Semester Project

Kristina Arevalo

Thu May 20 2021

Contents

1 Overview

In this lab, I will go over what to do when you have collected your own data, or are analyzing a data set that has been previously collected. We will go over how to clean the data (aka Data wrangling), conduct a statistical analyses and create your very own figure that you can use for a manuscript. I hope this lab helps those who are fearful of using R for their own analyses!

2 Prepare for Analysis

First, we want to load the packages we will need, if you don't have these installed, do so first with the 'install.packages()' function and make sure to put the package title with quotation marks.

There are many free and open data sets available in R, we will be using the carData package to access a data set called Salaries. This data set is from the 2008-09 academic salary for Assistant Professors, Associate Professors and Professors in a college in the U.S. The data was collected as part of the on-going effort of the college's administration to monitor salary differences between male and female faculty members.

```
library(tidyr)
library(dplyr)
library(ggplot2)
library(jtools)
library(xtable)
library(kableExtra)
library(carData)
library(summarytools)
```

This is how we will save and access the data but if you had conducted your own study and collected your own data you would download the data as an excel file (I prefer csv format), then you would save the file in the same folder as the R project you will be working in and use the code my.data <- read.csv("XXX.csv")

```
my.data <- carData::Salaries
```

If you haven't finished collecting data and you want to get a head start on cleaning your data, you can use the 'dim()' function, this allows you to continuously import new data and R will adjust to it so you don't have to keep changing your code for variables. Sometimes when you import your data, there are extra rows on the top, things like the question number, the question itself or something random, like in Qualtrics theres a row that says ImportId. I like to keep my data clear and concise so I don't need to keep those rows, with the 'dim()' function you could remove these rows. You would set it up as:

new_data_name <- dataname[starting_rownumber:dim(dataname)[1],]

```
my.data.example<- my.data[3:dim(my.data)[1],]
```

Here is just an example of how it would look. This basically says, lets start at row 3 and extend the dimension of our data set to however many rows there are in the my.data file and start with column 1, as in do not remove any of the columns.

2.1 View Data

The 'head()' function prints the first 6 values of the dataset, this gives you a quick view of the variables you will be working with. You can also use the 'dfSummary()' function from the summarytools package for a quick clean view of the data as well.

```r
my.data %>% head()
```

```
##        rank discipline yrs.since.phd yrs.service  sex salary
## 1      Prof          B            19          18 Male 139750
## 2      Prof          B            20          16 Male 173200
## 3  AsstProf          B             4           3 Male  79750
## 4      Prof          B            45          39 Male 115000
## 5      Prof          B            40          41 Male 141500
## 6 AssocProf          B             6           6 Male  97000
```

```r
dfSummary(my.data)
```

```
## Data Frame Summary
## my.data
## Dimensions: 397 x 6
## Duplicates: 4
##
## ------------------------------------------------------------------------------------------------------------
## No   Variable         Stats / Values                  Freqs (% of Valid)    Graph                 Va
## ---- --------------- ------------------------------- --------------------- --------------------- ---
## 1    rank            1. AsstProf                      67 (16.9%)           III                   39
##      [factor]        2. AssocProf                     64 (16.1%)           III                   (1
##                      3. Prof                         266 (67.0%)           IIIIIIIIIIIII
##
## 2    discipline      1. A                            181 (45.6%)           IIIIIIIII             39
##      [factor]        2. B                            216 (54.4%)           IIIIIIIIII            (1
##
## 3    yrs.since.phd   Mean (sd) : 22.3 (12.9)        53 distinct values      .   :               39
##      [integer]       min < med < max:                                      . : : : . .           (1
##                      1 < 21 < 56                                           : : : : : : : :
##                      IQR (CV) : 20 (0.6)                                   : : : : : : : : :
##                                                                            : : : : : : : : : : .
##
## 4    yrs.service     Mean (sd) : 17.6 (13)           52 distinct values    : .                   39
##      [integer]       min < med < max:                                      : :   .               (1
##                      0 < 16 < 60                                           : : . : .
##                      IQR (CV) : 20 (0.7)                                   : : : : : .
##                                                                            : : : : : : : . .
##
## 5    sex             1. Female                        39 ( 9.8%)           I                     39
##      [factor]        2. Male                         358 (90.2%)           IIIIIIIIIIIIIIIII     (1
##
## 6    salary          Mean (sd) : 113706.5 (30289)    371 distinct values       :                 39
##      [integer]       min < med < max:                                        : :                 (1
##                      57800 < 107300 < 231545                                 : : : .
##                      IQR (CV) : 43185 (0.3)                                . : : : : :
##                                                                            : : : : : : : .
## ------------------------------------------------------------------------------------------------------------
```

