Graphs in R Markdown: Advanced Tools

Course Instructor: Dr. Suborna Ahmed

# Preparing directory and data setup

Follow these steps:

Step 1: Import the data “VRI\_data.csv” and rename it as vri\_data. Step 2: Use the attach function to mask all variable names to use these names directly.

# current directory  
getwd()

## [1] "C:/Syd/UBC/Teaching/GAA FRST 232/Archivos finales"

# Importing VRI data  
vri\_data <- read.csv("VRI\_data.csv", header = TRUE)  
  
# Listing all files in the directory  
dir()

## [1] "Barplot in R Markdown.Rmd"   
## [2] "Example\_save.pdf"   
## [3] "FinalJun10"   
## [4] "Graph2.png"   
## [5] "Graphs-in-R-Markdown-Advanced-Tools.docx"  
## [6] "Graphs-in-R-Markdown-Advanced-Tools.html"  
## [7] "Graphs-in-R-Markdown-Advanced-Tools.Rmd"   
## [8] "Graphs in R Markdown Advanced Tools.Rmd"   
## [9] "Jun14finals"   
## [10] "One\_legend.png"   
## [11] "pie\_hist\_boxplot.Rmd"   
## [12] "Time tracking.xlsx"   
## [13] "VRI\_data.csv"   
## [14] "VRI\_data.RData"

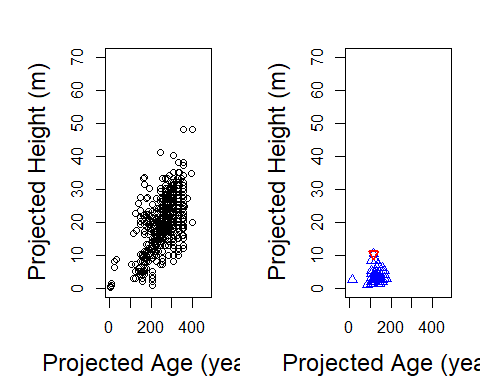
attach(vri\_data)

# Side by side graphs

From the vri data, we want to compare the height trend of two species by looking at their graphs. For “PLC” species show the maximum height on the graph.

Step 1: Create a separate data set for each species Step 2: Specify the margin to add space for two graphs side-by-side Step 3: Plot these two graphs Step 4: Select the maximum height from the “PLC” species and show it on the graph.

# separating data by species  
d1<- subset(vri\_data, (SPECIES\_CD\_1 =='CW'))  
d2<- subset(vri\_data, (SPECIES\_CD\_1 =='PLC'))  
  
  
# controlling margins: specifying the margin using mai  
  
par(mfrow=c(1,2),mai=c(0.9,1.1,0.5,0.3), cex=1.0 ) # mai is the margin  
  
ylabel="Projected Height (m)"  
xlabel="Projected Age (years)"  
ylm=c(0,70)  
xlm=c(0,470)  
  
cxlb=1.5 # axis level - fontsize  
cxaxis=1 # axis fontsize  
maincx=2.5 # title font  
  
age=3 # age column number  
height=4 # height column number  
ldcex=1 # size of points  
ld=2 # line width of points, we are making it thicker to emphasis on this maximum height value   
maxd1<- max(d2[,height])  
m1<-subset(d2, PROJ\_HEIGHT\_1==maxd1)  
  
  
# graph 1  
plot(d1[,age],d1[,height], type = "p", pch = 1, col = "black", main=" ", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab=ylabel)  
  
# graph 2  
plot(d2[,age],d2[,height], type = "p", pch = 2, col = "blue", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab=ylabel)  
points(m1[,age],m1[,height], type = "p", pch = 6, col = "red", lwd=ld, cex=ldcex)



# Save graphics as a permanent file

The side-by-side graphs you created above - we want to save it as an image in the folder where you kept this.RMD file. This is called “saving as a permanent” graph/file/image. We can save plots as permanent files in different formats in R. To save a plot as an image use the png() function and specify the name of the file in quotes (file=“name\_of\_new\_file”), the with and height of the graph (with= and height=), units (units=), and the resolution wanted for the plot (res=). Then use the par() function to define the number of subplots and margins. Finally, write the code for each of the graphs that you want to save and run all the chunks of code together. This will automatically create a png file with your graphs in your working directory.

png("Graph2.png", width=15, height=10, units = 'in', res=1200)  
par(mfrow=c(1,2),mai=c(1.1,0.9,0.8,0.3), cex=1.0, xpd=NA) # mai is the margin  
# graph 1  
plot(d1[,age],d1[,height], type = "p", pch = 1, col = "black", main=" ", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab=ylabel)  
  
