506 Problem Set 5

Xiaohan Liu

Table of contents

Problem 1 1
(a) 1
(b) 3
Problem 2 5
Problem 3 10
(a) 10
(b) 14

GitHub repository: https://github.com/EmiiilyLiu/STATS_506

```r
setwd("F:/Desktop/STATS 506/STATS_506")
```

Problem 1

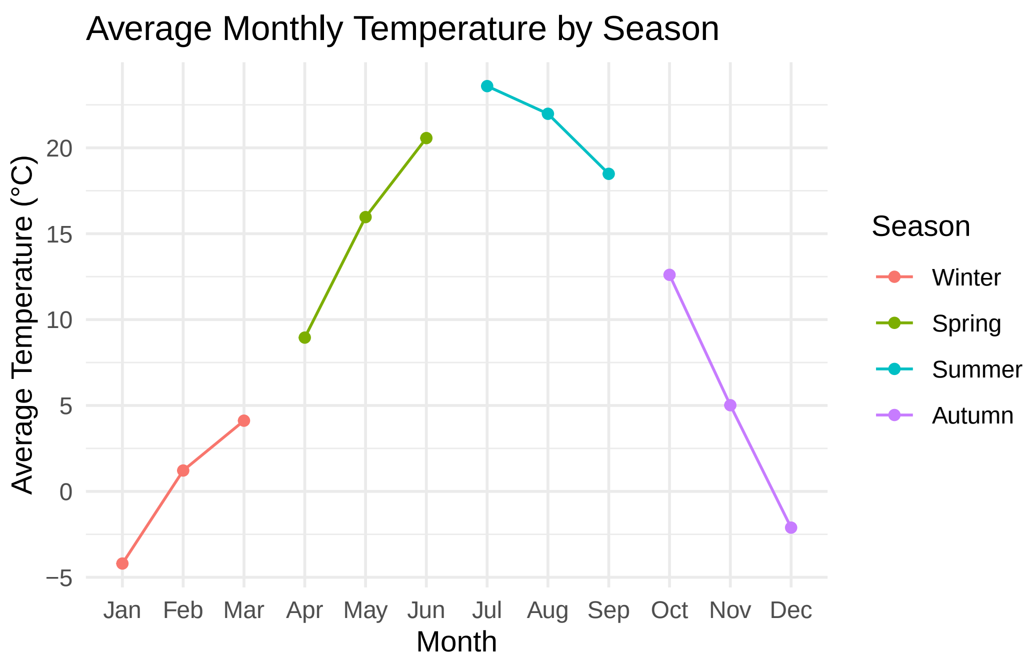
(a)

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

```r
library(tidyr)

nnmaps <- read.csv("chicago-nnmaps.csv")

## Convert temperature from Fahrenheit to Celsius
nnmaps$temp_celsius <- (nnmaps$temp - 32) * 5 / 9

## Define the order of the seasons
nnmaps$season <- factor(nnmaps$season,
                        levels = c("Winter", "Spring",
                                   "Summer", "Autumn"))

## Calculate monthly average temperature
monthly_avg_temp <- nnmaps %>%
  group_by(month, season) %>%
  summarise(mean_temp = mean(temp_celsius, na.rm = TRUE), .groups = 'drop') %>%
  ungroup() %>%
  arrange(match(month, month.abb))

## Plotting
## x-axis: month & y-axis: average monthly temperature in celsius
## A line connecting the points within each season
## Color the lines and points by season
ggplot(monthly_avg_temp, aes(x = month, y = mean_temp,
                             group = season, color = season)) +
  geom_point() +
  geom_line() +
  scale_x_discrete(limits = month.abb) +
  labs(title = "Average Monthly Temperature by Season",
       x = "Month",
       y = "Average Temperature (°C)",
       color = "Season") +
  theme_minimal()
```



