Analysis of Flow Data

Eric Hettler, Wisconsin DNR

2021-09-14

## Intro

This example uses the USGS station at the Manitowoc River in Wisconsin (04085427) to explore the functionality of the dataRetrieval package.

## Setup

## Step 1: Define station of interest and get information about the station

Information about the data availability and characteristics at the station of interest are evaluated and summarized. Code is provided to evaluate data availability for both discharge and water quality data. For the purposes of this exercise, however, only the results from the discharge data analysis are used.

library(dplyr)  
  
# set station\_no to USGS code - this example uses the USGS gage for the Manitowoc River at Manitowoc (04085427)  
station\_no <- "04085427"  
  
#select parameter for discharge ("00060")  
parameter\_cd <- "00060"  
  
# download information about the NWIS site  
station\_info <- dataRetrieval::readNWISsite(station\_no)  
  
# check station name  
station\_info$station\_nm

## [1] "MANITOWOC RIVER AT MANITOWOC, WI"

# load a table with NWIS codes  
nwis\_codes <- read.csv(file = "./Input/nwis\_parameter\_codes.csv")  
  
# add a column to nwis\_codes that is the same name as the parameter code in whatNWISdata  
nwis\_codes <- nwis\_codes %>%  
 mutate(parm\_cd = parameter\_cd)  
  
# identify what daily data are available for station  
daily\_data\_availability <- dataRetrieval::whatNWISdata(siteNumber = station\_no, service = "dv", statCd = "00003")  
  
# join the nwis\_code data with the daily\_data\_availability data  
daily\_data\_availability <- left\_join(daily\_data\_availability, nwis\_codes, by = "parm\_cd")  
  
# trim down daily\_data\_availablility to review only relevant parameters  
daily\_data\_availability <- daily\_data\_availability %>%  
 select(site\_no, station\_nm, parm\_cd, parameter\_nm, begin\_date, end\_date, count\_nu)  
  
# view daily\_data\_availability  
daily\_data\_availability

## site\_no station\_nm parm\_cd  
## 1 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00010  
## 2 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00060  
## 3 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00095  
## 4 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00300  
## 5 04085427 MANITOWOC RIVER AT MANITOWOC, WI 63680  
## parameter\_nm  
## 1 Temperature, water, degrees Celsius  
## 2 Discharge, cubic feet per second  
## 3 Specific conductance, water, unfiltered, microsiemens per centimeter at 25 degrees Celsius  
## 4 Dissolved oxygen, water, unfiltered, milligrams per liter  
## 5 Turbidity, water, unfiltered, monochrome near infra-red LED light, 780-900 nm, detection angle 90 +-2.5 degrees, formazin nephelometric units (FNU)  
## begin\_date end\_date count\_nu  
## 1 2011-03-18 2021-09-14 2414  
## 2 1972-07-26 2021-09-14 17521  
## 3 2011-03-18 2021-09-14 2328  
## 4 2011-03-18 2021-09-14 2440  
## 5 2011-03-18 2021-09-13 2001

# do the same for water quality data  
wq\_data\_availability <- whatNWISdata(siteNumber = station\_no, service = "qw", statCd = "00003")  
wq\_data\_availability <- left\_join(wq\_data\_availability, nwis\_codes, by = "parm\_cd")  
wq\_data\_availability <- wq\_data\_availability %>%  
 select(site\_no, station\_nm, parm\_cd, parm\_grp\_cd, parameter\_nm, begin\_date, end\_date, count\_nu)  
  
# water quality data have many parameters with only a few samples; look for parameters with >100  
wq\_data\_availability <- wq\_data\_availability %>%  
 filter(!is.na(parm\_cd)) %>% #remove NA values  
 filter(parm\_grp\_cd == "NUT") %>% #select only nutrient data  
 filter(count\_nu >= 100) #remove values with fewer than 100 samples  
  
# view wq\_data\_availability  
wq\_data\_availability

## site\_no station\_nm parm\_cd parm\_grp\_cd  
## 1 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00605 NUT  
## 2 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00608 NUT  
## 3 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00613 NUT  
## 4 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00618 NUT  
## 5 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00631 NUT  
## 6 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00660 NUT  
## 7 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00665 NUT  
## 8 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00671 NUT  
## 9 04085427 MANITOWOC RIVER AT MANITOWOC, WI 62855 NUT  
## 10 04085427 MANITOWOC RIVER AT MANITOWOC, WI 71846 NUT  
## 11 04085427 MANITOWOC RIVER AT MANITOWOC, WI 71851 NUT  
## 12 04085427 MANITOWOC RIVER AT MANITOWOC, WI 71856 NUT  
## parameter\_nm  
## 1 Organic nitrogen, water, unfiltered, milligrams per liter as nitrogen  
## 2 Ammonia (NH3 + NH4+), water, filtered, milligrams per liter as nitrogen  
## 3 Nitrite, water, filtered, milligrams per liter as nitrogen  
## 4 Nitrate, water, filtered, milligrams per liter as nitrogen  
## 5 Nitrate plus nitrite, water, filtered, milligrams per liter as nitrogen  
## 6 Orthophosphate, water, filtered, milligrams per liter as PO4  
## 7 Phosphorus, water, unfiltered, milligrams per liter as phosphorus  
## 8 Orthophosphate, water, filtered, milligrams per liter as phosphorus  
## 9 Total nitrogen [nitrate + nitrite + ammonia + organic-N], water, unfiltered, analytically determined, milligrams per liter  
## 10 Ammonia (NH3 + NH4+), water, filtered, milligrams per liter as NH4  
## 11 Nitrate, water, filtered, milligrams per liter as nitrate  
## 12 Nitrite, water, filtered, milligrams per liter as nitrite  
## begin\_date end\_date count\_nu  
## 1 1979-03-29 2021-08-03 321  
## 2 1979-10-23 2021-08-03 313  
## 3 1985-11-13 2021-08-03 270  
## 4 1985-11-13 2021-08-03 270  
## 5 1979-09-18 2021-08-03 314  
## 6 1980-05-07 2021-08-03 330  
## 7 1979-03-29 2021-08-03 364  
## 8 1980-05-07 2021-08-03 330  
## 9 2008-05-28 2021-08-03 234  
## 10 1979-10-23 2021-08-03 313  
## 11 1985-11-13 2021-08-03 270  
## 12 1985-11-13 2021-08-03 270

