Lab 3 Data Manipulation with dplyr

B. Sosnovski

10/04/2022

```
knitr::opts_chunk$set(echo = TRUE)
```

Data manipulation with dplyr

Data manipulation consists of modifying data to make it easier to read and more organized. We manipulate data for analysis and visualization.

There are different ways to manipulate data in R, and we will use a couple of functions in the package `dplyr`. We are going to look at filtering data, grouping data, and summarizing data.

General goals

In this lab, you will learn the following:

1. How to load some data into R
2. How to see a little bit of how the data is structured
3. How to manipulate data using dplyr

Important info

Data for NYC film permits used in this lab instructions was obtained from the NYC Open Data website. The Film_Permits.csv file for this lab is already conveniently in the lab's folder in your project's directory on Cocalc.com. But you can also find it here: Film_Permits.csv (https://data.cityofnewyork.us/api/views/tg4x-b46p/rows.csv?accessType=DOWNLOAD)

NYC makes data about many things open and free for anyone to download and look at. This is the NYC Open Data website: https://opendata.cityofnewyork.us (https://opendata.cityofnewyork.us).

We also are going to use a dataset called *us-cities-demographics.csv*. This dataset contains information about the demographics of all US cities and census-designated places with a population greater or equal to 65,000. This data comes from the US Census Bureau's 2015 American Community Survey. The data file is available here: https://public.opendatasoft.com/explore/dataset/us-cities-demographics/information/ (https://public.opendatasoft.com/explore/dataset/us-cities-demographics/information/)

More resources for leaning R

There are numerous resources for learning about R; you will find some available on this webpage: resouces page (https://crumplab.github.io/psyc3400/Resources.html).

It is not required for any of the MA336 Labs, but if you are interested in downloading and installing R and R Studio at home to your computer (it's free), you can check how to do so here: general introduction to R and Rstudio (https://crumplab.github.io/statisticsLab/software.html#r).

Get some data

To create a frequency table, graph data, and calculate the mean and standard deviation, we need to have some data first.

With R, that’s not entirely true. The default installation of R comes with several data sets.

Here are some of the pre-loaded data sets in R:

- `iris` -> Edgar Anderson’s iris data
- `women` -> Average heights and weights for American women
- `pressure` -> Vapor pressure of mercury as a function of temperature
- `mtcars` -> Motor trend car road tests
- `cars` -> Speed and stopping distances of cars

Let’s look at the data set called `women` .

```
women
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Because R is a language built for statistics, it contains many functions that allow you generate random data.

```
rnorm(20, mean=8, sd=1.5)
```

```
##  [1]  8.991169 10.670567  9.612825  7.280626  9.947510  8.744634 10.557744
##  [8]  5.670601  5.130413  8.570563  8.751443  9.159175  6.652833  9.058252
## [15]  5.912634 11.352714  7.455835  7.833809  9.370763  7.091717
```

With the code above, you just made R to sample 20 numbers with a normal distribution with a mean of 8 and a standard deviation of 1.5 (later in the course, we will cover this type of distribution).

In future labs, we will look at how to use R to generate samples of numbers according to other types of distributions.

Reading data

Let’s do something that might be more interesting for now. We will focus on grabbing data from a local file or from a URL.

Suppose we would like to know *what kind of film productions are being shot in NYC*.

To answer this question, let’s use the data set in the file *Film_Permits.csv.*

Searching through the data, we can find a data file that lists the locations of film permits for shooting movies throughout NYC’s five boroughs. There are multiple ways to load/import this data into R.

Use the following commands to load (import) the data.

