Introduction to R, Day 2 Handout

In this handout, we cover the following topics and R commands:

Topics

- Working with Logical Operators
- Review: Subsetting Vectors and Data Frames
- Subsetting Vectors and Data Frames with `Logical Operators`
- Descriptive Statistics
- Visualizing Univariate Data

R Commands

- Using logical operators: `<`, `<=`, `>`, `>=`, `==`, `!=`
- Using `&`, and `|` to combine logical operators
- Subsetting data with `[]` and `subset()` using logical expressions
- Using `ifelse()` for conditional statements
- Adding features to graphs with `lines()` and `abline()` (lines)
- Using `density()` together with `plot()` to create density plots
- Using `hist()` to generate histograms.
- Using `legend()` to add a legend to a plot

Working With Logical Operators

- When doing analysis, we will want to check and compare specific subsets of the data. Foe example, we might want to analyze how the relationship between two variables changes as we move from one subset of the data to the other (democratic regimes vs. authoritarian regimes, republicans vs. democrats, females vs. males, white candidates vs. black candidates etc.), and we will use this information to make inference.
- Logical operators (`<`, `<=`, `>`, `>=`, `==`, `!=`) allows us to extract subsets of our data by specifcying specific observations and variables.

Logical Operators (Relational Operators)

- `Logical operators` allows us to compare objects and subset data by determining whether a specified condition is `TRUE` or `FALSE`.
- The logic of operators correspond to what we know about them in our everyday life. For example, `>=` will help us evaluate whether a number is greater than or equal to another number. Or, for example, the symbol `!=` will evaluate whether two things we compare are not equal to each other.
- Here are some useful logical operators that we will often use:
- `>` greater than, `>=` greater than or equal to
- `<` less than, `<=` less than or equal to
- `==` equal to
- `!=` not equal to
- When we use these operators to compare two objects in R, we end us with a logical object. You can also compare a vector to a particular number.
- Test if 13 is smaller than 2

```
13 < 2
```

```
## [1] FALSE
```

- Test if 17 is bigger than 5

```
17 >= 5
```

```
## [1] TRUE
```

- Create an object called y which stores information about the logical statement

```
y <- 4 > 3
y
```

```
## [1] TRUE
```

- Previously, we discussed various types of objects such as `numeric`, `character`, and `data frame`. The distinctive feature of logical statements in R is that their class will be logical!

```r
class(y)
```

```
## [1] "logical"
```

- Test if `Hungary` is equal to `hungary` (Recall that R is case sensitive!)

```r
"Hungary" == "hungary"
```

```
## [1] FALSE
```

- Test wheter `Hungary` is `not` equal to `hungary`

```r
"Hungary" != "hungary"
```

```
## [1] TRUE
```

- Logical operators can be applied to individual data entries, entire vectors, or even a dataframe. When applied to a vector, logical operators evaluate each element of the vector.

```r
x <- c(-2, 4, 0, 1, 2)
x >= 1
```

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

```r
x != 1
```

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

- To use multiple logical operators together, we combine logical statements and operations with `&` (AND) and `|` (OR).
- The use of these two operators correspond to the way we think about them in everyday logic.
- For AND statements, both expressions have to be true for the whole expression to be true.

```r
x
```

```
## [1] -2  4  0  1  2
```

```r
(x >= 1) & (x <= 2)
```

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

- For OR statements, either statement being true makes the whole expression true:

```r
x
```

```
## [1] -2  4  0  1  2
```

```
(x >= 1) | (x <= 2)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

- Under the hood, R treats the logical values, TRUE and FALSE as integers where 1 and 0 representing TRUE and FALSE, respectively. This allows us to compute the number and proportion of TRUE elements in a vector. For example, we can add up the number of true statements in a vector using the function `sum()`.

```
x
```

```
## [1] -2  4  0  1  2
```

```
x.int <- (x > 1) | (x <= 13)  # logical vector
x.int
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```
sum(x.int)
```

```
## [1] 5
```

Review: Subsetting Vectors and Data Frames

- Yesterday, we learned how to subset vectors and data frames by using brackets:

Subsetting Vectors using `[ ]`

- When we subset a vector, we are looking only at a portion of the vector that satisfies certain conditions.
- At first, vectors were a little bit abstract, but when we did some examples, we actually saw we can think about our vectors as representing variables in a data frame.

Syntax: variable[condition]

where variable is the name of some variable, and condition is an expression saying what observations you want to look at [1]

Subsetting Data Frames using `[rows, columns]`

Syntax: data[rows, columns]

where the first argument in the brackets tells you what rows to consider and the second tells what columns

Acessing Individual Variables of a Data Frame Using `$`

- We also learned that we can use the `$` operator to access an individual variable of a data frame. It returns a vector containing the specified variable.
- Used in conjuction with the `<-` assignment operator, `$` sign also allows us to create new variables!

[1] Explanation of the syntax is from Will Lowe's teaching notes

Subsetting Vectors and Data Frames with Logical Operators

- Today, we will continue from where we left. But, this time we will explore `how to subset vectors and data frames using logical operators`.
- To make things more concrete, we will go through an example using a dataset from a study on racial discriminatin in labor markets. We will use this dataset first to revise what we did yesterday and then learn subsetting vectors and data frames using logical operators.

Does racial discrimination exist in the labor market? Or, should racial disparities in the unemployment rate be attributed to other factors such as racial gaps in educational attainment¿ To answer this question, Marianne Bertrand and Sendhil Mullainathan conducted a field experiment. Bertrand and Mullainathan sent out fictitious job candidates to potential employers as a response to newspaper ads. The setup of their experiment is as follows: they had every information in the resumes about candidates, but varied the names of job applicants. For some candidates, stereotypically African-American-sounding names such as Lakisha Washington or Jamal Jones were used, whereas other résumés contained stereotypically white-sounding names, such as Emily Walsh or Greg Baker. The researchers then compared the callback rates between these two groups and examined whether applicants with stereotypically black names received fewer callbacks than those with stereotypically white names. The positions to which the applications were sent were either in sales, administrative support, clerical, or customer services (Imai, 33) [2]

Table 1: Resume Experiment Data

Variable	Description
`firstname`	first name of the fictitious job applicant
`sex`	sex of applicant (female or male)
`race`	race of applicant (black or white)
`call`	whether a callback was made (1 = yes, 0 = no)

- First, let's read this dataset into R and jog our memory on summarizing data.
- Load the dataset into R as a data frame object called resume using the function read.csv()

```
resume <- read.csv("day2data/resume.csv")
```

- Check the first siz rows of the data frame

```
head(resume)
```

```
##   firstname    sex  race call
## 1   Allison female white    0
## 2   Kristen female white    0
## 3   Lakisha female black    0
## 4   Latonya female black    0
## 5    Carrie female white    0
## 6       Jay   male white    0
```

- Create a summary of the data frame

[2] The example is adapted from Kosuke Imai's QSS, Chapter 2

```
summary(resume)
```

```
##    firstname        sex          race           call        
##  Tamika : 256   female:3746   black:2435   Min.   :0.00000  
##  Anne   : 242   male  :1124   white:2435   1st Qu.:0.00000  
##  Allison: 232                              Median :0.00000  
##  Latonya: 230                              Mean   :0.08049  
##  Emily  : 227                              3rd Qu.:0.00000  
##  Latoya : 226                              Max.   :1.00000  
##  (Other):3457
```

- Look at the number of observations and variables. Each observation represents a fictitious job applicant.

