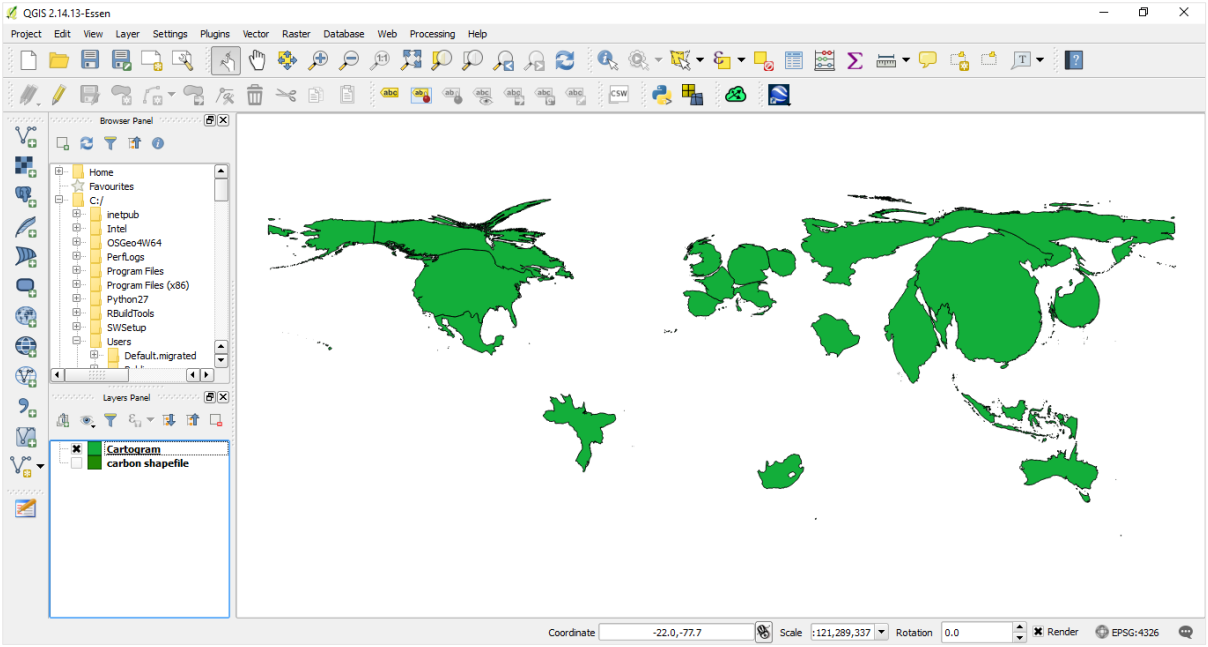
the 20 countries that we are interested in. Now save it as shapefile, remove everything else and open it.



Now click on Vector → Cartogram → Create cartogram.



We will use both columns so create two cartograms. Let's use the total emission first.

