Lab 2 - R as a GIS

Bailey McNichol

10/10/2021

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 packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.2.1, PROJ 7.2.1
```

```
library(tmap)
library(stringr)
```

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

```
# 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))

# combined them
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 syntax is different:

```
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.

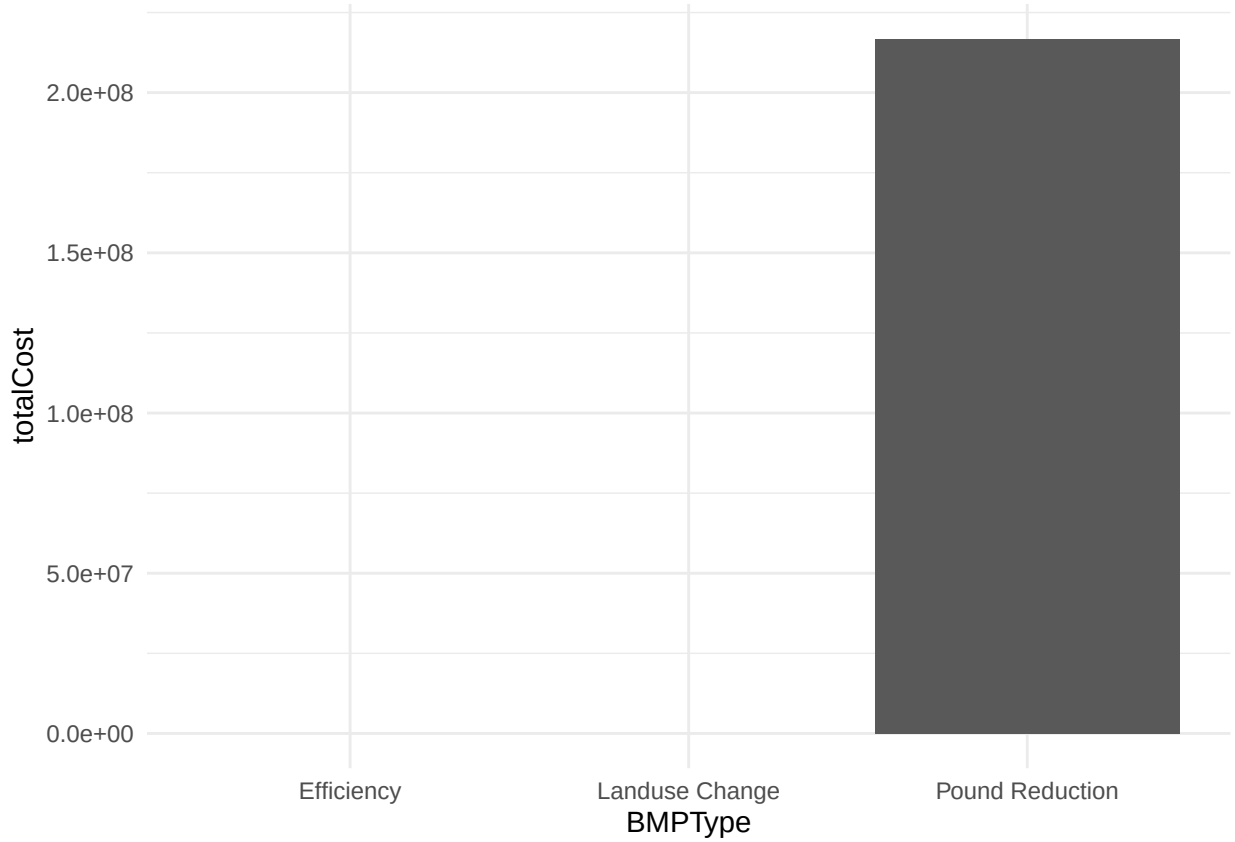
```r
# edit the bmps variable in place, which isn't always best practices
bmps <- bmps %>% mutate(., FIPS.trimmed = stringr::str_sub(GeographyName, 1, 5))
```

This can be sueful 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

```r
# 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 (position_stack).
```

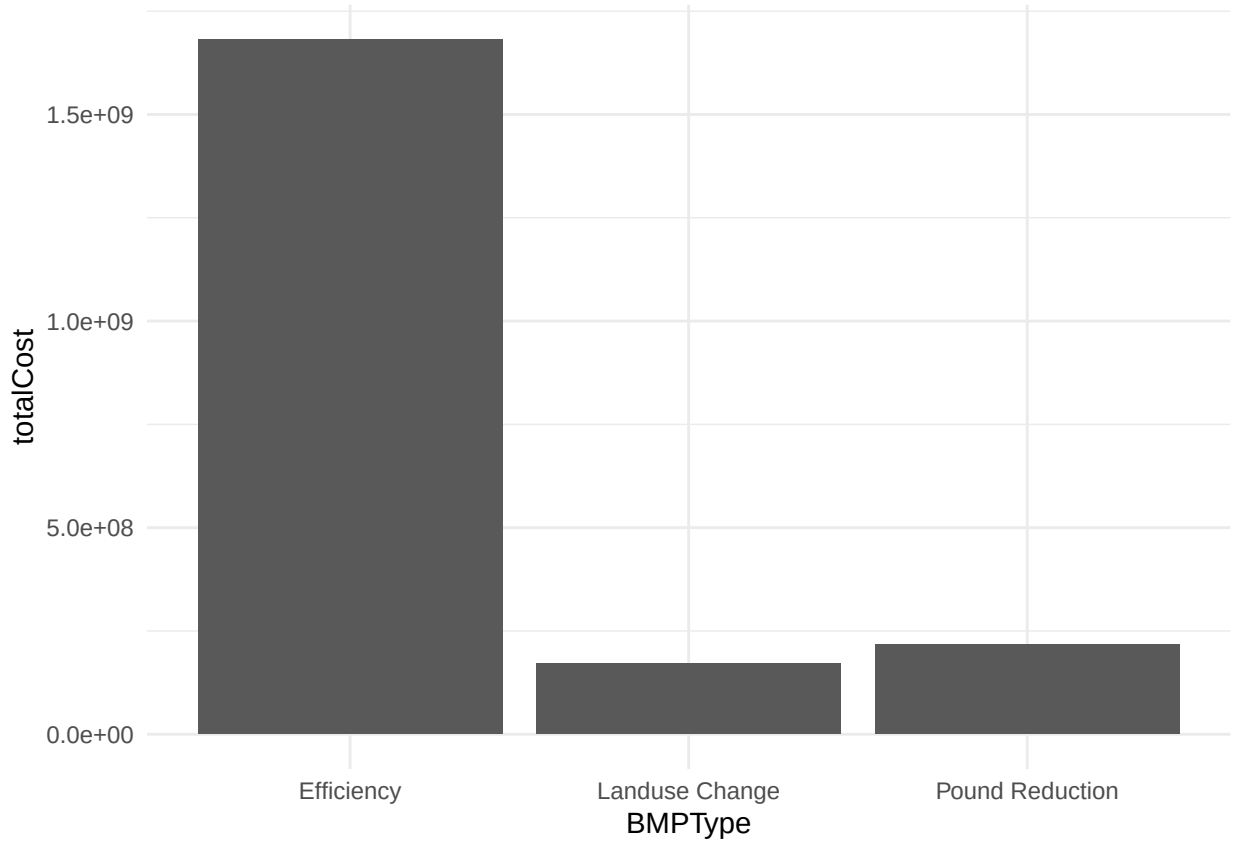



