GIS in R: Intro to Coordinate Reference Systems

This lesson covers the key spatial attributes that are needed to work with spatial data including: Coordinate Reference Systems (CRS), Extent and spatial resolution.

Learning Objectives

After completing this tutorial, you will be able to:

- Be able to describe what a Coordinate Reference System (`CRS`) is
- Be able to list the steps associated with working with 2 datasets stored using difference coordinate reference systems.
- Understand the basic differences between a geographic and a projected `CRS`.
- Become familiar with the Universal Trans Mercator (UTM) and Geographic (WGS84) CRSs

What you need

You will need a computer with internet access to complete this lesson and the data for week 4 of the course.

Download Week 4 Data (~500 MB){:data-proofer-ignore=" .btn }

Intro to coordinate reference systems

The short video below highlights how map projections can make continents look proportionally larger or smaller than they actually are.

What is a Coordinate Reference System

To define the location of something we often use a coordinate system. This system consists of an X and a Y value located within a 2 (or more) -dimensional space.

```
<a href="http://open.senecac.on.ca/clea/label/projectImages/15_276_xy-grid.jpg">
<img src="http://open.senecac.on.ca/clea/label/projectImages/15_276_xy-grid.jpg"></a>
<figcaption> We use coordinate systems with X, Y (and sometimes Z axes) to
define the location of objects in space.
Source: http://open.senecac.on.ca
</figcaption>
```

While the above coordinate system is 2-dimensional, we live on a 3-dimensional earth that happens to be "round". To define the location of objects on the earth, which is round, we need a coordinate system that adapts to the Earth's shape. When we make maps on paper or on a flat computer screen, we move from a 3-Dimensional space (the globe) to a 2-Dimensional space (our computer screens or a piece of paper). The components of the CRS define how the "flattening" of data that exists in a 3-D globe space. The CRS also defines the the coordinate system itself.

```
<a href="http://ayresriverblog.com/wp-content/uploads/2011/05/image.png">
<img src="http://ayresriverblog.com/wp-content/uploads/2011/05/image.png"></a>
<figcaption>A CRS defines the translation between a location on the round earth
and that same location, on a flattened, 2 dimensional coordinate system.
Source: http://ayresriverblog.com
</figcaption>
```

> A coordinate reference system (CRS) is a coordinate-based local, regional or global system used to locate geographical entities. – Wikipedia

The Components of a CRS

The coordinate reference system is made up of several key components:

- **Coordinate System:** the X, Y grid upon which our data is overlayed and how we define where a point is located in space.
- **Horizontal and vertical units:** The units used to define the grid along the x, y (and z) axis.
- **Datum:** A modeled version of the shape of the earth which defines the origin used to place the coordinate system in space. We will explain this further, below.
- **Projection Information:** the mathematical equation used to flatten objects that are on a round surface (e.g. the earth) so we can view them on a flat surface (e.g. our computer screens or a paper map).

Why CRS is Important

It is important to understand the coordinate system that your data uses - particularly if you are working with different data stored in different coordinate systems. If you have data from the same location that are stored in different coordinate reference systems, **they will not line up in any GIS or other program** unless you have a program like ArcGIS or QGIS that supports **projection on the fly**. Even if you work in a tool that supports projection on the fly, you will want to all of your data in the same projection for performing analysis and processing tasks.

Data Tip: Spatialreference.org provides an excellent online library of CRS information. {: .notice}

Coordinate System & Units

We can define a spatial location, such as a plot location, using an x- and a y-value - similar to our cartesian coordinate system displayed in the figure, above.

For example, the map below, generated in `R` with `ggplot2` shows all of the continents in the world, in a `Geographic` Coordinate Reference System. The units are Degrees and the coordinate system itself is **latitude** and **longitude** with the `origin` being the location where the equator meets the central meridian on the globe (0,0).

```
# load libraries
library(rgdal)
library(ggplot2)
library(rgeos)
library(raster)

# set your working directory
# setwd("~/Documents/data")
```

```
# read shapefile
worldBound <- readOGR(dsn="data/week4/global/ne_110m_land",
                      layer="ne_110m_land")
## OGR data source with driver: ESRI Shapefile
## Source: "data/week4/global/ne_110m_land", layer: "ne_110m_land"
## with 127 features
## It has 2 fields
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
```

