Data Inventory Script

L. Patterson; August 2019

Rmarkdown File: https://github.com/internetofwater/DataInventory

Read in Data

The following script can be run in R to generate the tables and graphs observed in the data inventory website: https://internetofwater.org/resources/inventory

```
#Load in libraries
library(dplyr); library(magrittr); library(tidyr);
library(plotly); library(ggplot2); library(tidyverse);
library(readxl); #in tidyverse
library(visNetwork);

#removes anything stored in memory
rm(list=ls())

#Set up directories. swd is the main directory where the spreadsheets are
located. You may have multiple inventories in different folders. Here, we
have Inventories for the federal government and three states.

#main directory **USER INPUT REQUIRED**
swd = "C:\\Users\\xxx\\Documents\\DataInventory\\"
folderName <- "federal\\";    #federal, ca, tx, nc
fileName <- "Federal";        #Federal, CA, TX, NC

#Load in the individual tabs on the worksheets
#paste0 concatenates the working directory, folder, filenames, etc.
orgNodes <- read_excel(paste0(swd,folderName,fileName,"_DataInventory.xlsx"),
sheet="orgNodes")

orgEdges <- read_excel(paste0(swd,folderName,fileName,"_DataInventory.xlsx"),
sheet="orgEdges") #this is optional to create in excel or in R

platformNodes <- read_excel(paste0(swd,folderName,fileName,"_DataInventory.xl
sx"), sheet="platformNodes")
dataNodes <- read_excel(paste0(swd,folderName,fileName,"_DataInventory.xlsx")
, sheet="dataNodes")

dataNodeTypes <- read_excel(paste0(swd,folderName,fileName,"_DataInventory.xl
sx"), sheet="dataNodeTypes");
#remove notes and extraneous columns
dataNodeTypes <- dataNodeTypes[,c(1:9)]

dataEdges <- read_excel(paste0(swd,folderName,fileName,"_DataInventory.xlsx")
, sheet="dataEdges") # this is optional to create in excel or in R
```

Create Edge Tables

The **orgEdges** and **dataEdges** worksheets can be manually create or created in R. The following scripts will create the edge files if they have not already been generated.

```
#############################################################################
#Create orgEdges Table
#############################################################################

#create data frame
lastRow <- dim(orgNodes)[1]
orgEdges2 <- as.data.frame(matrix(nrow=lastRow-1, ncol=2));
    colnames(orgEdges2) <- c("from","to");

#populate first row
orgEdges2$from <- orgNodes[c(2:lastRow),]$entityIDAbove
orgEdges2$to <- orgNodes[c(2:lastRow),]$entityID

#Sort org edges regardless
orgEdges <- orgEdges %>% arrange(from,to) %>% as.data.frame()
head(orgEdges)
```

```
##   from  to
## 1  n01 n02
## 2  n01 n07
## 3  n01 n10
## 4  n01 n13
## 5  n01 n23
## 6  n01 n27
```

```
#############################################################################
#Create dataEdges Table
#############################################################################

#create data frame
dataEdges <- as.data.frame(matrix(nrow=dim(dataNodeTypes)[1], ncol=2));
  colnames(dataEdges) <- c("from","to");

#populate first row
dataEdges$from <- dataNodeTypes$platformID;

#populate the to column
dataEdges$to <- dataNodeTypes$dataID

#remove duplicates if any and sort dataEdges
dataEdges <- dataEdges %>% distinct(from,to) %>% arrange(from, to) %>%
   as.data.frame();

head(dataEdges)
```

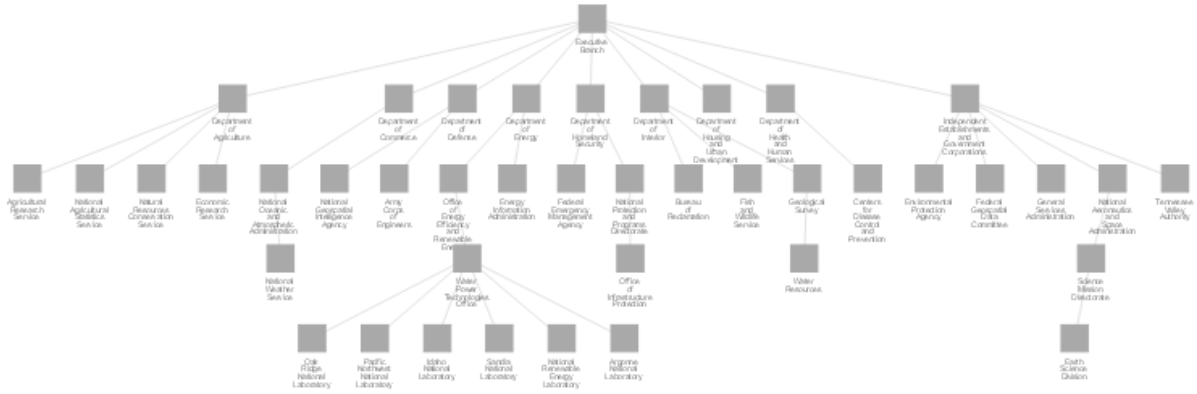
```
##   from  to
## 1  p01 d01
## 2  p02 d06
## 3  p03 d01
## 4  p05 d01
## 5  p05 d02
## 6  p05 d10
```

Create Organizational Network

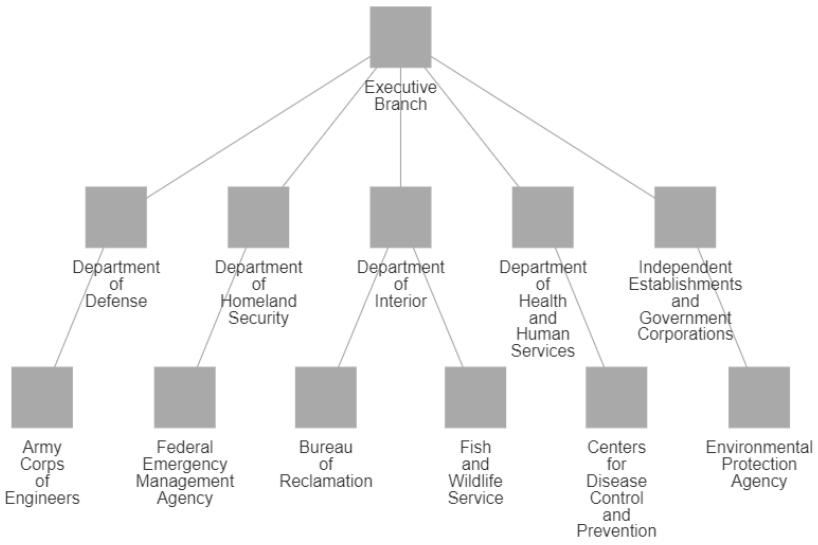
The following script will create a simple hierarchy of the organization. The organization can be changed to display those with a specific purpose by sub-setting.

```
# create nodes in format
nodes <- as.data.frame(matrix(nrow=dim(orgNodes)[1],ncol=0))
  nodes$id <- orgNodes$entityID;
  nodes$label <- gsub(" ", "\n", orgNodes$entity);
  nodes$title <- nodes$label;
  nodes$operational <- orgNodes$operational;
  nodes$decision <- orgNodes$decision;
  nodes$regulatory <- orgNodes$regulatory;
  nodes$research <- orgNodes$research;

  nodes$shape <- "square";
  nodes$color <- "darkgray";
  nodes$shadow <- FALSE;

# draw network
visNetwork(nodes, orgEdges, main="Organizational Layout",
  width="800px", height="400px") %>%
   visHierarchicalLayout()
```

Organizational Layout



```
#Create network showing regulatory data only
nodes.Reg <- subset(nodes, regulatory=="Yes");
edges.Reg <- orgEdges[orgEdges$from %in% nodes.Reg$id,]

visNetwork(nodes.Reg, edges.Reg, main="Regulatory Organizations") %>%
  visHierarchicalLayout()
```



Create network for data platforms

This code will create a file that is used to draw the platform network.

