R Workshop

April 15, 2021

1 R Coding Workshop

4/15/2021

1.1 Overview

Graduate Student Instructor: Kayleigh Barnes

Email: kayleighnb@berkeley.edu

1.1.1 Goals for today

This session is intended to guide you through the practical implementation of basic analytic techniques in R in Jupyter notebooks. R is an open-source statistical computing software used to analyze data. A Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. This workshop will be focused on interactive demonstration in R, but also include time for additional questions and guidance in working through the sample code. We will cover some fundamental coding techniques that will help you in Econ 140, basic data science classes, or research assistant positions. This workshop is for *beginners* that have little or no coding experience.

1.1.2 Important notes

- One attendee from today's workshop will be randomly selected to win a 20 dollar gift card to Amazon
- Attendance to this workshop comes with free access to datacamp through July. Datacamp offers online courses in both R and Python so that you can continue learning after today's workshop
- Link to join Berkeley Econ's datacamp group with @berkeley.edu ID: here (make sure you're signed out of datacamp before clicking this - otherwise the sign-up gets screwed and you'll be asked to pay after the first chapter of any course)

1.2 Jupyter and R Basics

- To create a new notebook, click the "New" button and select R

- Write R script by selecting the option “Code” from the dropdown list, or write text by selecting “Markdown”
- Select “Insert” to add a block of text or code
- Run code by highlighting and selecting “Run”
- Use the # symbol to add comments to the script, or to add headlines to text selections
- To clear your coding output, select Cell=>All Output=>Clear

[1]:

```r
# Clear the workspace, this removes all data and numbers you have stored or saved in R
  rm(list = ls())

  # The help function, using ? or help() before a command will bring up information on what the command does
  ?setwd
  help(setwd)
```

[2]:

```r
#The working directory is the location that R will look for data in
    # this is the same as telling your computer to look in a documents folder when uploading soemthing
getwd()
```

’/home/jovyan/my-work’

User written open-source packages are needed for specific functionality in R (e.g. nice graphics). However, we need to manually install these packages (once) and load them at the beginning of every script. Packages have been pre-installed in Jupyter notebooks. If you are wondering why a command you’ve used before is no longer working, it may be because you haven’t loaded the package.

[3]:

```r
#Install packages
  install.packages('ggplot2')

  # Load required packages
  library(ggplot2)
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done
```

1.3 Loading in data and summary statistics

Now let’s load in the data set. Make sure you have uploaded the data to Jupyter before running the next line of code. We are going to use data on a set of households in Mexico in the 1990’s. The data includes a village ID, a household ID, and demogrpahic variables like income, household size, age and gender of the head of household and a poverty indicator.

[4]:

```
# Reading data into R from a CSV file
#  ?read.table # delete the # at the beginning of this line to view the help␣
↪entry for the "read" command
 MyFirstData <- read.csv('MyFirstData.csv', header = TRUE)
```

Notice that there is no ouput from the code that reads in the data. Unlike excel, R stores the data in the background and we need to use specific comands to interact with it. Once it's read in, we can use several commands to describe the data

[5]:

```
# Structure of the Data
 str(MyFirstData)
```

```
'data.frame':   1200 obs. of  8 variables:
 $ villid   : int  1001106 1001106 1001106 1001106 1001106 1001106 1001106
1001106 1001106 1001106 …
 $ hogid    : Factor w/ 1200 levels "0101103002.0639",..: 10 11 12 13 14 15 16
17 18 19 …
 $ D_HH     : int  1 1 1 1 1 1 1 1 1 1 …
 $ IncomeLab: int  NA NA NA 3200 NA 4320 4800 NA NA 3200 …
 $ famsize  : int  6 6 6 5 5 5 5 6 6 3 …
 $ agehead  : int  29 43 43 25 40 40 39 45 42 22 …
 $ sexhead  : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 2 1 2 2 …
 $ pov_HH   : Factor w/ 2 levels "no pobre","pobre": 2 1 1 2 1 1 1 2 1 2 …
```

