Introduction to dplyr

POL-501

2024-09-16

Table of Contents

# 1 Introduction

This document will introduce you to the dplyr package in R, which is part of the tidyverse collection of packages designed for data science. The dplyr package is particularly powerful for data manipulation, providing an intuitive syntax and a set of versatile functions for managing data frames.

To illustrate the capabilities of dplyr, we will use data from Pew’s American Trends Panel (ATP) Wave 117.

We have obtained the variables’ coding (what each numeric value represents) from the PDF files ‘ATP W117 questionnaire’ and ‘Codebook-and-instructions-for-working-with-ATP-data’

### 1.0.1 Loading Necessary Libraries

Before we begin, ensure you have the required packages installed and loaded. If not, you can install them using install.packages().

# Install libraries if necessary  
# install.packages("haven")  
# install.packages("dplyr")  
  
# Load libraries  
library(haven)   
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

### 1.0.2 Setting the Working Directory

Set the working directory to the folder where your data file is located. Adjust the path accordingly.

# Set the working directory  
setwd('C:\\Users\\Ignacio\\Dropbox\\PhD SBU\\06\_Teaching\\00\_POL-501\\Problem Sets\\PS-1\\W117\_Nov22\\')

### 1.0.3 Loading the Dataset

We’ll load the dataset using the read\_spss() function from the haven package.

# Load the Pew's ATP Wave 117 dataset  
load\_savfile <- read\_spss("ATP W117.sav")

### 1.0.4 Subsetting Relevant Columns

We can use the select() function from dplyr to pick specific columns from the dataset.

# Subset relevant columns  
df\_subset <- load\_savfile %>%  
 select(QKEY, VOTED\_ATPCONG\_W117, VTCOUNT\_OWN\_W117, ATTENDPERSON2\_W117,  
 F\_AGECAT, F\_GENDER, F\_EDUCCAT)  
# View the structure of the subset data  
str(df\_subset)

## tibble [11,377 × 7] (S3: tbl\_df/tbl/data.frame)  
## $ QKEY : num [1:11377] 100314 100363 100598 100637 100803 ...  
## ..- attr(\*, "label")= chr "Unique ID"  
## ..- attr(\*, "format.spss")= chr "F12.0"  
## ..- attr(\*, "display\_width")= int 12  
## $ VOTED\_ATPCONG\_W117: dbl+lbl [1:11377] 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3...  
## ..@ label : chr "VOTED\_ATPCONG\_W117. Which of the following statements best describes you?"  
## ..@ format.spss: chr "F2.0"  
## ..@ labels : Named num [1:4] 1 2 3 99  
## .. ..- attr(\*, "names")= chr [1:4] "I did not vote in the 2022 congressional elections" "I planned to vote but wasn’t able to" "I definitely voted in the 2022 congressional elections" "Refused"  
## $ VTCOUNT\_OWN\_W117 : dbl+lbl [1:11377] 1, 1, NA, NA, 1, 4, 4, 4, 2, 1, 4, 1, 1, ...  
## ..@ label : chr "VTCOUNT\_OWN\_W117. How confident are you that your vote was accurately counted?"  
## ..@ format.spss: chr "F2.0"  
## ..@ labels : Named num [1:5] 1 2 3 4 99  
## .. ..- attr(\*, "names")= chr [1:5] "Very confident" "Somewhat confident" "Not too confident" "Not at all confident" ...  
## $ ATTENDPERSON2\_W117: dbl+lbl [1:11377] 5, 4, 5, 1, 2, 1, 2, 6, 2, 5, 2, 5, 5, 2, 5, 6, 5, 1...  
## ..@ label : chr "ATTENDPERSON2\_W117. In general how often do you attend religious services in person at a church, synagogue, mos"| \_\_truncated\_\_  
## ..@ format.spss: chr "F2.0"  
## ..@ labels : Named num [1:7] 1 2 3 4 5 6 99  
## .. ..- attr(\*, "names")= chr [1:7] "More than once a week" "Once a week" "Once or twice a month" "A few times a year" ...  
## $ F\_AGECAT : dbl+lbl [1:11377] 4, 2, 3, 4, 4, 4, 3, 3, 1, 4, 4, 4, 4, 4, 3, 4, 2, 4...  
## ..@ label : chr "Age category"  
## ..@ format.spss: chr "F8.2"  
## ..@ labels : Named num [1:5] 1 2 3 4 99  
## .. ..- attr(\*, "names")= chr [1:5] "18-29" "30-49" "50-64" "65+" ...  
## $ F\_GENDER : dbl+lbl [1:11377] 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 1, 2, 1, 2, 2, 1...  
## ..@ label : chr "Gender"  
## ..@ format.spss: chr "F8.2"  
## ..@ labels : Named num [1:4] 1 2 3 99  
## .. ..- attr(\*, "names")= chr [1:4] "A man" "A woman" "In some other way" "Refused"  
## $ F\_EDUCCAT : dbl+lbl [1:11377] 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 1, 1, 1, 2, 1, 3, 3...  
## ..@ label : chr "Education level category"  
## ..@ format.spss: chr "F8.2"  
## ..@ labels : Named num [1:4] 1 2 3 99  
## .. ..- attr(\*, "names")= chr [1:4] "College graduate+" "Some College" "H.S. graduate or less" "Refused"

