Lab 10 - Merging Data

Ramin Jabbarialghanab

November 2, 2017

Using your own dataset (which may include more than one table) carry out the following data cleaning steps. Knit together the PDF document and commit both the Lab 10 RMD file and the PDF document to Git. Push the changes to GitHub so both documents are visible in your public GitHub repository.

1. For your poster project, do you have multiple tables you'd like to join together to create your complete dataset? If so, describe what each table represents.

I have three tables. One is total unemployment rate of Iran for all people, one is for men, and another one is unemployment rate for women. I would join the two table which are based on gender together to be able to compare the data with each other in my analasis. I would know how the rate of unemployment differs based on gender.

2. What is/are your primary key(s)? If you have more than one table in your data, what is/are your foreign key(s)? Do your primary key(s) and foreign key(s) have the same name? If not, what does this mean for the way you need to specify potential data merges?

The primary key in my my data is "Province | Year" and this is same with the foreign keys. The keys are same because my data the first column of my data is the name of province. Just the datasets are seperated based on gender for the same provinces. I would add that I am going to change the name of the column from "Province | Year" to just "Province".

3. If you do not need to merge tables to create your final dataset, create a new dataset from your original dataset with a `grouped_by()` summary of your choice. You will use this separate dataset to complete the following exercises.

I need to merge tables because of this I did not create a new data set with group_by() function.

If you are merging separate tables as part of your data manipulation process, are your keys of the same data type? If not, what are the differences? Figure out the appropriate coercion process(es) and carry out the steps below.

Yes, all are characters. So, there is not difference between the keys or their types.

4. Perform each version of the mutating joins (don't forget to specify the `by` argument) and print the results to the console. Describe what each join did to your datasets and what the resulting data table looks like. For those joining two separate datasets, did any of these joins result in your desired final dataset? Why or why not?

Left Join: Left join augmented the primary table(unemployment men). This function returned a new dataframe with all with all the rows of primary table in their original order and the added to the rows new values in the column which maatches. For my data the function added x and y to the years to made easy to know which value came from which dataset. Since this table created the values seperately by data, that can be used for my purpose.

```
install.packages("dplyr", repos = "http://cran.us.r-project.org")
```

```
## Installing package into '/Users/Ramin/Desktop/Autumn 2017/Statistics 321/unemployment rate/unemployme
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/z9/b9hh4hpj6hl6x9r9dxjctv240000gn/T//RtmpXB33mv/downloaded_packages
```

```
install.packages("tidyr", repos = "http://cran.us.r-project.org")
```

```
## Installing package into '/Users/Ramin/Desktop/Autumn 2017/Statistics 321/unemployment rate/unemployme
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/z9/b9hh4hpj6hl6x9r9dxjctv240000gn/T//RtmpXB33mv/downloaded_packages
```

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readxl)

unemployment_men <- read_excel("~/Desktop/Autumn 2017/Statistics 321/unemployment rate/unemployment_men

unemployment_women <- read_excel("~/Desktop/Autumn 2017/Statistics 321/unemployment rate/unemployment_w

dplyr::left_join(unemployment_men, unemployment_women, by = "Province | Year")
```

```
## # A tibble: 32 x 19
##           `Province | Year` `2001.x`           `2006.x`           `2008.x`
##                       <chr>    <chr>              <chr>              <chr>
##  1                    Total     13.2                 10                9.1
##  2         East Azarbayejan        -                5.3                6.1
##  3         West Azarbayejan        -               10.8               10.7
##  4                  Ardebil        -               11.1 9.8000000000000007
##  5                  Esfahan        -                9.6                  8
##  6                   Alborz        -                  -                  -
##  7                     Ilam        -                 12               11.3
##  8                  Bushehr        - 9.8000000000000007                 10
##  9                   Tehran        -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari        -               11.7               11.3
## # ... with 22 more rows, and 15 more variables: `2009.x` <chr>,
## #   `2010.x` <chr>, `2011.x` <dbl>, `2012.x` <dbl>, `2013.x` <dbl>,
## #   `2014.x` <dbl>, `2001.y` <chr>, `2006.y` <chr>, `2008.y` <chr>,
## #   `2009.y` <chr>, `2010.y` <chr>, `2011.y` <dbl>, `2012.y` <dbl>,
## #   `2013.y` <dbl>, `2014.y` <dbl>
```

right join: This function reats the unemployment_women dataset as the primary dataset. As a result it returns a dataframe that contains all of the rows from the unemployment women augmented with the information from same unemployment_men. Since my columns and the primary and foreign keys are same the result of this function is the same with the left_join function. Again, R added to x and y to year to indicate make easy to distingusih the source of data.

