Lab 02: R as a GIS

Read the instructions COMPLETELY before starting the lab

This lab builds on many of the discussions and exercises from class, including previous labs.

Formatting your submission

This lab must be placed into a public repository on GitHub (www.github.com). Before the due date, submit **on Canvas** a link to the repository. I will then download your repositories and run your code. The code must be contained in either a .R script or a .Rmd markdown document. As I need to run your code, any data you use in the lab must be referenced using **relative path names**. Finally, answers to questions I pose in this document must also be in the repository at the time you submit your link to Canvas. They can be in a separate text file, or if you decide to use an RMarkdown document, you can answer them directly in the doc.

Data

The data for this lab can be found in the `./data/CBW/` directory within the course GitHub repository.

Spatial datasets:

1. Streams_Opened_by_Dam_Removal_2012_2017.shp
2. Dam_or_Other_Blockage_Removed_2012_2017.shp
3. County_Boundaries.shp

Non-spatial datasets

1. BMPreport2016_landbmps.csv

Working with tabular data

```
# setup
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.4.1

## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```r
library(tmap)
```

```
## Warning: package 'tmap' was built under R version 4.4.1
```

A "join" is a method to join multiple tables together using a matching "key" found in both datasets. For example:

```r
# table of names
t.names <- tibble(key = c(1, 2, 3),
             name = c("Huey", "Dewey", "Louis"))

# table of scores
t.scores <- tibble(name = c("Louis", "Huey", "Dewey"),
                   grade = c(99, 45, 33))

# join them together using a common variable, in this case the "name" variable
t.joined <- left_join(t.names, t.scores, by = "name")
t.joined
```

```
## # A tibble: 3 x 3
##     key name  grade
##   <dbl> <chr> <dbl>
## 1     1 Huey     45
## 2     2 Dewey    33
## 3     3 Louis    99
```

A "left join" finds starts with the table on the "left" and then finds matches in the table on the "right". See the documentation using `?left_join` for more details and for other types of joins. Sometimes the attributes you're using to join the tables won't have the same name, in which case the you need to tell the join function which variables to join on. Note the difference in the `by` argument in the below code block compared to the one above.

```r
t.wonkyNames <- tibble(nombre = c("Dewey", "Louis", "Huey"),
                       x = rep(999),
                       favoriteFood = c("banana", "apple", "carrot"))

t.joined2 <- left_join(t.names, t.wonkyNames, by = c("name" = "nombre"))
t.joined2
```

```
## # A tibble: 3 x 4
##     key name      x favoriteFood
##   <dbl> <chr> <dbl> <chr>
## 1     1 Huey    999 carrot
## 2     2 Dewey   999 banana
## 3     3 Louis   999 apple
```

Let's take a look at some tabular data

This dataset includes a list of best management practices ("BMPs") to reduce nutrient and sediment pollution in the Chesapeake Bay Watershed.

```
bmps <- read_csv("../data/CBW/BMPreport2016_landbmps.csv")
```

```
## Rows: 69601 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (11): StateAbbreviation, GeographyName, Geography, Agency, BMPShortName,...
## dbl  (7): AmountSubmitted, AmountBackedOut, AmountNotBackedOut, AmountCredit...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(bmps)
```

```
## Rows: 69,601
## Columns: 18
## $ StateAbbreviation   <chr> "DC", "DC", "DC", "DC", "DC", "DC", "DC", "DC", "D~
## $ GeographyName       <chr> "11001(cbwsonly)", "11001(cbwsonly)", "11001(cbwso~
## $ Geography           <chr> "Washington, DC (CBWS Portion Only)", "Washington,~
## $ Agency              <chr> "Department of Defense", "Department of Defense", ~
## $ BMPShortName        <chr> "wetpondwetland", "wetpondwetland", "wetpondwetlan~
## $ BMP                 <chr> "Wet Ponds and Wetlands", "Wet Ponds and Wetlands"~
## $ BMPType             <chr> "Efficiency", "Efficiency", "Efficiency", "Efficie~
## $ Unit                <chr> "Acres Treated", "Acres Treated", "Acres Treated",~
## $ Sector              <chr> "Developed", "Developed", "Developed", "Developed"~
## $ FromLoadSource      <chr> "Non-Regulated Roads", "Non-Regulated Buildings an~
## $ ToLoadSource        <chr> "Non-Regulated Roads", "Non-Regulated Buildings an~
## $ AmountSubmitted     <dbl> 8.709123261, 47.984171150, 1.358309392, 6.76626660~
## $ AmountBackedOut     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ AmountNotBackedOut  <dbl> 8.709123261, 47.984171150, 1.358309392, 6.76626660~
## $ AmountCredited      <dbl> 8.709123261, 47.984171150, 1.358309392, 6.76626660~
## $ Excess              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ TotalAmountCredited <dbl> 8.709123261, 47.984171150, 1.358309392, 6.76626660~
## $ Cost                <dbl> 11462.077120, 63151.967650, 1787.670991, 8905.0834~
```