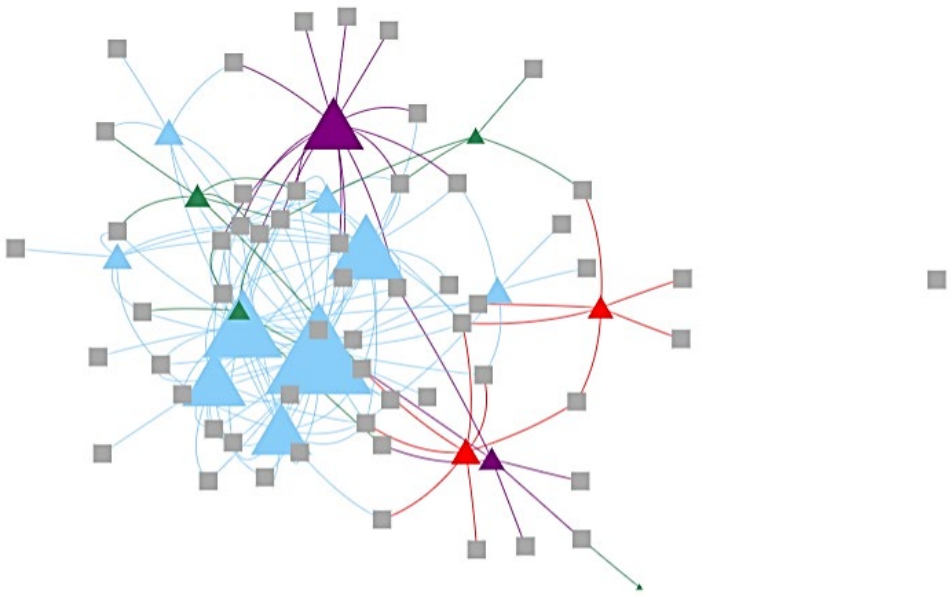
```
##############################################################################
#combine platformNodes and dataNodes into a single file for javascript
##############################################################################

#create dataframe
platformNodes$group <- "hub";
comboNodes <- platformNodes %>% as.data.frame();
  start <- dim(comboNodes)[1]+1;
  end <- dim(dataNodes)[1]+start-1;

# add data nodes to the end of the file
comboNodes <- rbind(comboNodes, comboNodes[c(1:dim(dataNodes)[1]),]);
comboNodes[c(start:end),] <- NA

#create new ids
comboIDs <- c(platformNodes$platformID,dataNodes$dataID)
  comboNodes$platformID <- comboIDs;

#create new platform names
```

```
comboNames <- c(platformNodes$platform, dataNodes$dataGroup)
  comboNodes$platform <- comboNames;

#create new web labels
comboLabel <- c(platformNodes$webLabel, dataNodes$dataCategory)
  comboNodes$webLabel <- comboLabel;

#create new groups
comboGroup <- c(platformNodes$group, dataNodes$dataCategory)
  comboNodes$group <- comboGroup;

#Count number of occurrences --- sets the size of the nodes
comboNodes$count = 5;
toCount = table(dataEdges$to) %>% as.data.frame();

#toCount;
comboNodes <- merge(comboNodes, toCount, by.x="platformID", by.y="Var1", all
= TRUE)

  comboNodes$count = ifelse(comboNodes$group != "hub" & is.na(comboNodes$Freq
)==TRUE, 0, comboNodes$count)

  comboNodes$count = ifelse(is.na(comboNodes$Freq)==TRUE, comboNodes$count, c
omboNodes$Freq)

#comboNodes$count = comboNodes$count+5;
comboNodes <- comboNodes %>% select(-Freq)

#head(comboNodes);
```

Now we create the characteristics to be displayed in the network visualization.

```
# create characteristics of the nodes
nodesP <- as.data.frame(matrix(nrow=dim(comboNodes)[1],ncol=0))
nodesP$id <- comboNodes$platformID;
  #nodesP$label <- gsub(" ", "\n", comboNodes$webLabel);
nodesP$title = ifelse(comboNodes$group=="hub",
  paste0("<p style='color: black';><strong>", comboNodes$webLabel,
  "</strong></p>"), paste0("<p style='color: black';><strong>",
  comboNodes$platform, "</strong></p>"))
nodesP[,c(3:16)] <- comboNodes[,c(10:23)]

nodesP$shape <- ifelse(nodesP$group=="hub","square","triangle");
nodesP$color <- ifelse(nodesP$group=="hub","darkgray","red");
  nodesP$color <- ifelse(nodesP$group=="Quality","purple",nodesP$color);
  nodesP$color <- ifelse(nodesP$group=="Quantity","lightskyblue",nodesP$color
);
  nodesP$color <- ifelse(nodesP$group=="Use","seagreen",nodesP$color);
nodesP$size <- nodesP$count*3.5;

#create edges and format
```

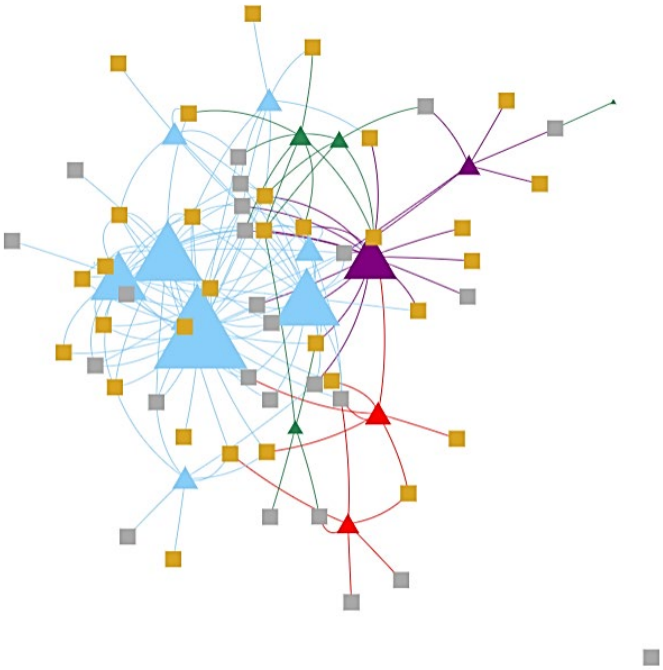
```
edgesD <- merge(dataEdges, nodesP[,c("id","color")], by.x="to", by.y="id")
  edgesD$width <- 2

#draw network
visNetwork(nodesP, edgesD, main = "Network of Platforms and Data Types",
  height="600px", width="600px")
```



Similarly, the nodes can be highlighted and the graph redrawn based on your interest. The example below will show how. More complicated parsing of data can be found in the scorecard section below.

```
#select those hubs that provide glossaries for their data.

nodesPHigh <- nodesP;
nodesPHigh$color <- ifelse(nodesPHigh$dataDefinitions=="Yes" &
  nodesPHigh$group=="hub", "goldenrod", nodesPHigh$color);

#redraw network
visNetwork(nodesPHigh, edgesD,
  main = "Highlight Platforms with Data Glossaries",
  height="600px", width="600px") %>%
    visLayout(randomSeed = 12) # to have always the same network
```

