Introduction to Data Manipulation in R

Intructions: Complete this lab at your own pace with your assigned lab group. Remember to work in the R project that you created for labs, and to store the data set, `colleges2015.csv`, in your data folder. Within this R project, you should create an R markdown file with your solutions. Please complete your solutions by 4 pm on Wednesday, September 28 and submit one solution per group on Moodle (you should submit both an .rmd and .html file).

Data manipulation is central to data analysis and is often the most time consuming portion of an analysis. The `dplyr` package contains a suite of functions to make data manipulation easier. The core functions of the `dplyr` package can be thought of as verbs for data manipulation.

Verb(s)	Meaning
`filter` and `slice`	pick specific observations (i.e. specific rows)
`arrange`	reorder the rows
`select`	pick variables by their names (i.e. specific columns)
`mutate`	add new calculated columns to a data frame
`summarize`	aggregate many rows into a single row

In this example we will explore how to use each of these functions, as well as how to combine them with the `group_by` function for group-wise manipulations.

To begin, let's make sure that our data set and the `dplyr` package are loaded

```
# I stored colleges2015.csv in the data folder of my lab project
# if you are doing something else, then change the file path
colleges <- read.csv("data/colleges2015.csv")
library(dplyr)
```

Data: The file `college2015.csv` contains information on predominantly bachelor's-degree granting institutions from 2015 that might be of interest to a college applicant.

To get a feel for what data are available, look at the first six rows

```
head(colleges)
```

```
##   unitid                              college    type       city state
## 1 100654            Alabama A & M University  public     Normal    AL
## 2 100663 University of Alabama at Birmingham  public Birmingham    AL
## 3 100690                  Amridge University private Montgomery    AL
## 4 100706 University of Alabama in Huntsville  public Huntsville    AL
## 5 100724            Alabama State University  public Montgomery    AL
## 6 100751           The University of Alabama  public Tuscaloosa    AL
##      region admissionRate ACTmath ACTenglish undergrads  cost gradRate
## 1 Southeast        0.8989      17         17       4051 18888   0.2914
## 2 Southeast        0.8673      23         26      11200 19990   0.5377
## 3 Southeast            NA      NA         NA        322 12300   0.6667
## 4 Southeast        0.8062      25         26       5525 20306   0.4835
## 5 Southeast        0.5125      17         17       5354 17400   0.2517
## 6 Southeast        0.5655      25         27      28692 26717   0.6665
##   FYretention fedloan    debt
## 1      0.6314  0.8204 33611.5
## 2      0.8016  0.5397 23117.0
## 3      0.3750  0.7629 26995.0
```

```
## 4      0.8098  0.4728 24738.0
## 5      0.6219  0.8735 33452.0
## 6      0.8700  0.4148 24000.0
```

the last six rows

```
tail(colleges)
```

and the structure of the data frame.

```
str(colleges)
```

1. Filtering rows

To extract the rows only for colleges and universities in a specific state we use the `filter` function. For example, we can extract the colleges in Wisconsin from the **colleges** data set using the following code:

```
wi <- filter(colleges, state == "WI")
head(wi)
```

```
##   unitid                     college    type      city state      region
## 1 238193             Alverno College private Milwaukee    WI Great Lakes
## 2 238324              Bellin College private Green Bay    WI Great Lakes
## 3 238333              Beloit College private    Beloit    WI Great Lakes
## 4 238430 Cardinal Stritch University private Milwaukee    WI Great Lakes
## 5 238458          Carroll University private  Waukesha    WI Great Lakes
## 6 238476            Carthage College private   Kenosha    WI Great Lakes
##   admissionRate ACTmath ACTenglish undergrads  cost gradRate FYretention
## 1        0.7887      19         19       1833 30496   0.3852      0.7173
## 2        0.5556      25         24        285    NA   0.6786      0.7500
## 3        0.6769      26         28       1244 48236   0.7815      0.9228
## 4        0.8423      22         22       2680 37563   0.4162      0.7099
## 5        0.8114      24         24       3024 37963   0.5629      0.7598
## 6        0.7018      24         24       2874 44910   0.6501      0.7841
##   fedloan  debt
## 1  0.8784 33110
## 2  0.8145 18282
## 3  0.5784 26500
## 4  0.7178 27875
## 5  0.7108 27000
## 6  0.8048 27000
```