Look at the attribute "GeographyName" - it's a character attribute that contains the counties' FIPS code, but also some ancillary explanatory data we need to get rid of. There are multiple ways of doing so, including some (very) fancy automated methods that detect patterns of numbers and characters. We're going to take a simpler approach and assume that all FIPS codes are only 5 characters long.

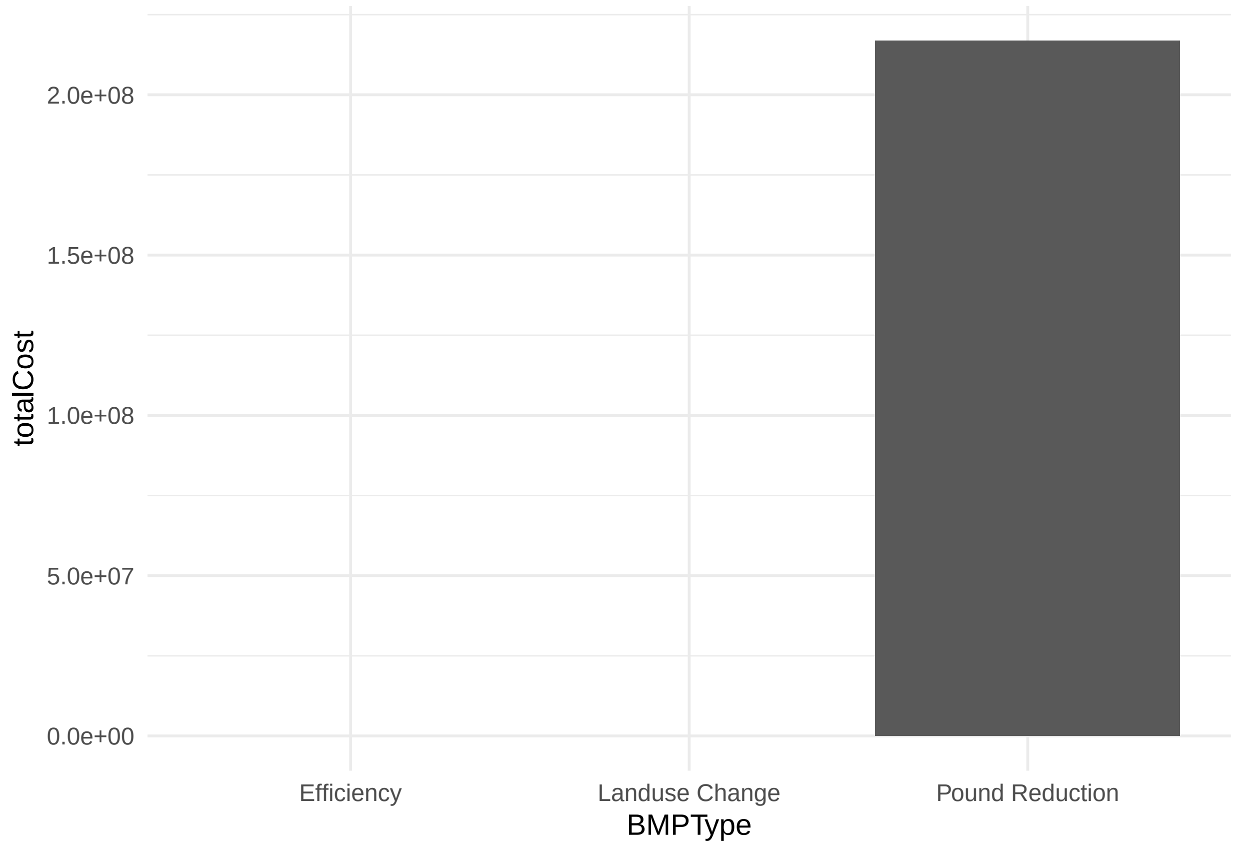
```
# edit the bmps variable in place, which isn't always best practices
bmps <- bmps %>% mutate(., FIPS.trimmed = stringr::str_sub(GeographyName, 1, 5))
```

This preparing raw data for further analysis is called "munging". While the term isn't the important part, these kinds of preparatory steps can be useful when you're trying to create "keys" by which to join tables or just clean your tables in general.

