Week 4

```
knitr::opts_chunk$set(echo = TRUE, error = TRUE)
```

1 9/23 - Data Tables/Frames

An R data.frame is the dedicated object to work with “tables”,

-non atomic (gets to keep their own data type)

```
dat = data.frame(name = c('Leia', 'Luke', "Han"),
                 height = c(150, 175, 185),
                 force = c(TRUE, TRUE, FALSE))
dat # under the names of the columns list what kind of vectors a
```

```
  name height force
1 Leia    150  TRUE
2 Luke    175  TRUE
3  Han    185 FALSE
```

1.0.1 Data frames stored?

R internally stores as lists, manipulate like a list.

2D objects: rows and columns

1.1 Basic Manipulation of data frames

- data[ ]
- data[[ ]]
- data$column_name

```
#as a list
dat
```

```
  name height force
1 Leia    150  TRUE
2 Luke    175  TRUE
3  Han    185 FALSE
```

```
dat[1] #first column, as a list
```

```
  name
1 Leia
2 Luke
3  Han
```

```
dat[[1]] #first column's elements; can use the name of column t
```

```
[1] "Leia" "Luke" "Han"
```

```
dat$height #similar to [[2]], but uses name of column
```

```
[1] 150 175 185
```

1.1.1 Manipulating data like matrix

- data[row #, col #]

```
# cell 1, 1
dat[1,1] #1st row, 1st column
```

```
[1] "Leia"
```

```
dat[2,3] #2nd row, 3rd column
```

```
[1] TRUE
```

```
dat[,2] #all rows, only second column
```

```
[1] 150 175 185
```

```
dat[,c(2,3,1)] #reshuffle columns
```

```
  height force name
1    150  TRUE Leia
2    175  TRUE Luke
3    185 FALSE  Han
```

```
dat[c(2,3,1),] #reshuffle rows (how they print)
```

```
  name height force
2 Luke    175  TRUE
3  Han    185 FALSE
1 Leia    150  TRUE
```

1.2 Libraries and DFs

```
library(tidyverse)
```

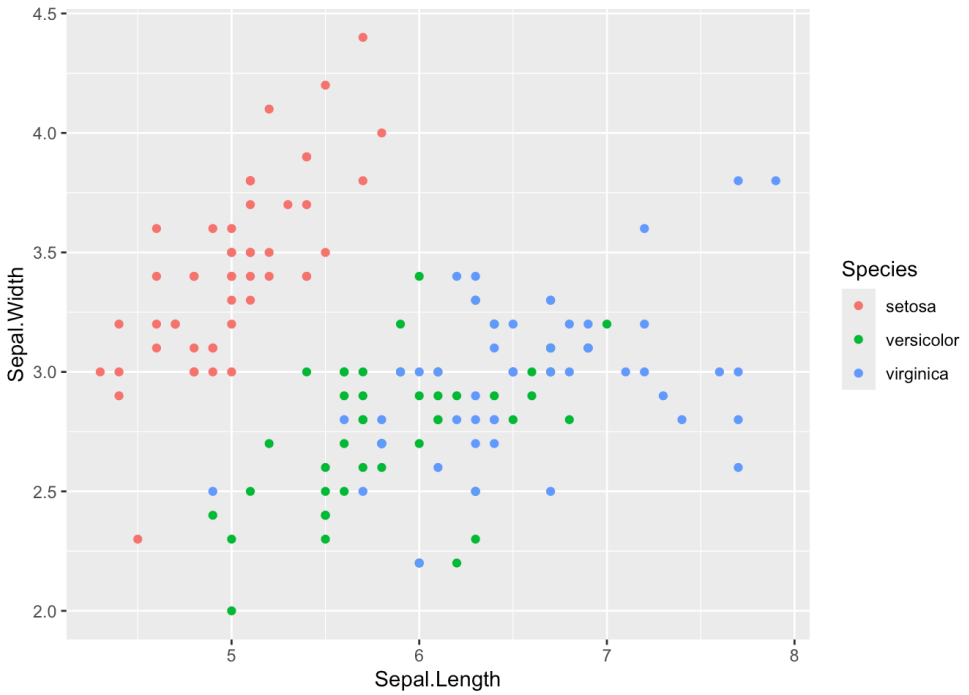
```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
head(iris) # head() shows first few rows/columns of the data tal
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

1.2.1 Scatter Plot - ggplot2

