Week 2: Summarizing Data

Wenhao Jiang

Department of Sociology
New York University

September 17, 2022

Data

Data Structures

- R also has a number of basic data *structures*.
- A data structure is either
  - homogeneous (all elements are of the same data type)
  - heterogeneous (elements can be of more than one data type).

Dimension	**Homogeneous**	**Heterogeneous**
1	Vector	List
2	Matrix	Data Frame
3+	Array	

Data frame

Data frame basics

- A **data frame** is the **most** common way that we store and interact with data

```
example_data <- data.frame(x = c(1, 3, 5, 7, 9, 1, 3, 5, 7, 9),
                           y = c(rep("Hello", 9), "NYU"),
                           z = rep(c(TRUE, FALSE), 5))
```

- A data frame is a collection of vertical vectors
  - Each vector must contain the same data type
  - The difference vectors can store different data types

```
example_data
```

```
##    x     y     z
## 1  1 Hello  TRUE
## 2  3 Hello FALSE
## 3  5 Hello  TRUE
## 4  7 Hello FALSE
## 5  9 Hello  TRUE
## 6  1 Hello FALSE
## 7  3 Hello  TRUE
## 8  5 Hello FALSE
## 9  7 Hello  TRUE
## 10 9   NYU FALSE
```

Read data frame

- In most cases, we do not deal with self-created data frames
- Instead, we use external data (e.g., surveys) and import them to `R`/`RStudio`

Read data frame

- External data can be stored in different formats
- One typical format is `.csv`
- Read `.csv` file into R
  - `read.csv()` function as default
  - `read_csv()` function from the `readr` package. This is faster for larger data

Read data frame

- But before that, you should let R know where the file is located
- In R, we call the **folder** that R currently is reading **the working directory**
- To see the current working directory, type in

```
getwd()
```

```
## [1] "/Users/wenhao/Dropbox/Teaching/Week 2"
```

Read data frame

- If you want to re-set the working directory to the folder that you store the file you want to read, use `setwd()` function

```
setwd()
```

- `RStudio` also allows a more "intelligent" way to reset the working directory

```
## set working directory
setwd("~/Dropbox/Teaching/Week 2")

## read the file
gss <- read.csv("GSS_SOCUA_W2.csv")
```

Examine dataframe basics

- To see the first n rows of the data

```
head(gss, n=5)
```

```
##   year id_ marital sibs childs age sex relig relig16
## 1 1972   1       5    3      0  23   2     3    -100
## 2 1972   2       1    4      5  70   1     2    -100
## 3 1972   3       1    5      4  48   2     1    -100
## 4 1972   4       1    5      0  27   2     5    -100
## 5 1972   5       1    2      2  61   2     1    -100
```

- To see the last n rows of the data

```
tail(gss, n=5)
```

```
##       year  id_ marital sibs childs  age sex relig relig16
## 68842 2021 4467       5    1      0   21   2     1     -80
## 68843 2021 4468       1    2      2   29   2     1     -80
## 68844 2021 4469       1    1      2 -100   2     7     -80
## 68845 2021 4470       3    3      2   68   2     1     -80
## 68846 2021 4471       1   13      1   48   1     1     -80
```

- To view the whole dataset

```
View(gss)
```

- `View()` function generally not recommended, especially if the data is large

Examine dataframe basics

- To check the general "structure" of the data frame

```
str(gss)
```

```
## 'data.frame':    68846 obs. of  9 variables:
##  $ year   : int  1972 1972 1972 1972 1972 1972 1972 1972 1972 1972 ...
##  $ id_    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ marital: int  5 1 1 1 1 5 3 5 5 1 ...
##  $ sibs   : int  3 4 5 5 2 1 7 1 2 7 ...
##  $ childs : int  0 5 4 0 2 0 2 0 2 4 ...
##  $ age    : int  23 70 48 27 61 26 28 27 21 30 ...
##  $ sex    : int  2 1 2 2 2 1 1 1 2 2 ...
##  $ relig  : int  3 2 1 5 1 1 2 3 1 1 ...
##  $ relig16: int  -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 ...
```

- `str()` will display the number of **observations** and **variables**, list the variables, give the type of each variable, and show some elements of each variable

Examine dataframe basics

- `colnames()` function to obtain names of the variables in the dataset

```
colnames(gss)
```

```
## [1] "year"    "id_"     "marital" "sibs"    "childs"  "age"     "sex"
## [8] "relig"   "relig16"
```

- To access one of the variables **as a vector**, we use the $ operator

```
gss$sibs
```

- We can use `min()` or `max()` to inspect the miminum and maximum value of the variable

```
min(gss$sibs)
```

```
## [1] -100
```

```
max(gss$sibs)
```

```
## [1] 68
```

- We can use `unique()` to inspect the unique values of the variable

```
unique(gss$marital)
```

```
## [1]   5   1   3   2   4 -99 -97 -98
```

- You can quickly spot some unusual observartions. Deal with them carefully!