Let's recall how to do some simple tasks

```
# Let's calculate the total cost by BMP and then plot it
bmps %>% group_by(BMPType) %>% summarise(totalCost = sum(Cost)) %>%
  ggplot(., aes(x = BMPType, y = totalCost)) +
  geom_bar(stat = "identity") +
  theme_minimal()
```

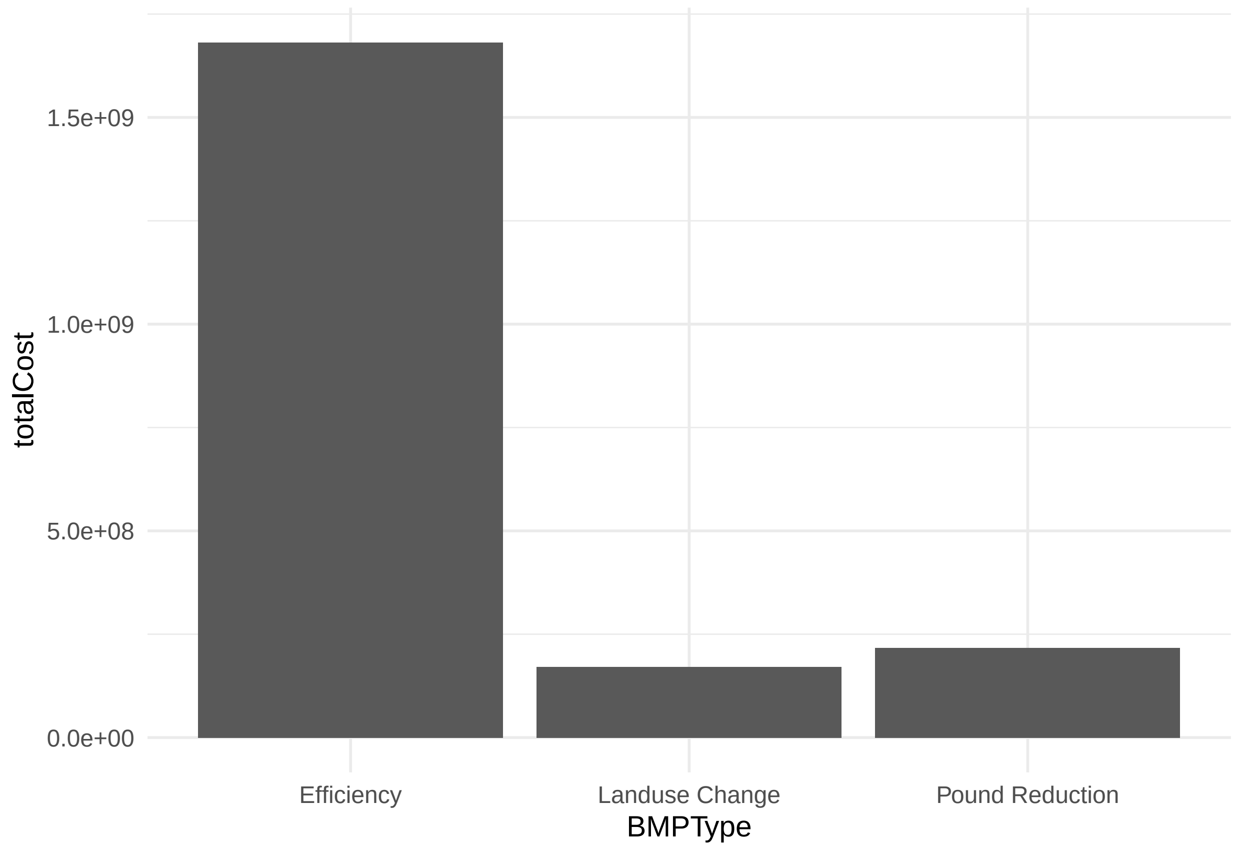
```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_bar()‘).
```



```
# Doesn't really work. This is because there are missing data in the cost attribute.
# Let's look at it (and this is why we do exploratory data analysis first)

summary(bmps$Cost)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##        0        0      142    37408     3583 39731974    14286
```

```
# Yup, lots of "NA's"... We can drop them in our analysis.
# Look carefully at the code in the 'sum' function. What is the na.rm = T parameter doing?


bmps %>% group_by(BMPType) %>% summarise(totalCost = sum(Cost, na.rm = T)) %>%
  ggplot(., aes(x = BMPType, y = totalCost)) +
  geom_bar(stat = "identity") +
  theme_minimal()
```