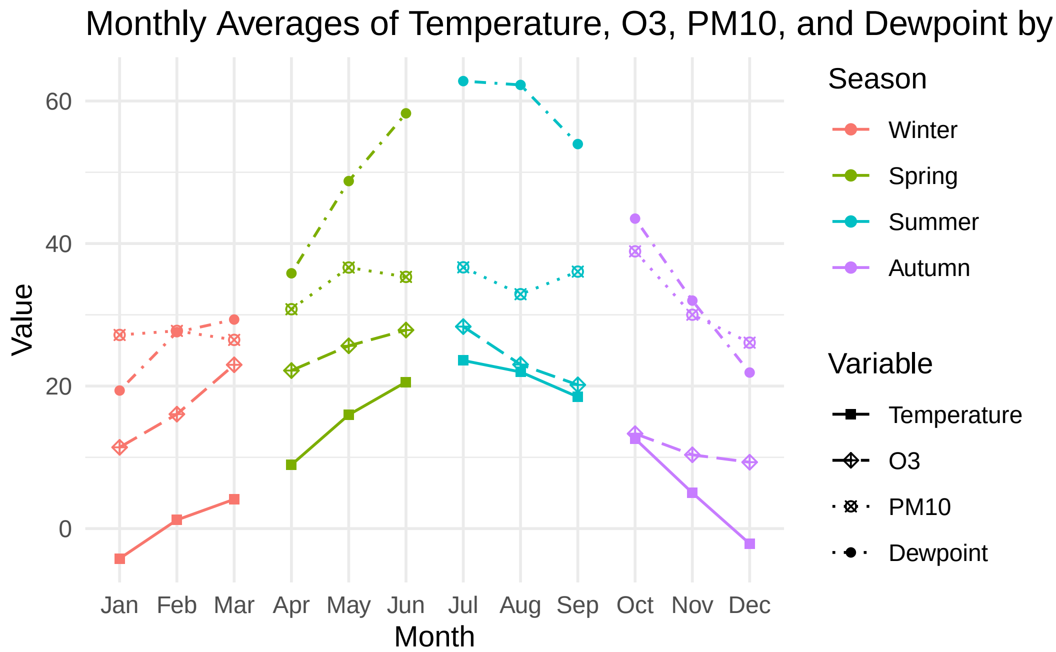
(b)

```r
## Calculate monthly average temperature, O3, PM10, and dewpoint
monthly_avg_data <- nnmaps %>%
  group_by(season, month) %>%
  summarize(mean_temp = mean(temp_celsius, na.rm = TRUE),
            mean_o3 = mean(o3, na.rm = TRUE),
            mean_pm10 = mean(pm10, na.rm = TRUE),
            mean_dewpoint = mean(dewpoint, na.rm = TRUE),
            .groups = 'drop') %>%
  ungroup() %>%
  arrange(match(month, month.abb))

## Define colors for the seasons
season_colors <- c("Winter" = "#F8766D", "Spring" = "#7CAE00",
                   "Summer" = "#00BFC4", "Autumn" = "#C77CFF")

## Define linetypes for the variables
variable_linetypes <- c("Temperature" = "solid", "O3" = "longdash",
                        "PM10" = "dotted", "Dewpoint" = "dotdash")
```

```r
## Define shapes for the variables
variable_shapes <- c("Temperature" = 15, "O3" = 9, "PM10" = 13, "Dewpoint" = 16)

## Create a new variable to map linetypes and shapes to variables
monthly_avg <- monthly_avg_data %>%
  pivot_longer(cols = starts_with("mean_"),
               names_to = "variable", values_to = "value") %>%
  mutate(variable = factor(variable, levels = c("mean_temp",
                                                 "mean_o3",
                                                 "mean_pm10",
                                                 "mean_dewpoint"),
                           labels = c("Temperature", "O3", "PM10", "Dewpoint")))

## Plot using the long format data
final_plot <- ggplot(monthly_avg, aes(x = month, y = value,
                                      group = interaction(season, variable))) +
  geom_point(aes(color = season, shape = variable)) +
  geom_line(aes(color = season, linetype = variable)) +
  scale_color_manual(values = season_colors) +
  scale_shape_manual(values = variable_shapes) +
  scale_linetype_manual(values = variable_linetypes) +
  scale_x_discrete(limits = month.abb) +
  scale_x_discrete(limits = month.abb) +
  labs(
    title = "Monthly Averages of Temperature, O3, PM10, and Dewpoint by Season",
    x = "Month",
    y = "Value",
    color = "Season",
    shape = "Variable",
    linetype = "Variable"
  ) +
  theme_minimal() +
  theme(legend.position = "right")
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```