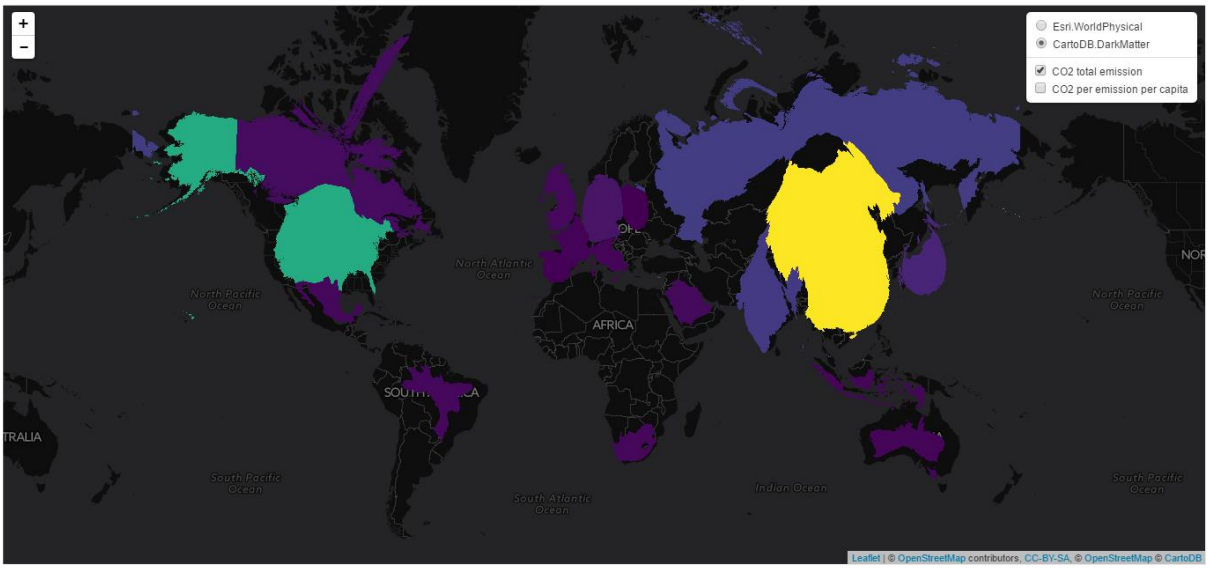


Here is the first cartogram based on the total carbon-dioxide emission. Save it as a shapefile. Create the second cartogram based on the carbon-dioxide emission per capita and save that too as a shapefile.

Now open R Studio.

```
library("maptools")
library("leaflet")

total <- readShapeSpatial("~/Course files/Extra/4. Cartogram/total
emission/total emission.shp")
percap <- readShapeSpatial("~/Course files/Extra/4. Cartogram/emission per
capita/emission per capita.shp")

pal <- colorNumeric("viridis", NULL)

leaflet(c) %>%
  addTiles() %>%
  addProviderTiles("Esri.WorldPhysical", group = "Esri.WorldPhysical") %>%
  addProviderTiles("CartoDB.DarkMatter", group = "CartoDB.DarkMatter") %>%
  setView(lng = 10, lat = 15, zoom = 3)  %>%
  addPolygons(data = total, stroke = FALSE, smoothFactor = 0.3, fillOpacity
= 1,
              fillColor = ~pal(co2_Total),
              popup = ~as.character(co2_Total),
              label=~as.character(NAME),
              group = "CO2 total emission") %>%
  addPolygons(data = percap, stroke = FALSE, smoothFactor = 0.3,
fillOpacity = 1,
              fillColor = ~pal(co2_Per.Ca),
              popup = ~as.character(co2_Per.Ca),
              label=~as.character(NAME),
              group = "CO2 per emission per capita") %>%
  addLayersControl(
                   baseGroups = c("Esri.WorldPhysical",
"CartoDB.DarkMatter"),
```

```
                    overlayGroups = c("CO2 total emission", "CO2 per
emission per capita"),
                    options = layersControlOptions(collapsed = TRUE),
                    position = 'topright')
```

