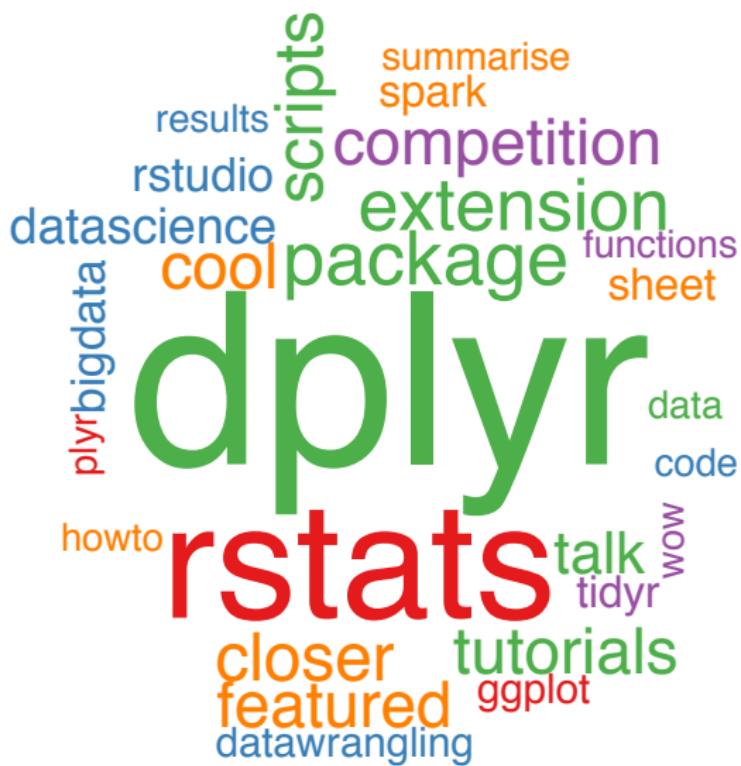


Data manipulation: ICCB 2015

Olivia Burge

21 July 2015



General structure

- ① Concept explanation
- ② Example code explanation
- ③ Class exercise
- ④ Individual task

I assume that you:

- have the most up to date R & Rstudio installed
- know how to install and load packages
- know how to set your working directory
- know that you can work in the console or work in a script
- have a rough idea of what a factor is compared to a numeric variable (for example)

I assume that you:

- know what format a csv needs to be in to import it into R
- know that a dataframe is roughly equivalent to a .csv file
- can read in data using `read.csv()`
- know what an object is

My aim is that you:

- **Understand** `dataframe[,]` & `dataframe$column` notation
- Can **subset** dataframes with `filter()` and `select()`, and order them with `arrange()`
- Can **chain** (aka pipe) commands using `%>%`
- Can **summarise** dataframes, and summarise **grouped** dataframes using `summarise` and `group_by()`
- Can **join** dataframes using `inner_join` & `left_join`
- Can **manipulate the format** of data (long-to-wide and wide-to-long) using `tidyverse`

Actually a lot of options:

- Various functions in base R
- data.table
- plyr/reshape2
- other packages

Benefits of dplyr:

- **not** speed (data.table can be worth it on big data)
- but **intuitive**

- No attach()
- No file.choose()

Data for today:



```
getwd()  
  
weta_data <- read.csv("data/weta_270.csv")
```

Square bracket notation - concept

- Dataframe[,] notation
- Dataframe[rownumber, colnumber]
- Dataframe[3, 4] is equivalent to the excel reference cell D3

Square bracket notation - examples

- Dataframe[2, 3]
- Dataframe[1:5, c(5:7, 9)]

```
weta_data[ , 5] # fifth column
```

```
weta_data[5]      # fifth column  
#(not using this form today)
```

```
weta_data[ 2:4, ]
```

Square bracket notation - class exercise

We've just been told that the "average.other.dbh" value in line 67 is wrong, and is actually 18.83. We don't change the raw data, but want to change in R before proceeding.

