

STATS 20: Sections 2.4 & 2.8

STATS 20 Team

Fall 2026

Today's Goal

For this discussion section, you are tasked to explore some more operations and commands using vectors and matrices.

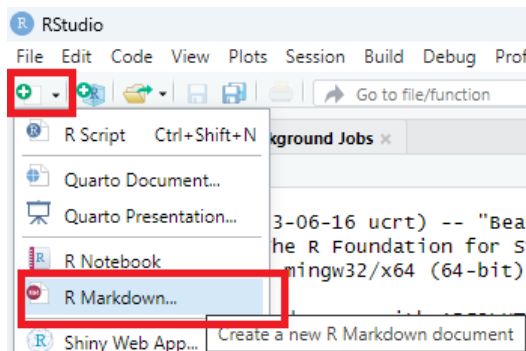
- ▶ Ensure specific packages like tidyverse is installed.
- ▶ Explore special values like NA.
- ▶ Use relational operators to compare values and create logical vectors.
- ▶ Use logical vectors as vector indices for subsetting and assignment.
- ▶ See the applications through an imported dataset.

When done, you'll have to knit your RMarkdown File and make sure you have a .pdf copy to submit.

- ▶ This .pdf will be in preparation for your Homework 4 assignment.

The Markdown

Make sure to open RStudio and create a new R Markdown File. Make sure your name, course, and date is on it.



Other Forms of Data File Formats

- ▶ As mentioned in lecture, R can be used to read various file formats in order to perform our desired data analyses.
- ▶ We focused on reading data that comes from `.txt` or `.csv` files, but R also has an inherent filing system called the `.RDS` file.
 - ▶ This system stores a single R object, preserving its exact structure and attributes.
 - ▶ It utilizes a *binary* file system, which reads and saves files incredibly fast, but is for machine readability only (and not decipherable for humans).
- ▶ To **read** an `.RDS` file, you use the command `readRDS(file)` and to **save** `.RDS` file, you use the command `saveRDS(object, file)`, where
 - ▶ `object` is the R object you wish to save.
 - ▶ `file` is the file path or name for the `.RDS` file (Example: `file="somefile.RDS"`)

Working directory time!

Make sure you can set your working directory, especially to import datasets into R. Using the techniques from lecture, create an R Chunk to:

- (1) set your working directory.
- (2) use the `readRDS()` command in class to upload the new dataset provided: `automobile_26.RDS`. Name the file `auto`.

```
#Set WD  
setwd("Your_Working_Directory")  
#Import Data  
auto<-readRDS(file='automobile_26.RDS')
```

Summary Data

- (3) Confirm you imported everything correctly by getting the *summary* values of this variable, using the command lines `head(auto)` and `summary(auto)`.

```
head(auto)
summary(auto) # Summary of data and all four variables
```

The auto Dataset

Here, the auto dataset contains a collection of vehicles with the following variables.

- ▶ *name*: Car name.
- ▶ *price*: Price in US Dollars.
- ▶ *mpg*: Miles per gallon achieved for vehicle (fuel efficiency).
- ▶ *weight*: Weight in Pounds.
- ▶ *length*: Length in inches.
- ▶ *foreign*: Whether the car is foreign-made (Yes/No).

Comparing to .csv file

- ▶ Suppose we want to see how this file compares to that from an imported .csv file.
- ▶ Use the `read.csv()` command in class to upload the new dataset provided: `automobile_26.csv`. Name the file `auto2`. We'll designate the entries that are blank with the argument `na.strings=""`.

```
setwd("Your_Working_Directory")  
auto2<-read.csv(file='automobile_26.csv', na.strings="")
```

- ▶ Are these two datasets equal? Check using the `identical()` and `all.equal()` functions. If not equal, visually inspect the data in the global environment and see if there are any clear differences that exists between these two datasets.

```
identical(auto,auto2)  
all.equal(auto,auto2)
```

- ▶ These aspects are worthwhile to keep in mind when dealing with the intricate nature of data storage.

Adding Variables via the \$ Operator

- ▶ Often, we wish to transform a variable within a dataset.
- ▶ We can do so through the \$ operation, creating a new variable as our object and assigning it to a transformation of a variable. Such a process would produce the transformation element-wise.
- ▶ Let's do so by looking at our auto dataset, taking our price variable, transforming it to a price_thousands variable by dividing by 1000.

```
#Getting Changing to Thousands  
auto$price_thousands <- auto$price/1000
```

Imprecision Brought on by Transformation

- ▶ One main issue with these transformations (brought on by operations) is that we run the risk of inducing error based on the floating point arithmetic, even when dividing by something as basic as 1000.
- ▶ Check out observation 4 of the `price_thousands` variable, first just as what R naturally outputs it and then print it out to twenty places after the decimal.

```
#Getting Observation 4 of price_thousands
```

```
auto$price_thousands[4]
```

```
#But... what if we expand it out a bunch of digits?
```

```
print(auto$price_thousands[4], digits=20) #Sneaky, sneaky, sneaky
```

- ▶ Keep this imprecision in mind with our computations.

Using operations with NA values.

