Article Plotting Notebook

Introduction

This Notebook outlines the code required to create the plot of different samples of nutrients tested in water around the world. This plot will be used in an article that will hopefully be published in the Canadian Journal of Chemical Engineering.

The following code is used to install the required libraries.

```
library(tidyverse)
library(dataRetrieval)
library(ggmap)
library(maps)
library(mapdata)
library(RColorBrewer)
library(lubridate)
library(ggthemes)
library(readxl)
```

USA dataset

This next chunk of code is uses the readWQPdata function from the dataRetriveal package to download data from the Water Quality Portal. The Water Quality Portal is a collection of three datasets of water quality sampling across the USA. Only the NWIS dataset is used as the other datasets are not as clean.

The parameters are passed to the function to define the scope for the data to collection, for this instance it limits the window for when the sample was first collected and limits the database to solely the concentration of total Phosphorus.

```
wqp.data <- readWQPdata('startDateLo' = "2017-01-01",
                        'startDateHi' = "2018-12-31",
                        characteristicName=c("Phosphorus"),
                        providers= "NWIS") %>%
        filter(ResultSampleFractionText=="Total") %>%
        mutate(Result = as.numeric(ResultMeasureValue)) %>%
        filter(!is.na(Result),
               ActivityMediaName=="Water",
               Result > 0,
               ResultMeasure.MeasureUnitCode == "mg/l as P")
```

The data comes from the years 2017-2018 so there are multiple measurements from the same stations. The average value is obtained and all extra unnecessary attributes are dropped.

```
wqp.data <- wqp.data %>%
        select(MonitoringLocationIdentifier, CharacteristicName, Result) %>%
        group_by(MonitoringLocationIdentifier, CharacteristicName) %>%
        summarise(Result = mean(Result), .groups = "keep")
```

All of the geo-spacial data for the USA testing location are kept in a separate database so the left merge is used to add longitude and latitude data.

```
wqp.site <- whatWQPsites(siteid=wqp.data$MonitoringLocationIdentifier) %>%
        select(MonitoringLocationIdentifier, lng = LongitudeMeasure,
               lat = LatitudeMeasure)

USA <- wqp.data %>%
        left_join(wqp.site, by = "MonitoringLocationIdentifier") %>%
        arrange(Result)
```

Canada dataset

The Canadian dataset is much easier to handle and can be simply downloaded from the Canadian open data website. There is a limitation to the data in that the last updated data is from 2018. This will be the limitation for all the data as all other datasets are more recent.

```
canurl <- "https://www.canada.ca/content/dam/eccc/documents/csv/cesindicators/water-quality-canadian-ri
download.file(url = canurl, "Candata.csv")
Can.Data <- read_csv("Candata.csv", col_types = cols(GUIDELINE_REFERENCE_RECOMMANDATION = col_character
        filter(VARIABLE_NAME=="PHOSPHORUS") %>%
        mutate(DATE=dmy(DATE)) %>%
        select(DATE, VALUE_VALEUR, lat = LATITUDE, lng = LONGITUDE) %>%
        filter(DATE > dmy(01012017)) %>%
        group_by(lat, lng) %>%
        summarise(Avg_Con = mean(VALUE_VALEUR)) %>%
        arrange(Avg_Con)
```

European datset

The european dataset comes from Waterbase. This data is open and easy to download from their website but is very large, so be careful with your available disk space.

```
eurourl <- "https://cmshare.eea.europa.eu/s/XdNc9oFeXyEJknN/download"
download.file(url = eurourl, "eurodata.zip", mode="wb")
unzip("eurodata.zip", exdir = ".")
eurosite <- read_csv("Waterbase_v2020_1_S_WISE6_SpatialObject_DerivedData.csv") %>%
        select(monitoringSiteIdentifier, lat, lon)
eurodata <- read_csv("Waterbase_v2020_1_T_WISE6_AggregatedData.csv") %>%
        select(monitoringSiteIdentifier, param = observedPropertyDeterminandLabel, year = phenomenonTim
        filter(param == "Total phosphorus") %>%
        filter(year == "2017"| year == "2018" ) %>%
        left_join(eurosite) %>%
        filter(!is.na(lat), !is.na(lon)) %>%
        arrange(Result)
```

