Introduction-to-R.R

2022-12-11

##########################################  
########Day 1: Introduction to R##########  
##########################################  
  
  
####Topics################################  
# R Studio and working with the console  
# Comments  
# Operators  
# Data Types  
# Creating New Variables  
# Complex data types  
# Functions  
# Packages  
# Coding Tips  
# Getting Help  
# Data exploration  
# Summarizing data  
# Plots  
##########################################  
  
  
#Comments#################################  
  
#Just how do you add comments?   
#Comments are made by using the hash tag, #, character in front of the text.   
#The hash tag ONLY comments the single line.  
  
#This is a comment line and will not be executed by R.  
#Comments are useful to describe what you are trying to do  
  
  
  
#Operators###############################  
  
#Arithmetic operators  
# + Addition  
# - Subtraction  
# \* Multiplication  
# / Division  
# ^ or \*\* exponential  
  
#Logical operators  
# > greater than  
# >= greater than or equal to  
# < less than  
# <= less than or equal to  
# == exactly equal to  
# != Not equal to  
  
#Other operators  
# <- or = assignment operators  
  
  
  
#Data Types#############################  
  
#Numeric  
#Assign the value of 20.5 to the object x  
x <- 20.5  
#print the values within the object x  
x

## [1] 20.5

#Determine the data type of object x  
class(x)

## [1] "numeric"

#Integer  
#Assign the integer value of 6 to the object y  
y <- as.integer(6)  
#print the values within the object y  
y

## [1] 6

#Determine the data type of object y  
class(y)

## [1] "integer"

#Using numeric variables   
#subtract y from x  
x-y

## [1] 14.5

#Character  
#Assign the character string "Goodson Pond"   
#to the object z  
z <- "Goodson Pond"  
#print the values within the object z  
z

## [1] "Goodson Pond"

#Determine the data type of object z  
class(z)

## [1] "character"

#You can also convert numeric or   
#integer data to characters using a   
#special function "as.character()"  
#Assign the character string "5.67" to   
#the object v  
v <- as.character(5.67)  
#print the values within the object v  
v

## [1] "5.67"

#Determine the data type of object v  
class(v)

## [1] "character"

#Creating new Variables/Objects########  
#create a new object called "temp" and   
#assign it the value of 18.2  
temp <- 18.2  
#print the value stored in the object "temp"  
temp

## [1] 18.2

#Complex data types########################  
  
#Vector  
#A vector is a sequence of data of the \*SAME\* basic data type,   
#data types can't be mixed. A single vector can contain all numeric,   
#all integer, or all character types.  
  
#create a vector of largemouth bass catch rates by lake. Each number represents   
#the total number of largemouth bass collected at a single lake. The name used   
#for the object which contains the total catch is "lmb".  
lmb <- c(12,35,66,10,11,8)  
  
#accessing elements within the lmb vector by specifying the index within   
#brackets. The following line returns the catch rate at the third lake.  
lmb[3]

## [1] 66

#create a vector of character data  
char1 <- c("one","two","four","two","five","two")  
  
char1[2]

## [1] "two"

#Matrix  
#Two-dimensional layout of data. All data   
#must be the same type in a matrix.  
  
lmb\_byrow <- matrix(c(12,16,25,33,  
 35,40,10,45,  
 66,85,33,21,  
 10,10,25,10,  
 11,3,6,18,  
 8,7,12,22), #the data elements entered one row at a time  
 nrow = 6, #number of rows  
 ncol = 4, #number of columns  
 byrow = TRUE) #fill in the matrix by ROW  
  
lmb\_byrow #print the matrix in the console

## [,1] [,2] [,3] [,4]  
## [1,] 12 16 25 33  
## [2,] 35 40 10 45  
## [3,] 66 85 33 21  
## [4,] 10 10 25 10  
## [5,] 11 3 6 18  
## [6,] 8 7 12 22

#Return the dimensions of the lmb\_byrow matrix  
dim(lmb\_byrow)

## [1] 6 4

#Elements of a matrix are accessed by specifying   
#the row then column in brackets. To access   
#the catch from the third lake in the fourth   
#year use:  
lmb\_byrow[3,4]