```r
## Print the final plot
print(final_plot)
```



From the plot, variable *pm10* seems to have the least seasonal trend.

Problem 2

```r
## Define 'poly' S4 class
##' @field coefficients A numeric vector containing the coefficients of the polynomial.
##' @field exponents A numeric vector containing the exponents
##         of the polynomial corresponding to each coefficient.
setClass("poly",
         slots = c(coefficients = "numeric",
                   exponents = "numeric"))
```

```r
## Constructor for 'poly' class
##' @param poly_string A string representing the polynomial.
##' @return A `poly` object
make_poly <- function(poly_string){
  # Split the string at '+' or '-' signs, keeping the signs with the terms
  terms <- unlist(strsplit(poly_string,
                           split = "(?<=\\d)\\s*(?=[+-])|(?<=x)\\s*(?=[+-])",
                           perl = TRUE))
```

```r
  coeffs <- numeric()
  exps <- numeric()

  for (term in terms){
    # Detect if term is negative
    sign <- ifelse(grepl("^-", term), -1, 1)
    term <- gsub("^[+-]\\s*", "", term)

    # Apply the sign
    coeff <- ifelse(grepl("x", term), 1, 0)
    coeff <- ifelse(grepl("^\\d+", term), as.numeric(sub("x.*$", "", term)), coeff)
    coeff <- sign * coeff

    exp <- ifelse(grepl("x", term), 1, 0)
    exp <- ifelse(grepl("x\\^", term), as.numeric(sub(".*\\^", "", term)), exp)

    coeffs <- c(coeffs, coeff)
    exps <- c(exps, exp)
  }

  new("poly", coefficients = coeffs, exponents = exps)
}
```

```r
## Validator for `poly` class
##' @param object A `poly` object
##' @return TRUE if the object is valid, otherwise stop with an error message
setValidity("poly", function(object){
  # Check if the lengths of coefficients and exponents are equal
  if(length(object@coefficients) != length(object@exponents)) {
    stop("Lengths of coefficients and exponents should match")
  }

  return(TRUE)
})
```

```
Class "poly" [in ".GlobalEnv"]

Slots:

Name:  coefficients    exponents
Class:      numeric      numeric
```

```r
## Display 'poly' class objects
##' @title Display a `poly` object
##' @param object A `poly` object.
setMethod("show", "poly", function(object){
  # Order the terms by decreasing exponents
  order_index <- order(object@exponents, decreasing = TRUE)
  sorted_coeffs <- object@coefficients[order_index]
  sorted_exps <- object@exponents[order_index]

  # Combine terms with the same exponent
  unique_exps <- unique(sorted_exps)
  new_coeffs <- sapply(unique_exps, function(exp){
    sum(sorted_coeffs[sorted_exps == exp])
  })

  # String representation for each term
  terms <- mapply(function(coeff, exp){
    if (coeff == 0){
      return(NULL)
    } else if (exp == 0){
      return(as.character(coeff))
    } else if (exp == 1){
      return(ifelse(coeff == 1, "x", ifelse(coeff == -1, "-x", paste0(coeff, "x"))))
    } else{
      return(ifelse(coeff == 1, paste0("x^", exp),
                    ifelse(coeff == -1, paste0("-x^", exp),
                           paste0(coeff, "x^", exp))))
    }
  }, new_coeffs, unique_exps, SIMPLIFY = FALSE)

  # Filter out terms with coefficient zero
  terms <- Filter(Negate(is.null), terms)

  # Construct the complete polynomial string
  polynomial_string <- paste(terms, collapse = " ")
  polynomial_string <- gsub(" ([^\\-])", " + \\1", polynomial_string) #
  polynomial_string <- gsub("-", "- ", polynomial_string)
  cat(polynomial_string, "\n")

  return(invisible(object))
})
```

```r
##' @title Add two `poly` objects
##' @param e1 The first `poly` object
##' @param e2 The second `poly` object
##' @return A new `poly` object representing the sum of the two inputs
setMethod("+", signature("poly", "poly"), function(e1, e2){
  # Extract and combine unique exponents from both polynomials
  all_exps <- unique(c(e1@exponents, e2@exponents))
  new_coeffs <- numeric(length(all_exps))

