In [this earlier post](https://www.displayr.com/chances-hit-by-meteorite/) we analysed the location of meteorite impacts from this [dataset](https://data.nasa.gov/Space-Science/Meteorite-Landings/gh4g-9sfh), including plotting their fall locations on a globe. Forming part of the analysis was this interactive globe visualization below, which plots the location and age of meteorites.

In this post I am going to show how to create this globe with code. I then go on to describe other options and variations on the same theme.

**Preparing and plotting the data**

The code below does everything we need. Stepping through each section:

* First, import the data then convert from the year the meteorite fell to an age in years.
* Tidy the data frame to collect just the variables that we need.
* Convert *age*into a number from 1 to 10 (the lowest 10% of ages map to 1, the next 10% to 2 .. etc), then map those numbers to shades of red.
* Finally, plot the data on the globe where *val* (the mass) determines the length of each line and fixed *pointsize* determines the line thickness.

library(threejs)

library(flipChartBasics)

# Read the data and calculate age in years

x = read.csv("https://data.nasa.gov/api/views/gh4g-9sfh/rows.csv")

current = as.numeric(format(Sys.Date(), "%Y"))

x$age = current - as.numeric(substr(x$year, 7, 10))

# Filter the required information

x = x[ , c("reclong", "reclat", "mass..g.", "age")]

colnames(x) = c("long","lat","value", "age")

# Set colors on a scale of 1 to 10 by percentile

colors = as.numeric(cut(x$age,

breaks = quantile(x$age, probs = seq(0, 1, 0.1),

include.lowest = TRUE, na.rm = TRUE)))

palette = ChartColors(10, "Reds", reverse = TRUE)

colors = palette[colors]

# Plot the data on the globe

globejs(lat = x$lat,

long = x$long,

val = 2 \* log(x$value),

color = colors,

pointsize = 0.5,

atmosphere = TRUE)

The output is slightly different from the original in the [meteorites post](https://www.displayr.com/chances-hit-by-meteorite/) because we have not excluded any data. You can spin the globe around with your mouse and zoom in or out.

**Capital cities**

As a second example, we’ll plot capital city data [available here](https://esa.un.org/unpd/wup/cd-rom/WUP2014_XLS_CD_FILES/WUP2014-F13-Capital_Cities.xls). The data is imported with the DownloadXLSX function from the flipAPI package.

The *Population (thousands)* column is imported as a factor. We amend it to be a number. The code to make the globe is broadly similar to that above.

library(threejs)

library(flipChartBasics)

library(flipAPI)

# Make a data.frame of the required information

url <- "https://esa.un.org/unpd/wup/cd-rom/WUP2014\_XLS\_CD\_FILES/WUP2014-F13-Capital\_Cities.xls"

x = DownloadXLSX(url, skip = 15)

x = x[, c("Longitude", "Latitude", "Population (thousands)", "Capital City")]

names(x) = c("long","lat", "population", "city")

# Convert population to numeric

x$population = as.character(x$population)

x$population[x$population == "\u2026"] = 0 # remove ellipsis

x$population = as.numeric(x$population)

# Set colors according to first letter of the city name

first.letters = sapply(substring(x$city, 1, 1),

utf8ToInt) - utf8ToInt("A") + 1

palette = ChartColors(26, "Blues")

colors = palette[first.letters]

# Plot the data on the globe

earth = "http://eoimages.gsfc.nasa.gov/images/imagerecords/73000/73909/world.topo.bathy.200412.3x5400x2700.jpg"

globejs(img = earth,

lat = x$lat,

long = x$long,

val = 10 \* log(x$population),

color = colors,

pointsize = 5,

atmosphere = FALSE,

bg = "white")

﻿

A few notable differences are:

* Addition of a more colorful background world image via the *img* argument.
* Line lengths related to population and colors according to the first letter of the city name (light blue = A, dark = Z).
* Removal of *atmosphere*, increase of *pointsize* and amending the default background (*bg)*color.

Finally, the globe image can be changed. If you wanted to confuse people by presenting the same information with an arbitrary citrus theme, you could make the following, again using the *img* argument.