```r
# Doesn't really work. This is because there are missing data in the cost attribute. Let's look at it (

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' func
```

```
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` argume
```

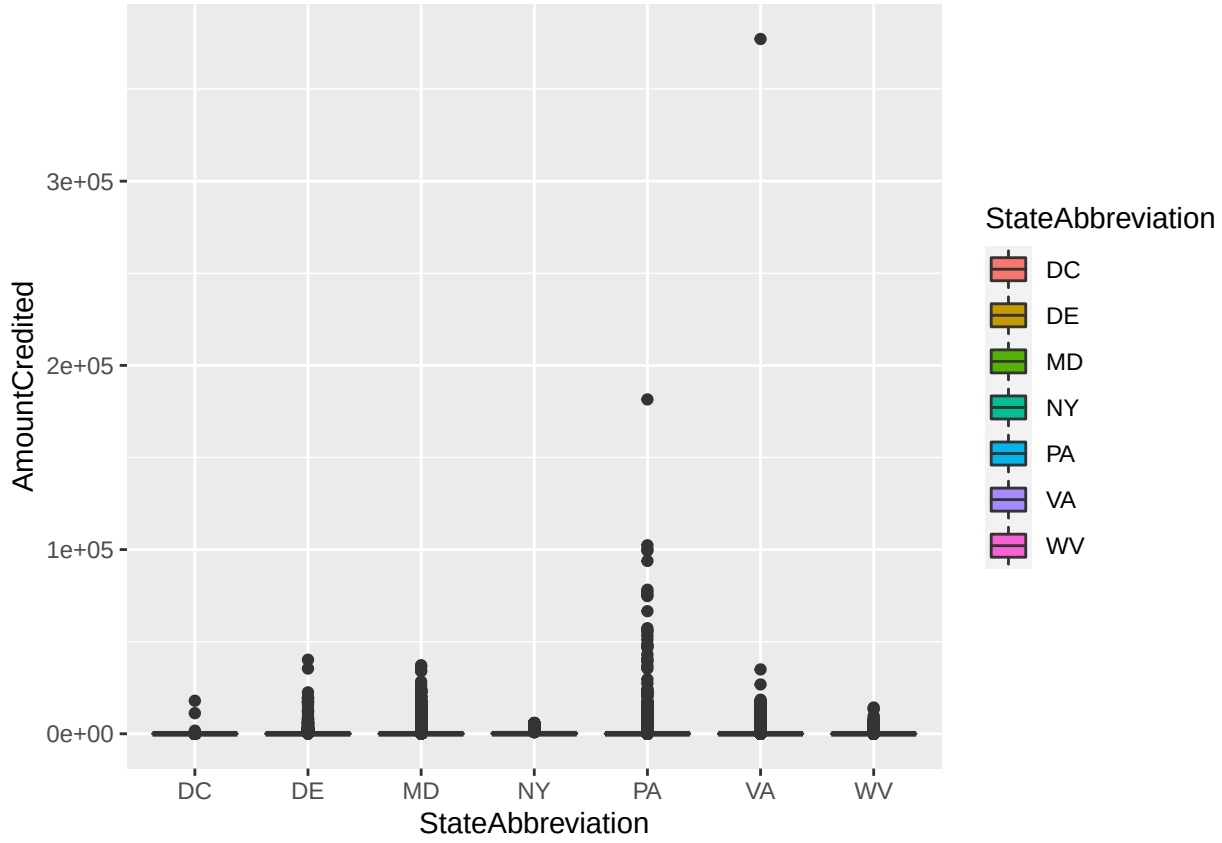
```
twofactors
```