3 Clean Data

3.1 Renaming Variables

When you download your data from a web-based survey platform (i.e Qualtrics, google forms, surveymonkey) the way your variables (column names) look may not be easy to interpret. I find it best to rename the columns to make data analysis simple and easy for anyone to understand. Conveniently, if you want to rename something you use the 'rename()' function. The new column name should be on the left side of the equal sign, and the old column name should be on the right side. To permanently save your data with the new names, you must save it as a new data set. Note, you can use the same name (my.data) but as a beginner, it is easier to save it under a new name (my.data1) so if you make a mistake down the line, it will be easier to fix.

```
my.data1 <- my.data %>%
  rename(Gender = sex,
         Years.Worked = yrs.service,
         Salary = salary)
```

3.2 Inspect Data for Problems

The 'class()' function tells you what "type" your variable saved as when it was converted into R. This is important because some analyses won't run if they are not in the right form (numeric/integer = ratio or interval, factor = nominal) To use this function, you want to access the specific variable/column within the data set with the $.

```
class(my.data1$Years.Worked)
```

```
## [1] "integer"
```

```
class(my.data1$Gender)
```

```
## [1] "factor"
```

```
class(my.data1$Salary)
```

```
## [1] "integer"
```

If your variable isn't in the right class you use the as.numeric or as.character to turn the variable you put inside of it into what you'd like it to be. Here we will change from integer to numeric (although I believe that you don't need to do this, but for the sake of the example). Then you check the class again to make sure it worked. note: if you need to turn a factor variable into a numeric variable, you have to use this code: as.numeric(as.character(my.data$my.variable)).

```
my.data1$Salary<- as.numeric(my.data1$Salary)
class(my.data1$Salary)
```

```
## [1] "numeric"
```

If you have a data set of hundreds of people it would take hours to comb through each column for each participant to see their varied responses. It also helps to see if there are any NAs present, the NA represents a missing answer. We can use the 'unique()' function to quickly sum all the different levels of responses in a column.

```r
my.data1$Gender %>% unique()
```

```
## [1] Male   Female
## Levels: Female Male
```

```r
my.data1$Years.Worked %>% unique()
```

```
##  [1] 18 16  3 39 41  6 23 45 20  8  2  1  0 34 36 26 31 30 19  4  9 21 27 38 15
## [26] 28 25 11  5 12 17 14 37  7 10 29 32 22 49 57 24 53 33 40 35 43 44 48 46 51
## [51] 13 60
```

```r
my.data1$Salary%>% unique()
```

```
##   [1] 139750 173200  79750 115000 141500  97000 175000 147765 119250 129000
##  [11] 119800  79800  77700  78000 104800 117150 101000 103450 124750 137000
##  [21]  89565 102580  93904 113068  74830 106294 134885  82379  77000 118223
##  [31] 132261  79916 117256  80225 155750  86373 125196 100938 146500  93418
##  [41] 101299 231545  94384 114778  98193 151768 140096  70768 126621 108875
##  [51]  74692 106639 103760  83900 117704  90215 100135  75044  90304  75243
##  [61] 109785 103613  68404 100522  99418 111512  91412 126320 146856 100131
##  [71]  92391 113398  73266 150480 193000  86100  84240 150743 135585 144640
##  [81]  88825 122960 132825 152708  88400 172272 107008  97032 105128 105631
##  [91] 166024 123683  84000  95611 129676 102235 106689 133217 126933 153303
## [101] 127512  83850 113543  82099  82600  81500 131205 112429  82100  72500
## [111] 104279 105000 120806 148500 117515  73500 115313 124309  97262  62884
## [121]  96614  78162 155500 113278  73000  83001  76840  77500 168635 136000
## [131] 108262 105668  73877 152664 100102 106608  89942 112696 119015  92000
## [141] 156938 144651  95079 128148 111168 103994 118971 113341  88000  95408
## [151] 137167  89516 176500  98510  88795 105890 167284 130664 101210 181257
## [161]  91227 151575  93164 134185 111751  95436 100944 147349 142467 141136
## [171] 100000 150000 134000 103750 107500 106300 153750 180000 133700 122100
## [181]  86250  90000 113600  92700 189409 114500 119700 160400 152500 165000
## [191]  96545 162200 120000  91300 163200  91000 111350 128400 126200 118700
## [201] 145350 146000 105350 109650 119500 170000 145200 107150 129600  87800
## [211] 122400  63900  70000  88175 133900  73300 148750 117555  69700  81700
## [221] 114000  63100  77202  96200  69200 122875 102600 108200  84273  90450
## [231]  91100 101100 128800 204000 109000 102000 132000 116450  83000 140300
## [241]  74000  73800  92550  88600 107550 121200 126000  99000 134800 143940
## [251] 104350  89650 103700 143250 194800  78500  93000 107200 107100 100600
## [261] 136500 103600  57800 155865  88650  81800 115800  85000 150500 174500
## [271] 168500 183800 107300  97150 126300 148800  72300  70700 127100 170500
## [281] 105260 144050  74500 122500 166800  92050 108100  94350 100351 146800
## [291]  84716  71065  67559 134550 135027 104428  95642 126431 161101 162221
## [301]  84500 124714 151650  99247 134778 192253 116518 105450 145098 104542
## [311] 151445  98053 145000 128464 137317 106231 124312 114596 162150 150376
## [321] 107986 142023 128250  80139 144309 186960  93519 142500 138000  83600
## [331] 145028  88709 107309 109954  78785 121946 109646 138771  81285 205500
## [341] 101036 115435 108413 131950 134690  78182 110515 109707 136660 103275
## [351] 103649  74856  77081 150680 104121  75996 172505  86895 125192 114330
## [361] 139219 109305 119450 186023 166605 151292 103106 150564 101738  95329
## [371]  81035
```