```
library(data.table)
nyc_films <-fread("Film_Permits.csv")
nyc_films
```

```
##        Event ID                    EventType  StartDateTime    EndDateTime
##     1:   446040              Shooting Permit 10/19/18 14:00  10/20/18 4:00
##     2:   446168              Shooting Permit 10/19/18 14:00  10/20/18 2:00
##     3:   186438              Shooting Permit  10/30/14 7:00  10/31/14 2:00
##     4:   445255              Shooting Permit  10/20/18 7:00 10/20/18 18:00
##     5:   128794 Theater Load in and Load Outs  11/16/13 0:01  11/17/13 6:00
##    ---
## 70985:   533990              Shooting Permit   2/26/20 6:00  2/26/20 20:00
## 70986:   465761              Shooting Permit   2/27/19 7:30  2/27/19 20:30
## 70987:   511630              Shooting Permit 10/11/19 10:00 10/11/19 23:00
## 70988:   584697              Shooting Permit    7/4/21 6:00   7/4/21 23:59
## 70989:   491040              Shooting Permit   6/12/19 7:00  6/12/19 20:00
##             EnteredOn                                    EventAgency
##     1: 10/16/18 11:57 Mayor's Office of Film, Theatre & Broadcasting
##     2: 10/16/18 19:03 Mayor's Office of Film, Theatre & Broadcasting
##     3: 10/27/14 12:14 Mayor's Office of Film, Theatre & Broadcasting
##     4:  10/9/18 21:34 Mayor's Office of Film, Theatre & Broadcasting
##     5:  11/7/13 15:48 Mayor's Office of Film, Theatre & Broadcasting
##    ---
## 70985:  2/21/20 14:40 Mayor's Office of Film, Theatre & Broadcasting
## 70986:  2/22/19 14:57 Mayor's Office of Film, Theatre & Broadcasting
## 70987:  10/8/19 14:05 Mayor's Office of Film, Theatre & Broadcasting
## 70988:   6/11/21 0:39 Mayor's Office of Film, Theatre & Broadcasting
## 70989:  6/10/19 12:38 Mayor's Office of Film, Theatre & Broadcasting
##
ParkingHeld
##     1:
THOMPSON STREET between PRINCE STREET and SPRING STREET,  SPRING STREET between WOOSTER
STREET and 6TH AVENUE,  SPRING STREET between THOMPSON STREET and 6TH AVENUE,  6TH AVENU
E between VANDAM STREET and BROOME STREET,  SULLIVAN STREET between WEST HOUSTON STREET
and PRINCE STREET,  PRINCE STREET between SULLIVAN STREET and 6 AVENUE
##     2:
MARBLE HILL AVENUE between WEST  227 STREET and WEST  225 STREET,  WEST  228 STREET betw
een ADRIAN AVENUE and MARBLE HILL AVENUE
##     3:
LAUREL HILL BLVD between REVIEW AVENUE and RUST ST,  REVIEW AVE between VAN DAM STREET a
nd LAUREL HILL BOULEVARD,  59 ROAD between 60 LANE and 61 STREET,  59 ROAD between 60 LA
NE and 61 STREET,  61 STREET between 59 ROAD and FRESH POND ROAD,  FRESH POND ROAD betwe
en 59 AVENUE and 59 DRIVE,  59 DRIVE between FRESH POND ROAD and 63 STREET,  59 DRIVE be
tween FRESH POND ROAD and 64 STREET
##     4:
JORALEMON STREET between BOERUM PLACE and COURT STREET
##     5:
WEST   31 STREET between 7 AVENUE and 8 AVENUE,  8 AVENUE between WEST   31 STREET and W
EST   33 STREET
##    ---
## 70985: CENTER BOULEVARD between NORTH BASIN ROAD and 46 AVENUE,  21 STREET between 43
AVENUE and 44 AVENUE,  46 AVENUE between 5 STREET and VERNON BOULEVARD,  46 AVENUE betwe
en CENTER BOULEVARD and 5 STREET,  22 STREET between 44 AVENUE and 43 AVENUE,  CENTER BO
ULEVARD between 46 AVENUE and 47 AVENUE,  5 STREET between NORTH BASIN ROAD and 46 AVENU
E,  5 STREET between 46 AVENUE and 46 ROAD,  43 AVENUE between 21 STREET and 22 STREET,
44 AVENUE between 21 STREET and 23 STREET,  21 STREET between 44 AVENUE and 44 ROAD,  21
```