```
## dimensions of the data
dim(resume)
```

```
## [1] 4870    4
```

```
## nrow() and ncol() also returns the number of rows and
## columns seperately
nrow(resume)  # 4870 observations
```

```
## [1] 4870
```

```
ncol(resume)  # 4 variables
```

```
## [1] 4
```

New function alert: creating a table with table()

- Let's create a table to summarize the categories of the sex variable

```
## create a table for the variable sex
table(resume$sex)
```

```
##
## female   male
##   3746   1124
```

- Turning back to the main question of the researchers: Do résumés with African-American- sounding names are less likely to receive callbacks?
- Let's create a contingency table (cross tabulation) that takes the categories of the race variable as rows and categories of the call variable as columns to answer this question.

```
## create a cross-tab for race and call
race.call.tab <- table(race = resume$race, call = resume$call)
race.call.tab
```

```
##        call
## race       0    1
##   black 2278  157
##   white 2200  235
```

```
## add margins
addmargins(race.call.tab)
```

```
##        call
## race       0    1  Sum
##   black 2278  157 2435
##   white 2200  235 2435
##   Sum   4478  392 4870
```

- The table shows that among 2435 (= 2278 + 157) résumés with stereotypically black names, only 157 received a callback. This seems like supporting the argument for racial discriminatin.
- Using this table, let's compute the callback rate, or the proportion of those who received a callback, for the entire sample.

```
## overall callback rate: total callbacks divided by the
## sample size
sum(race.call.tab[, 2])/nrow(resume)
```

```
## [1] 0.08049281
```

- Now, let's compute the callback rate for black and white applicants seperately and then separately for black and white applicants.

```
## callback rates for each race
black.call <- race.call.tab[1, 2]/sum(race.call.tab[1, ])  # black
white.call <- race.call.tab[2, 2]/sum(race.call.tab[2, ])  # white
diff <- black.call - white.call
diff
```

```
## [1] -0.03203285
```

- The callback rate for the résumés with African- American-sounding names is 0.032 (3.2 percentage points), lower than those with white-sounding names, which suggests the existence of a racial discrimination in the labor market.

Subsetting Using Logical Operators using the Resume Experimental Data

- Compute the callback rate among the résumés with black-sounding names.

⊙ Note: This command syntax subsets the call variable in the resume data frame for the observations whose values for the race variable are equal to black. This is where the logical operators become relevant. You can think `resume$race == "black"` as a logical statement.

```r
## callback rate for black-sounding names
mean(resume$call[resume$race == "black"])
```

```
## [1] 0.06447639
```

- Another way to do the same thing:

```r
## Subset black applicants
resumeB <- resume[resume$race == "black", ]
dim(resumeB)  # this data.frame has fewer rows than the original data.frame
```

```
## [1] 2435    4
```

```r
mean(resumeB$call)  # callback rate for blacks
```

```
## [1] 0.06447639
```

New function alert: subset()

- Subset applicants with black sounding names

```r
resumeB2 <- subset(resume, subset = (race == "black"))
dim(resumeB2)
```

```
## [1] 2435    4
```

```r
resumeB2[1:10, ]
```

```
##    firstname    sex  race call
## 3    Lakisha female black    0
## 4    Latonya female black    0
## 8      Kenya female black    0
## 9    Latonya female black    0
## 10    Tyrone   male black    0
## 11     Aisha female black    0
## 13     Aisha female black    0
## 15     Aisha female black    0
## 18    Tamika female black    0
## 20   Latonya female black    0
```

- Subset applicants who are female and have black sounding names

```r
resumeBF <- subset(resume, subset = (race == "black" & sex ==
    "female"))
resumeBF <- subset(resume, (race == "black" & sex == "female"))  # don't need to write subset
```

- Subset applicants with black sounding names

```r
resumeB2 <- subset(resume, subset = (race == "black"))
dim(resumeB2)
```

```
## [1] 2435    4
```

```r
resumeB2[1:10, ]
```

```
##    firstname    sex  race call
## 3    Lakisha female black    0
## 4    Latonya female black    0
## 8      Kenya female black    0
## 9    Latonya female black    0
## 10    Tyrone   male black    0
## 11     Aisha female black    0
## 13     Aisha female black    0
## 15     Aisha female black    0
## 18    Tamika female black    0
## 20   Latonya female black    0
```

- Subset applicants whose first name is Lakisha

```r
resumeL <- subset(resume, subset = (firstname == "Lakisha"))
dim(resumeL)
```

```
## [1] 200   4
```

```r
resumeL[1:10, ]
```

```
##     firstname    sex  race call
## 3     Lakisha female black    0
## 29    Lakisha female black    0
## 39    Lakisha female black    0
## 60    Lakisha female black    0
## 67    Lakisha female black    0
## 91    Lakisha female black    0
## 112   Lakisha female black    0
## 116   Lakisha female black    0
## 121   Lakisha female black    0
## 125   Lakisha female black    0
```

- Do the same using brackets instead

```r
resumeL2 <- resume[resume$firstname == "Lakisha", ]
dim(resumeL2)
```

```
## [1] 200   4
```

```r
resumeL2[1:10, ]
```

```
##     firstname    sex  race call
## 3     Lakisha female black    0
## 29    Lakisha female black    0
## 39    Lakisha female black    0
## 60    Lakisha female black    0
## 67    Lakisha female black    0
## 91    Lakisha female black    0
## 112   Lakisha female black    0
## 116   Lakisha female black    0
## 121   Lakisha female black    0
## 125   Lakisha female black    0
```

Conditional Statements

- We us the function `ifelse()` to create conditional statements.
- The function takes three main arguments:

1. test. A logical expression (one that is either true or false) e.g. x $<$ 2 or x == "Hungary"
2. yes. What to return if the test is TRUE
3. no. What to return if the test is FALSE

- Here, we are creating a new variable called Black Female. And, we want this variable to take the value of 1 when the applicant is black and female, and 0 when the applicant is not black and female.