```
#layered
ggplot(data = iris,
       mapping = aes(x = Sepal.Length, y = Sepal.Width)) + #can
  geom_point(aes(color = Species)) #mapping at the local/layere
```

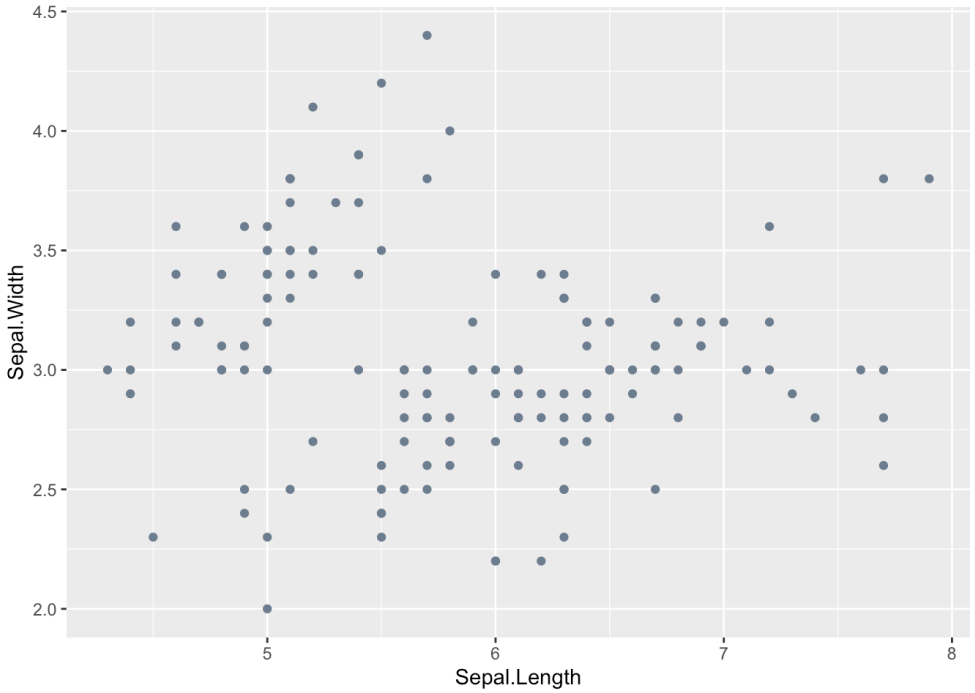



```
# 'mapping' is needed for species because it takes info from the
```

1.2.2 Mapping vs Settings

a setting is when you set (i.e. fix) a visual attribute to a constant variable

```
ggplot(data = iris,
       mapping = aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(color = 'slategrey') #setting the whole points to
```




```
#avoid putting the setting variable in aes( ) function. R would
```

```
ggplot(data = iris,
       mapping = aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(color = 'slategrey') +
  geom_smooth() +
  labs(title =  'Iris',
       xlab = ,
       ylab = )
```