Remarks

- The first argument given to `filter` is always the data frame (this is true for all the core functions in `dplyr`), followed by logical tests that the returned cases must pass. In our example, the test was whether the school was in Wisconsin, which is written as `state == "WI"`.
- We have to use `==` to indicate equality because `=` is equivalent to `<-`.
- When testing character variables, be sure to use quotes to specify the value of the variable that you are testing.
- **To specify multiple tests**, use a comma to separate the tests (think of the comma as the word "and"). For example,

  ```
  smallWI <- filter(colleges, state == "WI", undergrads < 2000)
  ```

returns only those rows corresponding to schools in Wisconsin with fewer than 2,000 undergraduate students.

- To specify that at least one test must be passed, use the `|` character instead of the comma. For example, the below test checks whether a college is in Wisconsin or Minnesota or Iowa, so it returns all of the colleges in Wisconsin, Minnesota, and Iowa.

  ```
  WiMnIa <- filter(colleges, state == "WI" | state == "MN" | state == "IA")
  ```

- You can use both `|` and `,` to specify multiple tests. For example, we can return all colleges with fewer than 2,000 undergraduate students in Wisconsin, Minnesota, and Iowa.

  ```
  smallWIM <- filter(colleges, state == "WI" | state == "MN" | state == "IA", undergrads < 2000)
  ```

- Common comparison operators for the tests include: `>`, `>=`, `<`, `<=`, `!=` (not equal), and `==` (equal).

- To remove rows with missing values, use the R command `na.omit`. For example,
  `colleges <- na.omit(colleges)`

  will reduce the data set to only rows with no missing values.

  ```
  colleges <- filter(colleges, !is.na(cost))
  ```

  will eliminate only rows with `NA` in the cost column.

Questions:

1) How many Maryland colleges are in the **colleges** data frame? (The abbreviation for Maryland is MD.)
2) How many private Maryland colleges with under 5000 undergraduates are in the **colleges** data frame?

2. Slicing rows

To extract rows 10 through 16 from the **colleges** data frame we use the `slice` function.

```
slice(colleges, 10:16)
```

```
##   unitid                       college    type         city state
## 1 100937   Birmingham Southern College private   Birmingham    AL
## 2 101073     Concordia College Alabama private        Selma    AL
## 3 101189           Faulkner University private   Montgomery    AL
## 4 101435            Huntingdon College private   Montgomery    AL
## 5 101453 Heritage Christian University private     Florence    AL
## 6 101480 Jacksonville State University  public Jacksonville    AL
## 7 101541               Judson College private       Marion    AL
##      region admissionRate ACTmath ACTenglish undergrads  cost gradRate
## 1 Southeast        0.6422      25         27       1181 44512   0.6192
## 2 Southeast            NA      NA         NA        523 17655   0.2115
## 3 Southeast            NA      NA         NA       2358 28485   0.2287
## 4 Southeast        0.6279      20         22       1100 31433   0.4319
## 5 Southeast            NA      NA         NA         67 21160   0.0000
## 6 Southeast        0.8326      21         22       7195 19202   0.3083
## 7 Southeast        0.7388      20         23        331 27815   0.4051
##   FYretention fedloan  debt
## 1      0.8037  0.4939 27000
## 2      0.4103  0.9100 26500
## 3      0.5000  0.7427 23750
## 4      0.6196  0.7227 27000
## 5      1.0000  0.4839    NA
```

```
## 6      0.7112  0.6811 23500
## 7      0.5974  0.7110 26000
```

Remarks

- **To select consecutive rows**, create a vector of the row indices by separating the first and last row numbers with a `:`.
- **To select non-consecutive rows**, create a vector manually by concatenating the row numbers using `c()`. For example, to select the 2nd, 18th, and 168th rows use `slice(colleges, c(2, 18, 168))`.