  # Sum coefficients of terms with the same exponent
  for (exp in all_exps){
    new_coeffs[which(all_exps == exp)] <- sum(e1@coefficients[e1@exponents == exp],
                                              e2@coefficients[e2@exponents == exp])
  }

  # Sort the terms by decreasing exponent
  order_index <- order(all_exps, decreasing = TRUE)
  new_coeffs <- new_coeffs[order_index]
  all_exps <- all_exps[order_index]

  new("poly", coefficients = new_coeffs, exponents = all_exps)
})
```

```r
##' @title Subtract two `poly` objects
##' @param e1 The first `poly` object
##' @param e2 The second `poly` object
##' @return A new `poly` object representing the difference of the two inputs
setMethod("-", signature("poly", "poly"), function(e1, e2){
  # Extract and combine unique exponents from both polynomials
  all_exps <- unique(c(e1@exponents, e2@exponents))
  new_coeffs <- numeric(length(all_exps))

  # Subtract coefficients of terms with the same exponent
  for (exp in all_exps) {
    new_coeffs[which(all_exps == exp)] <- sum(e1@coefficients[e1@exponents == exp],
                                              -e2@coefficients[e2@exponents == exp])
  }

  # Sort the terms by decreasing exponent
  order_index <- order(all_exps, decreasing = TRUE)
  new_coeffs <- new_coeffs[order_index]
```

```
  all_exps <- all_exps[order_index]

  new("poly", coefficients = new_coeffs, exponents = all_exps)
})
```

```
# Test
p1 <- make_poly("3x^2 + 2")
p2 <- make_poly("7x^3 - 2x^2 - x + 17")
p3 <- new("poly", coefficients=c(17, -2, 0, 17), exponents=c(3, 2, 1, 0))
p4 <- new("poly", coefficients=c(-7, -4, 0, 7), exponents=c(3, 2, 1, 0))
p5 <- new("poly", coefficients = c(1, -4, 3, 3), exponents = c(3, 4, 1, 1))
p6 <- new("poly", coefficients = c(1, -4, 3), exponents = c(2, 3, 1))
p1
```

```
3x^2 + 2
```

```
p2
```

```
7x^3 - 2x^2 - x + 17
```

```
p3
```

```
17x^3 - 2x^2 + 17
```

```
p4
```

```
- 7x^3 - 4x^2 + 7
```

```
p5
```

```
- 4x^4 + x^3 + 6x
```

```
p1+p2
```

```
7x^3 + x^2 - x + 19
```

```r
p1-p2
```

```
- 7x^3 + 5x^2 + x - 15
```

```r
p3-p4
```

```
24x^3 + 2x^2 + 10
```

Problem 3

(a)

```r
library(nycflights13)
library(data.table)
```

```
Attaching package: 'data.table'

The following objects are masked from 'package:dplyr':

    between, first, last
```

```r
# Convert flights and airports data frames to data tables
flights_dt <- as.data.table(flights)
airports_dt <- as.data.table(airports)

## Departure table
# Calculate mean and median departure delay per airport
departure_delay_dt <- flights_dt[, .(
  mean_delay = mean(dep_delay, na.rm = TRUE),
  median_delay = median(dep_delay, na.rm = TRUE)
), by = .(origin)][order(-mean_delay)]

# Join with airports data to get airport names
departure_delay_dt <- departure_delay_dt[airports_dt,
                                         on = .(origin = faa),
                                         nomatch = 0][, .(name,
                                                          mean_delay,
```

```r
                                                              median_delay)]

# Print
print(departure_delay_dt, nrow(departure_delay_dt))
```

```
                  name mean_delay median_delay
1: Newark Liberty Intl   15.10795           -1
2: John F Kennedy Intl   12.11216           -1
3:          La Guardia   10.34688           -3
```