```
STREET between 44 ROAD and 44 DRIVE,  44 ROAD between 11 STREET and 21 STREET
## 70986:
GREENWICH STREET between BARROW STREET and WEST   10 STREET,  GREENWICH STREET between B
ARROW STREET and MORTON STREET,  HUDSON STREET between BARROW STREET and GROVE STREET,
HUDSON STREET between GROVE STREET and CHRISTOPHER STREET,  CHRISTOPHER STREET between G
REENWICH STREET and WASHINGTON STREET,  BARROW STREET between HUDSON STREET and WASHINGT
ON STREET,  HUDSON STREET between BARROW STREET and GROVE STREET,  HUDSON STREET between
GROVE STREET and CHRISTOPHER STREET
## 70987:
44 ROAD between 24 STREET and HUNTER STREET,  24 STREET between 44 ROAD and 43 AVENUE,
JACKSON AVENUE between PEARSON STREET and COURT SQUARE,  COURT SQUARE between JACKSON AV
ENUE and DEAD END,  43 AVENUE between CRESCENT STREET and 24 STREET,  COURT SQUARE betwe
en JACKSON AVENUE and DEAD END,  COURT SQUARE between JACKSON AVENUE and DEAD END,  THOM
PSON AVENUE between COURT SQUARE and 44 DRIVE,  44 DRIVE between JACKSON AVENUE and THOM
PSON AVENUE,  COURT SQUARE WEST between JACKSON AVENUE and DEAD END
## 70988:
CENTER BOULEVARD between 54 AVENUE and 55 AVENUE,  55 AVENUE between CENTER BOULEVARD an
d 2 STREET,  2 STREET between 54 AVENUE and 55 AVENUE
## 70989:
SPOFFORD AVENUE between COSTER STREET and FAILE STREET,  HUNTS POINT AVENUE between SPOF
FORD AVENUE and BRYANT AVENUE,  HUNTS POINT AVENUE between BRYANT AVENUE and RANDALL AVE
NUE,  WALNUT AVENUE between EAST  139 STREET and EAST  140 STREET,  BRYANT AVENUE betwee
n HUNTS POINT AVENUE and RANDALL AVENUE,  RANDALL AVENUE between BRYANT AVENUE and LONGF
ELLOW AVENUE,  EAST  140 STREET between LOCUST AVENUE and WALNUT AVENUE,  EAST  139 STRE
ET between WALNUT AVENUE and LOCUST AVENUE
##          Borough CommunityBoard(s) PolicePrecinct(s)          Category
##     1: Manhattan                 2                 1        Television
##     2: Manhattan             12, 8            34, 50              Film
##     3:    Queens              2, 5          104, 108        Television
##     4:  Brooklyn                 2                84 Still Photography
##     5: Manhattan              4, 5                14           Theater
##    ---
## 70985:    Queens                 2               108        Television
## 70986: Manhattan                 2                 6        Television
## 70987:    Queens                 2               108        Television
## 70988:    Queens                 2               108        Television
## 70989:     Bronx           1, 2, 7        40, 41, 72        Television
##            SubCategoryName                  Country          ZipCode(s)
##     1:      Cable-episodic United States of America               10012
##     2:             Feature United States of America        10034, 10463
##     3:     Episodic series United States of America               11378
##     4:      Not Applicable United States of America               11201
##     5:             Theater United States of America        10001, 10121
##    ---
## 70985:     Episodic series United States of America        11101, 11109
## 70986:     Episodic series United States of America               10014
## 70987:     Episodic series United States of America               11101
## 70988: Special/Awards Show United States of America               11101
## 70989:     Episodic series United States of America 10454, 10474, 11220
```

If you have Internet access enabled in your project, you can also try loading the data from the source:

```
library(data.table)
nyc_films <- fread("https://data.cityofnewyork.us/api/views/tg4x-b46p/rows.csv?accessType=DOWNLOAD")
```

Note: The code above will not be evaluated in R because the option `eval` is set to `FALSE`. To run it, you must change the value of `eval` to `TRUE`.

Data frames

The data above is stored in a `data frame`. A data frame is a rectangular collection of variables (in the columns) and observations (in the rows). It looks like an Excel spreadsheet if you are familiar with Excel.

The variable `nyc_films` stores the data frame imported from the file *Film_Permits.csv*.

It is possible to build a data frame from scratch. But most of the time, we will use some actual data in the labs to obtain our data frames from reading files.

The data structure

Let's look at how the data is structured. For example, to the size of this data set (data frame), we can use the function `dim`.