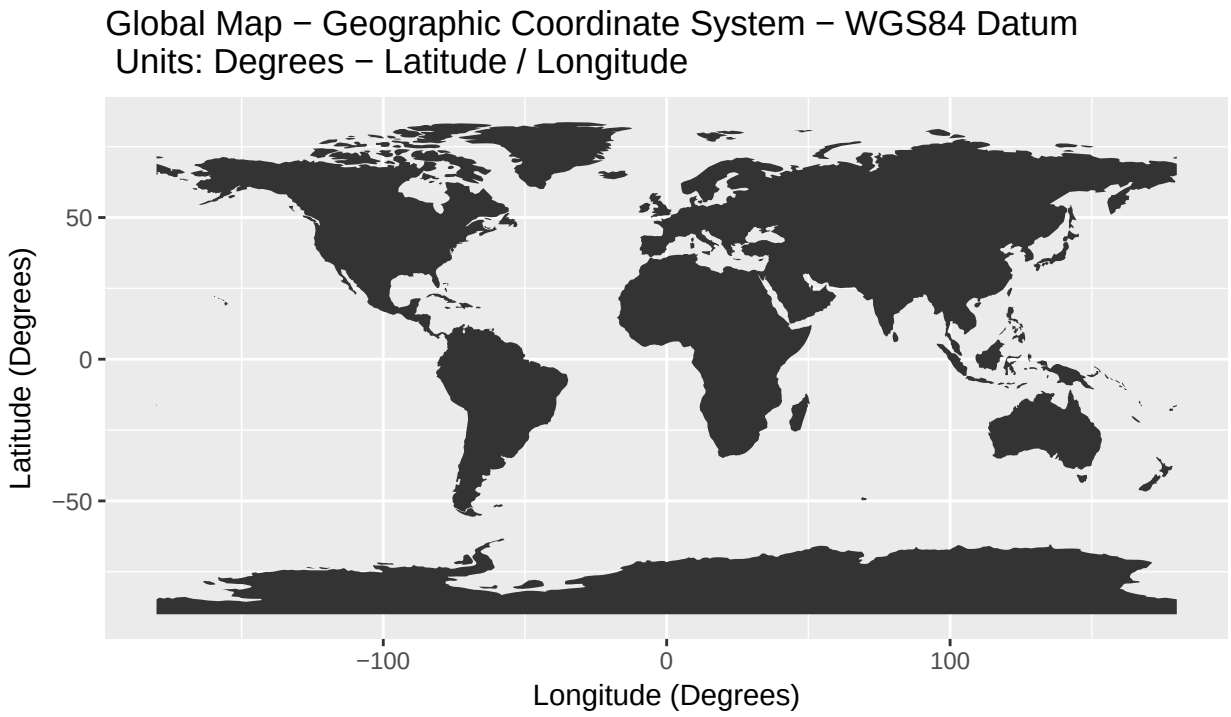

Figure 1: world map plot

```
# convert to dataframe
worldBound_df <- fortify(worldBound)
## Regions defined for each Polygons

# plot map
worldMap <- ggplot(worldBound_df, aes(long,lat, group=group)) +
  geom_polygon() +
  xlab("Longitude (Degrees)") + ylab("Latitude (Degrees)") +
  coord_equal() +
  ggtitle("Global Map - Geographic Coordinate System - WGS84 Datum\n Units: Degrees - Latitude / Longitu

worldMap
```

We can add three coordinate locations to our map. Note that the UNITS are in decimal **degrees** (latitude, longitude):

- Boulder, Colorado: 40.0274, -105.2519
- Oslo, Norway: 59.9500, 10.7500
- Mallorca, Spain: 39.6167, 2.9833

Let's create a second map with the locations overlayed on top of the continental boundary layer.

```
# define locations of Boulder, CO and Oslo, Norway
# store them in a data.frame format
loc.df <- data.frame(lon=c(-105.2519, 10.7500, 2.9833),
                     lat=c(40.0274, 59.9500, 39.6167))

# only needed if the above is a spatial points object
# loc.df <- fortify(loc)
```