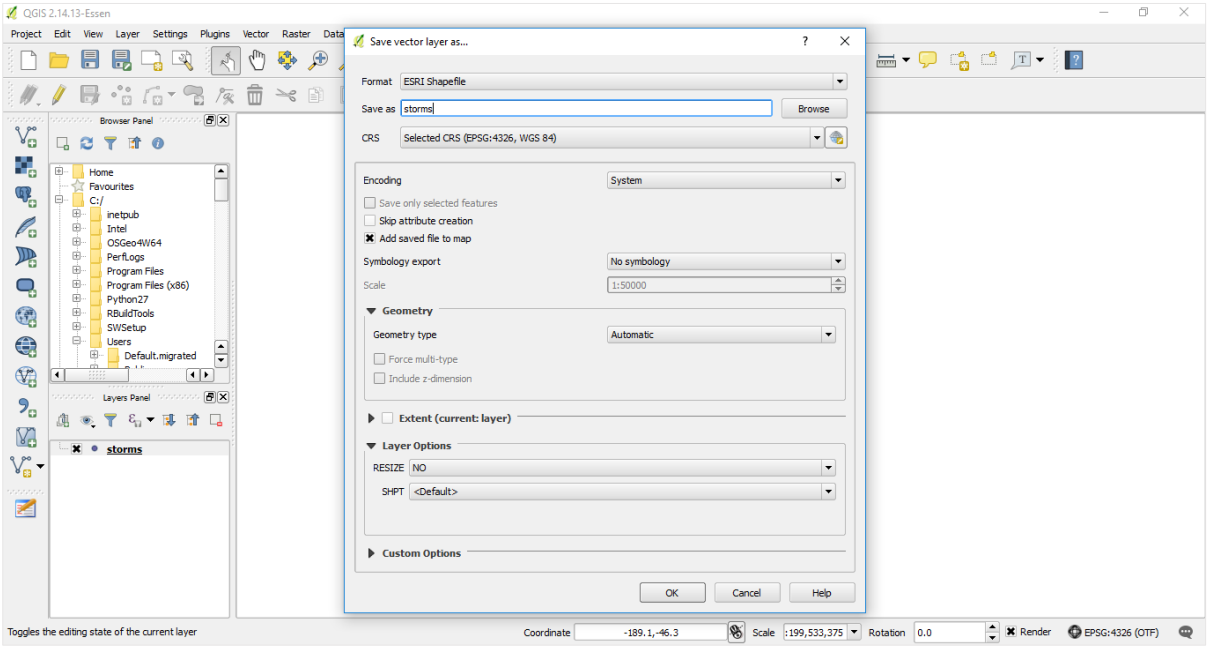


The results should appear in the Viewer pane.

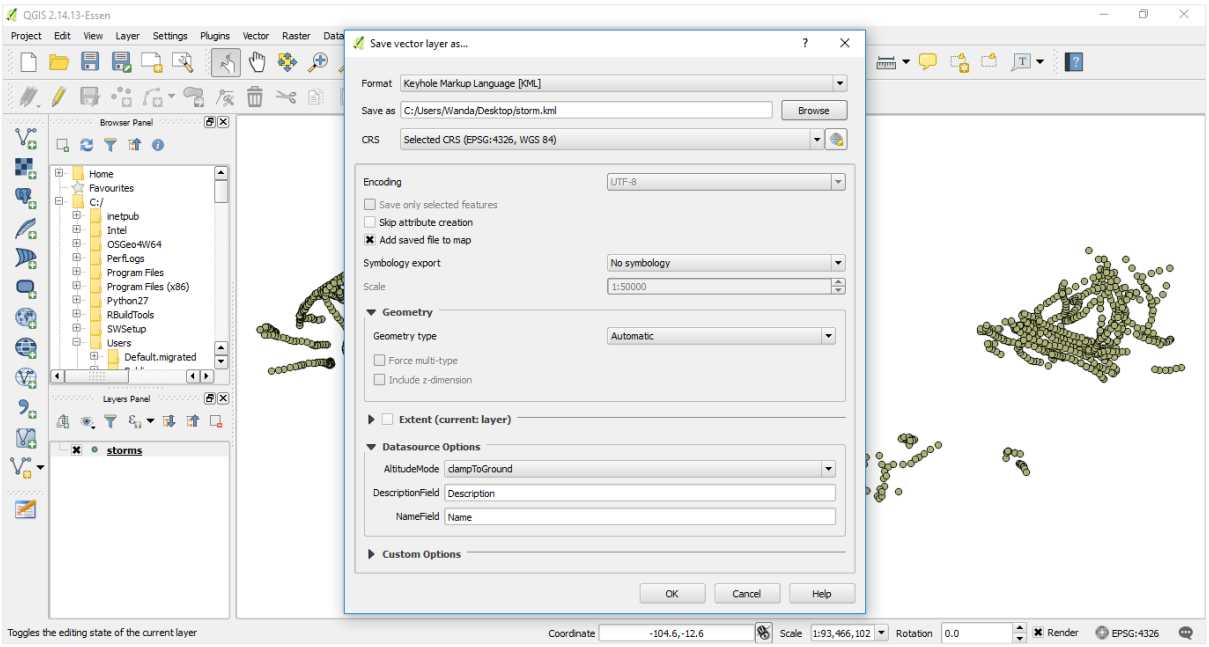
6. Using GoogleEarth and ArcGIS Earth

For both GoogleEarth and ArcGIS we will use the hurricane and tropical cyclones data.

Let's open QGIS first, add the .csv file and then save it as shapefile. Make sure that the CRS is set at EPSG:4326.