Highlight Platforms with Data Glossaries



Create the Heatmap

The following code reformats the data to create a heatmap.

```
#############################################################################
#Create heatmap table
#############################################################################
#create data frame
heatmap <- as.data.frame(matrix(nrow=dim(platformNodes)[1], ncol=(3+dim(dataN
odes)[1])));

#create column names from dataNodes
heatmapColNames <- c(str_replace_all(dataNodes$dataGroup, " ", ""))
colnames(heatmap) <- c("platformID", "entityID", "platform", heatmapColNames)

#fill first three columns
heatmap$platformID <- platformNodes$platformID;
heatmap$entityID <- platformNodes$entityID;
heatmap$platform <- platformNodes$heatmapLabel;

#loop through and subset dataEdges based on platform ID
for(i in 1:dim(platformNodes)[1]){
  zt.edges <- subset(dataEdges, from==platformNodes$platformID[i]) %>%
        as.data.frame();
```

```
  #if the value exists, then fill in with a 1 [row, col]
  heatmap[i,4] <- ifelse(dim(subset(zt.edges, to=="d01"))[1]==1,1,0);
  heatmap[i,5] <- ifelse(dim(subset(zt.edges, to=="d02"))[1]==1,1,0);
  heatmap[i,6] <- ifelse(dim(subset(zt.edges, to=="d03"))[1]==1,3,0);
  heatmap[i,7] <- ifelse(dim(subset(zt.edges, to=="d04"))[1]==1,3,0);
  heatmap[i,8] <- ifelse(dim(subset(zt.edges, to=="d05"))[1]==1,2,0);
  heatmap[i,9] <- ifelse(dim(subset(zt.edges, to=="d06"))[1]==1,2,0);
  heatmap[i,10] <- ifelse(dim(subset(zt.edges, to=="d07"))[1]==1,2,0);
  heatmap[i,11] <- ifelse(dim(subset(zt.edges, to=="d08"))[1]==1,2,0);
  heatmap[i,12] <- ifelse(dim(subset(zt.edges, to=="d09"))[1]==1,2,0);
  heatmap[i,13] <- ifelse(dim(subset(zt.edges, to=="d10"))[1]==1,2,0);
  heatmap[i,14] <- ifelse(dim(subset(zt.edges, to=="d11"))[1]==1,2,0);
  heatmap[i,15] <- ifelse(dim(subset(zt.edges, to=="d12"))[1]==1,2,0);
  heatmap[i,16] <- ifelse(dim(subset(zt.edges, to=="d13"))[1]==1,2,0);
  heatmap[i,17] <- ifelse(dim(subset(zt.edges, to=="d14"))[1]==1,4,0);
  heatmap[i,18] <- ifelse(dim(subset(zt.edges, to=="d15"))[1]==1,4,0);
  heatmap[i,19] <- ifelse(dim(subset(zt.edges, to=="d16"))[1]==1,4,0);
  heatmap[i,20] <- ifelse(dim(subset(zt.edges, to=="d17"))[1]==1,4,0);
  heatmap[i,21] <- ifelse(dim(subset(zt.edges, to=="d18"))[1]==1,4,0);
} #end loop

#sort by entityID and then platform
heatmap <- heatmap %>% arrange(platform)
head(heatmap)
```

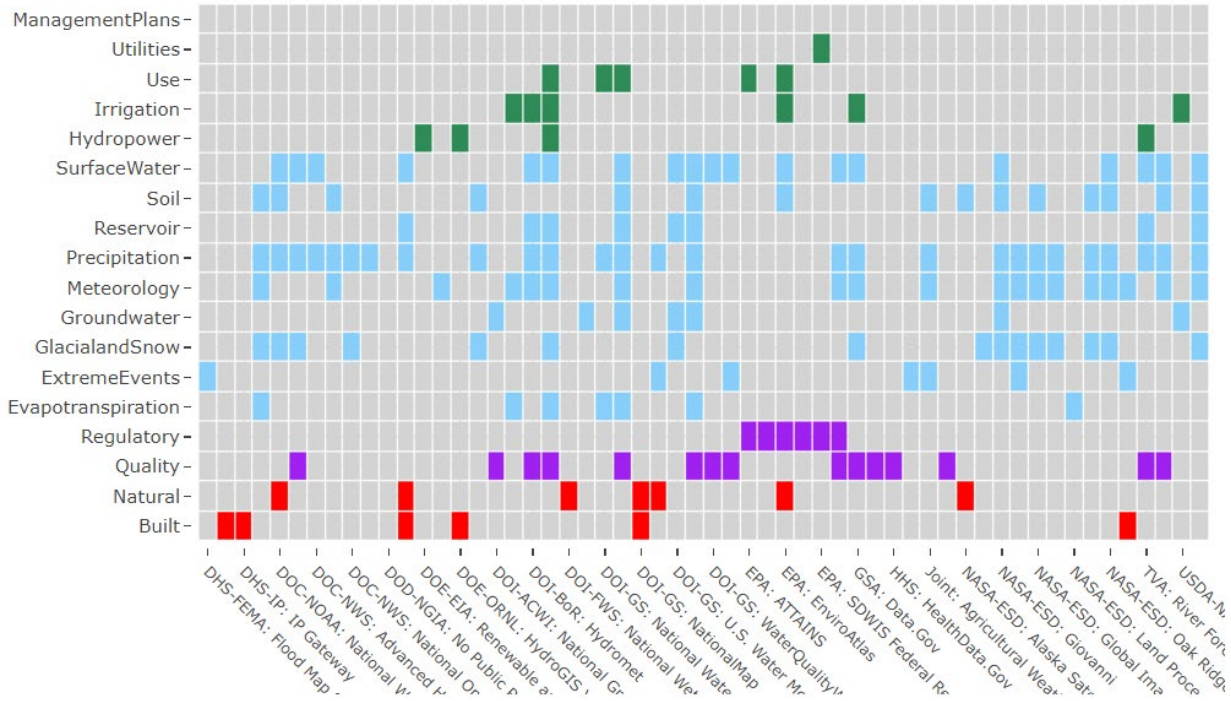
```
##   platformID entityID                           platform Built Natural
## 1        p02      n24 DHS-FEMA: Flood Map Service Center     0       0
## 2        p01      n26        DHS-IP: HSIN-CI Dams Portal     1       0
## 3        p03      n26                DHS-IP: IP Gateway     1       0
## 4        p49      n08      DOC-NOAA: Climate Data Online     0       0
## 5        p55      n08     DOC-NOAA: National Water Model     0       1
## 6        p50      n08                  DOC-NOAA: OneStop     0       0

##   Quality Regulatory Evapotranspiration ExtremeEvents GlacialandSnow
## 1       0          0                  0             2              0
## 2       0          0                  0             0              0
## 3       0          0                  0             0              0
## 4       0          0                  2             0              2
## 5       0          0                  0             0              2
## 6       3          0                  0             0              2

##   Groundwater Meteorology Precipitation Reservoir Soil SurfaceWater
## 1           0           0             0         0    0            0
## 2           0           0             0         0    0            0
## 3           0           0             0         0    0            0
## 4           0           2             2         0    2            0
## 5           0           0             2         0    2            2
## 6           0           0             2         0    0            2

##   Hydropower Irrigation Use Utilities ManagementPlans
## 1          0          0   0         0               0
## 2          0          0   0         0               0
```

```
## 3           0           0   0           0                   0
## 4           0           0   0           0                   0
## 5           0           0   0           0                   0
## 6           0           0   0           0                   0
```

Next we draw the heatmap using the **plotly** library.

```
#create color scheme

cols <- c("0" = "lightgray", "1" = "red", "2" = "lightskyblue", "3" = "purple
", "4" = "seagreen")


data=as.matrix(t(heatmap[,c(4:21)]))

plot_ly(x=heatmap$platform, y=colnames(heatmap[,c(4:21)]), z = data, type = "
heatmap", color=c(0,1,2,3,4), colors=cols, showscale = FALSE, xgap=1, ygap=1)
%>%

    layout(
        title = "<b>Heat map showing who provides what data</b>",
        width=800,
        xaxis=list(tickfont = list(size = 10), tickangle = 45),
        margin = list(l = 60, r = 10, b = 100, t = 30, pad = 4),
        showlegend = FALSE)
```

