Homework 4 Solutions

Stats 32: Introduction to R for Undergraduates

Due: Thursday, May 2, 2024, 11:59 am PT

Instructions

You may (in fact, are encouraged to) use the Internet (including AI assistants, like ChatGPT) to search up any information to help you with this assignment, though you must cite any external (i.e. non-course related) resources that you use. Similarly, *after attempting this assignment by yourself*, you may collaborate with other students in the course, but you must each write your own code and acknowledge all students with whom you collaborated *for each problem* (you don't need to cite by subpart). However, you may not post on Internet forums (e.g. Stack Exchange) for help with this assignment; doing so is considered an Honor Code violation. You also may not copy verbatim any significant amount of code from the Internet (including AI assistants, like ChatGPT), even with citation. Feeding in the problems directly into AI assistants (or substantively paraphrased version) is also not permitted.

Please provide your code responses to each problem in the `.Rmd` file in the R code chunks directly below each subpart, inserting additional R code chunks if needed. Any text response can go right underneath the corresponding question.

On Gradescope, please submit a single `.pdf` file created by knitting the document with your responses. Problem 0 will provide guidance on how to do this.

Credit is given based on the approach and code, not necessarily the final answer.

All visualizations should have an appropriate title and axis labels.

Problem 1: Coffee and mental math

In November 2021, I ran an experiment to see whether coffee would improve my mental math performance. I took a mental math challenge 6 times a day for 12 days. On each day I randomly assigned myself a coffee dosage of 0 mL, 125 mL, or 250 mL at 12 noon. Then I took the test twice at 3 pm, 6 pm, and 9 pm. `coffee.csv` contains all of my scores.

(a) (2 points) Read in `coffee.csv`. You will note that the data is not in "tidy" format; there is 1 row for each date, but to get "tidy" data we want 1 row for each observation (i.e. each time I took a test). Use `pivot_longer()` to convert the data into tidy format. The end result, which you should store in a variable called `coffee`, should have a total of 5 columns: `Date`, `Dosage`, `Time` (3pm, 6pm, or 9pm), `Repetition` (R1 or R2), and `Score` (the test score). Hint: You may want to use the `names_sep` argument.

Answer:

```
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   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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 error
```

```
coffee <- read_csv("coffee.csv") %>%
  pivot_longer(cols=!c("Date", "Dosage"),
               names_to = c("Time", "Repetition"),
               names_sep = c(" "),
               values_to = "Score")
```