## [1] 21

#Access all catches from the third lake (row)   
#in the matrix, leave the column index empty  
lmb\_byrow[3,]

## [1] 66 85 33 21

#Access all catches from the fourth year (column)   
#in the matrix, leave the row index empty  
lmb\_byrow[,3]

## [1] 25 10 33 25 6 12

#This is the same data above but the data are   
#entered by column to demonstrate a different   
#way to entering data  
lmb\_bycolumn <- matrix(c(12,35,66,10,11,8,  
 16,40,85,10,3,7,  
 25,10,33,25,6,12,  
 33,45,21,10,18,22), #the data elements,one column at a time  
 nrow = 6, #number of rows  
 ncol = 4, #number of columns  
 byrow = FALSE) #fill in the matrix by COLUMN  
lmb\_bycolumn #print the matrix in the console

## [,1] [,2] [,3] [,4]  
## [1,] 12 16 25 33  
## [2,] 35 40 10 45  
## [3,] 66 85 33 21  
## [4,] 10 10 25 10  
## [5,] 11 3 6 18  
## [6,] 8 7 12 22

#Data frame  
#Another two-dimensional object but can contain different data types in   
#separate columns. For example, one column can be all numeric and   
#another column all characters  
  
#Create a data frame with a column for lake names   
#and four years of data.  
lmb\_df <- data.frame("lake" = c("A","B","C","D","E","F"),  
 "2018" = c(12,35,66,10,11,8),  
 "2019" = c(16,40,85,10,3,7),  
 "2020" = c(25,10,33,25,6,12),  
 "2021" = c(33,45,21,10,18,22))  
  
#NOTE column headers can't start with a number  
#R will automatically add an "X" to "2018"  
  
lmb\_df

## lake X2018 X2019 X2020 X2021  
## 1 A 12 16 25 33  
## 2 B 35 40 10 45  
## 3 C 66 85 33 21  
## 4 D 10 10 25 10  
## 5 E 11 3 6 18  
## 6 F 8 7 12 22

#Elements of a data frame can be accessed the same way as matrices.  
#Catch from lake C in 2018  
lmb\_df[3,2]

## [1] 66

#All catches from year 2018  
lmb\_df[,2]

## [1] 12 35 66 10 11 8

#You can also return all elements from a single column by specifying the column name  
lmb\_df$X2018

## [1] 12 35 66 10 11 8

#List  
  
#A list is a vector containing other objects. A list can contain multiple vectors,   
#multiple matrices, multiple data frames, or any combination.   
#The example below combined one vector, one matrix, and one data frame together   
#in a single list.  
  
#See explanation of code for vector, matrix, and data frame above.  
lmb <- c(12,35,66,10,11,8)  
lmb\_byrow <- matrix(c(12,16,25,33,  
 35,40,10,45,  
 66,85,33,21,  
 10,10,25,10,  
 11,3,6,18,  
 8,7,12,22),   
 nrow = 6,  
 ncol = 4,  
 byrow = TRUE)   
lmb\_df <- data.frame("lake" = c("A","B","C","D","E","F"),  
 "2018" = c(12,35,66,10,11,8),  
 "2019" = c(16,40,85,10,3,7),  
 "2020" = c(25,10,33,25,6,12),  
 "2021" = c(33,45,21,10,18,22))  
  
#Combine the three objects created above into a   
#single list  
lmb\_list <- list(lmb, lmb\_byrow, lmb\_df)  
  
#Print all elements from the list  
lmb\_list

## [[1]]  
## [1] 12 35 66 10 11 8  
##   
## [[2]]  
## [,1] [,2] [,3] [,4]  
## [1,] 12 16 25 33  
## [2,] 35 40 10 45  
## [3,] 66 85 33 21  
## [4,] 10 10 25 10  
## [5,] 11 3 6 18  
## [6,] 8 7 12 22  
##   
## [[3]]  
## lake X2018 X2019 X2020 X2021  
## 1 A 12 16 25 33  
## 2 B 35 40 10 45  
## 3 C 66 85 33 21  
## 4 D 10 10 25 10  
## 5 E 11 3 6 18  
## 6 F 8 7 12 22

#Access a member of the list using double brackets [[]] returns the matrix   
#from the lmb\_list  
lmb\_list[[2]]

## [,1] [,2] [,3] [,4]  
## [1,] 12 16 25 33  
## [2,] 35 40 10 45  
## [3,] 66 85 33 21  
## [4,] 10 10 25 10  
## [5,] 11 3 6 18  
## [6,] 8 7 12 22

#Try to return the vector and data frame from the   
#list on your own.  
  