We can also group by multiple variables at the same time. For example:

```
# group by state and sector, sum total cost
twofactors <- bmps %>% group_by(StateAbbreviation, Sector) %>%
  summarise(totalCost = sum(Cost))
```

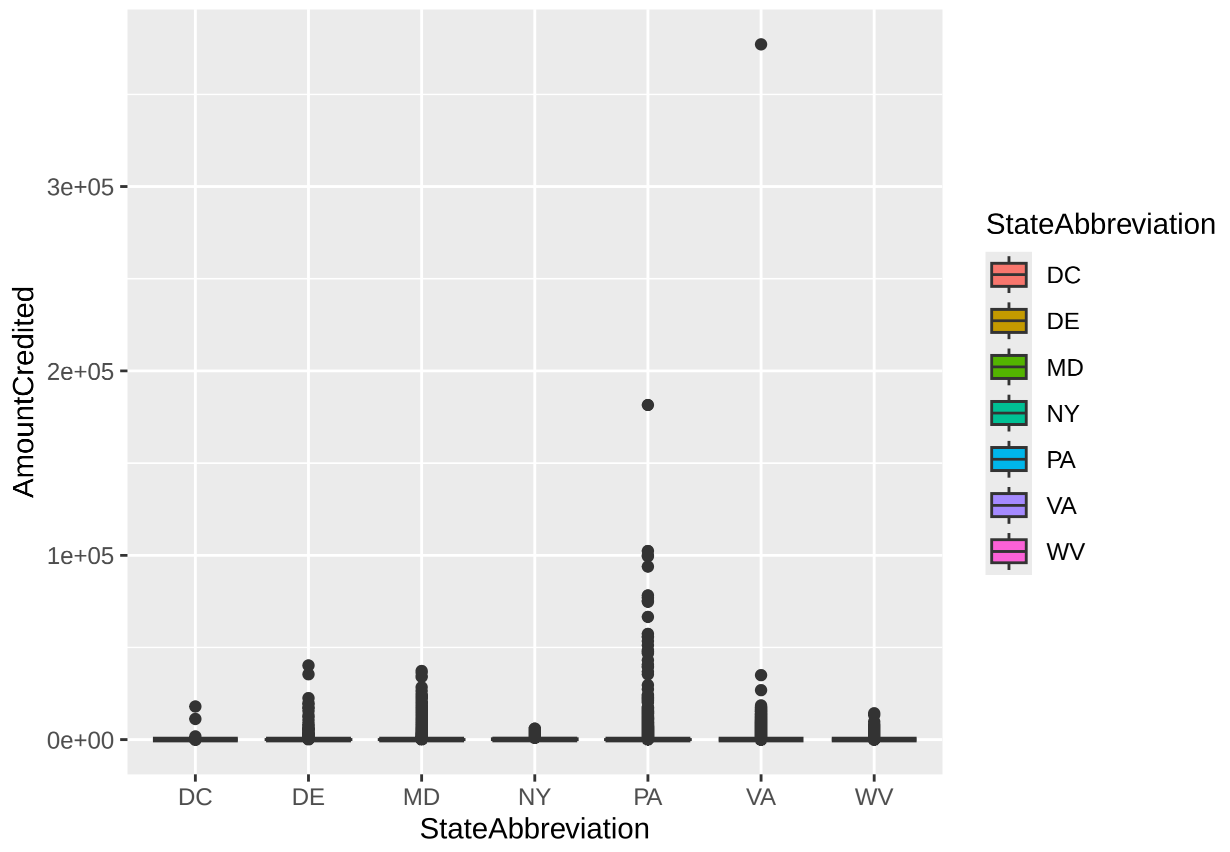
```
## ‘summarise()‘ has grouped output by ’StateAbbreviation’. You can override using
## the ‘.groups‘ argument.
```

```
twofactors
```