3. Arranging rows

To sort the rows by total cost, from the least expensive to the most expensive, we use the `arrange` function.

```
costDF <- arrange(colleges, cost)
head(costDF)
```

```
##   unitid                                  college    type        city
## 1 197027     United States Merchant Marine Academy  public Kings Point
## 2 176336              Southeastern Baptist College private      Laurel
## 3 241951 Escuela de Artes Plasticas de Puerto Rico  public    San Juan
## 4 241216               Atlantic University College private    Guaynabo
## 5 241377              Caribbean University-Bayamon private     Bayamon
## 6 243221     University of Puerto Rico-Rio Piedras  public    San Juan
##   state               region admissionRate ACTmath ACTenglish undergrads
## 1    NY U.S. Service Schools            NA      NA         NA        958
## 2    MS            Southeast            NA      NA         NA         37
## 3    PR       Outlying Areas        0.7154      NA         NA        529
## 4    PR       Outlying Areas            NA      NA         NA       1365
## 5    PR       Outlying Areas            NA      NA         NA       1572
## 6    PR       Outlying Areas        0.5248      NA         NA      11834
##   cost gradRate FYretention fedloan debt
## 1 6603   0.7365      0.9733  0.0780 4211
## 2 6753   0.6875      1.0000  0.0000   NA
## 3 7248   0.4127      0.8382  0.0000   NA
## 4 7695   0.3891      0.7200  0.1053 5000
## 5 8006   0.2166      0.7951  0.2210 9000
## 6 8020   0.4748      0.8968  0.0966 5500
```

Remarks

- By default, `arrange` assumes that we want the data arranged in ascending order by the specified variable(s).
- **To arrange the rows in descending order**, wrap the variable name in the `desc` function. For example, to arrange the data frame from most to least expensive we would use the following command:

  ```
  costDF <- arrange(colleges, desc(cost))
  ```

- To arrange a data frame by the values of multiple variables, list the variables in a comma separated list. The order of the variables specifies the order in which the data frame will be arranged. For example,

  ```
  actDF <- arrange(colleges, desc(ACTmath), desc(ACTenglish))
  ```

  reorders **colleges** first by the median ACT math score (in descending order) and then by the ACT english score (in descending order)

Questions

3) What school is most expensive?

4) What school has the least expensive tuition in Wisconsin?

4. Selecting columns

Suppose that you are only interested in a subset of the columns in the data set—say, `college`, `city`, `state`, `undergrads`, and `cost`—and want to create a data frame with only these columns. To do this, we `select` the desired columns:

```
lessCols <- select(colleges, college, city, state, undergrads, cost)
head(lessCols)
```

```
##                               college       city state undergrads  cost
## 1            Alabama A & M University     Normal    AL       4051 18888
## 2 University of Alabama at Birmingham Birmingham    AL      11200 19990
## 3                  Amridge University Montgomery    AL        322 12300
## 4 University of Alabama in Huntsville Huntsville    AL       5525 20306
## 5            Alabama State University Montgomery    AL       5354 17400
## 6           The University of Alabama Tuscaloosa    AL      28692 26717
```

