# Fall 2020 STAT 240 Midterm Exam

**Due Friday, October 16, 2020 at 11:59 PM CT (local time in Madison, WI)**

**NAME: Kyle Yeo**

### Preliminaries

- You will have 24 hours to complete the exam, and your solutions should be uploaded to Canvas by 11:59 PM CT (the time zone in Madison, WI).
- You are not allowed to communicate with anyone using any means (email, phone, text, social media, online discussion platforms, etc.) except the instructors of this course. You are allowed to use materials from the course and the internet.
- If you have a question during the exam, post your question as a *private* post on Piazza. To do this, select the "Individual Student(s) / Instructor(s)" option next to "Post to:" when creating your post.
- It is recommended that you begin the exam as soon as possible and read over it to see if you have any questions. You can expect for questions to be addressed during normal working hours in Madison, WI (9 AM CT - 5 PM CT). Questions posted outside that window *may* still be addressed.

### Submission

Once you have completed the exam, knit the R Markdown document to create an HTML file. To submit this Exam, go to our Canvas site and select "Assignments" on the left panel, and upload both the edited Rmd and HTML files to the place designated for the midterm exam. *Be sure to review the HTML to verify all your answers appear as you expect.*

### Data

The following data files are needed to complete this exam. More information about the data sets are provided later in the exam.

Election Data:
`states.csv`

Baseball Data:
`baseball_players.csv`
`baseball_salaries.csv`
`baseball_allstar.csv`

# Problems

## The exam has a total of six problems, some with several parts, totaling 55 points.

## Short Answer Questions

### Problem 1 (3 points)

Which lubridate function would you use to convert the string "03-Mar-2020" to a date?

dmy("03-Mar-2020")

### Problem 2 (3 points)

A data frame `groceries()` has three columns named `food`, `category` and `price`. The `food` column is categorical and unique for each row. The `category` column is categorical and has six different values. The `price` column is numerical.

What code would you add where it says "## write code here" in order to find the foods and prices of the highest priced food item in each category?

```
groceries %>%
  group_by(category) %>%
  slice_max(price, n = 1)
```

### Problem 3 (4 points)

Write a regular expression that matches part of a string which contains one or more plus signs followed by exactly three digits. Then, represent this regular expression as a string as you would need in order to use it in R.

"\+{1,}\d{3}"

## Election Data Questions

Data for the next two questions are contained in a file about US states.

- *states.csv*
  - This data set has one row for each of the 50 US states.
  - The first column is the name of the state.
  - The next two columns have `region` and `division`, two partitions of the 50 US states defined by the US Census Bureau.
    - There are a total of four regions and nine divisions.
    - Each region contains one or more divisions.
    - The map below shows these regions of the country.
  - The variable `urban_index` is a quantitative measure of urbanity in a state.
    - The value corresponds to the base 10 logarithm of the average number of people who live within five miles or each resident of the state (using centroids of census tracts).

    - A value of 5.0 means that the average number of people living within 5 miles of each person in the population is 100,000, for example, while a value of 4.0 means that this average is 10,000 people.
- The variable `lean` is a string which indicates over the past decade, or so, how have votes in national elections in the state leaned, relative to the national average in the same election, when comparing the two primary political parties and averaging over multiple elections.
    - The two primary political parties in the US are the Democratic (D) and Republican (R) parties.
    - A value of `R+27` means the difference between the percentages of Republican and Democratic votes is 27 percentage points higher than the national difference, on average.
    - A value of `D+24` means the difference between Democratic and Republican percentages is 24 percentage points higher in favor of Democrats, on average.
    - The value `Even` means the average difference is zero.
- The next two columns are the percentages of likely voters in a recent poll about the upcoming US presidential election who favor the Democratic candidate, Joe Biden, (`poll_2020_D`), or the Republican candidate, Donald Trump, (`poll_2020_R`).
    - Note that these values may not sum to 100 as some voters are undecided and some favor a candidate from a minor political party.
- The last column `pct_bach` is the percentage of adults aged 25 and older who hold a bachelor's degree.