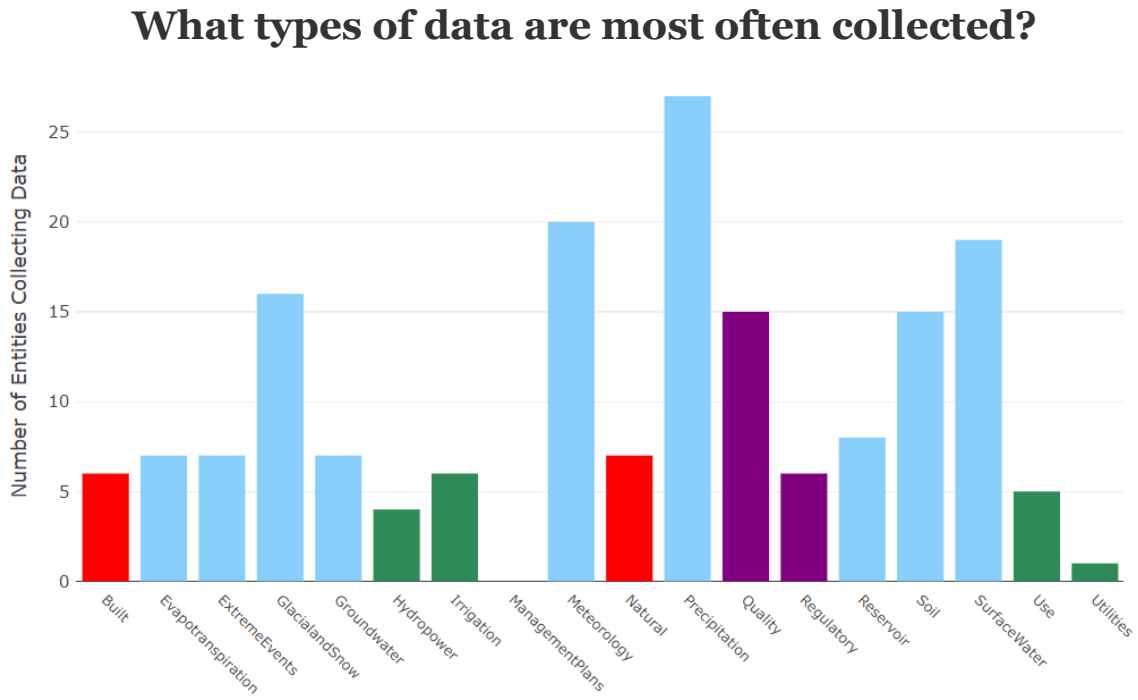
Heat map showing who provides what data



The following script summarizes the rows in the heat map to show the number of platforms collecting certain types of water data.

```
#summarize each of the columns
#make presence a 1 (not 2, 3, or 4)
heatmap2 <- heatmap
heatmap2[heatmap2>=2] <- 1

nTypes <- heatmap2 %>%  summarise_if(is.numeric, sum, na.rm = TRUE)
nTypes <- t(nTypes) %>% as.data.frame();

#set row names to column
colnames(nTypes) <- c("count")
nTypes <- tibble::rownames_to_column(nTypes, "name")

#graph - unfortunately plotly automatically re-orders charge alphabetically.
It is a known bug in the program.
plot_ly(
  x = nTypes$name,
  y = nTypes$count,
  #color = nTypes$color,
  marker = list(color = c("red","red","purple","purple","lightskyblue",
    "lightskyblue","lightskyblue","lightskyblue","lightskyblue",
    "lightskyblue", "lightskyblue","lightskyblue","lightskyblue","seagreen",
  "seagreen", "seagreen", "seagreen", "seagreen")),
```

```
  type = "bar",
  orientation = "vertical"
) %>%
  layout(width=800,
         title = "<b>What types of data are most often collected?</b>",
         yaxis = list(title = 'Number of Entities Collecting Data'),
         xaxis=list(tickfont = list(size = 10), tickangle = 45),
         margin = list(l = 50, r = 10, b = 90, t = 30, pad = 4),
         showlegend = FALSE)
```

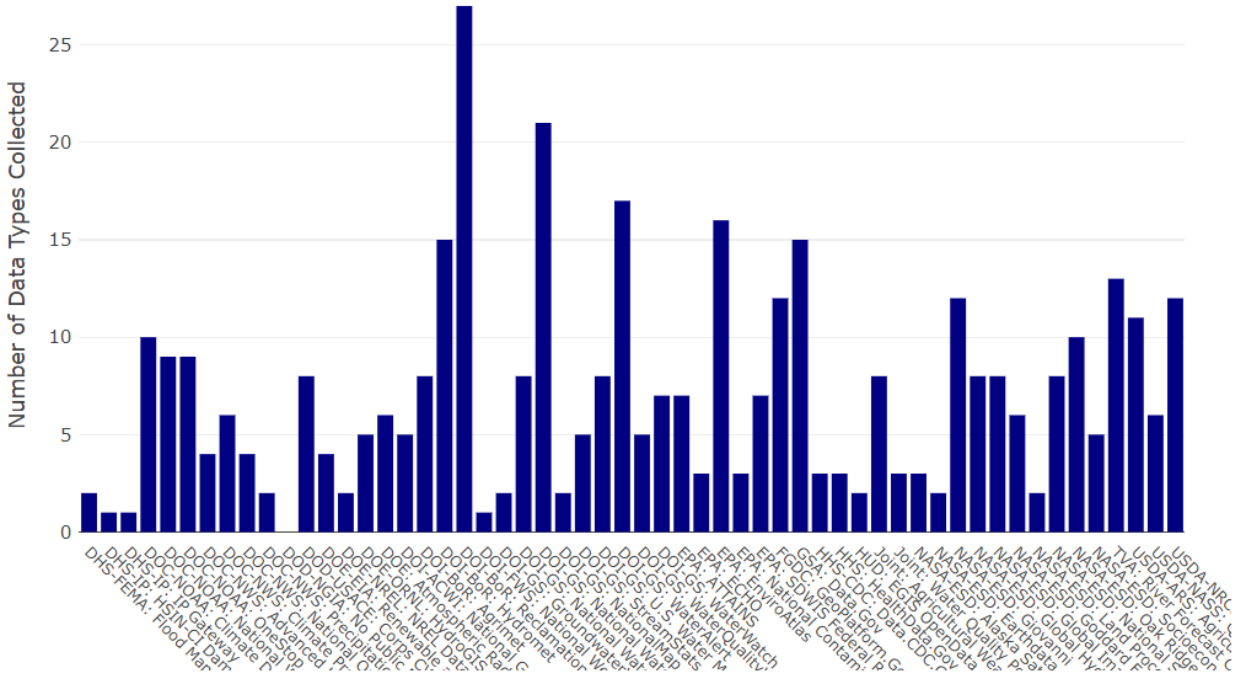


The following script summarizes the columns in the heat map to show the number of water data types collected by each platform.

```
#summarize each of the rows
nEntities <- heatmap %>% mutate(count = rowSums(.[4:21]))

#graph - unfortunately plotly automatically re-orders charge alphabetically.
It is a known bug in the program.
plot_ly(
  x = nEntities$platform,
  y = nEntities$count,
  marker = list(color = c("navy")),
  type = "bar",
  orientation = "vertical"
) %>%
  layout(width = 800,
    title = "<b>Which platforms provide the greatest variety of water-related
```

```
  data?</b>",
yaxis = list(title = 'Number of Data Types Collected'),
xaxis=list(tickfont = list(size = 10), tickangle = 45),
margin = list(l = 50, r = 30, b = 90, t = 30, pad = 4),
showlegend = FALSE)
```

Which platforms provide the greatest variety of water data?



Creating Scores for Discoverability, Accessibility, and Usability

The following scripts will create scores based on discoverability, accessibility, and usability characteristics of the data platform. Descriptions and reasoning behind the scores can be found in the white paper: *Inventory of Federal and State Water Data*.

Discoverability – how easy was it to find the data?

```
#create score for ease of discovery ----------------------------------------
scoreNodes <- platformNodes;
#make all lowercase
scoreNodes$easeDiscover <- tolower(scoreNodes$easeDiscover)
#find unique
unique(scoreNodes$easeDiscover); #should be no, low, medium, or high
```

```
## [1] "no"     "low"    "medium" "high"
```

```
#create score
scoreNodes$scoreEaseDiscover = NA;
  scoreNodes$scoreEaseDiscover = ifelse(scoreNodes$easeDiscover == "no" |
    scoreNodes$easeDiscover=="unknown" | scoreNodes$easeDiscover=="none", 0,
    scoreNodes$scoreEaseDiscover);

  scoreNodes$scoreEaseDiscover = ifelse(scoreNodes$easeDiscover == "low", 1,
    scoreNodes$scoreEaseDiscover);

  scoreNodes$scoreEaseDiscover = ifelse(scoreNodes$easeDiscover == "medium",
    2, scoreNodes$scoreEaseDiscover);

  scoreNodes$scoreEaseDiscover = ifelse(scoreNodes$easeDiscover == "high", 3,
    scoreNodes$scoreEaseDiscover);

#check to see all values scored
table(scoreNodes$scoreEaseDiscover, useNA = "ifany");
```

```
##  0  1  2  3
##  3 13 20 20
```