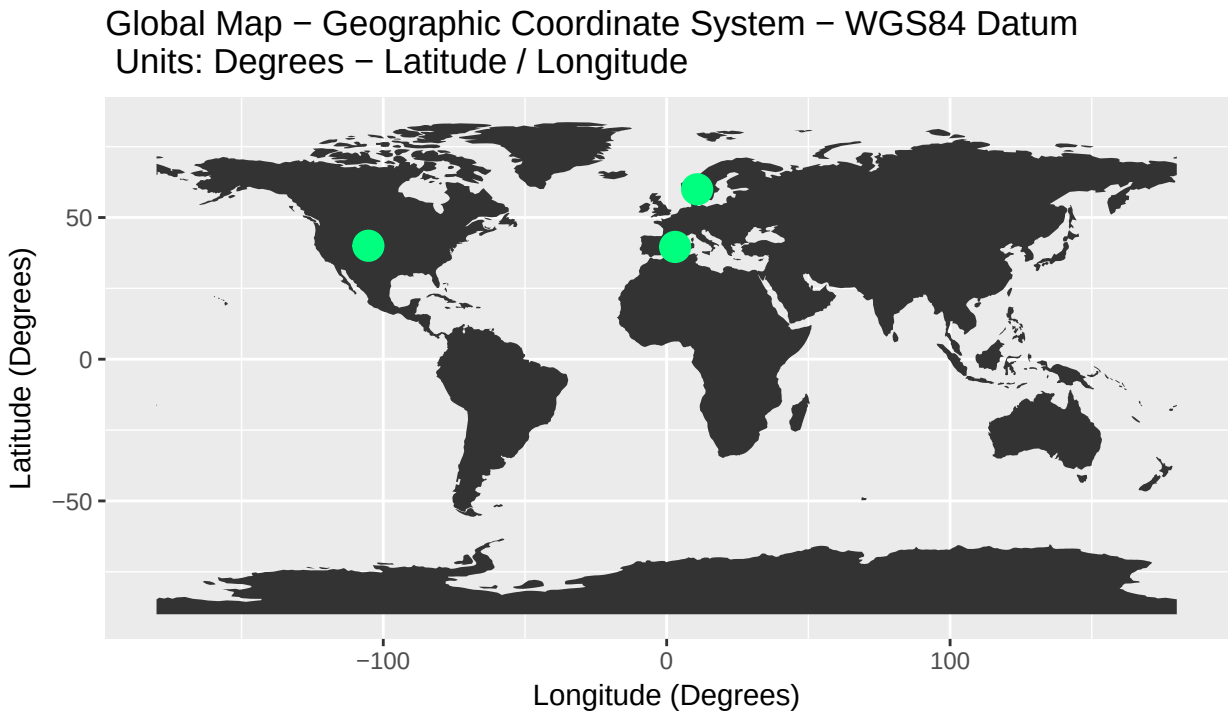

Figure 2: Map plotted using geographic projection with location points added.

```
# add a point to the map
mapLocations <- worldMap + geom_point(data=loc.df,
                        aes(x=lon, y=lat, group=NULL),
                      colour = "springgreen",
                      size=5)

mapLocations + theme(legend.position="none")
```

Geographic CRS - The Good & The Less Good

Geographic coordinate systems in decimal degrees are helpful when we need to locate places on the Earth. However, latitude and longitude locations are not located using uniform measurement units. Thus, geographic CRSs are not ideal for measuring distance. This is why other projected `CRS` have been developed.

```
<a href="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/LatLongfromGlobeCenter-ESRI.
<img src="{{ site.baseurl }}/images/course-materials/earth-analytics/week-4/LatLongfromGlobeCenter-ESRI
<figcaption>A geographic coordinate system locates latitude and longitude
location using angles. Thus the spacing of each line of latitude moving north
and south is not uniform.
Source: ESRI
</figcaption>
```

Projected CRS - Robinson

We can view the same data above, in another CRS - `Robinson`. `Robinson` is a **projected** `CRS`. Notice that the country boundaries on the map - have a different shape compared to the map that we created above in