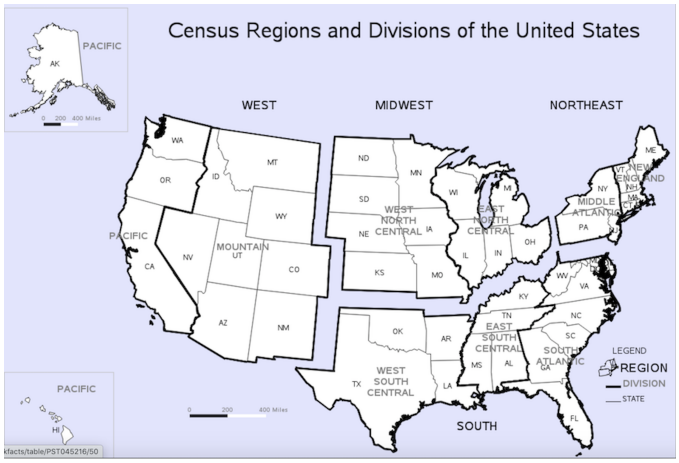



Figure 1: US Regions and Divisions

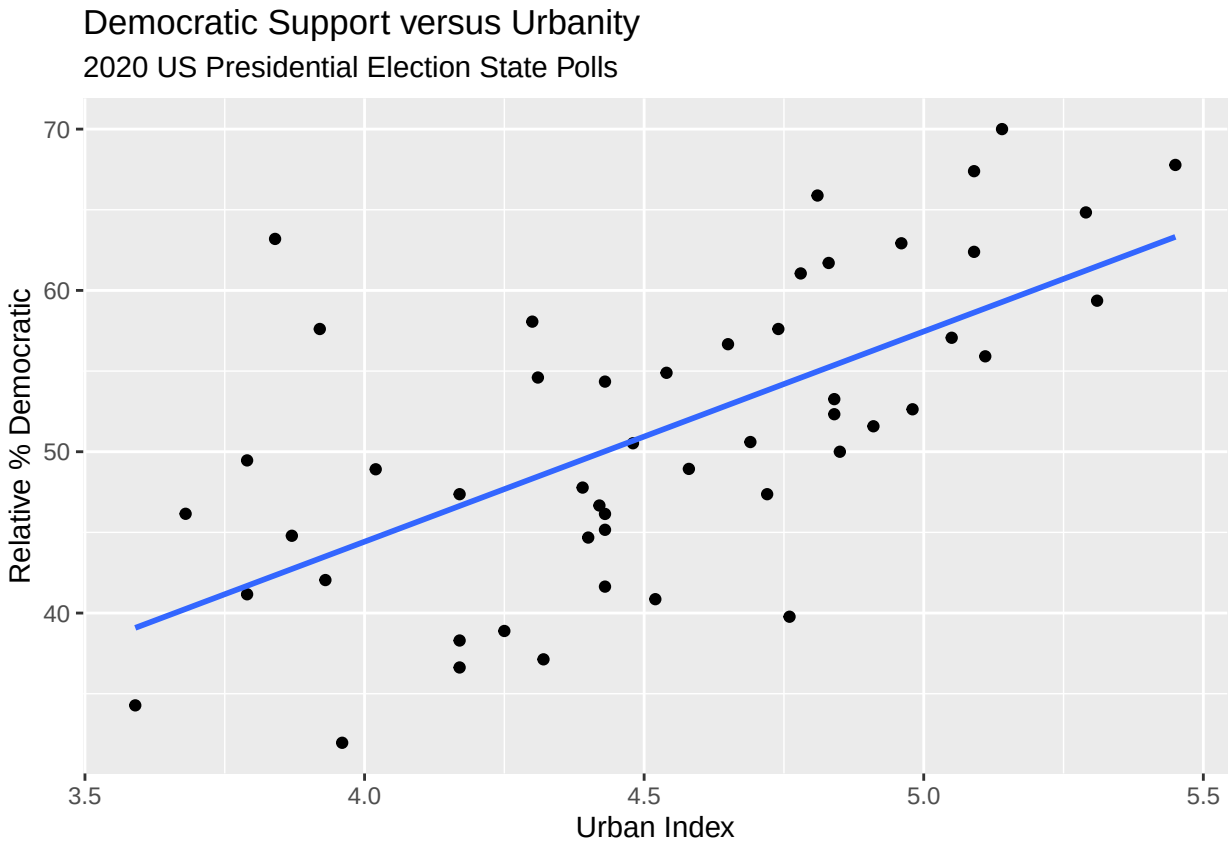
## Problem 4

**(a)** (5 points) The following chunk of code modifies the data frame from the file `states.csv` to create a new variable `pct_dem` with the relative support for the Democratic presidential candidate among likely voters with support for a candidate from the two major parties and begins a plot. Complete the code to create a scatter plot with this variable on the y-axis, the urban index on the x axis, and a straight line with no ribbon which models the trend of the data.

```
states <- read_csv("C:/stat_240/data/states.csv")
states = states %>%
```

```
  mutate(pct_dem = 100 * poll_2020_D / (poll_2020_D + poll_2020_R))

ggplot(states, aes(x=urban_index,y=pct_dem)) +
  geom_point() +
  geom_smooth(method = "lm", se=FALSE) +
  xlab("Urban Index") +
  ylab("Relative % Democratic") +
  ggtitle("Democratic Support versus Urbanity",
          subtitle = "2020 US Presidential Election State Polls")
```