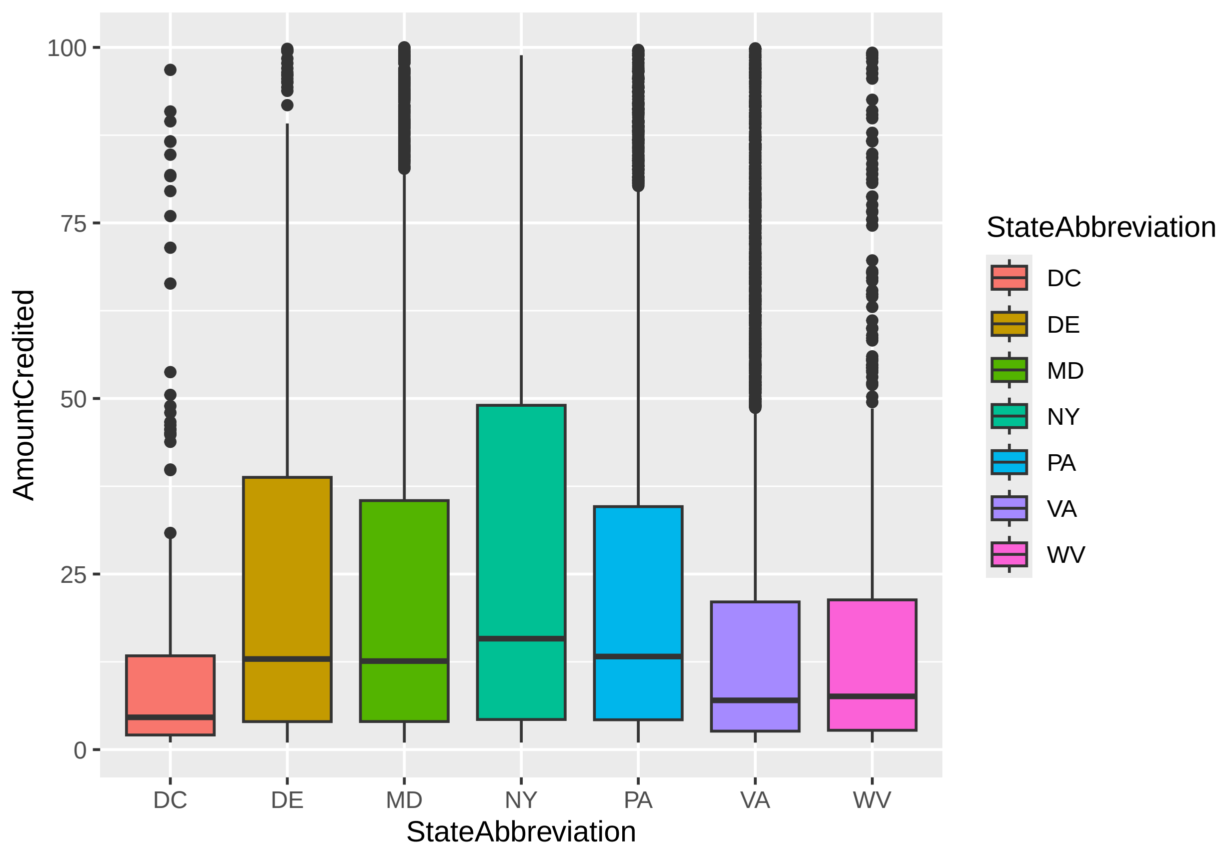
```
## # A tibble: 25 x 3
## # Groups:   StateAbbreviation [7]
##    StateAbbreviation Sector      totalCost
##    <chr>             <chr>           <dbl>
##  1 DC                Developed         NA
##  2 DC                Natural           NA
##  3 DE                Agriculture       NA
##  4 DE                Developed         NA
##  5 DE                Natural           NA
##  6 DE                Septic       2064477.
##  7 MD                Agriculture       NA
##  8 MD                Developed         NA
##  9 MD                Natural           NA
## 10 MD                Septic      50470281.
## # i 15 more rows
```

For our last bit of review, let's make a few box plots:

