Automating map generation for Montana Watersheds 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 Kathy's segments, watersheds, and data. To access the information for the SBase item, use the following code which utilizes an items Science Base ID **57114f7be4b0ef3b7ca554e8**. The function *list_files* shows the data files associated with the SB item as well as the download urls.

```
test_item=sbtools::item_get("57114f7be4b0ef3b7ca554e8")
```

```
## 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 Streamsegments_Qchange_Buffer.csv 31750
##
## 1 https://www.sciencebase.gov/catalog/file/get/57114f7be4b0ef3b7ca554e8?f=__disk__e4%2Fc5%2Fe6%2Fe4c5
```

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.

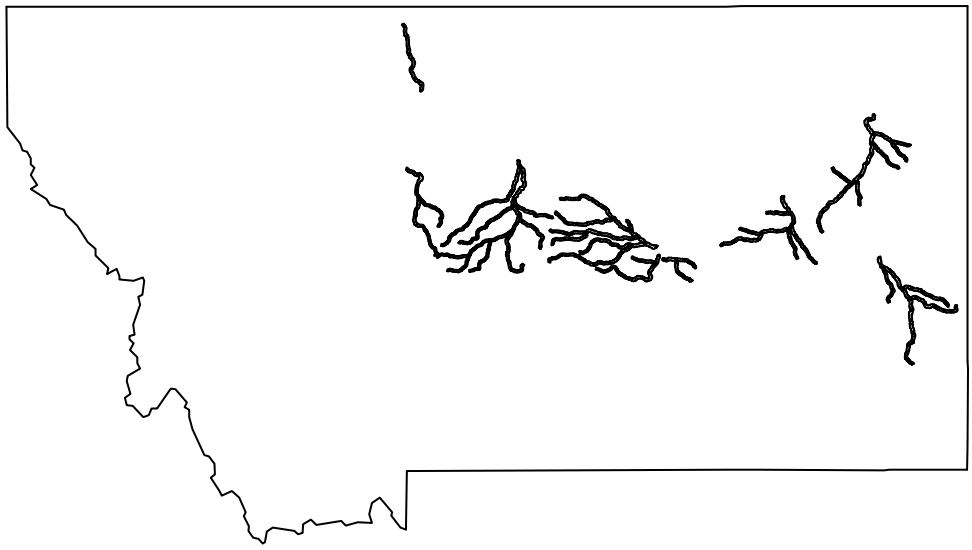
```
layer<-sbtools::item_get_wfs("57114f7be4b0ef3b7ca554e8")
```

```
## Loading required namespace: rgdal
```

```
## Loading required namespace: xml2
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\abock\AppData\Local\Temp\1\Rtmpkp12kd/file1bd018e95db3", layer: "StreamBuffer"
## with 185 features
## It has 11 fields
```

```
# re-project layer to decimal degrees, WGS84
layer_dd<-sp::spTransform(layer,"+init=epsg:4326")
map('state','montana')
sp::plot(layer_dd,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)
print(sbfiles)
```

```
##                               fname  size
## 1 Streamsegments_Qchange_Buffer.csv 31750
##
## 1 https://www.sciencebase.gov/catalog/file/get/57114f7be4b0ef3b7ca554e8?f=__disk__e4%2Fc5%2Fe6%2Fe4c5
```

```
data <- RCurl::getURL(sbfiles$url[1])
data2 <- read.csv(text=data)
names(data2)
```

```
##  [1] "FID"        "OBJECTID"   "POI_ID"     "MAX_SO"     "K_coef"
##  [6] "Basin"      "Segment"    "COMID"      "Years"      "Thirties"
## [11] "FiftyFives" "Eighties"
```

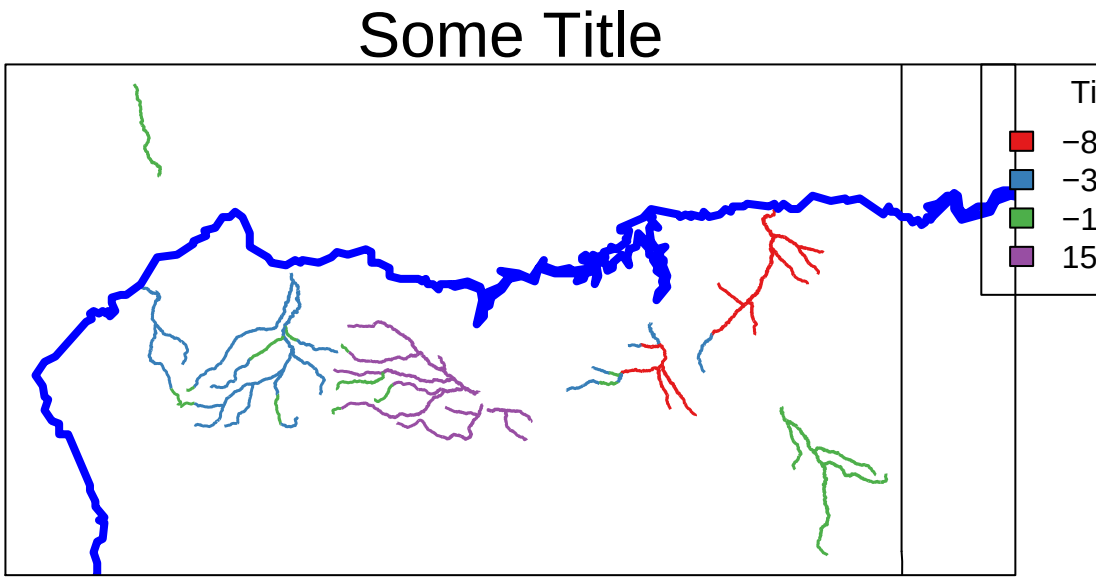
Setting up the plot

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. The base plotting function *op* assigns a font for title, legend, and axes. The *mar* function creates space along the plotting margins, so a legend can be partially fit.

```
colPal<-RColorBrewer::brewer.pal(4,"Set1")
# hard breaks for symbology
#fixedBreaks=c(-30,-15,0,15,30)
fixedBreaks=c(min(data2$FiftyFives), quantile(data2$FiftyFives,.25),median(data2$FiftyFives),quantile(da
symb<-cut(data2$FiftyFives,breaks=fixedBreaks,include.lowest=TRUE,right=TRUE)

op<-par(family="serif")

par(mar=c(0,0,0,16))
```

This next set of code is similar to the code that generates Roy's locations and data. The function *par(xpd=TRUE)* will add the legend (on the following line) partially outside the plotting area. Right now the legend is being cut off of the right side; I need to update my Rstudio version to see if there is a fix for this.

```
sp::plot(layer_dd,col=colPal[symb],border=FALSE)
map('state','montana',add=TRUE)
map('rivers',add=TRUE,col=4,lwd=4)
mtext("Some Title",cex=2,line=0)
box(which="plot",lty="solid")

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=""))
#par(xpd=NA,font=21)
par(xpd=TRUE)
legend("topright",inset=c(-0.3,0), legend=mapLegend, fill=colPal,col=colPal,title="Title (units)",horiz=
```




```
par(op)
```