Into The Tidyverse

Mike Keating

Load Dependencies

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

```
Attaching package: 'palmerpenguins'
```

```
The following objects are masked from 'package:datasets':

    penguins, penguins_raw
```

Task 1

The data for this task is called data.txt and data2.txt. Download these and put them in your data folder before answering the questions below.

We can use read_csv functions to read in data. CSV is a comma-separated file i.e. any text file that uses commas as a delimiter to separate the record values for each field. Therefore, to load data from a text file we can use the read_csv() method (or versions of it), even if the file itself does not have a .csv extension.

In the following question, we are going to read in txt data. Part a has us working with the data.txt file. Part b has you working with the data2.txt file.

Part a

We cannot use read_csv() to read the data in data.txt because it uses a comma (‘,’) as the delimiter (the separating character between values). Instead, we must use read_csv2(), which uses a semicolon (‘;’) as its delimiter. This is helpful in reading data from European countries where a comma may be used as a decimal point and not as a field separator.

```
data <- read_csv2('data/data.txt', 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 data delimited by “6” and assign factor, double, and character as datatypes for each column.

```
data2 <- read_delim('data/data2.txt', delim = '6', col_types = 'fdc')
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

The Portland Trailblazers are a National Basketball Association (NBA) sports team. These data reflect the points scored by 9 Portland Trailblazers players across the first 10 games of the 2021-2022 NBA season. We are going to use these data to show off our data tidying skills. The data we will be using for this task is called trailblazer, and can be found on Moodle.

Part a

Take a glimpse of the trailblazer data set to show that you have read in the data correctly.

```
trailblazer <- read_csv('data/trailblazer.csv', show_col_types = FALSE)

glimpse(trailblazer)
```

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

Part b

Pivot the data so that you have columns for Player, Game, Location, Points. Display the first five rows of your data set. Save your new data set as trailblazer_longer. Your data set should contain 90 rows and 4 columns.

Let's get a glimpse at just the original column names:

```
colnames(trailblazer)
```

```
 [1] "Player"      "Game1_Home"  "Game2_Home"  "Game3_Away"  "Game4_Home"
 [6] "Game5_Home"  "Game6_Away"  "Game7_Away"  "Game8_Away"  "Game9_Home"
[11] "Game10_Home"
```

```
trailblazer_longer <- trailblazer |>
  pivot_longer("Game1_Home":"Game10_Home",
               names_to = c("Game", "Location"),
               names_sep = "_",
               values_to = "Points")
```

```r
# Show first 5 rows
print(head(trailblazer_longer, 5))
```

```
# A tibble: 5 x 4
  Player         Game  Location Points
  <chr>          <chr> <chr>     <dbl>
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
```

```r
# And checking dimensions
print(dim(trailblazer_longer))
```

```
[1] 90  4
```

Part c

Which players scored more, on average, when playing at home versus away? Answer this question using a single pipeline

```r
trailblazer_home_v_away <-
  trailblazer_longer |>
  pivot_wider(names_from = "Location", values_from = "Points") |>
  group_by(Player) |>
  mutate(mean_home = mean(Home, na.rm = TRUE),
         mean_away = mean(Away, na.rm = TRUE),
         diff_home_away = mean_home - mean_away) |>
  arrange(desc(diff_home_away))

trailblazer_home_v_away[8:11,]
```

```
# A tibble: 4 x 7
# Groups:   Player [2]
  Player           Game    Home  Away mean_home mean_away diff_home_away
  <chr>            <chr>  <dbl> <dbl>     <dbl>     <dbl>          <dbl>
1 Jusuf Nurkic     Game8     NA     6      14.2       7.5           6.67
2 Jusuf Nurkic     Game9     10    NA      14.2       7.5           6.67
3 Jusuf Nurkic     Game10    15    NA      14.2       7.5           6.67
4 Robert Covington Game1      8    NA       9.5       3             6.5
```

The following players scored more points on average at home than away:

```
trailblazer_home_v_away |>
  filter(diff_home_away > 0) |>
  distinct(Player) # Distinct gives us unique values in our df
```

```
# A tibble: 5 x 1
# Groups:   Player [5]
  Player
  <chr>
1 Jusuf Nurkic
2 Robert Covington
3 Nassir Little
4 Damian Lillard
5 Cody Zeller
```

Task 3

For the next tasks, we are going to use the penguins data set in the palmerpenguins package.