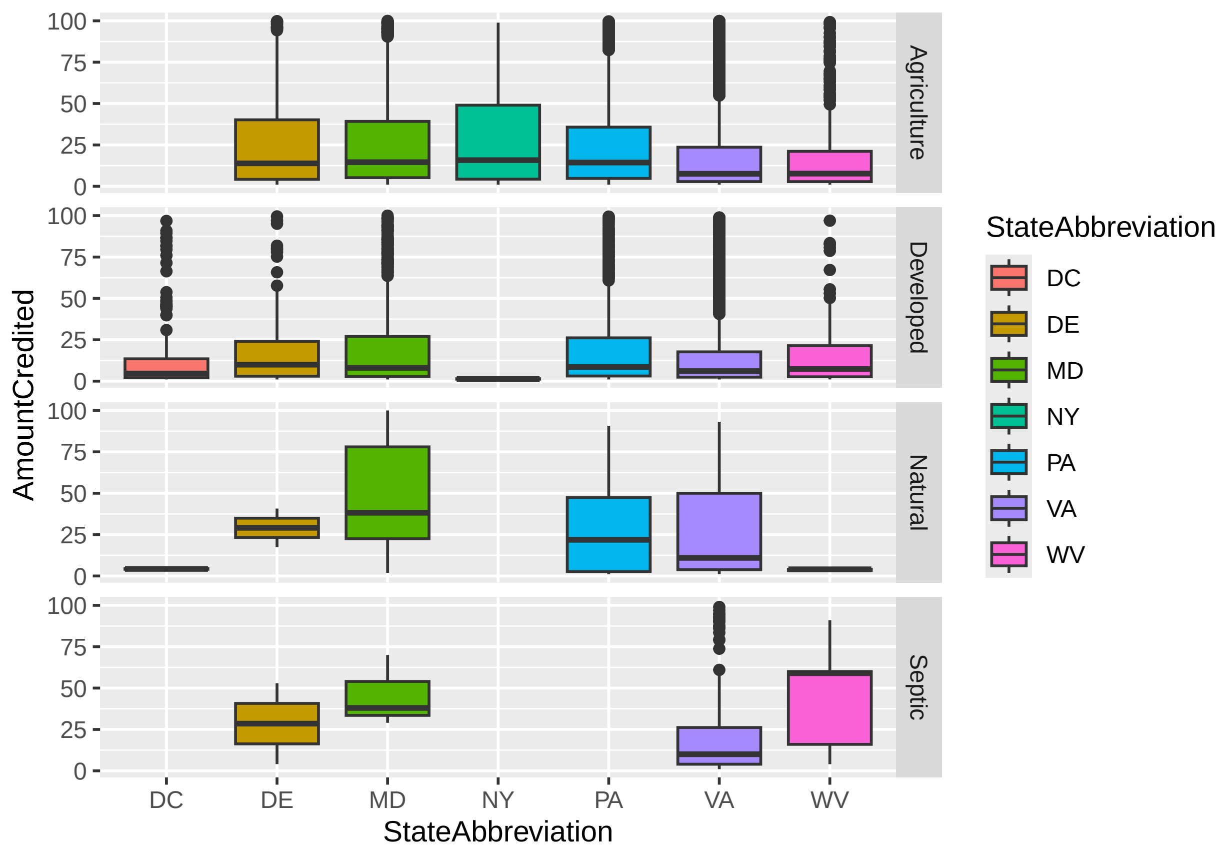
```r
# A simple one
bmps %>% ggplot(., aes(x = StateAbbreviation, y = AmountCredited)) +
  geom_boxplot(aes(fill = StateAbbreviation))
```



```r
# Very heavily skewed, so just for the sake of visualization,
# let's subset the data (dramatically)
bmps %>%
  dplyr::filter(., AmountCredited > 1 & AmountCredited < 100) %>%
  ggplot(., aes(x = StateAbbreviation, y = AmountCredited)) +
  geom_boxplot(aes(fill = StateAbbreviation))
```