GEM datset

```
gemsite <- read_excel("GEMS-Water_data_request.xls")

gemdata <- read_csv2("Total_Phosphorus.csv",col_types = cols("Sample Date"=col_date())) %>%
        select("GEMS Station Number", "Sample Date", "Value")
gemdata$Value <- as.numeric(gemdata$Value)
gemdata <- gemdata %>% filter(`Sample Date` > ymd("2017-01-01")) %>%
        left_join(gemsite %>% select("GEMS Station Number", "Longitude", "Latitude")) %>%
        rename("lng"= "Longitude", "lat" = "Latitude") %>%
        group_by(lng, lat) %>%
```

```
        summarise(Result = mean(Value)) %>%
        arrange(Result)
```

Map information

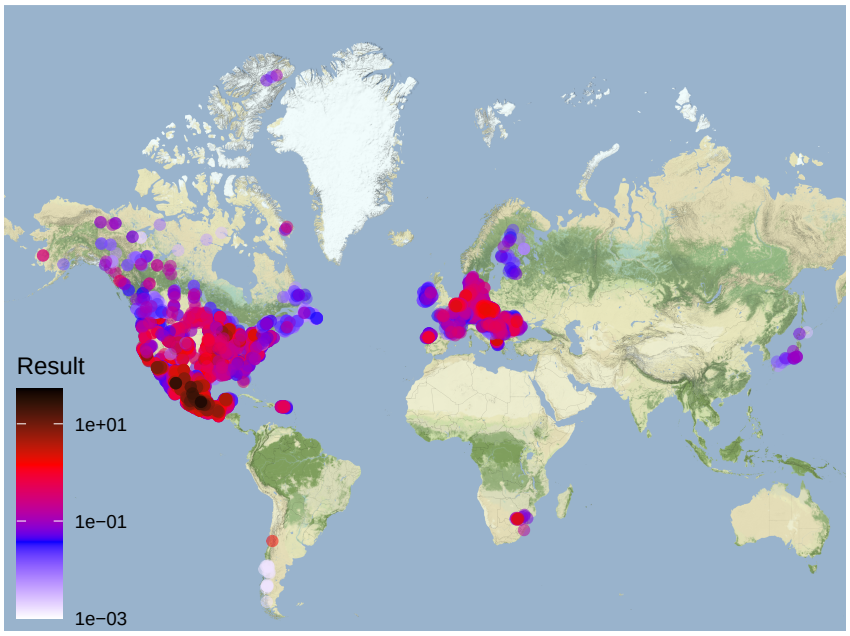
The map data is downloaded with the get_stamenmap function. The map is centered around the extreme values of the geo-spacial data.

```
Data <- data.frame(lon = c(USA$lng, Can.Data$lng, eurodata$lon, gemdata$lng),
                   lat = c(USA$lat, Can.Data$lat, eurodata$lat, gemdata$lat))
world_box <- make_bbox(lat=lat, lon = lon, data = Data)
gworld <- get_stamenmap(bbox = world_box,zoom = 5, maptype = "terrain-background")
```

Plotting

The color ramp palette function is used to create a custom gradient in the plotted concentration colours. With that, all the data can be plotted on the maps. The plot is of the log10 of the concentration as there are some values that are much higher than average.

```
colfunc<-colorRampPalette(c("white", "blue","red", "black"))
ggmap(gworld, extent = "device", padding = 0) +
  geom_point(data = USA, aes(x =lng, y = lat, color = Result), size = 1.5, alpha = 0.5) +
  geom_point(data = Can.Data, aes(x =lng, y = lat, color = Avg_Con), size = 1.5, alpha = 0.5) +
  geom_point(data = eurodata, aes(x =lon, y = lat, color = Result), size = 1.5, alpha = 0.5) +
  geom_point(data = gemdata, aes(x =lng, y = lat, color = Result), size = 1.5, alpha = 0.5) +
  scale_color_gradientn(colours = colfunc(4), trans = "log10") +
  theme_map() +
  theme(legend.background = element_blank()) +
  labs(caption = "Average measurements from 2017-2018")
```



Average measurements from 2017−2018

```
ggsave("Map.png")
```