Problem a

```
# Provided erroneous code.
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)
```

```
# 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]>
```

This error occurs because each key (Island in this case) is associated with multiple values (bill_length_mm) and the function does not know which one to assign, so it stores all values in a list.

```
# Using the suggested code to identify duplicates
penguins |> summarize(n = n(), .by=c(species,island)) |> filter(n > 1L)
```

```
# A tibble: 5 x 3
  species   island        n
  <fct>     <fct>     <int>
1 Adelie    Torgersen    52
2 Adelie    Biscoe       44
3 Adelie    Dream        56
4 Gentoo    Biscoe      124
5 Chinstrap Dream        68
```

Explain what <NULL>, <dbl [52]>, and <list> mean:

<NULL>

In this case, <NULL> means that the given combination of species and island do not exist. For example, there are no penguins of the species "Chinstrap" on the island "Torgersen".

We can check this by attempting to filter by these values:

```
penguins |> select(species, island, bill_length_mm) |>
  filter(species == "Chinstrap", island == "Torgersen")
```

```
# A tibble: 0 x 3
# i 3 variables: species <fct>, island <fct>, bill_length_mm <dbl>
```

We returned a df with 0 rows (no results matching our criteria!) and 3 columns.

<dbl [52]>

Each observation for bill length where species is "Adelie" and island is "Torgersen" was combined into a single list of numbers (double).

```
penguins |> select(species, island, bill_length_mm) |>
  filter(species == "Adelie", island == "Torgersen") |>
  str()
```

```
tibble [52 x 3] (S3: tbl_df/tbl/data.frame)
 $ species       : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ island        : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ bill_length_mm: num [1:52] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
```

<list>

As mentioned above, the column was converted to the list datatype.

Part b

Create the table our colleague was trying to create:

```
# We coerced the columns into the datatype double to match the given output.
penguins |>
  select(species, island, bill_length_mm) |>
  group_by(species, island) |>
  summarize(n = n()) |>
  pivot_wider(names_from = island,
              values_from = n,
              values_fill = 0) |>
  mutate(Biscoe = as.double(Biscoe),
         Dream = as.double(Dream),
         Torgersen = as.double(Torgersen))
```

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

```
# A tibble: 3 x 4
# Groups:   species [3]
  species   Biscoe Dream Torgersen
  <fct>      <dbl> <dbl>     <dbl>
```

```
1 Adelie        44    56        52
2 Chinstrap      0    68         0
3 Gentoo       124     0         0
```

Task 4

Fill in the missing values:

```
# We will filter by na values first and then assign by species.
# Missing value for Gentoo is 30
# Missing value for Adelie is 26

penguins_filled <- penguins |> select(species, island, bill_length_mm) |>
  mutate(bill_length_mm = case_when(species == "Adelie" &
                                      is.na(bill_length_mm) ~ 26,
            species == "Gentoo" & is.na(bill_length_mm) ~ 30,
            .default = bill_length_mm
            ))
# Show first 10 rows.
head(penguins_filled, 10)
```

```
# A tibble: 10 x 3
   species island    bill_length_mm
   <fct>   <fct>              <dbl>
 1 Adelie  Torgersen           39.1
 2 Adelie  Torgersen           39.5
 3 Adelie  Torgersen           40.3
 4 Adelie  Torgersen           26
 5 Adelie  Torgersen           36.7
 6 Adelie  Torgersen           39.3
 7 Adelie  Torgersen           38.9
 8 Adelie  Torgersen           39.2
 9 Adelie  Torgersen           34.1
10 Adelie  Torgersen           42
```