```
Error in `dots_list()`:
! Argument 1 can't be empty.
```

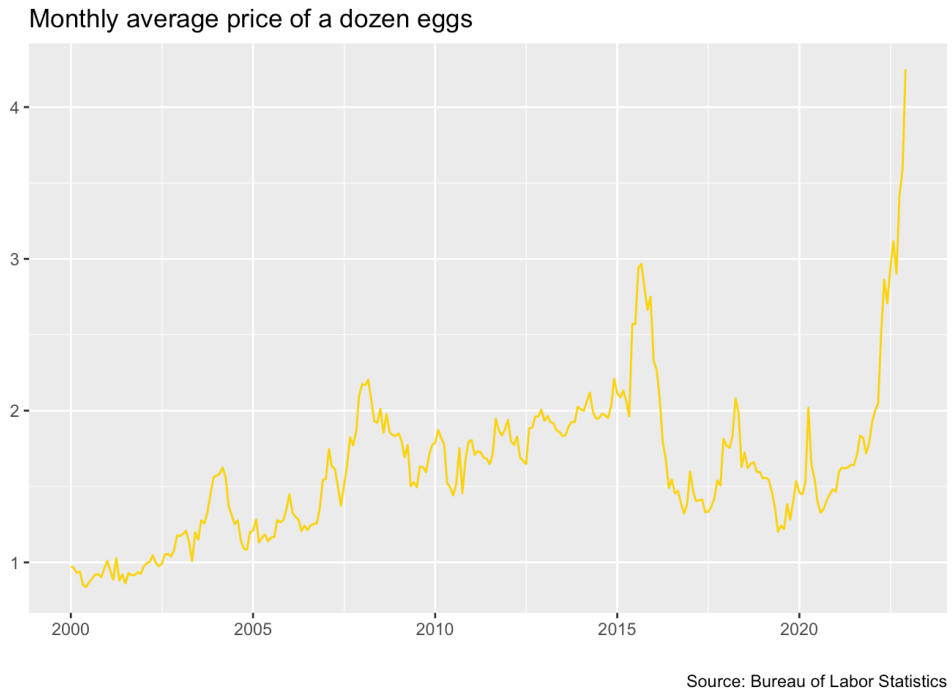
2 9/25 - Expanding on ggplot

import csv files

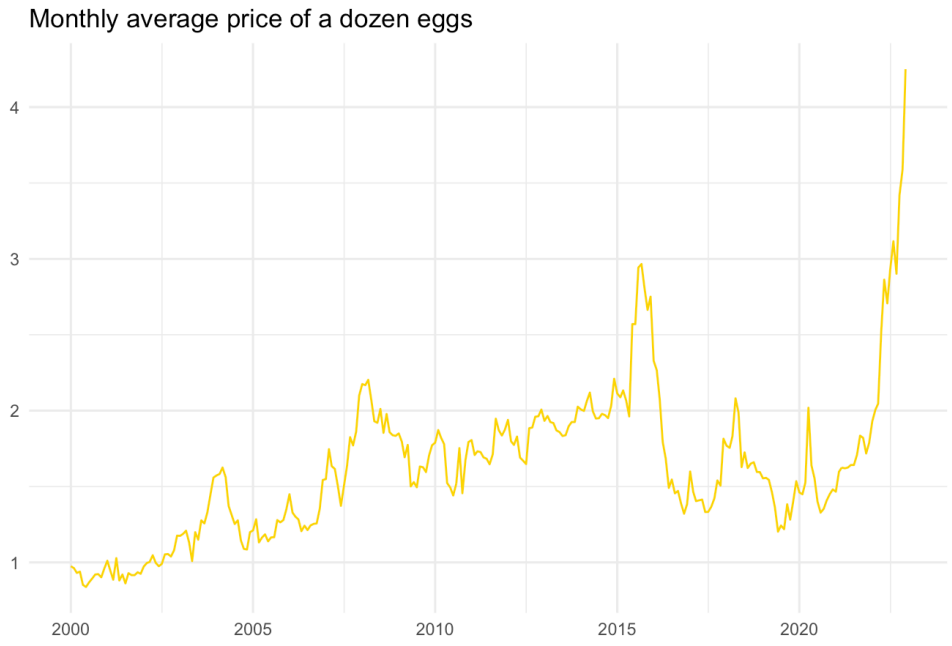
```
url = "~/Desktop/stat 133/data/price-of-eggs.csv"
dat = read_csv(file = url, col_types = 'dDdc')
```