**(b)** (5 points) Calculate the average values of `pct_dem` and `urban_index` for each region. Display the data frame with these values, arranged by `urban_index` from smallest to largest. (You answer should be a 4 by 3 table).

```
states <- read_csv("C:/stat_240/data/states.csv")

states_4 <- states %>%
  mutate(pct_dem = 100 * poll_2020_D / (poll_2020_D + poll_2020_R)) %>%
  group_by(region) %>%
  summarise(mean_pct = mean(pct_dem), mean_ur = mean(urban_index)) %>%
  arrange(mean_ur)

states_4
```

```
## # A tibble: 4 x 3
```

```
##   region    mean_pct mean_ur
##   <chr>        <dbl>   <dbl>
## 1 Midwest       47.3    4.42
## 2 South         48.0    4.46
## 3 West          51.9    4.52
## 4 Northeast     61.1    4.76
```

**(c)** (5 points) Below are five statements which interpret the previous results. Keep those that are justified and delete those that are not.

1. Likely voters in states that contain more urban areas tend to favor the Democratic US presidential candidate in the upcoming 2020 election.
2. Among all US Census regions, the Midwest is the least urban and the least supportive of the Democratic candidate in the 2020 US presidential election.
3. The Northeast region of the country has a much higher average level of support for the Democratic candidate in the 2020 US presidential election than any other region of the country.

### Problem 5

**(a)** (5 points) Using the data from the `states.csv`, write a code chunk to add a quantitative variable `lean_d` which measures the amount the state leans to the Democratic party using values in `lean` in the following way.

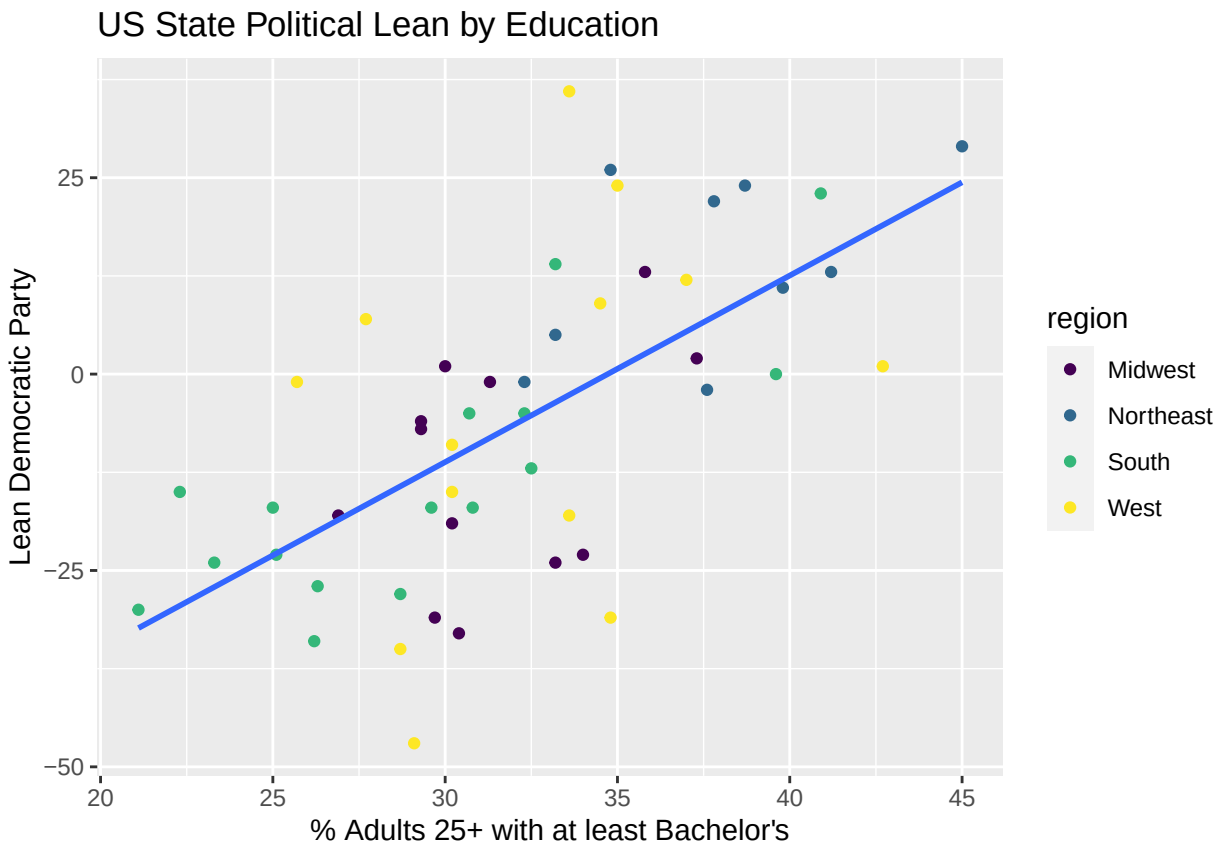
- If the state leans Republican, such as "R+27", the value should be -27.
- If the state leans Democratic, such as "D+24", the value should be 24.
- If the state leans evenly with `lean` equal to "EVEN", the value should be 0.

```
states_5 <- states %>%
  mutate(lean_d = case_when(str_detect(lean, "^R") ~ str_c("-", str_sub(lean, 3,5)),
        str_detect(lean, "^D") ~ str_sub(lean, 3,5),
        str_detect(lean, "^E") ~ "0")) %>%
  mutate(lean_d = as.numeric(lean_d))
states_5
```

```
## # A tibble: 50 x 9
##    state   region  division   urban_index lean  poll_2020_D poll_2020_R pct_bach
##    <chr>   <chr>   <chr>            <dbl> <chr>       <dbl>       <dbl>    <dbl>
##  1 Alabama South   East Sout~        4.17 R+27           36          58     26.3
##  2 Alaska  West    Pacific           3.79 R+15           46          47     30.2
##  3 Arizona West    Mountain          4.91 R+9            49          46     30.2
##  4 Arkans~ South   West Sout~        4.02 R+24           45          47     23.3
##  5 Califo~ West    Pacific           5.29 D+24           59          32     35
##  6 Colora~ West    Mountain          4.84 D+1            49          43     42.7
##  7 Connec~ Northe~ New Engla~        4.96 D+11           56          33     39.8
##  8 Delawa~ South   South Atl~        4.78 D+14           58          37     33.2
##  9 Florida South   South Atl~        4.98 R+5            50          45     30.7
## 10 Georgia South   South Atl~        4.58 R+12           46          48     32.5
## # ... with 40 more rows, and 1 more variable: lean_d <dbl>
```