## 1.1 Recoding Categorical Variables

### 1.1.1 Recoding Voting Status

We will use the mutate() and factor() functions to recode the VOTED\_ATPCONG\_W117 variable.

# Recode VOTED\_ATPCONG\_W117 (Voting status)  
df\_subset <- df\_subset %>%  
 mutate(VOTED\_ATPCONG\_W117 = factor(VOTED\_ATPCONG\_W117,  
 levels = c(1, 2, 3),  
 labels = c("Did not vote",   
 "Planned to vote but couldn’t",   
 "Definitely voted")))

### 1.1.2 Recoding Confidence in Vote Count

Similarly, we recode VTCOUNT\_OWN\_W117.

# Recode VTCOUNT\_OWN\_W117 (Confidence in vote count)  
df\_subset <- df\_subset %>%  
 mutate(VTCOUNT\_OWN\_W117 = factor(VTCOUNT\_OWN\_W117,  
 levels = c(1, 2, 3, 4),  
 labels = c("Very confident",   
 "Somewhat confident",   
 "Not too confident",   
 "Not at all confident")))

### 1.1.3 Recoding Religious Service Attendance

We now recode ATTENDPERSON2\_W117.

# Recode ATTENDPERSON2\_W117 (Religious service attendance frequency)  
df\_subset <- df\_subset %>%  
 mutate(ATTENDPERSON2\_W117 = factor(ATTENDPERSON2\_W117,  
 levels = c(1, 2, 3, 4, 5, 6),  
 labels = c("More than once a week",   
 "Once a week",   
 "Once or twice a month",   
 "A few times a year",   
 "Seldom",   
 "Never")))

## 1.2 Recoding Demographic Variables

### 1.2.1 Recoding Age Categories

Recode the F\_AGECAT variable into readable age ranges.

# Recode F\_AGECAT (Age categories)  
df\_subset <- df\_subset %>%  
 mutate(F\_AGECAT = factor(F\_AGECAT,  
 levels = c(1, 2, 3, 4, 99),  
 labels = c("18-29",   
 "30-49",   
 "50-64",   
 "65+",   
 "Refused")))

### 1.2.2 Recoding Gender Categories

Recode the F\_GENDER variable.

# Recode F\_GENDER (Gender categories)  
df\_subset <- df\_subset %>%  
 mutate(F\_GENDER = factor(F\_GENDER,  
 levels = c(1, 2, 3, 99),  
 labels = c("Man",   
 "Woman",   
 "Other",  
 "Refused")))

### 1.2.3 Recoding Educational Attainment

Finally, recode the F\_EDUCCAT variable.

