615-HW4

Gary Wang

2024-09-25

```r
library(data.table)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

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

```r
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

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

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

a

```r
# function to load buoy data for a given year
load_buoy <- function(year) {
  base_url <- "https://www.ndbc.noaa.gov/view_text_file.php?filename=44013h"
  suffix <- ".txt.gz&dir=data/historical/stdmet/"
  file_url <- paste0(base_url, year, suffix)

  # skip header lines based on year
  skip <- ifelse(year < 2007, 1, 2)

  # read column names
  headers <- scan(file_url, what = 'character', nlines = 1)

  # load the dataset with fill to handle missing columns
  buoy <- fread(file_url, header = FALSE, skip = skip, fill = TRUE)

  # adjust columns to match header count
  if (ncol(buoy) > length(headers)) {
    buoy <- buoy[, 1:length(headers), with = FALSE]
  } else if (ncol(buoy) < length(headers)) {
    for (i in 1:(length(headers) - ncol(buoy))) {
      buoy[, paste0("V", ncol(buoy) + i) := NA]
    }
  }

  # assign column names
  setnames(buoy, headers)

  # combine date and time into a single column
  buoy$datetime <- make_datetime(
    year = as.integer(buoy$YYYY),
    month = as.integer(buoy$MM),
    day = as.integer(buoy$DD),
    hour = as.integer(buoy$hh),
    min = as.integer(buoy$mm)
  )

  return(buoy)
}

# define years range
years <- 1985:2023

# load data for each year
buoy_list <- lapply(years, load_buoy)
```

```
## Warning in fread(file_url, header = FALSE, skip = skip, fill = TRUE): Stopped
## early on line 5114. Expected 16 fields but found 17. Consider fill=TRUE and
## comment.char=. First discarded non-empty line: <<2000 08 01 00 78 4.3 5.1 0.58
## 8.33 5.36 999 1022.9 17.3 17.5 15.0 99.0 99.00>>
```

```r
# combine all yearly data
combined_buoy <- rbindlist(buoy_list, fill = TRUE)
```

```r
fwrite(combined_buoy, "buoy_data.csv")
```

b

```r
buoy_data <- fread("buoy_data.csv")

# convert placeholder values to NA
missing_values <- c(999, 99.9, 9999)

# specify columns that may contain these placeholder values
columns_to_check <- c("WDIR", "WSPD", "GST", "WVHT", "APD", "MWD",
                      "PRES", "ATMP", "WTMP", "DEWP", "VIS")

# iterate over the columns and replace only the specified placeholders with NA
for (col in columns_to_check) {
  buoy_data[[col]] <- ifelse(buoy_data[[col]] %in% missing_values, NA, buoy_data[[col]])
}

# analyze the pattern of NAs
na_summary <- sapply(buoy_data, function(x) sum(is.na(x)))
print(na_summary)
```

```
##       YY       MM       DD       hh       WD     WSPD      GST     WVHT 
##   346151        0        0        0   280220        0        0        0 
##      DPD      APD      MWD      BAR     ATMP     WTMP     DEWP      VIS 
##        0        0   325297   280220   102761    13186   253613        0 
## datetime     YYYY     TIDE       mm      #YY     WDIR     PRES 
##   444870   396370   129610   164650   182081   210347   187776 
```

```r
# display a sample of the data to ensure correct replacements
head(buoy_data)
```

