TUTORIAL 2

Aim:

Practice data wrangling / manipulation using the `dplyr` package.

Mode of study:

You can work through this tutorial in your own time, or in one of the assigned venues during the tutorial slot. Help is available during the tutorial slot in the venues.

Solutions will be made available in Week 3.

Before you start:

Load the `tidyverse` packages into R.

When loading the `tidyverse`, you might have noticed that you get a message about conflicts which occur between the `dplyr` package and the `stats` package (which is part of base R). Both these packages have a `filter()` function, and the message informs you that the `dplyr` `filter()` function is masking the `stats` `filter()` function. In other words, if you use `filter()`, R will automatically use the `dplyr` version.

This is exactly what we want in this module, but do take note that if you ever want to use the `stats` version of `filter()` after you've loaded `dplyr`, you will have to explicitly do so by calling `stats::filter()`.

Additional comment:

Your A1 assessment will require you to make extensive use of these `dplyr` functions, so make sure you practice enough!

INTRODUCTION

There are many data sets available within R – some in the base R installation, and some in packages. In this tutorial, we will look at 2 data sets: `iris` and `diamonds`.

The `iris` data set is older and smaller; in fact, it is one of the most widely used data sets in the field of statistics and is often used when teaching classification. We will encounter this data set again in a few weeks' time.

Working with smaller data sets can be useful for various reasons; however, I would like you to get used to working with larger data sets as well. For this reason, we will also look at the `diamonds` data set, which is part of the `ggplot2` package. While not huge by today's "big data" standards, the `diamonds` data set is still substantially larger than the `iris` data set.
Look at the help documentation for both of these data sets, to get to know the data and what the various variables represent.
[If you can't remember how to do that, refer back to your notes from Tutorial 1.]

IRIS DATA

1. The `iris` data set – as it is made available in R – is a data frame.

1.1 **Check the number of rows and columns in this data set.**

1.2 **Print the first few rows of the data set**

1.3 **View the data types of the variables.**

Solutions to the tutorial will be made available next week. You will note that I do not provide you with the code to run; everything you need to know to answer the questions in this tutorial was covered in Tutorial 1 as well as Practicals 1 and 2, so you can refer back to those if you cannot remember how to do something. If you require anything not explicitly covered before, I will provide you with either the code to use, or the name of the function required.

---

2. As discussed in Practical 2, tibbles are often more convenient to work with than data frames. You can convert a data frame to a tibble using the `as_tibble()` function.

2.1 Convert the `iris` data frame to a tibble, and name it `my_iris`.

2.2 Print your `my_iris` tibble and compare it to the results obtained in 1.3. Also view the `iris` data frame in the RStudio viewer.

You should use the `my_iris` data set to answer the following questions, unless explicitly stated otherwise.

For the following questions, you should use the `dplyr` functions encountered in Practical 2.

3. **Select only the versicolor irises, and store this in a new object called `versicolor`.**

4. **Select all irises with a sepal length greater than 5.**
   (You don't have to store the results in a new object.)

5. **Select all versicolor irises with a sepal length greater than 5.**

6. **Sort the entire data set according to sepal length, in descending order.**

7. **Create a new data set called `iris_petal` which contains only the species of iris and the petal dimensions (i.e .excluding all the sepal dimensions).**

8. Using this new data set `iris_petal`, create a new variable called `total_petal` which is the sum of the petal length and the petal width.

9. Suppose you are interested to know whether there are differences in the average dimensions of the different iris species. **Using the `dplyr` verbs and the pipe operator, write code to help you determine this.** (You should use the full `my_iris` data set for this.)

DIAMONDS DATA

Now we consolidate our learning by looking at a different data set. Write suitable code to help you answer the following questions:

10. How many diamonds are there in each of the different cut categories?

11. What is the average carat of the diamonds in the data set?

12. What is the largest diamond (in terms of carat) in the data set?

13. Check whether there is a difference in price for diamonds of different colour.

14. Explore the relationship between price and cut.

15. Examine the *x*, *y*, and *z* dimensions of the diamonds in the data set. You should do this by extracting summary statistics for these variables. Do you see anything strange? Based on your findings, create a new data set called `my_diamonds` which excludes any unusual observations.

16. What is the average price and average carat of the diamonds in your new data set?

17. **For your new data set, explore the relationship between price and carat.**
[Note that these are both numerical variables. You might consider dividing the diamonds into groups based on their carat…]

After completing this week's practical and tutorial you should be able to:

* Subset data sets by row

* Reorder rows of a data set

* Subset data sets by column

* Create new variables with functions of existing variables

* Summarise data sets by group
