# 2: R and Data Visualization Boot Camp

*Hydrologic Data Analysis | Kateri Salk*

*Fall 2019*

## Lesson Objectives

1. Justify components of effective data visualization
2. Apply R coding skills to create effective visualizations
3. Review and/or develop skills in R coding and syntax

## Opening Discussion

What makes an effective data visualization? What are the essential components/characteristics, and what are optional but nice-to-have components/characteristics?

Come up with a consensus list among your group members.

Accessible and attractive and logical color patterns; units; axis labels; legible font sizes; data point-size-large/small enough; heading (context-dependent); appropriate graph type; axis scales interactive

Compare your list to other groups' lists.

## Exploring web resources for visualization

1. Navigate to https://www.data-to-viz.com/.
2. Think of a dataset that you or someone in your field might want to visualize. What types of variables does this dataset contain? What information might you want to convey with a visualization?

weather data: date, cloud thickness, rainfall, location

3. Scroll to the "Explore" section of the website. Follow the flow chart to find a type of graph that would suit your data. Click for more information and a link to the R Graph Gallery. If there are multiple options for graphing your data, how might you choose between them?
4. Discuss and compare your findings with a partner.
5. Scroll further down to the "Caveats" section of the website and click "See the Collection." Choose five topics to explore (some of these are incomplete so choose another if that is the case). Write some notes about what you learned below.
6. Discuss and compare your findings with a partner.

## A note on color palettes

In general, the ggplot base color palettes are not the most effective option for data visualization. Compiled below is a list of color palettes available in R that may serve your purposes better. Note that when working with multiple colors, you must choose between sequential, diverging, and qualitative color palettes. Under what circumstances might each be useful?

RColorBrewer (package)

- http://colorbrewer2.org
- https://moderndata.plot.ly/create-colorful-graphs-in-r-with-rcolorbrewer-and-plotly/

viridis and viridisLite (packages)

- https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html
- https://ggplot2.tidyverse.org/reference/scale_viridis.html

colorRamp (function; comes with base R as part of the grDevices package)

- https://bookdown.org/rdpeng/exdata/plotting-and-color-in-r.html#colorramp

LaCroixColoR (package)

- https://github.com/johannesbjork/LaCroixColoR

wesanderson (package)

- https://github.com/karthik/wesanderson

pal_futurama (function; comes in the ggsci package)ins

- https://nanx.me/ggsci/reference/pal_futurama.html

## Session Set Up

```
getwd()
```

```
## [1] "/Users/yixinwen/Box/Duke/2019 Fall/Hydrologic Data Analysis/Hydrologic_Data_Analysis/Lessons"
```

```
#install.packages(tidyverse)
#install.packages(dataRetrieval)
#install.packages(zoo)
#install.packages(ggrepel)

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