```
data_size <- dim(nyc_films)
data_size
```

```
## [1] 70989    14
```

It's helpful to know about it if you need to look at the data frame itself. But this data frame is massive; it has 70989 rows and 14 columns of data. That's a lot to look at.

We examine what is in a data frame in R using different functions. Here are some examples: `head()`, `tail()`, `str()`, `nrow`, `ncol`, `colnames()`, and `summary`.

```
colnames(nyc_films)
```

```
##  [1] "Event ID"            "EventType"           "StartDateTime"
##  [4] "EndDateTime"         "EnteredOn"           "EventAgency"
##  [7] "ParkingHeld"         "Borough"             "CommunityBoard(s)"
## [10] "PolicePrecinct(s)"   "Category"            "SubCategoryName"
## [13] "Country"             "ZipCode(s)"
```

```
head(nyc_films)
```

```
##    Event ID                     EventType  StartDateTime    EndDateTime
## 1:   446040               Shooting Permit 10/19/18 14:00  10/20/18 4:00
## 2:   446168               Shooting Permit 10/19/18 14:00  10/20/18 2:00
## 3:   186438               Shooting Permit  10/30/14 7:00  10/31/14 2:00
## 4:   445255               Shooting Permit  10/20/18 7:00 10/20/18 18:00
## 5:   128794 Theater Load in and Load Outs  11/16/13 0:01  11/17/13 6:00
## 6:    43547               Shooting Permit   1/10/12 7:00  1/10/12 19:00
##         EnteredOn                                    EventAgency
## 1: 10/16/18 11:57 Mayor's Office of Film, Theatre & Broadcasting
## 2: 10/16/18 19:03 Mayor's Office of Film, Theatre & Broadcasting
## 3: 10/27/14 12:14 Mayor's Office of Film, Theatre & Broadcasting
## 4:  10/9/18 21:34 Mayor's Office of Film, Theatre & Broadcasting
## 5:  11/7/13 15:48 Mayor's Office of Film, Theatre & Broadcasting
## 6:   1/4/12 12:25 Mayor's Office of Film, Theatre & Broadcasting
##
ParkingHeld
## 1:                                                      THOMPSON STREET between PRINC
E STREET and SPRING STREET,  SPRING STREET between WOOSTER STREET and 6TH AVENUE,  SPRIN
G STREET between THOMPSON STREET and 6TH AVENUE,  6TH AVENUE between VANDAM STREET and B
ROOME STREET,  SULLIVAN STREET between WEST HOUSTON STREET and PRINCE STREET,  PRINCE ST
REET between SULLIVAN STREET and 6 AVENUE
## 2:
MARBLE HILL AVENUE between WEST  227 STREET and WEST  225 STREET,  WEST  228 STREET betw
een ADRIAN AVENUE and MARBLE HILL AVENUE
## 3: LAUREL HILL BLVD between REVIEW AVENUE and RUST ST,  REVIEW AVE between VAN DAM ST
REET and LAUREL HILL BOULEVARD,  59 ROAD between 60 LANE and 61 STREET,  59 ROAD between
60 LANE and 61 STREET,  61 STREET between 59 ROAD and FRESH POND ROAD,  FRESH POND ROAD
between 59 AVENUE and 59 DRIVE,  59 DRIVE between FRESH POND ROAD and 63 STREET,  59 DRI
VE between FRESH POND ROAD and 64 STREET
## 4:
JORALEMON STREET between BOERUM PLACE and COURT STREET
## 5:
WEST   31 STREET between 7 AVENUE and 8 AVENUE,  8 AVENUE between WEST   31 STREET and W
EST   33 STREET
## 6:
EAGLE STREET between FRANKLIN STREET and WEST STREET,  WEST STREET between EAGLE STREET
and FREEMAN STREET,  FREEMAN STREET between WEST STREET and FRANKLIN STREET
##      Borough CommunityBoard(s) PolicePrecinct(s)          Category
## 1: Manhattan                 2                 1        Television
## 2: Manhattan             12, 8            34, 50              Film
## 3:    Queens              2, 5          104, 108        Television
## 4:  Brooklyn                 2                84 Still Photography
## 5: Manhattan              4, 5                14           Theater
## 6:  Brooklyn              1, 2           108, 94        Television
##    SubCategoryName                  Country   ZipCode(s)
## 1:  Cable-episodic United States of America        10012
## 2:         Feature United States of America 10034, 10463
## 3: Episodic series United States of America        11378
## 4:  Not Applicable United States of America        11201
## 5:         Theater United States of America 10001, 10121
## 6: Episodic series United States of America 11101, 11222
```