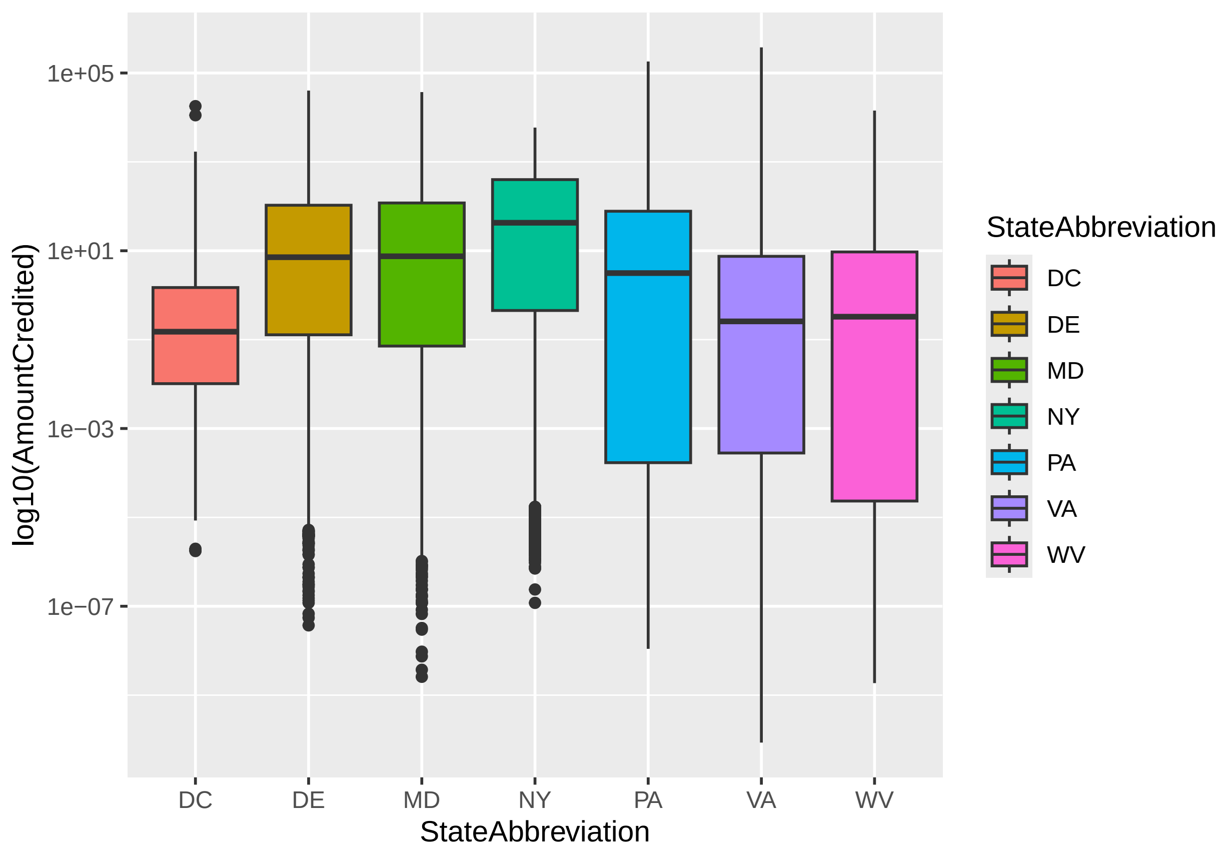
```
# We can also plot multiple dimensions in our plot using
# the `facet` family of commands in ggplot

bmps %>%
  dplyr::filter(., AmountCredited > 1 & AmountCredited < 100) %>%
  ggplot(., aes(x = StateAbbreviation, y = AmountCredited)) +
  geom_boxplot(aes(fill = StateAbbreviation)) +
  facet_grid(Sector~.)
```



```r
# finally, an alternative to removing the outliers would be to log-transform the y-axis
# But be careful, this removes a lot of the data because the log of 0 is undefined
bmps %>% ggplot(., aes(x = StateAbbreviation, y = AmountCredited)) +
  geom_boxplot(aes(fill = StateAbbreviation)) +
  scale_y_log10() +
  labs(y = "log10(AmountCredited)")
```

```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```

```
## Warning: Removed 15585 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```



The last new item uses the `%in%` command. It's a way to quickly figure out which elements are inside of another. In that sense, it's similar to a spatial intersection, but for other types of data.

```r
# setup a demonstration vector
x <- c(1, 2, 3, 4, 5)

# is 7 in our vector?
7 %in% x # should evaluate False
```

```
## [1] FALSE
```

```r
2 %in% x # should evaluate True
```

```
## [1] TRUE
```

```r
# can also do it with vectors
c(4, 99, 1) %in% x
```