- ① Select the cell by using square bracket notation and print to screen (it should read 8.83)
- ② Amend it using the assign (<-) function to 18.83

Square bracket notation - class exercise - answers

```
# checking which cell? (it should be 8.83)
weta_data[ 67, 8 ]  
  
## changing the error to 18.83
weta_data[ 67, 8 ] <- 18.83  
  
# checking that it has changed
weta_data[ 67, 8 ]
```

Square bracket notation - individual exercise

You are interested in whether weta presence varies with distance to edge canopy cover.

- ① Assign the columns motel, weta.presence2, and canopy.cover to a new dataframe called “weta_dist”

Square bracket notation - individual answers

```
names(weta_data) # we want cols 14, 5, 7  
weta_dist <- weta_data[ , c(1, 5, 7, 14)]  
  
head(weta_dist, 2)
```

- Dataframe\$column

Same as selecting an entire column with the mouse in excel.

e.g.

```
weta_dist$distance.to.edge
```

Dollarsign notation - use case

You will often see it used to create variables quickly. We will cover this in dplyr too.

e.g.

```
weta_dist$distance.log <- log(weta_dist$distance.to.edge)
```

Key base R knowledge: summary

We can use:

- ① The square bracket notation to select either rows, **or** columns; and
- ② Dataframe\$column notation to select columns.

- Questions?
- Anyone unable to install and
‘require(dplyr)’ or
‘require(tidyr)’?

Enough weta



Subsetting dataframes

Manuka data



Subsetting dataframes

```
require(dplyr)
require(tidyr)
manuka_data <- read.csv("data/manuka.csv")
head(manuka_data, 2)
```

Select chooses certain columns - to keep, or to get rid of. The format is

```
DATANAME %>% select(c(col1, col2, col3))
```

or

```
DATANAME %>% select(col1:col4)
```

```
# we can also rename cols on the fly:  
manuka_data %>% select(Plot = plotnumber, Age)  
  
# selecting all cols between plotnumber and GW  
manuka_data %>% select(c(plotnumber:GW))  
  
# dropping columns  
manuka_data %>% select(-c(Height:MV))
```

What is this? %>% %>% %>%



What is this? %>% %>% %>%

Chaining allows us to write code in the order we want it done. Otherwise, it must be wrapped in brackets with the first thing to be done right in the middle.

```
manuka_data %>% select(c(Height, CO))
```

means, take the manuka dataframe, and then select the columns Height and CO.

Filter selects rows in your dataframe, based on the conditions you specify. Same format as for select():

```
filter(DATA, CONDITION1)
```

- row only selected if condition satisfied

```
filter(DATA, CONDITION1 & CONDITION2)
```

- both conditions must be satisfied

```
filter(DATA, CONDITION1 | CONDITION2)
```

- one or both conditions must be satisfied

```
levels(manuka_data$Site)
manuka_data %>% filter(Site == "Clearwater")
```

```
manuka_data %>% filter(Age > 32)
manuka_data %>% filter(Age > 32 & Site == "Lawsons")
manuka_data %>%
  filter(Age > 32 | ( Site == "Clearwater" & Age > 28 ))
```

`arrange(COLUMN)` orders the dataframe rows by the specified values

```
manuka_data %>% arrange(CO)
```

#using desc() reverses the order

```
manuka_data %>% arrange(desc(CO))
```

```
manuka_data %>% arrange(desc(Site))
```

Select, filter & arrange - class exercise

We create a new dataframe which contains only sites Clearwater and Tiwai, and only the columns Site, Height, Age. We arrange it by Age.

Select, filter & arrange - class exercise

We need to add an extra step here.

```
manuka_data %>%  
  filter(Site %in% c("Clearwater", "Tiwai")) %>%  
  select(Site, Height, Age)
```

Select, filter & arrange - individual exercise

- ① Please create a dataframe with GW (groundwater) less than 50 and the columns CH and CO.
- ② Using the weta dataset, please select the rows which have either a cave weta (weta.cave) present, or more than two weta in total (total.weta). Please select the columns motel, total.weta.inlcave and canopy.cover, but rename motel to mansion. Please arrange it by mansion.