3.3 Mutating Variables

Lets say you download your data and the responses are written as "Strongly Agree," "Agree" etc. Or you're using a measure where some of the responses are reverse coded. We can do that in R too! Lets try it with the Gender category. It seems daunting, but we basically create our own function that allows us to replace within certain columns. The best part of this is that you can use this for any data set, just copy the code and punch in your own info.

What you need to change to fit your own data is the data set names, what goes inside the 'vars()' doesn't have to be just one variable you can use (x:xx) to represent how ever many columns you need to change. You can use the column name or the column number.

```
my.data2 <- my.data1 %>%
  mutate_at(vars("Gender"),
            function(x)
              recode(x,"Male" = 0,
                     "Female" = 1))
```

We can also create a composite measure in R too! If you are using something like the CES-D Depression scale, each column would represent a question, but we want to know what each participants actual score on the scale is. We could use the 'rowwise()' and 'mutate()' function to create a new column to represent that. Here lets pretend we want to see if theres a difference between the yrs since phd from yrs worked.

```
my.data3 <- my.data2%>%
  rowwise() %>%
  mutate(diff_btwn_yrs = diff(c_across(4:3)))
```

If the columns you want to combine aren't next to each other, you can also use this style as well but you would need to use the actual column name here.

```
my.data4 <- my.data2 %>%
  rowwise() %>%
  mutate(diff_btwn_yrs = ((yrs.since.phd) - (Years.Worked)))
```

You can compare my.data3 to my.data4 and see that both options work. Note that when you create a compoisite measure you will need to divide by the number of questions and you can do this by just adding "/#)" to the end but within the mutate function like:
mutate(diff_btwn_yrs = ((yrs.since.phd) - (Years.Worked))/2)

4 Exploring the Data

4.1 Descriptives

Usually the first step I take is to look at the descripitives, I want to know the means and standard deviations of my variables. I can use an xtable (from the xtable package) to make it APA style as well. Note we are returning to my.data1 because this data is already clean and clear.

