R wizardry course Week 4, 2017

Week 4

4.1 Finish week 3 materials (create an R project, random numbers and strings). Please remit to the week 3 rmd file, section 1.6 and on.
4.2 Brief intro to RMarkdown 4.3 Dplyr and its verbs

4.2 Brief intro to RMarkdown

4.2.1. Create an rmd file from scratch
4.2.2. Set up chunck
4.2.3. Set up inside individual chuncks 4.2.4. Rendering to PDF, doc, html, slideshow

4.3 Dplyr

Incorporates the use of UNIX's pipes (%>%) in R so you build "pipelines" of code.

Verbs

```
library(dplyr)
compounds <- read.csv("/home/oscar/Dropbox/R wizardry/2017/Datasets/compounds_triplicates_long.csv")

str(compounds)
```

Summarise

The `summarise` function is used to create statistical summaries of your data. Most of the time it is used in combination with `group_by`. Let's see how this works.

Let's find the variance among methane formation grouped by salinity:

```
compounds %>%
    group_by(compound) %>%
    summarise(var = var(mean_methane, na.rm = TRUE))

#Dplyr is very inclusive by taking into account the English-Australian and North American spelling of s
summarise
summarize
```

What about other stats?

```
compounds %>%
    group_by(salinity, compound, day) %>%
    summarise(mean = mean(mean_methane, na.rm = TRUE),
              median = median(mean_methane, na.rm = TRUE),
                        max = max(mean_methane, na.rm = TRUE),
                        min = min(mean_methane, na.rm = TRUE))
```

Adding calculations inside summarise, though, will bring errors to the code. Let's try to calculate the standard error of the mean:

```
#First, make sure the calculation works by itself:
compounds$sd_methane/sqrt(length(compounds$mean_methane))

#Now, include the code (note the use of periods to indicate the dataset):
compounds %>%
    group_by(salinity, compound, day) %>%
    summarise(se = .$sd_methane/sqrt(length(.$mean_methane)))
```

Mutate

Mutate is similar to summarise, however, it "mutates" a column instead of summarising it. Summarise takes a bunch of values and produces one value from those. Mutate returns a transformed vector that is the same size as the original. Usually we're not grouping but you can if needed.

Let's use mutate here to transform one of our variables to log scale. A new column will be added to the object "compounds" without overwritting it:

```
#
log10_compounds <- compounds %>% mutate(mean_methane_log10 = log10(mean_methane))
## Warning in eval(substitute(expr), envir, enclos): NaNs produced

#Notice warning
```

Look at the transformed variable. What's the problem here?

```
log10_compounds$mean_methane_log10

#NaN (not a number, or the result of 0 divided by 0, or the log10 of a negative number) versus NA (not
```

```
log10_compounds2 <- compounds %>% mutate(mean_methane_log10 = log10(mean_methane + 10))

#Another option would be selecting data greater than 0 but you'll lose data...
log10_compounds3 <- compounds %>% mutate(mean_methane_log10 = log10(mean_methane > 0))
```

Alternative way to do the same thing, although note that this modifies our dataframe in-place instead of returning a copy.

```
compounds$log10_mean_methane <- log10(compounds$mean_methane + 10) %>% round(2)
```

Transmute does the same as mutate but returns only the mutated column

```
compounds %>% transmute(log10_mean_methane = log10(compounds$mean_methane + 10)) %>% head(20)
```

Filter

`filter` returns only rows matching a particular condition

```
compounds %>% filter(compounds == "hexane") #Whats the error? Instead of a column, we specified a whole
compounds %>% filter(compound == "hexane")

#Numbers can be used with or without quotation marks
compounds %>% filter(day > 197)
compounds %>% filter(day < "197")
```

```r
#Can include several
compounds %>% filter(salinity == "brackish" | salinity == "saline", mean_methane >= 20)

#or simply exlude "fresh" (a bit of less typing...)

compounds %>% filter(salinity != "fresh", mean_methane >= 20)
```

In base R:

```r
compounds[compounds$salinity != "fresh" & compounds$mean_methane >= 20, ]

#Get rid of <NA> cells
na.omit(compounds[compounds$salinity != "fresh" & compounds$mean_methane >= 20, ])
```

Selecting columns

`select` gives us the columns we want. Use unqouted column names.

```r
compounds %>% select(day, sd_methane, group)
```

Columns are returned in the order we list them

```r
compounds %>% select(day, sd_methane, salinity, group, log10_mean_methane)
```

Dplyr provides helper functions for selecting columns. `?select_helpers`

```r
compounds %>% select(everything())
compounds %>% select(starts_with("s"))
compounds %>% select(ends_with("e"))
compounds %>% select(contains("LiNi")) #Is not case sensitive

# Move a column to the front
compounds %>% select(log10_mean_methane, everything()) %>% head(10) # or head(., 10)
```

Other "verbs"

`slice` gives us row indexing

```r
compounds %>%
    select(day, compound, salinity) %>%
    slice(1:10)

compounds %>%
    select(day, compound, salinity) %>% .[1:10, ] #The period is required here!
```

`arrange` orders the dataframe by a variable

```r
compounds %>% arrange(salinity)
compounds %>% arrange(desc(day), salinity) %>% head(55)
```

`distinct` gives us only the unique rows

```r
compounds %>% distinct(day) #Like unique()
compounds %>% distinct(salinity)
```

`count` and `tally` count occurances

```
compounds %>% count(day) #Number of observations per day
compounds %>% group_by(salinity, day) %>% tally() #Observations per salinity per day
```

`sample_n` and `sample_frac` (percentage) allow us to randomly sample rows

```
compounds %>% sample_n(5)
compounds %>% sample_frac(0.2)
```

Putting it together

Problem 1.

What are the maxima values in decreasing order, based on compound and salinity, for the mean of methane for days 102, 272, and 72?

```
compounds %>%
    filter(day == 102 | day == 272 | day == 72) %>%
    group_by(compound, salinity) %>%
    summarise(max = max(mean_methane, na.rm = TRUE)) %>%
    arrange(desc(max))

#or

compounds %>%
    filter(day %in% c("102", "272", "72")) %>% #shorter than typing "day == 102 | ...."
    group_by(compound, salinity) %>%
    summarise(max = max(mean_methane, na.rm = TRUE)) %>%
    arrange(desc(max))
```

Mean center (scale) the mean of methane.

```
compounds %>%
    group_by(compound) %>%
    mutate(centered = scale(mean_methane)) %>%
  slice(15:30)  %>% select(centered) #Times standard deviations each observation is above (positive) or
## Adding missing grouping variables: `compound`

#Scale = (x - mean(x)) / sd(x) , or simply the reminder divided by the sd. Allows standarization of the
```

Using only the treatment and unamended groups, create a new column called "transformation" that is the standarized values for the brackish observations only; values for any other salinity on that new column most be NAs. Addtionally, create a column called "pos_or_neg" that will have a negative sign if the transformatin value is le than 0 (zero) and positive sign for the values equal or greater than 0, preserving only numeric values (i.e. get rid of NAs).

```
?scale
compounds %>%
  filter(group %in% c("treatment", "unamended")) %>%
    mutate(transformation = if_else(salinity == "brackish", scale(mean_methane, scale=FALSE), NA_real_))
    mutate(pos_or_neg = if_else(sign(transformation) < 0, "-", "+")) %>% na.omit()
```