```
right_join(unemployment_men, unemployment_women, by = "Province | Year")
```

```
## # A tibble: 32 x 19
##           `Province | Year` `2001.x`           `2006.x`           `2008.x`
##                       <chr>    <chr>              <chr>              <chr>
##  1                    Total     13.2                 10                9.1
##  2         East Azarbayejan        -                5.3                6.1
##  3         West Azarbayejan        -               10.8               10.7
##  4                  Ardebil        -               11.1 9.8000000000000007
##  5                  Esfahan        -                9.6                  8
##  6                   Alborz        -                  -                  -
##  7                     Ilam        -                 12               11.3
##  8                  Bushehr        - 9.8000000000000007                 10
##  9                   Tehran        -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari        -               11.7               11.3
## # ... with 22 more rows, and 15 more variables: `2009.x` <chr>,
## #   `2010.x` <chr>, `2011.x` <dbl>, `2012.x` <dbl>, `2013.x` <dbl>,
## #   `2014.x` <dbl>, `2001.y` <chr>, `2006.y` <chr>, `2008.y` <chr>,
## #   `2009.y` <chr>, `2010.y` <chr>, `2011.y` <dbl>, `2012.y` <dbl>,
## #   `2013.y` <dbl>, `2014.y` <dbl>
```

inner join: This function returns rows from my first dataset, unemployment_men that have a match in the second dataset, unemployment_women. Since the rows or the name of provinces are same, the new outcome contains all the provinces and the values for both datasets. So, the new dataset still is same with previous outcomes and again can be used for my purpose of comparing the rate of unemployment rate chronolically for both genders(gender in the data is defined binary).

```
inner_join(unemployment_men, unemployment_women, by = "Province | Year")
```

```
## # A tibble: 32 x 19
##           `Province | Year` `2001.x`           `2006.x`           `2008.x`
##                       <chr>    <chr>              <chr>              <chr>
##  1                    Total     13.2                 10                9.1
##  2         East Azarbayejan        -                5.3                6.1
##  3         West Azarbayejan        -               10.8               10.7
##  4                  Ardebil        -               11.1 9.8000000000000007
##  5                  Esfahan        -                9.6                  8
##  6                   Alborz        -                  -                  -
##  7                     Ilam        -                 12               11.3
##  8                  Bushehr        - 9.8000000000000007                 10
##  9                   Tehran        -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari        -               11.7               11.3
## # ... with 22 more rows, and 15 more variables: `2009.x` <chr>,
## #   `2010.x` <chr>, `2011.x` <dbl>, `2012.x` <dbl>, `2013.x` <dbl>,
## #   `2014.x` <dbl>, `2001.y` <chr>, `2006.y` <chr>, `2008.y` <chr>,
## #   `2009.y` <chr>, `2010.y` <chr>, `2011.y` <dbl>, `2012.y` <dbl>,
## #   `2013.y` <dbl>, `2014.y` <dbl>
```

full join: This function returns every row in either unemployment_men and unemployment_women datasets. The outcoem is same with previous functions since the rows or the name of provinces are same, the new outcome contains all the provinces and the values for both datasets. So, the new dataset still is same with previous outcomes and again can be used for my purpose.