```
str(nyc_films)
```

```
## Classes 'data.table' and 'data.frame':   70989 obs. of  14 variables:
##  $ Event ID         : int  446040 446168 186438 445255 128794 43547 66846 104342 2448
63 446379 ...
##  $ EventType        : chr  "Shooting Permit" "Shooting Permit" "Shooting Permit" "Sho
oting Permit" ...
##  $ StartDateTime    : chr  "10/19/18 14:00" "10/19/18 14:00" "10/30/14 7:00" "10/20/1
8 7:00" ...
##  $ EndDateTime      : chr  "10/20/18 4:00" "10/20/18 2:00" "10/31/14 2:00" "10/20/18
18:00" ...
##  $ EnteredOn        : chr  "10/16/18 11:57" "10/16/18 19:03" "10/27/14 12:14" "10/9/1
8 21:34" ...
##  $ EventAgency      : chr  "Mayor's Office of Film, Theatre & Broadcasting" "Mayor's
Office of Film, Theatre & Broadcasting" "Mayor's Office of Film, Theatre & Broadcasting"
"Mayor's Office of Film, Theatre & Broadcasting" ...
##  $ ParkingHeld      : chr  "THOMPSON STREET between PRINCE STREET and SPRING STREET,
SPRING STREET between WOOSTER STREET and 6TH AVENUE, "| __truncated__ "MARBLE HILL AVENU
E between WEST  227 STREET and WEST  225 STREET,  WEST  228 STREET between ADRIAN AVENUE
and M"| __truncated__ "LAUREL HILL BLVD between REVIEW AVENUE and RUST ST,  REVIEW AVE b
etween VAN DAM STREET and LAUREL HILL BOULEVAR"| __truncated__ "JORALEMON STREET between
BOERUM PLACE and COURT STREET" ...
##  $ Borough          : chr  "Manhattan" "Manhattan" "Queens" "Brooklyn" ...
##  $ CommunityBoard(s): chr  "2" "12, 8" "2, 5" "2" ...
##  $ PolicePrecinct(s): chr  "1" "34, 50" "104, 108" "84" ...
##  $ Category         : chr  "Television" "Film" "Television" "Still Photography" ...
##  $ SubCategoryName  : chr  "Cable-episodic" "Feature" "Episodic series" "Not Applicab
le" ...
##  $ Country          : chr  "United States of America" "United States of America" "Uni
ted States of America" "United States of America" ...
##  $ ZipCode(s)       : chr  "10012" "10034, 10463" "11378" "11201" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Example

Use the file *us-cities-demographics.csv* to code the following:

1. Import the file into R using the name `demo` for storing the data.
2. Find out the size of the file and the columns' names and read the top lines of the file.

```
Enter your code below:
```

Note on column names in R

If you are new to R and mainly from an Excel background, think more about column names than you usually might. Excel is very flexible when it comes to naming columns.

To ease this process, it is best to keep column names simple, without spaces, and without special characters (e.g. !, @, &, $, etc.).