# Recode F\_EDUCCAT (Educational attainment categories)  
df\_subset <- df\_subset %>%  
 mutate(F\_EDUCCAT = factor(F\_EDUCCAT,  
 levels = c(1, 2, 3, 99),  
 labels = c("College graduate+",   
 "Some college",   
 "H.S. graduate or less",   
 "Refused")))

## 1.3 Confirming the Structure

Check the structure and ensure the new labels have been applied correctly.

# Confirm the structure and labels are correctly applied  
str(df\_subset)

## tibble [11,377 × 7] (S3: tbl\_df/tbl/data.frame)  
## $ QKEY : num [1:11377] 100314 100363 100598 100637 100803 ...  
## ..- attr(\*, "label")= chr "Unique ID"  
## ..- attr(\*, "format.spss")= chr "F12.0"  
## ..- attr(\*, "display\_width")= int 12  
## $ VOTED\_ATPCONG\_W117: Factor w/ 3 levels "Did not vote",..: 3 3 2 2 3 3 3 3 3 3 ...  
## $ VTCOUNT\_OWN\_W117 : Factor w/ 4 levels "Very confident",..: 1 1 NA NA 1 4 4 4 2 1 ...  
## $ ATTENDPERSON2\_W117: Factor w/ 6 levels "More than once a week",..: 5 4 5 1 2 1 2 6 2 5 ...  
## $ F\_AGECAT : Factor w/ 5 levels "18-29","30-49",..: 4 2 3 4 4 4 3 3 1 4 ...  
## $ F\_GENDER : Factor w/ 4 levels "Man","Woman",..: 1 2 2 2 1 1 2 1 2 2 ...  
## $ F\_EDUCCAT : Factor w/ 4 levels "College graduate+",..: 2 1 1 2 1 2 2 2 1 2 ...

## 1.4 Viewing the First Few Rows

Optionally, we can view the first few rows to verify the recoding worked as expected.

# View the first few rows to ensure everything looks correct  
head(df\_subset)

## # A tibble: 6 × 7  
## QKEY VOTED\_ATPCONG\_W117 VTCOUNT\_OWN\_W117 ATTENDPERSON2\_W117 F\_AGECAT  
## <dbl> <fct> <fct> <fct> <fct>   
## 1 100314 Definitely voted Very confident Seldom 65+   
## 2 100363 Definitely voted Very confident A few times a year 30-49   
## 3 100598 Planned to vote but could… <NA> Seldom 50-64   
## 4 100637 Planned to vote but could… <NA> More than once a … 65+   
## 5 100803 Definitely voted Very confident Once a week 65+   
## 6 101224 Definitely voted Not at all conf… More than once a … 65+   
## # ℹ 2 more variables: F\_GENDER <fct>, F\_EDUCCAT <fct>

With this, you have successfully recoded categorical variables and formatted demographic variables using dplyr. Understanding these basic yet powerful functions will enable you to manipulate your datasets more effectively in R. This concludes our introduction to dplyr.

# 2 Introduction to dplyr

dplyr is a powerful package for data manipulation in R, providing a range of functions (often referred to as “verbs”) to simplify and streamline data analysis tasks. Key verbs in dplyr include:

* select(): Pick columns by name
* filter(): Pick rows by their values
* mutate(): Create or transform variables
* summarize(): Reduce multiple values down to a single summary
* arrange(): Reorder rows
* group\_by(): Group data by one or more variables

We also make extensive use of the pipe operator (%>%), which allows us to chain together multiple operations in a readable, functional style.

Note that the following is the structure of each operation:

resulting\_df\_name <- df\_used %>%  
 verb1 %>%  
 verb2 %>%  
 verb3

Here are examples of using these verbs with the df\_subset data.