- We can use the `dim()`, `nrow()` and `ncol()` functions to obtain information about the dimension of the data frame

```
dim(gss)
```

```
## [1] 68846     9
```

```
nrow(gss)
```

```
## [1] 68846
```

```
ncol(gss)
```

```
## [1] 9
```

Subsetting data

- Subsetting data frames can work much like subsetting matrices using square brackets, `[,]`.
- But instead of using column indexes, we use column names
- We can select a single column by

```
gss[gss$year > 2018, "marital"]
```

- We can also select multiple columns by creating a vector for column names

```
gss[gss$year > 2018, c("marital","year")]
```

Package dplyr

- To subset data, another approach specifically designed for data frames is calling the `filter` and `select` functions from the `dplyr` package
- `select` function selects certain **columns**
- `filter` function filters certain **rows** (or **observations**) based on some conditions

```
library(dplyr)
gss %>%
  filter(year>2018) %>%
  select(marital,sibs,age,sex)
```

Summarizing Data

Mean, Median, Variance

- To calculate mean, median, and variance, R uses functions

```
mean(gss$sibs)
```

```
## [1] 1.213564
```

```
median(gss$sibs)
```

```
## [1] 3
```

```
var(gss$sibs)
```

```
## [1] 277.3183
```

- Note that the denominator of variance is `n-1` by default in R
  $var(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

```
var(c(1,2,3))
```

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

Combine data subsetting and summarizing (base R)

- Some values in the variable `sibs` are not real observations
- Negative values are missingness
- How do we calculate the mean value of `sibs` without negative `sibs` values?

Combine data subsetting and summarizing (base R)

- Some values in the variable `sibs` are not real observations
- Negative values are missingness
- How do we calculate the mean value of `sibs` without negative `sibs` values?

```
mean(gss[gss$sibs>=0,"sibs"])
```

```
## [1] 3.863355
```

Exercise

- How do we calculate the variance of `sibs` in year 2018 only?

Combine data subsetting and summarizing (dplyr)

- We can also use `dplyr` to subset and summarize data
- How do we calculate the mean value of `sibs` without negative `sibs` values?

```
library(dplyr)
gss %>% filter(sibs>0) %>% summarize(sibs_mean = mean(sibs))
```

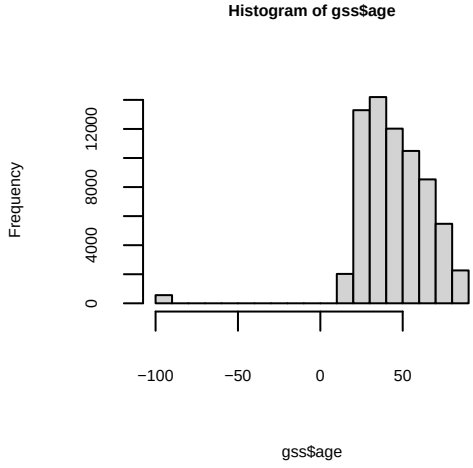
```
##   sibs_mean
## 1  4.066925
```

- `summarize()` follows the format: `summarize(your_summarize_name = function(variable))`

Histogram plot

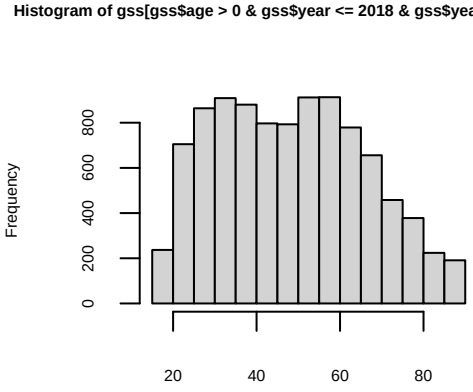
- Histograms describe the distribution of **numeric** data

```
hist(gss$age)
```