```
full_join(unemployment_men, unemployment_women, by = "Province | Year")
```

```
## # A tibble: 32 x 19
```

```
##           `Province | Year` `2001.x`           `2006.x`           `2008.x`
##                        <chr>    <chr>              <chr>              <chr>
##  1                    Total     13.2                 10                9.1
##  2         East Azarbayejan        -                5.3                6.1
##  3         West Azarbayejan        -               10.8               10.7
##  4                  Ardebil        -               11.1 9.8000000000000007
##  5                  Esfahan        -                9.6                  8
##  6                   Alborz        -                  -                  -
##  7                     Ilam        -                 12               11.3
##  8                  Bushehr        - 9.8000000000000007                 10
##  9                   Tehran        -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari        -               11.7               11.3
## # ... with 22 more rows, and 15 more variables: `2009.x` <chr>,
## #   `2010.x` <chr>, `2011.x` <dbl>, `2012.x` <dbl>, `2013.x` <dbl>,
## #   `2014.x` <dbl>, `2001.y` <chr>, `2006.y` <chr>, `2008.y` <chr>,
## #   `2009.y` <chr>, `2010.y` <chr>, `2011.y` <dbl>, `2012.y` <dbl>,
## #   `2013.y` <dbl>, `2014.y` <dbl>
```

5. Do the same thing with the filtering joins. What was the result? Give an example of a case in which a `semi_join()` or an `anti_join()` might be used with your primary dataset

Semi join: This function give a copy of my first dataset (unemployment_men) that has been filtered to just the rows that have a match in my secondary dataset, unemployment_women. Since my rows are same this outcome gives all the all rows in my first dataset. If I had a big data for my both current datasets, semi join would be helpful to find the maths in the both dataset and comparing the umeployment rate to each other.

```
semi_join(unemployment_men, unemployment_women, by = "Province | Year")
```

```
## # A tibble: 32 x 10
##           `Province | Year` `2001`            `2006`             `2008`
##                        <chr>  <chr>             <chr>              <chr>
##  1                    Total   13.2                10                9.1
##  2         East Azarbayejan      -               5.3                6.1
##  3         West Azarbayejan      -              10.8               10.7
##  4                  Ardebil      -              11.1 9.8000000000000007
##  5                  Esfahan      -               9.6                  8
##  6                   Alborz      -                 -                  -
##  7                     Ilam      -                12               11.3
##  8                  Bushehr      - 9.8000000000000007                10
##  9                   Tehran      -              10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari      -              11.7               11.3
## # ... with 22 more rows, and 6 more variables: `2009` <chr>, `2010` <chr>,
## #   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>
```

anti join: This function shows rows in my fist dataset, unemployment men, which do not have math in the second dataset, unemployment_women. Since my rows are same for both, this function did not give any row. This function can be used for finding any misspelling in my datasets.

```
anti_join(unemployment_men, unemployment_women, by = "Province | Year")
```

```
## # A tibble: 0 x 10
## # ... with 10 variables: Province | Year <chr>, 2001 <chr>, 2006 <chr>,
## #   2008 <chr>, 2009 <chr>, 2010 <chr>, 2011 <dbl>, 2012 <dbl>,
## #   2013 <dbl>, 2014 <dbl>
```

6. What happens when you apply the set operations joins to your tables? Are these functions useful for you for this project? Explain why or why not. If not, give an example in which one of them might be

usefully applied to your data.

For the semi join the function gives all the rows of my first dataset, unemployment_men, and anti_join gives no row for me. I dod not think that in the abow I explained how they can be helpful, for my project, to find any possible misspelling the of the name of provinces, the anti_join function can be used.

7. If you have any reason to compare tables, apply `setequal()` below. What were the results?

IfI would now that the rate of unemployment for both genders in the same province is equal, I can use the function. I tried and the outcome was false for all rows, so there is no province that the rate of unemployment is same.

```
setequal(unemployment_women, unemployment_men)
```

```
## FALSE: Rows in x but not y: 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22[...]. Rows in y but not x: 31
```

8. What is the purpose of binding data and why might you need to take extra precaution when carrying out this specific form of data merging? If your data requires any binding, carry out the steps below and describe what was accomplished by your merge.

The purpose of binding data is creating a new dataset from two datasets that contain the exact same columns and exact same rows and all are in the same order. My datasets' columns and rows are same and in the same order, so I can carry out binding for both columns and rows. Sometimes binding could be risky. For instance, since binding columns function doesn't do any work to match rows in the second dataset to the first, instead it assumes that the rows have already been placed in a matching order, so we need to be careful in usinf bind_cols because the row could be not matched and newly added columns maybe belond to another row or data.

Binding rows

```
bind_rows(unemployment_men, unemployment_women)
```

```
## # A tibble: 64 x 10
##           `Province | Year` `2001`             `2006`             `2008`
##                        <chr>  <chr>              <chr>              <chr>
##  1                    Total   13.2                 10                9.1
##  2         East Azarbayejan      -                5.3                6.1
##  3         West Azarbayejan      -               10.8               10.7
##  4                  Ardebil      -               11.1 9.8000000000000007
##  5                  Esfahan      -                9.6                  8
##  6                   Alborz      -                  -                  -
##  7                     Ilam      -                 12               11.3
##  8                  Bushehr      - 9.8000000000000007                 10
##  9                   Tehran      -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari      -               11.7               11.3
## # ... with 54 more rows, and 6 more variables: `2009` <chr>, `2010` <chr>,
## #   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>
```

Binding columns:

```
bind_cols(unemployment_men, unemployment_women)
```

```
## # A tibble: 32 x 20
##           `Province | Year` `2001`             `2006`             `2008`
##                        <chr>  <chr>              <chr>              <chr>
##  1                    Total   13.2                 10                9.1
##  2         East Azarbayejan      -                5.3                6.1
##  3         West Azarbayejan      -               10.8               10.7
##  4                  Ardebil      -               11.1 9.8000000000000007
##  5                  Esfahan      -                9.6                  8
```

```
##  6                   Alborz      -                  -                  -
##  7                     Ilam      -                 12               11.3
##  8                  Bushehr      - 9.8000000000000007                 10
##  9                   Tehran      -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari      -               11.7               11.3
## # ... with 22 more rows, and 16 more variables: `2009` <chr>,
## #   `2010` <chr>, `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>,
## #   `Province | Year1` <chr>, `20011` <chr>, `20061` <chr>, `20081` <chr>,
## #   `20091` <chr>, `20101` <chr>, `20111` <dbl>, `20121` <dbl>,
## #   `20131` <dbl>, `20141` <dbl>
```

9. Do you need to merge multiple tables together using the same type of merge? If so, utilize the `reduce()` function from the `purrr` package to carry out the appropriate merge below.

I have three tables, I will use two of them. For practice, I am going to merge tables here.

```
install.packages("purrr", repos = "http://cran.us.r-project.org")
```

```
## Installing package into '/Users/Ramin/Desktop/Autumn 2017/Statistics 321/unemployment rate/unemployme
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/z9/b9hh4hpj6hl6x9r9dxjctv240000gn/T//RtmpXB33mv/downloaded_packages
```

```
library(purrr)
unemployment_total <- read_excel("~/Desktop/Autumn 2017/Statistics 321/unemployment rate/unemployment_t
list(unemployment_men, unemployment_women, unemployment_total)
```

```
## [[1]]
## # A tibble: 32 x 10
##           `Province | Year` `2001`             `2006`             `2008`
##                       <chr>  <chr>               <chr>              <chr>
##  1                    Total   13.2                  10                9.1
##  2         East Azarbayejan      -                 5.3                6.1
##  3         West Azarbayejan      -                10.8               10.7
##  4                  Ardebil      -                11.1 9.8000000000000007
##  5                  Esfahan      -                 9.6                  8
##  6                   Alborz      -                   -                  -
##  7                     Ilam      -                  12               11.3
##  8                  Bushehr      - 9.8000000000000007                 10
##  9                   Tehran      -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari      -               11.7               11.3
## # ... with 22 more rows, and 6 more variables: `2009` <chr>, `2010` <chr>,
## #   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>
##
## [[2]]
## # A tibble: 32 x 10
##           `Province | Year`             `2001`             `2006`
##                       <chr>              <chr>              <chr>
##  1                    Total 19.899999999999999               16.2
##  2         East Azarbayejan                  -                5.4
##  3         West Azarbayejan                  -                7.3
##  4                  Ardebil                  -               11.4
##  5                  Esfahan                  -                 16
##  6                   Alborz                  -                  -
##  7                     Ilam                  -               18.8
```

```
##  8                 Bushehr                  -               14.5
##  9                  Tehran                  -                 24
## 10 Chaharmahal & Bakhtiyari                 - 16.399999999999999
## # ... with 22 more rows, and 7 more variables: `2008` <chr>, `2009` <chr>,
## #   `2010` <chr>, `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>
##
## [[3]]
## # A tibble: 32 x 10
##           `Province | Year`             `2001` `2006` `2008` `2009` `2010`
##                        <chr>              <chr>  <chr>  <chr>  <chr>  <chr>
##  1                  Total               14.2   11.3   10.4   11.9   13.5
##  2         East Azarbayejan              6.7    5.3    6.8     10   11.7
##  3         West Azarbayejan             10.6     10   10.3   10.7   12.4
##  4                 Ardebil              11.6   11.1    9.9     12   14.2
##  5                 Esfahan              13.1     11    9.4     12   15.3
##  6                  Alborz                 -      -      -      -      -
##  7                    Ilam 17.100000000000001   13.6   14.6   12.6   15.8
##  8                 Bushehr                12   10.5   10.7   11.7   13.3
##  9                  Tehran              12.2     13     11   11.9   14.2
## 10 Chaharmahal & Bakhtiyari              12.4   12.5   14.1      7   13.6
## # ... with 22 more rows, and 4 more variables: `2011` <dbl>, `2012` <dbl>,
## #   `2013` <dbl>, `2014` <dbl>
```