# extract data just for the discharge data  
discharge\_details <- daily\_data\_availability %>%  
 filter(parm\_cd == "00060")  
  
# view discharge\_details  
discharge\_details

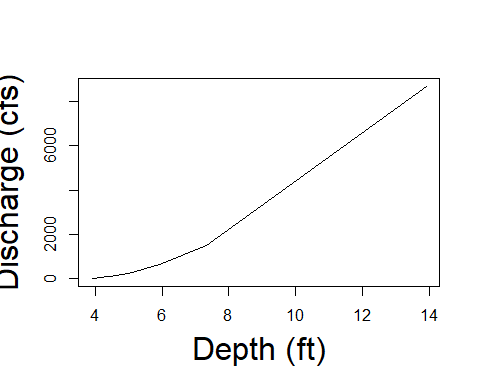
## site\_no station\_nm parm\_cd  
## 1 04085427 MANITOWOC RIVER AT MANITOWOC, WI 00060  
## parameter\_nm begin\_date end\_date count\_nu  
## 1 Discharge, cubic feet per second 1972-07-26 2021-09-14 17521

# find start date and end date of discharge data  
start\_date <- discharge\_details$begin\_date  
end\_date <- discharge\_details$end\_date  
  
  
# remove unncessary parameters  
rm(daily\_data\_availability, nwis\_codes, wq\_data\_availability, discharge\_details)

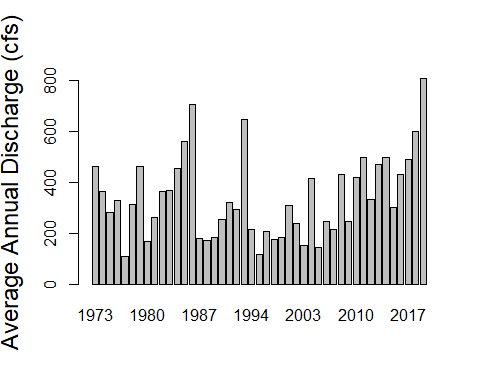
## Step 2: Download data from NWIS

Data are downloaded from NWIS. The following items are downloaded: - Daily discharge data for entire period of record - Instantaneous discharge data since 2020-01-01 - Rating curve for USGS site - Monthly flow statistics - Annual flow statistics

#select parameter for discharge ("00060")  
parameter\_code <- "00060"  
  
#load daily flow data for the site  
discharge\_data <- dataRetrieval::readNWISdv(siteNumbers = station\_no,   
 parameterCd = parameter\_code,   
 startDate = start\_date,   
 endDate = end\_date)  
  
# load instantaneous data for the site  
inst\_discharge\_data <- dataRetrieval::readNWISuv(siteNumbers = station\_no,   
 parameterCd = parameter\_code,   
 startDate = as.Date("2020-01-01"),   
 endDate = end\_date,   
 tz = "America/Chicago")  
  
# load rating curve data for the site  
rating\_curve <- dataRetrieval::readNWISrating(siteNumber = station\_no)  
  
# plot rating curve  
plot(DEP ~ INDEP,   
 data = rating\_curve,   
 type = "l",   
 xlab = "Depth (ft)",   
 ylab = "Discharge (cfs)",  
 cex.lab = 2)



# load monthly summary statistics for site  
monthly\_stats <- dataRetrieval::readNWISstat(siteNumbers = station\_no,   
 parameterCd = parameter\_code,   
 statReportType = "monthly")  
  
# load annual summary statistics for site  
annual\_stats <- dataRetrieval::readNWISstat(siteNumbers = station\_no,  
 parameterCd = parameter\_code,  
 statReportType = "annual")  
  
# plot annual summary  
barplot(mean\_va ~ year\_nu,  
 data = annual\_stats,  
 type = "bar",  
 xlab = element\_blank(),  
 ylab = "Average Annual Discharge (cfs)",  
 cex.lab = 1.5)

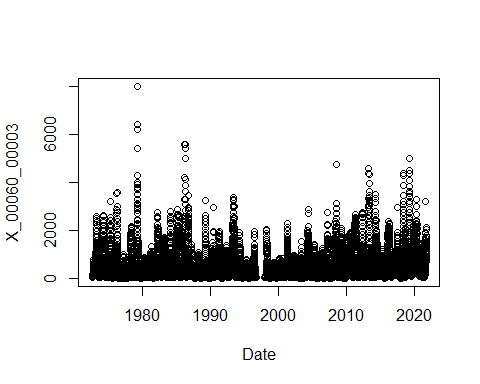


#remove newly created objects  
rm(inst\_discharge\_data, rating\_curve, monthly\_stats, annual\_stats)

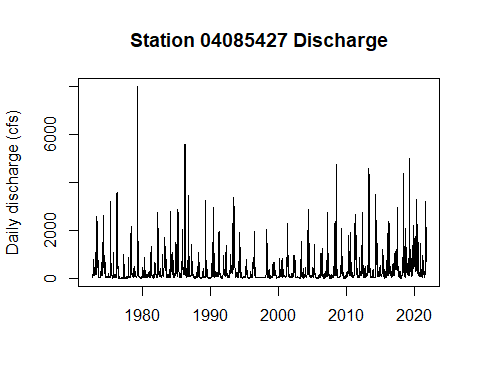
## Step 3: Explore data with base plotting functions in R

Data downloaded in Step 2 is evaluated and summarized using the base plotting functions provided in R. In this section, a function called make\_plot is created which allows the user to easily create formatted plots by entering specified start and end dates. The end of the chunk downloads the resulting figure to a jpeg file.