Remarks

- After specifying the data frame, list the variable names to select from the data frame separated by commas.
- In some cases you may want to drop a small number of variables from a data frame. In this case, putting a negative sign before a variable name tells `select` to select all but the negated variables. For example, if we only wished to drop the `unitid` variable we run the following command:

```
drop_unitid <- select(colleges, -unitid)
head(drop_unitid)
```

```
##                               college    type       city state    region
## 1            Alabama A & M University  public     Normal    AL Southeast
## 2 University of Alabama at Birmingham  public Birmingham    AL Southeast
## 3                  Amridge University private Montgomery    AL Southeast
## 4 University of Alabama in Huntsville  public Huntsville    AL Southeast
## 5            Alabama State University  public Montgomery    AL Southeast
## 6           The University of Alabama  public Tuscaloosa    AL Southeast
##   admissionRate ACTmath ACTenglish undergrads  cost gradRate FYretention
## 1        0.8989      17         17       4051 18888   0.2914      0.6314
## 2        0.8673      23         26      11200 19990   0.5377      0.8016
## 3            NA      NA         NA        322 12300   0.6667      0.3750
## 4        0.8062      25         26       5525 20306   0.4835      0.8098
## 5        0.5125      17         17       5354 17400   0.2517      0.6219
## 6        0.5655      25         27      28692 26717   0.6665      0.8700
##   fedloan    debt
## 1  0.8204 33611.5
## 2  0.5397 23117.0
## 3  0.7629 26995.0
## 4  0.4728 24738.0
## 5  0.8735 33452.0
## 6  0.4148 24000.0
```

5. Mutating data (adding new columns)

Data sets often do not contain the exact variables we need, but contain all of the information necessary to calculate the needed variables. In this case, we can use the `mutate` function to add a new column to a data frame that is calculated from other variables. For example, we may wish to report percentages rather than proportions for the admissions rate.

```
colleges <- mutate(colleges, admissionPct = 100 * admissionRate)
```

Remarks

- After specifying the data frame, give the name of the new variable and it's definition. Notice that we need to use `=` to assign the value of the new variable.
- **To add multiple variables once**, separate the list of new variables by commas. For example, we can also add percentage versions of `FYretention` and `gradRate`.

```
colleges <- mutate(colleges, FYretentionPct = 100 * FYretention,
                   gradPct = 100 * gradRate)
```

6. Summarizing rows

To create summary statistics for columns within the data set we must aggregate all of the rows using the `summarize` command. (Note that you can also use the British spelling: `summarise`.) For example, to calculate the median cost of all 1776 colleges in our data set we run the following command:

```
summarize(colleges, medianCost = median(cost, na.rm = TRUE))
```

```
##   medianCost
## 1      29849
```

Remarks

- As with all of the functions we have seen, the first argument should be the name of the data frame.
- We add `na.rm = TRUE` here to remove any missing values in the `cost` column before the calculation. Many functions, including this summarize function, will return an error if there are missing values (blanks, `NA`s or `NaN`s) in your data.
- `summarize` returns a data frame, with one row and one column.
- We can ask for multiple aggregations in one line of code by simply using a comma separated list. For example, we can calculate the five number summary of `cost` for all 1776 colleges in our data set

```
summarize(colleges,
          min = min(cost, na.rm = TRUE),
          Q1 = quantile(cost, .25, na.rm = TRUE),
          median = median(cost, na.rm = TRUE),
          Q3 = quantile(cost, .75, na.rm = TRUE),
          max = max(cost, na.rm = TRUE))
```

```
##    min    Q1 median    Q3   max
## 1 6603 19831  29849 41180 62636
```

- Notice that even when multiple statistics are calculated, the result is a data frame with one row and the number of columns correspond to the number of summary statistics.

Question

5) What happens if we remove `na.rm = TRUE` from the code above?

7. Groupwise manipulation

Often it is of interest to manipulate data within groups. For example, we might be more interested in creating separate summaries for each state, or for private and public colleges. To do this we must first tell R what groups are of interest using the `group_by` function, and then we can use any of the above functions. Most often `group_by` is paired with `summarise` or `mutate`.

