Tidyverse Programming

```r
#Requiring libraries
library(tidyverse)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```
Warning: package 'purrr' was built under R version 4.3.3
```

```
Warning: package 'lubridate' was built under R version 4.3.3
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(palmerpenguins)
```

```
Warning: package 'palmerpenguins' was built under R version 4.3.3
```

Task One: Reading in the data

Question a: Reading in `data.txt` file

```
?read_csv
```

CSV stands for Comma Separated Values. The data contained in the file data.txt are not comma delimited (they are semicolon delimited), so therefore we cannot use the function read_csv to read in this data file.

```
data <- read_delim("data/data.txt", #name and path of the data file
                   delim = "; ", #setting "; " as the delimiter
                   col_types = "ddd" #assigning all columns as double
                   )
data #displaying the data
```

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

Question b: Reading in `data2.txt` file

```
data2 <- read_delim("data/data2.txt", #name and path of the data file
                   delim = "6", #setting "6" as the delimiter
                   col_types = "fdc") #assigning columns as x=factor, y=double, z=character
data2 #displaying the data
```

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

Task Two

Question a: Reading in the `trailblazer` data and using `glimpse` to check

```
trailblazer <- read_csv("data/trailblazer.csv") #using read_csv function to read in the data
```

```
Rows: 9 Columns: 11
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (1): Player
dbl (10): Game1_Home, Game2_Home, Game3_Away, Game4_Home, Game5_Home, Game6_...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(trailblazer) #glimpsing the data
```

```
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
```

Question b: Pivoting the data

```
trailblazer_longer <- trailblazer |> #starting with the original trailblazer dataset
  pivot_longer(cols = 2:11, #holding the Player column constant and pivoting the rest
               names_to = "GameLocation", #creating a temporary combined column of Game and L
               values_to = "Points") |>
  #Splitting the combined GameLocation column into two separate columns using the "_" delimit
  separate_wider_delim(cols = GameLocation,
                       delim = "_",
                       names = c("Game", "Location"))

glimpse(trailblazer_longer) #looking at the data
```

```
Rows: 90
Columns: 4
$ Player   <chr> "Damian Lillard", "Damian Lillard", "Damian Lillard", "Damian~
$ Game     <chr> "Game1", "Game2", "Game3", "Game4", "Game5", "Game6", "Game7"~
$ Location <chr> "Home", "Home", "Away", "Home", "Home", "Away", "Away", "Away~
$ Points   <dbl> 20, 19, 12, 20, 25, 14, 20, 26, 4, 25, 24, 28, 20, 25, 14, 25~
```

Question c: Finding whether players scored more during home or away games

```
trailblazer_scoring <- trailblazer_longer |>
  pivot_wider(names_from = Location, values_from = Points) |> #Creating a 90 x4 dataset
  group_by(Player) |> #grouping by Player
  mutate(mean_home = mean(Home, na.rm = TRUE)) |> #creating a column for mean home points
  mutate(mean_away = mean(Away, na.rm = TRUE)) |> #creating a column for mean away points
  mutate(mean_diff = mean_home - mean_away) |> #creating a column for the difference in home
  arrange(desc(mean_diff)) #sorting by descending mean difference
```

On average, Jusuf Nurkic, Robert Covington, Nassir Little, Damian Lillard and Cody Zeller scored more points in home games than away through the first 10 days.

Task Three: Manipulating the Penguins datasets

Question a: Reviewing a coworker's data pivot

`NULL` means that there aren't any of that penguin species found on that island because the list of bill lengths is empty.

`dbl [52]>` means that the list contains 52 elements and that the data type of the elements is double.

`<list>` means that elements of the Torgersen, Biscoe, and Dream are list structures.

Question b: Creating the correct penguins data table

```
penguins_correct <- 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)
```

Task Four: Replacing `NA` values in the penguins dataset

```
penguins |>
  mutate(bill_length_mm = case_when(is.na(bill_length_mm) & species == "Adelie" ~ 26,
                   is.na(bill_length_mm) & species == "Gentoo" ~ 30,
                   TRUE ~ bill_length_mm)) |>
  arrange(bill_length_mm)
```

```
# A tibble: 344 x 8
   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
 1 Adelie  Torgersen           26            NA                  NA          NA
 2 Gentoo  Biscoe              30            NA                  NA          NA
 3 Adelie  Dream               32.1          15.5               188        3050
 4 Adelie  Dream               33.1          16.1               178        2900
 5 Adelie  Torgersen           33.5          19                 190        3600
 6 Adelie  Dream               34            17.1               185        3400
 7 Adelie  Torgersen           34.1          18.1               193        3475
 8 Adelie  Torgersen           34.4          18.4               184        3325
 9 Adelie  Biscoe              34.5          18.1               187        2900
10 Adelie  Torgersen           34.6          21.1               198        4400
# i 334 more rows
# i 2 more variables: sex <fct>, year <int>
```

```
y <- seq(-2, 2, by = .5)
case_when(
  y >= 0 ~ sqrt(y),
  .default = y
)
```

```
Warning in sqrt(y): NaNs produced
```

```
[1] -2.0000000 -1.5000000 -1.0000000 -0.5000000  0.0000000  0.7071068  1.0000000
[8]  1.2247449  1.4142136
```

```
starwars %>%
  select(name:mass, gender, species) %>%
  mutate(
    type = case_when(
      height > 200 | mass > 200 ~ "large",
      species == "Droid" ~ "robot",
      .default = "other"
    )
  )
```

```
# A tibble: 87 x 6
   name               height  mass gender    species type
   <chr>               <int> <dbl> <chr>     <chr>   <chr>
 1 Luke Skywalker        172    77 masculine Human   other
 2 C-3PO                 167    75 masculine Droid   robot
 3 R2-D2                  96    32 masculine Droid   robot
 4 Darth Vader           202   136 masculine Human   large
 5 Leia Organa           150    49 feminine  Human   other
 6 Owen Lars             178   120 masculine Human   other
 7 Beru Whitesun Lars    165    75 feminine  Human   other
 8 R5-D4                  97    32 masculine Droid   robot
 9 Biggs Darklighter     183    84 masculine Human   other
10 Obi-Wan Kenobi        182    77 masculine Human   other
# i 77 more rows
```