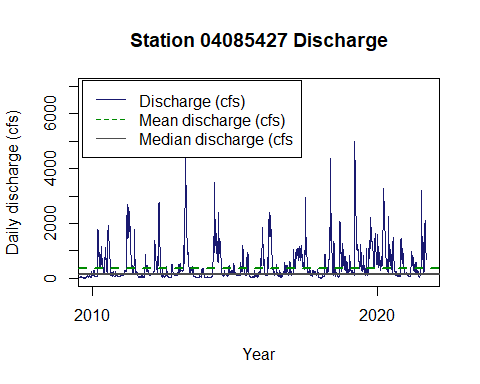
#use base plot functions to explore data - plots as points  
plot(X\_00060\_00003 ~ Date, data = discharge\_data)



##use additional parameters in base plot function to clean up figure  
#set plot title name  
plot\_title <- paste("Station", station\_no, "Discharge", sep = " ")  
  
#use the base plot function to plot discharge data  
plot(X\_00060\_00003 ~ Date,   
 data = discharge\_data,  
 main = plot\_title,  
 xlab = element\_blank(),  
 ylab = "Daily discharge (cfs)",  
 type = "l")



# create a function that creates the plot  
make\_plot <- function(startDate, endDate){  
   
 # use the base plot function to format the plot  
 plot(X\_00060\_00003 ~ Date,   
 data = discharge\_data,  
 main = plot\_title,  
 xlab = "Year",  
 ylab = "Daily discharge (cfs)",  
 type = "l",  
 col = "midnightblue",  
 xlim = c(startDate, endDate),  
 ylim = c(0,7000)  
 )  
   
 # add lines for mean and median discharges  
 abline(h = median(discharge\_data$X\_00060\_00003, na.rm = TRUE),   
 col = "gray30",   
 lwd = 2)  
 abline(h = mean(discharge\_data$X\_00060\_00003, na.rm = TRUE),   
 col = "green4",   
 lty = "dashed",   
 lwd = 2)  
   
 #add a legend  
 legend("topleft",   
 legend = c("Discharge (cfs)", "Mean discharge (cfs)", "Median discharge (cfs"),   
 col = c("midnightblue", "green4", "gray30"),   
 lty = 1:2,   
 inset = 0.01)  
  
}  
  
#call function make\_plot to plot function  
make\_plot(startDate = as.Date("2010-01-01"), endDate = end\_date)



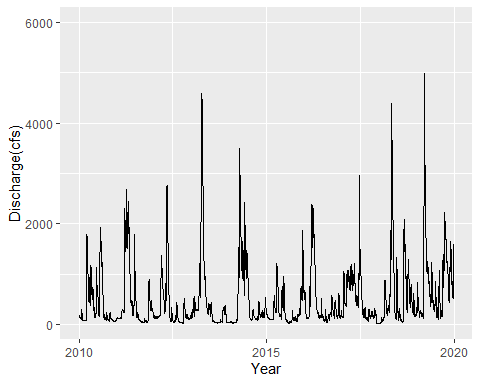
##save figure to a .jpeg file  
#create a file name for the .jpeg file  
file\_name <- paste0("./Output/discharge\_", station\_no, ".jpeg")  
  
#initialize the jpeg save function  
jpeg(file = file\_name, width = 600, height = 400, units = "px")  
  
# remove unncessary variables  
rm(file\_name, plot\_title, make\_plot)

## Step 4: Explore data with ggplot function

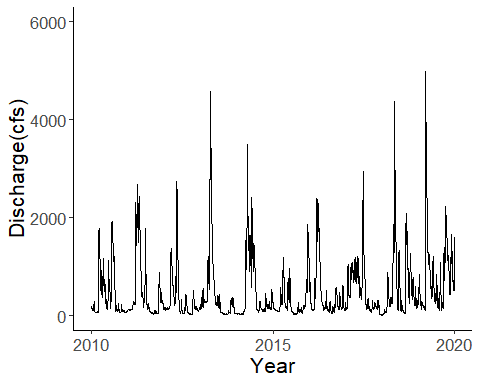
Data are plotted using the ggplot2 package. ggplot allows users to easily create and format plots. Plotting with ggplot is more flexible than the base R plotting functions because it uses the “Grammar of Graphics” to allow easy additions to the plots. The following plots are created:

* Discharge over time
* Comparison of two years
* Regression comparing annual flow over time
* Average monthly flow

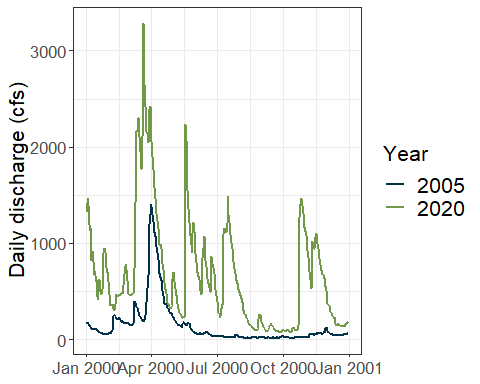
# plot the discharge data with ggplot  
p <- ggplot2::ggplot(aes(Date, X\_00060\_00003), data = discharge\_data) +   
 geom\_line() +  
 xlab("Year") +   
 ylab ("Discharge(cfs)") +   
 xlim(c(as.Date("2010-01-01"), as.Date("2019-12-31"))) +   
 ylim(c(0,6000))  
  
# print the figure  
p



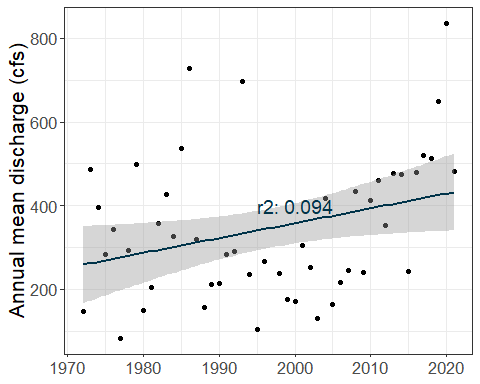
#set a new theme for the figure  
theme\_set <- theme\_bw() +  
 theme(axis.line = element\_line(color = 'black'),  
 panel.grid.major = element\_blank(),   
 panel.grid.minor = element\_blank(),  
 panel.background = element\_blank(),  
 panel.border = element\_blank(),  
 text = element\_text(size = 16))  
   
# update figure to include labels, themes, and axis limits  
p1 <- p +  
 xlab("Year") +   
 ylab ("Discharge(cfs)") +   
 xlim(c(as.Date("2010-01-01"), as.Date("2019-12-31"))) +   
 ylim(c(0,6000)) +   
 theme\_set  
  
# print the figure  
p1



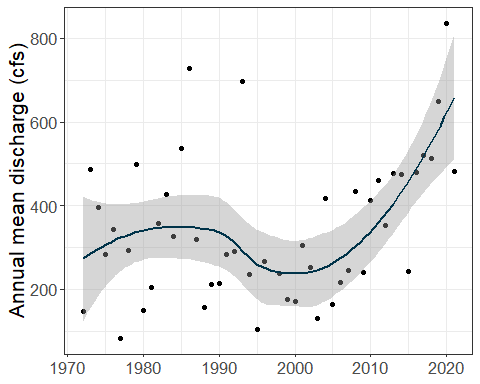
#remove unncessary objects  
rm(p, p1, theme\_set)  
  