```
## # 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
```

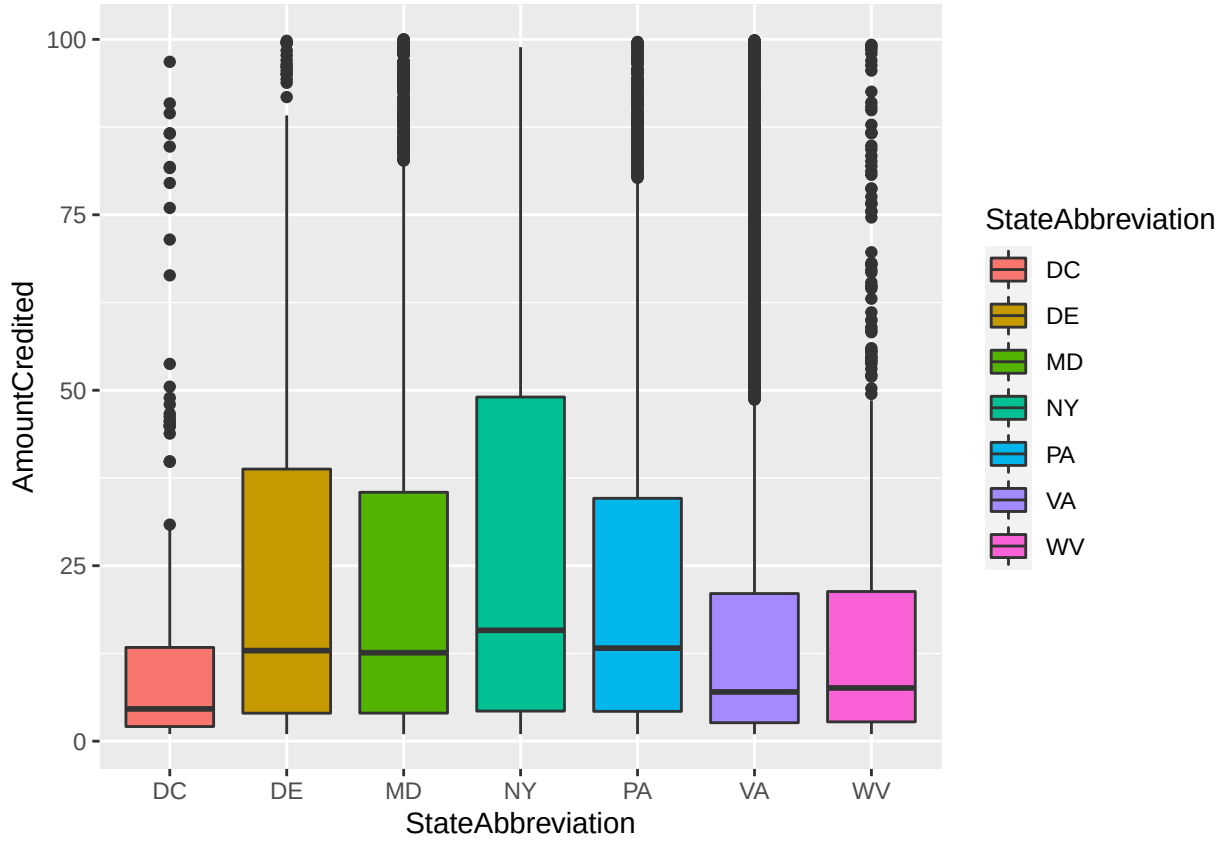
```
##  5 DE                Natural           NA
##  6 DE                Septic       2064477.
##  7 MD                Agriculture       NA
##  8 MD                Developed         NA
##  9 MD                Natural           NA
## 10 MD                Septic      50470281.
## # ... with 15 more rows
```

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

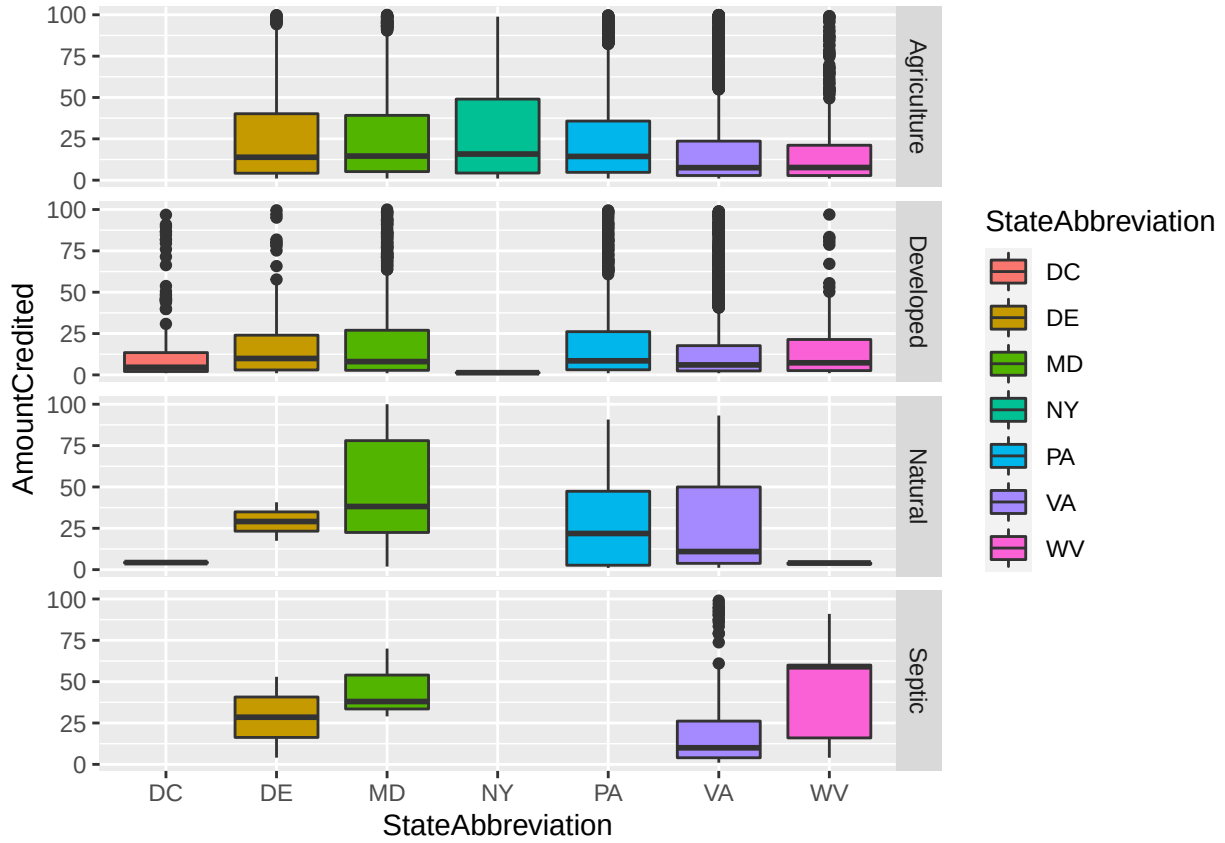
```
# A simple one
bmps %>% ggplot(., aes(x = StateAbbreviation, y = AmountCredited)) +
  geom_boxplot(aes(fill = StateAbbreviation))
```




```
# 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~.)
```



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.

```
x <- c(1, 2, 3, 4, 5)

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

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

```
2 %in% x # should be True
```

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

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

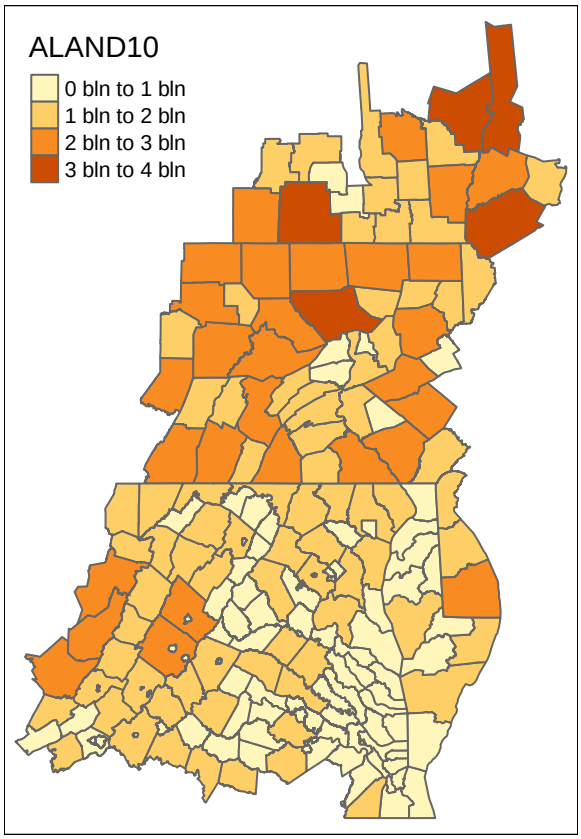
```
## [1]  TRUE FALSE  TRUE
```

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(col = "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...

```
# spatial
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 peform some exploratory data analysis on your own prior to starting work.

Lab 2 Tasks

Task 1: Basic data manipulation

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

First, I grouped the BMPs by state, and then used summarize to calculate the mean, median, minimum, and maximum cost of BMPs for each State. The output is this table of values, where each row is a state.

```
# Look at the variable names
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~
```