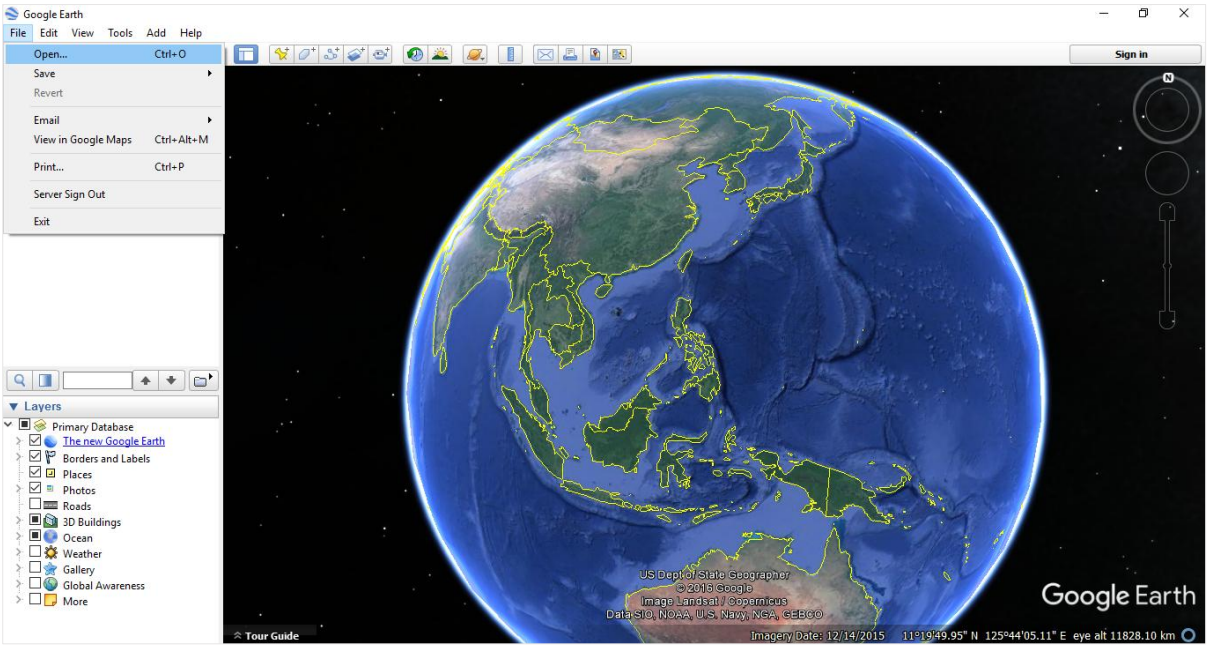


Now open the saved shapefile and save it as KML (Keyhole Markup Language).

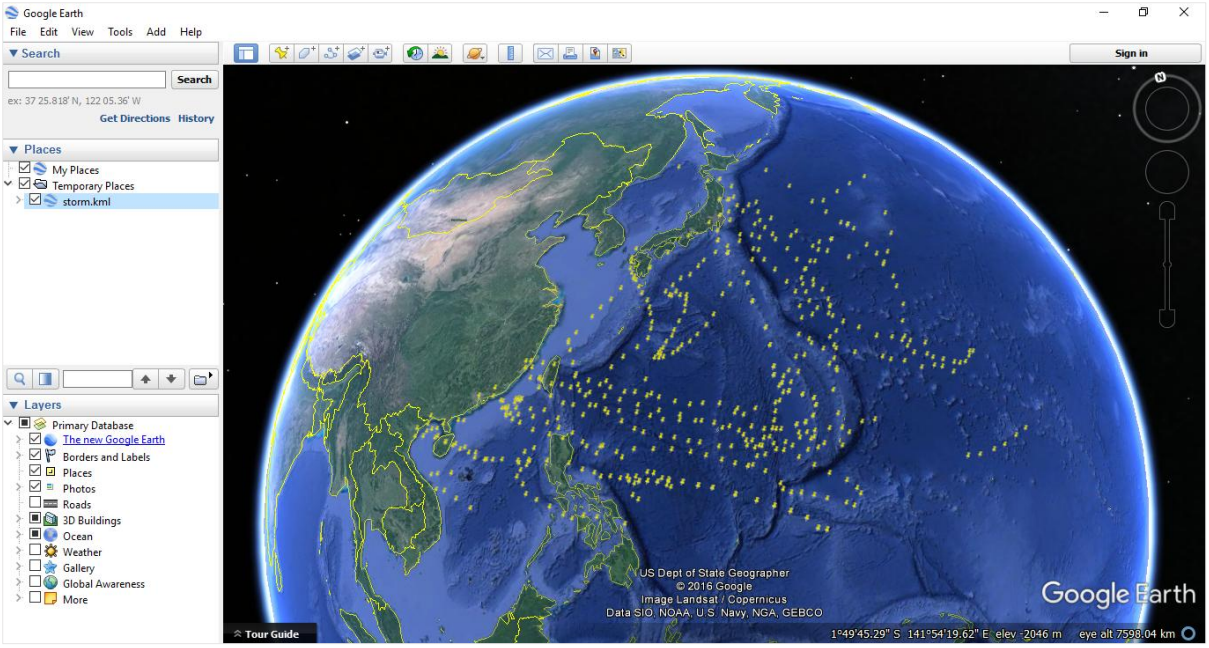


GoogleEarth

GoogleEarth is available to download here. Now open it and add the KML file using File → Open.

