---
title: "Bio 381 Presentation RMD"
author: "Diana Hackenburg"
date: "May 1, 2018"
output: html_document
runtime: shiny

This presentation will cover two main R packages: `timevis` and `shiny`. Timevis lets you create fully interactive timeline visualizations. Timelines can be included in Shiny apps and R markdown documents, or viewed from the R console and RStudio Viewer. Shiny can be used to build interactive web apps in R that can act as standalone apps,be embedded in R Markdown documents, or serve as dashboards.

Resources for this presentation include:

* the [timevis github repository](https://github.com/daattali/timevis)
* the [timevis demo page](http://daattali.com/shiny/timevis-demo/)
* the [official Timeline documentation page](http://visjs.org/docs/timeline/#Configuration_Options)
* the [shiny tutorial page](https://shiny.rstudio.com/tutorial/)

```{r}
#Timevis and Shiny Packages
#2 May 2018
#DMH

#timevis allows you to create interactive timeline visualizations in R
#shiny allows you to create interactive web apps in R

#----------------------------------------
#Preliminaries
#Install timevis
##install.packages("devtools")
##devtools::install_github("daattali/timevis")
#----------------------------------------
##Load timevis library
library(timevis)

#Look at example timelines: http://daattali.com/shiny/timevis-demo/
```

##Let's create our first timeline
```{r}
#Create a simple data frame with events from this class

#Every item must have a `content` and a `start` variable
#id is optional but recommended
#start has a specific formatting of year - 2-digit month - 2-digit date, if time use hour : minute : second using 24-hour clock

simpleTL <- data.frame(
  id = 1:4,
  #include 2 lectures, a homework, and the first day of presentations
  content = c("Randomization Tests", "ggPlots",
              "Homework 12", "Presentations 1"),
  start = c("2018-04-03", "2018-04-05", "2018-04-11",
            "2018-04-24 14:50:00"),
# single point in time? Use NA
  end = c(NA, "2018-04-17",NA,NA)
)

#print data frame
simpleTL

#create first timeline
timevis(simpleTL)
```

##Building onto a timeline: Groups, add items, hyperlinks, options, and styles
```{r}
#-----------------------------------------------
#Add groups

#Create a data frame to define your groups
#Group id = used in data frame with timeline items
#Group content = what will show on your timeline

groups <- data.frame(
  id = c("lec", "hw", "pt"),
  content = c("Lecture", "Homework", "Presentation")
)

#Bind groups to data frame
groupTL <- cbind(simpleTL,group=c(rep("lec",2),"hw","pt"))

#Add groups to final timeline
timevis(groupTL,groups=groups)

#Add item to timeline
#html is supported
#Use "piping notation" to add onto the next command like with leaflet package!

timevis(groupTL,groups=groups)  %>%
  addItem(list(id=5, content="<b>Presentations 2</b>", start="2018-04-25", group="pt"))

#Add a hyperlink to a timeline item
#Look at structure - variables are factors, need them as characters
str(groupTL)

#First change content from a factor to a character
groupTL$content <- as.character(groupTL$content)

#Then call the specific content cell
groupTL[3,2] <- "<a href='https://gotellilab.github.io/Bio381/Homeworks/Homework12_2018.html'>Homework 12</a>"
```

```
# You can re-factorize with the as.factor function or leave as characters
groupTL$content <- as.factor(groupTL$content)
timevis(groupTL,groups=groups)

#-----------------------------------------------
#Your timeline has options!
#showZoom = TRUE/FALSE
#Some things are options and must be included as a list
#Make items editable
#Timeline automatically resizes to window but can set height or width

timevis(simpleTL, showZoom=FALSE,options = list(editable = TRUE, width="500px",height = "400px"))

#Add some style
styles <- c("color:white; background-color:black;")
#Styles associated with items or with groups
simpleTL <- cbind(simpleTL,style=styles)
simpleTL
timevis(simpleTL)
```