```
tables <- list(unemployment_men, unemployment_women, unemployment_total)
reduce(tables, left_join, by = "Province | Year")
```

```
## # A tibble: 32 x 28
##           `Province | Year` `2001.x`           `2006.x`            `2008.x`
##                        <chr>    <chr>              <chr>               <chr>
##  1                  Total     13.2                 10                 9.1
##  2         East Azarbayejan        -                5.3                6.1
##  3         West Azarbayejan        -               10.8               10.7
##  4                 Ardebil        -               11.1 9.8000000000000007
##  5                 Esfahan        -                9.6                  8
##  6                  Alborz        -                  -                  -
##  7                    Ilam        -                 12               11.3
##  8                 Bushehr        - 9.8000000000000007                 10
##  9                  Tehran        -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari        -               11.7               11.3
## # ... with 22 more rows, and 24 more variables: `2009.x` <chr>,
## #   `2010.x` <chr>, `2011.x` <dbl>, `2012.x` <dbl>, `2013.x` <dbl>,
## #   `2014.x` <dbl>, `2001.y` <chr>, `2006.y` <chr>, `2008.y` <chr>,
## #   `2009.y` <chr>, `2010.y` <chr>, `2011.y` <dbl>, `2012.y` <dbl>,
## #   `2013.y` <dbl>, `2014.y` <dbl>, `2001` <chr>, `2006` <chr>,
## #   `2008` <chr>, `2009` <chr>, `2010` <chr>, `2011` <dbl>, `2012` <dbl>,
## #   `2013` <dbl>, `2014` <dbl>
```

10. Are there any other steps you need to carry out to further clean, transform, or merge your data into one, final, tidy dataset? If so, describe what they are and carry them out below.

Binding row, assiging number for genders, and deleting 2001 because has not data.

```
bind_rows(unemployment_men, unemployment_women)
```

```
## # A tibble: 64 x 10
##           `Province | Year` `2001`             `2006`              `2008`
```

```
##                       <chr>  <chr>              <chr>              <chr>
##  1                    Total   13.2                 10                9.1
##  2         East Azarbayejan      -                5.3                6.1
##  3         West Azarbayejan      -               10.8               10.7
##  4                  Ardebil      -               11.1 9.8000000000000007
##  5                  Esfahan      -                9.6                  8
##  6                   Alborz      -                  -                  -
##  7                     Ilam      -                 12               11.3
##  8                  Bushehr      - 9.8000000000000007               10
##  9                   Tehran      -               10.9 9.3000000000000007
## 10 Chaharmahal & Bakhtiyari      -               11.7               11.3
## # ... with 54 more rows, and 6 more variables: `2009` <chr>, `2010` <chr>,
## #   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>
```

```
bind_rows(men = unemployment_men, women = unemployment_women, .id = "gender")
```

```
## # A tibble: 64 x 11
##    gender        `Province | Year` `2001`             `2006`
##     <chr>                    <chr>  <chr>              <chr>
##  1    men                    Total   13.2                 10
##  2    men         East Azarbayejan      -                5.3
##  3    men         West Azarbayejan      -               10.8
##  4    men                  Ardebil      -               11.1
##  5    men                  Esfahan      -                9.6
##  6    men                   Alborz      -                  -
##  7    men                     Ilam      -                 12
##  8    men                  Bushehr      - 9.8000000000000007
##  9    men                   Tehran      -               10.9
## 10    men Chaharmahal & Bakhtiyari      -               11.7
## # ... with 54 more rows, and 7 more variables: `2008` <chr>, `2009` <chr>,
## #   `2010` <chr>, `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>
```