```
# assigning graphic to "gg" object
gg = ggplot(data = dat, mapping = aes(x = Date, y = Price)) +
  geom_line(color = 'gold')
```

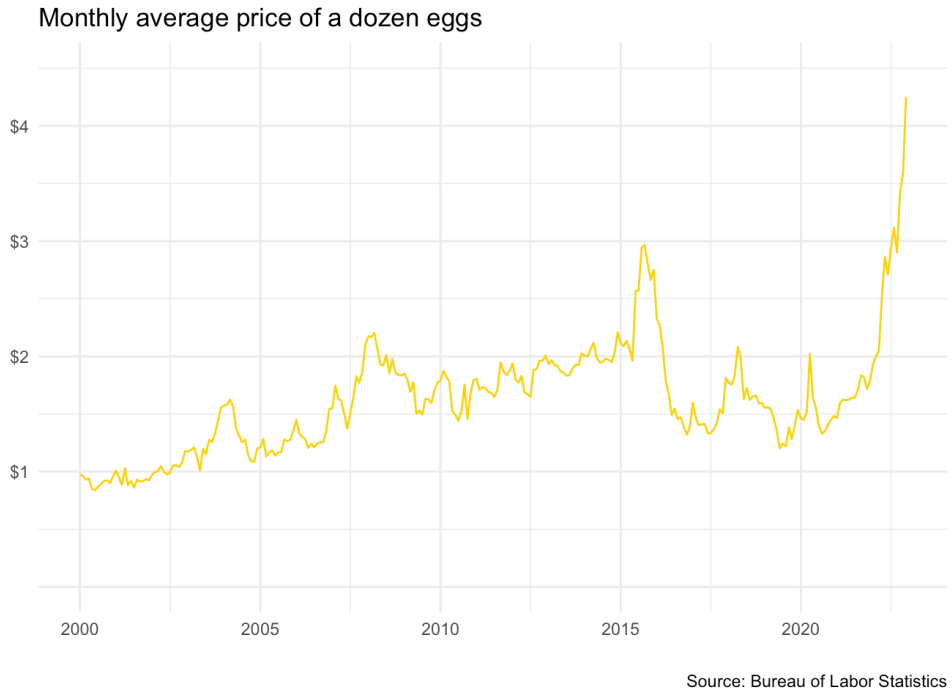
```
# can build off of assigned gg name
gg2 = gg + labs(title = "Monthly average price of a dozen eggs",
       x = "", y = " ",
       caption = "Source: Bureau of Labor Statistics")
gg2
```



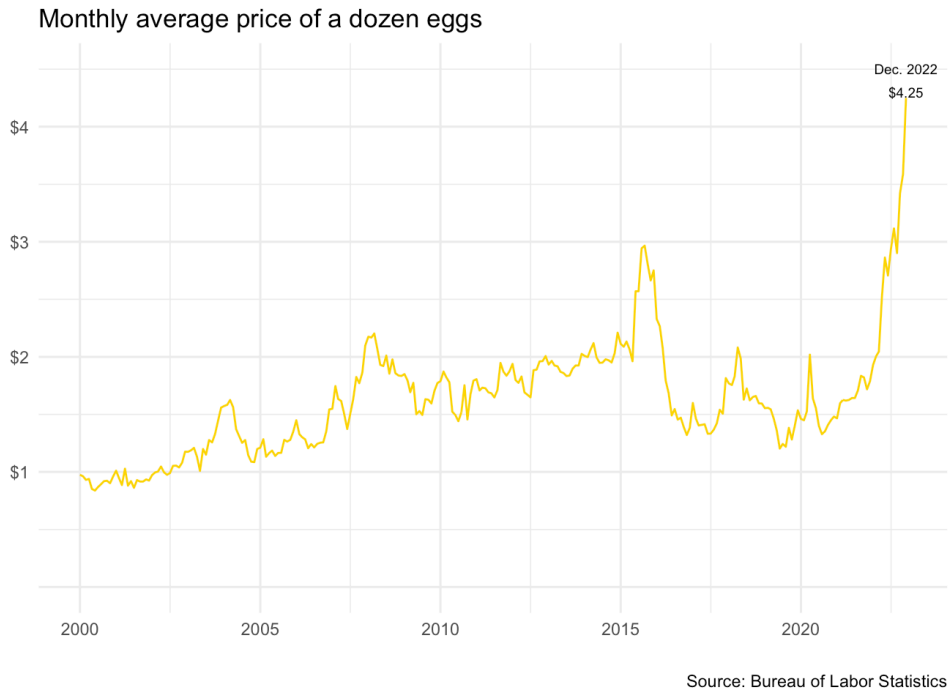