But sometimes, we import a data set with spaces or other symbols in the name of the variables (columns).

Option 1: One way to deal with this is to invoke the columns with spaces or special characters by surrounding their names between single backticks: `column_name`.

```
head(nyc_films$`Event ID`)
```

```
## [1] 446040 446168 186438 445255 128794  43547
```

Note: the above code asks to display the top values in the column *Event ID* in the data frame code *nyc_films*. Note the use of the dollar sign `$`.

Option 2: Another way to do that is by renaming the columns without spaces or special characters.

```
colnames(nyc_films)[1]<-"Event_ID"
colnames(nyc_films)
```

```
##  [1] "Event_ID"           "EventType"          "StartDateTime"
##  [4] "EndDateTime"        "EnteredOn"          "EventAgency"
##  [7] "ParkingHeld"        "Borough"            "CommunityBoard(s)"
## [10] "PolicePrecinct(s)"  "Category"           "SubCategoryName"
## [13] "Country"            "ZipCode(s)"
```

Note: column 1 was renamed `Event_ID` in the above code.

Example

For the data frame corresponding to the file *us-cities-demographics.csv*, remove spaces from the variables *Male Population*, *Female Population*, *Total Population*, and *State Code*. Male Population*,* Female Population*,* Total Population*, and* State Code*.

```
Enter your code below:
```

The package dplyr in tidyverse

The `tidyverse` package is designed for data science and contains other packages that all work harmoniously.

The packages in `tidyverse` that we will cover in this and future labs are `dplyr` and `ggplot2`. You can use the function `library()` to load the whole tidyverse package or the individual packages.

The package `dplyr` provides tools for the most common data manipulation tasks. It is built to work directly with data frames.

We are going to learn a couple of key `dplyr` functions that allow you to deal with data manipulation challenges:

- `filter()`: Pick observations (rows) by their values.
- `summarize()`: Collapse many values into a single summary.

These functions can be used in conjunction with `group_by()`, which changes each of the functions above from operating on the entire dataset to working on it group by group.

Let’s dive in and see how these functions work.

Summaries with summarize() and group_by()

The `group_by()` function splits the data into groups upon which some operations can be run.

`group_by()` is often used together with `summarize()` , which collapses each group into a single-row summary of that group.

We might have questions about this data in the file *Film_Permits.csv* that can help us understand how to use these functions.

Question 1: Where are the most film permits being requested?

For which of the five boroughs of NYC most of the film permits are being requested?

We can find out by creating a frequency table for the data. We need to count how many film permits are made in each borough.

```
library(dplyr)
```

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

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

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

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

```
counts <- nyc_films %>%
          group_by(Borough) %>%
          summarize(count_of_permits = length(Borough))
counts
```

```
## # A tibble: 5 × 2
##   Borough       count_of_permits
##   <chr>                    <int>
## 1 Bronx                     2280
## 2 Brooklyn                 22087
## 3 Manhattan                34200
## 4 Queens                   11484
## 5 Staten Island              938
```

Pipes (`%>%`) let you take the output of one function and send it directly to the next.

The above code grouped the data by each of the five boroughs and then counted the number of times each Borough occurred (using the `length` function). The result is a new variable called `counts` .

Example

Enter an R code below that shows how many cities each state had a population of more than 65K in 2015.

```
Enter your code below:
```

Question 2: What kind of “films” are being made, and what is the category?

You might be skeptical of what you are doing here, copying and pasting things. Soon you’ll see how fast you can do something by copying, pasting, and making a few changes.

Let’s quickly ask another question about what kinds of films are being made. The column `Category` gives us some information about that. Let’s copy-paste the code we already made above and see what categories the films fall into. See if you can tell what was changed in the code to make this work:

```
counts <- nyc_films %>%
          group_by(Category) %>%
          summarize(count_of_permits = length(Category))
counts
```

```
## # A tibble: 10 × 2
##    Category             count_of_permits
##    <chr>                           <int>
##  1 Commercial                       5569
##  2 Documentary                       296
##  3 Film                            11061
##  4 Music Video                       236
##  5 Red Carpet/Premiere                 1
##  6 Still Photography                4165
##  7 Student                           433
##  8 Television                      39840
##  9 Theater                          6802
## 10 WEB                              2586
```

Let’s notice the changes.

