p8105_hw1_rh3195

Ruijie He

2023-09-23

Problem 1

```
library ("moderndive")
data("early_january_weather")
```

variables in dataset

```
variables <- colnames(early_january_weather)
variables
```

```
##  [1] "origin"     "year"       "month"      "day"        "hour"
##  [6] "temp"       "dewp"       "humid"      "wind_dir"   "wind_speed"
## [11] "wind_gust"  "precip"     "pressure"   "visib"      "time_hour"
```

Size of dataset

```
num_rows <- nrow(early_january_weather)
num_cols <- ncol(early_january_weather)
num_rows
```

```
## [1] 358
```

```
num_cols
```

```
## [1] 15
```

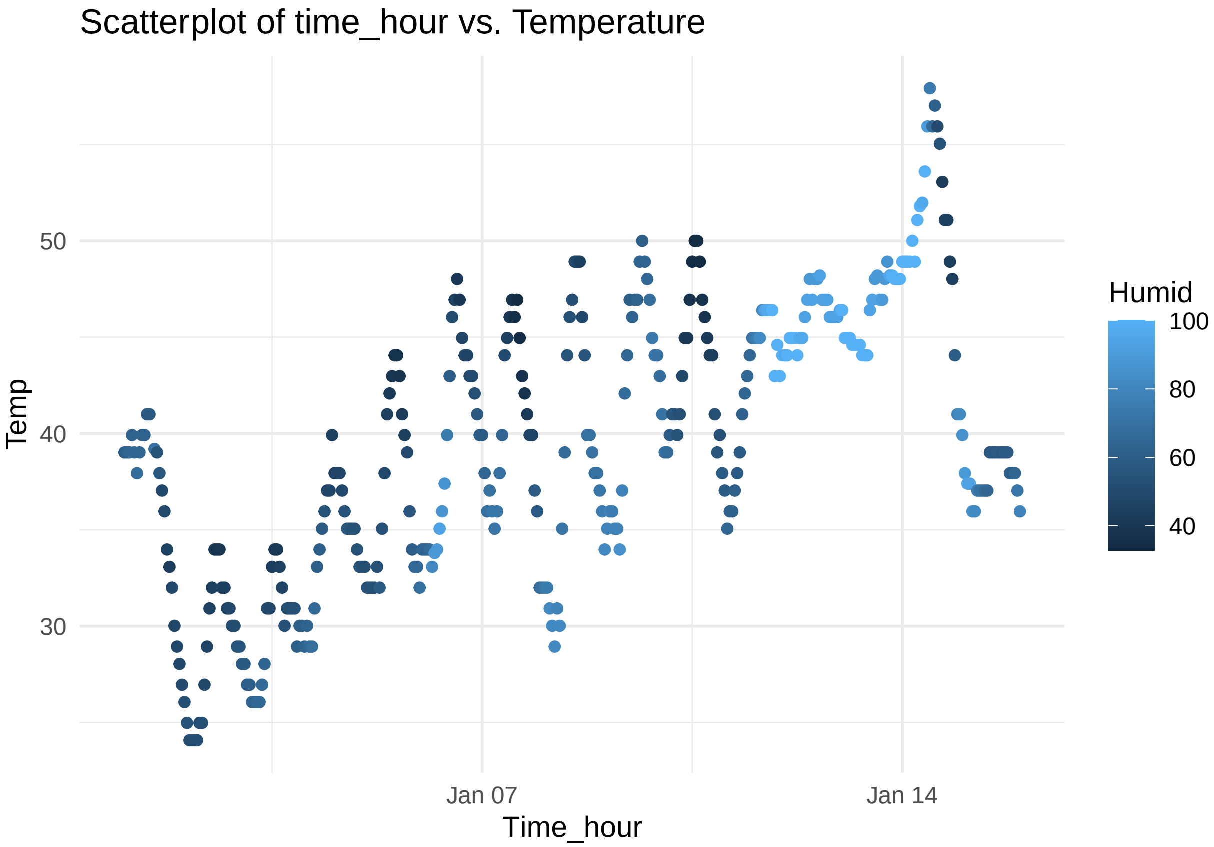
```
mean_temp <- mean (early_january_weather$temp)
mean_temp
```

```
## [1] 39.58212
```

- There are 15 variables in this dataset. Some important variables including year, month, day, and time_hour that tells the time. The wind direction, wind speed, and wind-gust that is related to the wind. Also the "temp" tells temperature and "humid" tells the humidity. It has 358 rows and 15 columns. The mean temperature is 39.58 degree. # Making scatterplot of temp (y) vs. time_hour (x)

```
library (ggplot2)

ggplot(early_january_weather, aes(x = time_hour, y = temp, color = humid)) +
  geom_point() +
  labs(x = "Time_hour", y = "Temp", color = "Humid") +
  ggtitle("Scatterplot of time_hour vs. Temperature") +
  theme_minimal()
```



##Describe pattern of scatterplot##

- The scatterplot shows that the two variables time_hour and tempeartue are having a positive association. Because as the temperature increases as the time_hour do. And the data points on this plot are assembled closely nearly to a linear line, which illustrating a linear relationship.

```
ggsave("scatterplot_of_time_vs_temperature.pdf", width = 6, height =4)
```

Problem 2

```
library (tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
## -- 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
```

```
my_df =
  tibble(
    vec_numeric = rnorm (10),
   vec_logical = vec_numeric > 0,
   vec_char = c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J"),
   vec_factor = factor(
     sample(c("L0", "L1", "L2"), 10, rep = TRUE)
   )
  )
print (my_df)
```

```
## # A tibble: 10 x 4
##    vec_numeric vec_logical vec_char vec_factor
##          <dbl> <lgl>       <chr>    <fct>
##  1       0.921 TRUE        A        L2
##  2      -0.744 FALSE       B        L1
##  3       1.36  TRUE        C        L2
##  4       0.776 TRUE        D        L0
##  5       1.62  TRUE        E        L2
##  6       0.449 TRUE        F        L2
##  7      -0.861 FALSE       G        L0
##  8       0.298 TRUE        H        L1
##  9       0.776 TRUE        I        L1
## 10       0.133 TRUE        J        L0
```

```
mean_numeric = mean(pull(my_df, vec_numeric))
mean_logical = mean(pull(my_df, vec_logical))
mean_char = mean(pull(my_df, vec_char))
```

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

```
mean_factor = mean(pull(my_df, vec_factor))
```

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

```r
print (mean_numeric)
```

```
## [1] 0.4732425
```

```r
print (mean_logical)
```

```
## [1] 0.8
```

```r
print (mean_char)
```

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

```r
print (mean_factor)
```

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

- The variable of vec_numeric and vec_logical have mean. Charactor variable and factor variable do not generates mean because they are not numeric number or logical.

```r
as.numeric(pull(my_df, vec_logical))
```

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

```r
as.numeric(pull(my_df, vec_char))
```

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

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

```r
as.numeric(pull(my_df, vec_factor))
```

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

- The vec_logical is converted to binary datapoints where 0 is false and 1 is true. The vec_factor also get translated to 3 categories based on given lavels. The only variable that is unable to convert to numeric is the vec_char. This helps me to understand what happens when trying to take the mean. It tells that mean should be numeric numbers.