Select, filter & arrange - individual exercise answers (1)

```
answer1 <- manuka_data %>%  
  filter(GW < 50) %>%  
  select(CH, CO)
```

Select, filter & arrange - individual exercise answers (2)

```
answer2 <- weta_data %>%
  filter(weta.cave == 1 | total.weta.inclcave > 2) %>%
  select(mansion = motel,
         total.weta.inclcave,
         canopy.cover) %>%
  arrange(mansion)
```

Creating new variables: mutate

We use `mutate` to create new variables (or columns) that are the same length as our existing dataset. For example, you may wish to:

- transform data (log, squareroot, etc)
- change the units (cm to km, or feet to miles)
- modify an existing variable based on another existing variable (e.g. height/age).

We create a variable that converts height to height in centimetres, and then standardise it by age. We also create a column which takes the log of canopy openness (CO).

```
manuka_data %>%  
  mutate(log_co = log(CO),  
        height_cm = Height * 100,  
        height_age = height_cm/Age) %>%  
  select(CO, log_co, Height, height_cm, Age, height_age)
```

In the one dataframe:

- ① Please firstly use filter to select only the records in which spiders were absent (spider.presence).
- ② Please create a column which adds the number of cave weta (weta.cave) to the number of total weta (total.weta).
- ③ Please also create a column which is the DBH of the host tree divided by the DBH of surrounding trees (dbh.host divided by the average.other.dbh).

Mutate - class exercise

```
weta_data %>%
  filter(spider.presence == 0) %>%
  mutate(all_the_weta = weta.cave + total.weta,
        dbh_index = dbh.host/average.other.dbh)
```

A short break



Summarising data means taking various statistics of the existing dataframe. When used by itself, `summarise` returns a statistic which you have chosen, with a name of your choosing.

```
data %>% summarise(NEWNAME = FUNCTION(COLUMN))
```

Summarising by group - concept

Summarising by group returns one summary statistic for each group, rather than for the entire column. We use the `group_by` command to tell R which column we want to group on.

Format:

```
data %>%  
  group_by(COLUMN) %>%  
  summarise(NEWNAME = FUNCTION(COLUMN))
```

Summarising, and summarising by group - examples (1)

```
manuka_data %>%  
  summarise(mean_age = mean(Age),  
            n = n())
```

```
manuka_data %>%  
  group_by(Site) %>%  
  summarise(mean_age = mean(Age))
```

Summarising, and summarising by group - examples (2)

```
manuka_summarise <- manuka_data %>%
  group_by(Site) %>%
  summarise(mean_age = mean(Age),
            se_age = sd(Age)/sqrt(length(Age)),
            mean_CO = mean(CO),
            se_CO = sd(CO)/sqrt(length(CO)),
            min_MV = min(MV))

write.csv(manuka_summarise,
          file = "data/manuka_summarise.csv",
          row.names = FALSE)
```

In one dataframe:

- ① Please calculate the mean canopy cover for all motels.

In another dataframe:

- ① Calculate the mean canopy cover for motels with and without weta present (weta.presence).
- ② Please also report the number of observations per group.

Summarising: group exercise

```
weta_data %>% summarise(mean_cc = mean(canopy.cover),  
                           n = n())  
  
weta_data %>% group_by(weta.presence) %>%  
  summarise(mean_cc = mean(canopy.cover),  
            n = n())
```

Summarising: individual exercise

- ① Please calculate the mean and standard deviation of the distance to edge, grouped by the number of cave weta (cave.weta.presence) in motels.
- ② Please also report the number of observations per group.

Summarising: individual exercise - answers

```
weta_data %>%
  group_by(cave.weta.presence) %>%
  summarise(mean_weta = mean(total.weta),
            sd_weta = sd(total.weta),
            n_obs = n())
```

Slice:

- ① `slice(1)` slices the first occurrence.
- ② `slice(1:3)` slices the first three instances.

More useful than you might think.

```
require(lme4)
require(datasets)
data(sleepstudy)
head(sleepstudy)

# what is the minimum reaction time for each person?
sleepstudy %>% group_by(Subject) %>%
  summarise(min_reaction = min(Reaction))
```

But what if we want to know what is the minimum reaction person after day 3, taking the later day if days are tied (same reaction time)?

```
sleepstudy %>%
  filter(Days > 3) %>%
  group_by(Subject) %>%
  arrange(Reaction, desc(Days)) %>%
  slice(1)
```

Joining allows us to combine two dataframes into one. Useful for analysis and plotting. dplyr has a number of joins available, depending on what you want to do.