```
# Use group_by to group by states
bmps %>% group_by(StateAbbreviation) %>%
  # Use summarize to calculate the summary statistics
             summarise(meanCost = mean(Cost, na.rm = T),
                       medCost = median(Cost, na.rm = T),
                       minCost = min(Cost, na.rm = T),
                       maxCost = max(Cost, na.rm = T))
```

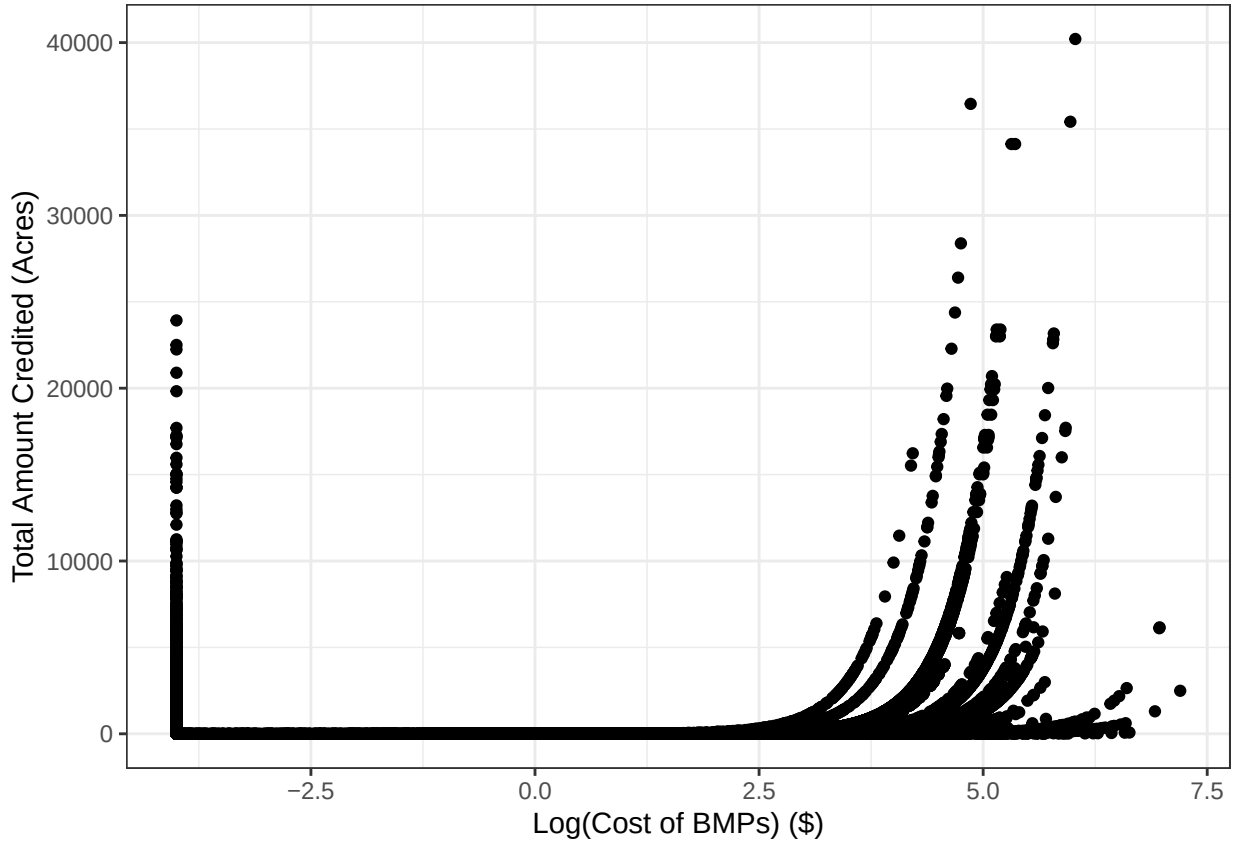
```
## # A tibble: 7 x 5
##   StateAbbreviation meanCost medCost minCost   maxCost
##   <chr>                <dbl>   <dbl>   <dbl>     <dbl>
## 1 DC                  22193.   629.   0.0510  1891064.
## 2 DE                  30682.   906.   0       5041838.
## 3 MD                  53118.   876.   0      12086857.
## 4 NY                   8129.  1954.   0        622138.
## 5 PA                  59904.   431.   0      19968186
## 6 VA                  25380.    15.2  0      39731974.
## 7 WV                  50014.    96.8  0      15796521.
```

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

First, I filtered the BMPs to only include observations where the units were acres, and then I dropped all NA values for Cost. Then, I plotted the Cost of BMPs versus the Total Amount of Credited Acres. To improve data visualization and deal with the heavy left-skew of the cost data (losts of small values with a few large outliers), I applied a log base 10 transformation to Cost, first adding 0.0001 to each value to accommodate values of 0 for Cost (and we can see that there are many different total amount credited values that had a cost of \$0 - the weird vertical line of data on the left). I chose to leave in the zero values on my figure, as there are many credited acres that a cost of zero (so I didn't want to exclude relevant data). On the x-axis of the figure, the log 10 values can be back-transformed to represent costs of approximately 0.003, 1.00, 316.3, 100,023, and 31,630,060 dollars, respectively.

```
bmps %>%
  # Only include units of acres
  dplyr::filter(., Unit == "Acres") %>%
  # Remove NAs in Cost
  drop_na(Cost) %>%
  # Transform Cost to log10 scale, adding 0.0001 to each value to avoid undefined numbers
  ggplot(., aes(x = log10(Cost+0.0001), y = TotalAmountCredited)) +
  geom_point() +
```

```
  theme_bw() +
  labs(x = "Log(Cost of BMPs) ($)", y = "Total Amount Credited (Acres)")
```




```
# Back-transform the log10 values to actual costs in dollars
10^c(-2.5001, 0.0001, 2.5001, 5.0001, 7.5001)
```

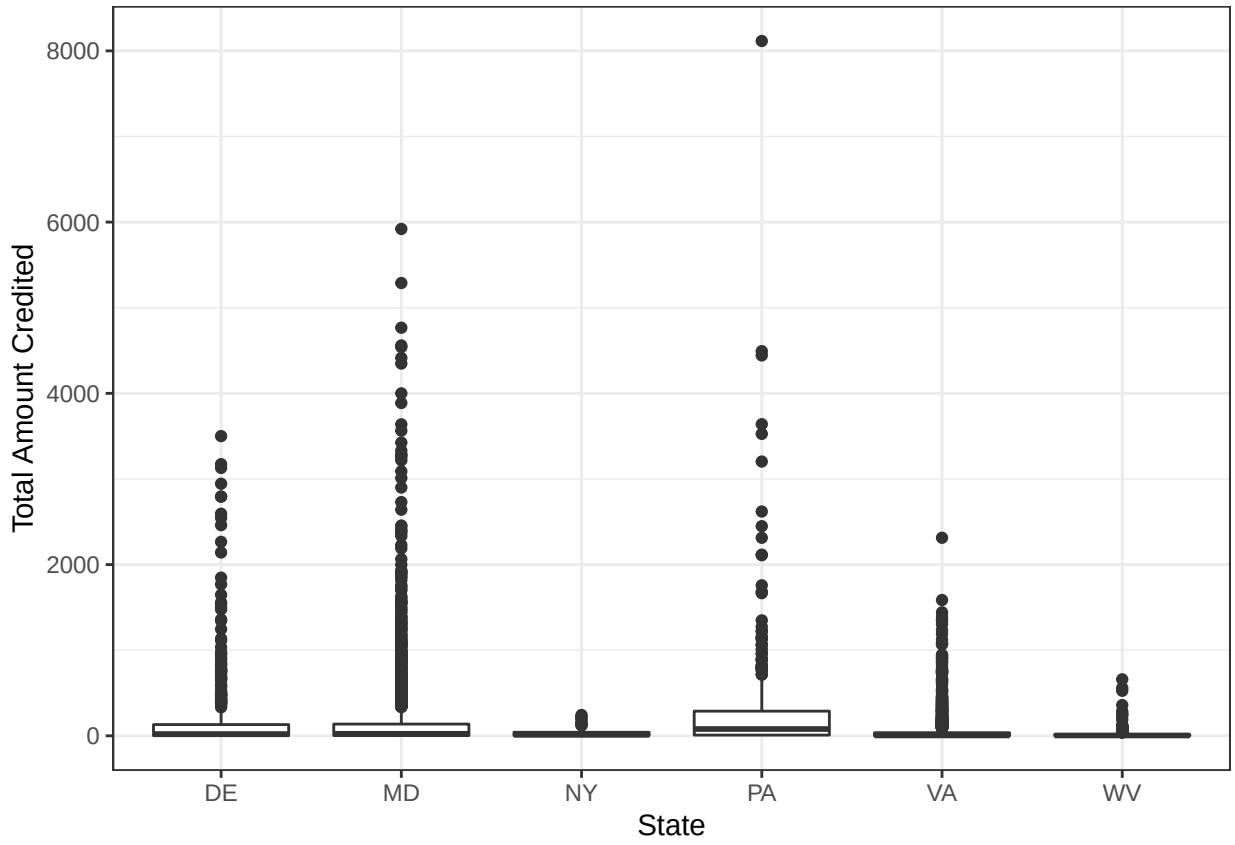
```
## [1] 3.161550e-03 1.000230e+00 3.163006e+02 1.000230e+05 3.163006e+07
```

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.

First, I filtered the data to include any BMPShortName values with the string "covercrop" using the filter and str_detect functions. Then, I was able to produce the boxplot of Total Amount Credited by State; I opted to leave the outliers on the figure, as there are many and there is no clear cut-off that improves the data visualization considerably.