  
#Break?  
  
  
#Functions#######################################################  
  
#A function performs a calculation or action to the data.   
#Nothing happens in R without using a function. You have already been using   
#functions in R!   
#data.frame() is a function  
  
#A function includes the function name followed by parentheses.   
#The example below will calculate the mean of a specific row in a dataframe  
  
#First, lets create a data frame with some data  
lmb\_df <- data.frame("lake" = c("A","B","C","D","E","F"),  
 "2018" = c(12,35,66,10,11,8),  
 "2019" = c(16,40,85,10,3,7),  
 "2020" = c(25,10,33,25,6,12),  
 "2021" = c(33,45,21,10,18,22))  
  
lmb\_df

## lake X2018 X2019 X2020 X2021  
## 1 A 12 16 25 33  
## 2 B 35 40 10 45  
## 3 C 66 85 33 21  
## 4 D 10 10 25 10  
## 5 E 11 3 6 18  
## 6 F 8 7 12 22

#Now calculate the average catch across all lakes in a single  
#Determine the average catch in 2018  
mean(lmb\_df[,2])

## [1] 23.66667

#Or reference the column header  
mean(lmb\_df$X2018)

## [1] 23.66667

#Determine the average catch at lake B  
#Note that you must use a different function for row mean than column means.  
#Specify the row and columns since the first columns includes lake  
rowMeans(lmb\_df[2,2:5])

## 2   
## 32.5

#You can also save the output of a function into a   
#new object  
Mean\_2018 = mean(lmb\_df[,2])  
#Print the mean from 2018 in the console  
Mean\_2018

## [1] 23.66667

#Packages#########################################################  
  
#Packages contain the functions you will use to manipulate and analyze data.  
#R comes pre-installedwith several important packages.You will eventually   
#need to install other packages for specific analysis.  
#The FSA package that we will use tomorrow does not come with R  
  
#You only have to install a package once but you will need to update it.   
#After installing a package, you have to tell R to load it before using it.  
  
#You can either use the "Install" button on the "Packages" tab,  
#or use the following code  
#install.packages("matrixStats")  
  
  
#load the matrixStats package. You will have to do this every time you open R.  
library(matrixStats)  
  
  
#Getting Help###################################################  
  
help(c)

## starting httpd help server ... done

#Or Google  
  
  
#Lunch?  
  
  
#Data exploration/summary in R#########################################  
  
#Start with a data set  
lmb\_df <- data.frame("lake" = c("A","B","C","D","E","F"),  
 "2018" = c(12,35,66,10,11,8),  
 "2019" = c(16,40,85,10,3,7),  
 "2020" = c(25,10,33,25,6,12),  
 "2021" = c(33,45,21,10,18,22))  
  
  
#Wide vs long format  
#The lmb\_df data frame is in in the "wide" format.   
#Wide format does not have values that repeat in any column.   
#Most R functions prefer data to be in the long format.  
  
#This might be a change for users of other stats programs, such as SPSS.   
#SPSS likes data to be in the wide format. Long format contains values that repeat   
#in a column, usually the first column.  
  
#There are many different ways to convert from wide to long or long to wide  
#We will use the tidyr package  
#You might have to install the tidyr package  
library(tidyr)  
  
#pivot\_longer() converts from wide to long  
#pivot\_wider() converts from long to wide  
  
#Covert from wide to long  
#Preview the data to refresh what we are looking at  
lmb\_df

## lake X2018 X2019 X2020 X2021  
## 1 A 12 16 25 33  
## 2 B 35 40 10 45  
## 3 C 66 85 33 21  
## 4 D 10 10 25 10  
## 5 E 11 3 6 18  
## 6 F 8 7 12 22