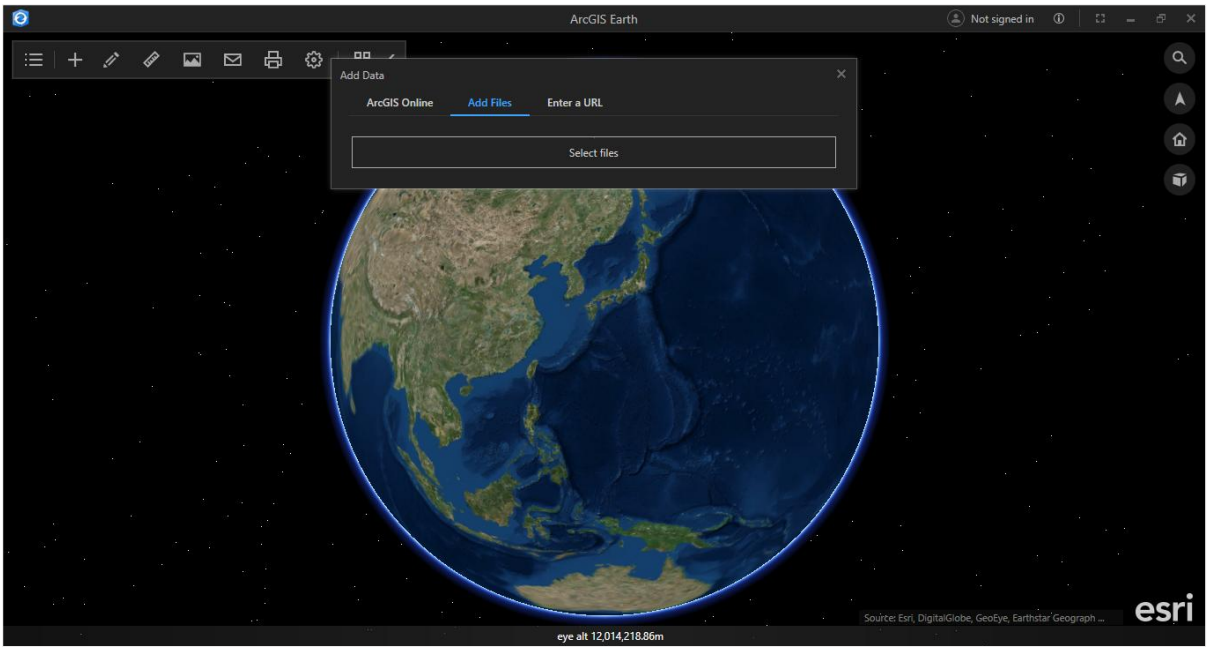


You may need to zoom in but the hurricane and tropical cyclone tracks should appear.

