In-class Exercise 2: Geospatial Data Wrangling

1/16/23

## 1 Overview

### 1.1 Setting the Scene

Water is an important resource to mankind. Clean and accessible water is critical to human health. It provides a healthy environment, a sustainable economy, reduces poverty and ensures peace and security. Yet over 40% of the global population does not have access to sufficient clean water. By 2025, 1.8 billion people will be living in countries or regions with absolute water scarcity, according to UN-Water. The lack of water poses a major threat to several sectors, including food security. Agriculture uses about 70% of the world’s accessible freshwater.

Developing countries are most affected by water shortages and poor water quality. Up to 80% of illnesses in the developing world are linked to inadequate water and sanitation. Despite technological advancement, providing clean water to the rural community is still a major development issues in many countries globally, especially countries in the Africa continent.

To address the issue of providing clean and sustainable water supply to the rural community, a global [Water Point Data Exchange (WPdx)](https://www.waterpointdata.org/about/) project has been initiated. The main aim of this initiative is to collect water point related data from rural areas at the water point or small water scheme level and share the data via WPdx Data Repository, a cloud-based data library. What is so special of this project is that data are collected based on [WPDx Data Standard](https://www.waterpointdata.org/wp-content/uploads/2021/04/WPDx_Data_Standard.pdf).

### 1.2 Objectives

Geospatial analytics hold tremendous potential to address complex problems facing society. In this study, you are tasked to apply appropriate geospatial data wrangling methods to prepare the data for water point mapping study. For the purpose of this study, Nigeria will be used as the study country.

### 1.3 The Data

#### 1.3.1 Apstial data

For the purpose of this assignment, data from [WPdx Global Data Repositories](https://www.waterpointdata.org/access-data/) will be used. There are two versions of the data. They are: WPdx-Basic and WPdx+. You are required to use WPdx+ data set.

#### 1.3.2 Geospatial data

Nigeria Level-2 Administrative Boundary (also known as Local Government Area) polygon features GIS data will be used in this take-home exercise. The data can be downloaded either from The [Humanitarian Data Exchange](https://data.humdata.org/) portal or [geoBoundaries](https://www.geoboundaries.org/index.html).

### 1.4 The Task

The specific tasks of this take-home exercise are as follows:

* Using appropriate sf method, import the shapefile into R and save it in a simple feature data frame format. Note that there are three Projected Coordinate Systems of Nigeria, they are: EPSG: 26391, 26392, and 26303. You can use any one of them.
* Using appropriate tidyr and dplyr methods, derive the number of functional and non-functional water points at LGA level.
* Combining the geospatial and aspatial data frame into simple feature data frame.
* Visualising the distribution of water point by using appropriate statistical methods.

## 2 Getting started

For the purpose of this in-class exercise, three R packages will be used. They are: sf, tidyverse and funModeling.

|  |
| --- |
| Your turn |
| Using the step you had learned, check if these three R packages have been installed in you laptop, if not install the missing R packages. If Yes, launch the R packages into R environment |

pacman::p\_load(sf, tidyverse, funModeling)

## 3 Handling Geospatial Data

### 3.1 Importing Geospatial

|  |
| --- |
| Your turn |
| Using the step you had learned, import the LGA boundary GIS data of Nigeria downloaded from both sources recommend above. |

#### 3.1.1 The geoBoundaries data set

geoNGA <- st\_read("data/geospatial/",  
 layer = "geoBoundaries-NGA-ADM2") %>%  
 st\_transform(crs = 26392)

Reading layer `geoBoundaries-NGA-ADM2' from data source   
 `D:\tskam\IS415-GAA\In-class\_Ex\In-class\_Ex02\data\geospatial'   
 using driver `ESRI Shapefile'  
Simple feature collection with 774 features and 5 fields  
Geometry type: MULTIPOLYGON  
Dimension: XY  
Bounding box: xmin: 2.668534 ymin: 4.273007 xmax: 14.67882 ymax: 13.89442  
Geodetic CRS: WGS 84

#### 3.1.2 The NGA data set

NGA <- st\_read("data/geospatial/",  
 layer = "nga\_admbnda\_adm2") %>%  
 st\_transform(crs = 26392)

Reading layer `nga\_admbnda\_adm2' from data source   
 `D:\tskam\IS415-GAA\In-class\_Ex\In-class\_Ex02\data\geospatial'   
 using driver `ESRI Shapefile'  
Simple feature collection with 774 features and 16 fields  
Geometry type: MULTIPOLYGON  
Dimension: XY  
Bounding box: xmin: 2.668534 ymin: 4.273007 xmax: 14.67882 ymax: 13.89442  
Geodetic CRS: WGS 84