```
# theme changes
gg3 = gg2 + theme_minimal()
gg3
```




```
# x and y visual manipulation
gg4 = gg3 + scale_y_continuous(limits = c(0, 4.5),
  labels = c("", "$1", "$2", "$3", "$4"),
  breaks = c(0, 1,2,3,4))

gg4
```



```
# adding text
gg5 = gg4 + annotate(geom = "text",
                x = as.Date("2022-12-01"),
                y = 4.5,
                label = "Dec. 2022",
                size = 2.5) +
  annotate(geom = "text",
                x = as.Date("2022-12-01"),
                y = 4.3,
                label = "$4.25",
                size = 2.5)
gg5
```

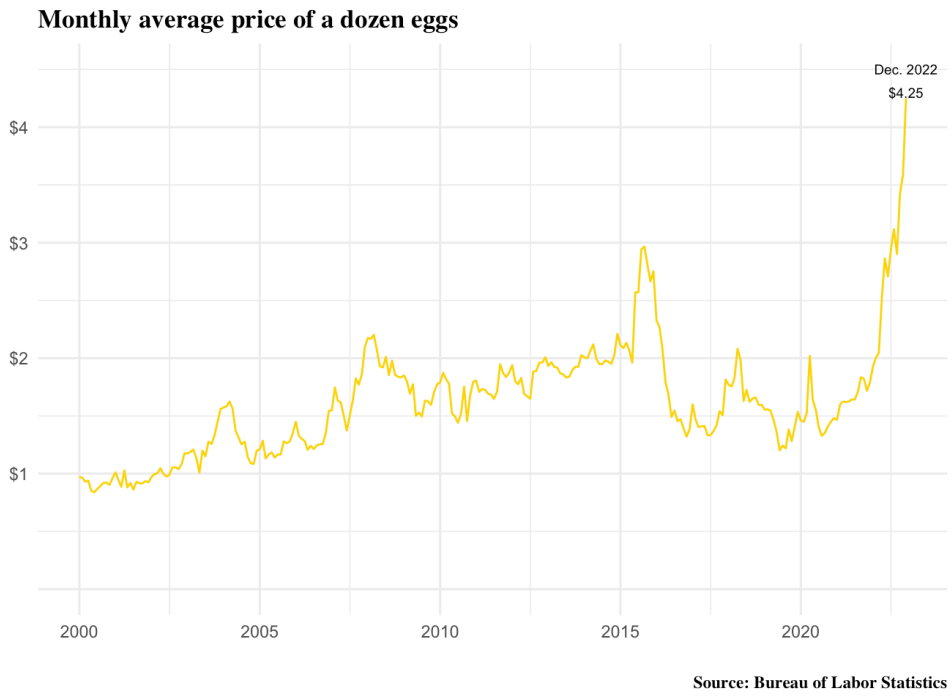



```
x = c(1:10, NA)
is.na(x) == TRUE
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE  TRUE
```

```
# tested code:
# if (is.na(x) == TRUE) {
#   stop('one is true')
# }
```

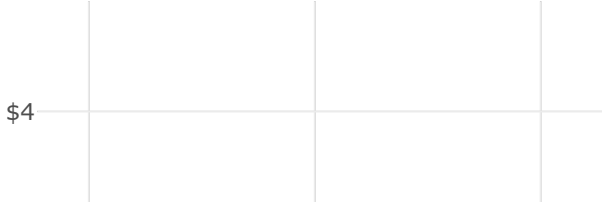
```
gg6 =  gg5 + theme(title = element_text(family = "serif", face =
gg6
```

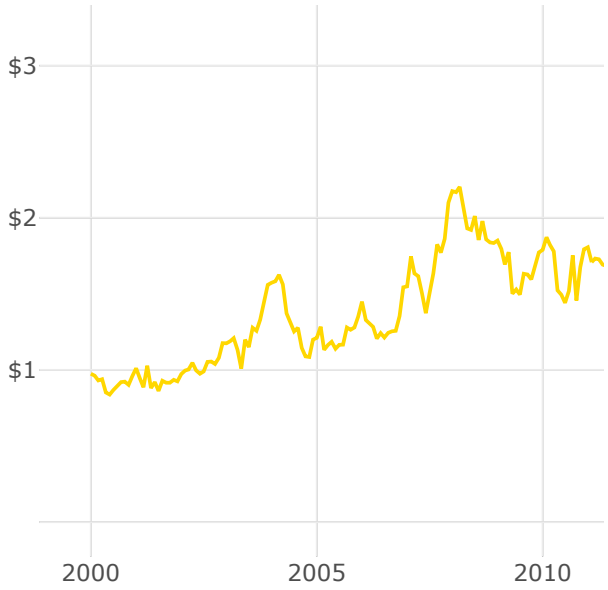