[6]:

```
# Summary of the Data
 summary(MyFirstData)
```

```
     villid                         hogid           D_HH          IncomeLab
 Min.   :1001106   0101103002.0639:   1   Min.   :0.0000   Min.   :  160
 1st Qu.:7011004   0101103004.0640:   1   1st Qu.:1.0000   1st Qu.: 1200
 Median :7011019   0101103006.0641:   1   Median :1.0000   Median : 1550
 Mean   :5951112   0101103008.0642:   1   Mean   :0.8107   Mean   : 2243
 3rd Qu.:7015003   0101103012.0644:   1   3rd Qu.:1.0000   3rd Qu.: 2800
 Max.   :7015038   0101103014.0645:   1   Max.   :1.0000   Max.   :27000
                   (Other)        :1194   NA's   :17       NA's   :1024
    famsize          agehead          sexhead         pov_HH
 Min.   : 1.000   Min.   :16.00   Female:257   no pobre:203
 1st Qu.: 3.000   1st Qu.:37.00   Male  :943   pobre   :996
 Median : 5.000   Median :47.00                NA's    :  1
 Mean   : 4.801   Mean   :49.12
 3rd Qu.: 6.000   3rd Qu.:60.00
 Max.   :16.000   Max.   :96.00
                  NA's   :1
```

[7]:

```
# Variable Names
 colnames(MyFirstData)
```

1. 'villid' 2. 'hogid' 3. 'D_HH' 4. 'IncomeLab' 5. 'famsize' 6. 'agehead' 7. 'sexhead' 8. 'pov_HH'

```
[8]: #Number of Observations
       nrow(MyFirstData)
```

1200

```
[9]: #Display first 6 rows of the data
       head(MyFirstData)
```

A data.frame: 6 × 8

	villid <int>	hogid <fct>	D_HH <int>	IncomeLab <int>	famsize <int>	agehead <int>	sexhead <fct>	pov_ <fct>
1	1001106	0101103050.0539	1	NA	6	29	Female	pobr
2	1001106	0101103052.0540	1	NA	6	43	Male	no po
3	1001106	0101103054.0541	1	NA	6	43	Male	no po
4	1001106	0101103056.0542	1	3200	5	25	Female	pobr
5	1001106	0101103058.0543	1	NA	5	40	Male	no po
6	1001106	0101103060.0544	1	4320	5	40	Male	no po

```
[10]: #Tabulate a specific variable (to refer to a variable, use Dataset$VariableName)
        table(MyFirstData$sexhead)
```

```
Female   Male
   257    943
```

1.4 Basic Data Cleaning and Formatting

1.4.1 Category Variable

Right now, we have two categorical variables: sexhead, which indicates the sex of the head of household and pov_HH, which indicates whether a household is below the poverty line. The data entries for these variables are text rather than numbers (we call these string variables in the data science world). Often when doing data analysis, it is easier to map categorical text variables to numbers, particularly 0 and 1. These variables that contain only 0's and 1's are called dummy variables.

Now, suppose we want to create a poor_male variable, which will be defined as 1 if the household is categorized as poor (pov_HH = pobre) and the head of the household is male (sexhead is Male), and 0 otherwise.

```
[11]: #Create one dummy variable based on T/F condition
      MyFirstData$poor_male <- ifelse(MyFirstData$pov_HH == 'pobre' &␣
       ↪MyFirstData$sexhead == 'Male', 1, 0)
        #tabulate the observations
        table(MyFirstData$poor_male)
```

```
  0   1
413 786
```

1.4.2 Numerical Variable

We can use regular mathematical operations to create numerical variables from other variables.

[12]:

```
#Squaring an existing variable
MyFirstData$agehead2 <-  MyFirstData$agehead^2
summary(MyFirstData$agehead2)

#Creating a constant
MyFirstData$constant <- 1
summary(MyFirstData$constant)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
    256    1369    2209    2656    3600    9216       1 


   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1       1       1       1       1       1 
```

New Datasets We may also want to create a new data that summarizes the old, or is a subset of the original dataset.