```
bmps %>%
  # Use str_detect within filter to select only (and all of) the cover crop BMPs
  filter(str_detect(BMPShortName, "covercrop")) %>%
  # Plot the variables
  ggplot(., aes(x = StateAbbreviation, y = TotalAmountCredited)) +
  geom_boxplot() +
```

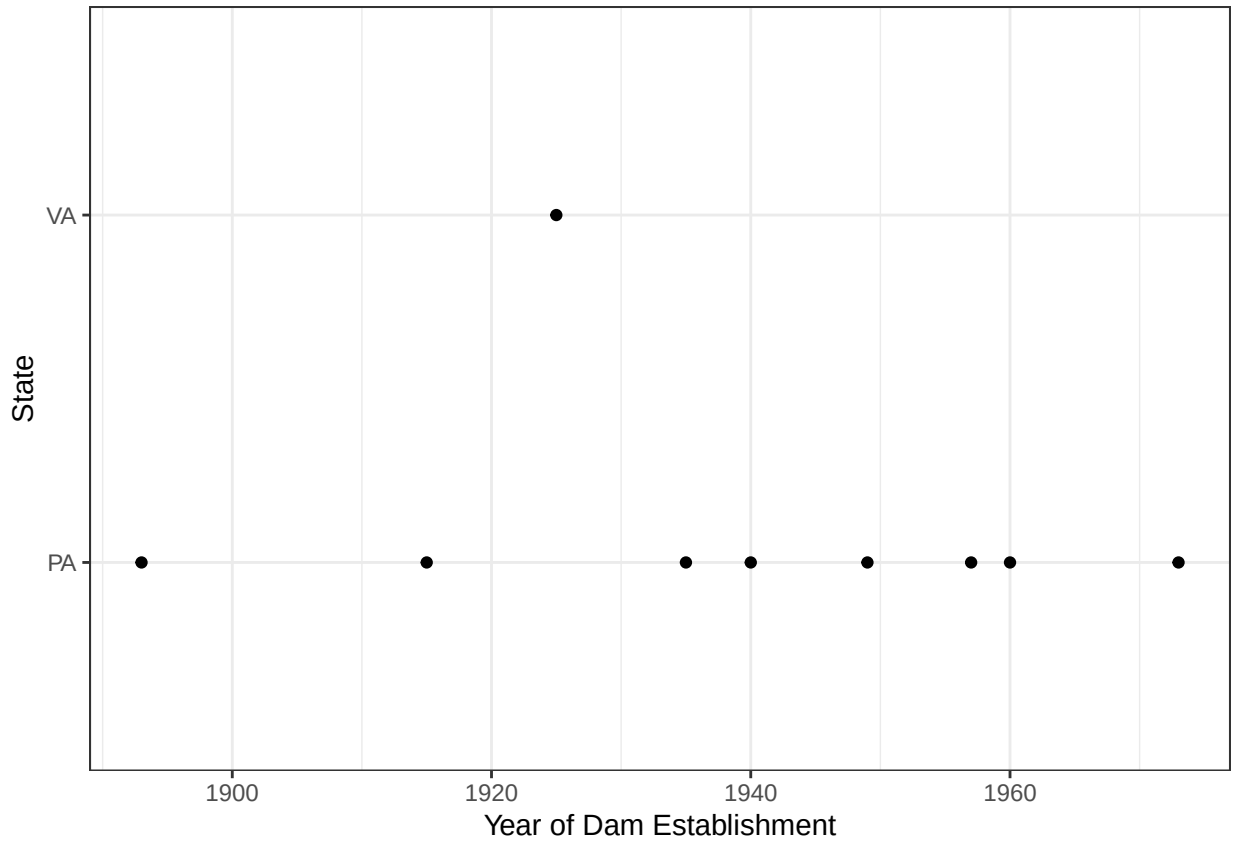
```
  theme_bw() +
  labs(x = "State", y = "Total Amount Credited")
```



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.

First, I filtered the YEAR variable to exclude values of zero, which eliminated MD (all YEAR values were zero for Maryland). Then, I removed NA values for YEAR, and made a scatterplot of the Year of Dam Establishment (YEAR) by STATE - the data now only includes PA and VA, and only data on one dam for VA.