### 3.2 Importing Aspatial data

|  |
| --- |
| Your turn |
| Using the steps you had learned, import the downloaded water point data set into R. |

wp\_nga <- read\_csv("data/aspatial/WPdx.csv") %>%  
 filter(`#clean\_country\_name` == "Nigeria")

#### 3.2.1 Converting water point data into sf point features

|  |
| --- |
| Your turn |
| Using the steps you had learned, convert the newly extracted wp\_NGA into point sf data frame |

wp\_nga$Geometry = st\_as\_sfc(wp\_nga$`New Georeferenced Column`)  
wp\_nga

# A tibble: 95,008 × 71  
 row\_id `#source` #lat\_…¹ #lon\_…² #repo…³ #stat…⁴ #wate…⁵ #wate…⁶ #wate…⁷  
 <dbl> <chr> <dbl> <dbl> <chr> <chr> <chr> <chr> <chr>   
 1 429068 GRID3 7.98 5.12 08/29/… Unknown <NA> <NA> Tapsta…  
 2 222071 Federal Minis… 6.96 3.60 08/16/… Yes Boreho… Well Mechan…  
 3 160612 WaterAid 6.49 7.93 12/04/… Yes Boreho… Well Hand P…  
 4 160669 WaterAid 6.73 7.65 12/04/… Yes Boreho… Well <NA>   
 5 160642 WaterAid 6.78 7.66 12/04/… Yes Boreho… Well Hand P…  
 6 160628 WaterAid 6.96 7.78 12/04/… Yes Boreho… Well Hand P…  
 7 160632 WaterAid 7.02 7.84 12/04/… Yes Boreho… Well Hand P…  
 8 642747 Living Water … 7.33 8.98 10/03/… Yes Boreho… Well Mechan…  
 9 642456 Living Water … 7.17 9.11 10/03/… Yes Boreho… Well Hand P…  
10 641347 Living Water … 7.20 9.22 03/28/… Yes Boreho… Well Hand P…  
# … with 94,998 more rows, 62 more variables: `#water\_tech\_category` <chr>,  
# `#facility\_type` <chr>, `#clean\_country\_name` <chr>, `#clean\_adm1` <chr>,  
# `#clean\_adm2` <chr>, `#clean\_adm3` <chr>, `#clean\_adm4` <chr>,  
# `#install\_year` <dbl>, `#installer` <chr>, `#rehab\_year` <lgl>,  
# `#rehabilitator` <lgl>, `#management\_clean` <chr>, `#status\_clean` <chr>,  
# `#pay` <chr>, `#fecal\_coliform\_presence` <chr>,  
# `#fecal\_coliform\_value` <dbl>, `#subjective\_quality` <chr>, …

wp\_sf <- st\_sf(wp\_nga, crs=4326)  
wp\_sf

Simple feature collection with 95008 features and 70 fields  
Geometry type: POINT  
Dimension: XY  
Bounding box: xmin: 2.707441 ymin: 4.301812 xmax: 14.21828 ymax: 13.86568  
Geodetic CRS: WGS 84  
# A tibble: 95,008 × 71  
 row\_id `#source` #lat\_…¹ #lon\_…² #repo…³ #stat…⁴ #wate…⁵ #wate…⁶ #wate…⁷  
 \* <dbl> <chr> <dbl> <dbl> <chr> <chr> <chr> <chr> <chr>   
 1 429068 GRID3 7.98 5.12 08/29/… Unknown <NA> <NA> Tapsta…  
 2 222071 Federal Minis… 6.96 3.60 08/16/… Yes Boreho… Well Mechan…  
 3 160612 WaterAid 6.49 7.93 12/04/… Yes Boreho… Well Hand P…  
 4 160669 WaterAid 6.73 7.65 12/04/… Yes Boreho… Well <NA>   
 5 160642 WaterAid 6.78 7.66 12/04/… Yes Boreho… Well Hand P…  
 6 160628 WaterAid 6.96 7.78 12/04/… Yes Boreho… Well Hand P…  
 7 160632 WaterAid 7.02 7.84 12/04/… Yes Boreho… Well Hand P…  
 8 642747 Living Water … 7.33 8.98 10/03/… Yes Boreho… Well Mechan…  
 9 642456 Living Water … 7.17 9.11 10/03/… Yes Boreho… Well Hand P…  
10 641347 Living Water … 7.20 9.22 03/28/… Yes Boreho… Well Hand P…  
# … with 94,998 more rows, 62 more variables: `#water\_tech\_category` <chr>,  
# `#facility\_type` <chr>, `#clean\_country\_name` <chr>, `#clean\_adm1` <chr>,  
# `#clean\_adm2` <chr>, `#clean\_adm3` <chr>, `#clean\_adm4` <chr>,  
# `#install\_year` <dbl>, `#installer` <chr>, `#rehab\_year` <lgl>,  
# `#rehabilitator` <lgl>, `#management\_clean` <chr>, `#status\_clean` <chr>,  
# `#pay` <chr>, `#fecal\_coliform\_presence` <chr>,  
# `#fecal\_coliform\_value` <dbl>, `#subjective\_quality` <chr>, …