### 2.0.1 Select Columns

To select specific columns, we use the select() function:

# Select specific columns  
selected\_data <- df\_subset %>%  
 select(QKEY, VOTED\_ATPCONG\_W117, F\_AGECAT, F\_GENDER, F\_EDUCCAT)  
  
# View the selected columns  
head(selected\_data)

## # A tibble: 6 × 5  
## QKEY VOTED\_ATPCONG\_W117 F\_AGECAT F\_GENDER F\_EDUCCAT   
## <dbl> <fct> <fct> <fct> <fct>   
## 1 100314 Definitely voted 65+ Man Some college   
## 2 100363 Definitely voted 30-49 Woman College graduate+  
## 3 100598 Planned to vote but couldn’t 50-64 Woman College graduate+  
## 4 100637 Planned to vote but couldn’t 65+ Woman Some college   
## 5 100803 Definitely voted 65+ Man College graduate+  
## 6 101224 Definitely voted 65+ Man Some college

### 2.0.2 Filter Rows

To filter rows based on a condition, we use the filter() function:

# Filter data for respondents who definitely voted  
voted\_data <- df\_subset %>%  
 filter(VOTED\_ATPCONG\_W117 == "Definitely voted")  
  
# View the filtered data  
head(voted\_data)

## # A tibble: 6 × 7  
## QKEY VOTED\_ATPCONG\_W117 VTCOUNT\_OWN\_W117 ATTENDPERSON2\_W117 F\_AGECAT  
## <dbl> <fct> <fct> <fct> <fct>   
## 1 100314 Definitely voted Very confident Seldom 65+   
## 2 100363 Definitely voted Very confident A few times a year 30-49   
## 3 100803 Definitely voted Very confident Once a week 65+   
## 4 101224 Definitely voted Not at all confident More than once a week 65+   
## 5 101437 Definitely voted Not at all confident Once a week 50-64   
## 6 101493 Definitely voted Not at all confident Never 50-64   
## # ℹ 2 more variables: F\_GENDER <fct>, F\_EDUCCAT <fct>

### 2.0.3 Mutate Columns

To create new variables or transform existing ones, we use the mutate() function:

# Create a new variable "Age\_Group" based on F\_AGECAT  
df\_subset <- df\_subset %>%  
 mutate(Age\_Group = case\_when(  
 F\_AGECAT == "18-29" ~ "Young Adult",  
 F\_AGECAT == "30-49" ~ "Adult",  
 F\_AGECAT == "50-64" ~ "Middle Aged",  
 F\_AGECAT == "65+" ~ "Senior"  
 ))  
  
# View the modified dataset  
head(df\_subset)

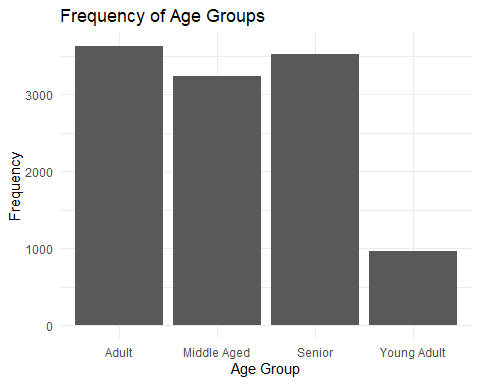
## # A tibble: 6 × 8  
## QKEY VOTED\_ATPCONG\_W117 VTCOUNT\_OWN\_W117 ATTENDPERSON2\_W117 F\_AGECAT  
## <dbl> <fct> <fct> <fct> <fct>   
## 1 100314 Definitely voted Very confident Seldom 65+   
## 2 100363 Definitely voted Very confident A few times a year 30-49   
## 3 100598 Planned to vote but could… <NA> Seldom 50-64   
## 4 100637 Planned to vote but could… <NA> More than once a … 65+   
## 5 100803 Definitely voted Very confident Once a week 65+   
## 6 101224 Definitely voted Not at all conf… More than once a … 65+   
## # ℹ 3 more variables: F\_GENDER <fct>, F\_EDUCCAT <fct>, Age\_Group <chr>

In R, we use == for comparison in conditional statements to check if two values are equal, whereas = is used for assigning values to variables. In the case\_when() function, == is necessary because we are evaluating conditions (e.g., checking if a variable equals a specific value) rather than assigning values. The tilde ~ is used in case\_when() to link each condition with its corresponding output value. It essentially means “if this condition is true, then return this value.” Therefore, == checks the condition, and ~ specifies the result to be assigned when the condition holds true.