**(b)** (5 points) Modify the existing code to make scatter plot with `pct_bach` on the x axis and `lean_d` on the y-axis. Color points in differently by `region` and add a single regression line to the plot.

```
ggplot(states_5, mapping = aes(x = pct_bach, lean_d)) +
  geom_point(aes(color = region)) +
  geom_smooth(aes(group = 1), method = "lm", se = FALSE) +
  xlab("% Adults 25+ with at least Bachelor's") +
  ylab("Lean Democratic Party") +
  ggtitle("US State Political Lean by Education")
```




**(c)** (2 points) Which region of the country have the largest outliers from the trend line?

According to the above plot, West has the largest outlier.

## Baseball Data Question

Data for the next question are contained in three files obtained from Sean Lahman's Baseball Database. Variable names that are shared among the data sets have common values. For example, any given player will have the same value of `playerID` in all three data sets.

- *baseball_players.csv*
  - This data file contains information on individual players.

  Variables:
  - `playerID` - A unique code assigned to each player

  - `birthYear` - The year the player was born
  - `deathYear` - The year the player died
  - `nameFirst` - The first name of the player
  - `nameLast` - The last name of the player
  - `bats` - The player's batting hand
  - `throws` - The player's throwing hand
  - `debut` - The date of the player's first major league appearance
  - `finalGame` - The date of the player's final major league appearance (it is blank if the player is still active)
- *baseball_salaries.csv*
  - This data file contains information on players salaries by year.

  Variables:
  - `yearID` - Year
  - `teamID` - The major league team
  - `lgID` - The league (American League or National League)
  - `playerID` - A unique code assigned to each player
  - `salary` - The player's salary
- *baseball_allstar.csv*
  - This data file contains the players who were selected to be on an All Star team, which is a special game that (typically) happens around mid-season.
  - A player can be (and often is) selected multiple years and there is a row for each year a player is selected.
  - There are different ways a player can get selected for an All Star team, and generally the best players make the roster. (Of course who the best players are is debatable! If you are unfamiliar with All-Star games, just think of the All-Star players as those who are among the top performers in their sport.)

  Variables
  - `playerID` - A unique code assigned to each player
  - `yearID` - Year
  - `teamID` - The major league team
  - `lgID` - The league (American League or National League)

**Problem 6**

Do *All Star* players generally have a higher salary than *non-All Star* players?

