Maps are used in a variety of fields to express data in an appealing and interpretive image. Data can be expressed into simplified complex patterns, and this data interpretation is generally lost if the data is written out into spread sheets. Maps can add vital context by incorporating many variables into and easy to read and relatable context. Maps are also very important in the information world because they can quickly allow the public to gain better insight to make better personal decisions. It's critical to have maps be effective, which means creating maps that can be understood by your target audience. Maps that need to be understood by children would very different from maps intended for geographers. Knowing what elements are required to enhance your data is key into making effective maps.

Some elements of a map that should be considered are polygon, points, lines, and text. With that in mind, we need to think about what in our map is required to really make an impact on our intended audience. Layout and formatting can also be used to enhance your data visually. It's important to think about the arrangement of these maps elements and how they will be placed to make a maximum impact on the target audience. Polygons, in a map, are shapes in a map such as counties. Lines are shapes that are not filled with any aspect, such as highways, streams, or roads. Points on a map are used to specify small specific areas such as city locations.

Using R to create maps is very user friendly. You can add and remove elements on a map with ease. It is also easy to reproduce the same maps for different data sets. It is important to be able to script the elements of a map so that it can used and interpreted by any user. It's relatively easy to e-mail R codes to colleagues, so the community aspect of R is very developed. Any code in R can be tweaked to make major enhancements with a stroke of a key.

Current solutions for creating maps are ArcGIS, QGIS, eSpatial, etc. The reason for using R is because it is a powerful and flexible tool. R can be used from calculating data sets to create graphs and maps with the same data set. R is also free, which makes it very easily accessible to anyone. Some other advantages of using R is that it has an interactive language, data structures, graphics availability, a developed community, repairing/replacing missing data, and the advantage of adding more tools through packages.

R is a free and open-source software development environment (IDE) that is used for computing statistical data and graphic in a programmable language. R requires different packages installed to allow for linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. CRAN, the comprehensive R archive network, which consists of a network of ftp and web servers that store identical versions of code and documentation for R. R is a scriptable language that allows the user to write out a code in which it will execute the command specified.

Packages in R create an ecosystem of different data analysis and interpretation. There are packages that are already programmed into a newly installed R, and we will be using several of those packages, as well as some developing packages.

The package `ggplot2` implements the grammar of graphics in R, as a

way to create code that make sense to the user: The grammar of

graphics is a term used to breaks up graphs into semantic components,

such as geometries and layers. Practically speaking, it allows (and

forces!) the user to focus on graph elements at a higher level of

abstraction, and how the data must be structured to achieve the

expected outcome.

While `ggplot2` is becoming the \*de facto\* standard for R graphs, it

does not handle spatial data specifically. The current standard for

spatial objects in R is defined as `Spatial` classes in the package

`sp`, but a new standard following the "simple feature" standard is

currently in development in the package `sf`. The development version

of the package `ggplot2` now allows the programmer to create layers

using simple features from the package `sf`. The combination of

`ggplot2` and `sf` therefore enable to create maps, using the grammar

of graphics, just as informative or visually appealing as any GIS

software.

First let’s start with creating a base map of the world using

` rworldmap`. This base map will then be extended with different

geographic or graph elements, as well as zoomed in to an area of

interest.

The package `rworldmap` provides a map of countries of the entire

world; a map with higher resolution is available in the package

`rworldxtra`. We use the function `getMap` to extract the world map

(the resolution can be set to `"low"`, if preferred):

The world map is available as a `SpatialPolygonsDataFrame`; we thus

convert it to a simple feature using `st\_as\_sf` (from package `sf`):

This call nicely introduces the structure of a `ggplot` call: The

first part `ggplot(data = world)` initiates the `ggplot` graph, and

indicates that the main data is stored in the `world` object. The line

ends up with a `+` sign, which indicate that the call is not complete

yet, and each subsequent line correspond to another layer or scale. In

this case, we use the `geom\_sf` function, which simply add a geometry

stored in a `sf` object. By default, all geometry functions use the

main data set up in `ggplot()`, but we will see later how to provide

additional data.

Note that layers are added one at a time in a `ggplot` call, so the

order of each layer is very important. All data will have to be in an

`sf` format to be used by `ggplot2`, and should be converted from `sp` to

`sf` if necessary.

We can add a title and subtitle to our map using the command

`ggtitle`. Make sure you add the quotation marks around your title and

subtitle names:

We can change the axis names to something more suitable. Depending on what your map displays, axis labels could be different.

In this example, we will be filling the polygons of the countries

with a green color (argument `fill`), and using black for the outline

of the countries (argument `color`):

`ggplot2` allows the use of more complex color schemes, such as a

gradient on one variable of the data. Let us see how to use the

population of each country (variable `POP\_EST` of the `world` object)

here:

We can set the coordinates of the map to "zoom" in the area we are

interested in mapping. To do this, we use `coord\_sf` to create our

coordinate settings, with limits on the x-axis (`xlim`), and on the

y-axis (`ylim`). Note that the limits are automatically expanded by a

fraction, so that selected data and axes don't overlap. It is possible

to set the zoom to the exact limits provided with `expand = FALSE`).

