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)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.3
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
## Warning: package 'forcats' was built under R version 4.0.5
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dataRetrieval)
library(ggmap)
```

```
## Warning: package 'ggmap' was built under R version 4.0.5
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(mapdata)
```

```
## Warning: package 'mapdata' was built under R version 4.0.5
```

```
library(RColorBrewer)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.5
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

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 globe.

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")) %>%
        filter(ResultSampleFractionText=="Total") %>%
        mutate(Result = as.numeric(ResultMeasureValue)) %>%
        filter(!is.na(Result),
               ActivityMediaName=="Water")
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

The USA dataset come from a wide arrange of sampling stations, 26446 stations in total. These different stations measure different units on concentration so the data needs to be cleaned and the units converted to mg/L.

```
Case1 <- wqp.data$ResultMeasure.MeasureUnitCode=="ug/l"| wqp.data$ResultMeasure.MeasureUnitCode=="ug"
Case2 <- wqp.data$ResultMeasure.MeasureUnitCode=="ppm"| wqp.data$ResultMeasure.MeasureUnitCode=="mg/l as
Case3 <- wqp.data$ResultMeasure.MeasureUnitCode=="ppb"
Case4 <- wqp.data$ResultMeasure.MeasureUnitCode=="mg/kg"
Case5 <- wqp.data$ResultMeasure.MeasureUnitCode=="%"

wqp.data$Result[Case1|Case3] <- wqp.data$Result[Case1|Case3]/1000
wqp.data$Result[Case5] <- wqp.data$Result[Case5]*10

wqp.data$ResultMeasure.MeasureUnitCode[Case1|Case2|Case3|Case4|Case5] <- "mg/l"
wqp.data <- wqp.data %>%
        filter(ResultMeasure.MeasureUnitCode=="mg/l") %>%
        filter(Result > 0)
```

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("Cansite.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)
```

```
## `summarise()` has grouped output by 'lat'. You can override using the `.groups` argument.
```

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)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   lat = col_double(),
##   lon = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
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) %>%
        arrange(Result)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   monitoringSiteIdentifier = col_character(),
##   monitoringSiteIdentifierScheme = col_character(),
##   parameterWaterBodyCategory = col_character(),
##   observedPropertyDeterminandCode = col_character(),
##   observedPropertyDeterminandLabel = col_character(),
##   procedureAnalysedMatrix = col_character(),
##   resultUom = col_character(),
##   parameterSamplingPeriod = col_character(),
##   procedureAnalyticalMethod = col_character(),
##   resultObservationStatus = col_logical(),
##   remarks = col_character(),
##   metadata_versionId = col_character(),
##   metadata_beginLifeSpanVersion = col_datetime(format = ""),
##   metadata_statusCode = col_character(),
##   metadata_observationStatus = col_character(),
##   metadata_statements = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
## Warning: 460491 parsing failures.
##     row                     col           expected actual                                                                 
## 1838944 resultObservationStatus 1/0/T/F/TRUE/FALSE      A 'Waterbase_v2020_1_T_WISE6_AggregatedData.
## 1838945 resultObservationStatus 1/0/T/F/TRUE/FALSE      A 'Waterbase_v2020_1_T_WISE6_AggregatedData.
## 1838946 resultObservationStatus 1/0/T/F/TRUE/FALSE      A 'Waterbase_v2020_1_T_WISE6_AggregatedData.
## 1838947 resultObservationStatus 1/0/T/F/TRUE/FALSE      A 'Waterbase_v2020_1_T_WISE6_AggregatedData.
## 1838948 resultObservationStatus 1/0/T/F/TRUE/FALSE      A 'Waterbase_v2020_1_T_WISE6_AggregatedData.
## ....... ...................... .................. ...... .....................................
## See problems(...) for more details.
```

```
## Joining, by = "monitoringSiteIdentifier"
```

Map information

There are two maps used, a watercolour map from google and a map the world as an outline with the "worldHires" map. The google map does require the use of the google maps api which will require an acount an the use of the "register_google" function.

```
gworld <- get_map(location = c(lon = -42, lat  = 54),
                  maptype = "watercolor",
                  zoom = 2,
                  source = "stamen")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=54,-42&zoom=2&size=640x640&scale=2&map
```

```
## Source : http://tile.stamen.com/watercolor/2/0/0.jpg
```

```
## Source : http://tile.stamen.com/watercolor/2/1/0.jpg
```

```
## Source : http://tile.stamen.com/watercolor/2/2/0.jpg
```

```
## Source : http://tile.stamen.com/watercolor/2/0/1.jpg
```

```
## Source : http://tile.stamen.com/watercolor/2/1/1.jpg
```

```
## Source : http://tile.stamen.com/watercolor/2/2/1.jpg
```

```
## Source : http://tile.stamen.com/watercolor/2/0/2.jpg
## Source : http://tile.stamen.com/watercolor/2/1/2.jpg
## Source : http://tile.stamen.com/watercolor/2/2/2.jpg
```

```
world <- map_data("worldHires")
```

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.

```
colfunc<-colorRampPalette(c("yellow","red","springgreen","royalblue"))
gplot <- ggmap(gworld, extent = "device", padding = 1) +
        geom_polygon(data = world, aes(x=long, y = lat, group = group),fill = NA, color="black") +
        geom_point(data = USA, aes(x =lng, y = lat, color = Result), size = 2, alpha = 0.5) +
        geom_point(data = Can.Data, aes(x =lng, y = lat, color = Avg_Con), size = 2, alpha = 0.5) +
        geom_point(data = eurodata, aes(x =lon, y = lat, color = Result), size = 2, alpha = 0.5) +
        scale_color_gradientn(colours = colfunc(4))
gplot
```

```
## Warning: Removed 88 rows containing missing values (geom_point).
## Warning: Removed 78 rows containing missing values (geom_point).
```