[13]:

```
#Subset of only observations with male head of hh
data_males<-MyFirstData[ which(MyFirstData$sexhead=='Male'),]
summary(data_males)

#First select variables to aggregate
myvars <- c("villid", "IncomeLab", "famsize", "agehead")
meandata <- MyFirstData[myvars]

#Collapse data to get average values by village.  Could also use "sum" as the␣
 ↪function to get totals
meandata<-aggregate(meandata, by = list(meandata$villid), FUN = mean)
nrow(meandata)
summary(meandata)
```

```
     villid                        hogid          D_HH          IncomeLab    
 Min.   :1001106   0101103002.0639:  1   Min.   :0.0000   Min.   :  160  
 1st Qu.:7011004   0101103004.0640:  1   1st Qu.:1.0000   1st Qu.: 1388  
 Median :7011019   0101103006.0641:  1   Median :1.0000   Median : 1600  
 Mean   :5954876   0101103008.0642:  1   Mean   :0.8254   Mean   : 2277  
 3rd Qu.:7015003   0101103012.0644:  1   3rd Qu.:1.0000   3rd Qu.: 2850  
 Max.   :7015038   0101103018.0647:  1   Max.   :1.0000   Max.   :27000  
                   (Other)        :937   NA's   :15       NA's   :807    
    famsize          agehead         sexhead          pov_HH       poor_male
 Min.   : 1.000   Min.   :18.00   Female:  0   no pobre:156   Min.   :0.0000  
 1st Qu.: 4.000   1st Qu.:36.25   Male  :943   pobre   :786   1st Qu.:1.0000  
 Median : 5.000   Median :47.00                NA's    :  1   Median :1.0000  
 Mean   : 5.022   Mean   :49.05                               Mean   :0.8344  
 3rd Qu.: 6.000   3rd Qu.:60.00                               3rd Qu.:1.0000  
 Max.   :16.000   Max.   :94.00                               Max.   :1.0000  
                  NA's   :1                                   NA's   :1       
    agehead2       constant
 Min.   : 324   Min.   :1  
 1st Qu.:1314   1st Qu.:1  
 Median :2209   Median :1  
 Mean   :2638   Mean   :1  
 3rd Qu.:3600   3rd Qu.:1  
 Max.   :8836   Max.   :1  
 NA's   :1                 
```

24

```
    Group.1            villid          IncomeLab      famsize     
 Min.   :1001106   Min.   :1001106   Min.   : NA   Min.   :4.504  
 1st Qu.:1002032   1st Qu.:1002032   1st Qu.: NA   1st Qu.:4.721  
 Median :1008514   Median :1008514   Median : NA   Median :5.348  
 Mean   :2506884   Mean   :2506884   Mean   :NaN   Mean   :5.316  
 3rd Qu.:2509100   3rd Qu.:2509100   3rd Qu.: NA   3rd Qu.:5.688  
 Max.   :7015038   Max.   :7015038   Max.   : NA   Max.   :6.800  
                                     NA's   :24                   
    agehead     
 Min.   :36.00  
 1st Qu.:40.96  
 Median :45.58  
 Mean   :45.34  
 3rd Qu.:50.25  
 Max.   :52.27  
 NA's   :1      
```

1.5 Making comparisons - T-Tests

A main goal of working with data is to make inferences about the population we are interested in. Much of Econ 140 will be focused on methods to make these inferences: What is the relationship between two variables? Did an experiment have a significant treatment effect?

If you have taken Stats 20, you are likely already familiar with a t-test. T-tests compare the difference in the means of a variable between two groups. The test statistic tells us whether the difference is *significant*, that is we can confidently say that the two groups are different.