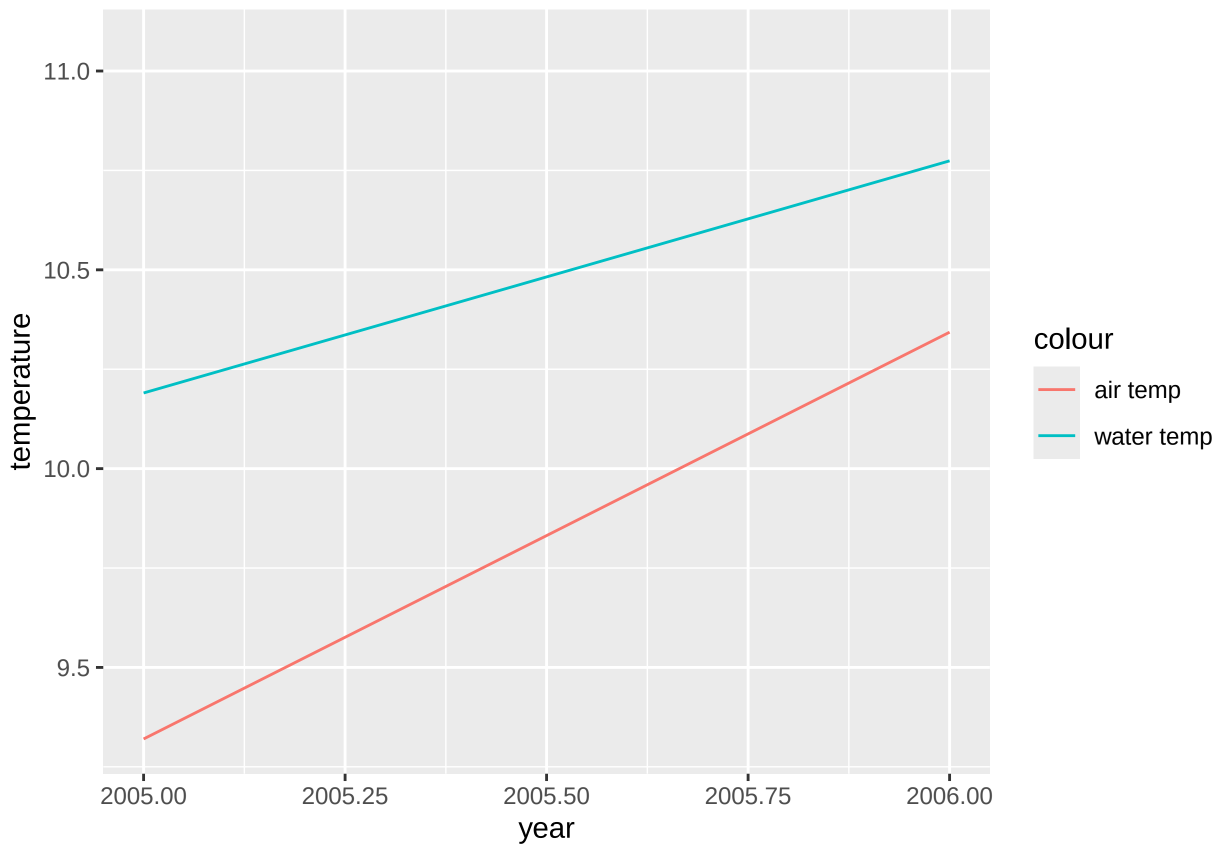
```
##    YY MM DD hh  WD WSPD GST WVHT DPD APD MWD    BAR ATMP WTMP DEWP VIS datetime
## 1: 85  1  1  0  60    4   5   99  99  99  NA 1030.3  4.7  6.7   NA  99     <NA>
## 2: 85  1  1  1  80    4   5   99  99  99  NA 1030.0  5.1  6.7   NA  99     <NA>
## 3: 85  1  1  2 100    4   5   99  99  99  NA 1030.1  5.6  6.6   NA  99     <NA>
## 4: 85  1  1  3 100    4   5   99  99  99  NA 1029.4  5.8  6.7   NA  99     <NA>
## 5: 85  1  1  4 110    4   5   99  99  99  NA 1028.6  5.8  6.7   NA  99     <NA>
## 6: 85  1  1  5  90    4   5   99  99  99  NA 1027.8  5.3  6.7   NA  99     <NA>
##    YYYY TIDE mm #YY WDIR PRES
## 1:   NA   NA NA  NA   NA   NA
## 2:   NA   NA NA  NA   NA   NA
## 3:   NA   NA NA  NA   NA   NA
## 4:   NA   NA NA  NA   NA   NA
## 5:   NA   NA NA  NA   NA   NA
## 6:   NA   NA NA  NA   NA   NA
```

```r
# Converting missing/null data to NA is not always a good idea. Because the placeholder values, such as
# The NA values appear to be distributed in a structured way as they clustered around certain variables
```