```
resume$BlackFemale <- ifelse(resume$race == "black" & resume$sex ==
    "female", 1, 0)
```

- We can look at this new variable with a table

```
table(resume$BlackFemale)
```

```
##
##    0    1
## 2984 1886
```

Descriptive Statistics

- Previously, we learned to following functions to summarize data: `length()`, `min()`, `max()`, `range()`, `mean()`, `median()`, `sum()`, `abs()`, `unique()`
- Today, we will add new functions to this that might be useful for summarizing data: `var()` (variance), `sd()` (standard deviation), and `IQR()` (Inter-quartile Range)
- Our running example will be a policy-focused data from LaLonde's (1986) analysis of the National Supported Work Demonstration. This is a program in 1970s that helped long-term unemployed individuals find private sector jobs and covered the labor costs of their employment for a year [3]

The table below shows the variables and their descriptions:

Table 2: The National Supported Work Demonstration Program Data

[3] Adapted from Monogan's Political Analysis Using R

Variable	Description
treated	Indicator variable for whether the participant received the treatment.
age	Measured in years.
education	Years of education.
black	Indicator variable for whether the participant is African-American.
married	Indicator variable for whether the participant is married.
nodegree	Indicator variable for not possessing a high school diploma.
re74	Real earnings in 1974.
re75	Real earnings in 1975.
re78	Real earnings in 1978.
hispanic	Indicator variable for whether the participant is Hispanic.
u74	Indicator variable for unemployed in 1974.
u75	Indicator variable for unemployed in 1975.

- Let's first load the dataset

```
workdata <- read.csv("day2data/LL.csv")
```

- Check first few rows

```
head(workdata)
```

```
##   treated age education black married nodegree      re74      re75       re78
## 1       1  33        12     0       1        0     0.000     0.000 12418.0703
## 2       1  20        12     0       1        0  8644.156  8644.156 11656.5059
## 3       0  39        12     1       1        0 19785.320  6608.137   499.2572
## 4       1  49         8     0       1        1  9714.597  7285.948 16717.1211
## 5       0  26         8     0       1        1 37211.758 36941.266 30247.5000
## 6       0  38        10     1       1        1 14759.063 14701.947  4393.5229
##   hispanic u74 u75
## 1        0   1   1
## 2        0   0   0
## 3        0   0   0
## 4        0   0   0
## 5        0   0   0
## 6        0   0   0
```

- Number of observations and variables

```
dim(workdata)
```

```
## [1] 722  12
```

- Name of the variables in the data frame

```
names(workdata)
```

```
##  [1] "treated"   "age"       "education" "black"     "married"   "nodegree"
##  [7] "re74"      "re75"      "re78"      "hispanic"  "u74"       "u75"
```

Measures of Central Tendency

- Compute the mean of real earnings in 1974

```
mean(workdata$re74)
```

```
## [1] 3630.738
```

- Most of the time, we would also like to round our quantities of interest to a specific number of decimal places. In R, we use the `round()` function to achieve this. The syntax of the function is as follows: `round(x, digits = X)`. Let's apply this to the mean we calculated above.

```
rmean74 <- round(mean(workdata$re74), digits = 2)
rmean74
```

```
## [1] 3630.74
```

- We can also compute the median of the real earnings in 1974, which might be useful since median is a central tendency measure that is more robust to extreme values than the mean

```
median(workdata$re74)
```

```
## [1] 823.8215
```

- We can observe that the median value is much lower than the mean value. It seems if we were to draw the density plot of the mean, we will see a positive skew, with some extreme values pulling the mean up somewhat. We will visually verify this by drawing a density plot very soon.
- Another useful measure of central tendency reports a range of central values. The inter-quartile range is the middle 50 % of the data.
- Compute IQR of the real earnings in 1974

```
IQR(workdata$re74)
```

```
## [1] 5211.795
```

- Previously, we learned the function `summary()` to get the summary of each variable in a data frame. We can also get a summary of a specific variable.

```
summary(workdata$re74)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0   823.8  3630.7  5211.8 39570.7
```

- The two quantities reported are called the first and third quartiles. The first quartile is a value for which 25 % of the data are less than or equal to the value. Similarly, 75 % of the data are less than or equal to the third quartile. This command reports the spread between the bottom and top of the interquartile range
- The measures we just discussed are informative when we have continuous variables. For categorical and ordinal variables, we can instead use the `table()` function that we already learned to show the frequency of each values.

```r
table(workdata$black)
```

```
##
##   0   1
## 144 578
```

Measures of Dispersion

- Compute the variance of the real earnings in 1974

```r
var(workdata$re74)
```

```
## [1] 38696328
```

- Compute the standard deviation of the real earnings in 1974

```r
## first way of calculating standard deviation
sd(workdata$re74)
```

```
## [1] 6220.637
```

```r
## second way of calculating the standard deviation,
## taking the square root of variance
sqrt(var(workdata$re74))
```