Let's first consider comparing the cost of private and public colleges. First, we must specify that the variable `type` defines the groups of interest.

```
colleges_by_type <- group_by(colleges, type)
```

Remarks

- After specifying the data frame, list the categorical variable(s) defining the groups.
- Multiple variables can be used to specify the groups. For example, to specify groups by state and type, we would run the following command:

  ```
  colleges_state_type <- group_by(colleges, state, type)
  ```

Combining `group_by` with other commands

Once we have a grouped data frame, we can obtain summaries by group via `summarize`. For example, the five number summary of cost by institution type is obtained below

```
summarize(colleges_by_type,
          min = min(cost, na.rm = TRUE),
          Q1 = quantile(cost, .25, na.rm = TRUE),
          median = median(cost, na.rm = TRUE),
          Q3 = quantile(cost, .75, na.rm = TRUE),
          max = max(cost, na.rm = TRUE))
```

```
## # A tibble: 2 x 6
##      type   min       Q1 median      Q3   max
##    <fctr> <int>    <dbl>  <dbl>   <dbl> <int>
## 1 private  6753 28787.75  37302 45728.5 62636
## 2  public  6603 16822.00  19303 21909.0 33208
```

We can also calculate new variables within groups, such as the standardized cost of attendance within each state:

```
colleges_by_state <- group_by(colleges, state)
colleges_by_state <- mutate(colleges_by_state,
                            mean.cost = mean(cost, na.rm = TRUE),
                            sd.cost = sd(cost, na.rm = TRUE),
                            std.cost = (cost - mean.cost) / sd.cost)
head(colleges_by_state)
```

```
## Source: local data frame [6 x 21]
## Groups: state [1]
##
##   unitid                              college    type       city  state
##    <int>                               <fctr>  <fctr>     <fctr> <fctr>
## 1 100654            Alabama A & M University  public     Normal     AL
## 2 100663 University of Alabama at Birmingham  public Birmingham     AL
## 3 100690                  Amridge University private Montgomery     AL
## 4 100706 University of Alabama in Huntsville  public Huntsville     AL
```

```
## 5 100724            Alabama State University  public Montgomery     AL
## 6 100751           The University of Alabama  public Tuscaloosa     AL
## # ... with 16 more variables: region <fctr>, admissionRate <dbl>,
## #   ACTmath <int>, ACTenglish <int>, undergrads <int>, cost <int>,
## #   gradRate <dbl>, FYretention <dbl>, fedloan <dbl>, debt <dbl>,
## #   admissionPct <dbl>, FYretentionPct <dbl>, gradPct <dbl>,
## #   mean.cost <dbl>, sd.cost <dbl>, std.cost <dbl>
```

Remarks

- `mutate` allows you to use variables defined earlier to calculate a new variable. This is how `std.cost` was calculated.
- The `group_by` function returns an object of class `c("grouped_df", "tbl_df", "tbl", "data.frame")`, which looks confusing, but essentially allows the data frame to be printed neatly. Notice that only the first 10 rows print when we print the data frame in the console by typing `colleges_by_state`, and the width of the console determines how many variables are shown.
- To print all columns we can convert the results back to a `data.frame` using the `as.data.frame` function. Try running `head(as.data.frame(colleges_by_state))`.
- You can also use the viewer by running the command `View(colleges_by_state)`.
- Another option is to `select` a reduced number of columns to print.

8. On Your Own

1. Filter the rows for colleges in Great Lakes or Plains regions.
2. Arrange the subset from part #1 to reveal what school has the highest first-year retention rate in this reduced data set.
3. Arrange the subset from part #1 to reveal what school has the lowest admissions rate in this reduced data set.
4. Using the full data set, create a column giving the average cost of attendance in 2016, assuming that costs increased 3% for all institutions. Name this new column `cost2016`.
5. Using the full data set, summarize the distribution of total cost of attendance by region using the five number summary. Briefly describe any differences in total cost that you observe.

9. Additional Resources

- RStudio's data wrangling cheat sheet provides a nice summary of the functions in the `dplyr` package, including those covered in this tutorial.
- The introductory vignette to `dplyr` provides an example of wrangling a data set consisting of 336,776 flights that departed from New York City in 2013.
- Roger Peng's video overview of the `dplyr` package.