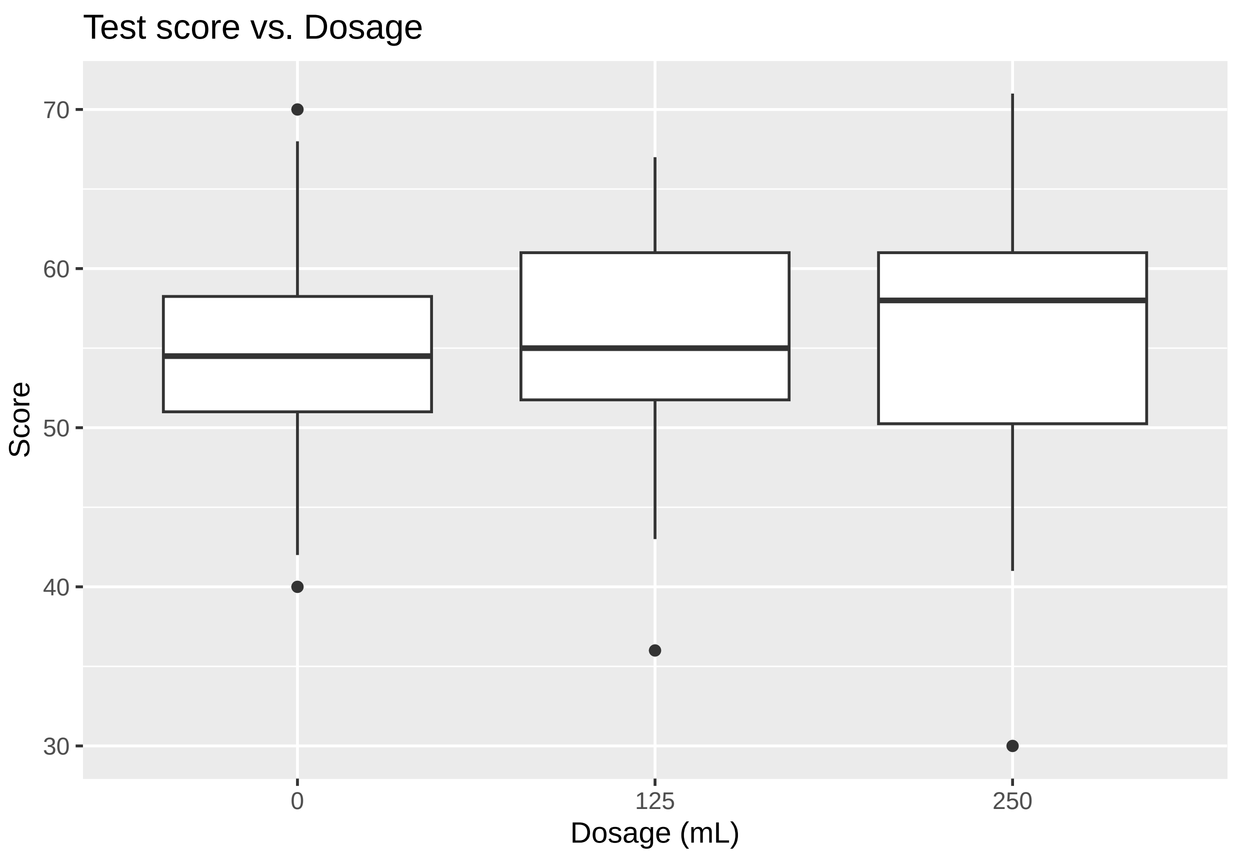
```
## Rows: 12 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Date
## dbl (7): Dosage, 3pm R1, 3pm R2, 6pm R1, 6pm R2, 9pm R1, 9pm R2
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

(b) (3 points) We consider `Dosage` as a categorical variable. Convert that column to a factor. Then create an appropriate visualization that shows the distribution of test scores for different dosages.

Answer: Since we are trying to visualize a quantitative variable vs. a categorical variable, we can use side-by-side boxplots.

```
coffee <- coffee %>%
  mutate(Dosage=as.factor(Dosage))
```

```
coffee %>%
  ggplot(aes(x=Dosage, y=Score)) +
  geom_boxplot() +
  labs(x="Dosage (mL)", y="Score", title="Test score vs. Dosage")
```



(c) (2 points) Fit a simple linear regression model to predict test score based on `Dosage`.

Answer:

```
model <- lm(Score ~ Dosage, data=coffee)
```

(d) (2 points) Compute the predictions for each of the 3 possible values of `Dosage`, under the model in (b). Verify, using some `dplyr` verbs, that these predictions are equivalent to the average score across all test scores from days with the particular coffee dosage.

Answer: We start with the predictions.

```
predict(model, newdata=tibble(Dosage=c("0", "125", "250")))
```

```
##        1        2        3
## 55.37500 55.54167 55.66667
```

We verify these predictions correspond to the mean test score per dosage using `group_by()` and `summarise()`:

```
coffee %>%
  group_by(Dosage) %>%
  summarise(mean_scores = mean(Score))
```

```
## # A tibble: 3 x 2
##   Dosage mean_scores
##   <fct>        <dbl>
## 1 0             55.4
## 2 125           55.5
## 3 250           55.7
```

(e) (3 points) Now, fit a multiple regression model using all three predictors `Dosage`, `Repetition`, and

Time, all categorical. Interpret each of the resulting coefficient estimates.

Answer:

```r
summary(lm(Score ~ Dosage + Repetition + Time, data=coffee))
```

```
##
## Call:
## lm(formula = Score ~ Dosage + Repetition + Time, data = coffee)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.9861  -4.4479   0.6111   5.2396  16.0139
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   56.5417     2.4269  23.298   <2e-16 ***
## Dosage125      0.1667     2.4269   0.069    0.945
## Dosage250      0.2917     2.4269   0.120    0.905
## RepetitionR2   0.7778     1.9816   0.393    0.696
## Time6pm       -2.6250     2.4269  -1.082    0.283
## Time9pm       -2.0417     2.4269  -0.841    0.403
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.407 on 66 degrees of freedom
## Multiple R-squared:  0.02163,    Adjusted R-squared:  -0.05249
## F-statistic: 0.2918 on 5 and 66 DF,  p-value: 0.9159
```

The intercept of 56.542 corresponds to the predicted score on the first repetition of the test taken at 3pm with no coffee. The Dosage: 125 coefficient suggests that keeping Repetition and Time constant, a 125 mL dosage of coffee tended to increase my score by 0.167 relative to no coffee. The Dosage: 250 coefficient suggests that keeping Repetition and Time constant, a 250 mL dosage of coffee tended to increase my score by 0.292 relative to no coffee. The Repetition: R2 coefficient suggests that keeping Dosage and Time constant, the second repetition tended to have 0.778 higher score than the first. The Time: 6pm coefficient suggests that keeping Dosage and Repetition constant, the 6 pm test tended to have 2.625 lower score than the 3pm test. The Time: 9pm coefficient suggests that keeping Dosage and Repetition constant, the 9 pm test tended to have 2.042 lower score than the 3pm test.

Problem 2: Temperatures in San Jose

Daily high and low Fahrenheit temperatures in San Jose for 8,179 dates between 1998 and 2020 can be found in `san_jose.csv` (they are taken from Climate Data Online, at the station USW00023293). Your goal will be to see if you can predict the 2020 temperatures based on the data from 1998 to 2019.

(a) (2 points) Read the data into a tibble and create a new column called `day_of_year` that converts the date to the day of the year. For example, January 4 would be the 4th day of the year, March 1 would be the 60th day of the year in non-leap years and the 61st day in leap years Hint: The `yday()` function in the `lubridate` package may be helpful.

Answer:

```r
library(lubridate)
san_jose <- read_csv("san_jose.csv") %>%
  mutate(day_of_year = yday(DATE))
```

```
## Rows: 8179 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (2): TMAX, TMIN
## date (1): DATE
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(san_jose)
```

```
## # A tibble: 6 x 4
##   DATE        TMAX  TMIN day_of_year
##   <date>     <dbl> <dbl>       <dbl>
## 1 1998-07-04    79    56         185
## 2 1998-07-05    78    53         186
## 3 1998-07-06    84    55         187
## 4 1998-07-07    88    59         188
## 5 1998-07-08    75    57         189
## 6 1998-07-09    71    57         190
```

(b) (2 points) Fit a simple linear regression model to predict the high temperature based on `day_of_year` **using only the data from 1998 through 2019**. Provide graphical evidence that suggests the linear model is a poor fit.

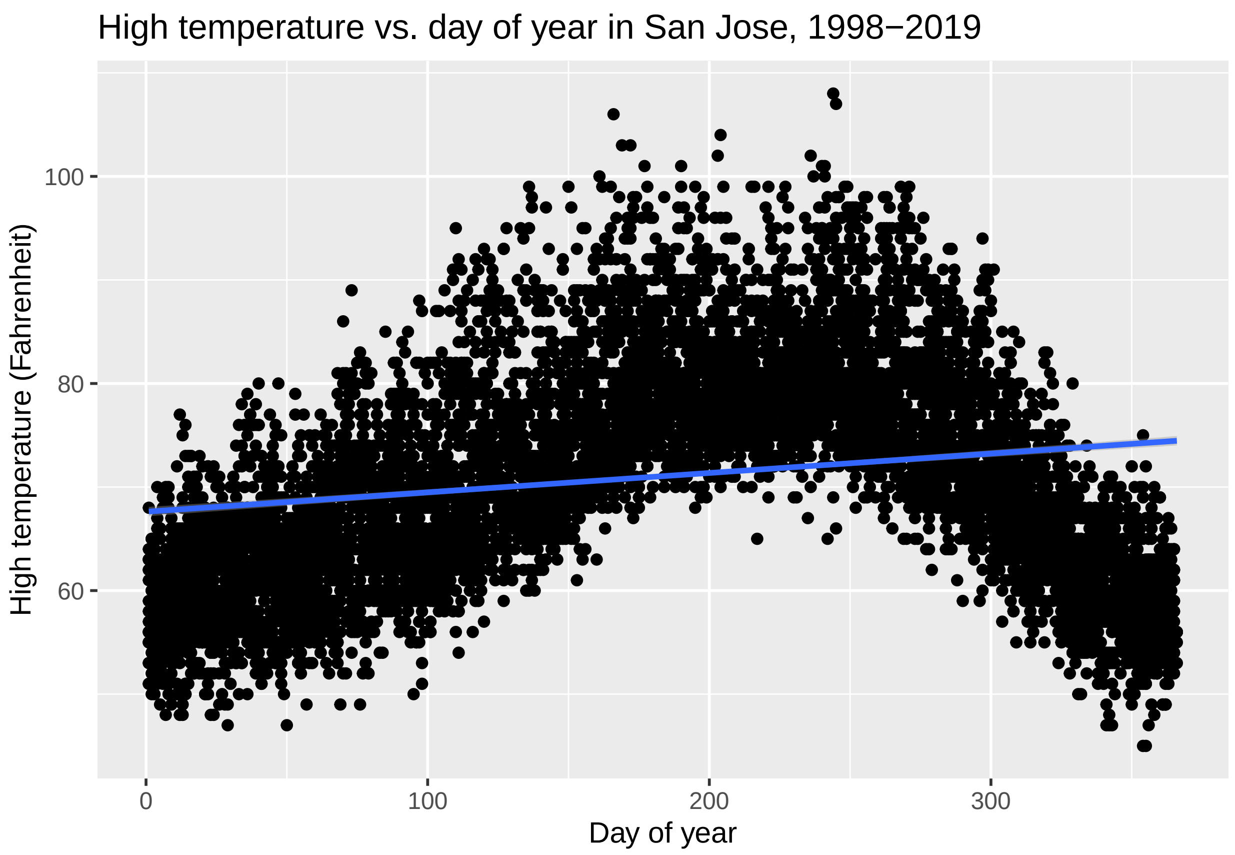
Answer:

```r
san_jose_train <- san_jose %>%
  mutate(YEAR=as.numeric(substr(DATE, 1, 4))) %>%
  filter(YEAR >= 1998 & YEAR <= 2019)
tmax_lm <- lm(TMAX ~ day_of_year, data=san_jose_train)
```

We construct a scatterplot of `TMAX` vs `day_of_year` and see that the linear trend is a poor fit to the data, which shows a clear nonlinear pattern. This is not surprising as we know the summer (when temperatures should tend to be the highest) occurs in the middle of the year.

```r
san_jose_train %>%
  ggplot(aes(x=day_of_year, y=TMAX)) +
  geom_point() +
  geom_smooth(method="lm") +
  ggtitle("High temperature vs. day of year in San Jose, 1998-2019") +
  xlab("Day of year") +
  ylab("High temperature (Fahrenheit)")
```

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



(c) (3 points) Fit a better model for predicting the high temperature based on `day_of_year` **using only the data from 1998 through 2019**. Explain why your model choice is better than the simple linear regression in part (b) for this data, based on your visualizations in part (b).

Answer: With the nonlinear trend seen above, we fit a locally weighted regression via `loess()`. If we wanted to be principled, we could set the `span` parameter based on cross validation as discussed in week 5, but that's not required for this homework; the default span looks visually okay.

```
tmax_loess <- loess(TMAX ~ day_of_year, data=san_jose_train)
```

(d) (2 points) Repeat part (c) for the low temperature. What is the predicted low temperature on May 4, 2023?

Answer:

```
tmin_loess <- loess(TMIN ~ day_of_year, data=san_jose_train)
predict(tmin_loess, newdata=tibble("day_of_year"=yday("2023-05-04")))
```

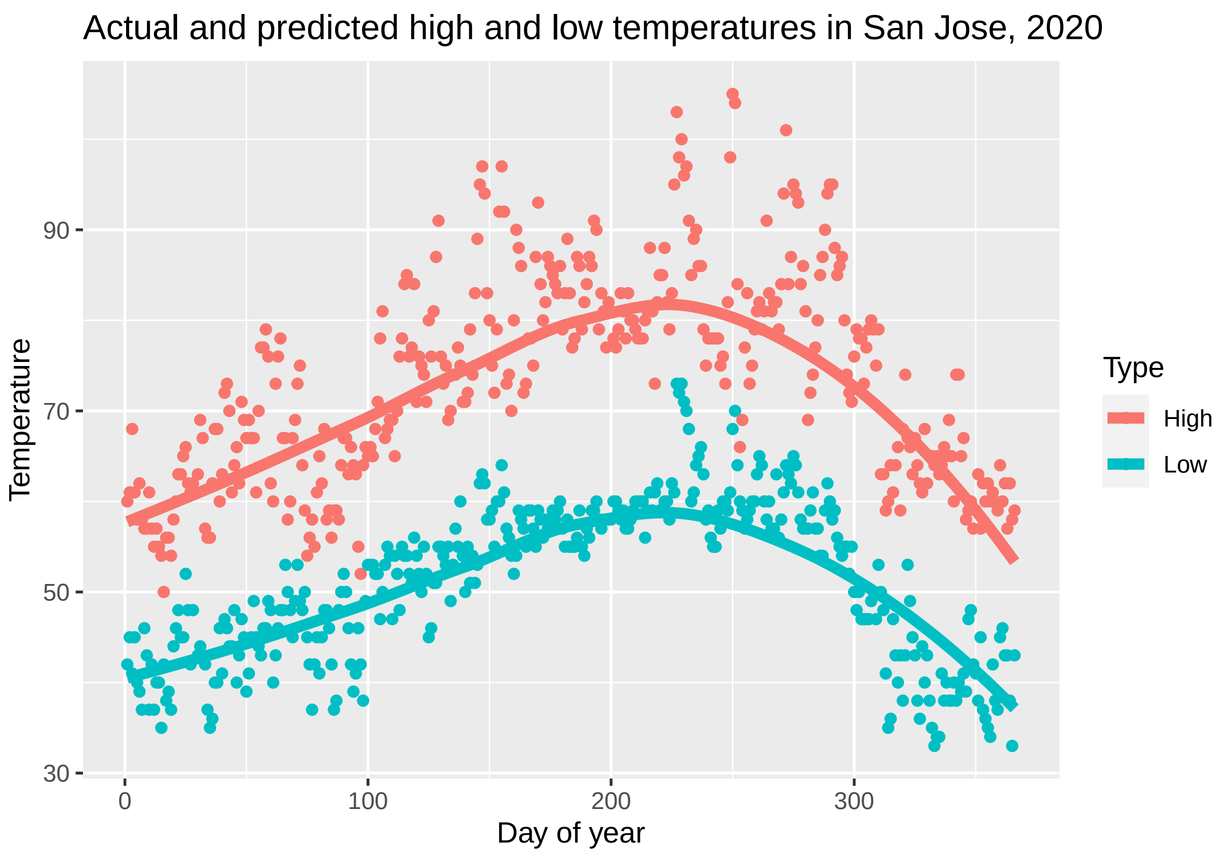
```
##       1
## 51.2141
```

(e) (4 points) Create a scatterplot with date on the horizontal axis and (actual) high and low temperatures on the vertical axis. Only include data from 2020 (i.e. the year for which we did not train the model). Color the points corresponding to high temperatures in red, and the points corresponding to low temperatures in blue. Then, compute high and low temperature predictions for all dates in 2020 using the models you fit in parts (c) and (d). Plot those predictions as lines on the same axes (red for high temperatures, blue for low temperatures). Make sure there is a legend for these colors.

Answer: To the `san_jose` tibble (filtered out to only the year 2020), we add columns with the predictions of the models from parts (c) and (d). We use `pivot_longer()` to create a tibble that has `TMAX` and `TMIN` in a single column called `Temperature`, which will make the plotting easier (as our vertical axis for the plot will need to accommodate both types of temperatures).

```
san_jose_test <- san_jose %>%
  mutate(YEAR=as.numeric(substr(DATE, 1, 4)),) %>%
  filter(YEAR == 2020)
high_predictions <- predict(object=tmax_loess, newdata=san_jose_test$day_of_year)
low_predictions <- predict(object=tmin_loess, newdata=san_jose_test$day_of_year)
san_jose_plot <- san_jose_test %>%
  mutate(Predicted_High=high_predictions,
         Predicted_Low=low_predictions) %>%
  rename(High=TMAX, Low=TMIN) %>%
  pivot_longer(cols=c("High", "Low"),
               names_to="Type",
               values_to="Temperature") %>%
  mutate(Prediction=ifelse(Type=="High", Predicted_High, Predicted_Low))
```

```
san_jose_plot %>%
  ggplot(aes(x=day_of_year, y=Temperature)) +
  geom_point(aes(color=Type)) +
  geom_line(aes(y=Prediction, color=Type), linewidth=2) +
  xlab("Day of year") +
  ggtitle("Actual and predicted high and low temperatures in San Jose, 2020")
```


Actual and predicted high and low temperatures in San Jose, 2020
Temperature
Day of year
Type
High
Low