Discoverability – how were data found?

```
#create score for method to find data --------------------------------------
#make all lowercase
scoreNodes$methodDiscover <- tolower(scoreNodes$methodDiscover);

#find unique
```

```
unique(scoreNodes$methodDiscover);
```

```
## [1] "unknown"                "website"                "map"
## [4] "catalog"                "map; website"           "map; catalog"
## [7] "map; catalog; website" "website; catalog"
```

```
#create score
scoreNodes$scoreMethodDiscover = 0;

#loop through and separate by "; " to find all the values.
We will use the mean score.
for (i in 1:dim(scoreNodes)[1]){
  foo = as.list(strsplit(as.character(scoreNodes$methodDiscover[i]),"; "));
  foo = as.data.frame(foo); colnames(foo) = "value";
  foo$score = 0;
  foo$score = ifelse(foo$value=="website",1, foo$score);
  foo$score = ifelse(foo$value=="map",2, foo$score);
  foo$score = ifelse(foo$value=="catalog",2, foo$score);

  scoreNodes$scoreMethodDiscover[i] = round(mean(foo$score),2);
}

#check to make sure all nodes scored
table(scoreNodes$scoreMethodDiscover, useNA = "ifany");
```

```
##    0    1  1.5 1.67    2
##    3   14    2    1   36
```

Accessibility – how easy was it to obtain the data?

```
# create score for ease of access  ----------------------------------------
#make all lowercase
scoreNodes$easeAccess <- tolower(scoreNodes$easeAccess);

#find unique values
unique(scoreNodes$easeAccess);
```

```
## [1] "permission required"   "software required"    "training required"
## [4] "no"                    "yes"                  "registration required"
```

```
#create score
```

```
scoreNodes$scoreEaseAccess = NA;

scoreNodes$scoreEaseAccess <- ifelse(scoreNodes$easeAccess=="unknown" |
  scoreNodes$easeAccess=="no", 0, scoreNodes$scoreEaseAccess)

scoreNodes$scoreEaseAccess <- ifelse(scoreNodes$easeAccess ==
  "software required" | scoreNodes$easeAccess=="training required" |
  scoreNodes$easeAccess == "permission required", 1,
  scoreNodes$scoreEaseAccess)

scoreNodes$scoreEaseAccess <- ifelse(scoreNodes$easeAccess ==
  "registration required", 3, scoreNodes$scoreEaseAccess)  #assumes
   permission granted

scoreNodes$scoreEaseAccess <- ifelse(scoreNodes$easeAccess=="yes", 3,
  scoreNodes$scoreEaseAccess)

#Check to make sure all platforms are scored
table(scoreNodes$scoreEaseAccess, useNA = "ifany");
```

```
##  0  1  3
##  1  3 52
```

Accessibility – how were data obtained?

```
# create score for method of access ----------------------------------------
#make all lower case
scoreNodes$methodAccess <- tolower(scoreNodes$methodAccess)

#find unique values
unique(scoreNodes$methodAccess)
```

```
##  [1] "unknown"
##  [2] "individual export; web services"
##  [3] "batch export"
##  [4] "copy and paste"
##  [5] "individual export"
##  [6] "web services; individual export; full export"
##  [7] "web services; individual export"
##  [8] "individual export; link to source"
##  [9] "individual export; web services; batch export"
## [10] "link to source"
## [11] "web services; copy and paste"
## [12] "ftp; individual export; batch export; web services"
## [13] "individual export; batch export"
## [14] "individual export; batch export; web services"
## [15] "web services; ftp; batch export"
## [16] "web services; batch export"
```

```
## [17] "web services; link to source"
## [18] "individual export; web services; link to source"
## [19] "web services; individual export; batch export"
## [20] "web services; batch export; individual export"
## [21] "link to source; individual export"
## [22] "copy and paste; individual export"
## [23] "web services; ftp; individual export; batch export"
## [24] "web services; ftp; link to source; individual export"
## [25] "individual export; copy and paste"
## [26] "individual export; ftp"
## [27] "ftp; individual export"
```

```
#create score
scoreNodes$scoreMethodAccess = NA;
for (i in 1:dim(scoreNodes)[1]){
  # split string using ;
  foo = as.list(strsplit(as.character(scoreNodes$methodAccess[i]),"; "));

  #set up data frame
  foo = as.data.frame(foo); colnames(foo) = "value";
    foo$score = NA;

    foo$score = ifelse(foo$value=="unknown" | foo$value=="none",0,foo$score);

    foo$score = ifelse(foo$value=="link to source" | foo$value ==
      "copy and paste",1, foo$score);

    foo$score = ifelse(foo$value=="individual export",2, foo$score);

    foo$score = ifelse(foo$value=="batch export" | foo$value=="full export",
      3, foo$score);

    foo$score = ifelse(foo$value=="ftp", 4, foo$score);

    foo$score = ifelse(foo$value=="web services",5, foo$score);

  #take the maximum score
  scoreNodes$scoreMethodAccess[i] = round(sum(foo$score)/dim(foo)[1],2);
}

#check to make sure all platforms are scored
table(scoreNodes$scoreMethodAccess, useNA = "ifany")
#subset(scoreNodes, is.na(scoreMethodAccess)==TRUE) %>% as.data.frame() #make
sure not missing any
```

```
##    0    1  1.5    2  2.5 2.67    3 3.33  3.5    4
##    5    6    4    5    2    1    8    6   17    2
```

Interoperability – Are the file formats machine readable?

```
#create score for file type  ------------------------------------------------
#make all lower case
scoreNodes$fileFormat <- tolower(scoreNodes$fileFormat);

#find unique values
unique(scoreNodes$fileFormat)
```

```
##  [1] "unknown"
##  [2] "pdf; software specific; shapefile"
##  [3] "csv; png; image"
##  [4] "netcdf"
##  [5] "excel"
##  [6] "excel; ascii; zip"
##  [7] "access; excel; shapefile"
##  [8] "csv; tsv; zip"
##  [9] "html"
## [10] "txt; excel; json"
## [11] "json; csv; wml2; zip"
## [12] "shapefile"
## [13] "shapefile; txt"
## [14] "csv; pdf; shapefile"
## [15] "csv; json"
## [16] "shapefile; csv; json"
## [17] "png; csv; html"
## [18] "not applicable"
## [19] "shapefile; json; csv; xml; txt; html"
## [20] "raster; shapefile"
## [21] "shapefile; csv"
## [22] "shapefile; excel; csv"
## [23] "shapefile; csv; zip"
## [24] "zip; txt"
## [25] "csv; shapefile; tsv; excel; kml"
## [26] "csv; json; xml; rdf"
## [27] "csv; rdf; xml"
## [28] "shapefile; excel"
## [29] "image; pdf; shapefile; raster; hdf5; csv"
## [30] "netcdf; hdf5"
## [31] "ascii; xml; hdf5"
## [32] "netcdf; raster; shapefile"
## [33] "pdf"
## [34] "html; csv; excel"
## [35] "csv; html"
## [36] "csv; txt; zip; pdf"
## [37] "shapefile; csv; xml; txt; kmz"
## [38] "netcdf; shapefile; raster; kmz"
## [39] "raster; shapefile; html; png"
## [40] "netcdf; raster; shapefile; zip"
## [41] "html; csv; xml; ascii; shapefile"
## [42] "csv"
```

```
#create score
scoreNodes$scoreFileFormat = NA

#loop through and separate by ";" will then take the mean of the file formats
for (i in 1:dim(scoreNodes)[1]){
  foo = as.list(strsplit(as.character(scoreNodes$fileFormat[i]),"; "));
    foo = as.data.frame(foo); colnames(foo) = "value";

  foo$score = NA;
  foo$score = ifelse(foo$value=="unknown" | foo$value=="not applicable" |
    foo$value=="none",0, foo$score);

  foo$score = ifelse(foo$value=="pdf" | foo$value=="png" | foo$value=="image"
    | foo$value=="jpg" | foo$value=="zip",1, foo$score);

  foo$score = ifelse(foo$value=="access" | foo$value=="excel" | foo$value ==
    "raster" | foo$value=="shapefile",3, foo$score);

  foo$score = ifelse(foo$value=="word" | foo$value=="geodatabase" |
    foo$value=="kmz" | foo$value=="software specific",3, foo$score);

  foo$score = ifelse(foo$value=="csv" | foo$value=="json" | foo$value=="txt"
    | foo$value == "xml",5, foo$score);

  foo$score = ifelse(foo$value=="ascii" | foo$value=="netcdf" | foo$value==
    "html" | foo$value == "hdf5",5, foo$score);

  foo$score = ifelse(foo$value=="kml" | foo$value=="geojson" | foo$value ==
    "wml2" | foo$value == "rdf" | foo$value == "tsv",5, foo$score);

  scoreNodes$scoreFileFormat[i] = round(max(foo$score, is.na=FALSE),2);
}
#check to make sure all platforms are scored
table(scoreNodes$scoreFileFormat, useNA = "ifany")
#subset(scoreNodes, is.na(scoreFileFormat)==TRUE)  #if find a bug (such as a
last ; with nothing after it - go into original file and fix)
```

```
##  0  1  3  5
## 12  1  7 36
```