## compare two specific years of data  
# create a function that accepts two years for flow comparison and returns a plotting function  
compare\_yrs\_plot <- function(year1, year2){  
   
 #add year, month, and day columns to the discharge data  
 discharge\_date\_adj <- discharge\_data %>%  
 mutate(d = day(Date),  
 mo = month(Date),  
 yr = year(Date),  
 discharge = X\_00060\_00003) %>%  
 select(yr, mo, d, discharge)  
   
 #filter discharge data for the first year of the input function  
 yr1 <- discharge\_date\_adj %>%  
 filter(yr == year1) %>%  
 mutate(adj\_date = make\_date(year = 2000, month = mo, day = d)) %>%  
 mutate(yr = as.character(yr)) %>%  
 select(yr, adj\_date, discharge)  
  
 #filter discharge data for the second year of the input function  
 yr2 <- discharge\_date\_adj %>%  
 filter(yr == 2020) %>%  
 mutate(adj\_date = make\_date(year = 2000, month = mo, day = d)) %>%  
 mutate(yr = as.character(yr)) %>%  
 select(yr, adj\_date, discharge)  
   
 #combine the two years of data into a single data frame  
 yr1yr2\_compare <- rbind(yr1, yr2)  
   
 #set the year of data as a character for plotting  
 yr1yr2\_compare <- yr1yr2\_compare %>%  
 mutate(yr = as.character(yr))  
   
 theme\_set <- theme\_bw() +  
 theme(  
 panel.background = element\_blank(),  
 axis.title.x = element\_blank(),  
 text = element\_text(size = 16),  
 legend.text = element\_text(size = 16)  
 )   
   
 #create a plotting function in ggplot to compare the years  
 p\_compare\_yrs <- ggplot2::ggplot(data = yr1yr2\_compare) +   
 geom\_line(aes(adj\_date, discharge, color = yr), size = 1) +  
 scale\_color\_manual(values = c("#003349", "#719949")) +   
 ylab("Daily discharge (cfs)") +  
 labs(color = "Year", size = 16) +  
 theme\_set  
   
 #return the plotting function when the function is called  
 return(p\_compare\_yrs)  
   
 }  
  
# set varible p to the returned value of function compare\_yrs\_plot(first date, second date)  
p <- compare\_yrs\_plot(2005, 2020)  
  
# print plot comparing two years  
p



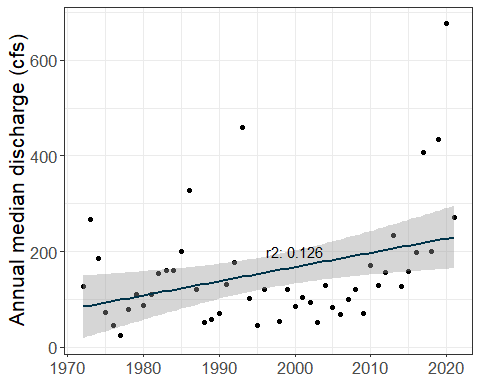
# print plot using ggplotly, which allows for interactive exploration of plot  
#plotly::ggplotly(p)  
  
# remove unncessary objects  
rm(compare\_yrs\_plot, p)  
  
## explore plots for annual average discharge  
# define water year as data between October and September  
  
#add year, month, and day columns to the discharge data  
discharge\_date\_adj <- discharge\_data %>%  
 mutate(d = day(Date),  
 mo = month(Date),  
 yr = year(Date),  
 discharge = X\_00060\_00003) %>%  
 select(yr, mo, d, discharge)  
  
wy\_stats <- discharge\_date\_adj %>%  
 mutate(water\_yr = ifelse(mo >= 10, yr+1, yr))  
  
# summarize discharge data by water year  
wy\_stats <- wy\_stats %>%  
 group\_by(water\_yr) %>%  
 summarize(q\_avg = mean(discharge), q\_med = median(discharge)) %>%  
 ungroup()  
  
#create regression line for the average annual discharge stats  
reg\_line <- lm(wy\_stats$water\_yr ~ wy\_stats$q\_avg)  
reg\_line\_summary <- summary(reg\_line)  
  
#create a theme for the figures  
theme\_set <- theme\_bw() +  
 theme(  
 panel.background = element\_blank(),  
 axis.title.x = element\_blank(),  
 text = element\_text(size = 16),  
 legend.text = element\_text(size = 16)  
 )   
  
  
# create plot with linear regression line  
p\_lreg <- ggplot2::ggplot(aes(water\_yr, q\_avg), data = wy\_stats) +   
 geom\_point() +  
 geom\_smooth(method = 'lm', color = "#003349") +  
 annotate("text",   
 x = 2000,   
 y = 400,   
 label = paste0("r2: ", round(reg\_line\_summary$r.squared,3)),   
 size = 5,   
 color = "#003349") +  
 ylab("Annual mean discharge (cfs)") +  
 theme\_set  
  
# print plot with linear regression  
p\_lreg



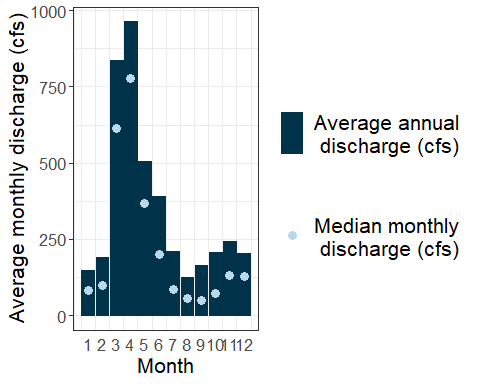
# create plot with smoothed value  
p\_smooth <- ggplot2::ggplot(aes(water\_yr, q\_avg), data = wy\_stats) +   
 geom\_point() +  
 geom\_smooth(method = 'auto', color = "#003349") +  
 ylab("Annual mean discharge (cfs)") +  
 theme\_set  
  
# print plot with smoothed value  
p\_smooth



# create regression line for the median annual discharge stats  
reg\_line\_median <- lm(water\_yr ~ q\_med, data = wy\_stats)  
reg\_line\_median <- summary(reg\_line\_median)  
  