c

```
buoy_data$year <- year(buoy_data$datetime)

# Calculate annual averages for ATMP, WTMP, and PRES
annual_avg <- buoy_data[, .(avg_ATMP = mean(ATMP, na.rm = TRUE),
                            avg_WTMP = mean(WTMP, na.rm = TRUE)), by = year]

# Plotting air and water temperature trends
ggplot(annual_avg, aes(x = year)) +
  geom_line(aes(y = avg_ATMP, color = "air temp")) +
  geom_line(aes(y = avg_WTMP, color = "water temp")) +
  labs(x = "year", y = "temperature")
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_line()‘).
## Removed 1 row containing missing values or values outside the scale range
## (‘geom_line()‘).
```



```
# linear regression to find trends
model_ATMP <- lm(avg_ATMP ~ year, data = annual_avg)
model_WTMP <- lm(avg_WTMP ~ year, data = annual_avg)

# summary of regression models
summary_ATMP <- summary(model_ATMP)
summary_WTMP <- summary(model_WTMP)
```

```r
print(summary_ATMP)
```

```
##
## Call:
## lm(formula = avg_ATMP ~ year, data = annual_avg)
##
## Residuals:
## ALL 2 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2041.884        NaN     NaN      NaN
## year            1.023        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 1 and 0 DF,  p-value: NA
```

```r
print(summary_WTMP)
```

```
##
## Call:
## lm(formula = avg_WTMP ~ year, data = annual_avg)
##
## Residuals:
## ALL 2 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1160.2926        NaN     NaN      NaN
## year            0.5838        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 1 and 0 DF,  p-value: NA
```

d

```r
rainfall_data <- fread("Rainfall.csv")

# inspect the structure of the data
str(rainfall_data)
```

```
## Classes 'data.table' and 'data.frame':   31714 obs. of  6 variables:
##  $ STATION         : chr  "COOP:190770" "COOP:190770" "COOP:190770" "COOP:190770" ...
##  $ STATION_NAME    : chr  "BOSTON LOGAN INTERNATIONAL AIRPORT MA US" "BOSTON LOGAN INTERNATIONAL AIRP
##  $ DATE            : chr  "19850101 01:00" "19850101 09:00" "19850101 10:00" "19850101 11:00" ...
##  $ HPCP            : num  0 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ...
##  $ Measurement Flag: chr  "g" "" "" "" ...
##  $ Quality Flag    : logi  NA NA NA NA NA NA ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
summary(rainfall_data)
```

```
##    STATION          STATION_NAME           DATE                HPCP        
##  Length:31714       Length:31714       Length:31714       Min.   :0.00000  
##  Class :character   Class :character   Class :character   1st Qu.:0.00000  
##  Mode  :character   Mode  :character   Mode  :character   Median :0.01000  
##                                                           Mean   :0.03875  
##                                                           3rd Qu.:0.04000  
##                                                           Max.   :2.03000  
##  Measurement Flag   Quality Flag  
##  Length:31714       Mode:logical  
##  Class :character   NA's:31714    
##  Mode  :character                 
##                                   
##                                   
## 
```

