CodingChallenge5_DataWrangeling_SK

Shakiba Kazemian

2025-03-20

Q1: Reading the CSV files using relative paths

```
# Loading necessary package
library(tidyverse)
```

```
## -- 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   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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 become errors
```

```
# Read the files
diversity_data <- read.csv("DiversityData.csv")
metadata <- read.csv("Metadata.csv")

# Check structure of the data
str(diversity_data)
```

```
## 'data.frame':    70 obs. of  5 variables:
##  $ Code      : chr  "S01_13" "S02_16" "S03_19" "S04_22" ...
##  $ shannon   : num  6.62 6.61 6.66 6.66 6.61 ...
##  $ invsimpson: num  211 207 213 205 200 ...
##  $ simpson   : num  0.995 0.995 0.995 0.995 0.995 ...
##  $ richness  : int  3319 3079 3935 3922 3196 3481 3250 3170 3657 3177 ...
```

```
str(metadata)
```

```
## 'data.frame':    70 obs. of  5 variables:
##  $ Code         : chr  "S01_13" "S02_16" "S03_19" "S04_22" ...
##  $ Crop         : chr  "Soil" "Soil" "Soil" "Soil" ...
##  $ Time_Point   : int  0 0 0 0 0 0 6 6 6 6 ...
##  $ Replicate    : int  1 2 3 4 5 6 1 2 3 4 ...
##  $ Water_Imbibed: chr  "na" "na" "na" "na" ...
```

Q2: Joining the two dataframes together by the common column 'Code'.

```
# Join the two dataframes by the common column "Code"
alpha <- left_join(metadata, diversity_data, by = "Code")

# Check the structure of the resulting dataframe
str(alpha)
```

```
## 'data.frame':    70 obs. of  9 variables:
##  $ Code         : chr  "S01_13" "S02_16" "S03_19" "S04_22" ...
##  $ Crop         : chr  "Soil" "Soil" "Soil" "Soil" ...
##  $ Time_Point   : int  0 0 0 0 0 0 6 6 6 6 ...
##  $ Replicate    : int  1 2 3 4 5 6 1 2 3 4 ...
##  $ Water_Imbibed: chr  "na" "na" "na" "na" ...
##  $ shannon      : num  6.62 6.61 6.66 6.66 6.61 ...
##  $ invsimpson   : num  211 207 213 205 200 ...
##  $ simpson      : num  0.995 0.995 0.995 0.995 0.995 ...
##  $ richness     : int  3319 3079 3935 3922 3196 3481 3250 3170 3657 3177 ...
```

```
# Display the first few rows
head(alpha)
```

```
##     Code Crop Time_Point Replicate Water_Imbibed  shannon invsimpson   simpson
## 1 S01_13 Soil          0         1            na 6.624921   210.7279 0.9952545
## 2 S02_16 Soil          0         2            na 6.612413   206.8666 0.9951660
## 3 S03_19 Soil          0         3            na 6.660853   213.0184 0.9953056
## 4 S04_22 Soil          0         4            na 6.660671   204.6908 0.9951146
## 5 S05_25 Soil          0         5            na 6.610965   200.2552 0.9950064
## 6 S06_28 Soil          0         6            na 6.650812   199.3211 0.9949830
##   richness
## 1     3319
## 2     3079
## 3     3935
## 4     3922
## 5     3196
## 6     3481
```

Q3: Calculating Pielou's evenness index: Pielou's evenness is an ecological parameter calculated by the Shannon diversity index (column shannon) divided by the log of the richness column.

```
# Create a new column for Pielou's evenness index
alpha_even <- alpha %>%
  mutate(Pielou_Evenness = shannon / log(richness))

# Check the structure of the new dataframe
str(alpha_even)
```

```
## 'data.frame':    70 obs. of  10 variables:
##  $ Code           : chr  "S01_13" "S02_16" "S03_19" "S04_22" ...
##  $ Crop           : chr  "Soil" "Soil" "Soil" "Soil" ...
##  $ Time_Point     : int  0 0 0 0 0 0 6 6 6 6 ...
##  $ Replicate      : int  1 2 3 4 5 6 1 2 3 4 ...
##  $ Water_Imbibed  : chr  "na" "na" "na" "na" ...
##  $ shannon        : num  6.62 6.61 6.66 6.66 6.61 ...
##  $ invsimpson     : num  211 207 213 205 200 ...
##  $ simpson        : num  0.995 0.995 0.995 0.995 0.995 ...
##  $ richness       : int  3319 3079 3935 3922 3196 3481 3250 3170 3657 3177 ...
##  $ Pielou_Evenness: num  0.817 0.823 0.805 0.805 0.819 ...
```

```
# Display the first few rows
head(alpha_even)
```

```
##     Code Crop Time_Point Replicate Water_Imbibed  shannon invsimpson   simpson
## 1 S01_13 Soil          0         1            na 6.624921   210.7279 0.9952545
## 2 S02_16 Soil          0         2            na 6.612413   206.8666 0.9951660
## 3 S03_19 Soil          0         3            na 6.660853   213.0184 0.9953056
## 4 S04_22 Soil          0         4            na 6.660671   204.6908 0.9951146
## 5 S05_25 Soil          0         5            na 6.610965   200.2552 0.9950064
## 6 S06_28 Soil          0         6            na 6.650812   199.3211 0.9949830
##   richness Pielou_Evenness
## 1     3319       0.8171431
## 2     3079       0.8232216
## 3     3935       0.8046776
## 4     3922       0.8049774
## 5     3196       0.8192376
## 6     3481       0.8155427
```