1. `Borough` was changed to `Category` . That was the main thing.
2. Note that none of the `library()` commands are used again, and I didn’t re-run the very early code to get the data. R already has those things in its memory, so we don’t need to do that again. If you ever clear the memory of R, you will need to reload those things.

Example

Now let’s use another R function, `sum` , to add the female population per state.

```
Enter your code below:
```

Question 3: What is the number of permits for each borough category?

We can use the function `group_by` with more than one parameter, i.e., group by more than one variable.

```
counts <- nyc_films %>%
          group_by(Borough,Category) %>%
          summarize(count_of_permits = length(Category))
```

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

```
head(counts)
```

```
## # A tibble: 6 × 3
## # Groups:   Borough [1]
##   Borough Category          count_of_permits
##   <chr>   <chr>                        <int>
## 1 Bronx   Commercial                     152
## 2 Bronx   Documentary                     11
## 3 Bronx   Film                           466
## 4 Bronx   Music Video                      6
## 5 Bronx   Still Photography               64
## 6 Bronx   Student                         12
```

```
tail(counts)
```

```
## # A tibble: 6 × 3
## # Groups:   Borough [1]
##   Borough       Category          count_of_permits
##   <chr>         <chr>                        <int>
## 1 Staten Island Music Video                      4
## 2 Staten Island Still Photography                2
## 3 Staten Island Student                         12
## 4 Staten Island Television                     589
## 5 Staten Island Theater                          1
## 6 Staten Island WEB                              4
```

We did two critical things. First, we added `Borough` and `Category` into the `group_by()` function. This automatically gives separate counts for each film category for each Borough.

Example

Enter a chunk of code to group the demographic data by *State Code* and *Race*, which adds population (Count) for each race for each state.

```
Enter your code below:
```

Filtering data with dplyr

`filter()` allows you to subset observations (rows) based on their values. The first argument is the name of the data frame. The second and subsequent arguments are the expressions that filter the data frame.

The library `dplyr` allows us to filter data by a single Borough. Let’s use the function `filter()` for the NYC permits data to find the number of permits only for Manhattan.

```
counts <- nyc_films %>%
         group_by(Category) %>%
         filter(Borough=='Manhattan')%>%
         summarize(count_of_permits = length(Category))

counts
```

```
## # A tibble: 10 × 2
##    Category           count_of_permits
##    <chr>                         <int>
##  1 Commercial                     3290
##  2 Documentary                     193
##  3 Film                           5106
##  4 Music Video                     108
##  5 Red Carpet/Premiere               1
##  6 Still Photography              2611
##  7 Student                         249
##  8 Television                    15583
##  9 Theater                        5628
## 10 WEB                            1431
```

Now let’s do something similar with the demographic data.

Example

Enter a chunk of code to group the demographic data by *State Code* and add the population of Hispanic or Latino (Count) for each state.

```
Enter your code below:
```

dplyr summary

For all functions in `dplyr`, we have the following:

1. The first argument is a data frame.
2. The subsequent arguments describe what to do with the data frame using the variable names (without quotes).
3. The result is a new data frame.

References

This document contains excerpts and adaptions from:

- Matthew J. C. Crump, Anjali Krishnan, Stephen Volz, and Alla Chavarga (2018) “Answering questions with data: Lab Manual.” Last compiled on 2019-04-06. https://www.crumplab.com/statisticsLab/ (https://www.crumplab.com/statisticsLab/)
- Data Carpentry. “Aggregating and analyzing data with dplyr.” https://datacarpentry.org/dc_zurich/R-ecology/04-dplyr.html (https://datacarpentry.org/dc_zurich/R-ecology/04-dplyr.html)
- Hadley Wickham and Garrett Grolemund. “R for Data Science.”https://r4ds.had.co.nz/index.html (https://r4ds.had.co.nz/index.html)