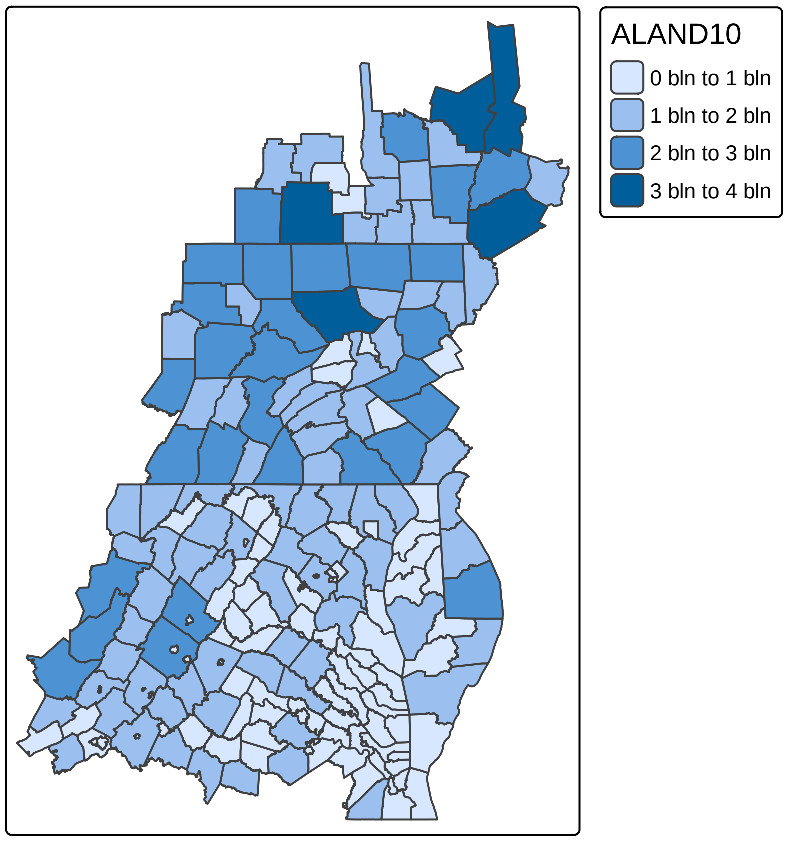
```
## [1]  TRUE FALSE  TRUE
```

Lastly, let's recall using tmap on our data. Also remember you can use `sf::st_make_valid` to fix offending geometry

```
counties <- sf::read_sf("../data/CBW/County_Boundaries.shp")
counties %>% sf::st_is_valid()
```

```
##   [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [97]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [157]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [169]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [181]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [193]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [205]  TRUE FALSE  TRUE
```

```
counties <- counties %>% sf::st_make_valid()

# quick map of the data
tm_shape(counties) + tm_polygons(fill = "ALAND10")
```


ALAND10
0 bln to 1 bln
1 bln to 2 bln
2 bln to 3 bln
3 bln to 4 bln

Your tasks

Using the following data…

```r
# spatial
# Note, the st_make_valid() calls are NOT necessary in all cases
# But we know it's needed for the counties, and doesn't really hurt anything
# for the others
counties <- sf::read_sf("../data/CBW/County_Boundaries.shp") %>%
  sf::st_make_valid()
dams <- sf::read_sf("../data/CBW/Dam_or_Other_Blockage_Removed_2012_2017.shp") %>%
  sf::st_make_valid()
streams <- sf::read_sf("../data/CBW/Streams_Opened_by_Dam_Removal_2012_2017.shp") %>%
  sf::st_make_valid()

# aspatial
bmps <- read_csv("../data/CBW/BMPreport2016_landbmps.csv")
```

```
## Rows: 69601 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (11): StateAbbreviation, GeographyName, Geography, Agency, BMPShortName,...
## dbl  (7): AmountSubmitted, AmountBackedOut, AmountNotBackedOut, AmountCredit...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

…complete the following tasks.

Complete each task COMPLETELY USING R CODE. YOU MUST SHOW YOUR WORK FOR EACH ANSWER. Label your variables sensibly and use comments such that I can find your answers and your work. The following tasks draw upon lecture, your assigned readings, and the examples shown above. As always, there are multiple ways of completing each task. Remember, it's always a good idea to perform some exploratory data analysis on your own prior to starting work.

Task 1: Aspatial operations

1.1 Calculate summary statistics for the Cost of BMPs for each State (including DC)

1.2 Make a scatterplot of Cost vs. TotalAmountCredited, ONLY FOR BMPs with `Units` of type "Acres". You may need to apply a data transformation to one or more axes if the data are heavily skewed.

1.3 Make a boxplot with "StateAbbreviation" on the x-axis and "TotalAmountCredited" on the y-axis. HOWEVER, the only data I want plotted are for cover crop BMPs. Note, there are many types of cover crops in this dataset, and I want you to include them ALL. There are handy functions within the `stringr` package that can help you here. Use the help function `?stringr` or read the documentations/vignettes found online. A Google search for "r stringr vignettes documentation" would be a good place to start if you're having trouble.

1.4 make a scatterplot of the `dam` dataset, this time with "YEAR" on the x-axis and "STATE" on y-axis (think of it like a timeline). Assume no dams were built in year 0, so you'll need to remove those data points. You know how to remove observations based on a given criteria.

1.5 make one last (aspatial) visualization. But this time, it's your choice what data and plots to use. The only requirement is that you link two of the datasets together in some manner. Be creative. Make it look nice (e.g., use proper labels, interesting colors/shading/size).

Task 2: Spatial operations

2.1 Find the 5 longest streams in the ‘streams opened by dam removal’ dataset

2.2 Find the three counties with the greatest TOTAL length of streams (opened by dam removal) in them

2.3 Make a map of the counties, shading each county by the total cost of BMPs funded/implemented in that county. This will require you to join multiple datasets together

2.4 For each removed dam, find the closest stream segment

2.5 Calculate how many removed dams are (or were) in each state