Automating map generation for Montana Fisheries data

abock@usgs.gov

This document is a summary of work and code being developed for the CDI project "Automated Mapping of SB OBjects"

Loading Libraries

R libraries required - *sbtools*, *sp*, *gdalUtils*, *RCurl*, *RColorBrewer*, *maps*, *mapdata* Functions are a part of package specificed (*sbtools::item_get item_get* is function in *sbtools* library) Functions utilizing the libraries *maps* and *mapdata* need to have libraries loaded to access the databases

```
# sb tools is the CIDA developed ScienceBase access library
#suppressWarnings(suppressMessages(library(sbtools)))
suppressWarnings(suppressMessages(library(maps)))
suppressWarnings(suppressMessages(library(mapdata)))
```

Calling up items from ScienceBase

I've set up a test page with Roy's fish data. To access the information for the SBase item, use the following code which utilizes an items Science Base ID **57115024e4b0ef3b7ca554f3**. The function *list_files* shows the data files associated with the SB item as well as the download urls.

```
test_item=sbtools::item_get("57115024e4b0ef3b7ca554f3")
```

```
## Setting endpoint to www.sciencebase.gov
```

```
names(test_item)
```

```
##  [1] "link"              "relatedItems"      "id"
##  [4] "title"             "provenance"        "hasChildren"
##  [7] "parentId"          "browseCategories"  "browseTypes"
## [10] "systemTypes"       "facets"            "files"
## [13] "distributionLinks" "previewImage"
```

```
#parent<-item_get(test_item$parentId)
#item_list_children(parent)
sbtools::item_list_files(test_item)
```

```
##                              fname    size
## 1 StreamflowChange_SampleSites.csv  995792
## 2                  ECHAM5_2055.csv 1079022
##
## 1 https://www.sciencebase.gov/catalog/file/get/57115024e4b0ef3b7ca554f3?f=__disk__8b%2Fbd%2F9f%2F8bb
## 2 https://www.sciencebase.gov/catalog/file/get/57115024e4b0ef3b7ca554f3?f=__disk__4a%2F05%2F6b%2F4a0
```

The function *item_get_wfs* retrieves the web feature service that is featured on the SB item's front page. The map can then be displayed.

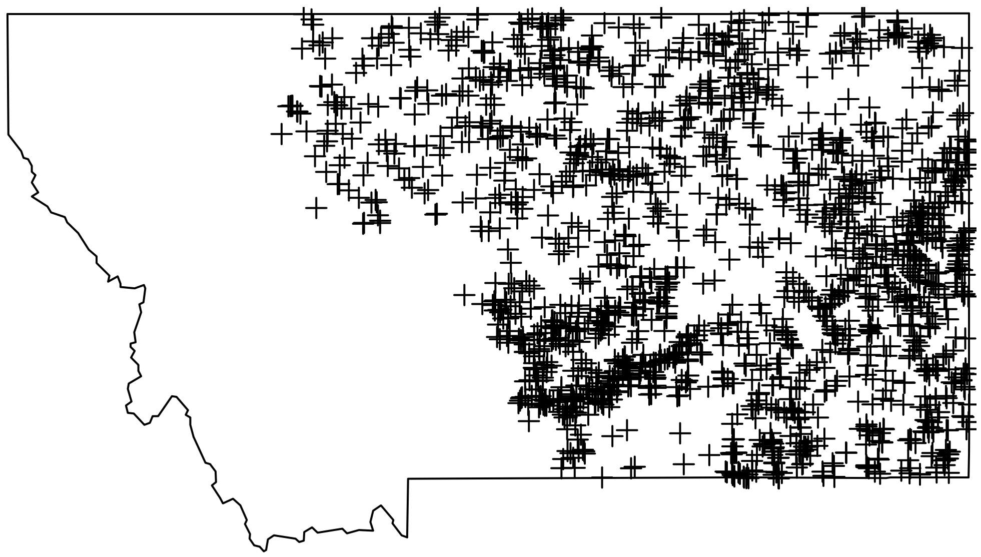
```
layer<-sbtools::item_get_wfs("57115024e4b0ef3b7ca554f3")
```

```
## Loading required namespace: rgdal

## Loading required namespace: xml2

## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\abock\AppData\Local\Temp\1\RtmpC2SVe3/file1c081fc6bb2", layer: "Fish_sites"
## with 1707 features
## It has 93 fields
```

```
map('state','montana')
sp::plot(layer,add=TRUE)
```