# Create a table with the absolute frequency of each age group  
tab\_Age <- as.data.frame(table(df\_subset$Age\_Group)) # Convert the frequency table to a data frame (use the function: as.data.frame())  
  
# Rename the columns of the tab\_Age dataframe  
colnames(tab\_Age) <- c("Age Group", "Freq")  
  
# Print the table  
print(tab\_Age)

## Age Group Freq  
## 1 Adult 3624  
## 2 Middle Aged 3231  
## 3 Senior 3520  
## 4 Young Adult 966

# Load ggplot2 for plotting  
library(ggplot2)  
  
# Create a bar plot using ggplot2  
ggplot(tab\_Age, aes(x = `Age Group`, y = Freq)) +  
 geom\_bar(stat = "identity") +  
 labs(title = "Frequency of Age Groups", x = "Age Group", y = "Frequency") +  
 theme\_minimal()



In ggplot2, the structure of a plot is built in layers, starting with the base ggplot() function, which initializes the plot and specifies the data and aesthetic mappings. In this example, ggplot(tab\_Age, aes(x = Age Group, y = Freq)) tells R that the data comes from the tab\_Age dataframe and maps the “Age Group” variable to the x-axis and “Freq” (frequency) to the y-axis. After initializing the plot, the geom\_bar(stat = "identity") layer is added to create the bars. Here, stat = "identity" ensures that the height of each bar corresponds to the actual frequency values in the dataset. Additional layers like labs() are used to label the plot, providing a title and axes labels. Finally, theme\_minimal() applies a clean, minimal theme to enhance the visual appeal of the plot. Each of these components builds upon the last, allowing for flexible, modular creation of complex visualizations.

### 2.0.4 Summarize Data

To summarize data, often in combination with group\_by(), we use summarize():

# Summarize data to count the number of respondents in each age group  
age\_summary <- df\_subset %>%  
 group\_by(Age\_Group) %>%  
 summarize(Count = n())  
  
# View the summarized data  
age\_summary

## # A tibble: 5 × 2  
## Age\_Group Count  
## <chr> <int>  
## 1 Adult 3624  
## 2 Middle Aged 3231  
## 3 Senior 3520  
## 4 Young Adult 966  
## 5 <NA> 36

In this code, the pipe operator (%>%) is used to chain multiple dplyr functions in a readable sequence. The group\_by(Age\_Group) function groups the dataset (df\_subset) by the Age\_Group column, meaning that subsequent operations will be performed separately for each unique Age\_Group. The summarize() function then creates a summary for each group, in this case calculating the number of rows (or observations) in each Age\_Group using n(), which counts the number of rows in each group. The result is stored in a new column called Count, and the resulting dataset age\_summary will have one row for each unique Age\_Group, with a corresponding count of observations in that group.

### 2.0.5 Arrange Rows

To reorder rows, we use the arrange() function:

# Arrange data by age group  
arranged\_data <- df\_subset %>%  
 arrange(F\_AGECAT)  
  
# View the arranged data  
head(arranged\_data)

## # A tibble: 6 × 8  
## QKEY VOTED\_ATPCONG\_W117 VTCOUNT\_OWN\_W117 ATTENDPERSON2\_W117 F\_AGECAT  
## <dbl> <fct> <fct> <fct> <fct>   
## 1 102198 Definitely voted Somewhat confid… Once a week 18-29   
## 2 274825 Definitely voted Somewhat confid… Seldom 18-29   
## 3 309053 Did not vote <NA> Seldom 18-29   
## 4 601394 Planned to vote but could… <NA> Never 18-29   
## 5 615859 Definitely voted Very confident A few times a year 18-29   
## 6 633392 Planned to vote but could… <NA> Never 18-29   
## # ℹ 3 more variables: F\_GENDER <fct>, F\_EDUCCAT <fct>, Age\_Group <chr>