[14]:

```
#let's run a t-test comparing the average family size for households above and below the poverty line
t.test(MyFirstData$famsize ~ MyFirstData$pov_HH, var.equal = TRUE)
```

```
Two Sample t-test

data:  MyFirstData$famsize by MyFirstData$pov_HH
t = -5.2032, df = 1197, p-value = 2.303e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.221131 -0.552397
sample estimates:
mean in group no pobre    mean in group pobre
              4.064039               4.950803
```

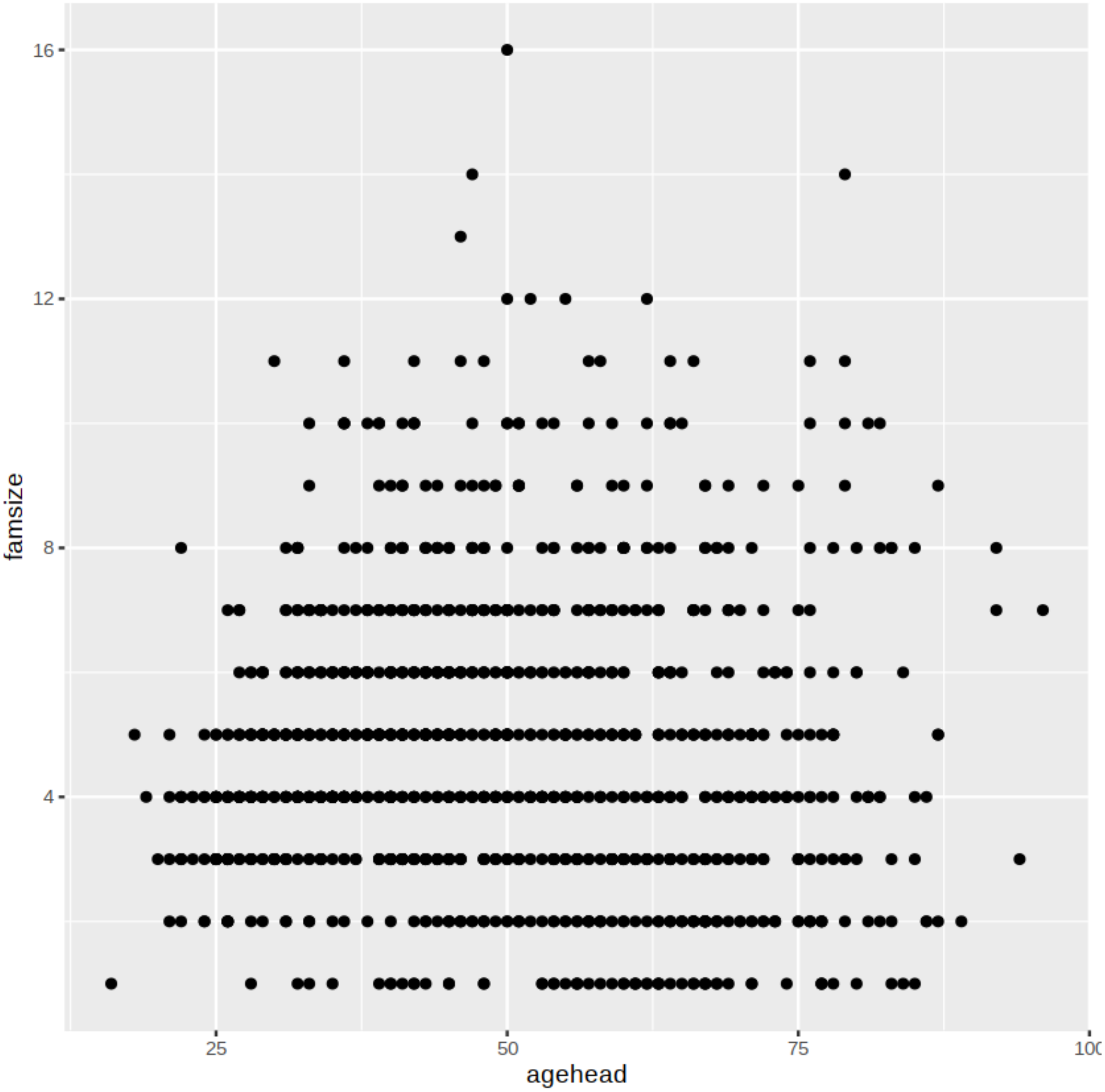
1.6 Visualizing Data

Make sure that the ggplot2 package is included at the top of the script. Below, we show an example of a scatterplot using ggplot. “geom” can be used to denote different types of graphs such as a line graph.