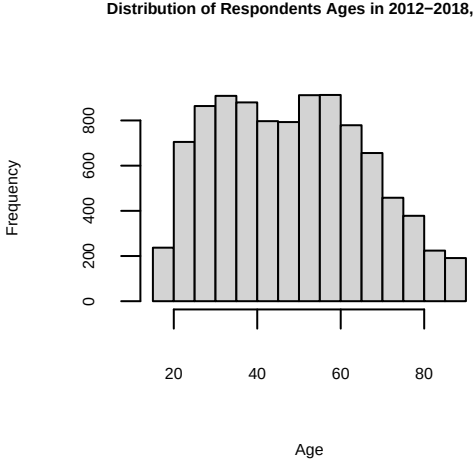
- Again, we want to drop negative values of age
- We may also want to see the distribution for some years but not others

```
hist(gss[gss$age>0 & gss$year<=2018 & gss$year>=2012,"age"])
```



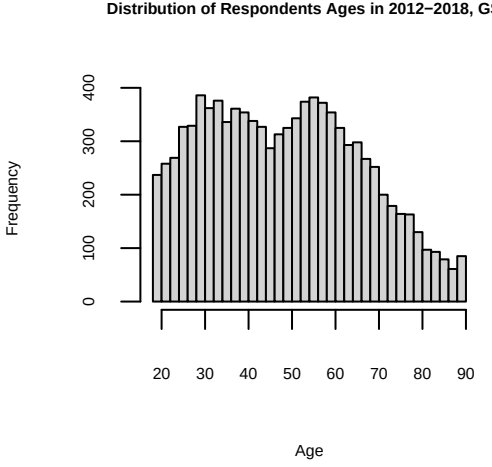
- To change plot titles and x-axis label

```
hist(gss[gss$age>0 & gss$year<=2018 & gss$year>=2012,"age"],
     main = "Distribution of Respondents Ages in 2012-2018, GSS",
     xlab = "Age")
```



- To control the number of `bins`

```
hist(gss[gss$age>0 & gss$year<=2018 & gss$year>=2012,"age"],
     main = "Distribution of Respondents Ages in 2012-2018, GSS",
     xlab = "Age",
     breaks = 30)
```



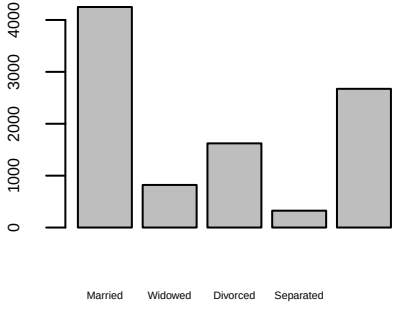
Barplot

- Barplots describe the distribution of **categorical** data

```
## create a count summary in each category by function `table`
counts <- table(gss[gss$age>0 & gss$year<=2018 &
           gss$year>=2012 & gss$marital>=0, "marital"])
counts
```

```
##
##    1    2    3    4    5
## 4250  821 1622  325 2673
```

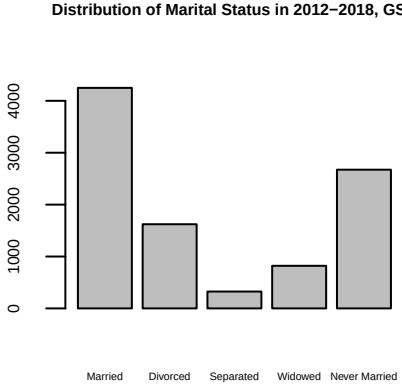
```
barplot(counts, main="Distribution of Marital Status in 2012-2018, GSS",
   xlab="Marital Status",
   names.arg=c("Married", "Widowed", "Divorced", "Separated",
               "Never Married"),
   cex.lab=0.5, cex.axis=0.5, cex.main=0.5, cex.sub=0.5, cex.names=0.32)
```



- We can also change the order of the categories by first `factorizing` the variable with manual `levels`

```
gss$marital <- factor(gss$marital,levels = c(1,3,4,2,5))
counts <- table(gss[gss$age>0 & gss$year<=2018 &
           gss$year>=2012, "marital"])
```

```
barplot(counts, main="Distribution of Marital Status in 2012-2018, GSS",
   xlab="Marital Status",
   names.arg=c("Married", "Divorced", "Separated", "Widowed", "Never Married"),
   cex.lab=0.5, cex.axis=0.5, cex.main=0.5, cex.sub=0.5, cex.names=0.35)
```



Export the data

- `write.csv` save (or export) the dataframe in `.csv` format.

```
write.csv(gss, "cleaned_gss.csv")
```

- We will cover more advanced data summarization next week, including how we can summarize data by group
- We will also cover package `ggplot2` next week, a powerful and more flexible tool in data visualization, including what we introduced today

- Any questions from textbook or lectures?