# compare median and mean values for each year  
p\_med <- ggplot2::ggplot(aes(water\_yr, q\_med), data = wy\_stats) +  
 geom\_point() +  
 geom\_smooth(method = 'lm', color = "#003349") +  
 annotate("text",   
 x = 2000,  
 y = 200,  
 label = paste0("r2: ", round(reg\_line\_median$r.squared, 3))) +  
 ylab("Annual median discharge (cfs)") +  
 theme\_set  
  
# print plot to compare mean and median values  
p\_med



# remove unncessary objects  
rm(wy\_stats, reg\_line, reg\_line\_summary, reg\_line\_median, p\_lreg, p\_smooth,   
 p\_med, theme\_set)  
  
## Explore plots for monthly average values  
# summarize data by month  
mo\_stats <- discharge\_date\_adj %>%  
 group\_by(mo) %>%  
 summarize(q\_avg = mean(discharge), q\_med = median(discharge))  
  
# plot average monthly flow for entire time period  
p <- ggplot2::ggplot(data = mo\_stats) +   
 geom\_bar(aes(mo, q\_avg, fill = "Average annual \n discharge (cfs)"), stat = "identity", color = "#003349") +  
 scale\_x\_continuous(breaks = seq(1, 12, 1)) +  
 xlab("Month") +  
 ylab("Average monthly discharge (cfs)") +  
 theme\_bw() +   
 theme(  
 panel.background = element\_blank(),  
 text = element\_text(size = 16),  
 legend.text = element\_text(size = 16),  
 panel.grid.major.x = element\_blank(),  
 axis.ticks.x = element\_blank()  
 ) +  
 geom\_point(aes(mo, q\_med, color = "Median monthly \n discharge (cfs)"), data = mo\_stats, size = 3) +  
 scale\_fill\_manual("", values = c("#003349")) +   
 scale\_color\_manual("", values = c("#B9D9EB"))   
  
#print plot comparing average monthly flow  
p



# remove unncessary objects  
rm(mo\_stats, p, discharge\_date\_adj)

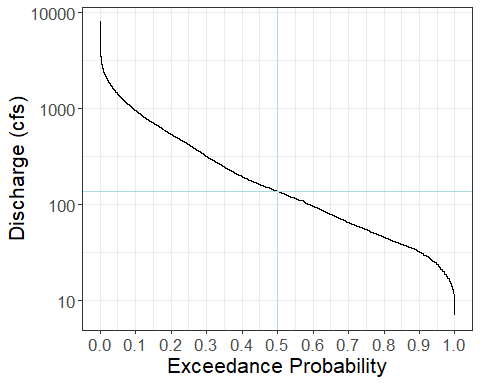
## Step 5: Create flow duration curve

A flow duration curve is created to evaluate the exceedance probability for a specific date in the analysis.

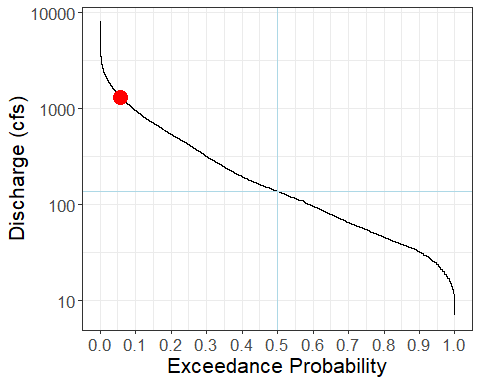
# set a new variable equal to all discharges from discharge data  
fl\_dur\_data <- discharge\_data %>%  
 select(X\_00060\_00003)  
  
# sort the discharge data at the site by decreasing order  
fl\_dur\_data <- as.data.frame(sort(fl\_dur\_data$X\_00060\_00003,   
 decreasing = TRUE))  
  
# change the name of the column from X\_00060\_00003 to discharge  
colnames(fl\_dur\_data) <- c("discharge")  
  
# count the number of rows in the fl\_dur\_data dataframe  
fd\_rows <- nrow(fl\_dur\_data)  
  
# rank the flows and calculate an exceedance probablity  
fl\_dur\_data <- fl\_dur\_data %>%   
 mutate(ranked = 1:fd\_rows) %>%  
 mutate(exceed\_prob = ranked/fd\_rows)  
  
# create a plot in ggplot to display the exceedance probablity versus the discharge  
p <- ggplot2::ggplot(data = fl\_dur\_data, aes(exceed\_prob, discharge)) + geom\_line()  
  
# print plot for exceedance probability  
p



# set a theme for the plotting functions  
theme\_set <- theme\_bw() +  
 theme(  
 panel.background = element\_blank(),  
 text = element\_text(size = 16)  
 )  
  
# create a plot with additional formatting  
p1 <- p + theme\_set +  
 scale\_y\_continuous(trans = "log10") +  
 scale\_x\_continuous(breaks = seq(0,1,0.1)) +  
 xlab("Exceedance Probability") +  
 ylab("Discharge (cfs)") +   
 geom\_vline(xintercept = 0.5, color = "lightblue") +  
 geom\_hline(yintercept = median(fl\_dur\_data$discharge), color = "lightblue")  
  
# print plot with additional formating  
p1



# identify a date of interest to estimate exceedance probablity  
flow\_date = as.Date("2020-01-01")  
  
#filter the discharge data to identify the discharge associated with the specified date  
find\_exceed\_prob <- discharge\_data %>%  
 filter(Date == flow\_date) %>%  
 mutate(discharge = X\_00060\_00003) %>%  
 select(Date, discharge)  
  
#find the exceedance probablity associated with the discharge of interest  
find\_exceed\_prob <- fl\_dur\_data %>%  
 filter(discharge == find\_exceed\_prob$discharge) %>%  
 select(exceed\_prob, discharge)  
  
#add exceedance data for the date of interest to the plot  
p2 <- p1 + geom\_point(data = find\_exceed\_prob, aes(exceed\_prob, discharge), color = "red", size = 5)  
  
# print the plot to identify point of selected flow date  
p2



#remove unncessary objects  
rm(p, p1, p2, fl\_dur\_data, find\_exceed\_prob, theme\_set, fd\_rows, flow\_date)

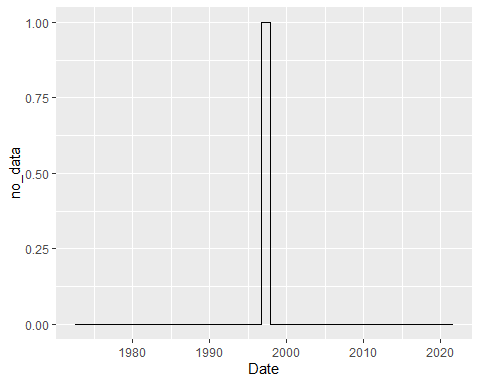
## Step 6: Perform baseflow separation

Baseflow separation helps visualize and calculate the flow originating from baseflow versus flow originating from runoff. Many methods of baseflow separation are available. This example compares baseflow separation calculated by recursive digital filters and the smoothed minima method.