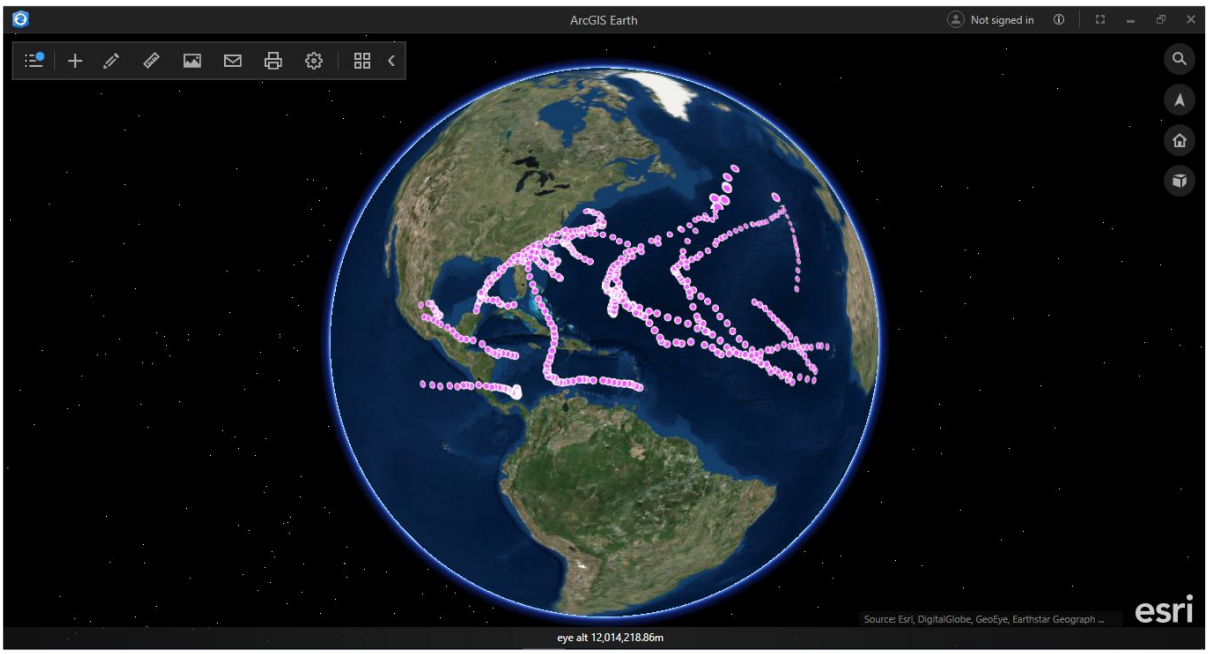


ArcGIS Earth

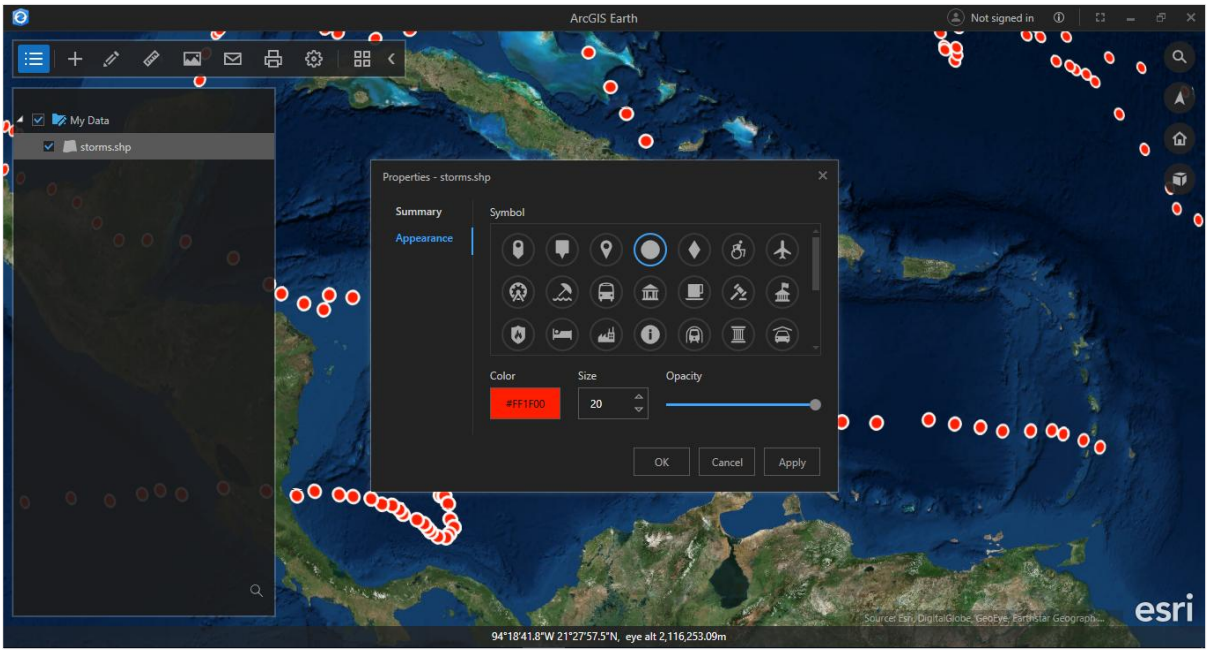
After downloading ArcGIS Earth open it. Click on Add data then Add Files.



Now select the created storm shapefile and add it.

