p8105_hw1_yl5828

Leah Li yl5828

2025-09-19

HW1

Problem 1 We first install the required "moderndive" package in Console.

```
# Load the dataset
library(moderndive)
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   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- 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
```

```
data("early_january_weather")
```

```
help("early_january_weather")
```

The dataset `early_january_weather` from the `moderndive` package has character variable **origin**, integer variables **year, month, day, hour**, double class variables **temp, dewp, humid, wind_dir, wind_speed, wind_gust, precip, precip, visib**, and date and time variable **time_hour**.

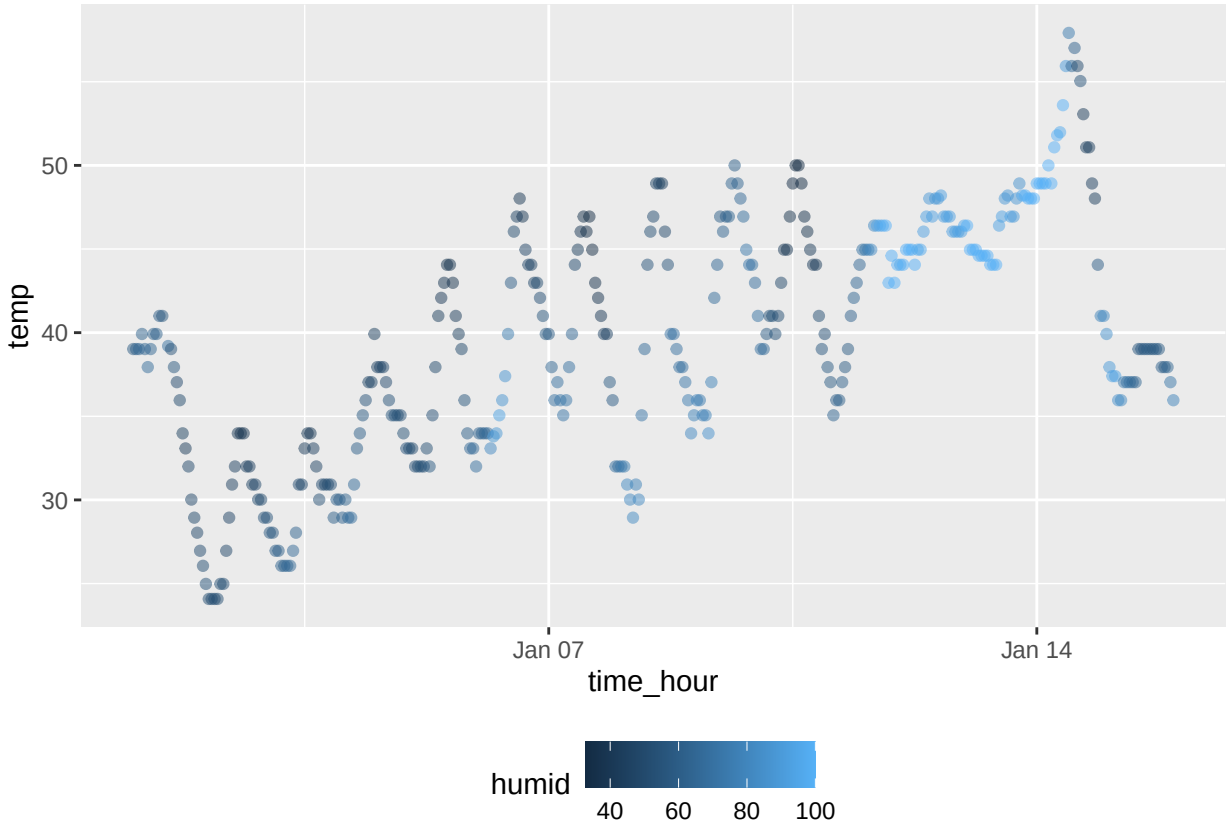
The dataset has **358** rows and **15** columns.

The mean temperature is **39.5821229** F.