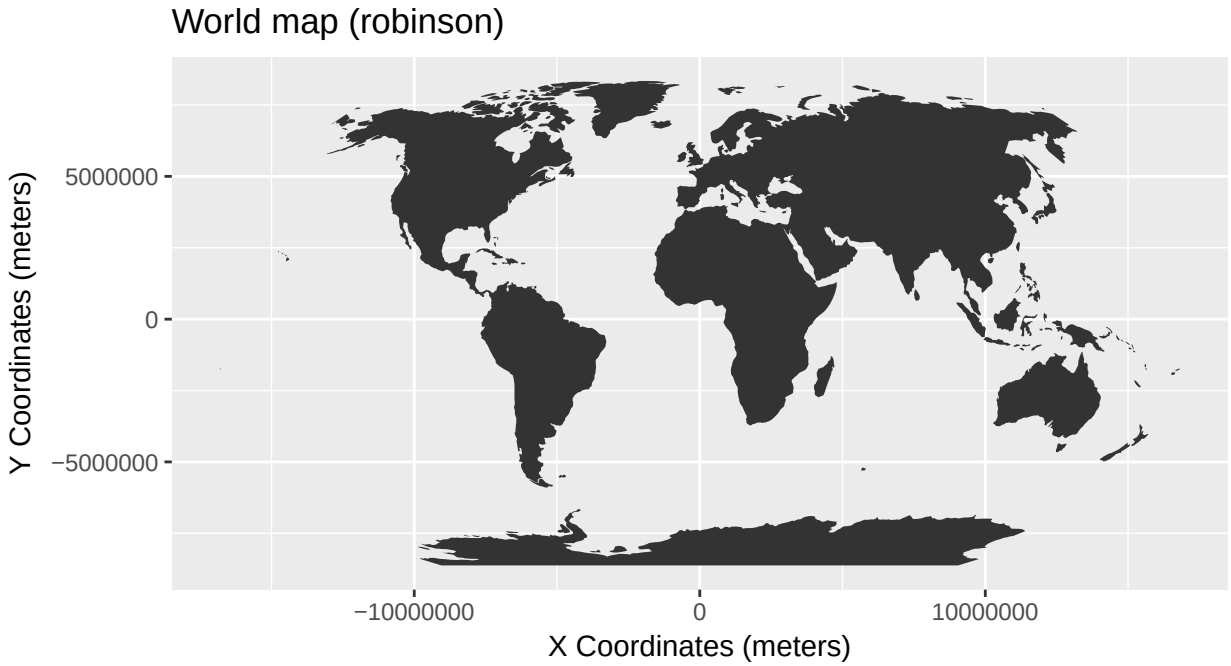

Figure 3: Map reprojected to robinson projection.

the CRS: **Geographic lat/long WGS84**.

```
# reproject from longlat to robinson
worldBound_robin <- spTransform(worldBound,
                                CRS("+proj=robin"))
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files

worldBound_df_robin <- fortify(worldBound_robin)
## Regions defined for each Polygons

# force R to plot x and y values without rounding digits
options(scipen=100)

robMap <- ggplot(worldBound_df_robin, aes(long,lat, group=group)) +
  geom_polygon() +
  labs(title="World map (robinson)") +
  xlab("X Coordinates (meters)") + ylab("Y Coordinates (meters)") +
  coord_equal()

robMap
```

Now what happens if you try to add the same Lat / Long coordinate locations that we used above, to our map, with the CRS of Robinsons?

```
# add a point to the map
newMap <- robMap + geom_point(data=loc.df,
                      aes(x=lon, y=lat, group=NULL),
                      colour = "springgreen",
                      size=5)

newMap + theme(legend.position="none")
```