Interoperability – Are the metadata in a machine-readable format?

```
#create score for metadata ------------------------------------------------
#make all lower case
scoreNodes$metadataFormat <- tolower(scoreNodes$metadataFormat)

#find unique values
unique(scoreNodes$metadataFormat)
```

```
## [1] "unknown" "xml"     "html"    "no"      "in data" "pdf"     "txt"
## [8] "json"
```

```
#create values for machine read and for non-machine read
no.machineread <- c("in data", "pdf")
prop.machineread <- c("word", "excel")
yes.machineread <- c("xml","html","json","txt", "csv")

#score
scoreNodes$scoreMetadataFormat <- NA

for (i in 1:dim(scoreNodes)[1]){
  foo = as.list(strsplit(as.character(scoreNodes$metadataFormat[i]),"; "));
  foo = as.data.frame(foo); colnames(foo) = "value";
  foo$score = NA;

  foo$score = ifelse(foo$value=="no" | foo$value=="broken link",0,foo$score);
  foo$score = ifelse(foo$value %in% no.machineread, 1, foo$score);
  foo$score = ifelse(foo$value %in% prop.machineread, 2, foo$score);
  foo$score = ifelse(foo$value %in% yes.machineread, 3, foo$score);

  scoreNodes$scoreMetadataFormat[i] = round(mean(foo$score, is.na=FALSE),2);
}

#adjust for whether there was metadata to start with or if unknown
scoreNodes$scoreMetadataFormat <- ifelse(scoreNodes$metadataFormat ==
  "unknown" & scoreNodes$metadata=="No", 0, scoreNodes$scoreMetadataFormat)
scoreNodes$scoreMetadataFormat <- ifelse(scoreNodes$metadataFormat ==
  "unknown" & scoreNodes$metadata=="Yes", 1, scoreNodes$scoreMetadataFormat)
scoreNodes$scoreMetadataFormat <- ifelse(scoreNodes$metadataFormat ==
  "unknown" & scoreNodes$metadata=="Unknown", 1,
   scoreNodes$scoreMetadataFormat)

#check to make sure all platforms are scored
table(scoreNodes$scoreMetadataFormat, useNA = "ifany")
#subset(scoreNodes, is.na(scoreMetadataFormat)==TRUE)  #if find a bug (such a
s a last ; with nothing after it - go into original file and fix)
```

```
##  0  1  3
## 13 16 27
```

Interoperability – is there a glossary for the data?

```
#create score for metadata ------------------------------------------------
#make all lower case
```

```
scoreNodes$dataDefinitions <- tolower(scoreNodes$dataDefinitions)

#find unique values
unique(scoreNodes$dataDefinitions)
```

```
## [1] "unknown" "yes"     "no"      "some"
```

```
#score
scoreNodes$scoreDataDefinitions <- NA
scoreNodes$scoreDataDefinitions <- ifelse(scoreNodes$dataDefinitions == "no"
  | scoreNodes$dataDefinitions=="unknown", 0,scoreNodes$scoreDataDefinitions)

scoreNodes$scoreDataDefinitions <- ifelse(scoreNodes$dataDefinitions ==
  "some", 1, scoreNodes$scoreDataDefinitions)

scoreNodes$scoreDataDefinitions <- ifelse(scoreNodes$dataDefinitions ==
  "yes", 2, scoreNodes$scoreDataDefinitions)

table(scoreNodes$scoreDataDefinitions, useNA = "ifany")
```

```
##  0  1  2
## 21  4 31
```

Interoperability – what types of metadata are present?

```
#create score for metadata ------------------------------------------------------
#metadata will be composed of 4 attributes - not the yes/no present question
scoreNodes$metaAttributes <- tolower(scoreNodes$metaAttributes)

#find unique values
unique(scoreNodes$metaAttributes)
```

```
## [1] "unknown"
## [2] "administrative; descriptive; structural"
## [3] "no"
## [4] "administrative"
## [5] "descriptive; structural"
## [6] "descriptive"
## [7] "administrative; descriptive"
```

```
#create score
scoreNodes$scoreMetaAttributes = NA
```

```
#loop through and separate by ";" - will then take the mean of the file formats
for (i in 1:dim(scoreNodes)[1]){
  foo = as.list(strsplit(as.character(scoreNodes$metaAttributes[i]),"; "));
  foo = as.data.frame(foo); colnames(foo) = "value";

  foo$score = NA;
  foo$score = ifelse(foo$value=="unknown" | foo$value=="no",0, foo$score);
  foo$score = ifelse(foo$value=="administrative" | foo$value=="structural" |
    foo$value=="descriptive", 1, foo$score);

  scoreNodes$scoreMetaAttributes[i] = round(sum(foo$score, is.na=FALSE),2);
}

#check to make sure platform scored
table(scoreNodes$scoreMetaAttributes, useNA = "ifany")
#subset(scoreNodes, is.na(scoreMetaAttributes)==TRUE)  #if find a bug (such as a last ; with nothing after it - go into original file and fix)
```

```
##  0  1  2  3
## 14 13 12 17
```

Interoperability – are metadata standards provided?

```
#create score for metadata ------------------------------------------------
#make all lower case
scoreNodes$metadataStandards <- tolower(scoreNodes$metadataStandards)

#find unique values
unique(scoreNodes$metadataStandards)
```

```
## [1] "unknown" "no"      "yes"     "some"
```

```
#create score
scoreNodes$scoreMetadataStandards <- NA
scoreNodes$scoreMetadataStandards <- ifelse(scoreNodes$metadataStandards ==
  "no" | scoreNodes$metadataStandards=="unknown", 0,
  scoreNodes$scoreMetadataStandards)
scoreNodes$scoreMetadataStandards <- ifelse(scoreNodes$metadataStandards ==
  "some", 1, scoreNodes$scoreMetadataStandards)
scoreNodes$scoreMetadataStandards <- ifelse(scoreNodes$metadataStandards ==
  "yes", 2, scoreNodes$scoreMetadataStandards)

#check to make sure platforms scored
table(scoreNodes$scoreMetadataStandards, useNA = "ifany")
```

```
##  0  1  2
## 47  2  7
```

Interoperability – how often are data being updated?

```
#create score for timeliness ------------------------------------------------
#make all lower case
scoreNodes$timeliness <- tolower(scoreNodes$timeliness)

#find unique values
unique(scoreNodes$timeliness)
```

```
## [1] "unknown"          "irregular"        "daily or higher"
## [4] "yearly or higher"
```

```
#create score
scoreNodes$scoreTimeliness <- NA
scoreNodes$scoreTimeliness = ifelse(scoreNodes$timeliness=="unknown", 0,
  scoreNodes$scoreTimeliness);
scoreNodes$scoreTimeliness = ifelse(scoreNodes$timeliness=="irregular" |
  scoreNodes$timeliness=="varies", 1, scoreNodes$scoreTimeliness);
scoreNodes$scoreTimeliness = ifelse(scoreNodes$timeliness==
  "yearly or higher", 2, scoreNodes$scoreTimeliness);
scoreNodes$scoreTimeliness = ifelse(scoreNodes$timeliness==
  "monthly or higher" | scoreNodes$timeliness=="weekly or higher", 3,
  scoreNodes$scoreTimeliness);
scoreNodes$scoreTimeliness = ifelse(scoreNodes$timeliness=="daily or higher",
  4, scoreNodes$scoreTimeliness);

#check to make sure platforms scored
table(scoreNodes$scoreTimeliness, useNA = "ifany");
#subset(scoreNodes, is.na(timeliness)==TRUE)
```