Retrieving SB Data

Next step is to retrieve the environmental data/variables and map them to the WFS. The function *getURL* from the *RCurl* library retrieves the URL for each file, and downloads and opens the file to a local directory.

```
sbfiles<-sbtools::item_list_files(test_item)
sites <- RCurl::getURL(sbfiles$url[1])
data <- RCurl::getURL(sbfiles$url[2])
sites2 <- read.csv(text=sites)
print (colnames(sites2))
```

```
##  [1] "Site_ID"   "CONTDA"    "Lat"       "Long"      "M1p25"    
##  [6] "M2p25"     "M3p25"     "M4p25"     "M5p25"     "M6p25"    
## [11] "M7p25"     "M8p25"     "M9p25"     "M10p25"    "M11p25"   
## [16] "M12p25"    "M1p50"     "M2p50"     "M3p50"     "M4p50"    
## [21] "M5p50"     "M6p50"     "M7p50"     "M8p50"     "M9p50"    
## [26] "M10p50"    "M11p50"    "M12p50"    "M1p75"     "M2p75"    
## [31] "M3p75"     "M4p75"     "M5p75"     "M6p75"     "M7p75"    
## [36] "M8p75"     "M9p75"     "M10p75"    "M11p75"    "M12p75"   
## [41] "M1mean"    "M2mean"    "M3mean"    "M4mean"    "M5mean"   
## [46] "M6mean"    "M7mean"    "M8mean"    "M9mean"    "M10mean"  
## [51] "M11mean"   "M12mean"   "S1minp25"  "S2minp25"  "S3minp25" 
## [56] "S4minp25"  "S1minp50"  "S2minp50"  "S3minp50"  "S4minp50" 
## [61] "S1maxp75"  "S2maxp75"  "S3maxp75"  "S4maxp75"  "S1maxmean"
## [66] "S2maxmean" "S3maxmean" "S4maxmean" "Aminp25"   "Aminp50"  
## [71] "Amaxp75"   "Amaxmean"  "S1p25"     "S2p25"     "S3p25"    
## [76] "S4p25"     "S1p50"     "S2p50"     "S3p50"     "S4p50"    
## [81] "S1p75"     "S2p75"     "S3p75"     "S4p75"     "S1mean"   
## [86] "S2mean"    "S3mean"    "S4mean"    "Ap25"      "Ap50"     
## [91] "Ap75"      "Amean"     "Arange"
```

```r
data2 <- read.csv(text=data)
print (dim(data2))
```

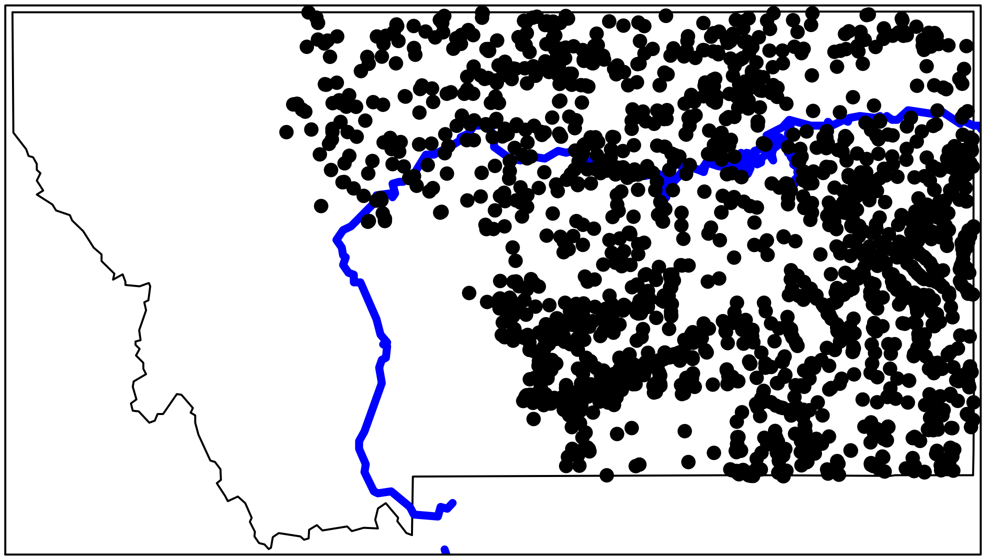
```
## [1] 1707   93
```

From the downloaded data, we can retrieve the latitude/longitude of the sites (same as shown in the WFS map) and give them proper nomenclature for OGC application and convert them to a spatial object with WGS 84 Geographic coordinate system.

```r
names(sites2) <- sub("Long", "x", names(sites2))
names(sites2) <- sub("Lat", "y", names(sites2))
sp::coordinates(sites2)<-~x+y
sp::proj4string(sites2)<-sp::CRS("+init=epsg:4326")
```