```
dams %>%
  # Filter to remove 0 values for YEAR
  filter(YEAR != 0) %>%
  # Remove NAs for year
  drop_na(YEAR) %>%
  # Plot the variables
  ggplot(., aes(x = YEAR, y = STATE)) +
  geom_point() +
  theme_bw() +
  labs(x = "Year of Dam Establishment", y = "State")
```



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).

I created a new variable in the bmps dataset that represents the County FIPS Code, following the example in the introduction. Then, I joined the counties and bmps datasets using the County FIPS codes (called GEOID10 in counties). I assessed the levels (unique values) of the BMPs, and then filtered this new dataset to only include BMPs involving wet areas. Then, I created a figure showing the relationship between water area and numbers of different BMPs that rely on water, filtering the data for visualization to exclude outliers and be easier to interpret (changing label angles, etc).

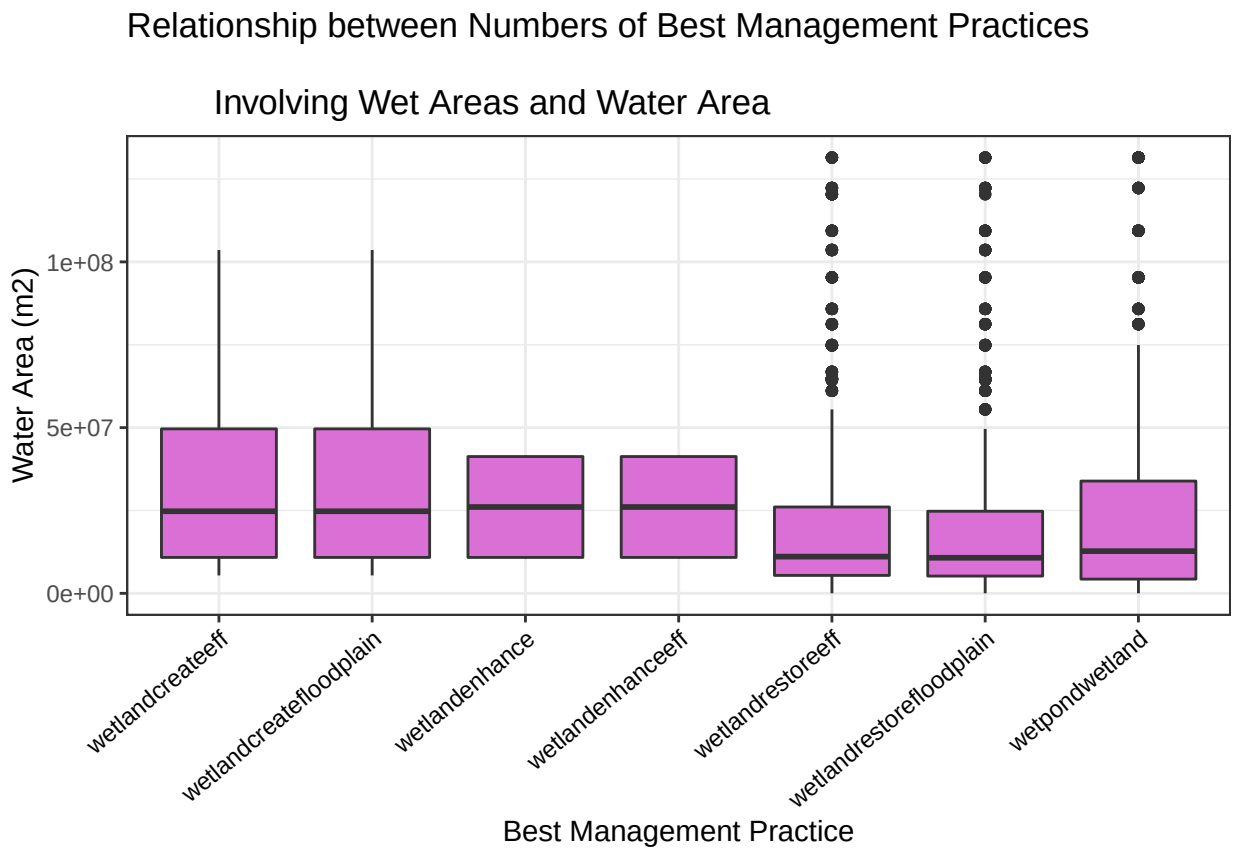
```
# Create County FIPS Codes for bmps data
bmps <- bmps %>% mutate(., FIPS.trimmed = stringr::str_sub(GeographyName, 1, 5))

# Join the bmps data with the counties, by county code
county_bmp <- left_join(counties, bmps, by = c("GEOID10" = "FIPS.trimmed"))

# Examine the levels of BMPShortName
levels(as.factor(county_bmp$BMPShortName))
```

```
##   [1] "abanminerec"             "advancedgi"
##   [3] "barnrunoffcont"          "bioretnoudab"
##   [5] "bioretudab"              "bioretudcd"
##   [7] "bioswale"                "carseqaltcrop"
##   [9] "conplan"                 "conservetill"
```

```
##  [11] "covercropcomearly"        "covercropcomlate"
##  [13] "covercropcomnormal"       "covercroptradarea"
##  [15] "covercroptradared"        "covercroptradbea"
##  [17] "covercroptradbed"         "covercroptradbeo"
##  [19] "covercroptradbnd"         "covercroptradbno"
##  [21] "covercroptradbrea"        "covercroptradbred"
##  [23] "covercroptradbreo"        "covercroptradfea"
##  [25] "covercroptradfed"         "covercroptradfeo"
##  [27] "covercroptradfped"        "covercroptradfpnd"
##  [29] "covercroptradlea"         "covercroptradled"
##  [31] "covercroptradleo"         "covercroptradlgled"
##  [33] "covercroptradlgleo"       "covercroptradlglnd"
##  [35] "covercroptradlnd"         "covercroptradnutreo"
##  [37] "covercroptradnutrnd"      "covercroptradnutrno"
##  [39] "covercroptradnutweo"      "covercroptradnutwno"
##  [41] "covercroptradohea"        "covercroptradohed"
##  [43] "covercroptradohnd"        "covercroptradokeo"
##  [45] "covercroptradrea"         "covercroptradred"
##  [47] "covercroptradreo"         "covercroptradrld"
##  [49] "covercroptradrlo"         "covercroptradrnd"
##  [51] "covercroptradrno"         "covercroptradtea"
##  [53] "covercroptradted"         "covercroptradteo"
##  [55] "covercroptradtld"         "covercroptradtlo"
##  [57] "covercroptradtnd"         "covercroptradtno"
##  [59] "covercroptradwea"         "covercroptradwed"
##  [61] "covercroptradweo"         "covercroptradwld"
##  [63] "covercroptradwlo"         "covercroptradwnd"
##  [65] "covercroptradwno"         "csoconnect"
##  [67] "dirtgraveldsa"            "dryponds"
##  [69] "eands1"                   "eands2"
##  [71] "extdryponds"              "filter"
##  [73] "forestbuffers"            "forestbufferseffn"
##  [75] "forestbufferseffps"       "forestbuffexcl"
##  [77] "forestbuffexcleffn"       "forestbuffexcleffps"
##  [79] "forestbuffexclnar"        "forestbuffnarrow"
##  [81] "forestbufurban"           "forestbufurbaneff"
##  [83] "forharvestbmp"            "grassbuffers"
##  [85] "grassbufferseffn"         "grassbufferseffps"
##  [87] "grassbuffexcl"            "grassbuffexcleffn"
##  [89] "grassbuffexcleffps"       "grassbuffexclnar"
##  [91] "grassbuffnarrow"          "horsepasman"
##  [93] "hrtill"                   "imperviousdisconnection"
##  [95] "impsurred"                "incorplowlate"
##  [97] "infiltration"             "infiltwithsv"
##  [99] "injection"                "landretireopen"
## [101] "landretirepas"            "loaflot"
## [103] "lowrestill"               "nmcoren"
## [105] "nmcorep"                  "nmplacen"
## [107] "nmplacep"                 "nmraten"
## [109] "nmratep"                  "nmtimen"
## [111] "nmtimep"                  "nonurbstrmrest"
## [113] "oswnofence"               "permpavnosvnoudab"
## [115] "permpavnosvudab"          "permpavnosvudcd"
## [117] "permpavsvnoudab"          "permpavsvudab"
```

```
## [119] "permpavsvudcd"            "precrotgrazing"
## [121] "rr"                       "scp1"
## [123] "scp11"                    "scp2"
## [125] "scp3"                     "scp4"
## [127] "scp5"                     "scp6"
## [129] "septicconnect"            "septicdecon"
## [131] "septicdeenhance"          "septiceffenhance"
## [133] "septicpump"               "septicseccon"
## [135] "septicsecenhance"         "shoreag"
## [137] "shoreagnoveg"             "shoreurb"
## [139] "shoreurbnoveg"            "shoreurbveg"
## [141] "st"                       "treeplant"
## [143] "urbanforplant"            "urbannmmdca"
## [145] "urbannmmddiy"             "urbannmplan"
## [147] "urbantreeplant"           "urbfilterrr"
## [149] "urbfilterst"              "urbstrmrest"
## [151] "vegopchannoudab"          "vegopchannoudcd"
## [153] "watercontstruc"           "wetlandcreateeff"
## [155] "wetlandcreatefloodplain"  "wetlandenhance"
## [157] "wetlandenhanceeff"        "wetlandrestoreeff"
## [159] "wetlandrestorefloodplain" "wetpondwetland"
```