- ▶ Last time, we noted how the NA values will impact trying to perform computations like the sum, mean, and standard deviation.
- ▶ Try to run these functions with the NA values still in there. Do any of them work?

```
# Note that there are six NA values  
summary(auto$weight)  
#Using mean & sd without removing NA values  
mean(auto$weight)  
sd(auto$weight)
```

Fixing the NA Value Issue

- ▶ One option to deal with NA values is to use the `na.omit()` function, but we may wish to not remove the NA values altogether. Instead, we will use the argument `na.rm=TRUE` to our functions to get these values.
- ▶ Let's try it out with the `weight` variable, which, from our `summary()` function, has some missing values. We'll find the sample size, mean, and standard deviation using the `sum(!is.na())`, `mean()` and `sd()` functions.

```
#Using sum(!is.na())  
sum(!is.na(auto$weight))  
#Using mean & sd with na.rm=TRUE  
mean(auto$weight, na.rm=TRUE)  
sd(auto$weight, na.rm=T) #T is shorthand for TRUE
```

Transforming Categorical Vectors

- ▶ Now, we'll transform the categorical variable (character variable) `foreign`.
- ▶ To find the proportion of cars that were foreign-made, we can check to see whether the observation recorded was "Yes". If "No", it will report as `FALSE`, and `NA` will remain the same.
- ▶ Create a new variable `foreign_YES`, which performs the relational operation on whether the `foreign` variable is "Yes".
- ▶ Likewise, we can create an *indicator* (or *dummy*) variable that outputs 1 for "Yes", 0 for "otherwise", when the observation isn't missing, using the `as.numeric()` command. Call this variable `foreign_YES_ind`.
- ▶ Finally, we can obtain the proportion of observation which were foreign-born by taking the `mean()` of either variable. Be sure to include the argument `na.rm=T`.

R Code for Last Slide's Steps

```
#Logical vector "Is the observation equal to 'Yes'?"  
auto$foreign_YES <- auto$foreign == "Yes"  
  
#Numeric Indicator Variable  
auto$foreign_YES_ind <- as.numeric(auto$foreign_YES)  
  
#Can find proportion either way.  
mean(auto$foreign_YES, na.rm=T)  
mean(auto$foreign_YES_ind, na.rm=T)
```

Outlier Detection

- ▶ In your introductory statistics course, you were taught how to evaluate data to determine whether any observations are suspected outliers.
- ▶ One of the first detection methods comes from comparing values via formulas from the five number summary.
- ▶ In introductory statistics, we learned that if we obtaining the interquartile range (IQR) by computing $Q3 - Q1$, an observation is a suspected outlier if it is $1.5IQR$ above $Q3$ or $1.5IQR$ below $Q1$.
- ▶ We'll use this criteria to see whether there are any suspected outliers for the variable mpg.

#Use the summary function to see what the first & third quartiles are
`summary(auto$mpg)`

- ▶ Based on our summary values, it looks like $Q1$ is 18 and $Q3$ is 25.

Determining Outliers

- ▶ Now, we'll use this condition to detect which observations could be suspected outliers and what the value(/s) is(/are).
- ▶ We'll use the “or” operation along with our conditional statements to detect which observations fit the situation.

```
#Define values and IQR
```

```
Q1=18
```

```
Q3=25
```

```
IQR=Q3-Q1
```

```
# Recall suspected outliers
```

```
#Which Observation is a Suspected Outlier
```

```
which(auto$mpg < Q1-1.5*IQR | auto$mpg > Q3+1.5*IQR)
```

Viewing the Outlier Values

- ▶ Then, we'll use the indices to extract the elements/observation of interest. What happens if we don't include `which()` inside the brackets? []

#Value of the Suspected Outlier

```
auto$mpg[which(auto$mpg < Q1-1.5*IQR | auto$mpg > Q3+1.5*IQR)]
```

#Without Which

```
auto$mpg[(auto$mpg < Q1-1.5*IQR | auto$mpg > Q3+1.5*IQR)]
```

Try It Yourself!

- ▶ Another way to detect suspected outliers is to see if an observation is more than three standard deviations away from the mean.
- ▶ Alternatively, we can take the *z-score* (standardized score) of each observation and see if it is above 3 or below -3 (or, if the absolute value is above 3). Using sample data, $Z_x = \frac{x - \bar{x}}{s_x}$.
- ▶ Now try it yourself! Do so through the following steps:
 - (1) Create a new variable, called `mpg_Z` in the `auto` dataset, which computes the *z-score* of your observations for `mpg`. Be sure to remove NA values when computing the mean and standard deviation.
 - (2) Find the observations which are suspect outliers, using `which()`. Do they match with the IQR method?
 - (3) Find the value of the observations.

That's It for Today!

You did the things!

All that is left is to work on converting your Markdown file to a pdf file.

- ▶ Either do it directly, or convert it to an html and print to a pdf (see lecture slides for directions). **Make sure your name and UCLA Student ID is on the file somewhere**
- ▶ Upload the either the html or pdf file only (no .Rmd file needed) in the *Discussions & Participation* Section of the *Modules* tab in BruinLearn. Go to *Discussion 10/16* and upload there to get your 1 point (Exit Slip).

Enjoy the weekend!