Q4: Calculating the mean and standard error evenness grouped by crop over time.

```
# Summarize the data: Mean and standard error of Pielou's evenness index grouped by Crop and Time_Point
alpha_average <- alpha_even %>%
  group_by(Crop, Time_Point) %>%  # Group by Crop and Time_Point
  summarise(
    Mean_Evenness = mean(Pielou_Evenness, na.rm = TRUE),  # Calculate mean evenness
    n = n(),  # Count the number of observations per group
    sd_evenness = sd(Pielou_Evenness, na.rm = TRUE)  # Standard deviation
  ) %>%
  mutate(std_err_evenness = sd_evenness / sqrt(n))  # Calculate standard error
```

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

```
# Check the structure of the resulting dataframe
str(alpha_average)
```

```
## gropd_df [12 x 6] (S3: grouped_df/tbl_df/tbl/data.frame)
##  $ Crop            : chr [1:12] "Cotton" "Cotton" "Cotton" "Cotton" ...
##  $ Time_Point      : int [1:12] 0 6 12 18 0 6 12 18 0 6 ...
##  $ Mean_Evenness   : num [1:12] 0.82 0.805 0.767 0.755 0.814 ...
##  $ n               : int [1:12] 6 6 6 5 6 6 6 5 6 6 ...
##  $ sd_evenness     : num [1:12] 0.00556 0.0092 0.01567 0.01689 0.00765 ...
##  $ std_err_evenness: num [1:12] 0.00227 0.00376 0.0064 0.00755 0.00312 ...
##  - attr(*, "groups")= tibble [3 x 2] (S3: tbl_df/tbl/data.frame)
##   ..$ Crop : chr [1:3] "Cotton" "Soil" "Soybean"
##   ..$ .rows: list<int> [1:3]
##   .. ..$ : int [1:4] 1 2 3 4
##   .. ..$ : int [1:4] 5 6 7 8
##   .. ..$ : int [1:4] 9 10 11 12
##   .. ..@ ptype: int(0)
##   ..- attr(*, ".drop")= logi TRUE
```

```
# Display the first few rows
head(alpha_average)
```

```
## # A tibble: 6 x 6
## # Groups:   Crop [2]
##   Crop   Time_Point Mean_Evenness     n sd_evenness std_err_evenness
##   <chr>       <int>         <dbl> <int>       <dbl>            <dbl>
## 1 Cotton          0         0.820     6     0.00556          0.00227
## 2 Cotton          6         0.805     6     0.00920          0.00376
## 3 Cotton         12         0.767     6     0.0157           0.00640
## 4 Cotton         18         0.755     5     0.0169           0.00755
## 5 Soil            0         0.814     6     0.00765          0.00312
## 6 Soil            6         0.810     6     0.00587          0.00240
```

Q5: Calculating the differences in mean evenness between Soybean, Cotton, and Soil.

```
# Transform the data from long to wide format
alpha_average2 <- alpha_average %>%
  select(Time_Point, Crop, Mean_Evenness) %>%  # Select relevant columns
  pivot_wider(names_from = Crop, values_from = Mean_Evenness) %>%  # Reshape to wide format
  mutate(
    diff.cotton.even = Soil - Cotton,   # Difference between Cotton and Soil
    diff.soybean.even = Soil - Soybean   # Difference between Soybean and Soil
  )

# Check the structure of the resulting dataframe
str(alpha_average2)
```

```
## tibble [4 x 6] (S3: tbl_df/tbl/data.frame)
##  $ Time_Point       : int [1:4] 0 6 12 18
##  $ Cotton           : num [1:4] 0.82 0.805 0.767 0.755
##  $ Soil             : num [1:4] 0.814 0.81 0.798 0.8
##  $ Soybean          : num [1:4] 0.822 0.764 0.687 0.716
```

```
##  $ diff.cotton.even : num [1:4] -0.00602 0.00507 0.03129 0.0449
##  $ diff.soybean.even: num [1:4] -0.0074 0.0459 0.1119 0.0833
```

```r
# Display the first few rows
head(alpha_average2)
```