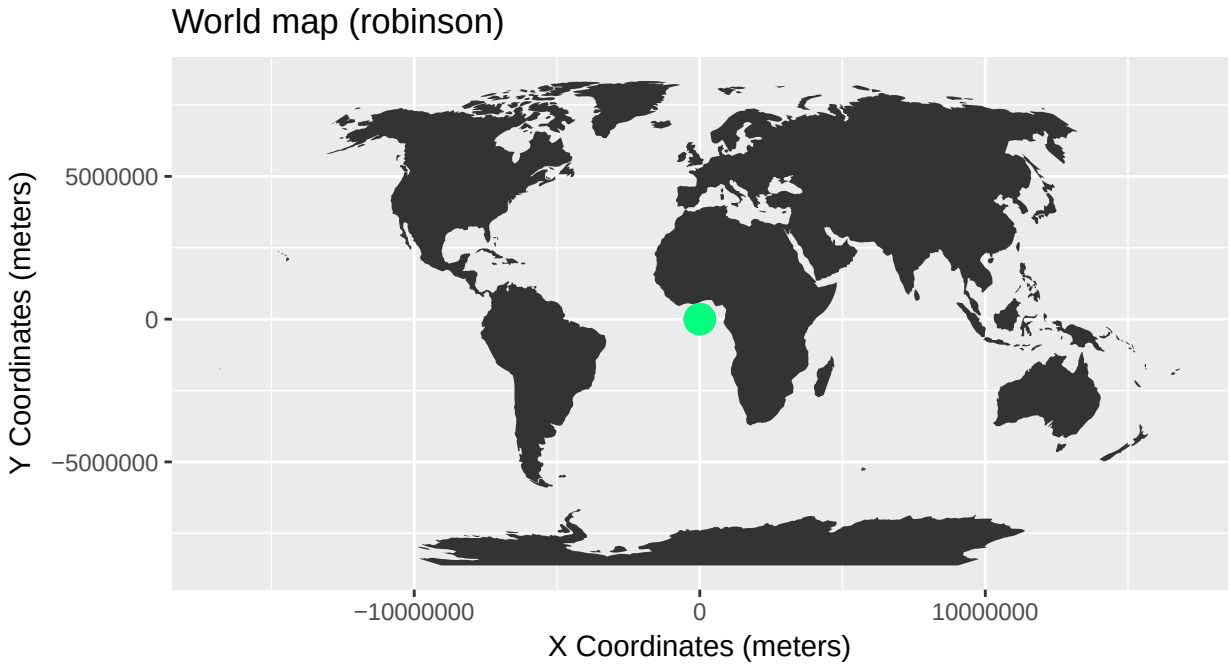

Figure 4: map with point locations added - robinson projection.

Notice above that when we try to add lat/long coordinates in degrees, to a map in a different `CRS`, that the points are not in the correct location. We need to first convert the points to the new projection - a process often referred to as **reprojection** but performed by the `spTransform()` function in `R`.

```
# define locations of Boulder, CO and Oslo, Norway
loc.df
##         lon     lat
## 1 -105.2519 40.0274
## 2   10.7500 59.9500
## 3    2.9833 39.6167

# convert to spatial Points data frame
loc.spdf <- SpatialPointsDataFrame(coords = loc.df, data=loc.df,
                           proj4string=crs(worldBound))

loc.spdf
## class       : SpatialPointsDataFrame
## features    : 3
## extent      : -105.2519, 10.75, 39.6167, 59.95  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## variables   : 2
## names       :       lon,     lat
## min values  : -105.2519, 39.6167
## max values  :     10.75,   59.95
# reproject data to Robinson
loc.spdf.rob <- spTransform(loc.spdf, CRSobj = CRS("+proj=robin"))
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
## NOTE: rgdal::checkCRSArgs: no proj_defs.dat in PROJ.4 shared files
```