```
## [1] 6220.637
```

Visualizing Univariate Data

- Thus far, we have been summarizing the distribution of variables by reporting their measures of central tendencies such as mean, median, and quantiles. However, visualizing our data is often very helpful to understand and summarize our data. When it comes to data visualization, R has great graphical capabilities that allows us to create beautiful and informative graphs.

Some common graphs for visualizing univariate data:

1. Density Plot

- The function `density()` will calculate the smooth density of a `numeric` object as an output. We will then put this as an input to the `plot()` function to draw a density plot.

2. Bar Chart

- The `barplot()` function will be useful to visualize the distribution of a `factor` variable.

3. Histogram

- The function `hist()` is a common method for visualizing the distribution of a `numeric` variable rather than a `factor` variable

Summary of Options Used in Plots

- Below are some common graphing parameters we can you can use with plot, lines, points:

- `main`. Set the main figure title.
- `xlab`. Set the x-axis label.
- `ylab`. Set the y-axis label.
- `col`. Set the line or point color, e.g. “red”, or “blue”
- ‘xlim. Specify the x-axis limits, e.g. c(0, 4) for the interval [0, 4]
- ‘ylim. Specify the y-axis limits
- `lty`. Change the line type. 1 is solid, 2 is dashed, and 3–5 are different of dashed lines.
- `pch`. Set the plotting character of points. 1 is an unfilled circle.

- Legend adds a legend to your plot. It has the following arguments:

- The first argument is the legend’s location. We can choose one of ‘topleft’, ‘bottomleft’, ‘topright’, or ‘bottom right’.
- `legend`. is a vector of character strings, indicating what the legend should contain.
- `col`. A vector of colors, corresponding to the elements of legend.
- `lty`. A vector of line types, corresponding with the elements of legend.
- `bg = "grey"`. Turns the background of the legend from the default background color, usually white, to grey.

Density Plot

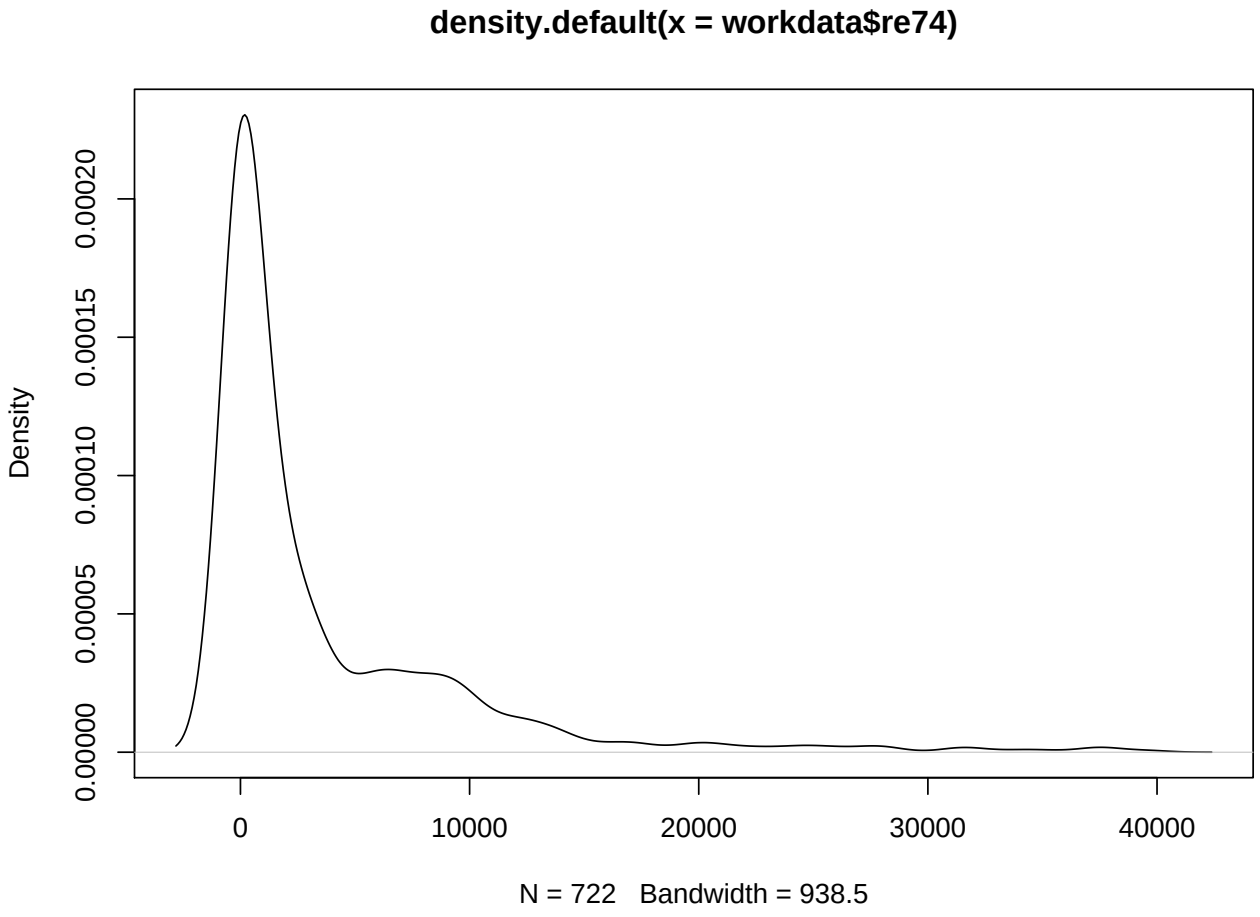
Variable: Real Earnings from The National Supported Work Demonstration Program Data

1. Producing a single density plot

Step 1: Produce a simple figure

- The basic syntax for producing a density plot in R is as follows:

```
plot(density(workdata$re74))
```

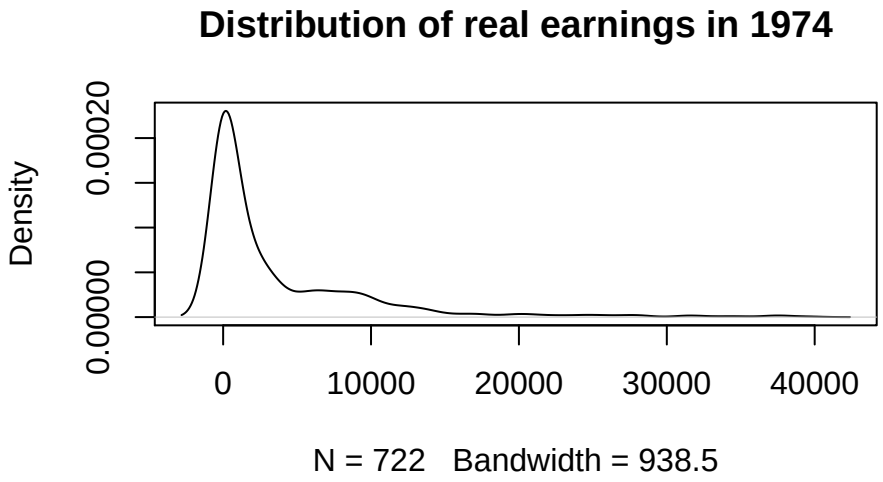


Step 2: Iteratively make your graph better by adding some common graphing paramaters

Fixing the Title

- The parameter `main` will allow you to set the title

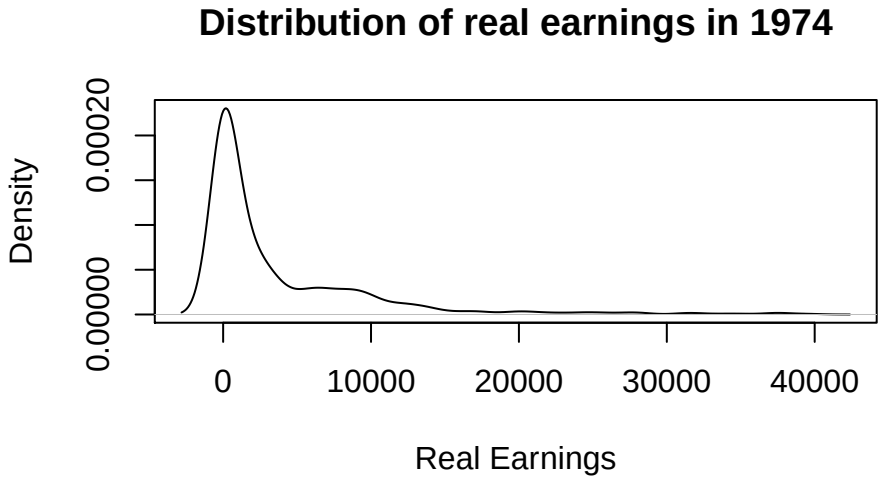
```
plot(density(workdata$re74), main = "Distribution of real earnings in 1974")
```



Fixing the X-axis label

- The parameter `xlab` will allow you change the x-axis label. Similarly, the parameter `ylab` will allow you change the y-axis label.

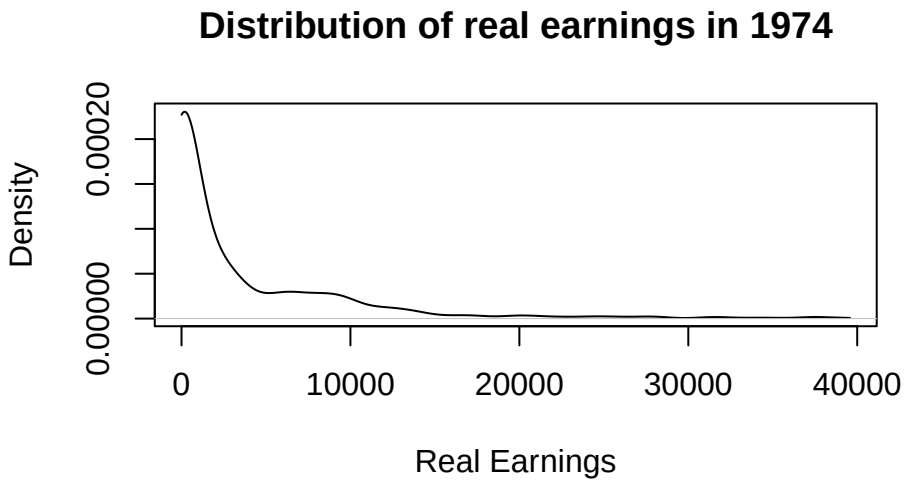
```
plot(density(workdata$re74), main = "Distribution of real earnings in 1974",
    xlab = "Real Earnings")
```



Truncating the Density Plot at 0

- The density plot goes below 0. Let's fix this as well. We will use the parameter `cut` to do this.

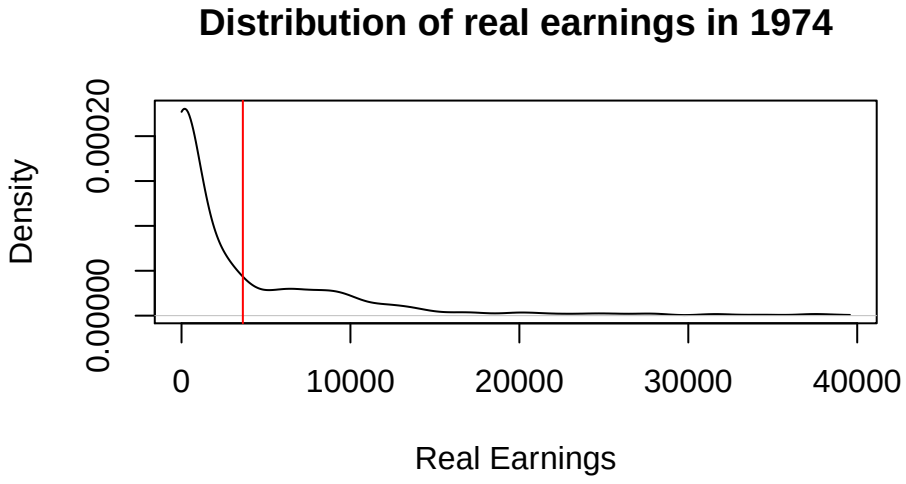
```
plot(density(workdata$re74, cut = 0), main = "Distribution of real earnings in 1974",
    xlab = "Real Earnings")
```



Adding a vertical line for the mean

- Let's also add the mean of real earnings in 1974 on the graph as a vertical line. We will use `abline` to do this.

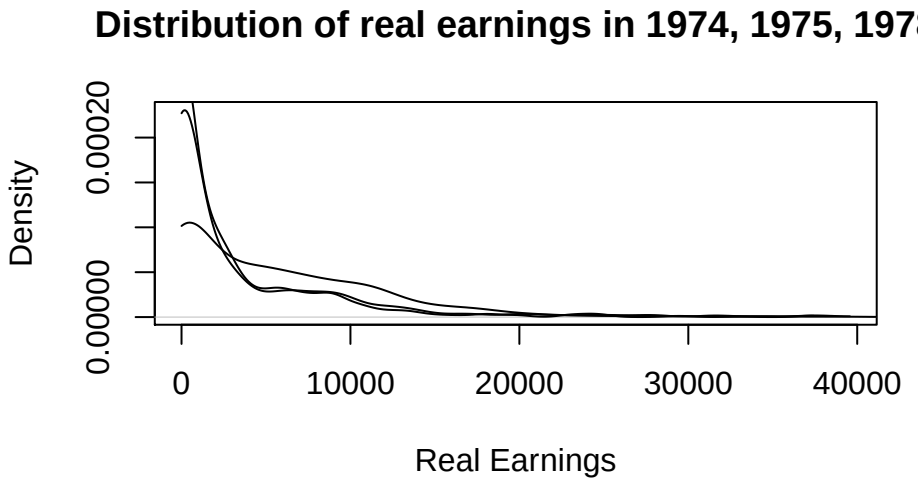
```
plot(density(workdata$re74, cut = 0), main = "Distribution of real earnings in 1974",
    xlab = "Real Earnings")
abline(v = mean(workdata$re74), col = "red")
```



2. Adding Density Plots to an Existing Figure

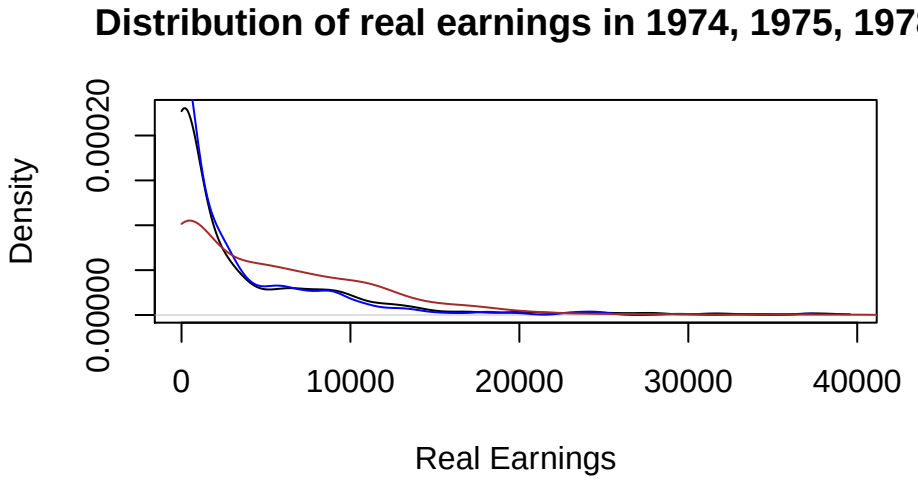
- Now let's put the density plots of different years 74, 75, and 78 in a single plot. This can be helpful when we want to explain how our quantities of interest varies over time.