#We will introduce the magical "pipe" %>% symbol here first. The %>% is a useful operator to chain multiple   
#functions together. The basic syntax is:   
#data %>% function  
  
#The object on the left of %>% becomes the first argument of the function to the right. The first argument of a   
#function is typically the data object that you want to apply the function to. Chained %>% can be used so the   
#output of one function is the input data for a new function:  
  
# -------> -------> -------> -------> ------->  
#data %>% function1 %>% function 2 %>% function 3  
  
#pivot\_longer(cols = columns to pivot into longer,  
# names\_to = the name of the new column with groups,  
# values\_to = the name of the new column that holds the values)  
  
#Trick to change column headers to numeric without the "X"  
names(lmb\_df) = c("lake", 2018, 2019, 2020, 2021)  
  
lmb\_df %>% pivot\_longer(cols = c("2018", "2019", "2020", "2021"),  
 names\_to = "year",  
 values\_to = "cpue")

## # A tibble: 24 × 3  
## lake year cpue  
## <chr> <chr> <dbl>  
## 1 A 2018 12  
## 2 A 2019 16  
## 3 A 2020 25  
## 4 A 2021 33  
## 5 B 2018 35  
## 6 B 2019 40  
## 7 B 2020 10  
## 8 B 2021 45  
## 9 C 2018 66  
## 10 C 2019 85  
## # … with 14 more rows

#Now create a new data frame with the "long" data format  
lmb\_long <- lmb\_df %>% pivot\_longer(cols = c("2018", "2019", "2020", "2021"),  
 names\_to = "year",  
 values\_to = "cpue")  
  
  
  
#Next we will convert our long format back to wide  
#pivot\_wider(names\_from = column with the names of the groups,  
# values\_from = column with the values for each group)  
  
  
lmb\_wide <- lmb\_long %>% pivot\_wider(names\_from=year,  
 values\_from=cpue)  
  
#There are many more arguments for the pivot\_longer and pivot\_wider functions. These allow for more complex   
#conversions such as transforming data and omitting data.   
#We won't go into those details here but they do exist!  
  
  
#The next set of functions will use a data set from the FSAdata package. You might have to install the   
#FSAdata package.  
  
#Load the FSAdata package  
library(FSAdata)

## ## FSAdata v0.3.9. See ?FSAdata to find data for specific fisheries analyses.

#Load the bass data set from Florida  
BassFL <- FSAdata::BassFL  
#Check out the help file associated with it to see   
#what the columns represent  
help("BassFL")  
  
  
#Exploring the BassFL data  
  
#There are times you need to know the list of values in a column. This can be helpful when you need to filter   
#data (which we'll do later).  
  
#What are the unique species?  
unique(BassFL$species)

## [1] Suwanee Largemouth  
## Levels: Largemouth Suwanee

#what are the unique locations? Try on your own.  
  
  
#What are the minimum and maximum number caught across species, location, and year?  
range(BassFL$num)

## [1] 3 129

#What about the minimum and maximum age? Try on your own  
  
  
  
  
#Next, we will demonstrate filter, sorting, summarizing,   
#and adding new columns  
  
#Filter all records of Suwanee bass  
Suwanee <- BassFL %>%   
 dplyr::filter(species == "Suwanee")  
Suwanee

## species loc year num age  
## 1 Suwanee SantaFe 2001 129 2  
## 2 Suwanee SantaFe 2001 28 3  
## Omitted additional rows

#Note there are a few different packages that use the "filter" function. We specifically want the "filter"   
#function from the "dplyr" package. To do this we need to specify   
#the package then :: then function   
#package::function  
  
#Filter only Suwanee bass that are age 4 and older  
Suwanee\_4p <- BassFL %>%  
 dplyr::filter(species == "Suwanee" & age >= 4)  
Suwanee\_4p

## species loc year num age  
## 1 Suwanee SantaFe 2001 18 4  
## 2 Suwanee SantaFe 2001 19 5  
## 3 Suwanee SantaFe 2001 3 8  
## 4 Suwanee SantaFe 2002 37 4  
## 5 Suwanee SantaFe 2002 11 5  
## 6 Suwanee SantaFe 2002 9 6  
## 7 Suwanee Wacissa 2002 34 4  
## 8 Suwanee Wacissa 2002 14 5  
## 9 Suwanee Wacissa 2002 5 7  
## 10 Suwanee Withlacoochee 2002 9 4  
## 11 Suwanee Withlacoochee 2002 3 7

