Create a World Map Using R

Package rnaturalearth

2D World Map

1. Import the package
2. Use the ne_countries to get the world data.
3. Use the geom_sf() function to create the world map.

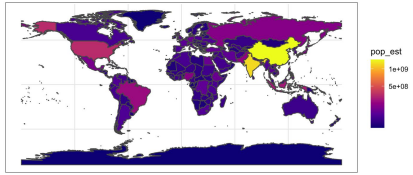
The above steps in code:

```
world <- ne_countries(scale = "medium", returnclass = "sf")
ggplot(data = world) +
  geom_sf(fill = "antiquewhite", color = "darkgray") +
  theme(panel.grid.major = element_line(color = gray(.5), linetype = "dashed", size = 0.1),
panel.background = element_rect(fill = "aliceblue"))
```

And we can get…

Change the color of each country by some data:

```
ggplot(data = world) +
    geom_sf(aes(fill = pop_est)) +
    scale_fill_viridis_c(option = "plasma", trans = "sqrt")
```



3D Globe Projection World Map

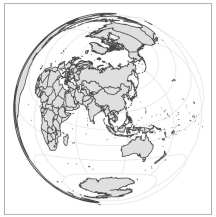
1. Import the package
2. Use the geom_sf() function to create the world map.
3. Use cood_sf() to determine the position

The above steps in code:

```
ggplot(data = world) +
    geom_sf() +
    coord_sf(crs = "+proj=laea +lat_0=25 +lon_0=100 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=m +no_defs ")
```

The angle is determined by the parameter "crs" in coord_sf. There are two common options: PROJ4 string or EPSG code.

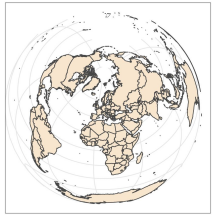
And we can get…



The example above is the projection from Asia.
If you change lat_0=52 and lon_0=10, you will get the projection from Europe (this is the same as using crs = st_crs(3035)).

More information about PROJ4 String and EPGS code can be found at: https://proj.org/usage/projections.html and https://epsg.io

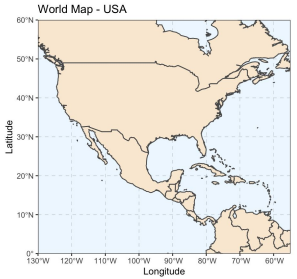
```
ggplot(data = world) +
    geom_sf(fill = "antiquewhite") +
    coord_sf(crs = st_crs(3035))
```



Zoom in 2D World Map for a Specific Area

1. Choose the area you want to plot
2. Specific the longitude and latitude

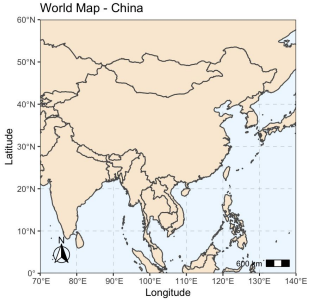
```
ggplot(data = world) +
  geom_sf(fill = "antiquewhite") +
  coord_sf(xlim = c(-130, -55), ylim = c(0, 60.0), expand = FALSE) + xlab("Longitude") + ylab("Latitude") +
  ggtitle("World Map - USA") +
  theme(panel.grid.major = element_line(color = gray(.5), linetype = "dashed", size = 0.1), panel.background = element_rect(fill = "aliceblue"))
```



Adding Scale Bar and North Arrow

1. Import the package ggspatial
2. Plot the map as above
3. Add annotation_scale()
4. Add annotation_north_arrow()

```
library("ggspatial")
ggplot(data = world) +
  geom_sf(fill= "antiquewhite") +
  coord_sf(xlim = c(70, 140), ylim = c(0, 60), expand = FALSE) +
  xlab("Longitude") +
  ylab("Latitude") +
  ggtitle("World Map - China") +
  theme(panel.grid.major = element_line(color = gray(.5), linetype = "dashed", size = 0.1), panel.background = element_rect(fill = "aliceblue"))+
  annotation_scale(location = "br", width_hint = 0.1) +
  annotation_north_arrow(location = "bl", which_north = "true",
      pad_x = unit(0.3, "cm"), pad_y = unit(0.3, "cm"),height = unit(1, "cm"),width = unit(1, "cm"),
      style = north_arrow_fancy_orienteering)
```



Create a World Map Using R

ggplot2

Use geom_map() and map_data() to create a world map.

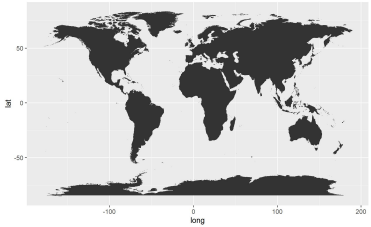
The steps of creating a world map:

1. Import the package
2. Use the map_data function to get the world data.
3. Use the geom_map() function to create the base map.

The above steps in code:

```
library(ggplot2)

world_data <- map_data("world")

ggplot() +
  geom_map(data = world_data, map = world_data,
aes(long, lat, map_id = region))
```

And we can get…



4. Use functions like geom_point() to map our own data to the map created.

Package rworldmap

The package *rworldmap* allows you to map different levels of data (e.g. country level, regional level) to a world map. It also contains datasets that you can use for exploratory analysis. Using this package also requires a package called "sp".

Sample datasets in the package

Two example datasets(calls in R)

```
1.data(countryExData)
2.data(gridExData)
```

Functions that join the data

1.joinCountryData2Map(joinCode, nameCountryColumn): Join the data with country codes to a world map.
2.joinData2Map(nameMap, nameJoinColumnData): Join polygon attribute data to a map.

Functions that display the map

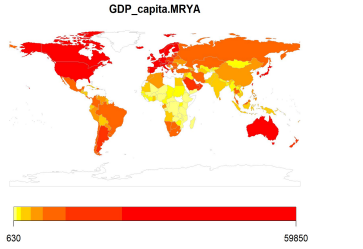
1.mapCountryData(mapToPlot, nameColumnToPlot, mapRegion): Map country-level data.
2.mapByRegion(inFile, nameDataColumn, joinCode, nameJoinColumn): Map regional-level data.
3.mapGriddedData(dataset, nameColumnToPlot): Map global gridded data at half resolution.
4.mapPloys(mapToPlot, nameColumnToPlot): Map polygon data.

Functions that adds info/detail

1.addMapLegends(colourVector, legendLabels): Add a legend to the map.
2.addMapLegendBoxes(cutVector, colourVector, horiz, title): Add a legend of colored boxes to the map.
3.mapBars(): Produce bar plots on a map.
4.mapBubbles(): Produce bubble plots on a map.
5.mapPies(): Produce pie charts on a map.

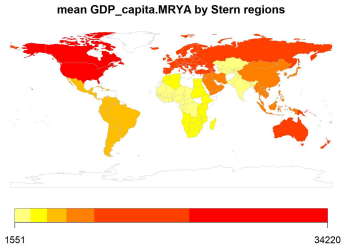
Example map of GDP per capita

```
data("countryExData")

dat <- joinCountryData2Map(countryExData,
joinCode = "ISO3", nameJoinColumn = "ISO3V10")

mapCountryData(dat,nameColumnToPlot =
"GDP_capita.MRYA")
```



Country level data to global level

```
data("countryExData")

mapByRegion(countryExData,
nameDataColumn="GDP_capita.MRYA",
joinCode="ISO3" , nameJoinColumn="ISO3V10" ,
regionType="Stern" , FUN="mean" )
```



Example of Gridded Data

```
data(gridExData)
mapGriddedData(gridExData)
```