```
plot(density(workdata$re74, cut = 0), main = "Distribution of real earnings in 1974, 1975, 1978",
    xlab = "Real Earnings")
lines(density(workdata$re75, cut = 0))
lines(density(workdata$re78, cut = 0))
```



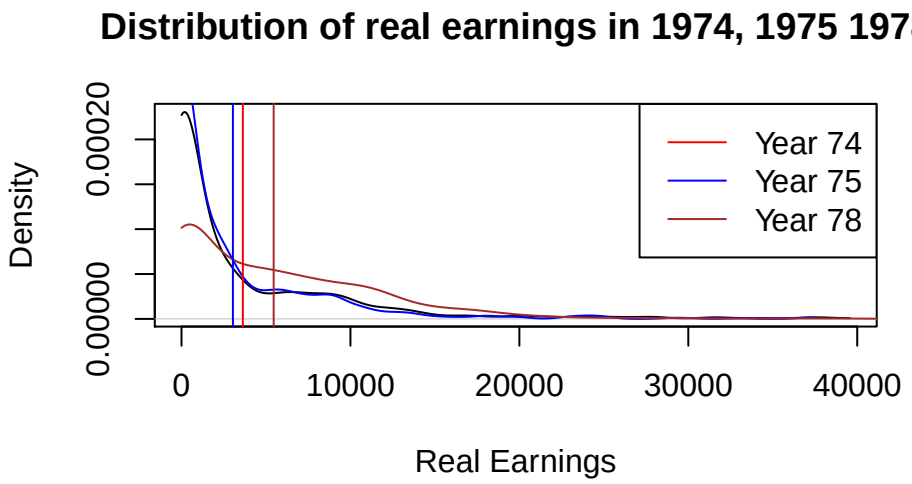
Adding Color to Differentiate Lines

```
plot(density(workdata$re74, cut = 0), main = "Distribution of real earnings in 1974, 1975, 1978",
    xlab = "Real Earnings")
lines(density(workdata$re75, cut = 0), col = "blue")
lines(density(workdata$re78, cut = 0), col = "brown")
```



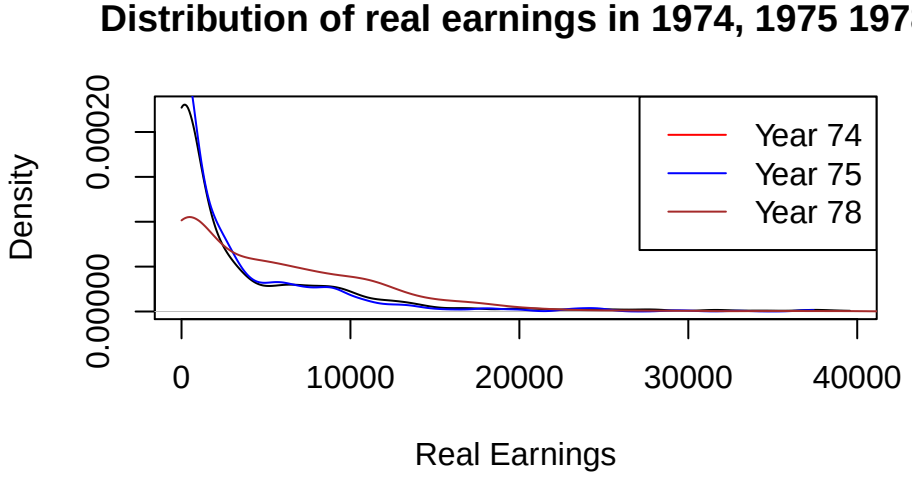
Adding Vertical Lines for the Mean of Each Year

```
plot(density(workdata$re74, cut = 0), main = "Distribution of real earnings in 1974, 1975 1978",
    xlab = "Real Earnings")
lines(density(workdata$re75, cut = 0), col = "blue")
lines(density(workdata$re78, cut = 0), col = "brown")
legend("topright", legend = c("Year 74", "Year 75", "Year 78"),
    col = c("red", "blue", "brown"), lty = c(1, 1, 1))
abline(v = mean(workdata$re74), col = "red")
abline(v = mean(workdata$re75), col = "blue")
abline(v = mean(workdata$re78), col = "brown")
```



Adding a Legend