```r
## Arrival table
# Calculate mean, median delay, and flight count for each destination
arrival_delay_dt <- flights_dt[, .(
  mean_delay = mean(arr_delay, na.rm = TRUE),
  med_delay = median(arr_delay, na.rm = TRUE),
  numflights = .N
), by = .(dest)]

# Filter out destinations with under 10 flights
arrival_delay_dt <- arrival_delay_dt[numflights >= 10]

# Join with airports data to get airport names
arrival_delay_dt <- merge(arrival_delay_dt, airports_dt,
                          by.x = "dest", by.y = "faa", all.x = TRUE)

# Replace NA names with FAA codes
arrival_delay_dt[, name := fcoalesce(name, dest)]

# Select and arrange columns
arrival_delay_dt <- arrival_delay_dt[, .(name,
                                         mean_delay,
                                         med_delay)][order(-mean_delay)]

# Print
print(arrival_delay_dt, nrow(arrival_delay_dt))
```

```
                                 name  mean_delay med_delay
  1:              Columbia Metropolitan 41.76415094      28.0
  2:                         Tulsa Intl 33.65986395      14.0
  3:                  Will Rogers World 30.61904762      16.0
```

4:	Jackson Hole Airport	28.09523810	15.0
5:	Mc Ghee Tyson	24.06920415	2.0
6:	Dane Co Rgnl Truax Fld	20.19604317	1.0
7:	Richmond Intl	20.11125320	1.0
8:	Akron Canton Regional Airport	19.69833729	3.0
9:	Des Moines Intl	19.00573614	0.0
10:	Gerald R Ford Intl	18.18956044	1.0
11:	Birmingham Intl	16.87732342	-2.0
12:	Theodore Francis Green State	16.23463687	1.0
13:	Greenville-Spartanburg International	15.93544304	-0.5
14:	Cincinnati Northern Kentucky Intl	15.36456376	-3.0
15:	Savannah Hilton Head Intl	15.12950601	-1.0
16:	Manchester Regional Airport	14.78755365	-3.0
17:	Eppley Afld	14.69889841	-2.0
18:	Yeager	14.67164179	-1.5
19:	Kansas City Intl	14.51405836	0.0
20:	Albany Intl	14.39712919	-4.0
21:	General Mitchell Intl	14.16722038	0.0
22:	Piedmont Triad	14.11260054	-2.0
23:	Washington Dulles Intl	13.86420212	-3.0
24:	Cherry Capital Airport	12.96842105	-10.0
25:	James M Cox Dayton Intl	12.68048606	-3.0
26:	Louisville International Airport	12.66938406	-2.0
27:	Chicago Midway Intl	12.36422360	-1.0
28:	Sacramento Intl	12.10992908	4.0
29:	Jacksonville Intl	11.84483416	-2.0
30:	Nashville Intl	11.81245891	-2.0
31:	Portland Intl Jetport	11.66040210	-4.0
32:	Greater Rochester Intl	11.56064461	-5.0
33:	Hartsfield Jackson Atlanta Intl	11.30011285	-1.0
34:	Lambert St Louis Intl	11.07846451	-3.0
35:	Norfolk Intl	10.94909344	-4.0
36:	Baltimore Washington Intl	10.72673385	-5.0
37:	Memphis Intl	10.64531435	-2.5
38:	Port Columbus Intl	10.60132291	-3.0
39:	Charleston Afb Intl	10.59296847	-4.0
40:	Philadelphia Intl	10.12719014	-3.0
41:	Raleigh Durham Intl	10.05238095	-3.0
42:	Indianapolis Intl	9.94043412	-3.0
43:	Charlottesville-Albemarle	9.50000000	-5.0
44:	Cleveland Hopkins Intl	9.18161129	-5.0
45:	Ronald Reagan Washington Natl	9.06695204	-2.0
46:	Burlington Intl	8.95099602	-4.0

47:	Buffalo Niagara Intl	8.94595186	-5.0
48:	Syracuse Hancock Intl	8.90392501	-5.0
49:	Denver Intl	8.60650021	-2.0
50:	Palm Beach Intl	8.56297210	-3.0
51:	BQN	8.24549550	-1.0
52:	Bob Hope	8.17567568	-3.0
53:	Fort Lauderdale Hollywood Intl	8.08212154	-3.0
54:	Bangor Intl	8.02793296	-9.0
55:	Asheville Regional Airport	8.00383142	-1.0
56:	PSE	7.87150838	0.0
57:	Pittsburgh Intl	7.68099053	-5.0
58:	Gallatin Field	7.60000000	-2.0
59:	NW Arkansas Regional	7.46572581	-2.0
60:	Tampa Intl	7.40852503	-4.0
61:	Charlotte Douglas Intl	7.36031885	-3.0
62:	Minneapolis St Paul Intl	7.27016886	-5.0
63:	William P Hobby	7.17618819	-4.0
64:	Bradley Intl	7.04854369	-10.0
65:	San Antonio Intl	6.94537178	-9.0
66:	South Bend Rgnl	6.50000000	-3.5
67:	Louis Armstrong New Orleans Intl	6.49017497	-6.0
68:	Key West Intl	6.35294118	7.0
69:	Eagle Co Rgnl	6.30434783	-4.0
70:	Austin Bergstrom Intl	6.01990875	-5.0
71:	Chicago Ohare Intl	5.87661475	-8.0
72:	Orlando Intl	5.45464309	-5.0
73:	Detroit Metro Wayne Co	5.42996346	-7.0
74:	Portland Intl	5.14157973	-5.0
75:	Nantucket Mem	4.85227273	-3.0
76:	Wilmington Intl	4.63551402	-7.0
77:	Myrtle Beach Intl	4.60344828	-13.0
78:	Albuquerque International Sunport	4.38188976	-5.5
79:	George Bush Intercontinental	4.24079040	-5.0
80:	Norman Y Mineta San Jose Intl	3.44817073	-7.0
81:	Southwest Florida Intl	3.23814963	-5.0
82:	San Diego Intl	3.13916574	-5.0
83:	Sarasota Bradenton Intl	3.08243131	-5.0
84:	Metropolitan Oakland Intl	3.07766990	-9.0
85:	General Edward Lawrence Logan Intl	2.91439222	-9.0
86:	San Francisco Intl	2.67289152	-8.0
87:	SJU	2.52052659	-6.0
88:	Yampa Valley	2.14285714	2.0
89:	Phoenix Sky Harbor Intl	2.09704733	-6.0