#Using the code above as a template, modify to filter   
#for all observations from SantaFe that captured more   
#than 39 fish  
  
  
#Answer###############  
Suwanee\_n40 <- BassFL %>%  
 dplyr::filter(loc == "SantaFe" & num >= 40)  
Suwanee\_n40

## species loc year num age  
## 1 Suwanee SantaFe 2001 129 2  
## 2 Suwanee SantaFe 2002 83 2  
## 3 Suwanee SantaFe 2002 73 3  
## 4 Largemouth SantaFe 2001 99 2  
## 5 Largemouth SantaFe 2002 41 2  
## 6 Largemouth SantaFe 2002 49 3

#End Answer##################  
  
#Sort##########  
#The arrange function sorts alphabetically and   
#ascending by default  
BassFL %>%  
 dplyr::arrange(species,num)

## species loc year num age  
## 1 Largemouth SantaFe 2001 3 8  
## 2 Largemouth Ochlockonee 2001 3 5  
## Omitted additional rows

#You can specify descending using desc() function  
BassFL %>%  
 dplyr::arrange(desc(species),desc(num))

## species loc year num age  
## 1 Suwanee SantaFe 2001 129 2  
## 2 Suwanee SantaFe 2002 83 2  
## 3 Omitted additional rows

#Summarizing data#################  
  
#Determine the average age by species and location  
#To accomplish this, we will chain together multiple functions using pipes.  
  
BassFL %>% #specify data set  
 dplyr::group\_by(species,loc) %>% #identify groups to summarize  
 dplyr::summarise(Mean\_age = mean (age, na.rm = TRUE))

## `summarise()` has grouped output by 'species'. You can override using the  
## `.groups` argument.

## # A tibble: 5 × 3  
## # Groups: species [2]  
## species loc Mean\_age  
## <fct> <fct> <dbl>  
## 1 Largemouth Ochlockonee 4.3  
## 2 Largemouth SantaFe 4.2  
## 3 Suwanee SantaFe 4.2  
## 4 Suwanee Wacissa 4.2  
## 5 Suwanee Withlacoochee 4

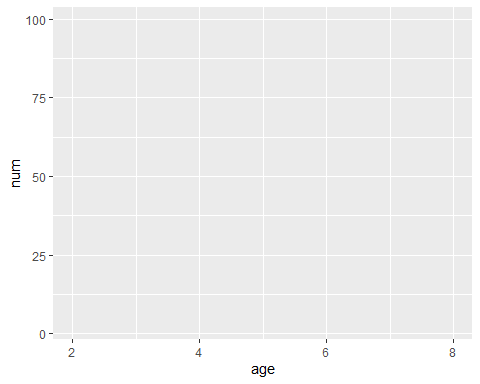
#Use what you have learned above to  
#1) filter for only Suwanee bass  
#2) group data by location  
#3) create a new column titled "Mean\_num" that will include the average catch by location  
#4) Sort the data by ascending "Mean\_num"  
  
  
#Answer###########  
BassFL %>%  
 dplyr::filter(species == "Suwanee") %>%  
 dplyr::group\_by(loc) %>%  
 dplyr::summarise(Mean\_num = mean (num, na.rm = TRUE)) %>%  
 dplyr::arrange(Mean\_num)

## # A tibble: 3 × 2  
## loc Mean\_num  
## <fct> <dbl>  
## 1 Withlacoochee 9   
## 2 Wacissa 40.4  
## 3 SantaFe 41

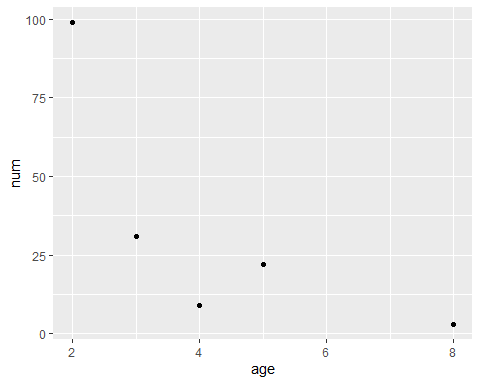
#End Answer#######  
  
  
  
