Contents

This is the draft document for in-session slides.

Pre-requisites

- Downloading and installing R packages
- What is a data frame
- Principles of Tidy Data

1 dplyr : Grammar of data manipulation

- dplyr is mainly authored by Hadley Wickham and Romain Francois. It is designed to be intuitive and easy to learn, thereby making doing things in `R` more user friendly.
- dplyr is a new package which provides a set of tools for efficiently manipulating datasets in `R`.
- dplyr is the next iteration of plyr, focussing on only data frames.
- dplyr is faster, has a more consistent API and should be easier to use.

1.1 dplyr : abstract by Hadley Wickham

There are three key ideas that underlie dplyr:

1. Your time is important, so Romain Francois has written the key pieces in Rcpp to provide blazing fast performance. Performance will only get better over time, especially once we figure out the best way to make the most of multiple processors.
2. Tabular data is tabular data regardless of where it lives, so you should use the same functions to work with it.

   With dplyr, anything you can do to a local data frame you can also do to a remote database table. PostgreSQL, MySQL, SQLite and Google bigquery support is built-in; adding a new backend is a matter of implementing a handful of S3 methods.
3. The bottleneck in most data analyses is the time it takes for you to figure out what to do with your data, and dplyr makes this easier by having individual functions that correspond to the most common operations group_by, summarise,mutate, filter, select and arrange). Each function does one only thing, but does it well.

Author: Hadley Wickham

1.2 Working with dplyr

dplyr focussed on tools for working with data frames (hence the d in the name). **dplyr** has three main goals:

- Identify the most important data manipulation tools needed for data analysis and make them easy to use from `R`.
- Provide very fast performance for in-memory data by writing key pieces in C++.
- Use the same interface to work with data no matter where it's stored, whether in a data frame, a data table or database.

1.3 Installing dplyr

You can install the latest released version from CRAN with the code below. You can also install and load the data packages used in most examples:

```
install.packages("dplyr")
install.packages(c("nycflights13", "Lahman"))

library(dplyr) # for functions
library(nycflights13) # for data
```

1.4 Tidy Data

To make the most of dplyr, Hadley Wickham recommends that you familiarise yourself with the **principles of tidy data**. This will help you get your data into a form that works well with **dplyr**, **ggplot2** and `R`'s many modelling functions.

Three Principles from Hadley Wickham's paper

1. Each variable forms a column,
2. Each observation forms a row,
3. Each table/file stores data about one kind of observation.

Remark: The paper "***Tidy data***" by Hadley Wickham (RStudio) can be downloaded from

`http://vita.had.co.nz/papers/tidy-data.pdf`

1.5 Key data structures

The key object in **dplyr** is a `tbl`, a representation of a tabular data structure. Currently dplyr supports:

- data frames - the most commonly encountered R data structure.
- data tables - a data structure that is designed for intensive data analysis.

For this class, We will concentrate on **dplyr** exercises with data frames mostly. However we would advise you to try and learn more about working with data tables in the future.

For advanced users, **dplyr** also supports the following databases: *SQLite, PostgreSQL, Redshift, MySQL/MariaDB, Bigquery, MonetDB* and data cubes with arrays (partial implementation). We will not cover those topics in this workshop.

2 Worked Examples

New York Flights Exercise

To explore the basic data manipulation verbs of dplyr, well start with the built in *nycflights13* data frame. This dataset comes from the US Bureau of Transporation Statistics contains information on flights that departed from New York City in 2013.

```
library(dplyr) # for functions
library(nycflights13) # for data
data(flights)
```

Preliminary Exercises

Using simple data inspection functions, answer the following questions.

- How many columns (i.e. variables) are there in this data set?
- How many rows (i.e. cases) are there?
- Which column has the most missing values?
- What are names of the last three variables?
- What type of data object is "flights"?

```
#>    year month day dep_time dep_delay arr_time arr_delay carrier tailnum
#> 1  2013     1   1      517         2      830        11      UA  N14228
#> 2  2013     1   1      533         4      850        20      UA  N24211
#> 3  2013     1   1      542         2      923        33      AA  N619AA
#> 4  2013     1   1      544        -1     1004       -18      B6  N804JB
#> ..  ...   ... ...      ...       ...      ...       ...     ...     ...
```

2.1 Converting to tbl

dplyr can work with data frames as is, but if youre dealing with large data, its worthwhile to convert them to a `tbl_df`: this is a wrapper around a data frame that wont accidentally print a lot of data to the screen.

2.2 Filter rows with `filter()`

`filter()` allows you to select a subset of the rows of a data frame. The first argument is the name of the data frame, and the second and subsequent are filtering expressions evaluated in the context of that data frame: For example, we can select all flights on January 1st with:

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

2.3 group_by()

The `tbl` also comes in a grouped variant which allows you to easily perform operations "by group". Suppose we wish to group by carrier.

```
carriers_df  <- group_by(flights, carrier)
```