In ecology, this often occurs where we have **environmental data for each plot** and also **species data**, and the species data may be a different length to the plot data.

Please read in the following data:

```
species_270 <- read.csv("data/veg_species.csv")
env_270 <- read.csv("data/veg_environment.csv")

head(species_270)
head(env_270)
glimpse(species_270)
glimpse(env_270)
```

Quick break



Have a look at ?inner_join for all examples. We will need an example which returns all the species rows, but duplicates the environmental variables where necessary. So we'll use left_join.

```
left_join(x = DATA1, y = DATA2)
left_join(x = DATA1, y = DATA2, by = JOINING_VAR)
```

We are joining the species data (1008 rows = 36 sites * 28 species) to the environmental data (36 rows = 1 per site).

```
combined_270 <- left_join(species_270, env_270)
```

```
## Joining by: "plot"
```

```
combined_270 %>% group_by(species, site) %>%
  summarise(mean_cover = mean(cover),
            min_cover = min(cover),
            max_cover = max(cover)) %>%
ungroup() %>%
group_by(site) %>%
arrange(desc(mean_cover)) %>%
slice(1:3)
```

It is generally recommended to have data in 'long form' in R (the form in which the species data was in). However, sometimes we may want it in wide form, particularly for ordinations.

Key terms:

- ① **Key** is the variable that describes the group (e.g. "species")
- ② **Value** is the variable which describes what you've measured (e.g. "cover", or "proportion_alive", etc)

Data format - example code (1)

To go from long-to-wide, we use `spread()`

```
head(species_270)
species_270_wide <- species_270 %>%
  spread(key = species, value = cover)
head(species_270_wide)
```

Data format - example code (2)

To go from wide-to-long, we use gather()

```
weta_wide <- weta_data %>%
  select(motel,
         weta_tree = weta.presence2,
         weta_cave = cave.weta.presence,
         spider = spider.presence,
         cockroach = cockroach.presence)
head(weta_wide)

weta_long <- weta_wide %>%
  gather(key = species,
         value = presence,
         weta_tree:cockroach)
head(weta_long)
```

- ① Please read in the mites dataset (from package vegan, with one amendment)

```
mites <- read.csv("data/vegan_mites.csv")
head(mites)
```

- ② Please turn the data from wide format (currently) to long format. Please call the columns 'species' and 'count'.
- ③ Please summarise by species the mean count.
- ④ Please select the 10 most numerous species, based on mean count.

Data format - individual exercise - answers (1)

- Please turn the data from wide format (currently) to long format. Please call the columns 'species' and 'count'

```
mites_wide <- mites %>%
  gather(key = species,
         value = count,
         Brachy:Trimalc2)
head(mites_wide)
```

Data format - individual exercise - answers (2)

Please summarise by species the mean count

```
mites_wide %>%
  group_by(species) %>%
  summarise(mean_count = mean(count))
```

Data format - individual exercise - answers (3)

- Please select the 10 most numerous species, based on mean count.
- This requires us to arrange by mean_count (descending order, then slice the top ten)

```
mites_wide %>%
  group_by(species) %>%
  summarise(mean_count = mean(count)) %>%
  arrange(desc(mean_count)) %>%
  slice(1:10)
```

And now

Questions?

interactors

new seeing
following hope littlemix
via tweet hey soon
ever insight heart wellstoryeveryday
mean today sharing couldve just id
never share done day read get word wow
dinahjane now miss create everything get tons
fans harrystylesplsamp can beating follow
back need

thanks

respectful

good check best love follow many you've know
got making shk much make end still video
harry happy please sos life support everyone
part see times guys people bro top
help monday ariana grande gift
person you my niallofficial
inspiring will like coming
bother towards

Please do!

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit
<http://creativecommons.org/licenses/by-nc-sa/4.0/>.