Many packages are available to create a scale bar on a map

(e.g. `prettymapr` or `vcd`"). The

package `legendMap` provides an easy-to-use function that allows to

add simultaneously the north symbol and a scale bar into the `ggplot`

map:

Five arguments need to be set manually: `lon`, `lat`, `distance\_lon`,

`distance\_lat`, and `distance\_legend`. The location of the scale bar

has to be specified in longitude/latitude in the `lon` and `lat`

arguments. The shaded distance inside the scale bar is controlled by

the `distance\_lon` argument.

In our case, 500 km is a good distance to use.

Additionally, it is possible to change the font size for the legend of

the scale bar (argument `legend\_size`, which defaults to 3). The North

arrow behind the "N" north symbol can also be adjusted for its length

(`arrow\_length`), its distance to the scale (`arrow\_distance`), or the

size the N north symbol itself (`arrow\_north\_size`, which defaults to

6).

Note that all distances (`distance\_lon`, `distance\_lat`,

`distance\_legend`, `arrow\_length`, `arrow\_distance`) are set to `"km"`

by default in `distance\_unit`; they can also be set to nautical miles

with "nm", or miles with "mi".

The `world` data set already contains country names and the

coordinates of the centroid of each country (among more

information). We can use this information to plot country names, using

`world` as a regular `data.frame` in `ggplot2`. We first check the

country name information:

The command `geom\_text` can be used to add a layer of text to a

graphic. The command `geom\_text` requires the data needed to enter the

country names, which is the same data as the world map. We can adjust

the size (argument `size`). By default, the text is centered on the

coordinates provided; we can adjust the alignment horizontally or

vertically using the arguments `hjust` and `vjust`, which can either

be a number between 0 (right/bottom) and 1 (top/left) or a character

("left", "middle", "right", "bottom", "center", "top"), or we can

offset the text horizontally or vertically with the argument `nudge\_x`

and `nudge\_y`.

We can also adjust the font of the text, for instance its color

(argument `colour`) or the type of font (`fontface`, which was set to

`"bold"`). Finally, the argument `check\_overlap` remove country names

that are overlapping each other.

Now to make the final touches. We want to edit the theme of the map to

make it more appealing. We added `theme\_classic` for a standard theme,

but there are many other themes that can be selected from. You can

experiment adding and removing themes to see which one best suits your

map. The argument `legend.position`, wasn't use since we don't require

a legend in our map. Otherwise, such variables such as `topright` or

`bottomleft` might be used. The `panel.grid.major` will allow us to

edit the grid lines in the map. We chose a gray color, with a line

type of dashed to break up the map borders from the grid lines. We

wanted to add some color to the map backgraound, which is the ocean

essentially, so we changed that with `panel.background`. We really

wanted a border around the map, so the `panel.border` argument was

used to add one. It will take some time to see elements of the theme

need to be changed or added to best meet your map needs. There are

many more aesthetics can be added to your custom theme, and they are

all listed in the ggplot2 pdf in Helpful Resources.

## Introduction

In the previous section, we presented general concepts with a map with

little information (country borders). The modular approach of

`ggplot2` allows to add additional layers in subsequent additions of

`geom\_sf`, for instance water bodies or terrain features, as will be

illustrated in this section.

To present fully reproducible examples, we will be using the

environmental layers found

on

[MapCruzin.com](http://www.mapcruzin.com/free-united-states-shapefiles/free-florida-arcgis-maps-shapefiles.htm). Following

the link, data packages derived

from [OpenStreetMap.org](http://www.openstreetmap.org/) can be freely

downloaded and used under

the

[Open Data Commons Open Database License](http://opendatacommons.org/licenses/odbl/) (ODbL).

We start by defining two study sites, according to their longitude and

latitude. The information is stored in a `data.frame`, with two

variables `longitude` and `latitude`:

The easiest way to add point coordinates is with the general-purpose

function `geom\_point`, which works on any X/Y coordinates,

of regular data points (i.e. not geographic). As such, we can adjust

all characteristics of points (e.g. color of the outline and the

filling, shape, size, etc.), for all points, or using grouping from

the data (i.e defining their "aesthetics"). In this example, we add

the two points as diamonds (`shape = 23`), filled in dark red (`fill =

"darkred"`) and of bigger size (`size = 6`):

Many layers can be added using `geom\_sf`. Many times, if a layer doesn't display correctly, it might have to be converted as an `sf` object.

Very similar to the previous map, we first need to retrieve a map from the package `rworldmap` and `rworldxtra`. As you have already figured out, we normally start with a very broad map, and then narrow it down to what we are interested in showing.

Very similar to the previous map, we first need to retrieve a map from the package `rworldmap` and `rworldxtra`. As you have already figured out, we normally start with a very broad map, and then narrow it down to what we are interested in displaying.

In this map with coordinate points we require more detail, since the

map will have no landscape features.