##Create a timeline with prepared data
```{r}
#-------------------------------------
#Timeline with prepared data

#Read in data
timedata <- read.csv("timeline_example.csv", header=TRUE,sep=",")

#Create the groups data frame
groups <- data.frame(
  id=c("N","SP","NP","LP","TMDL"),
  content=c("News","State Policy","National Policy","Local Policy","TMDL"),
  #add style to group
  style=c("background-color:lightblue;","background-color:plum;","background-color:pink;","background-color:khaki;","background-
color:coral;"))

#Assign timeline to a vector
#shortcut assignment operator "alt" + "-" <- (Mac "Opt" + "-")
#orientation of timeline axis
timevisHAB <-
timevis(timedata,groups=groups,options=list(selectable=TRUE,editable=TRUE,verticalScroll=TRUE,horizontalScroll=TRUE,moveable=TRUE,multisele

timevisHAB


#Select an item by clicking it, and use ctrl+click to or shift+click to select multiple items (when multiselect: true).
#Move selected items by dragging them.
#Create a new item by double tapping on an empty space
#Create a new range item by dragging on an empty space with the ctrl key down.
#Update an item by double tapping it.
#Delete a selected item by clicking the delete button on the top right.
#you can change the editability of each item by adding to it in the data frame editable:false

#How does the timeline look in shiny?
#https://dhackenburg.shinyapps.io/shinyHABtimeline/
```

##Create a shiny app
```{r}
#This example derived from https://shiny.rstudio.com/articles/build.html

#Shiny apps are contained in a single script called app.R. The script app.R lives in a directory (for example, newdir/) and the app can
be run with runApp("newdir")

#Save new R document as app.R in its own folder (exampleapp)

#ShinyApp Example
#2 May 2018
#DMH

#app.R has three components:
#ui: a user interface object
#server: a server function
#a call to the shinyApp function

#-------------------------------
#Preliminaries
library(shiny)
library(RColorBrewer)
library(shinythemes)
library(ggplot2)
#-------------------------------
#We are going to use the mpg dataset to create an interactive boxplot that looks at city or highway mpg based on another variable in the
dataset

#Load mpg dataset
d<-mpg
#str(d)

#First build our user interface
#controls the layout and appearance of your app
#FluidPage creates a display that automatically adjusts to the dimensions of your user's browser window
#You can also choose fixedPage, navbarPage, or fluidRow or Column
ui <- fluidPage(

  #choose a shiny theme (united, darkly, cosmo)
  theme = shinytheme("united"),

  #give your app a title
  titlePanel("Miles Per Gallon"),

  #create a sidebar for inputs
  sidebarLayout(position = "right",
```

```
    sidebarPanel(
      #Give sidebarPanel a name
      h3("Choose Your Inputs"),

      #Create widgets for choosing inputs
      selectInput("mpgtype","MPG Standard",c("Highway"="hwy","City"="city")),
      selectInput("variable","Variable:",c("Manufacturer"="manufacturer","Year"="year","Fuel Type"="fl","Class"="class"))),

    #Main panel for displaying outputs
    mainPanel(
      h3(textOutput("caption"),align="center"),
      plotOutput("mpgPlot")
    )
  )
)
#server  contains the instructions that your computer needs to build your app
server <- function(input,output){
  #A reactive expression uses widget input to return a value and updates value whenever widget changes
  formulaText <- reactive({
    if (input$mpgtype=="hwy")
      paste("hwy~",input$variable)
    else
      paste("cty~",input$variable)
  })
  #A render expression acts as a function
  output$caption <- renderText({formulaText()})
  output$mpgPlot <- renderPlot({
    par(mar=c(6,6,0,0))
    boxplot(as.formula(formulaText()), data=d,las=2,col=brewer.pal(n=8,name="Set2"),pch=19,xlab="",ylab="")
    mtext(input$variable,side=1,line=5,font=2)
    mtext(c("(mpg)",input$mpgtype),side=2,line=3:4,font=2)
})
}


shinyApp(ui,server)

#Your R session will be busy while the Hello Shiny app is active, so you will not be able to run any R commands. R is monitoring the app
and executing the app's reactions

#Can also run using:
#library(shiny)
#runApp("exampleapp")
```