[15]:

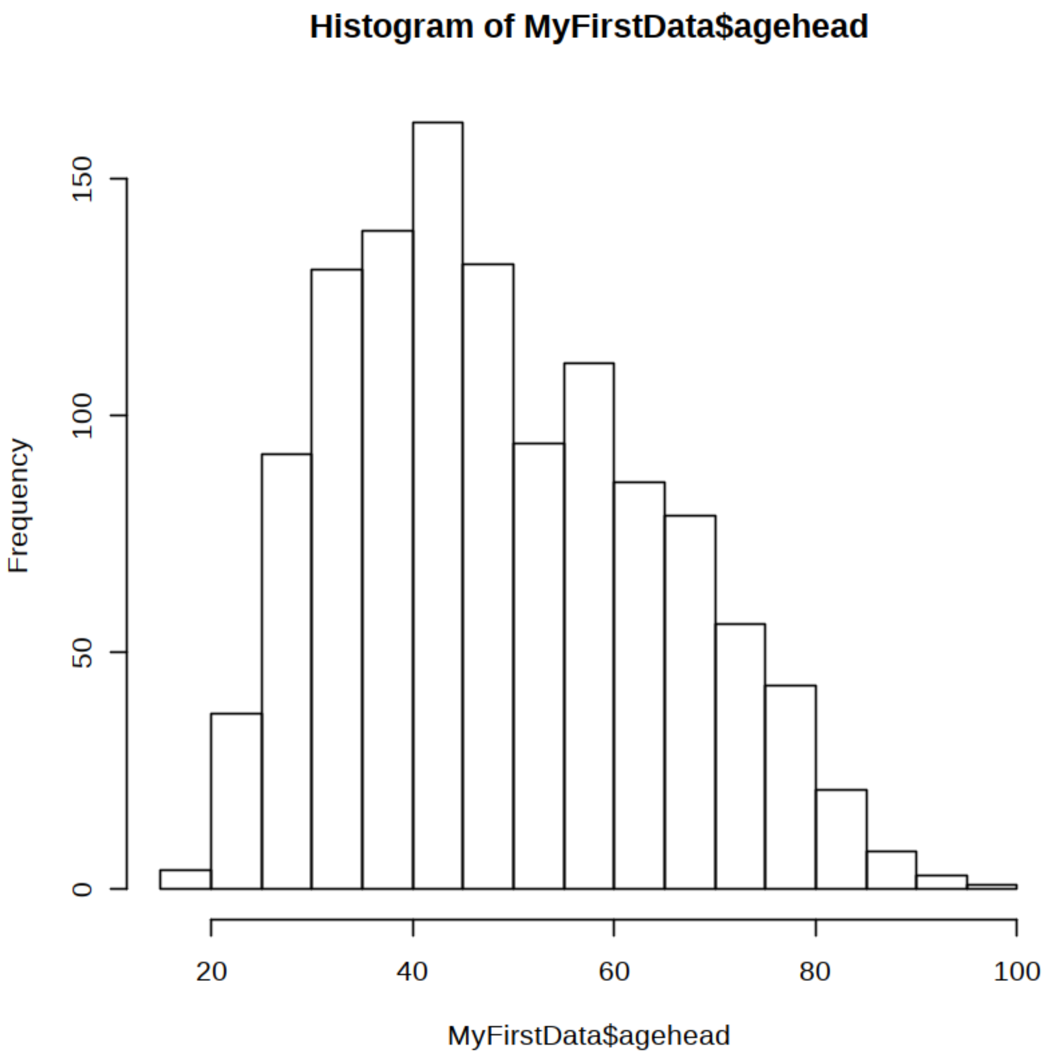
```
ggplot(MyFirstData, aes(x = agehead, y=famsize)) + geom_point()
  ?geom_line
```

```
Warning message:
"Removed 1 rows containing missing values (geom_point)."
```