#Count the number of observations per species#########  
BassFL %>%  
 dplyr::group\_by(species) %>%  
 dplyr::summarise(num\_obs = dplyr::n())

## # A tibble: 2 × 2  
## species num\_obs  
## <fct> <int>  
## 1 Largemouth 20  
## 2 Suwanee 19

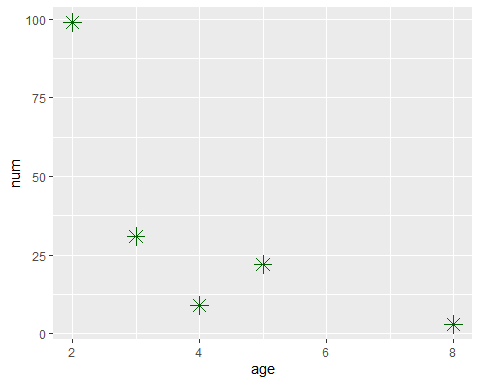
#Basic graphing in R#########################  
#Base R graphs are helpful to quickly visualize data but they are not very aesthetically pleasing.  
#We will use ggplot2 instead. Advanced graphs and ultimate flexibility can be achieved using advanced   
#ggplot2 functions. We won't cover advanced ggplot2 here - that could be an entire day!  
  
#You might have to install the ggplot2 package.   
#Install ggplot2 using the "Install" button under the   
#"Packages" tab.   
  
#load ggplot2  
library(ggplot2)  
  
#Scatterplot#########  
  
#Create a scatterplot of number at age  
#Filter the bass data to only include Largemouth from SantaFe in 2001  
LMB\_SantaFe <- BassFL %>%   
 dplyr::filter(species == "Largemouth" &   
 loc == "SantaFe" &   
 year == 2001)  
  
#ggplot2 uses "layers" to build a figure. All ggplot2   
#figures begin with specifying the data  
  
#ggplot(data, aes(x = x values, y = y values))  
  
#Produces a blank graph  
ggplot(LMB\_SantaFe, aes(x = age, y = num))



#After declaring the data layer, you can add elements to build it using the "+" symbol  
#The geom\_point function tells R to make a scatterplot of points using the specified data  
ggplot(LMB\_SantaFe, aes(x = age, y = num)) +  
 geom\_point()

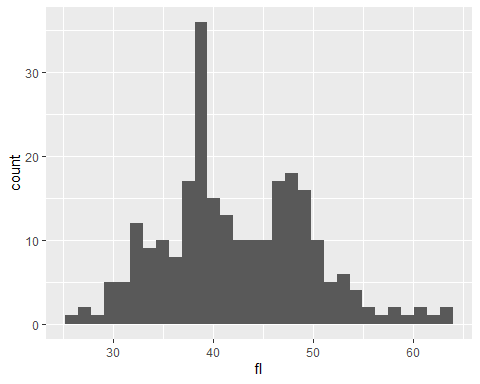


#Change the point size and color  
ggplot(LMB\_SantaFe, aes(x = age, y = num)) +  
 geom\_point(shape = 8, size = 4, color = "darkgreen")

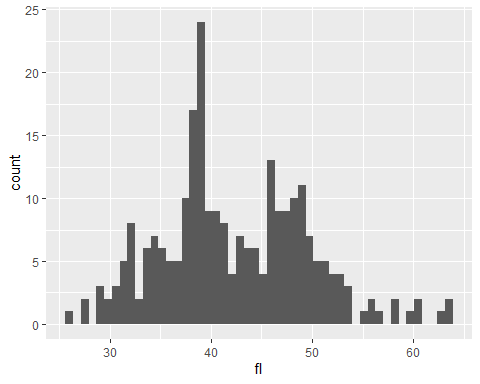


#Try different values for size, shape, and different   
#colors.  
  