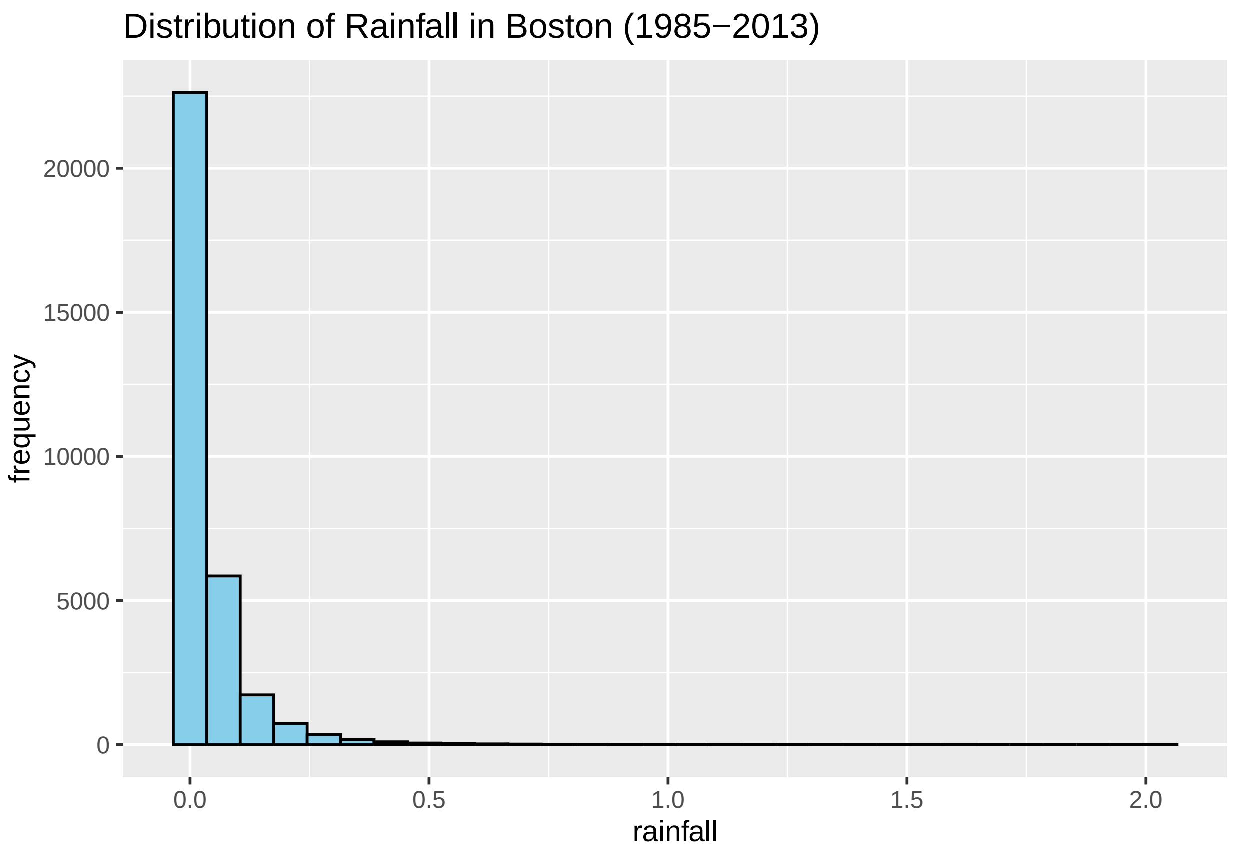
```r
# check for missing values
colSums(is.na(rainfall_data))
```

```
##          STATION     STATION_NAME             DATE             HPCP 
##                0                0                0                0 
## Measurement Flag     Quality Flag 
##                0            31714
```