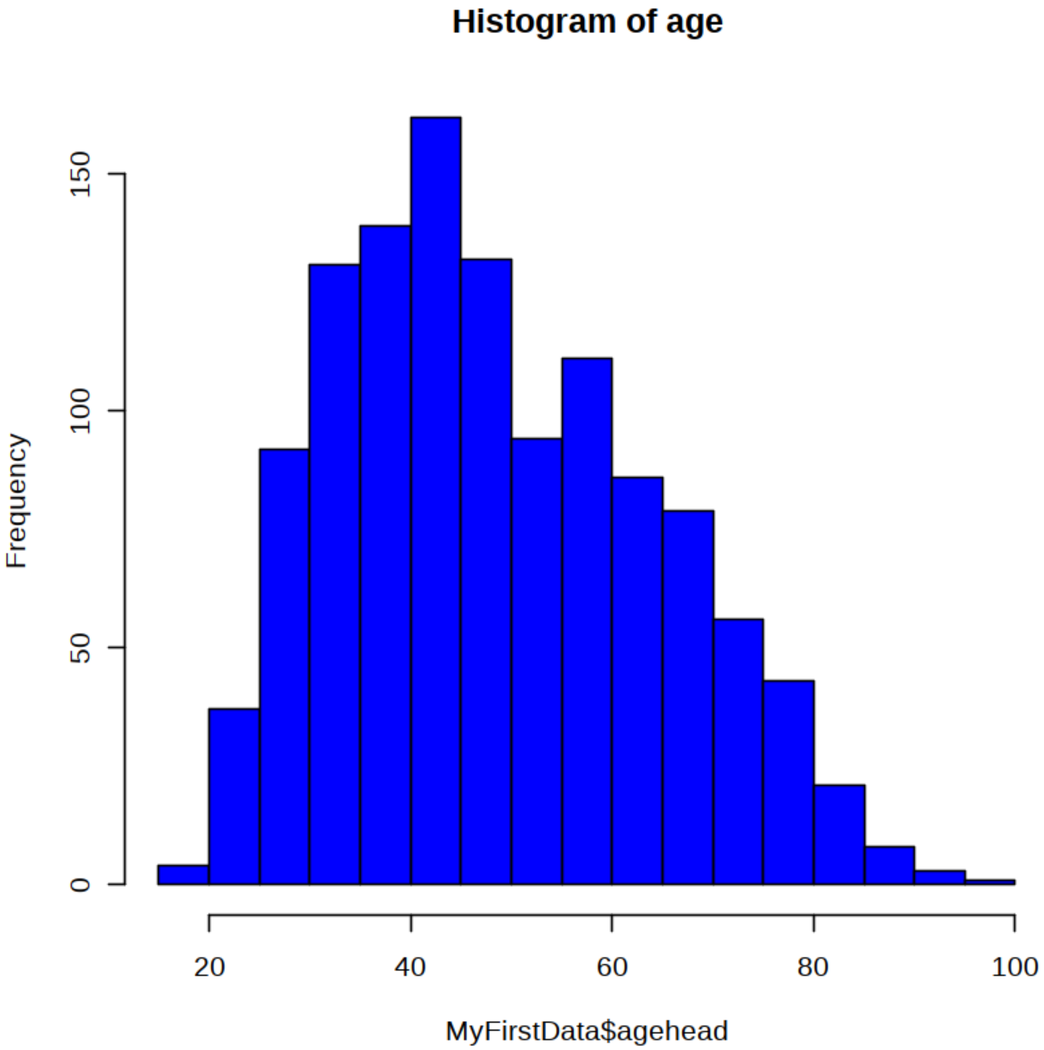
We can use a direct function or ggplot to create a histogram. Notice that changing the options in the function allows you to customize the graph. Use the help function to learn more about the options for each command.