# graph 2  
plot(d2[,age],d2[,height], type = "p", pch = 2, col = "blue", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab=ylabel)  
points(m1[,age],m1[,height], type = "p", pch = 6, col = "red", lwd=ld, cex=ldcex)  
  
graphics.off()

# Another example to save a graph as a permanent file

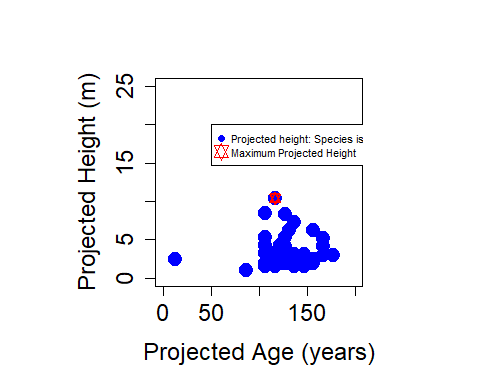
You can also save your plots as pdf files by using the pdf() function. To save multiple graphs use the par() function and define the order of your plots by using the function plot(). This will order the graphs from the top left to top right, bottom left, and bottom right in case of using four subplots. Use mtext() to add subtitles to each of your plots if desired.

# Creating Data:  
x <- seq(0,3, length=100) # vector with 100 values from 0 to 3  
  
# create three functions of x  
y1 <- x^2+1  
y2 <- -x^2+9  
y3 <- sqrt(x)  
  
# saving as a pdf file  
pdf(file="Example\_save.pdf")  
par(mfrow = c(2, 2), # 2 x 2 pictures on one plot  
 mar = c(4,4,2,2), # gives # of lines of margin on four sides of plot  
 # c(bottom, left, top, right)  
 pty = "s", # square plotting region,  
 bg="lightyellow") # set background color  
  
# top left:  
plot(x,y1)  
mtext("Function 1", side = 3, line = 0.2) # add title for top left panel  
  
# top right:  
plot(x,y2)  
mtext("Function 2", side = 3, line = 0.2)  
  
# bottom left:  
plot(x,y3)  
mtext("Function 3", side = 3, line = 0.2)   
  
# bottom right:  
plot(x,y1, axes=FALSE,  
 ylim=c(0,10), xlim=c(0,3),  
 xlab = "", ylab = "", type="n")  
lines(x,y1, lty=2)  
lines(x,y2, lty=1, lwd=2)  
lines(x,y3, lty=4, lwd=3, col=2)  
  
axis(1) # x-axis  
axis(2) # y-axis  
  
mtext("x", side = 1, line = 3)  
mtext("y", side = 2, line = 3)  
  
box() # add frame around graph  
  
legend(0,6.5, c("function 1"," function 2","function 3"),  
 lty=c(2,1,4), lwd=c(1,2,3), col=c(1,1,2), cex = 0.9)  
  
mtext("All three functions", side = 3, line = 0.2)  
  
  
# You can also use:  
#dev.off()   
graphics.off()  
  
  
  
# other examples are:  
# postscript(file="filename.eps") Adobe PostScript file  
# jpeg(file="filename.jpeg") JPEG bitmap file

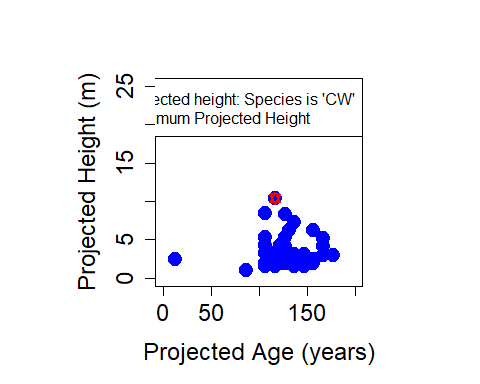
# Add legends

We can add legends to our plot by using the legend() function. In this function, we must first define the coordinates where we want to place our legend in the graph. We can do this by defining the specific coordinates in the x and y axes where we want to place the legend or by writing “toprigth” or “topleft” (e.g. legend(0, 3.1) or legend(“toprigth”)). Then, we define the name of items in the legend by using c() (e. g. legend(“toprigth”, c(“item1”, “item2”))). Finally, we can add more details to the graph by changing the colour or plot symbols and size. Make sure that the number of colors and symbols matches the number of items you want to show in your legend.