```
plot(density(workdata$re74, cut = 0), main = "Distribution of real earnings in 1974, 1975 1978",
    xlab = "Real Earnings")
lines(density(workdata$re75, cut = 0), col = "blue")
lines(density(workdata$re78, cut = 0), col = "brown")
legend("topright", legend = c("Year 74", "Year 75", "Year 78"),
    col = c("red", "blue", "brown"), lty = c(1, 1, 1))
```



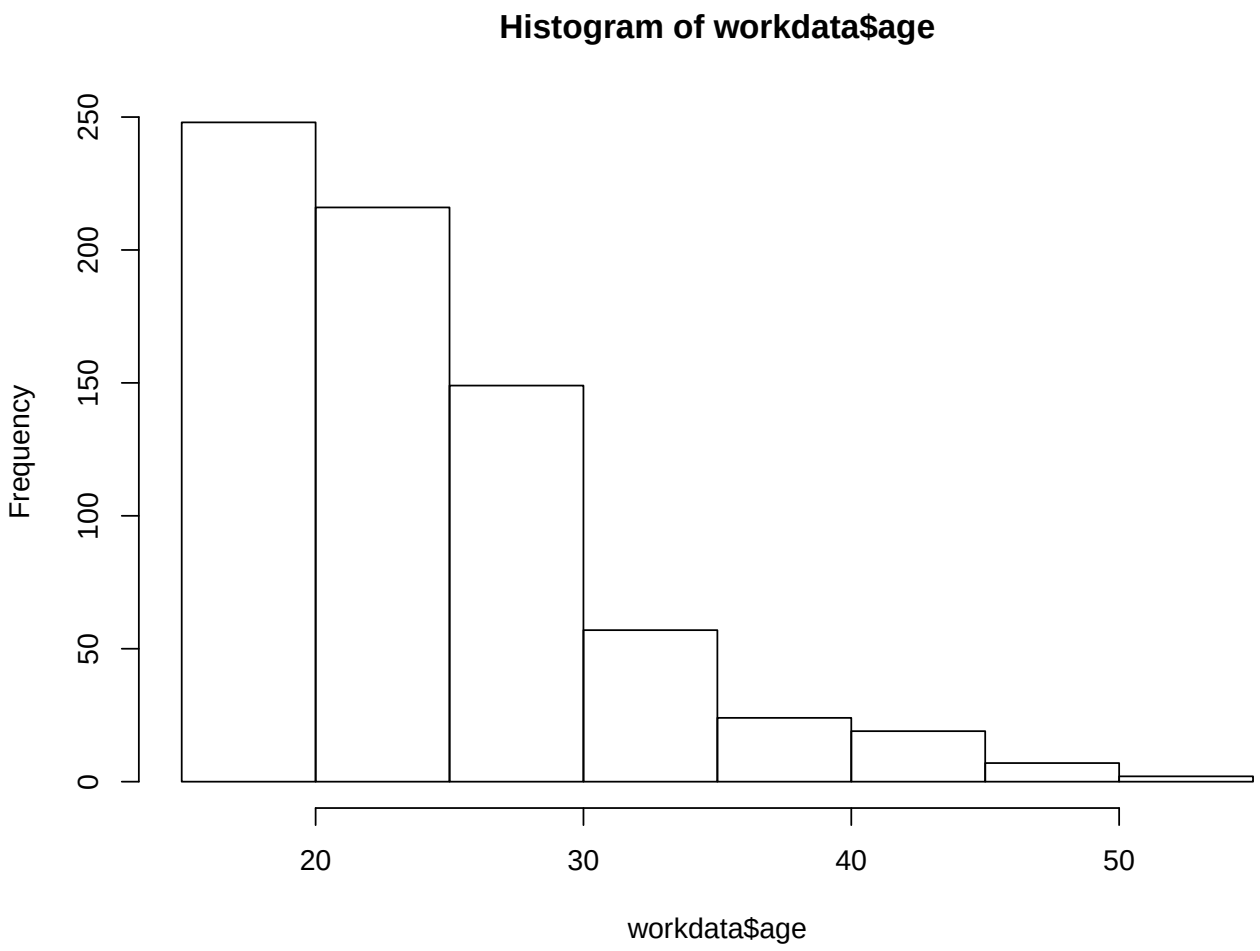
Histogram

Variable: Age from The National Supported Work Demonstration Program Data

Step 1: Produce a simple figure

- The basic syntax for producing a histogram in R is as follows:

```
hist(workdata$age)
```



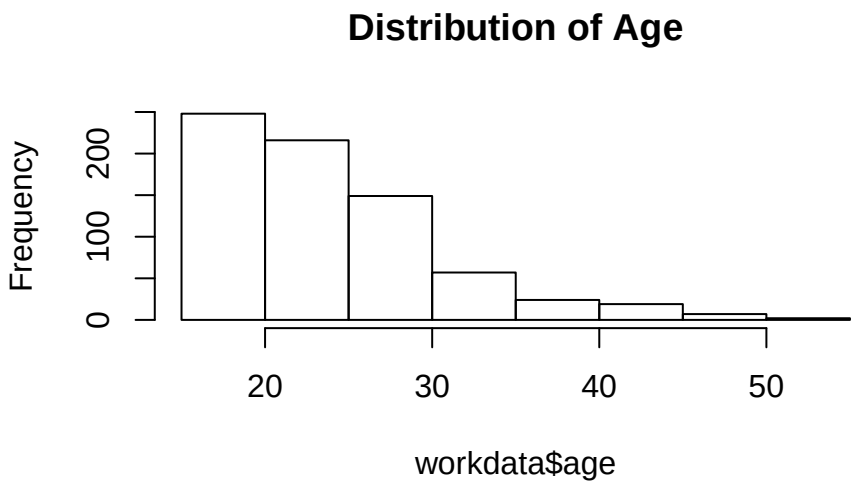
- Set the freq (frequency) paramater to `FALSE`.

⊙ Note: The default for this parameter is TRUE, which plots the frequency, i.e., counts, instead of using density as the height of each bin. Using density rather than frequency is useful for comparing two distributions, because the density scale is comparable across distributions even when the number of observations is different (Imai, QSS, 82)

Step 2: Iteratively make your graph better by adding some common graphing paramaters

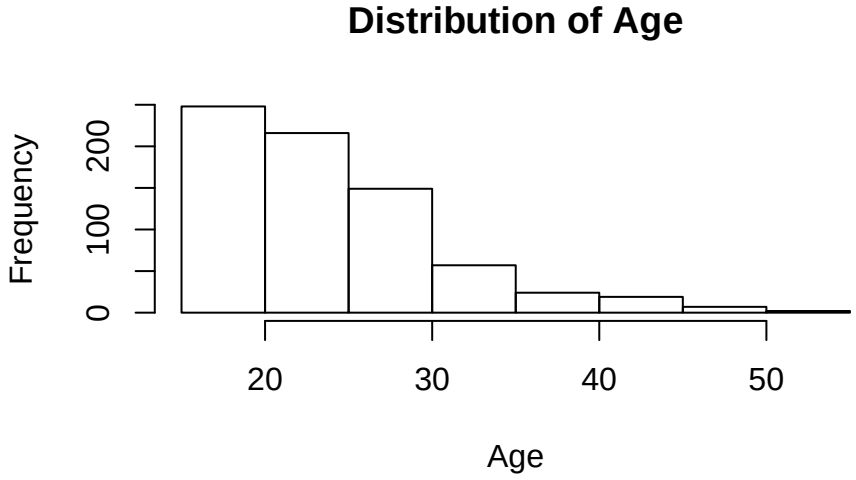
Fixing the Title

```
hist(workdata$age, main = "Distribution of Age")
```



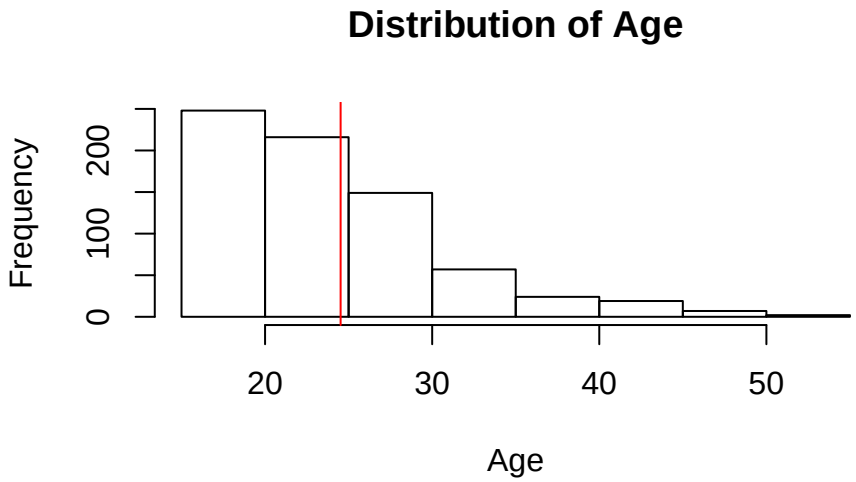
Fixing the X-axis label