#### 3.2.2 Transforming into Nigeria projected coordinate system

|  |
| --- |
| Your turn |
| Using the steps you had learned, transform the projection from wgs84 to appropriate projected coordinate system of Nigeria. |

wp\_sf <- wp\_sf %>%  
 st\_transform(crs = 26392)

## 4 Geospatial Data Cleaning

NGA <- NGA %>%  
 select(c(3:4, 8:9))

### 4.1 Checking for duplicate name

It is always important to check for duplicate name in the data main data fields. Using duplicated() of Base R, we can flag out LGA names that might be duplicated as shown in the code chunk below.

NGA$ADM2\_EN[duplicated(NGA$ADM2\_EN)==TRUE]

[1] "Bassa" "Ifelodun" "Irepodun" "Nasarawa" "Obi" "Surulere"

The printout above shows that there are 6 LGAs with the same name. A Google search using the coordinates showed that there are LGAs with the same name but are located in different states. For instances, there is a Bassa LGA in Kogi State and a Bassa LGA in Plateau State.

Let us correct these errors by using the code chunk below.

NGA$ADM2\_EN[94] <- "Bassa, Kogi"  
NGA$ADM2\_EN[95] <- "Bassa, Plateau"  
NGA$ADM2\_EN[304] <- "Ifelodun, Kwara"  
NGA$ADM2\_EN[305] <- "Ifelodun, Osun"  
NGA$ADM2\_EN[355] <- "Irepodun, Kwara"  
NGA$ADM2\_EN[356] <- "Irepodun, Osun"  
NGA$ADM2\_EN[519] <- "Nasarawa, Kano"  
NGA$ADM2\_EN[519] <- "Nasarawa, Nasarawa"  
NGA$ADM2\_EN[546] <- "Obi, Benue"  
NGA$ADM2\_EN[547] <- "Obi, Nasarawa"  
NGA$ADM2\_EN[693] <- "Surulere, Lagos"  
NGA$ADM2\_EN[694] <- "Surulere, Oyo"

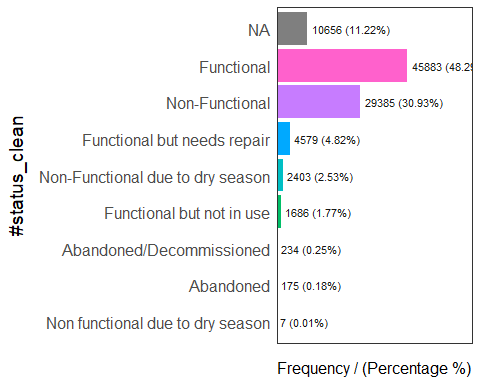
Now, let us rerun the code chunk below to confirm that the duplicated name issue has been addressed.

NGA$ADM2\_EN[duplicated(NGA$ADM2\_EN)==TRUE]

character(0)

## 5 Data Wrangling for Water Point Data

freq(data = wp\_sf,  
 input = '#status\_clean')



#status\_clean frequency percentage cumulative\_perc  
1 Functional 45883 48.29 48.29  
2 Non-Functional 29385 30.93 79.22  
3 <NA> 10656 11.22 90.44  
4 Functional but needs repair 4579 4.82 95.26  
5 Non-Functional due to dry season 2403 2.53 97.79  
6 Functional but not in use 1686 1.77 99.56  
7 Abandoned/Decommissioned 234 0.25 99.81  
8 Abandoned 175 0.18 99.99  
9 Non functional due to dry season 7 0.01 100.00

We will then load the data in rds format. In the following code chunk, we will also rename the column from #status\_clean to status\_clean for easier handling in subsequent steps. In addition, replace\_na() is used to recode all the NA values in status\_clean into unknown.

wp\_sf\_nga <- wp\_sf %>%   
 rename(status\_clean = '#status\_clean') %>%  
 select(status\_clean) %>%  
 mutate(status\_clean = replace\_na(  
 status\_clean, "unknown"))