  
  
#Histogram#######  
#Load the Bonito data set from the FSAdata package  
Bonito <- FSAdata::Bonito  
  
#Allow ggplot to pick bin width  
ggplot(Bonito, aes(x = fl)) +   
 geom\_histogram()

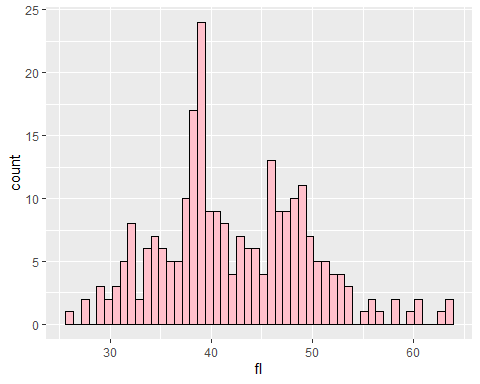
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



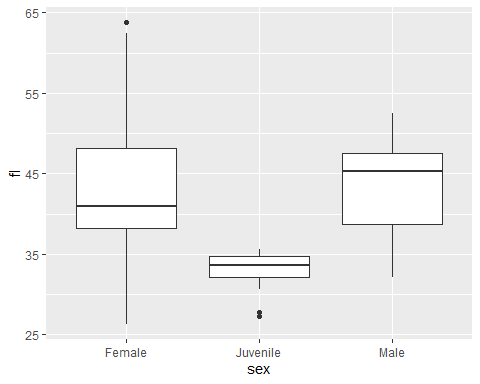
#Adjust bin width  
ggplot(Bonito, aes(x=fl)) +   
 geom\_histogram(bins = 50)



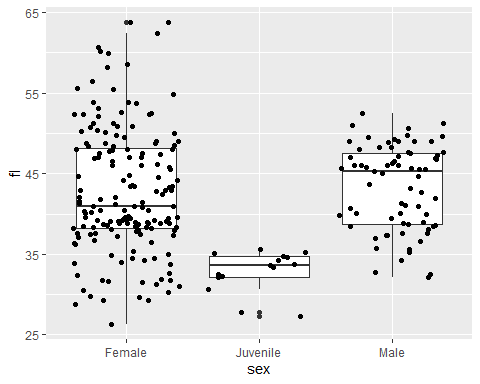
#Adjust the color of the bars  
#fill will change the fill color, color will change   
#the bar borders  
ggplot(Bonito, aes(x=fl)) +   
 geom\_histogram(bins = 50, fill = "pink", color = "black")



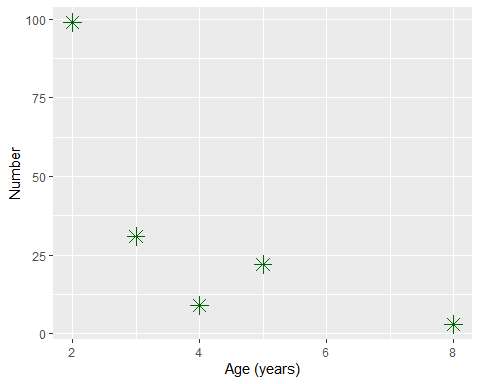
#Boxplot#############  
#Boxplot of fork length by sex  
ggplot(Bonito, aes(x = sex, y = fl)) +  
 geom\_boxplot()



#Overlay raw data that are "jittered"  
ggplot(Bonito, aes(x = sex, y = fl)) +  
 geom\_boxplot() +  
 geom\_jitter()



#Modify axes#########  
#Controlling the axes format of the scatterplot introduced earlier  
#Modify axes labels  
ggplot(LMB\_SantaFe, aes(x = age, y = num)) +  
 geom\_point(shape = 8, size = 4, color = "darkgreen") +  
 xlab("Age (years)") +   
 ylab("Number")



#Change the font size  
ggplot(LMB\_SantaFe, aes(x = age, y = num)) +  
 geom\_point(shape = 8, size = 4, color = "darkgreen") +  
 xlab("Age (years)") +   
 ylab("Number") +  
 theme(axis.text.x = element\_text(size = 20),  
 axis.text.y = element\_text(size = 20),  
 axis.title.x = element\_text(size = 24),  
 axis.title.y = element\_text(size = 24))