```
hist(workdata$age, main = "Distribution of Age", xlab = "Age")
```



Adding a vertical line for the mean

```
hist(workdata$age, main = "Distribution of Age", xlab = "Age")
abline(v = mean(workdata$age), col = "red")
```



Note: Some of the material and practice exercises in this handout are adapted from and inspired by Kosuke Imai, Matthew Blackwell, and James Monogan's textbooks and online teaching materials.

Practice Exercises

Data Set: drugCoverage.csv

Peake and Eshbaugh-Soha's (2008) analysis of drug policy coverage includes the following variables: a character-based time index showing month and year (Year), news coverage of drugs (drugsmedia), an indicator for a speech on drugs that Ronald Reagan gave in September 1986 (rwr86), an indicator for a speech George H.W. Bush gave in September 1989 (ghwb89), the president's approval rating (approval), and the unemployment rate (unemploy) [4]

```
## Load the data
drugdat <- read.csv("day2data/drugCoverage.csv")

## Check the first six rows
head(drugdat)
```

```
##     Year drugsmedia unemploy approval rwr86 ghwb89
## 1 Jan-77          1      7.5     66.0     0      0
## 2 Feb-77          7      7.6     71.0     0      0
## 3 Mar-77         11      7.4     72.3     0      0
## 4 Apr-77          4      7.2     65.0     0      0
## 5 May-77          6      7.0     65.0     0      0
## 6 Jun-77          4      7.2     63.0     0      0
```

```
## Dimensions of the data
dim(drugdat)
```

```
## [1] 192   6
```

```
ncol(drugdat)
```

```
## [1] 6
```

```
nrow(drugdat)
```

```
## [1] 192
```

1. What can you learn simply by applying the summary command to the full data set? What jumps out most clearly from this output? Are there any missing values in these data?

```
## the descriptive statistics summary also list the
## number of missing observations we have on a given
## variable (under NA's), if any are missing.
summary(drugdat)  # seems like there are no NAs
```

```
##       Year       drugsmedia        unemploy         approval
##  Apr-77 :  1   Min.   :  1.00   Min.   : 5.000   Min.   :29.00
##  Apr-78 :  1   1st Qu.:  8.00   1st Qu.: 5.900   1st Qu.:44.38
##  Apr-79 :  1   Median : 15.00   Median : 7.000   Median :53.25
```

[4] Adapted from Monogan's Political Analysis Using R

```
##  Apr-80 :  1   Mean   : 20.54   Mean   : 6.978   Mean   :53.30
##  Apr-81 :  1   3rd Qu.: 26.00   3rd Qu.: 7.500   3rd Qu.:63.00
##  Apr-82 :  1   Max.   :156.00   Max.   :10.800   Max.   :84.80
##  (Other):186
##      rwr86              ghwb89
##  Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.000000   1st Qu.:0.000000
##  Median :0.000000   Median :0.000000
##  Mean   :0.005208   Mean   :0.005208
##  3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :1.000000   Max.   :1.000000
##
```

```
#################################################################

## A quick note of checking missing values/NAs in a data
## frame

## another way to check NAs in a data frame is by using
## the 'is.na()' function.  If you use is.na() alone it
## will display information about all the observations,
## which won't help us to understand the number of NAs We
## can instead create a table for missing values.  TRUE
## represents the number of missing values Here we don't
## have any missing values, therefore we don't see TRUE
table(is.na(drugdat))  # No NAs
```

```
##
## FALSE
##  1152
```

```
## Or you can find the sum of NAs
sum(is.na(drugdat))
```

```
## [1] 0
```

2. Using the mean function, compute the following:

- What is the mean of the indicator for George H.W. Bush's 1989 speech?

```
## mean of Bush's 1989 speech
mean(drugdat$ghwb89)
```

```
## [1] 0.005208333
```

```
## as the question says, this is an indicator variable
## can only take 0 or 1
range(drugdat$ghwb89)
```

```
## [1] 0 1
```

```
table(drugdat$ghwb89)
```

```
##
##   0   1
## 191   1
```

- What is the mean level of presidential approval?

```
## mean of approval
mean(drugdat$approval)
```

```
## [1] 53.29896
```

3. What is the median level of media coverage of drug-related issues?

```
## median level of media coverage of drug related issues
median(drugdat$drugsmedia)
```

```
## [1] 15
```

4. What is the interquartile range of media coverage of drug-related issues?

```
## IQR of media coverage of drug related issues
IQR(drugdat$drugsmedia)
```

```
## [1] 18
```

5. Report two frequency tables:

- In the first, report the frequency of values for the indicator for Ronald Reagan's 1986 speech.

```
## freq for Reagan
table(drugdat$rwr86)
```

```
##
##   0   1
## 191   1
```

```
## we can also add an informative label to this
table(Reagan = drugdat$rwr86)
```

```
## Reagan
##   0   1
## 191   1
```

- In the second, report the frequency of values for the unemployment rate in a given month.

```
## freq for unemployment
table(drugdat$unemploy)
```

```
##
##    5  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9    6  6.1  6.2  6.3  6.4  6.6  6.7
##    1    7    6   11    1    3    7    4   10    8    2    3    8    3    5    2
##  6.8  6.9    7  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9    8  8.3  8.5  8.6
##    5    8    9    4   14    7   13   10    6    3    4    1    1    2    2    1
##  8.8  8.9    9  9.2  9.3  9.4  9.5  9.6  9.8 10.1 10.2 10.3 10.4 10.8
##    1    1    1    1    1    2    1    1    2    3    1    1    3    2
```

```
## modal value of unemploymenet
which.max(table(drugdat$unemploy))
```

```
## 7.2
##  21
```

6. What are the variance, standard deviation, and median absolute deviation for news coverage of drugs?

```
## variance of drugs media coverage
var(drugdat$drugsmedia)
```

```
## [1] 428.7526
```

```
## standard deviation of drugs media coverage
sqrt(var(drugdat$drugsmedia))
```

```
## [1] 20.70634
```

```
## or alternatively
sd(drugdat$drugsmedia)
```

```
## [1] 20.70634
```

```
## median abs deviation of drugs media coverage
mad(drugdat$drugsmedia)
```

```
## [1] 11.8608
```

7. What are the 10th and 90th percentiles of presidential approval in this 1977–1992 time frame?

```
quantile(drugdat$drugsmedia, c(0.1, 0.9))
```

```
##  10%  90%
##  4.0 41.9
```