```
library(plotly)
```

```
Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout
```

```
ggplotly(gg6)
```



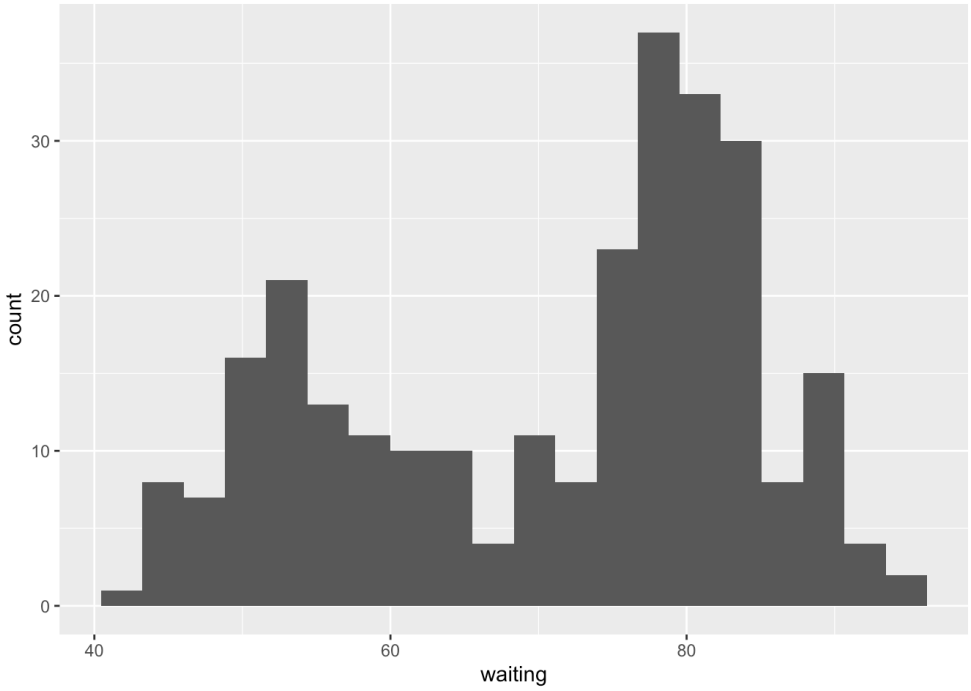


3 9/27 - Strings as Data

```
#making a vector
input = list(variable = "waiting",
             bins = 20)

ggplot(faithful, aes(x = input$variable)) + #error here bc the 
  geom_histogram(bins = input$bins)
```

```
Warning in geom_histogram(bins = input$bins): All aesthetics
have length 1, but the data has 272 rows.
ℹ Please consider using `annotate()` or provide this layer with
data containing
  a single row.

Error in `geom_histogram()`:
! Problem while computing stat.
ℹ Error occurred in the 1st layer.
Caused by error in `setup_params()`:
! `stat_bin()` requires a continuous x aesthetic.
✖ the x aesthetic is discrete.
ℹ Perhaps you want `stat="count"`?
```

```
# using a string as a variable, use .data[[]]
input = list(variable = "waiting",
             bins = 20)

ggplot(faithful, aes(x = .data[["waiting"]])) + #any string, cou
  geom_histogram(bins = input$bins)
```



3.0.0.1 Shiny App Applications

an alternative method to .data[[ ]] would be !!input$name. see below

```
 #selectInput("variable",     #requires in the server section, 
           #             label = "Select a Variable",
           #             choices = colnames(faithful),
           #             selected = "waiting")

# goes with server output:
           # ggplot(faithful, aes(x = .data[["waiting"]])) + #
               # geom_histogram(bins = input$bins)
```

```
 # varSelectInput("variable",     # does the same thing as sele
 #                          label =  "Select a variable",
 #                          data = faithful,
 #                          selected = "waiting")



# output$distPlot <- renderPlot({
#       ggplot(faithful, aes(x = !!input$variable)) + # need !!
#         geom_histogram(bins = input$bins)
```

Open fiathufltestweb qmd file on here to learn how to do a quarto doc for a shiny app.