### 2.0.6 Group By

To group data by one or more variables before summarizing, we use group\_by():

# Group data by gender and summarize age categories  
gender\_age\_summary <- df\_subset %>%  
 group\_by(F\_GENDER, F\_AGECAT) %>%  
 summarize(Count = n())

## `summarise()` has grouped output by 'F\_GENDER'. You can override using the  
## `.groups` argument.

# View the grouped and summarized data  
gender\_age\_summary

## # A tibble: 19 × 3  
## # Groups: F\_GENDER [4]  
## F\_GENDER F\_AGECAT Count  
## <fct> <fct> <int>  
## 1 Man 18-29 344  
## 2 Man 30-49 1523  
## 3 Man 50-64 1409  
## 4 Man 65+ 1714  
## 5 Man Refused 13  
## 6 Woman 18-29 586  
## 7 Woman 30-49 2052  
## 8 Woman 50-64 1796  
## 9 Woman 65+ 1791  
## 10 Woman Refused 22  
## 11 Other 18-29 35  
## 12 Other 30-49 41  
## 13 Other 50-64 13  
## 14 Other 65+ 8  
## 15 Refused 18-29 1  
## 16 Refused 30-49 8  
## 17 Refused 50-64 13  
## 18 Refused 65+ 7  
## 19 Refused Refused 1

### 2.0.7 Example Analysis

Now, let’s perform a more comprehensive example analysis where we combine multiple dplyr functions. Suppose we want to compare voting confidence based on different levels of education.

# Compare voting confidence based on education levels  
edu\_voting\_confidence <- df\_subset %>%  
 group\_by(F\_EDUCCAT, VTCOUNT\_OWN\_W117) %>%  
 summarize(Count = n()) %>%  
 arrange(F\_EDUCCAT, VTCOUNT\_OWN\_W117)

## `summarise()` has grouped output by 'F\_EDUCCAT'. You can override using the  
## `.groups` argument.

# View the analysis results  
edu\_voting\_confidence

## # A tibble: 20 × 3  
## # Groups: F\_EDUCCAT [4]  
## F\_EDUCCAT VTCOUNT\_OWN\_W117 Count  
## <fct> <fct> <int>  
## 1 College graduate+ Very confident 3080  
## 2 College graduate+ Somewhat confident 1149  
## 3 College graduate+ Not too confident 202  
## 4 College graduate+ Not at all confident 92  
## 5 College graduate+ <NA> 1047  
## 6 Some college Very confident 1459  
## 7 Some college Somewhat confident 968  
## 8 Some college Not too confident 231  
## 9 Some college Not at all confident 95  
## 10 Some college <NA> 911  
## 11 H.S. graduate or less Very confident 643  
## 12 H.S. graduate or less Somewhat confident 466  
## 13 H.S. graduate or less Not too confident 127  
## 14 H.S. graduate or less Not at all confident 45  
## 15 H.S. graduate or less <NA> 827  
## 16 Refused Very confident 7  
## 17 Refused Somewhat confident 10  
## 18 Refused Not too confident 3  
## 19 Refused Not at all confident 3  
## 20 Refused <NA> 12

This pipeline will group the data by F\_EDUCCAT (education levels) and VTCOUNT\_OWN\_W117 (voting confidence), summarize the counts in each group, and then arrange the results for easier interpretation.

By understanding these key functions and how to use pipes to combine them, you’ll be able to manipulate and analyze your data more effectively. This concludes our comprehensive introduction to dplyr. Explore these functions further to enhance your data analysis skills in R.