Minilab 2c Worksheet

Data Slicing and Dicing

We often want to extract particular rows and columns from a data table (tibble). The R tidyverse package "dplyr" makes this very easy.

1. What is dplyr?

The R tidyverse package "dplyr" (pronounced "DEE-ply-er") is a powerful R package to manipulate, clean and summarise structured data. In short, it makes data exploration and data manipulation easy and fast in R. The (very handy) RStudio Cheat Sheet "Data Transformation with dplyr" (orange coloured) is available at: https://rstudio.com/resources/cheatsheets/

When working with data you must figure out what you want to do, describe those tasks in the form of an R script, and execute the script. Remember that in the R tidyverse, data is stored in a *tibble* (tidy table). Each row of the tibble is a "case" or "observation" and each column of the tibble is a "feature" or "variable".

We often wish to extract data from the tibble, i.e., only those rows and columns that we wish to work further with. One great advantage of dplyr is that we can extract rows based on logical criteria and columns based on column names.

Also, all the same functions can be used to access data from a database using the tidyverse package dbplyr.

(1) We will work with an example dataset which contains all the 336776 flights that departed from New York City in 2013.

```
install.packages("nycflights13")    # run only once
library(tidyverse)
library(nycflights13)
dim(flights)
flights
```

Take a look at the description of the variables in this dataset available from: https://www.rdocumentation.org/packages/nycflights13/versions/1.0.1/topics/flights

All of the functions described below have a similar syntax. The first argument is a dataset (tibble). The subsequent arguments describe what to do with the dataset, using the variable/column names (without quotes). The result returned by each function is a new tibble, i.e., a **copy** of the original flights dataset with the requested change made. When we Run our R code (in RStudio we highlight the code in the editor pane and hit the Run button), the output in the R console pane shows the **new** tibble (nicely formatted).

2. Extracting columns (variables)

(1) Often you work with large datasets with many columns but only a few are actually of interest to you.

```
names(flights)
select(flights, year, month, day)
select(flights, year:day)
select(flights, -year)
select(flights, -(year:day))
```

There are a number of "helper functions" you can use with select().

```
select(flights, starts_with("sched"))
select(flights, ends_with("delay"))
select(flights, contains("arr"))
select(flights, matches(".*_.*_.*"))    # regular expression matching
select(flights, last_col())
```

There are more helper functions, such as all_of() and any_of(), and it is possible to take union, intersection and differences of different operations. Have a look at https://www.rdocumentation.org/packages/tidyselect/versions/1.1.0/topics/language for further examples using the "starwars" and "iris" datasets.

(2) Once particular columns have been extracted, it may be useful to remove duplicate rows.

```
dim(flights)
dim(distinct(flights))
distinct(select(flights, month, day))
distinct(flights, month, day)    # does the same as line above
```

(3) Using select() to extract columns (variables) from a tibble always returns a new tibble. But sometimes we wish to extract a column of data as a *vector*.

```
sat = pull(flights,sched_arr_time)
sat
summary(sat)
```

(4) You can also change the name of a column.

```
rename(flights, destination=dest, tail_num=tailnum)
```

Note that this does not change the flights tibble, but rather makes a copy with new column names.

3. Adding new columns (variables)

(1) We can create new columns that are calculated from existing columns. From https://www.rdocumentation.org/packages/nycflights13/versions/1.0.1/topics/flights we can see that arr_delay, dep_delay and air_time are all measured in minutes and distance is measured in miles. Hence speed (calculated below) will be in miles per hour.

```
mutate(flights, gain=arr_delay-dep_delay, speed=distance/(air_time/60))
```

It is possible to use the first new variable in subsequent calculations on the same line.

See https://r4ds.had.co.nz/transform.html#mutate-funs (Section 5.5.1 of R for Data Science) for how to use various operators and functions when creating new variables.

(2) Sometimes you only want to keep the new variables.

```
transmute(flights, gain=arr_delay-dep_delay, gain_per_hour=gain /(air_time/60))
transmute(flights, distance_residual=distance-mean(distance))
```

4. Extracting Rows (cases)

(1) We can select particular rows by row number.

```
slice(flights, 87:96)   # particular 10 rows
top_n(flights, 20)
```

This is quite restrictive since we need to know exactly which row numbers we wish to select, and hence we would have to look at each row of data to decide whether or not to keep it. Note that top_n() will include more than n rows if there are ties.

(3) Often we select rows according to some *condition* which involves the values in some of the columns (variables).

```
filter(flights, month==1, day==1)
```

The result is a new tibble which has all the same columns (variables) as flights but only those rows for which both the logical conditions are TRUE.

There are many further examples of comparison operators (<, >, <=, >=) and logical operators (!, &, | for not, and, or) in Section 5.1 of R for Data Science: see https://r4ds.had.co.nz/transform.html#filter-rows-with-filter

(4) It is also possible to take a random sample of rows. It is good practice (debugging, scientific repeatability) to set the seed for the random number generator.

```
set.seed(123)
sample_n(flights, 20)
0.001*nrow(flights)  # how many rows expected if 0.1% of the rows
sample_frac(flights, 0.001)
```

(5) Suppose we wish to reorder (sort) the data by the content of particular columns. The first argument is the tibble and the subsequent arguments are column names (without quotes) or more complicated expressions. If you provide more than one column name, each additional column will be used to break ties in the values of preceding columns.

```
arrange(flights, year, month, day)
arrange(flights, desc(arr_delay)) # descending order
```

Exercise. Firstly, check the documentation at https://www.rdocumentation.org/packages/nycflights13/versions/1.0.1/topics/flights for format and units of the variables in the *flights* dataset.

Now, find all flights (rows) that:

(a) Departed between midnight and 6am (inclusive).

(b) Had an arrival delay of two or more hours.

(c) Flew to Houston (IAH or HOU).

(d) Were operated by United (UA), American (AA), or Delta (DL).

(e) Arrived more than two hours late, but didn't leave late.

(f) Were delayed by at least an hour, but made up over 30 minutes in flight

Summary

In this minilab, we have looked at slicing and dicing a dataset using the tidyverse R package dplyr. The main functions that we have seen so far are:

- **select()** to select columns in different ways
- **mutate()** to create new columns
- **filter()** to select rows by some criterion
- **arrange()** to sort the rows

The (very handy) RStudio Cheat Sheet "Data Transformation with dplyr" (orange coloured) is available at: https://rstudio.com/resources/cheatsheets/.