Interoperability – how much of the data are available?

```
#create score for length of record ------------------------------------------
#make all lower case
scoreNodes$lengthAvailable <- tolower(scoreNodes$lengthAvailable)
#find unique values
unique(scoreNodes$lengthAvailable)
```

```
## [1] "current only"     "unknown"          "limited record"
```

```
## [4] "period of record" "varies"
```

```
#set score
scoreNodes$scoreLengthAvailable = NA;
scoreNodes$scoreLengthAvailable = ifelse(scoreNodes$lengthAvailable==
  "unknown",0, scoreNodes$scoreLengthAvailable);
scoreNodes$scoreLengthAvailable = ifelse(scoreNodes$lengthAvailable==
  "not applicable" | scoreNodes$lengthAvailable=="varies", 1,
  scoreNodes$scoreLengthAvailable);
scoreNodes$scoreLengthAvailable = ifelse(scoreNodes$lengthAvailable==
  "current only", 2, scoreNodes$scoreLengthAvailable);
scoreNodes$scoreLengthAvailable = ifelse(scoreNodes$lengthAvailable==
  "limited record", 3, scoreNodes$scoreLengthAvailable);
scoreNodes$scoreLengthAvailable = ifelse(scoreNodes$lengthAvailable==
  "period of record", 4, scoreNodes$scoreLengthAvailable);

#check to make sur eplatforms scored
table(scoreNodes$scoreLengthAvailable, useNA = "ifany");
#subset(scoreNodes, is.na(lengthAvailable)==TRUE)
```

Convert Scores to a Scale from 0 (low) to 100 (high)

```
#############################################################################
#total score for discoverable, accessible, usable
scoreNodes$discoverTotal = scoreNodes$scoreEaseDiscover +
  scoreNodes$scoreMethodDiscover;

scoreNodes$accessTotal = scoreNodes$scoreEaseAccess +
  scoreNodes$scoreMethodAccess;

scoreNodes$usableTotal = scoreNodes$scoreFileFormat +
  scoreNodes$scoreMetadataFormat + scoreNodes$scoreDataDefinitions +
  scoreNodes$scoreMetaAttributes + scoreNodes$scoreMetadataStandards +
  scoreNodes$scoreTimeliness + scoreNodes$scoreLengthAvailable

scoreNodes$scoreTotal <- scoreNodes$discoverTotal + scoreNodes$accessTotal +
scoreNodes$usableTotal;

#max score for each variable
easeDiscoverMax = 3;   methodDiscoverMax = 2;
totalDiscoverMax = easeDiscoverMax + methodDiscoverMax;

easeAccessMax = 3;     methodAccessMax = 5;
totalAccessMax = easeAccessMax + methodAccessMax;

fileFormatMax = 5;      metadataFormatMax = 3;     dataDefinitionsMax = 2;
metaAttributesMax = 3;  metadataStandardsMax = 2;  timelinessMax = 4;
lengthAvailableMax = 4;
```

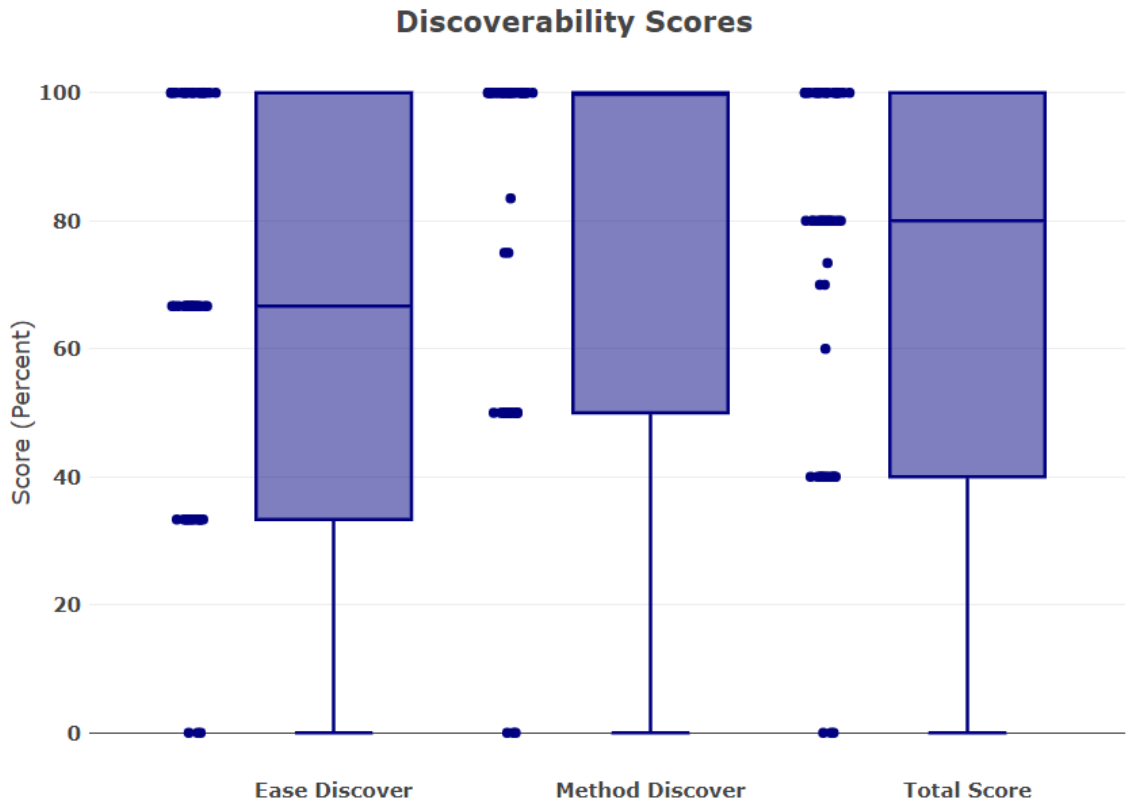
```
totalUseMax = fileFormatMax + metadataFormatMax + dataDefinitionsMax +
  metaAttributesMax + metadataStandardsMax + timelinessMax +
  lengthAvailableMax;

totalMax = totalDiscoverMax + totalAccessMax + totalUseMax;
################################################################################

################################################################################
# CONVERT SCORES TO PERCENT
################################################################################
percent = scoreNodes;

percent$scoreEaseDiscover = round(scoreNodes$scoreEaseDiscover /
  easeDiscoverMax*100,2);
percent$scoreMethodDiscover = round(scoreNodes$scoreMethodDiscover /
  methodDiscoverMax*100,2);

percent$scoreEaseAccess = round(scoreNodes$scoreEaseAccess/easeAccessMax*100,
  2);
percent$scoreMethodAccess = round(scoreNodes$scoreMethodAccess /
  methodAccessMax*100,2);

percent$scoreFileFormat = round(scoreNodes$scoreFileFormat/fileFormatMax*100,
  2);
percent$scoreMetadataFormat = round(scoreNodes$scoreMetadataFormat /
  metadataFormatMax*100,2);
percent$scoreDataDefinitions = round(scoreNodes$scoreDataDefinitions /
  dataDefinitionsMax*100,2);
percent$scoreMetaAttributes = round(scoreNodes$scoreMetaAttributes /
  metaAttributesMax*100,2);
percent$scoreMetadataStandards = round(scoreNodes$scoreMetadataStandards /
  metadataStandardsMax*100,2);
percent$scoreTimeliness = round(scoreNodes$scoreTimeliness/timelinessMax*100,
  2);
percent$scoreLengthAvailable = round(scoreNodes$scoreLengthAvailable /
  lengthAvailableMax*100,2);

percent$discoverTotal = round(scoreNodes$discoverTotal/totalDiscoverMax*100,
  2);
percent$accessTotal = round(scoreNodes$accessTotal/totalAccessMax*100,2);
percent$usableTotal = round(scoreNodes$usableTotal/totalUseMax*100,2);

percent$scoreTotal = round(scoreNodes$scoreTotal/totalMax*100, 2);
#summary(percent)
```