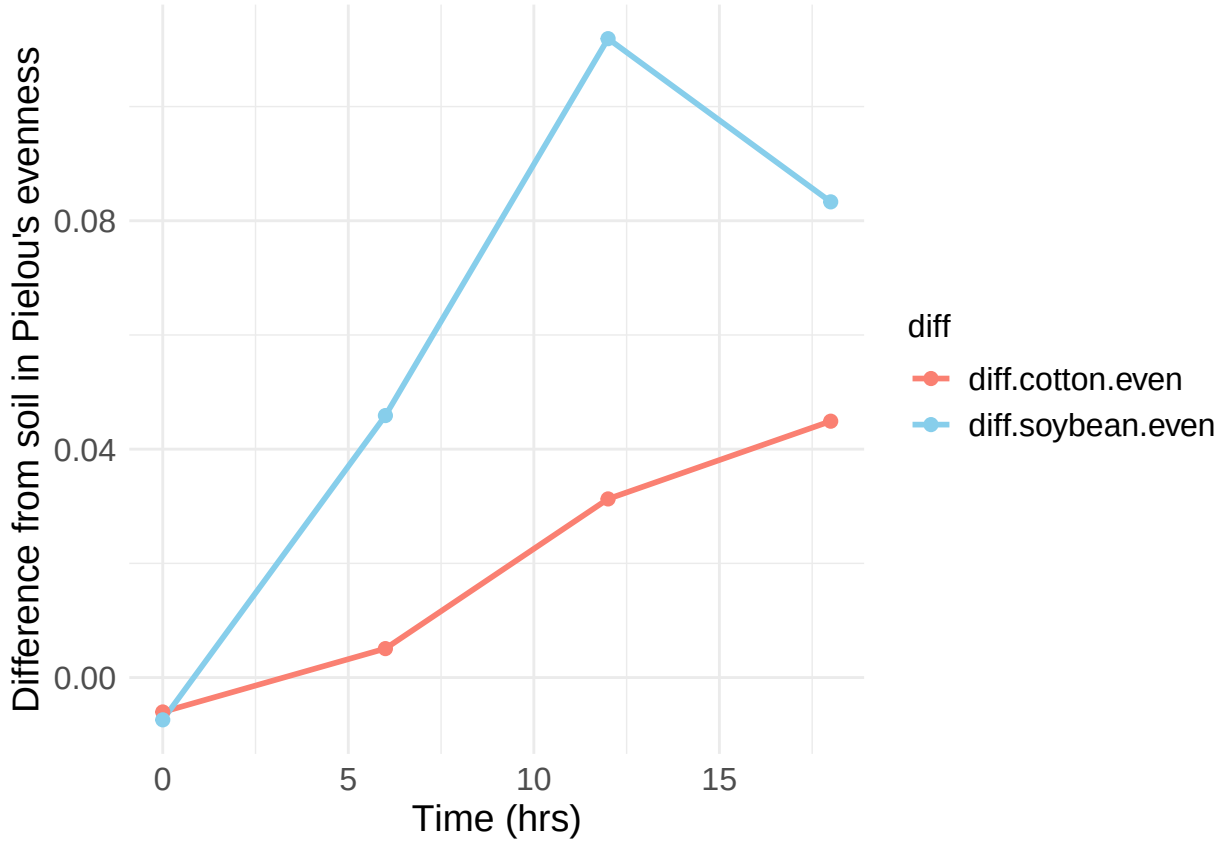
```
## # A tibble: 4 x 6
##   Time_Point Cotton  Soil Soybean diff.cotton.even diff.soybean.even
##        <int>  <dbl> <dbl>   <dbl>            <dbl>             <dbl>
## 1          0  0.820 0.814   0.822         -0.00602          -0.00740
## 2          6  0.805 0.810   0.764          0.00507           0.0459
## 3         12  0.767 0.798   0.687          0.0313            0.112
## 4         18  0.755 0.800   0.716          0.0449            0.0833
```

Q6: creating the plot using ggplot2 after reshaping the data with pivot_longer()

```r
# Reshape the data from wide to long format
alpha_plot_data <- alpha_average2 %>%
  select(Time_Point, diff.cotton.even, diff.soybean.even) %>%  # Select relevant columns
  pivot_longer(cols = c(diff.cotton.even, diff.soybean.even),
               names_to = "diff",
               values_to = "values")  # Reshape data

# Create the plot
ggplot(alpha_plot_data, aes(x = Time_Point, y = values, color = diff, group = diff)) +
  geom_line(size = 1) +  # Line plot
  geom_point(size = 2) +  # Add points
  theme_minimal() +  # Minimal theme
  labs(
    x = "Time (hrs)",  # X-axis label
    y = "Difference from soil in Pielou's evenness",  # Y-axis label
    color = "diff"  # Legend title
  ) +
  scale_color_manual(values = c("diff.cotton.even" = "salmon", "diff.soybean.even" = "skyblue")) +  # A
  theme(
    text = element_text(size = 12),  # General text size
    axis.title.x = element_text(size = 14),  # X-axis title size
    axis.title.y = element_text(size = 14),  # Y-axis title size
    axis.text = element_text(size = 12),  # Tick labels size
    legend.text = element_text(size = 12),  # Legend text size
    legend.title = element_text(size = 12)  # Legend title size
  )
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



Q7: This is my submission for Coding Challenge 5.

Click here to view my GitHub repository