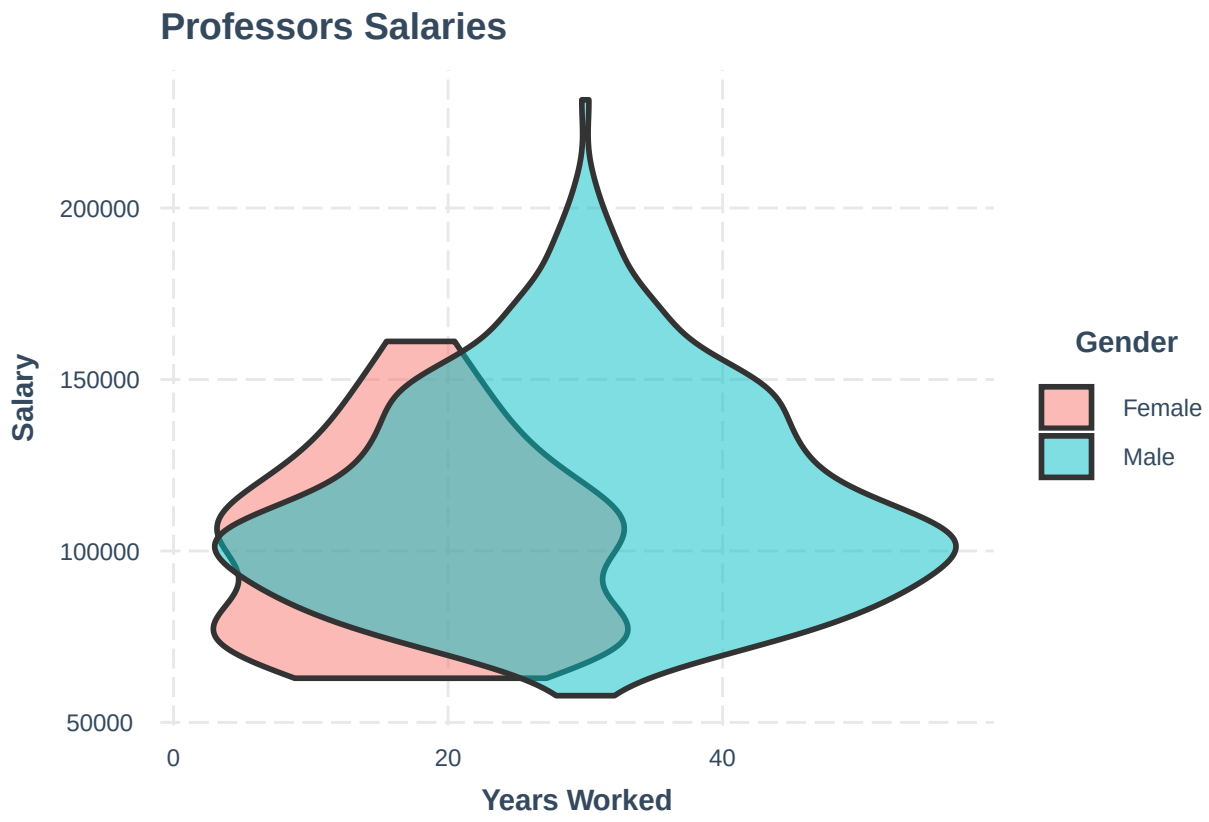
4.2 Visualization Example

This is an example of a visualization we can create. Violin plots are used to look at the distribution and are good for making comparisons to show each group's density curves. Things to note, within the 'geom_violin()'

Table 1: Descriptives

	Gender	Mean.Salary	SD.Salary	Mean.Years.Worked	SD.Years.Worked
1	Female	101002.41	25952.13	11.56	8.81
2	Male	115090.42	30436.93	18.27	13.23

function, alpha represents the opaqueness within the violin plot, and size represents the thickness of the line around the violin plot.



5 Generalization Problem

Now you can run our own analysis! Yayyy!! Lets use everything we've learned in stats to use the data to create your own hypothesis, run your own analysis, interpret and explain your analysis, and then create a figure that can be used in a manuscript.

5.1 Hypothesis

Based on this data set we can look at some predictor variables, gender and years worked, and the outcome variable is salary. I expect to see a positive linear relationship between salary & years worked and a stronger relationship between males and salary/years worked, i.e I expect men to have a higher salary then women despite the amount of years worked.

5.2 Analyses

Table 2: My Linear Regression

	Est.	2.5%	97.5%	t val.	p	partial.r	part.r
(Intercept)	92356.95	83037.72	101676.17	19.48	0.00		
GenderMale	9071.80	-486.21	18629.81	1.87	0.06	0.09	0.09
Years.Worked	747.61	528.61	966.62	6.71	0.00	0.32	0.32

A linear regression revealed that gender (male or female) and amount of years worked predict a significant amount of your nine-month salary. The p-value here represents the probability of observing a relationship between salary and gender this large or larger due to sampling variability if the null were true. The years you work had a more siginifcant p-value, but gender was also significant. The confidence interval tells us that, with the sample size and with a confidence level of 95%, 95 out of 100 samples will contain the true population level relationship between gender and salary. Because years worked does not contain a CI of 0, I am more certain that the amount of years worked is a better predictor of the salary you earn. Both variables share variance explained both partially and semi-partially, but years worked again, shows more variance explained in the salary earned. Overall, years worked has a stronger relationship, gender also has an effect, but together they do not based on a R2 of 0.12.

Based on this graph, you can see that on the x-axis is the amount of years the professor worked, the y-axis shows how much that professor earns and it is divided into two graphs, female and male professors. As you can see, female professors make less money then their male counterparts for the same amount of years worked! For 20 years of working, the highest salary a female professor in this dataset has made was about 150,000 but for a male professor working for 20 years, the highest pay is almost 200,000! Tragic. However, you can

also see that the majority of the sample was male professors so maybe, lets hope that our higher-earning female professors were just too busy to take part in this study!