

HW3: Data Manipulations and Packages

```
library(tidyverse)
library(palmerpenguins)
```

Task 1

Part A

The `read_csv` function is a specific use-case of the `read_delim` function that specifies that the delimiter must be a comma (,). The data that we're trying to read in here is delimited by semi-colons (;) so we cannot use the `read_csv` function as a result. The help file `?read_csv` specifies that `read_csv2` supports semi-colons (;) so we can use that function instead.

```
#Read in and display data.txt
data <- read_csv2(".*\\data\\.txt", col_names = TRUE,
                  show_col_types = FALSE)
```

i Using `"', '"` as decimal and `"'.'"'` as grouping mark. Use ``read_delim()`` for more control.

```
data
```

```
# A tibble: 2 x 3
      x     y     z
  <dbl> <dbl> <dbl>
1     1     2     3
2     5     3     8
```

Part B

```
#Read in and display data2.txt
data2 <- read_delim(".*\\data\\data2.txt", col_names = TRUE,
                    col_types = "fdc", delim = '6')
data2
```

```
# A tibble: 3 x 3
  x     y z
  <fct> <dbl> <chr>
1 1     2 3
2 5     3 8
3 7     4 2
```

Task 2

Part A

```
#Read in trailblazer data and take a look to make sure it read properly
trailblazer <- read.csv(".*\\data\\trailblazer.csv", header = TRUE)
glimpse(trailblazer)
```

```
Rows: 9
Columns: 11
$ Player      <chr> "Damian Lillard", "CJ McCollum", "Norman Powell", "Robert ~
$ Game1_Home  <int> 20, 24, 14, 8, 20, 5, 11, 2, 7
$ Game2_Home  <int> 19, 28, 16, 6, 9, 5, 18, 8, 11
$ Game3_Away  <int> 12, 20, NA, 0, 4, 8, 12, 5, 5
$ Game4_Home  <int> 20, 25, NA, 3, 17, 10, 17, 8, 9
$ Game5_Home  <int> 25, 14, 12, 9, 14, 9, 5, 3, 8
$ Game6_Away  <int> 14, 25, 14, 6, 13, 6, 19, 8, 8
$ Game7_Away  <int> 20, 20, 22, 0, 7, 0, 17, 7, 4
$ Game8_Away  <int> 26, 21, 23, 6, 6, 7, 15, 0, 0
$ Game9_Home  <int> 4, 27, 25, 19, 10, 0, 16, 2, 7
$ Game10_Home <int> 25, 7, 13, 12, 15, 6, 10, 4, 8
```

Part B

```
#Pivot the dataset into a longer format and separate
#games by Home and Away status
trailblazer_longer <- trailblazer |>
  pivot_longer(cols = 2:11,
               names_to = "game",
               values_to = "points") |>
  separate(game, into = c("game", "location"), sep = "_")

#Display results!
head(trailblazer_longer, n = 5)
```

```
# A tibble: 5 x 4
  Player      game location points
  <chr>      <chr> <chr>    <int>
1 Damian Lillard Game1 Home      20
2 Damian Lillard Game2 Home      19
3 Damian Lillard Game3 Away      12
4 Damian Lillard Game4 Home      20
5 Damian Lillard Game5 Home      25
```

Part C

```
#We wish to know who scored more when playing at home versus playing away

trailblazer_wider <- trailblazer_longer |>
  #Start with a wide pivot
  pivot_wider(names_from = location,
               values_from = points) |>
  #Group by players
  group_by(Player) |>
  #Add mean values for home and away scoring, then take the difference
  mutate(mean_home = mean(Home, na.rm = TRUE),
         mean_away = mean(Away, na.rm = TRUE),
         mean_diff = mean_home - mean_away) |>
  #Sort by descending mean difference
  arrange(desc(mean_diff)) |>
  #Subset to the variables we care about
  select(Player, mean_diff) |>
  #Only include distinct values
```

```
distinct(Player, .keep_all = TRUE)

#Display results!
trailblazer_wider
```

```
# A tibble: 9 x 2
# Groups:   Player [9]
  Player      mean_diff
  <chr>      <dbl>
1 Jusuf Nurkic      6.67
2 Robert Covington  6.5
3 Nassir Little     4.08
4 Damian Lillard     0.833
5 Cody Zeller       0.583
6 Larry Nance Jr    -0.5
7 CJ McCollum      -0.667
8 Anfernee Simons   -2.92
9 Norman Powell    -3.67
```

In the first 10 games of the 2021-2022 NBA season, the following players scored more points at home games than they did at away games, on average: Jusuf Nurkic, Robert Covington, Nassir Little, Damian Lillard, and Cody Zeller.

Task 3

Part A

```
#Incorrect pivot
incorrect <- penguins |>
  select(species, island, bill_length_mm) |>
  pivot_wider(
    names_from = island, values_from = bill_length_mm
  )
```

Warning: Values from `bill_length_mm` are not uniquely identified; output will contain list-cols.

- * Use `values_fn = list` to suppress this warning.
- * Use `values_fn = {summary_fun}` to summarise duplicates.

```
* Use the following dplyr code to identify duplicates.
{data} |>
dplyr::summarise(n = dplyr::n(), .by = c(species, island)) |>
dplyr::filter(n > 1L)
```

```
#Display
incorrect
```

```
# A tibble: 3 x 4
  species   Torgersen   Biscoe     Dream
  <fct>     <list>      <list>    <list>
1 Adelie   <dbl [52]> <dbl [44]> <dbl [56]>
2 Gentoo   <NULL>     <dbl [124]> <NULL>
3 Chinstrap <NULL>     <NULL>     <dbl [68]>
```

Notice that the output from dplyr is telling us that the measurements from `bill_length_mm` are not uniquely identified. This is because there are multiple measurements for each penguin species at each island, so the package doesn't know how we want the data to be handled in this instance. The solution that it defaults to is to store all of the measurements it can find as a `list` object in each cell of the tibble. For instance, notice that each column is of type `<list>`. This means that each observation in the column is a `list` object which itself is storing multiple observations. We see that there are a few different lists in the cells. For example, the observations for the Adelie penguins on Torgersen Island are stored as `<dbl [52]>`, which means that the list is storing observations in the double format, and that the list is has 52 observations. We also see that there are some cells with the observation `<NULL>`. These are combinations in the original dataset that had no observations, so there was no object to create. Since the package doesn't know what format to make the values, it just assumes they are null, and hence the null list.

Part B

```
#Correct pivot
penguins_correct <- penguins |>
  #Select relevant variables
  select(species, island, bill_length_mm) |>
  #Group variables by species and island
  group_by(species, island) |>
  #Count the frequency of each group in the dataset
  summarise(bill_length_mm = n()) |>
  #Pivot for a cleaner display
```

```

pivot_wider(names_from = island, values_from = bill_length_mm) |>
#Replace empty values with 0 for consistency
mutate(Biscoe = replace_na(Biscoe, 0),
       Dream = replace_na(Dream, 0),
       Torgersen = replace_na(Torgersen, 0))

```

`summarise()` has grouped output by 'species'. You can override using the
 `.groups` argument.

```

#Display results!
penguins_correct

```

```

# A tibble: 3 x 4
# Groups:   species [3]
  species Biscoe Dream Torgersen
  <fct>    <int> <int>    <int>
1 Adelie     44    56        52
2 Chinstrap    0    68         0
3 Gentoo    124     0         0

```