Create boxplots of scores

Discoverability Boxplot

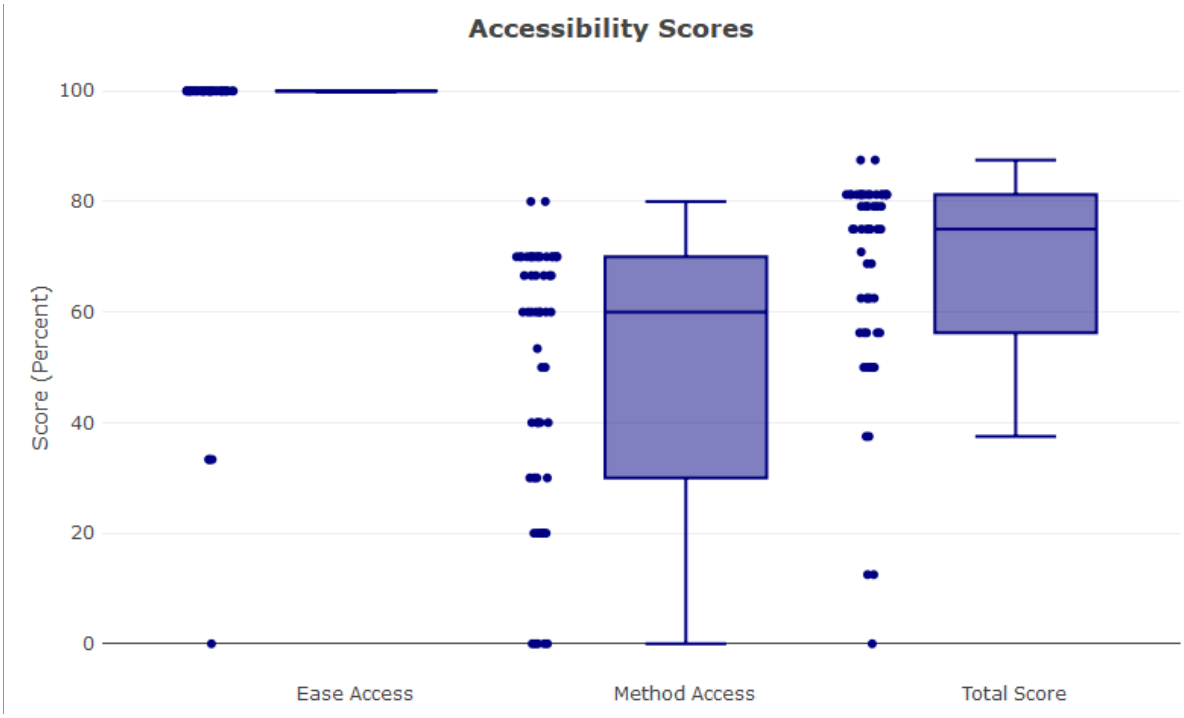
```
#create a boxplot of discovaerability / findability
plot_ly(y = percent$scoreEaseDiscover, x="Ease Discover", type = "box",
```

```
        boxpoints = "all", jitter = 0.3, pointpos = -1.8,
          marker = list(color = 'navy'),
          line = list(color = 'navy'),
          name = "Ease Discover") %>%
  add_trace(y = percent$scoreMethodDiscover, x= "Method Discover",
          marker = list(color = 'navy'),
          line = list(color = 'navy'),
          name = "Method Discover") %>%
  add_trace(y = percent$discoverTotal, x="Total Score",
          marker = list(color = 'navy'),
          line = list(color = 'navy'),
          name = "Total Score") %>%
  layout(xaxis=list(tickfont = list(size = 12)),
        yaxis=list(title="Score (Percent)"),
        margin = list(l = 50, r = 10, b = 30, t = 30, pad = 4),
        title = "<b>Discoverability Scores</b>",
        showlegend = FALSE)
```



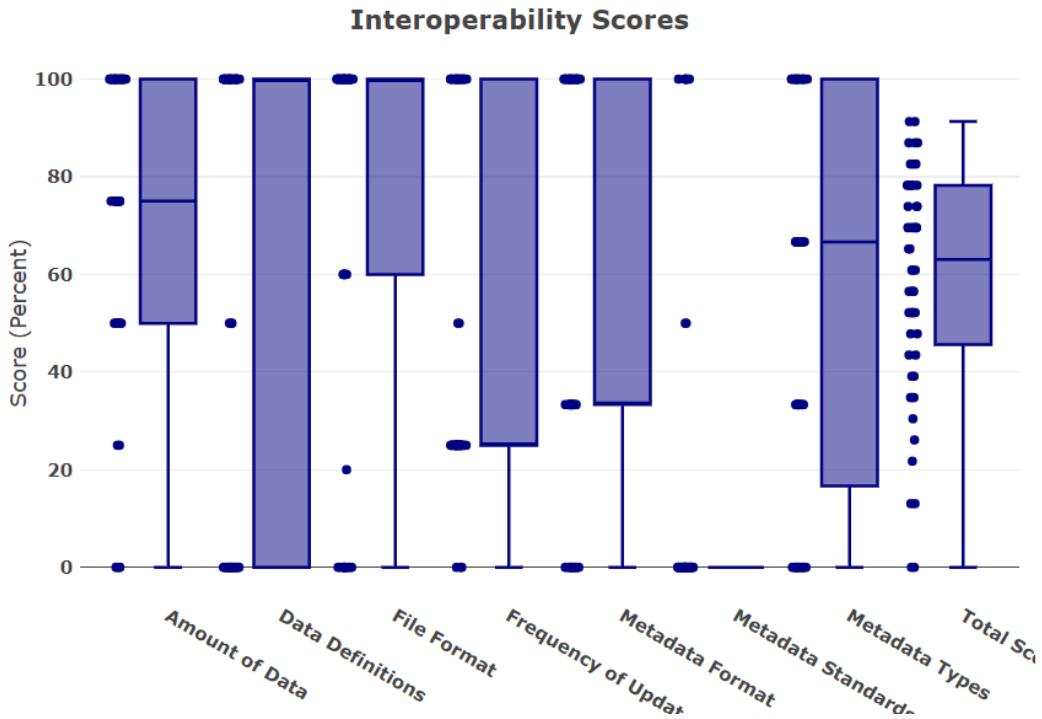

```
#create a boxplot of accessibility
plot_ly(y = percent$scoreEaseAccess, x="Ease Access", type = "box",
        boxpoints = "all", jitter = 0.3, pointpos = -1.8,
          marker = list(color = 'navy'),
```

```
        line = list(color = 'navy'),
        name = "Ease Access") %>%
  add_trace(y = percent$scoreMethodAccess, x= "Method Access",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Method Access") %>%
  add_trace(y = percent$accessTotal, x="Total Score",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Total Score") %>%
  layout(xaxis=list(tickfont = list(size = 12)),
       yaxis=list(title="Score (Percent)"),
       margin = list(l = 50, r = 10, b = 30, t = 30, pad = 4),
       title = "<b>Accessibility Scores</b>",
       showlegend = FALSE)
```




```
#create a boxplot of interoperability / usability
plot_ly(y = percent$scoreFileFormat, x="File Format", type = "box",
        boxpoints = "all", jitter = 0.3, pointpos = -1.8,
          marker = list(color = 'navy'),
```

```
        line = list(color = 'navy'),
        name = "File Format") %>%
  add_trace(y = percent$scoreMetadataFormat, x= "Metadata Format",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Metadata Format") %>%
  add_trace(y = percent$scoreDataDefinitions, x= "Data Definitions",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Data Definitions") %>%
  add_trace(y = percent$scoreMetaAttributes, x= "Metadata Types",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Metadata Types") %>%
  add_trace(y = percent$scoreMetadataStandards, x= "Metadata Standards",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Metadata Standards") %>%
  add_trace(y = percent$scoreTimeliness, x= "Frequency of Updates",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Frequency of Updates") %>%
  add_trace(y = percent$scoreLengthAvailable, x= "Amount of Data",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Amount of Data") %>%
  add_trace(y = percent$usableTotal, x="Total Score",
        marker = list(color = 'navy'),
        line = list(color = 'navy'),
        name = "Total Score") %>%
  layout(xaxis=list(tickfont = list(size = 12)),
       yaxis=list(title="Score (Percent)"),
       margin = list(l = 50, r = 10, b = 30, t = 30, pad = 4),
       title = "<b>Interoperability Scores</b>",
       showlegend = FALSE)
```



The files can be saved by:
write.csv(percent, paste0(swd,folderName,fileName,"_percent.csv"), row.names=FALSE).