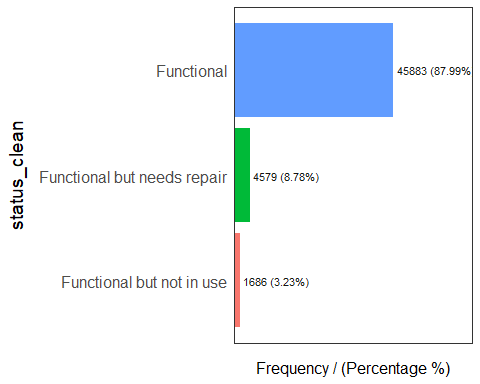
Extracting Water Point Data

wp\_functional <- wp\_sf\_nga %>%  
 filter(status\_clean %in%  
 c("Functional",  
 "Functional but not in use",  
 "Functional but needs repair"))

wp\_nonfunctional <- wp\_sf\_nga %>%  
 filter(status\_clean %in%  
 c("Abandoned/Decommissioned",  
 "Abandoned",  
 "Non-Functional due to dry season",  
 "Non-Functional",  
 "Non functional due to dry season"))

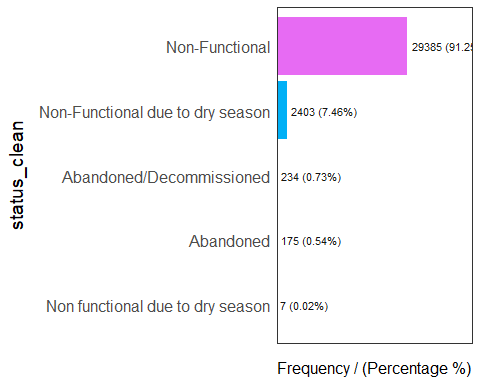
wp\_unknown <- wp\_sf\_nga %>%  
 filter(status\_clean == "unknown")

freq(data = wp\_functional,  
 input = 'status\_clean')



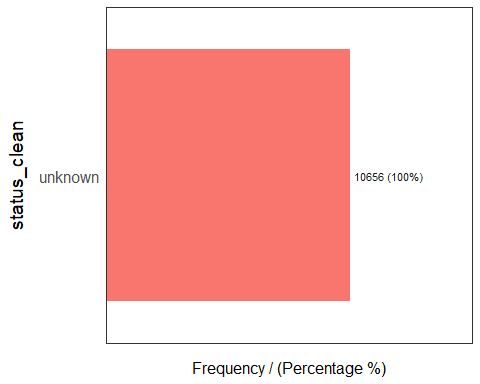
status\_clean frequency percentage cumulative\_perc  
1 Functional 45883 87.99 87.99  
2 Functional but needs repair 4579 8.78 96.77  
3 Functional but not in use 1686 3.23 100.00

freq(data = wp\_nonfunctional,  
 input = 'status\_clean')



status\_clean frequency percentage cumulative\_perc  
1 Non-Functional 29385 91.25 91.25  
2 Non-Functional due to dry season 2403 7.46 98.71  
3 Abandoned/Decommissioned 234 0.73 99.44  
4 Abandoned 175 0.54 99.98  
5 Non functional due to dry season 7 0.02 100.00

freq(data = wp\_unknown,  
 input = 'status\_clean')



status\_clean frequency percentage cumulative\_perc  
1 unknown 10656 100 100

Performing Point-in-Polygon Count

Next, we want to find the number of functional water points in each LGA as well as the number of total, functional, non-functional, and unknown water points in each LGA. This is performed in the following code chunk. First, it identifies the functional water points in each LGA by using st\_intersects(). Next, length() is used to calculate the number of functional water points that fall inside each LGA.

NGA\_wp <- NGA %>%   
 mutate(`total\_wp` = lengths(  
 st\_intersects(NGA, wp\_sf\_nga))) %>%  
 mutate(`wp\_functional` = lengths(  
 st\_intersects(NGA, wp\_functional))) %>%  
 mutate(`wp\_nonfunctional` = lengths(  
 st\_intersects(NGA, wp\_nonfunctional))) %>%  
 mutate(`wp\_unknown` = lengths(  
 st\_intersects(NGA, wp\_unknown)))

write\_rds(NGA\_wp, "data/rds/NGA\_wp.rds")

ggplot(data = NGA\_wp,  
 aes(x = total\_wp)) +   
 geom\_histogram(bins=20,  
 color="black",  
 fill="light blue") +  
 geom\_vline(aes(xintercept=mean(  
 total\_wp, na.rm=T)),  
 color="red",   
 linetype="dashed",   
 size=0.8) +  
 ggtitle("Distribution of total water points by LGA") +  
 xlab("No. of water points") +  
 ylab("No. of\nLGAs") +  
 theme(axis.title.y=element\_text(angle = 0))