```
 90:            Montrose Regional Airport   1.78571429     -10.5
 91:                    Los Angeles Intl   0.54711094      -7.0
 92:              Dallas Fort Worth Intl   0.32212685      -9.0
 93:                          Miami Intl   0.29905978      -9.0
 94:                      Mc Carran Intl   0.25772849      -8.0
 95:                Salt Lake City Intl   0.17625459      -8.0
 96:                          Long Beach  -0.06202723     -10.0
 97:                 Martha\\\\'s Vineyard  -0.28571429     -11.0
 98:                Seattle Tacoma Intl  -1.09909910     -11.0
 99:                      Honolulu Intl  -1.36519258      -7.0
100:                                 STT  -3.83590734      -9.0
101:            John Wayne Arpt Orange Co  -7.86822660     -11.0
102:                   Palm Springs Intl -12.72222222     -13.5
                                    name   mean_delay med_delay
```

(b)

```r
## Convert planes data frames to data tables
planes_dt <- as.data.table(planes)

## Join flights with planes
fastest_model_dt <- flights_dt[planes_dt, on = .(tailnum), nomatch = 0]

## Ensure air_time and distance are numeric
fastest_model_dt[, c("air_time", "distance") := list(as.numeric(air_time),
                                                     as.numeric(distance))]

## Calculate speed in mph
fastest_model_dt[!is.na(time) & !is.na(distance),
                 mph := distance / (air_time / 60)]
```

```
Warning in is.na(time): is.na() applied to non-(list or vector) of type
'closure'
```

```r
## Group by model and calculate average mph and flight count
fastest_model_dt <- fastest_model_dt[, .(
  avgmph = mean(mph, na.rm = TRUE),
  nflights = .N
), by = .(model)][order(-avgmph)][1]
```

```r
## Print the model with the fastest average speed
print(fastest_model_dt)
```

```
     model   avgmph nflights
1: 777-222 482.6254        4
```