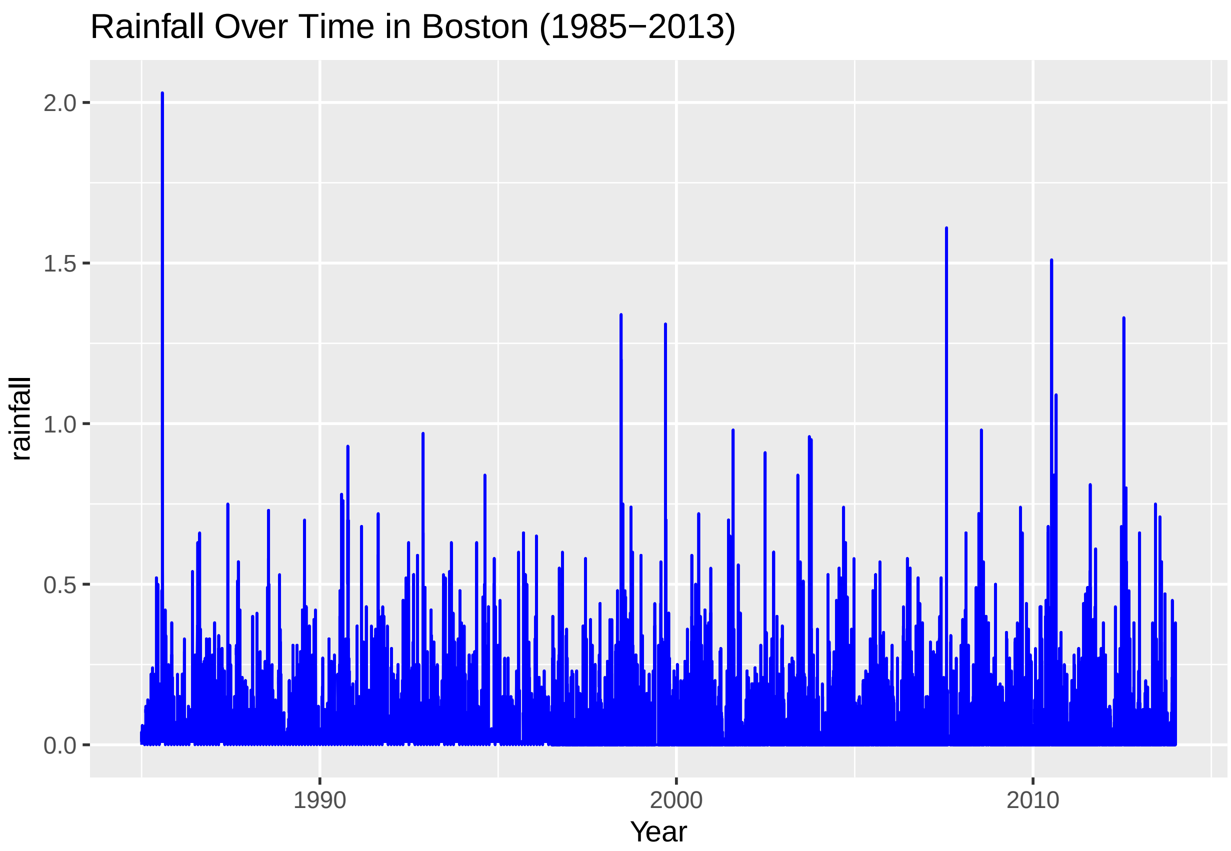
```r
# convert date to date-time format
rainfall_data$Date <- as.POSIXct(rainfall_data$DATE, format = "%Y%m%d %H:%M", tz = "UTC")

# calculate summary statistics for rainfall
rainfall_stats <- rainfall_data %>%
  summarise(
    mean_rainfall = mean(HPCP, na.rm = TRUE),
    median_rainfall = median(HPCP, na.rm = TRUE),
    max_rainfall = max(HPCP, na.rm = TRUE),
    min_rainfall = min(HPCP, na.rm = TRUE)
  )

print(rainfall_stats)
```

```
##   mean_rainfall median_rainfall max_rainfall min_rainfall
## 1     0.0387485            0.01         2.03            0
```

```r
# plot rainfall distribution
ggplot(rainfall_data, aes(x = HPCP)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Rainfall in Boston (1985-2013)",
       x = "rainfall", y = "frequency")
```



```r
# plot time series of rainfall
ggplot(rainfall_data, aes(x = Date, y = HPCP)) +
  geom_line(color = "blue") +
  labs(title = "Rainfall Over Time in Boston (1985-2013)",
       x = "Year", y = "rainfall")
```