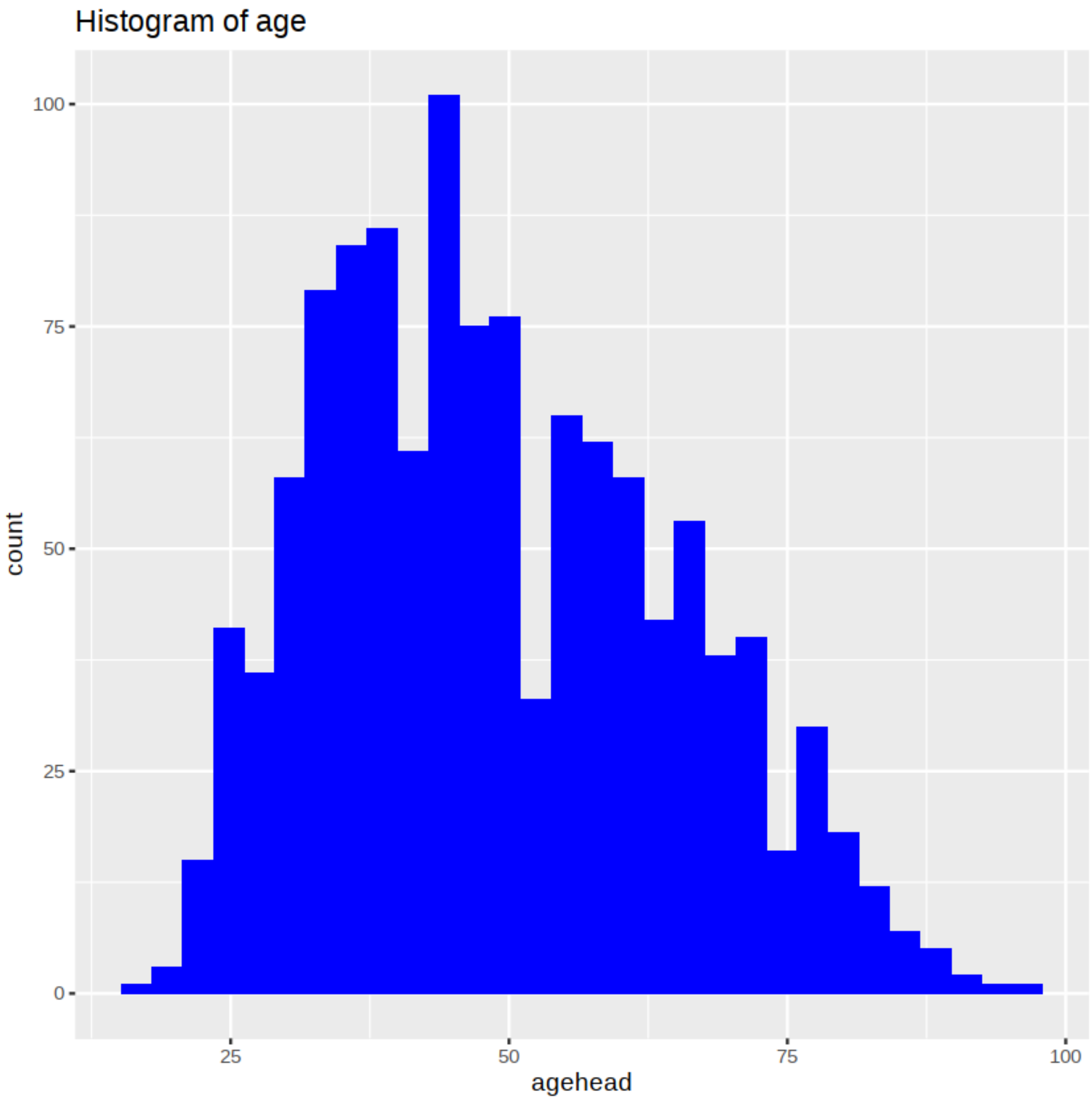
```
[16]: # Base Graphics
      hist(MyFirstData$agehead)
      hist(MyFirstData$agehead, col = "blue", main = "Histogram of age")
      # ggplot2
      ggplot(MyFirstData, aes(x = agehead)) + geom_histogram(fill = "blue") +␣
      ↪ggtitle("Histogram of age")
```



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning message:
"Removed 1 rows containing non-finite values (stat_bin)."
```


Histogram of age
Frequency
0
50
100
150
20
40
60
80
100
MyFirstData$agehead


Histogram of age
count
100
75
50
25
0
25
50
75
100
agehead

[ ]: