Lecture 2 Code

Alex Hoagland

2025-01-16

If you want to customize your .rmd file’s look, you can add some options here. You can also load some packages here.

## Variables

First, we can define some useful variables:

# Basic variables  
a <- 1  
b <- 2  
  
# Let's try out copilot! (This didn't help me)  
# For each letter a, b, c, and d, assign a random number between 1 and 100  
  
# Let's try out chattr! (This did help me)  
# Load necessary libraries  
  
  
# Create a data frame with a column 'letter' containing letters a to z  
letters\_df <- data.frame(letter = letters)  
  
# Assign a random number between 1 and 1000 to each letter  
set.seed(123) # for reproducibility  
letters\_df <- letters\_df %>%  
 mutate(random\_number = sample(1:1000, n(), replace = TRUE))  
  
letters\_df

## letter random\_number  
## 1 a 415  
## 2 b 463  
## 3 c 179  
## 4 d 526  
## 5 e 195  
## 6 f 938  
## 7 g 818  
## 8 h 118  
## 9 i 299  
## 10 j 229  
## 11 k 244  
## 12 l 14  
## 13 m 374  
## 14 n 665  
## 15 o 602  
## 16 p 603  
## 17 q 768  
## 18 r 709  
## 19 s 91  
## 20 t 953  
## 21 u 348  
## 22 v 649  
## 23 w 989  
## 24 x 355  
## 25 y 840  
## 26 z 26

Notice that when you knit your file, this will “spit out” the table above. This is like circling your answer in math homework – you want to publish your work outside of the code so that we can see your final answer.

What about other variable types?

mychar <- "test"  
mybool <- TRUE  
# we'll cover factors more later.   
  
as.numeric(mybool)

## [1] 1

as.numeric(mychar) # what does this mean?

## Warning: NAs introduced by coercion

## [1] NA

as.numeric(mychar == "test")

## [1] 1

## Calculations in R

Let’s try some simple calculations – shout them out. Something to think about – do we want to save them as objects or export them as output?

a + b

## [1] 3

## Now vectors

# Manual assignment  
myvec <- c(1,3,45,6,0)  
myvec2 <- 1:5  
myvec + myvec2

## [1] 2 5 48 10 5

# Evenly spaced vectors  
myvec3 <- seq(from=10,to=1000,length=5)  
myvec+myvec2+myvec3

## [1] 12.0 262.5 553.0 762.5 1005.0

# Random numbers  
myvec4 <- runif(5, min=0, max=100)  
myvec+myvec2+myvec3+myvec4

## [1] 40.91597 277.21136 649.30242 852.72990 1074.07053

Do we like the way this output looks? What if you run the whole chunk at once? How can we do better?

Now let’s summarize a vector here:

outvec <- myvec + 3\*myvec3  
  
# What do each of these tell us?   
mean(outvec)

## [1] 1526

min(outvec)

## [1] 31

median(outvec)

## [1] 1560

sd(outvec)

## [1] 1174.31

range(outvec)

## [1] 31 3000

length(outvec)

## [1] 5

## What have we got so far?

str(a)

## num 1

str(myvec3)

## num [1:5] 10 258 505 752 1000

str(letters\_df)

## 'data.frame': 26 obs. of 2 variables:  
## $ letter : chr "a" "b" "c" "d" ...  
## $ random\_number: int 415 463 179 526 195 938 818 118 299 229 ...

## Matrices

We can go from vectors to matrices easily.

# Manual matrix  
mymat <- matrix(c(1:15), nrow=5, ncol=3)  
mymat

## [,1] [,2] [,3]  
## [1,] 1 6 11  
## [2,] 2 7 12  
## [3,] 3 8 13  
## [4,] 4 9 14  
## [5,] 5 10 15

# Combine myvec, myvec2, and myvec3 into a matrix (copilot)  
mymat2 <- cbind(myvec, myvec2, myvec3)  
mymat + mymat2

## myvec myvec2 myvec3  
## [1,] 2 7 21.0  
## [2,] 5 9 269.5  
## [3,] 48 11 518.0  
## [4,] 10 13 766.5  
## [5,] 5 15 1015.0

mymat \* mymat2 # what is this doing?

## myvec myvec2 myvec3  
## [1,] 1 6 110  
## [2,] 6 14 3090  
## [3,] 135 24 6565  
## [4,] 24 36 10535  
## [5,] 0 50 15000

# mymat %\*% mymat2 # Why doesn't this work?   
t(mymat) %\*% mymat2 # What does this do?

## myvec myvec2 myvec3  
## [1,] 166 55 10050  
## [2,] 441 130 22675  
## [3,] 716 205 35300

# Most of the time, we just want the data  
mydata <- as.data.frame(mymat2)  
# VieW(mydata)

## Now let’s practice working with data

Let’s load the dataset “Uninsured.xlsx” from our GitHub repo, which you will use for one of the upcoming assignments.

# Load the data  
uninsured <- read\_excel("C:\\Users\\alexh\\Dropbox\\Teaching\\HAD5772\_Stats\\HAD5772\_2025W\\HAD5772\\Datasets\\Uninsured.xlsx")   
# View(uninsured)  
  
# Summmarize the data  
names(uninsured)

## [1] "frac\_uninsured" "frac\_bankrupt" "Population"

mean(uninsured)

## Warning in mean.default(uninsured): argument is not numeric or logical:  
## returning NA

## [1] NA

summary(uninsured)

## frac\_uninsured frac\_bankrupt Population   
## Min. :0.01356 Min. :0.02398 Min. : 28438   
## 1st Qu.:0.02520 1st Qu.:0.03074 1st Qu.: 38269   
## Median :0.02935 Median :0.03338 Median : 54112   
## Mean :0.02932 Mean :0.03330 Mean : 73916   
## 3rd Qu.:0.03505 3rd Qu.:0.03717 3rd Qu.: 80351   
## Max. :0.03948 Max. :0.03875 Max. :285749

uninsured %>% head()

## # A tibble: 6 × 3  
## frac\_uninsured frac\_bankrupt Population  
## <dbl> <dbl> <dbl>  
## 1 0.0355 0.0373 95072  
## 2 0.0136 0.0240 92271  
## 3 0.0334 0.0365 76377  
## 4 0.0255 0.0315 75574  
## 5 0.0234 0.0293 56468  
## 6 0.0290 0.0333 38318

str(uninsured)

## tibble [20 × 3] (S3: tbl\_df/tbl/data.frame)  
## $ frac\_uninsured: num [1:20] 0.0355 0.0136 0.0334 0.0255 0.0234 ...  
## $ frac\_bankrupt : num [1:20] 0.0373 0.024 0.0365 0.0315 0.0293 ...  
## $ Population : num [1:20] 95072 92271 76377 75574 56468 ...

# What is the average bankruptcy rate in areas with large populations?   
summary(uninsured$Population)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 28438 38269 54112 73916 80351 285749

uninsured %>% filter(Population >= 80351) %>% summarize(avgbank = mean(frac\_bankrupt))

## # A tibble: 1 × 1  
## avgbank  
## <dbl>  
## 1 0.0324

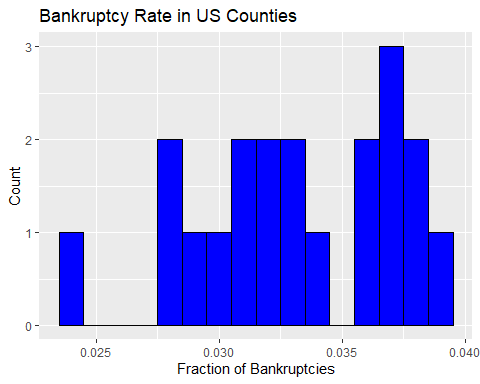
uninsured %>% filter(Population < 80351) %>% summarize(avgbank = mean(frac\_bankrupt)) # Do these seem different? We will learn how to test!

## # A tibble: 1 × 1  
## avgbank  
## <dbl>  
## 1 0.0336

## Visualizing data

Finally, let’s visualize some data.

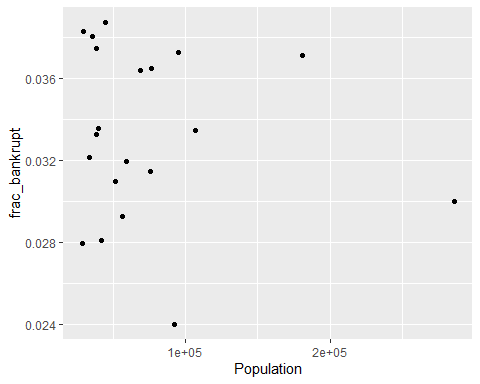
# Let's make a histogram of the bankruptcy rate using ggplot (copilot)  
ggplot(uninsured, aes(x=frac\_bankrupt)) + geom\_histogram(binwidth = 0.001, fill="blue", color="black") + labs(title="Bankruptcy Rate in US Counties", x="Fraction of Bankruptcies", y="Count")



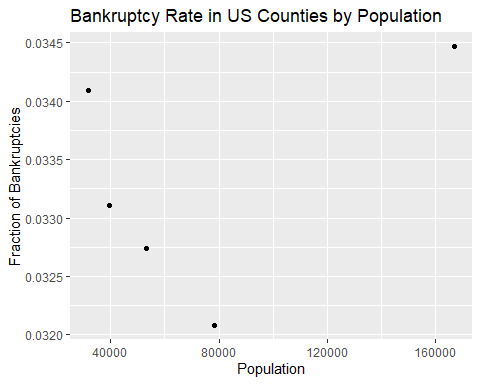
# How can you customize this?

Now this is where we will practice knitting our file! (The binscatter code segment below is for the second half of the lecture)

# What does a raw scatterplot look like   
ggplot(uninsured, aes(x=Population, y=frac\_bankrupt)) + geom\_point()



# Create a binned scatterplot between bankruptcy rate and population using ggplot  
uninsured %>%  
 mutate(bin = ntile(Population, 5)) %>%  
 group\_by(bin) %>% summarize(xmean = mean(Population), ymean = mean(frac\_bankrupt)) %>%   
 ggplot(aes(x = xmean, y = ymean)) +  
 geom\_point() +  
 labs(title = "Bankruptcy Rate in US Counties by Population", x = "Population", y = "Fraction of Bankruptcies")



What stories do you see here? Are they robust (to what)?

## Simple tables and figures

Remember that tables are generally easier to work with outside of knitting, but you might want some simple code to generate the numbers

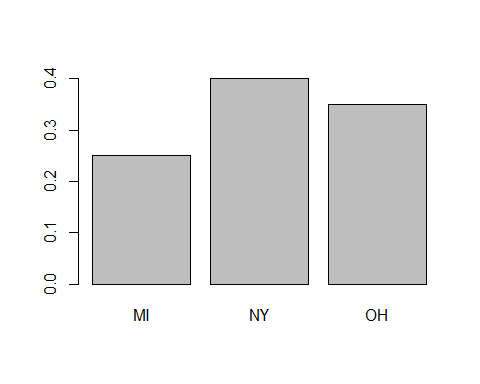
# Suppose we randomly assign each observation to one of three states: NY, MI, or OH  
set.seed(0813)  
uninsured <- uninsured %>%  
 mutate(state = sample(c("NY", "MI", "OH"), n(), replace = TRUE))  
uninsured$state <- as.factor(uninsured$state)  
levels(uninsured$state)

## [1] "MI" "NY" "OH"

# simple table  
table(uninsured$state)

##   
## MI NY OH   
## 5 8 7

# simple histogram  
barplot(prop.table(table(uninsured$state)))



## Measures of central tendency

uninsured <- uninsured %>% mutate(frac\_bankrupt= frac\_bankrupt\*100)  
mean(uninsured$frac\_bankrupt)

## [1] 3.329766

median(uninsured$frac\_bankrupt)

## [1] 3.338064

quantile(uninsured$frac\_bankrupt, probs = c(0.25, 0.5, 0.75))

## 25% 50% 75%   
## 3.074012 3.338064 3.717177

mean(uninsured$Population)

## [1] 73916

median(uninsured$Population)

## [1] 54111.5

quantile(uninsured$Population, probs = c(0.25, 0.5, 0.75))

## 25% 50% 75%   
## 38268.5 54111.5 80350.5