Now, we would like to generate a scatterplot of temperature versus time.

```
weather_plot <- ggplot(
  early_january_weather,
  aes(x = time_hour, y = temp, color = humid) ) +
  geom_point(alpha = .5) +
  theme(legend.position = "bottom")

# Display the plot
print(weather_plot)
```



Several patterns are apparent in this plot:

1. **Diurnal Cycle:** There is a clear and repeating cyclical pattern in the temperature. The temperature consistently rises during the daytime and falls during the nighttime. This creates the wave-like shape visible across the x-axis, representing the daily temperature cycle.
2. **Time and Humidity Relationship:** The color of the points reveals a relationship between temperature and time. The humidity increases as the time goes by.
3. **Time and Temperature Relationship:** While the daily cycle is consistent, the overall peaks and troughs vary from day to day. For instance, the temperatures around January 4th are noticeably lower than those around January 11th, indicating a cold spell followed by a warmer period. Overall, the temperature increases as the time goes by.

Now, we save the scatterplot to our project directory.

```
ggsave("temp_vs_time_plot.png", plot = weather_plot, width = 10, height = 6, dpi = 300)
```

Problem 2 Now, we would like to construct a dataframe with some required vectors.

```
# Set a seed for reproducibility
set.seed(123)

# Create the required vectors
normal_sample <- rnorm(10)
logical_vector <- normal_sample > 0
char_vector <- c("apple", "banana", "cherry", "date", "elderberry",
                 "fig", "grape", "honeydew", "kiwi", "lemon")
factor_vector <- factor(c("low", "medium", "high", "low", "medium",
```

```
                          "high", "low", "medium", "high", "medium"))

set.seed(123)
# Combine them into a data frame
dataframe <- data.frame(
  NormalSample = normal_sample,
  GreaterThan0 = logical_vector,
  Character = char_vector,
  Level = factor_vector
)

set.seed(123)
# Print the data frame
print(dataframe)
```

```
##    NormalSample GreaterThan0  Character  Level
## 1   -0.56047565        FALSE      apple    low
## 2   -0.23017749        FALSE     banana medium
## 3    1.55870831         TRUE     cherry   high
## 4    0.07050839         TRUE       date    low
## 5    0.12928774         TRUE elderberry medium
## 6    1.71506499         TRUE        fig   high
## 7    0.46091621         TRUE      grape    low
## 8   -1.26506123        FALSE   honeydew medium
## 9   -0.68685285        FALSE       kiwi   high
## 10  -0.44566197        FALSE      lemon medium
```

Now, we take the mean of each variable in the dataframe.

```
# This works because the column is numeric
mean_normal <- mean(pull(dataframe, NormalSample))
print(paste("Mean of NormalSample is ", mean_normal))
```

```
## [1] "Mean of NormalSample is  0.0746256440971619"
```

```
# This works due to coercion (TRUE -> 1, FALSE -> 0)
mean_logical <- mean(pull(dataframe, GreaterThan0))
print(paste("Mean of GreaterThan0 is ", mean_logical))
```

```
## [1] "Mean of GreaterThan0 is  0.5"
```

```
# The following line will generate a warning, which is expected.
mean_char <- mean(pull(dataframe, Character))
```

```
## Warning in mean.default(pull(dataframe, Character)): argument is not numeric or
## logical: returning NA
```

```
print(mean_char) # This shows NA
```

```
## [1] NA
```

We could notice that for character vector the mean returns **NA**.

```
# The following line generates a warning.
mean_factor <- mean(pull(dataframe, Level))
```

```
## Warning in mean.default(pull(dataframe, Level)): argument is not numeric or
## logical: returning NA
```

```r
print(mean_factor) # This shows NA
```

```
## [1] NA
```

We could notice that for factor vector the mean also returns **NA**.

That is, we can conclude that take means can work only for **normal sample** and **logical** variables, but not for **character** and **factor** variables.

```r
#echo = T, results = 'hide'
as.numeric(pull(dataframe, GreaterThan0))
```

```
##  [1] 0 0 1 1 1 1 1 0 0 0
```

```r
as.numeric(pull(dataframe, Character))
```

```
## Warning: NAs introduced by coercion
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

```r
as.numeric(pull(dataframe, Level))
```

```
##  [1] 2 3 1 2 3 1 2 3 1 3
```

After we apply as.numeric function, we could observe that for logic variable, the function works well since TRUE values will become 1 and FALSE values will become 0.

For factor vector, function works since each level is being represent as a number from 1-3.

For character vector, a warning message is showed since the text cannot be represent as a number, so the result will only generate NAs.