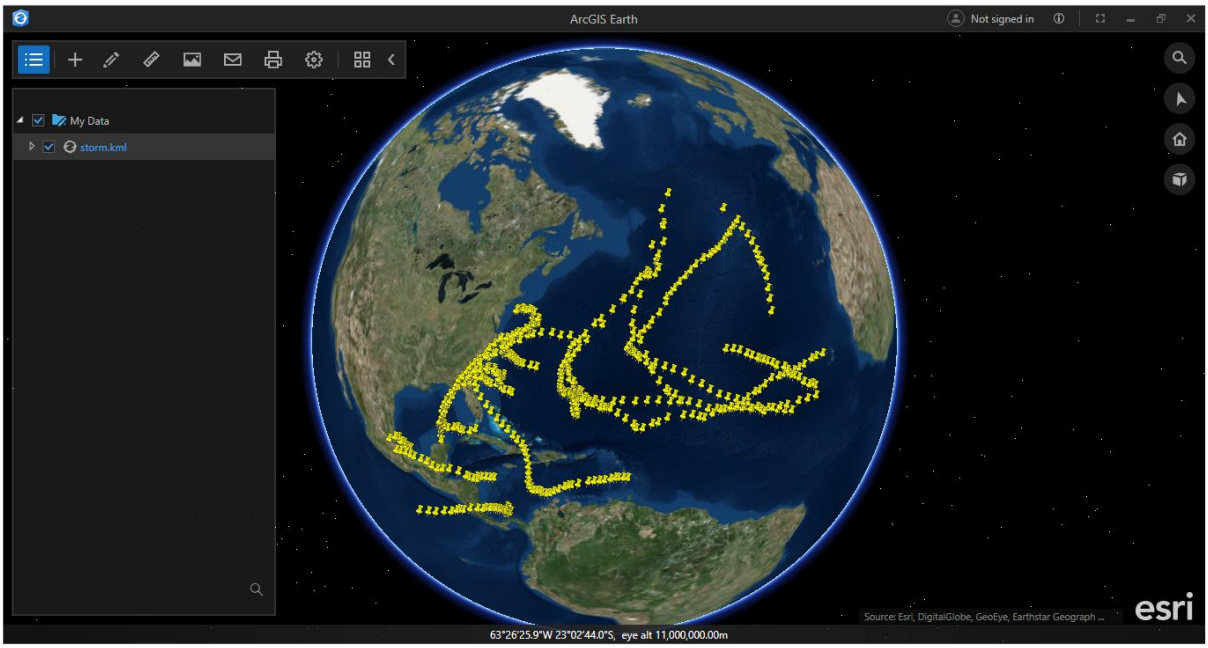


The storm tracks should appear. If you right click on the storm.shp layer and then properties, you will have the option to change the appearance.



You can also change the basemap.

You can also use the KML file.



7. Creating word cloud

```
library(RColorBrewer)
library(wordcloud)
library(tm)

display.brewer.all()
pal = brewer.pal(8, "Dark2")

windowsFonts(JP1 = windowsFont("xkcd"))

par(family = "JP1")

wordcloud(c("gis", "data", "rstudio", "qgis",
            "gif", "interactive", "mapping", "animated",
            "sst", "globe", "carthography", "hurricane",
            "climate", "cartogram"),
          freq = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14),
          col = pal, scale = c(3, 3), min.freq = 1)
```

References

Data:

Met Office Hadley Centre observations:

http://www.metoffice.gov.uk/hadobs/hadsst3/data/download.html (accessed on 12/02/2017)

2005 hurricane data

https://www.wunderground.com/hurricane/hurrarchive.asp (accessed on 05/05/2017)

2005 sea surface temperature:

https://giovanni.gsfc.nasa.gov/giovanni/ (accessed on 17/02/2017)

National Grid Reference shapefiles:

https://github.com/charlesroper/OSGB_Grids (accessed on 20/02/2017)

Natural Earth Data:

http://www.naturalearthdata.com/downloads/10m-physical-vectors/ (accessed on 02/03/2017)

Atmospheric baseline observatories:

https://www.esrl.noaa.gov/gmd/obop/ (accessed on 16/03/2017)

NASA global land surface temperature:

https://data.giss.nasa.gov/gistemp/ (accessed on 23/03/2017)

Atmospheric oxygen and carbon-dioxide:

http://scrippso2.ucsd.edu/index (accessed on 23/03/2017)

Bathymetry:

https://neo.sci.gsfc.nasa.gov/view.php?datasetId=GEBCO_BATHY (accessed on 24/03/2017)

Hurricanes and tropical cyclones:

https://www.wunderground.com/hurricane/at2016.asp (accessed on 27/04/2017)

Earthquakes:

https://earthquake.usgs.gov/earthquakes/browse/largest-world.php (accessed on 27/04/2017)

Tectonic plates:

https://github.com/fraxen/tectonicplates (accessed on 27/04/2017)