### Step 6a. Evaluate dates discharge data to determine any gaps in data

The baseflow separation methods require a continuous dataset. The long-term discharge data are evaluate to create a continuous dataset with no missing dates.

# find the dates with discharge data  
flow\_dates <- discharge\_data$Date  
  
# create a vector with continuous daily time series between the maximum and minimum dates of discharge data  
date\_range <- seq(min(flow\_dates), max(flow\_dates), by = 1)  
  
# identify missing dates in discharge data  
missing\_dates <- as.data.frame(date\_range[!date\_range %in% flow\_dates])  
colnames(missing\_dates) <- c("Date")  
  
# convert date\_range to a data frame and change column name to Date  
date\_range <- as.data.frame(date\_range)  
colnames(date\_range) <- c("Date")  
  
# identify the dates within the total date range that have no data; set a value equal to 1  
date\_range <- date\_range %>%   
 mutate(no\_data = ifelse(Date %in% missing\_dates$Date, 1, 0))  
  
# create an object to generate a plot for date\_range   
p <- ggplot(aes(Date, no\_data), data = date\_range) +   
 geom\_line()  
p

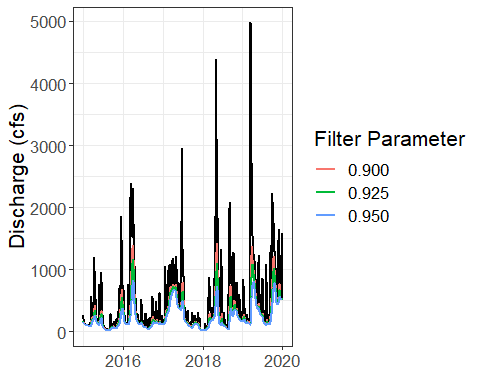


# plot the data using ggplotly (interactive exploration)  
#plotly::ggplotly(p)  
  
## discharge data between 1996-09-30 and 1997-11-30 & 2020-11-10 and 2020-11-16 are missing  
# trim discharge data to 1998-01-01 and 2020-10-31  
bf\_sep\_discharge <- discharge\_data %>%  
 filter(Date >= as.Date("1998-01-01") & Date <= as.Date("2020-10-31")) %>%  
 mutate(discharge = X\_00060\_00003) %>%  
 select(Date, discharge)  
  
# remove objects from previous assessment  
rm(flow\_dates, date\_range, missing\_dates, p)  
  
#create a vector of only discharge data  
discharge\_values <- bf\_sep\_discharge$discharge

### Step 6b. Perform baseflow separation using recursive digital filter

The recursive digital filter uses methods from signal processing to estimate the baseflow. It evaluates discharge data to remove spikes and create a smoothed dataset. The smoothed dataset represents the baseflow.

## perform baseflow separation using BaseflowSeparation in EcoHydRology package  
## BaseflowSeparation uses recursive digital filtering to find baseflow (more info in vignette)  
  
# create a vector with three filter parameters to be evaluated (0.900, 0.925, 0.950)  
filter\_parameters <- c(0.900, 0.925, 0.950)  
#assign a counter function to use in the for loop  
counter <- 1  
  
# run a for loop through all values of filter\_parameters  
for(i in filter\_parameters){  
   
 # use BaseflowSeparation function to estimate baseflow using filter parameter assigned in loop (i)  
 bf\_rdf\_temp <- EcoHydRology::BaseflowSeparation(streamflow = discharge\_values, filter\_parameter = i, passes = 3)  
   
 # combine the original bf\_sep\_discharge data with the baseflow values from BaseflowSeparation  
 bf\_rdf\_temp <- cbind(bf\_sep\_discharge, bf\_rdf\_temp)  
   
 # change the column names in the bf\_rdf\_tmp data frame  
 colnames(bf\_rdf\_temp) <- c("Date", "discharge", "bf\_rdf", "qf\_rdf")  
   
 # add a column to identify which filter paramter is used  
 bf\_rdf\_temp <- bf\_rdf\_temp %>% mutate(filter\_param = i)  
   
 # if the filter parameter is the first in the list, set bf\_rdf\_temp to a new variable bf\_rdf\_all  
 if(counter == 1){bf\_rdf\_all = bf\_rdf\_temp}  
 # if the filter paramter is not the first in the list, append bf\_rdf\_temp to bf\_rdf\_all  
 else{bf\_rdf\_all = rbind(bf\_rdf\_all, bf\_rdf\_temp)}  
  
 #remove temporary data frame  
 rm(bf\_rdf\_temp)  
 counter <- counter + 1  
   
}  
  
rm(filter\_parameters, counter)  
  
# add zeros to the end of the filter parameter id name so decimal taken out to three places  
bf\_rdf\_all$filter\_param <- sprintf("%.3f", bf\_rdf\_all$filter\_param)  
  
# create a ggplot object to compare the results from different filter parameters  
p <- ggplot(data = bf\_rdf\_all, aes(Date, discharge, color = "Discharge")) + geom\_line(color = "black", size = 1) +   
 geom\_line(data = bf\_rdf\_all, aes(Date, bf\_rdf, color = filter\_param), size = 1)  
  
# add formatting to plot  
p <- p + theme\_bw() +   
 xlab(element\_blank()) +   
 ylab("Discharge (cfs)") +   
 xlim(as.Date("2015-01-01"), as.Date("2019-12-31")) +   
 theme(text = element\_text(size = 16)) +  
 labs(color = "Filter Parameter")  
  
# print plot to compare the results from different filter parameters  
p

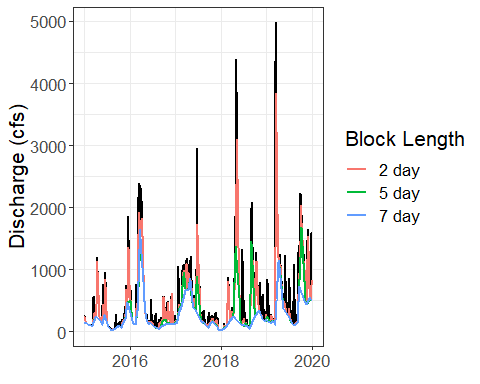


# remove ggplot object  
rm(p, i)  
  
## calculate baseflow index for rdf method  
# create a new matrix to store values for baseflow index for each filter parameter  
bfi\_rdf\_matrix <- matrix(0,length(unique(bf\_rdf\_all$filter)),2)  
  
#rename columns of bfi\_rdf\_matrix  
colnames(bfi\_rdf\_matrix) = c("Filter Parameter", "bfi")  
  
#create a new list of unique block names in bf\_rdf\_all  
filter\_name <- unique(bf\_rdf\_all$filter\_param)  
  
#create a row counting variable  
row\_num = 1  
  
# run a for loop to calculate baseflow index for each filter parameter  
for(i in filter\_name){  
   
 # filter out the value from bfi\_rdf\_all by the block name of the loop  
 bfi\_calc <- bf\_rdf\_all %>%  
 filter(filter\_param == i)  
   
 # calculate the baseflow index by dividing the total baseflow by the total flow  
 bfi\_value <- sum(bfi\_calc$bf\_rdf, na.rm = TRUE)/sum(bfi\_calc$discharge, na.rm = TRUE)  
   
 # assign filter parameter name and baseflow index value to matrix  
 bfi\_rdf\_matrix[row\_num, 1] = i  
 bfi\_rdf\_matrix[row\_num, 2] = round(bfi\_value,3)  
   
 # remove bfi\_calc and bfi\_value   
 rm(bfi\_calc, bfi\_value)  
   
 # add one to row counting varaible  
 row\_num <- row\_num + 1  
   
}  
  
rm(filter\_name, row\_num)

### Step 6c. Perform baseflow separation using smoothed minima method

The smoothed minima method uses an approach that is different from the recursive digital filter. The smoothed minima method evaluates the discharge data to identify local minima. Local minima are connected by straight lines, and the straight lines reprsent the baseflow.

# create a list of different block lengths in days  
block\_length <- c(2, 5, 7)  
  
#create a counter variable for the for loop  
counter <- 1  
  
# run a four loop for the block\_length list   
for(i in block\_length){  
 # calculate the baseflow using the smoothed minima method  
 bf\_smm\_temp <- lfstat::baseflow(discharge\_values, tp.factor = 0.9, block.len = i)  
   
 # combine the discharge data with the smoothed minima data  
 bf\_smm\_temp <- cbind(bf\_sep\_discharge, bf\_smm\_temp)  
   
 # update the column names in bf\_smm\_temp  
 colnames(bf\_smm\_temp) <- c("Date", "discharge", "bf\_smm")  
   
 # add a column to identify the block length  
 bf\_smm\_temp <- bf\_smm\_temp %>% mutate(block = paste0(i, " day"))  
   
 # if the block length is the first in the list, set bf\_sm\_temp equal to a new variable, bf\_smm\_all  
 if(counter == 1){bf\_smm\_all = bf\_smm\_temp}  
 # if block length is not the first in the list, append bf\_smm\_temp to bf\_smm\_all  
 else{bf\_smm\_all = rbind(bf\_smm\_all, bf\_smm\_temp)}  
   
 #remove bf\_smm\_temp  
 rm(bf\_smm\_temp)  
   
 #add one to counter variable  
 counter <- counter + 1  
  
 }  
  
# remove block\_length and counter variables  
rm(block\_length, counter, i)  
  
# create ggplot object for plotting  
p <- ggplot(data = bf\_smm\_all, aes(Date, discharge, color = "Discharge")) + geom\_line(color = "black", size = 1) +  
 geom\_line(data = bf\_smm\_all, aes(Date, bf\_smm, color = block), size = 1)  
  
#add formatting to plot  
p <- p + theme\_bw() +   
 xlab(element\_blank()) +   
 ylab("Discharge (cfs)") +   
 xlim(as.Date("2015-01-01"), as.Date("2019-12-31")) +   
 theme(text = element\_text(size = 16)) +  
 labs(color = "Block Length")  
  
# print plot comparing the block lengths  
p

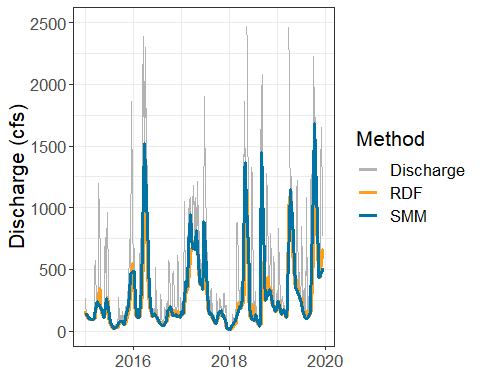


#remove ggplot object  
rm(p)  
  
## calculate baseflow index for smm method  
# create a new matrix to store values for baseflow index for each block  
bfi\_smm\_matrix <- matrix(0,length(unique(bf\_smm\_all$block)),2)  
  
#rename columns of bfi\_smm\_matrix  
colnames(bfi\_smm\_matrix) = c("Block Length", "bfi")  
  
#create a new list of unique block names in bf\_smm\_all  
block\_name <- unique(bf\_smm\_all$block)  
  
#create a row counting variable  
row\_num = 1  
  
# run a for loop to calculate baseflow index for each block length  
for(i in block\_name){  
   
 # filter out the value from bfi\_smm\_all by the block name of the loop  
 bfi\_calc <- bf\_smm\_all %>%  
 filter(block == i)  
   
 # calculate the baseflow index by dividing the total baseflow by the total flow  
 bfi\_value <- sum(bfi\_calc$bf\_smm, na.rm = TRUE)/sum(bfi\_calc$discharge, na.rm = TRUE)  
   
 # assign baseflow name and baseflow index value to matrix  
 bfi\_smm\_matrix[row\_num, 1] = i  
 bfi\_smm\_matrix[row\_num, 2] = round(bfi\_value, 3)  
   
 # remove bfi\_calc and bfi\_value   
 rm(bfi\_calc, bfi\_value)  
   
 # add one to row counting varaible  
 row\_num <- row\_num + 1  
   
}  
  
# remove variables used for calculation of baseflow index  
rm(block\_name, row\_num)  
rm(bfi\_rdf\_matrix, bfi\_smm\_matrix, i)

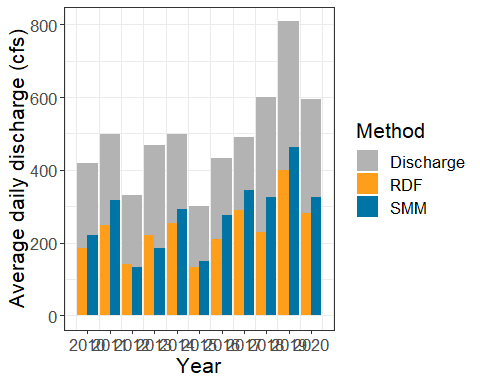
### Step 6d. Compare results from two baseflow methods

The two baseflow methods use different theorteical approaches to estimate baseflow, and the outcome of the separation are not the same. The two methods can be compared to better understand how each method characterizes the baseflow.

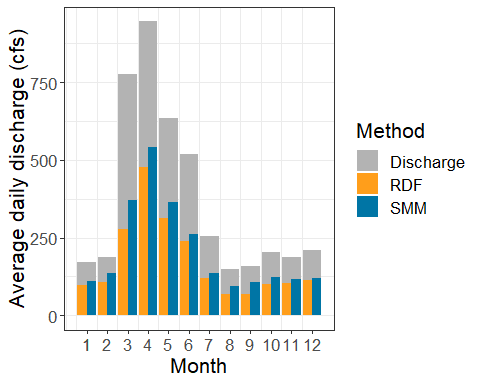
# load lubridate package (contains 'year' and 'month' functions)  
library(lubridate)  
  
# filter the bf\_rdf\_all data frame for specified filter parameter; reformat table  
bf\_rdf <- bf\_rdf\_all %>%  
 filter(filter\_param == "0.925") %>%  
 mutate(method = "RDF", bf = bf\_rdf)%>%  
 select(Date, discharge, bf, method)  
  
# filter the bf\_smm\_all data frame for the specified block length; reformat table  
bf\_smm <- bf\_smm\_all %>%  
 filter(block == "5 day") %>%  
 mutate(method = "SMM", bf = bf\_smm) %>%  
 select(Date, discharge, bf, method)  
  
# combine the results from the recursive digital filter and smoothed minima methods  
bf\_compare <- rbind(bf\_rdf, bf\_smm)  
  
# plot the data for RDF and SMM to compare the results of baseflow separation  
p <- ggplot(data = bf\_compare, aes(Date, discharge, color = "Discharge")) +   
 geom\_line() +  
 geom\_line(data = bf\_compare, aes(Date, bf, color = method), size = 1.1) +  
 labs(color = "Method") +  
 scale\_color\_manual(values = c("grey70","#FF9E1B","#0075A5")) +  
 xlab(element\_blank()) +  
 ylab("Discharge (cfs)") +  
 xlim(c(as.Date("2015-01-01"), as.Date("2019-12-13"))) +  
 scale\_y\_continuous(limits = c(0,2500)) +  
 theme\_bw() +  
 theme(text = element\_text(size = 16))  
  
# print the graph object  
p



# remove unncessary objects  
rm(p, bf\_rdf, bf\_rdf\_all, bf\_smm, bf\_smm\_all, bf\_sep\_discharge)  
## summarize baseflow data by year  
# add year, month, and day columns to bf\_compare  
bf\_compare <- bf\_compare %>%  
 mutate(yr = year(Date), mo = month(Date), d = day(Date))  
  
# summarize data by year and by method (divide discharge by 365 and methods by 365)  
bf\_compare\_yr <- bf\_compare %>%  
 group\_by(yr, method) %>%  
 summarize(discharge = sum(discharge, na.rm = TRUE)/365, bf = sum(bf, na.rm = TRUE)/365)  
  
# create a data frame for average daily discharge per year  
bf\_compare\_yr\_discharge <- bf\_compare\_yr %>%  
 filter(method == "RDF") %>%  
 select(yr, discharge)  
  
# create a bar plot to compare reuslts for the methods  
p <- ggplot(data = bf\_compare\_yr\_discharge, aes(x = yr, y = discharge, fill = "Discharge")) +   
 geom\_bar(stat = "identity") +  
 geom\_bar(data = bf\_compare\_yr,   
 aes(x = yr, y = bf, fill = method),   
 stat = "identity",   
 position = position\_dodge()) +  
 scale\_fill\_manual(values = c("grey70","#FF9E1B","#0075A5")) +  
 theme\_bw() +  
 labs(fill = "Method") +  
 theme(text = element\_text(size = 16), panel.grid.major.x = element\_blank()) +  
 scale\_x\_continuous(name = "Year", breaks = c(2010:2020, 1),limits = c(2009.5,2020.5)) +  
 ylab("Average daily discharge (cfs)")  
  
# plot the bar chart  
p



# remove unncessary objects  
rm(bf\_compare\_yr, bf\_compare\_yr\_discharge, p)  
  
## summarize baseflow data by month  
# calculate the average daily baseflow by month and method  
bf\_compare\_mo <- bf\_compare %>%  
 group\_by(mo, method) %>%  
 summarize(discharge = mean(discharge, na.rm = TRUE), bf = mean(bf, na.rm = TRUE))  
  
# create a data frame for average discharge by month  
bf\_compare\_mo\_discharge <- bf\_compare\_mo %>%  
 filter(method == "RDF") %>%  
 select(mo, discharge)  
  
# create a bar chart to compare the two methods by month  
p <- ggplot(data = bf\_compare\_mo\_discharge, aes(mo, discharge, fill = "Discharge")) +   
 geom\_bar(stat = "identity") +  
 geom\_bar(data = bf\_compare\_mo,   
 aes(mo, bf, fill = method),   
 stat = "identity",   
 position = position\_dodge()) +  
 scale\_fill\_manual(values = c("grey70","#FF9E1B","#0075A5")) +  
 theme\_bw() +  
 labs(fill = "Method") +  
 theme(text = element\_text(size = 16), panel.grid.major.x = element\_blank()) +  
 scale\_x\_continuous(name = "Month", breaks = c(1:12, 1),limits = c(0.5, 12.5)) +  
 ylab("Average daily discharge (cfs)")  
  
#print the bar chart  
p



# remove unncessary objects  
rm(bf\_compare, bf\_compare\_mo, bf\_compare\_mo\_discharge, p)