Data 650 Midterm solution

Ok before we get to the midterm solution, R studio has an update so that you can access the terminal directly in R studio. If you download the latest version (dated 2017-10-9). Open R-Studio, go to tools & in the drop down “Terminal” will now be a choice. Once you do this, you’ll need to re-install all your packages. (dplyr, flexdashboard,ggplot2, ggiraph, stringr, lubridate, devtools, blogdown, Hugo, etc)

Now for the midterm:

https://bendazz.github.io/grad-garbanzo/

The Midterm had 3 “problems”

The Set-Up:

Download the Kaggle data set US Candy Production by Month From January 1972 to August 2017. The data is daily United States candy production as a percentage of 2012 production. In order to download the set you must sign into Kaggle.

Create a working folder on the desktop called midterm

Load the libraries needed: flexdashboard, dplyr, ggplot2, stringr, and lubridate.

Open a new Rmarkdown file after you click on new R markdown choose “from template” and then choose the “flexdashboard” template.

Read the data into R:

production<-read.csv("candy\_production.csv",header=TRUE,stringsAsFactors=FALSE)

Take a look at the csv file there are only 2 columns (you can either open the excel file or type “head(production)” into the command line in R studio & it will show the first few lines. If we look at the csv file, we see that the dates are in M-D-Y format. All three of our graphs will use the dates, but we want the year to be first Y-M-D format. So we are going to change the download file & put the dates in that format

production$observation\_date<-ymd(production$observation\_date)

if we type ‘production$observation\_date’ into the command line it will return the formatted dates.

**Problem 1 Line graph**

Look at the first plot in the dashboard https://bendazz.github.io/grad-sol. It is a time series plot indicating the candy production for every day in the data set. Reproduce this plot.

We don't need to create a new data frame as we have everything we need in what we've done so far. We just need to create our line graph:

ggplot()+

geom\_line(data=production, aes(x=observation\_date,y=IPG3113N))

Now onto the next problem.

**Problem 2 Line Graph**

Look at the second plot in the dashboard https://bendazz.github.io/grad-sol. It is a time series plot for only the days in 1990. Reproduce this plot, being sure to have each month labeled as shown.

Our hint for this problem was Hint: “%b”

So for this graph, we want to define our data frame as our data.

df<-production

However we don’t want all the data, we only want the data for 1990. We want to filter. We can do this on either one line or two:

df<-df%>%

filter(observation\_date<='1990-12-31')%>%

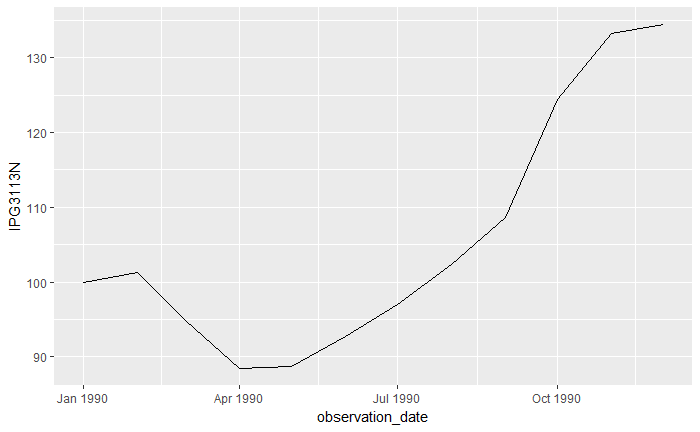
filter(observation\_date>='1990-01-01')

We can get our graph with one line of code:

ggplot()+

geom\_line(data=df, aes(x=observation\_date, y=IPG3113N))

But the graph defaults to 3 month increments.



We want the graph to show one month increments. We need to look at the “strftime” formats

If we google search on “strftime” we find this site: <http://man7.org/linux/man-pages/man3/strftime.3.html> whch gives us the code for various time parameters. We want month, so we need ‘%b’ (which was the hint).

So our whole graph is:

ggplot()+

geom\_line(data=df, aes(x=observation\_date, y=IPG3113N))+

scale\_x\_date(date\_breaks= '1 month', date\_labels ='%b')

So we are down to out last problem.

**Problem 3**

Look at the third plot in the dashboard https://bendazz.github.io/grad-sol. It is a bar plot comparing the average candy production for the 1980’s, 1990’s, and 2000’s (2000-2009). Reproduce this plot as it is shown. We received 3 hints for this plot.

So for this graph, we need to be able to get the data into groups by decade. Our first hint:

Hint 1:

str\_sub("1980-01-01",3,3)

## [1] "8"

If we type ‘str\_sub("1980-01-01",3,3)’ into the command line it returns the ‘[1] "8".’