```
library(dataRetrieval)
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggrepel)
```

## Data Visualization Challenge

We will be exploring discharge data for the Eno River, aided by the dataRetrieval package. A basic ggplot is provided for you, and an image of an edited graph is provided along with hints of how to produce it.

Work with your group to reproduce the graph. This may require dividing up tasks and sharing code as you complete each piece of editing.

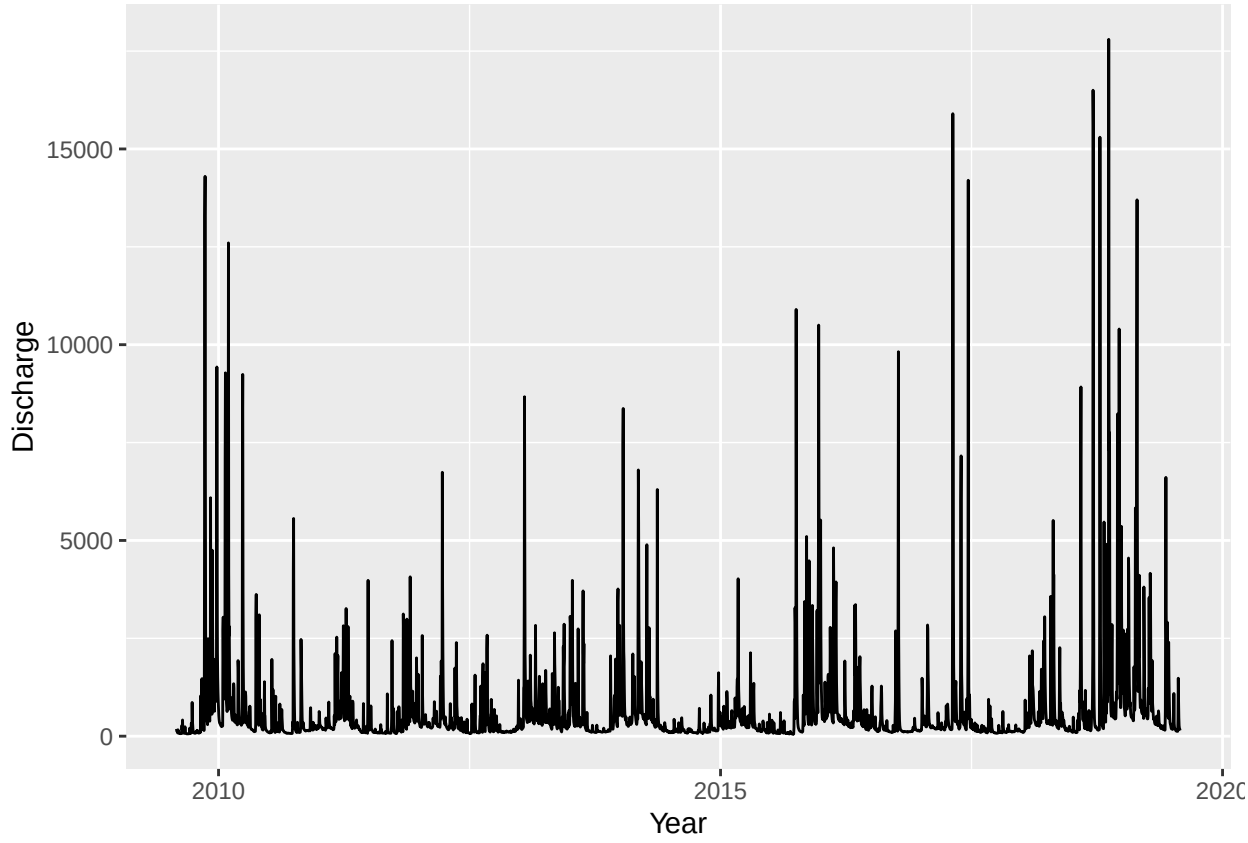
```r
# Import data
EnoDischarge <- readNWISdv(siteNumbers = "02096500",
                     parameterCd = "00060", # discharge (ft3/s)
                     startDate = "2009-08-01",
                     endDate = "2019-07-31")

# Look at the data frame in your Environment tab.

# Renaming columns (one method of multiple)
names(EnoDischarge)[4:5] <- c("Discharge", "Approval.Code")

# dataRetrieval also includes attribute information
attr(EnoDischarge, "variableInfo")
```

```
##   variableCode           variableName              variableDescription
## 1        00060 Streamflow, ft&#179;/s Discharge, cubic feet per second
##       valueType  unit options noDataValue
## 1 Derived Value ft3/s    Mean          NA
```

```r
  # note: imperial, not metric
attr(EnoDischarge, "siteInfo")
```

```
##                    station_nm  site_no agency_cd timeZoneOffset
## 1 HAW RIVER AT HAW RIVER, NC 02096500      USGS         -05:00
##   timeZoneAbbreviation dec_lat_va dec_lon_va       srs siteTypeCd    hucCd
## 1                  EST   36.08722  -79.36611 EPSG:4326         ST 03030002
##   stateCd countyCd network
## 1      37    37001    NWIS
```

```r
# Build a ggplot
EnoPlot <-
  ggplot(EnoDischarge, aes(x = Date, y = Discharge)) +
         geom_line() +
         xlab("Year")
print(EnoPlot)
```




Challenge:

1. Build a plot called EnoPlot2.
2. Add a column to your data frame for discharge in meters cubed per second

hint: package dplyr in tidyverse includes a `mutate` function

3. Add a column in your data frame for a 30-day rolling mean of the metric discharge

hint: package dplyr in tidyverse includes a `mutate` function hint: package zoo includes a `rollmean` function

4. Update your ggplot theme

hint: https://ggplot2.tidyverse.org/reference/ggtheme.html

5. Edit axes
6. Create two geom_line aesthetics, one for metric discharge and one for rolling mean of metric discharge. Color these differently.
7. Label the day that Hurricane Florence hit North Carolina.

hint: look up this date hint: package ggrepel includes a `geom_label_repel` function

8. Add a legend.

hint: Google "add legend two geom layers ggplot"

## Closing Discussion

In what ways was the second plot a more effective visualization than the first?

What portions of the coding were challenging for you?