Reshaping Your Data with the tidyr | Lab assignment 8

Noah Gallego

2024-09-25

Loading the installed package to your R session

This is the process of loading and attaching packages or libraries to your R session.

If you're installing a package for the first time install.packages("dplyr")

```
## Loading required package: tidyr

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Question 1

The "produceWide.txt" dataset contains the information about yields of produce in all seasons. Reshape the dataset so the new data frame will have three columns: ID, season, and yield, where the variable season has four levels (spring, summer, fall and winter).

Read the data set as produceWide

```
df_wide = read.table("produceWide.txt", header = TRUE)
head(df_wide)

##    ID Spring Summer Fall Winter
## 1 101     56     78  105     54
## 2 102     61     85   12     51
## 3 103     83     10   62     15
## 4 104     57     95   46     28

df_long = df_wide %>%
  pivot_longer(cols = c(Spring, Summer, Fall, Winter),
               names_to = "season",
               values_to = "yields")
head(df_long)

## # A tibble: 6 × 3
##      ID season yields
##   <int> <chr>   <int>
## 1   101 Spring     56
## 2   101 Summer     78
## 3   101 Fall      105
## 4   101 Winter     54
## 5   102 Spring     61
## 6   102 Summer     85
```

Question 2

The "produceLong.txt" dataset contains the information about yields of produce in all seasons. Read it as produceLong and Reshape the produceLong data frame to wide format.

```
produce_df_long = read.table("produceLong.txt", header = TRUE)
head(produce_df_long)

##    ID season yield
## 1 101 Spring    56
## 2 102 Spring    61
## 3 103 Spring    83
## 4 104 Spring    57
## 5 101 Summer    78
## 6 102 Summer    85

produce_df_wide = produce_df_long %>%
  pivot_wider(names_from = season,
              values_from = yield)
head(produce_df_wide)

## # A tibble: 4 × 5
##      ID Spring Summer  Fall Winter
##   <int>  <int>  <int> <int>  <int>
## 1   101     56     78   105     54
## 2   102     61     85    12     51
## 3   103     83     10    62     15
## 4   104     57     95    46     28
```

Question 3

Read the S&p 500 index data from "^GSPC.csv". Separate the "Date" column into three columns as "month","day","year"

```
sp_df = read.csv("^GSPC.csv")
sp_df = sp_df %>%
  separate(col = Date,
           into = c("month", "day", "year"),
           sep = "/")
head(sp_df)

##   month day year  Open  High   Low Close AdjClose Volume
## 1    12  30 1927 17.66 17.66 17.66 17.66    17.66      0
## 2     1   3 1928 17.76 17.76 17.76 17.76    17.76      0
## 3     1   4 1928 17.72 17.72 17.72 17.72    17.72      0
## 4     1   5 1928 17.55 17.55 17.55 17.55    17.55      0
## 5     1   6 1928 17.66 17.66 17.66 17.66    17.66      0
## 6     1   9 1928 17.50 17.50 17.50 17.50    17.50      0
```

Question 4

Read the “Diamonds.txt” file as Diamonds_data and create a new column price_weight by uniting the “WEIGHT” and “PRICE” columns by “/” .

```
diamonds_data = read.table("Diamonds.txt", header = TRUE)
diamonds_data  = diamonds_data %>%
  unite(col = price_weight, WEIGHT, PRICE, sep = "/")

head(diamonds_data)

##   IDNO price_weight COLOR CLARITY RATER
## 1    1      0.3/1302     D     VS2   GIA
## 2    2      0.3/1510     E     VS1   GIA
## 3    3      0.3/1510     G    VVS1   GIA
## 4    4      0.3/1260     G     VS1   GIA
## 5    5     0.31/1641     D     VS1   GIA
## 6    6     0.31/1555     E     VS1   GIA
```

Question 5

Read the “student_performance_missing.xlsx” file as student_data. You need “readxl” package for this. Install it first

```
library(readxl)
student_data = read_excel("student_performance_missing.xlsx")
head(student_data)

## # A tibble: 6 × 5
##   Name    Attendance Exam_Score Study_Time Major
##   <chr>   <chr>      <chr>           <dbl> <chr>
## 1 Charlie Good       80               6.14 Psychology
## 2 Charlie Very Good  98               8    Math
## 3 Jack    Good       NA               6.11 Biology
## 4 Bob     Very bad   47               3.32 Business
## 5 Frank    Very bad   59                4.31 Psychology
## 6 Emily    Bad        61                2.95 Math
```

Now replace missing values in Exam_score with mean of the Exam_score values

```
mean_exam_score = mean(student_data$Exam_Score, na.rm = TRUE)

## Warning in mean.default(student_data$Exam_Score, na.rm = TRUE): argument i
s not
## numeric or logical: returning NA

student_data <- student_data %>%
  mutate(Exam_Score = ifelse(is.na(Exam_Score), mean_exam_score, Exam_Score))

head(student_data)

## # A tibble: 6 × 5
##   Name    Attendance Exam_Score Study_Time Major
##   <chr>   <chr>      <chr>           <dbl> <chr>
## 1 Charlie Good       80               6.14 Psychology
## 2 Charlie Very Good  98               8    Math
## 3 Jack    Good       NA               6.11 Biology
## 4 Bob     Very bad   47               3.32 Business
## 5 Frank   Very bad   59               4.31 Psychology
## 6 Emily   Bad        61               2.95 Math
```