This next set of code plots the spatial objects. The R library *maps* and *mapdata* contain coarse resolution spatial data that can be used to supplement maps. Unforunately there doesn't seem to be an adequate WFS for NHD data that is medium resolution, but I'll keep looking for that.

```r
map('state','montana')
map('rivers',add=TRUE,col=4,lwd=4)
sp::plot(layer,pch=16,add=TRUE)
box(which="plot",lty="solid")
```



Setting up the plot

Next we set up the plotting. We can use the library *RColorBrewer* to automatically assign a 4-color symbology; alternatively this is something we can decide ourselves (such as in Roy's example maps). Then accessing the properties of one of environmental variables, bound these colors by the min/max and three quartiles.

```
#colPal<-c("#00A4DE","#5DFC21","#FFD701","#FF3300")
colPal<-RColorBrewer::brewer.pal(4,"Set1")
print(colPal)
```

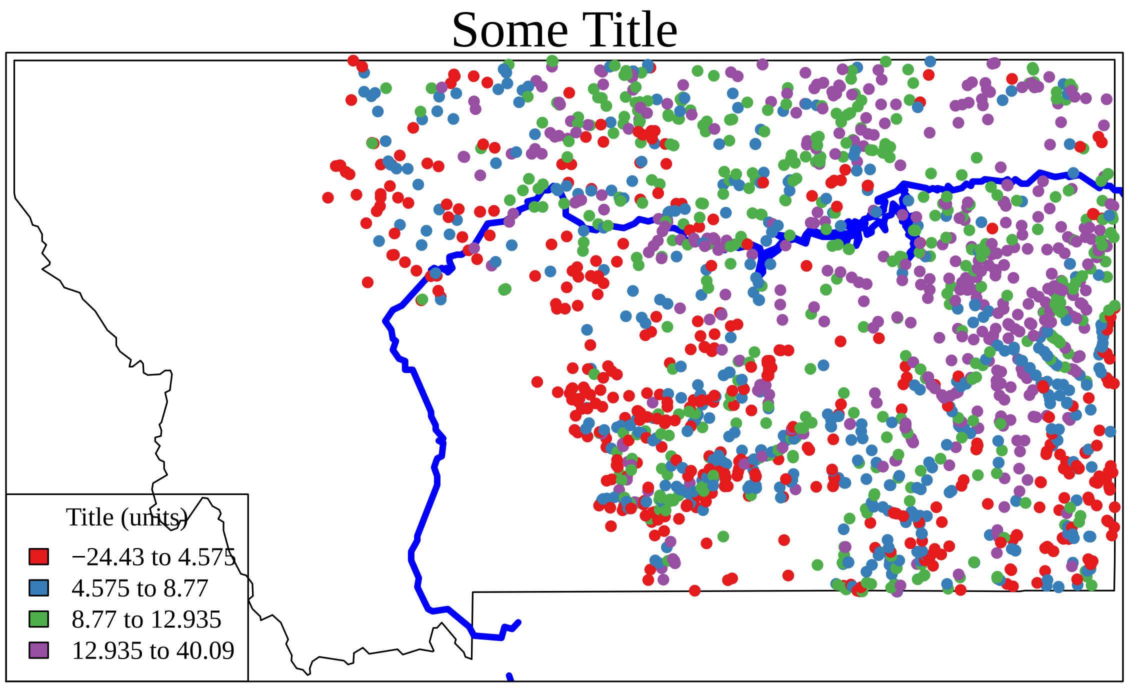
```
## [1] "#E41A1C" "#377EB8" "#4DAF4A" "#984EA3"
```

```
#fixedBreaks=c(-30,-15,0,15,30)
fixedBreaks=c(min(sites2$M2p25), quantile(sites2$M2p25,.25),median(sites2$M2p25),quantile(sites2$M2p25,
symb<-cut(sites2$M2p25,breaks=fixedBreaks,include.lowest=TRUE,right=TRUE)
```

This block of code adds a title and legends to the graph. We can also set the font to the more formal looking serif family.

```
op<-par(family="serif")
map('state','montana')
map('rivers',add=TRUE,col=4,lwd=4)
sp::plot(layer,col=colPal[symb],pch=16,add=TRUE)
```

```
box(which="plot",lty="solid")
mtext("Some Title",cex=2,line=0)
cutsChar<-as.character(symb)
cuts<-as.numeric(levels(factor (fixedBreaks)))
mapLegend = c(paste(cuts[1]," to ",cuts[2],sep=""),paste(cuts[2]," to ",cuts[3],sep=""),paste(cuts[3],"
                  paste(cuts[4]," to ",cuts[5],sep=""))
legend("bottomleft", legend=mapLegend, fill=colPal,col=colPal,title="Title (units)",horiz=F,bty="y")
```



```
par(op)
```