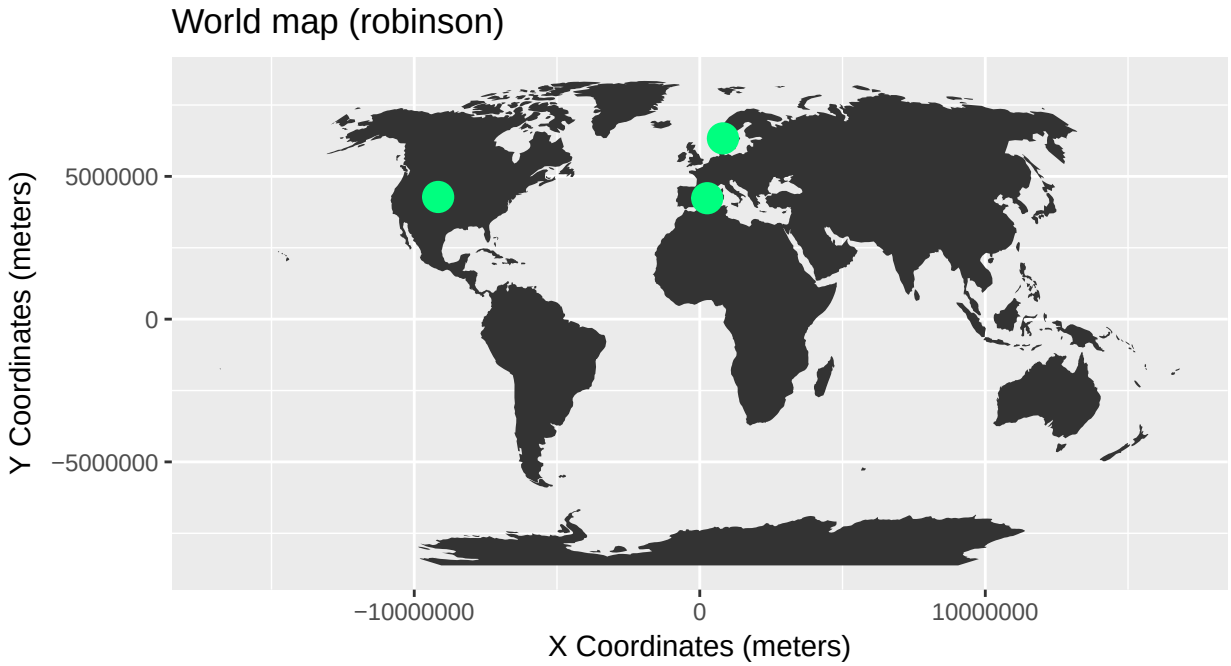

Figure 5: Map plotted using robinson projection.

```
loc.rob.df <- as.data.frame(cbind(loc.spdf.rob$lon, loc.spdf.rob$lat))
# rename each column
names(loc.rob.df ) <- c("X","Y")

# convert spatial object to a data.frame for ggplot
loc.rob <- fortify(loc.rob.df)

# notice the coordinate system in the Robinson projection (CRS) is DIFFERENT
# from the coordinate values for the same locations in a geographic CRS.
loc.rob
##            X       Y
## 1 -9162993.5 4279263
## 2   811462.5 6331141
## 3   260256.6 4235608

# add a point to the map
newMap <- robMap + geom_point(data=loc.rob,
                      aes(x=X, y=Y, group=NULL),
                      colour = "springgreen",
                      size=5)

newMap + theme(legend.position="none")
```

Compare Maps

Both of the plots above look visually different and also use a different coordinate system. Let's look at both, side by side, with the actual **graticules** or latitude and longitude lines rendered on the map.

```r
# this is not taught in the lesson but use it to display ggplot next to each other
require(gridExtra)

# turn off axis elements in ggplot for better visual comparison
newTheme <- list(theme(line = element_blank(),
      axis.text.x = element_blank(),
      axis.text.y = element_blank(),
      axis.ticks = element_blank(), # turn off ticks
      axis.title.x = element_blank(), # turn off titles
      axis.title.y = element_blank(),
      legend.position="none")) # turn off legend

## add graticules
graticule <- readOGR("data/week4/global/ne_110m_graticules_all",
                     layer="ne_110m_graticules_15")
# convert spatial object into a ggplot ready, data.frame
graticule_df <- fortify(graticule)

bbox <- readOGR("data/week4/global/ne_110m_graticules_all", layer="ne_110m_wgs84_bounding_box")
bbox_df<- fortify(bbox)


latLongMap <- ggplot(bbox_df, aes(long,lat, group=group)) +
  geom_polygon(fill="white") +
  geom_polygon(data=worldBound_df, aes(long,lat, group=group, fill=hole)) +
  geom_path(data=graticule_df, aes(long, lat, group=group), linetype="dashed", color="grey70") +
  labs(title="World Map - Geographic (long/lat degrees)") +
  coord_equal() + newTheme +
  scale_fill_manual(values=c("black", "white"), guide="none") # change colors & remove legend

latLongMap <- latLongMap + geom_point(data=loc.df,
                      aes(x=lon, y=lat, group=NULL),
                      colour="springgreen",
                      size=5)

# reproject grat into robinson
graticule_robin <- spTransform(graticule, CRS("+proj=robin"))  # reproject graticule
grat_df_robin <- fortify(graticule_robin)
bbox_robin <- spTransform(bbox, CRS("+proj=robin"))  # reproject bounding box
bbox_robin_df <- fortify(bbox_robin)

# plot using robinson

finalRobMap <- ggplot(bbox_robin_df, aes(long,lat, group=group)) +
  geom_polygon(fill="white") +
  geom_polygon(data=worldBound_df_robin, aes(long,lat, group=group, fill=hole)) +
  geom_path(data=grat_df_robin, aes(long, lat, group=group), linetype="dashed", color="grey70") +
  labs(title="World Map Projected - Robinson (Meters)") +
  coord_equal() + newTheme +
  scale_fill_manual(values=c("black", "white"), guide="none") # change colors & remove legend

# add a point to the map
finalRobMap <- finalRobMap + geom_point(data=loc.rob,
```