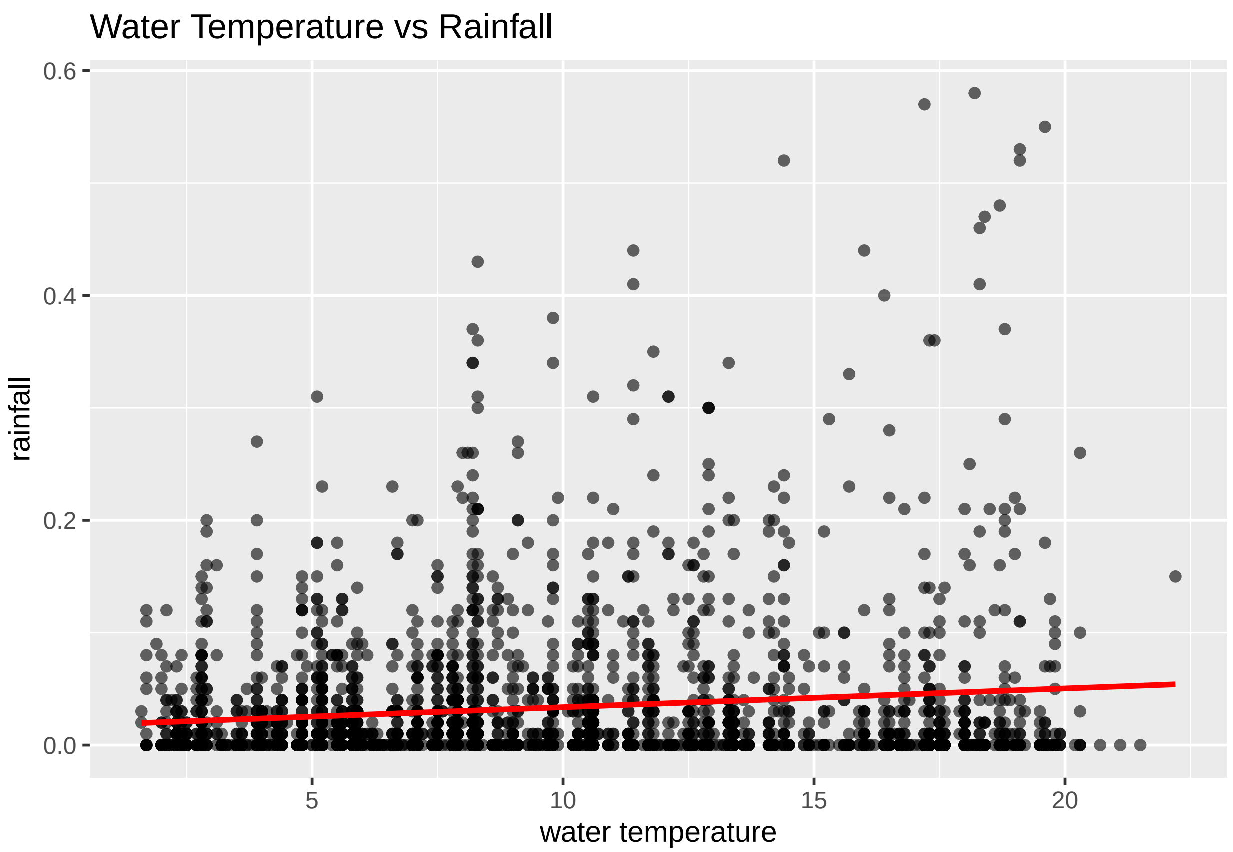


```r
# merge datasets by date
rainfall_buoy <- merge(rainfall_data, buoy_data, by.x = "Date", by.y = "datetime")

# explore relationships between rainfall (HPCP) and buoy readings
ggplot(rainfall_buoy, aes(x = WTMP, y = HPCP)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Water Temperature vs Rainfall",
         x = "water temperature", y = "rainfall")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 3 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



```
# Yes, this exercise shows just how tough forecasting really is. With messy data, sensor glitches, and
```