```r
# Plot the number of BMPs involving wet areas versus the total area of water
county_bmp %>%
  # Filter BMPShortName to only include BMPs in wet areas
    filter(str_detect(BMPShortName, "wet")) %>%
  # Remove outlier values of Water Area for visualization
    filter(AWATER10 < 150000000) %>%
  # Create a boxplot of water area across BMPs involving wet areas
    ggplot(., aes(x = BMPShortName, y = AWATER10)) +
  # Change fill color to orchid
    geom_boxplot(fill = "orchid") +
    theme_bw() +
  # Make x-axis labels readable
    theme(axis.text.x=element_text(colour="black",
                                   angle = 40, hjust = 1)) +
    labs(x = "Best Management Practice", y = "Water Area (m2)",
         title = "Relationship between Numbers of Best Management Practices \n
         Involving Wet Areas and Water Area")
```



Task 2: Spatial operations

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

First, I arranged the rows of data from longest to shortest stream length (LengthKM). Then, I used the slice function to just subset first five rows of data. The five longest streams in descending order are: Little Chiques Creek, Toms Run, Johns Run, Little Trough Creek, and Great Trough Creek.

```
streams %>%
    # Arrange the data by stream length, in descending order
    arrange(desc(LengthKM)) %>%
    # Subset the data with slice to just include the top 5 rows
    slice(1:5)
```

```
## Simple feature collection with 5 features and 89 fields
## Geometry type: LINESTRING
## Dimension:     XY
## Bounding box:  xmin: -79.06165 ymin: 37.96144 xmax: -76.49969 ymax: 40.43248
## Geodetic CRS:  WGS 84
## # A tibble: 5 x 90
##   OBJECTID_1    ComID Permanent_ FDate      Resolution GNIS_ID GNIS_Name LengthKM
##        <int>    <int> <chr>      <date>     <chr>      <chr>   <chr>        <dbl>
## 1       7470 57464201 NA         1970-01-01 NA         NA      Little C~     6.85
## 2       9032 56409913 NA         1970-01-01 NA         NA      Toms Run      6.54
## 3       6693 41377791 NA         1970-01-01 NA         NA      Johns Run     5.43
```

```
## 4       7794 65839879 NA         1970-01-01 NA         NA      Little T~     5.07
## 5       7848 65842019 NA         1970-01-01 NA         NA      Great Tr~     4.89
## # ... with 82 more variables: ReachCode <chr>, FlowDir <chr>, WBAreaComI <chr>,
## #   WBArea_Per <chr>, FType <int>, FCode <int>, Enabled <chr>, SourceHUC <chr>,
## #   comments <chr>, to_steward <chr>, DeleteMe <chr>, Edits <chr>,
## #   Bifurc <chr>, CtchSqMi <chr>, BayWater <chr>, HYDROID <int>,
## #   FROM_NODE <int>, TO_NODE <int>, NextDownID <int>, DA_SqMi <chr>,
## #   NESZCL <chr>, strahler <int>, shreve <int>, Above10per <chr>,
## #   Shape_Leng <chr>, Region <chr>, Fnode <chr>, Tnode <chr>, ...
```

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

First, I joined the counties and streams spatial data using st_join with an st_intersects join. Then, using this new joined data, I grouped values by county, removed NAs for stream length, calculated the total stream length per county with summarize, reordered these values to be in descending order, and then selected the three counties with the greatest total length of streams. The top three counties were: Augusta, Huntingdon, and Cameron.

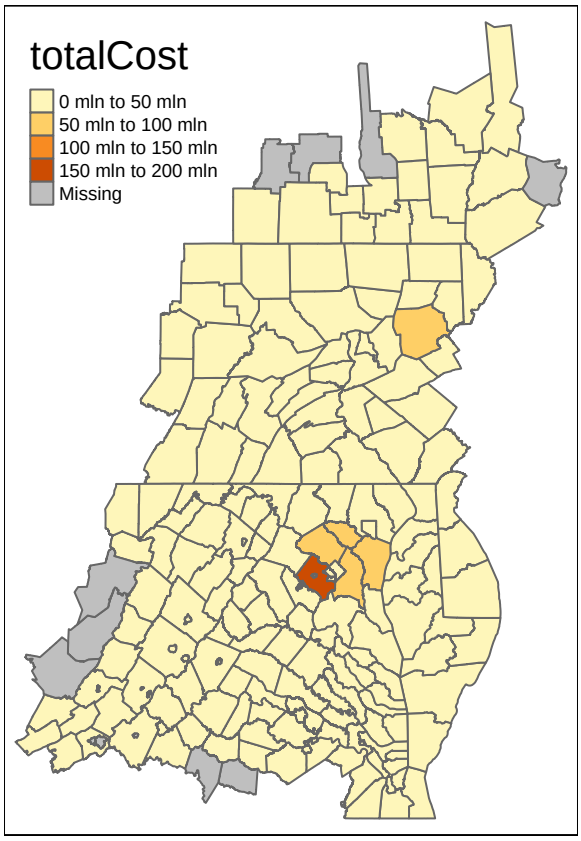
```
# Spatially join the counties and streams data using st_intersects
co_streams <- st_join(counties, streams, join = st_intersects)

co_streams %>%
  # Group the joined data by county name
    group_by(NAME10) %>%
  # Remove NA values for stream length
    drop_na(LengthKM) %>%
  # Create a variable that represents the sum (total) length of streams in each county
    summarize(totalStreamsKM = sum(LengthKM)) %>%
  # Arrange the total length of streams per county in descending order
    arrange(desc(totalStreamsKM)) %>%
  # Show the top 3 counties
    slice(1:3)
```

```
## # A tibble: 3 x 2
##   NAME10     totalStreamsKM
##   <chr>               <dbl>
## 1 Augusta              425.
## 2 Huntingdon           309.
## 3 Cameron              242.
```

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

First, I used the same code as above to trim and create a county FIPS code in the bmps data using str_sub(). Then, I grouped the bmps data by county, removed NA values for the Cost variable, and calculated the total cost of BMPs by county using summarise(), creating a new variable, totalCost, in a new tibble, BMPs. I then performed a left join by county on the counties and BMPs data sets, and mapped the counties, color coded by cost. We see on the map that most counties had a total cost of BMPs less than or equal to $50 million, although a few counties had higher costs.

```
BMPs <- bmps %>%
     # Create County FIPS Codes for bmps data with str_sub()
        mutate(., FIPS.trimmed = stringr::str_sub(GeographyName, 1, 5))  %>%
     # Group the BMP data by county
        group_by(FIPS.trimmed) %>%
     # Remove NA values for Cost
        drop_na(Cost) %>%
     # Calculate the total cost of BMPs in that county
        summarise(totalCost = sum(Cost))

# Join the bmps data with the counties, by county code
county_bmp <- left_join(counties, BMPs, by = c("GEOID10" = "FIPS.trimmed"))

# Map counties and shade by BMP cost
tm_shape(county_bmp) +
          tm_polygons(col = "totalCost") +
          tm_layout(legend.title.size = 1.5,
          legend.text.size = 0.6,
          legend.position = c("left","top"))
```



2.4 For each removed dam, find the closest stream segment

First, I reduced the sizes of the dams and streams datasets by only selecting the relevant varibles in each with select(). Then, I used st_nearest_feature() to find the nearest stream to each dam, and then took it a step further by calculating the distance between the dam/stream (not surprisingly, most were very close).

Finally, I used cbind() to combine the dam names, nearest stream names, and information about the distance between them and the geometry of each feature.