Carbon-dioxide emission:

http://www.ucsusa.org/global_warming/science_and_impacts/science/each-countrys-share-of-co2.html (accessed on 27/04/2017)

World Country Borders:

http://thematicmapping.org/downloads/world_borders.php (accessed on 27/04/2017)

Sea level trends:

https://tidesandcurrents.noaa.gov/sltrends/sltrends.html (accessed on 02/05/2017)

Coral reefs thermal history

https://coralreefwatch.noaa.gov/satellite/thermal_history/index.php (accessed on 02/05/2017)
Marine Ecoregions:
https://www.worldwildlife.org/publications/marine-ecoregions-of-the-world-a-bioregionalization-of-coastal-and-shelf-areas (accessed on 08/05/2017)
Geographic lines:
http://www.naturalearthdata.com/downloads/10m-physical-vectors/10m-geographic-lines/ (accessed on 10/05/2017)
Insured losses by state
http://www.iii.org/sites/default/files/docs/pdf/HurricaneKatrinaFactFile-0320101.pdf (accessed on 10/05/2017)

Books:

The R Book by Michael J. Crawley (2007)
R Graphics Cookbook by Winston Chang (2013)
R Data Visualization Cookbook by Atmajitsinh Gohil (2015)
Mapping – A Critical Introduction to Cartography and GIS by Jeremy W. Crampton (2010)
GIS for Dummies by Michael N. DeMers (2009)
The Dictionary of GIS Terminology by The Esri Press, edited by Heather Kennedy (2000)
Earth's Climate: Past and Future by William Ruddiman (2001)
Hurricane Climatology – A Modern Statistical Guide Using R by James B. Elsner and Thomas H. Jagger (2013)
Climatology – An Atmospheric Science by John J. Hidore and John E. Oliver (1993)

PDFs:

XKCD: An R Package for Plotting XKCD Graphs by Emilio Torres-Manzanera (2014), http://cran.r-project.org/web/packages/xkcd/vignettes/xkcdintro.pdf
Shapefiles: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

Publications:

Sea-Level Rise and Its Impact on Coastal Zones (2010) Robert J. Nicholls, Anny Cazenave, *Science* 18 Jun 2010: Vol. 328, Issue 5985, pp. 1517-1520, DOI: 10.1126/science.1185782
Climate Change, Human Impacts, and the Resilience of Coral Reefs (2003) T. P. Hughes, A. H. Baird, D. R. Bellwood, M. Card, S. R. Connolly, C. Folke, R. Grosberg, O. Hoegh-Guldberg, J. B. C. Jackson, J. Kleypas, J. M. Lough, P. Marshall, M. Nyström, S. R. Palumbi, J. M. Pandolfi, B. Rosen, J. Roughgarden, *Science* 15 Aug 2003:Vol. 301, Issue 5635, pp. 929-933, DOI: 10.1126/science.1085046

Websites:

https://www.giss.nasa.gov/tools/gprojector/help/projections.html (accessed 18/02/2017)

http://desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/what-are-map-projections.htm (accessed on 19/02/2017)

http://proj4.org/projections/index.html (accessed on 19/02/2017)

http://www.qgistutorials.com/en/docs/making_a_map.html (accessed on 20/02/2017)

http://www.asu.cas.cz/~bezdek/vyzkum/rotating_3d_globe/ (accessed on 19/03/2017)

http://digimap.edina.ac.uk/webhelp/os/data_information/os_data_issues/ng_converter.htm (accessed on 20/02/2017)

http://monde-geospatial.com (accessed on 23/03/2017)

https://image.slidesharecdn.com/neuralogsmiwellgeodetics211rev2-13047911609554-phpapp01-110507132018-phpapp01/95/neuralog-well-geodetics-20-728.jpg?cb=1304774478 (accessed on 26/03/2017)

Software & Non-CRAN Package

R: https://cran.r-project.org/

R Studio: https://www.rstudio.com/products/rstudio/download/

QGIS: http://www.qgis.org/en/site/forusers/download.html#

ImageMagick: http://ftp.icm.edu.pl/packages/ImageMagick/binaries/, ImageMagick-6.9.3-9-Q16-x64-dll.exe

gganimate: https://github.com/dgrtwo/gganimate

threejs: https://github.com/bwlewis/rthreejs

Atom: https://atom.io/

Leaflet.extras: https://bhaskarvk.github.io/leaflet.extras/

ArcGIS Earth: http://www.esri.com/software/arcgis-earth

Exifr: https://github.com/paleolimbot/exifr