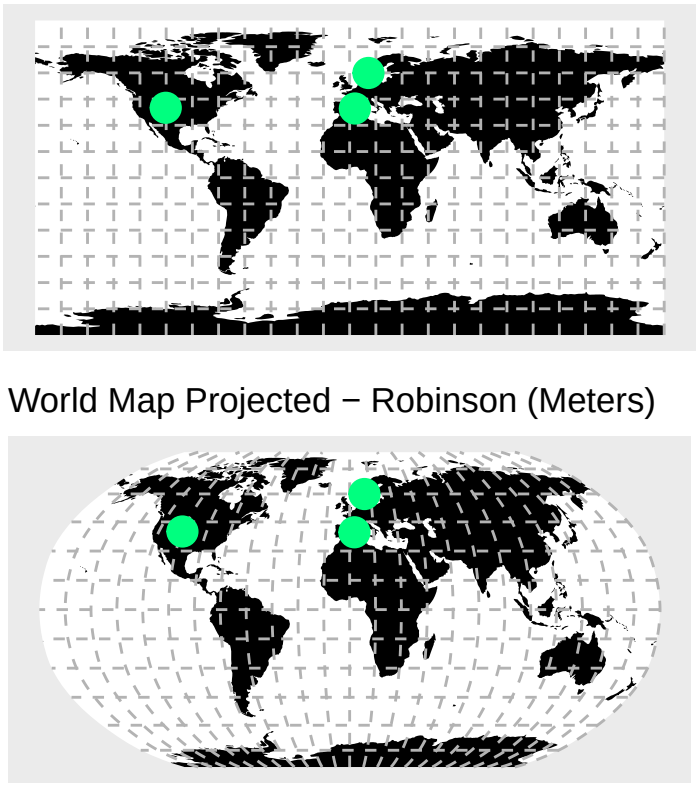

Figure 6: plots in different projections, side by side.

```
                    aes(x=X, y=Y, group=NULL),
                    colour="springgreen",
                    size=5)


# display side by side
grid.arrange(latLongMap, finalRobMap)
```

Why Multiple CRS?

You may be wondering, why bother with different CRSs if it makes our analysis more complicated? Well, each `CRS` is optimized to best represent:

- Shape and/or
- Scale / distance and/or
- Area

of features in the data. And no one CRS is great at optimizing shape, distance AND area. Some CRSs are optimized for shape, some distance, some area. Some CRSs are also optimized for particular regions - for instance the United States, or Europe. Discussing `CRS` as it optimizes shape, distance and area is beyond the scope of this tutorial, but it's important to understand that the `CRS` that you chose for your data, will impact working with the data!

Challenge

1. Compare the maps of the globe above. What do you notice about the shape of the various countries. Are there any signs of distortion in certain areas on either map? Which one is better?
2. Look at the image below - which depicts maps of the United States in 4 different `CRS`s. What visual differences do you notice in each map? Look up each projection online, what elements (shape,area or distance) does each projection used in the graphic below optimize?

```
<a href="https://source.opennews.org/media/cache/b9/4f/b94f663c79024f0048ae7b4f88060cb5.jpg">
<img src="https://source.opennews.org/media/cache/b9/4f/b94f663c79024f0048ae7b4f88060cb5.jpg">
</a>

<figcaption>Maps of the United States in different CRS including Mercator
(upper left), Albers equal area (lower left), UTM (Upper RIGHT) and
WGS84 Geographic (Lower RIGHT).
Notice the differences in shape and orientation associated with each
CRS. These differences are a direct result of the
calculations used to "flatten" the data onto a two dimensional map.
Source: opennews.org</figcaption>
```

Geographic vs Projected CRS

The above maps provide examples of the two main types of coordinate systems:

1. **Geographic coordinate systems:** coordinate systems that span the entire globe (e.g. latitude / longitude).
2. **Projected coordinate Systems:** coordinate systems that are localized to minimize visual distortion in a particular region (e.g. Robinson, UTM, State Plane)

In the next tutorial, we will discuss the differences between these CRSs in more detail. Feel free to skip over this section and come back to it with fresh eyes if the concept of a `CRS` is becoming too complex. It's easisest to take on in bite sized pieces!

Additional Resources

- Read more on coordinate systems in the QGIS documentation.
- The Relationship Between Raster Resolution, Spatial extent & Number of Pixels - in R - NEON
- For more on types of projections, visit ESRI's ArcGIS reference on projection types..
- Read more about choosing a projection/datum.