```
# Create a new tibble with only the relevant variables in the dams data
damsGeom <- dams %>%
          dplyr::select(STATE,DAM_NAME, geometry)
# Create a new tibble with only the relevant variables in the streams data
streamsGeom <- streams %>%
          dplyr::select(GNIS_Name,LengthKM,Shape_Leng, Shape_Le_1, geometry)

st_join(damsGeom, streamsGeom, st_nearest_feature)
```

```
## Simple feature collection with 34 features and 6 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -79.03091 ymin: 37.23352 xmax: -75.53803 ymax: 41.71974
## Geodetic CRS:  WGS 84
## # A tibble: 34 x 7
##    STATE DAM_NAME                    geometry GNIS_Name      LengthKM Shape_Leng
##  * <chr> <chr>                    <POINT [°]> <chr>             <dbl> <chr>
##  1 MD    BARREN CREEK D~  (-75.73443 38.4618) Barren Creek    0.0277  NA
##  2 PA    KLADDER RESERV~  (-78.3887 40.38991) NA              0.00593 NA
##  3 PA    Lake Lehman Dam (-76.86169 39.86292) NA              0.0466  NA
##  4 PA    PICRIC           (-78.2551 41.51775) Driftwood Bra~  0.215   NA
##  5 PA    DUCK MARSH PON~ (-78.32606 41.23462) NA              0.0989  NA
##  6 PA    LONG RUN        (-77.35531 41.06447) Long Run        0.00685 NA
##  7 PA    TROUGH CREEK    (-78.12468 40.33215) NA              0.102   NA
##  8 PA    OLYPHANT NO 1   (-75.53803 41.46384) Grassy Island~  0.0710  NA
##  9 PA    HEISTAND SAWMI~ (-76.52639 40.05524) Chiques Creek   0.0840  NA
## 10 PA    SICO            (-76.49158 40.11082) Little Chique~  0.899   NA
## # ... with 24 more rows, and 1 more variable: Shape_Le_1 <dbl>
```

```
# Determine the closest stream segment to each dam using st_nearest_feature()
(nearest = st_nearest_feature(damsGeom,streamsGeom))
```

```
##  [1]   53 2483 1711 2282 1758 2667 2689 2674 1700 1670 1747 2192 2656 2589 2653
## [16] 2590  450 2690 1049 1050 1595 2622 1626 2170 1502 1592 2477 2595 3188 3189
## [31] 2908 3158 3184 3186
```

```
# Calculate the distance between each dam and the closest stream
(dist = st_distance(damsGeom,streamsGeom[nearest,], by_element=TRUE))
```

```
## Units: [m]
##  [1]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##  [7]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [13]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000 297.5348452
## [19]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [25]   0.0000000   0.0000000   0.0000000   0.0000000   0.9910594   0.6072447
## [31]  14.0294981   2.9177132   7.5914935   0.7485873
```

```
# Create a tibble that includes the dam names, nearest stream names, and distances between them
(DamStreamjoin = cbind(damsGeom, streamsGeom[nearest,], dist))
```

```
## Simple feature collection with 34 features and 7 fields
## Active geometry column: geometry
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -79.03091 ymin: 37.23352 xmax: -75.53803 ymax: 41.71974
## Geodetic CRS:  WGS 84
## First 10 features:
##    STATE               DAM_NAME                          GNIS_Name
## 1     MD       BARREN CREEK DAM                       Barren Creek
## 2     PA      KLADDER RESERVOIR                              <NA>
## 3     PA        Lake Lehman Dam                              <NA>
## 4     PA                PICRIC Driftwood Branch Sinnemahoning Creek
## 5     PA DUCK MARSH POND NO. 26                              <NA>
## 6     PA               LONG RUN                         Long Run
## 7     PA           TROUGH CREEK                              <NA>
## 8     PA          OLYPHANT NO 1                Grassy Island Creek
## 9     PA       HEISTAND SAWMILL                      Chiques Creek
## 10    PA                   SICO                Little Chiques Creek
##       LengthKM Shape_Leng  Shape_Le_1  dist                   geometry
## 1  0.027667973       <NA>   35.382376 0 [m]  POINT (-75.73443 38.4618)
## 2  0.005931834       <NA>    7.782023 0 [m]  POINT (-78.3887 40.38991)
## 3  0.046644505       <NA>   60.972156 0 [m] POINT (-76.86169 39.86292)
## 4  0.215383229       <NA>  287.506387 0 [m]  POINT (-78.2551 41.51775)
## 5  0.098910443       <NA>  132.062271 0 [m] POINT (-78.32606 41.23462)
## 6  0.006846220       <NA>    9.060954 0 [m] POINT (-77.35531 41.06447)
## 7  0.101567334       <NA>  133.673744 0 [m] POINT (-78.12468 40.33215)
## 8  0.070986217       <NA>   95.063234 0 [m] POINT (-75.53803 41.46384)
## 9  0.083981675       <NA>  110.529909 0 [m] POINT (-76.52639 40.05524)
## 10 0.899333612       <NA> 1176.651266 0 [m] POINT (-76.49158 40.11082)
##                       geometry.1
## 1  LINESTRING (-75.7342 38.461...
## 2  LINESTRING (-78.38866 40.38...
## 3  LINESTRING (-76.86123 39.86...
## 4  LINESTRING (-78.25676 41.51...
## 5  LINESTRING (-78.32711 41.23...
## 6  LINESTRING (-77.35527 41.06...
## 7  LINESTRING (-78.12563 40.33...
## 8  LINESTRING (-75.53728 41.46...
## 9  LINESTRING (-76.5254 40.055...
## 10 LINESTRING (-76.49178 40.11...
```

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

First, I grouped the dams data by state using group_by(), and then used the n_distinct() function within summarize() to count the number of removed dams per state (all dams in this dataset were removed, so no extra filtering was required). We see that 2 dams were removed in Maryland, 27 were removed in Pennsylvania, and 5 were removed in Virginia.

```
dams %>%
     # Group the BMP data by county
```

```
    group_by(STATE) %>%
  # Calculate the total cost of BMPs in that county
    summarise(NumberDamsRemoved = n_distinct(DAM_NAME))
```

```
## Simple feature collection with 3 features and 2 fields
## Geometry type: MULTIPOINT
## Dimension:     XY
## Bounding box:  xmin: -79.03091 ymin: 37.23352 xmax: -75.53803 ymax: 41.71974
## Geodetic CRS:  WGS 84
## # A tibble: 3 x 3
##   STATE NumberDamsRemoved                                              geometry
##   <chr>             <int>                                       <MULTIPOINT [°]>
## 1 MD                    2             ((-75.73443 38.4618), (-76.06189 39.048))
## 2 PA                   27 ((-75.53803 41.46384), (-75.90023 41.20816), (-76.053~
## 3 VA                    5 ((-77.41335 37.23352), (-78.89962 38.05994), (-78.891~
```