d1<- subset(vri\_data, (SPECIES\_CD\_1 =='CW'))  
d2<- subset(vri\_data, (SPECIES\_CD\_1 =='PLC'))  
  
ylabel="Projected Height (m)"  
xlabel="Projected Age (years)"  
ylm=c(0,25)  
xlm=c(0,200)  
  
cxlb=1.5 # axis level - fontsize  
cxaxis=1.5 # axis fontsize  
maincx=1.5 # title font  
  
age=3 # age column number  
height=4 # height column number  
ldcex=3 # size ofpoints  
ld=2  
maxd1<- max(d2[,height])  
m1<-subset(d2, PROJ\_HEIGHT\_1==maxd1)  
par(mfrow=c(1,1), pty="s")  
# graph 1  
plot(d2[,age],d2[,height], type = "p", pch = 20, col = "blue", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab=ylabel)  
points(m1[,age],m1[,height], type = "p", pch =11, col = "red", lwd=ld)  
  
# Adding legend   
legend(50, 20, c("Projected height: Species is 'CW'", "Maximum Projected Height"), col = c("blue", "red"), pch = c(20, 11), cex=.7, pt.cex=1.5)



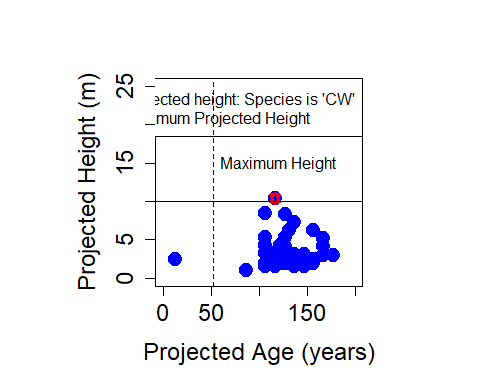
# Legend location: another procedure  
  
par(mfrow=c(1,1), pty="s") # mai is the margin  
  
# graph 1  
plot(d2[,age],d2[,height], type = "p", pch = 20, col = "blue", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab=ylabel)  
points(m1[,age],m1[,height], type = "p", pch =11, col = "red", lwd=ld)  
legend("topright", c("Projected height: Species is 'CW'", "Maximum Projected Height"), col = c("blue", "red"), pch = c(20, 11), cex=1, pt.cex=ld)



# Adding more elements: Extra lines and texts on a graph

To add extra lines to our plots we can use the abline() function. In this function we must state the type of line that we want, vertical or horizontal, and the coordinates where we want the line in the plot (abline(v=coordinate\_in\_y\_axis) or abline(h(coordinate\_in\_x\_axis))). Line types can be specified using lty(), use 0=blank lines, 1=solid lines(default), 2=dashed lines, 3=dotted lines, 4=dotdash lines, 5=longdash lines, and 6=twodash lines. Finally, add text to the graph using text(), specify first the coordinates where you want the text, and then write what you want to show on the graph.

par(mfrow=c(1,1), pty="s") # mai is the margin  
  
# graph 1  
plot(d2[,age],d2[,height], type = "p", pch = 20, col = "blue", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab=ylabel)  
points(m1[,age],m1[,height], type = "p", pch =11, col = "red", lwd=ld)  
legend("topright", c("Projected height: Species is 'CW'", "Maximum Projected Height"), col = c("blue", "red"), pch = c(20, 11), cex=1, pt.cex=ld)  
# additional lines  
# add a vertical line at x=2  
abline(v=52, lty=2)  
  
# add a horizontal line at y=0  
abline(h=10)  
  
# abline(a,b) draws a line of slope b and intercept a  
  
  
# add text  
text(120,15,"Maximum Height")



# Add a mathematical expression

To add mathematical expressions use the expression() function. Define the mathematical expressions on label axes between quotes and tildes ( ~ ) (e.g. expression( “Total volume(“~ m^3 ~”)”)).

par(mfrow=c(1,1),mai=c(0.9,1.1,0.8,0.3), cex=1.0 ) # mai is the margin  
  
ylabel="Projected Volume(m3)"  
xlabel="Projected Age (years)"  
ylm=c(0,500)  
xlm=c(0,470)  
  
cxlb=2 # axis level - fontsize  
cxaxis=2.5 # axis fontsize  
maincx=2.5 # title font  
  
age=3 # age column number  
height=4 # height column number  
ldcex=2 # size ofpoints  
ld=2  
cx=2  
ft=4  
# graph 1  
plot(d1[,age],d1[,height], type = "p", pch = 1, col = "black", main=" ", ylim=ylm, xlim=xlm, cex=ldcex, cex.lab=cxlb, cex.axis=cxaxis, xlab=xlabel, ylab="")  
mtext(expression("Total volume ("~m^3~")"), side=2, line=3, las=0, cex=cx, font =ft)