The “3,3” means everything after the 3rd character, ending with the 3rd character.

(If we wanted to group by year, we would use 3,5 which would give us the last 2 digits of the year.)

So it is reading the ‘8’ in 1980. We can use this to be able to group our dates.

We need to set up our data frame as our database. And then modify the observation date variable for the

df<-production

df$observation\_date<-str\_sub(df$observation\_date,3,3)

If we type df$observation\_date into the command line, we see only 7,8,9,0 & 1’s for each line of data.

Now we need to filter out the data we don’t need. We have data from 1972 to 2017; but we only want data from 1980 to 2009:

df<-df%>%

filter(observation\_date != 1)%>%

filter(observation\_date != 7)

We have the decade digit, in order to make it look like the two digit decade dates we are used to seeing, we have to put a zero on the end. This is where our second hint comes in:

Hint2:

paste("8","0's",sep='')

## [1] "80's"

If we type ‘paste("8","0's",sep='')’ into the command line it returns ‘[1] "80's"’

So we want to “paste” the zero’s onto our decade numbers.

df$observation\_date<-paste(df$observation\_date,"0's", sep='')

Now, if we type df$observation\_date into the command line, we see only 80’s, 90’s &, 00’s for each line of data (since we got rid of the 70’s & 10’s).

We can now group by the observation date (decade) & sum the data

group\_by(observation\_date)%>%

summarize(ave=mean(IPG3113N))

Our total code looks like:

df<-df%>%

filter(observation\_date !='7')%>%

filter(observation\_date !='1')%>%

group\_by(observation\_date)%>%

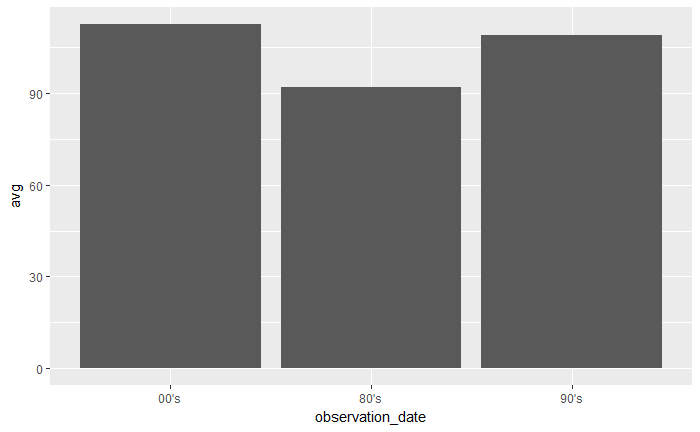
summarize(ave=mean(IPG3113N))

df$observation\_date<-paste(df$observation\_date,"0's", sep='')

ggplot()+

geom\_bar(data=df,aes(x=observation\_date,y=avg),stat='identity')

So we have a bar graph, BUT the 2000’s is showing BEFORE the 80’s (because it is sorting by number). We’ll need to tell it how to order the data.



So this is where hint 3 is useful:

Hint3:

standing<-c("junior","freshman","sophomore")

standing<-factor(standing,levels=c("freshman","sophomore","junior"))

levels(standing)

## [1] "freshman" "sophomore" "junior"

(side note: in class I said I didn’t remember this -well that’s true because I never got past the first graph on the test-I spent the whole time trying to get it to read the dates for the first graph)

So in our hit, the variable ‘standing’ refers to class (‘c’) those classes are: “junior","freshman", & "sophomore". If we type in ‘standing<-c("junior","freshman","sophomore")’ into the command line it returns: ‘[1] "junior" "freshman" "sophomore"’

We can make the variable ‘standing’ a factor and assign levels (and the order of the levels)

If we type in the second & then third lines of the hint into the command line:

‘standing<-factor(standing,levels=c("freshman","sophomore","junior"))’

‘levels(standing)’

It returns: ‘[1] "freshman" "sophomore" "junior"’ for the “levels.”

So we have told it the order to use.

Now we need to add to the factor & level before our graph:

df$observation\_date=factor(df$observation\_date,levels=c("80's","90's","00's"))

So now our whole code is:

df<-production

df$observation\_date<-str\_sub(df$observation\_date,3,3)

df<-df%>%

filter(observation\_date != 1 & observation\_date != 7)%>%

group\_by(observation\_date)%>%

summarize(avg=mean(IPG3113N))

df$observation\_date<-paste(df$observation\_date,"0's",sep='')

df$observation\_date=factor(df$observation\_date,levels=c("80's","90's","00's"))

ggplot()+

geom\_bar(data=df,aes(x=observation\_date,y=avg),stat='identity')

So the final graph has the decades in the correct order.

In order to make the flex dashboard into a website (so it can be viewed):