The following question parts work toward analyzing this question.

**(a)** (4 points) Consider the salaries of players who had their first game (`debut`) in the major leagues after January 1, 1985 and who played their last game (`finalGame`) before January 1, 2015, and for whom salary information is available. Transform and combine the necessary data sets so that you have a data frame with `playerID`, `yearID`, `salary`, `debut`, `finalGame`, which only includes the players with their first and final games within the noted dates, and have with a salary strictly greater than $0. Also, only consider the salaries in the years 1985 - 2014. You can modify the existing code below.

An observation in the resulting data frame can be identified by `playerID` and `yearID`(because a `playerID` may have multiple years with salary information).

Use the function `head()` to print the first 6 rows of this data frame.

```
## Read in the appropriate data files
players <- read_csv("C:/stat_240/data/baseball_players.csv")
salaries <- read_csv("C:/stat_240/data/baseball_salaries.csv")

## Find players based on debut and finalGame
players_1 <- players %>%
  select(playerID, debut, finalGame) %>%
  drop_na() %>%
  filter(debut > ymd("1985-01-01") & finalGame < ymd("2015-01-01"))

## Create salary data frame
salaries_1 <- salaries %>%
  select(playerID, yearID, salary) %>%
  filter(salary > 0 & yearID < 2015)

## Build requested data frame
df_1 <- left_join(players_1, salaries_1, by = "playerID") %>%
  drop_na()

head(df_1)
```

```
## # A tibble: 6 x 5
##   playerID  debut      finalGame  yearID salary
##   <chr>     <date>     <date>      <dbl>  <dbl>
## 1 abadan01  2001-09-10 2006-04-13   2006 327000
## 2 abbotje01 1997-06-10 2001-09-29   1998 175000
## 3 abbotje01 1997-06-10 2001-09-29   1999 255000
## 4 abbotje01 1997-06-10 2001-09-29   2000 255000
## 5 abbotje01 1997-06-10 2001-09-29   2001 300000
## 6 abbotji01 1989-04-08 1999-07-21   1989  68000
```

**(b)** (4 points) Next add a variable `allstar_status` that takes the value `allstar` if the player ever

was selected to be on an All Star team and `not_allstar` if the player was never selected to be on an All Star team; the `baseball_allstar.csv` data file contains the All Star appearances by players so you can assume if a player appears in `baseball_allstar.csv` then they should be labeled an `allstar` player. You can modify the existing code below.

Use the function `head()` to print the first 6 rows of this data frame.

```
## Read in the appropriate data file
allstar <- read_csv("C:/stat_240/data/baseball_allstar.csv")

## Find unique playerIDs in allstar data frame
## This includes a count of the number appearances per playerID
allstar_1 <- allstar %>%
  group_by(playerID) %>%
  summarize(n = n())



## Build requested data frame
df_1 <- df_1 %>%
  mutate(allstar_status = ifelse(playerID %in% allstar_1$playerID, "allstar", "not_allstar"
  drop_na()
head(df_1)
```

```
## # A tibble: 6 x 6
##   playerID  debut      finalGame  yearID salary allstar_status
##   <chr>     <date>     <date>      <dbl>  <dbl> <chr>
## 1 abadan01  2001-09-10 2006-04-13   2006 327000 not_allstar
## 2 abbotje01 1997-06-10 2001-09-29   1998 175000 not_allstar
## 3 abbotje01 1997-06-10 2001-09-29   1999 255000 not_allstar
## 4 abbotje01 1997-06-10 2001-09-29   2000 255000 not_allstar
## 5 abbotje01 1997-06-10 2001-09-29   2001 300000 not_allstar
## 6 abbotji01 1989-04-08 1999-07-21   1989  68000 not_allstar
```

**(c)** (4 points) Add a variable called `period` that divides up the years from 1985 through 2014 into groups of five consecutive years ("1985-1989", "1990-1994", . . . , "2010-2014"). You can modify the existing code below.

Use the function `head()` to print the first 6 rows of this data frame.

```
## Set labels for period variable
breaks <- seq(1984,2014,5)
labels <- str_c((breaks+1)[-length(breaks)],breaks[-1],sep="-")

## Build requested data frame
df_1 <- df_1 %>%
  mutate(period = case_when(yearID >= 1985 & yearID <= 1989 ~ labels[1],
                            yearID >= 1990 & yearID <= 1994 ~ labels[2],
                            yearID >= 1995 & yearID <= 1999 ~ labels[3],
                            yearID >= 2000 & yearID <= 2004 ~ labels[4],
```

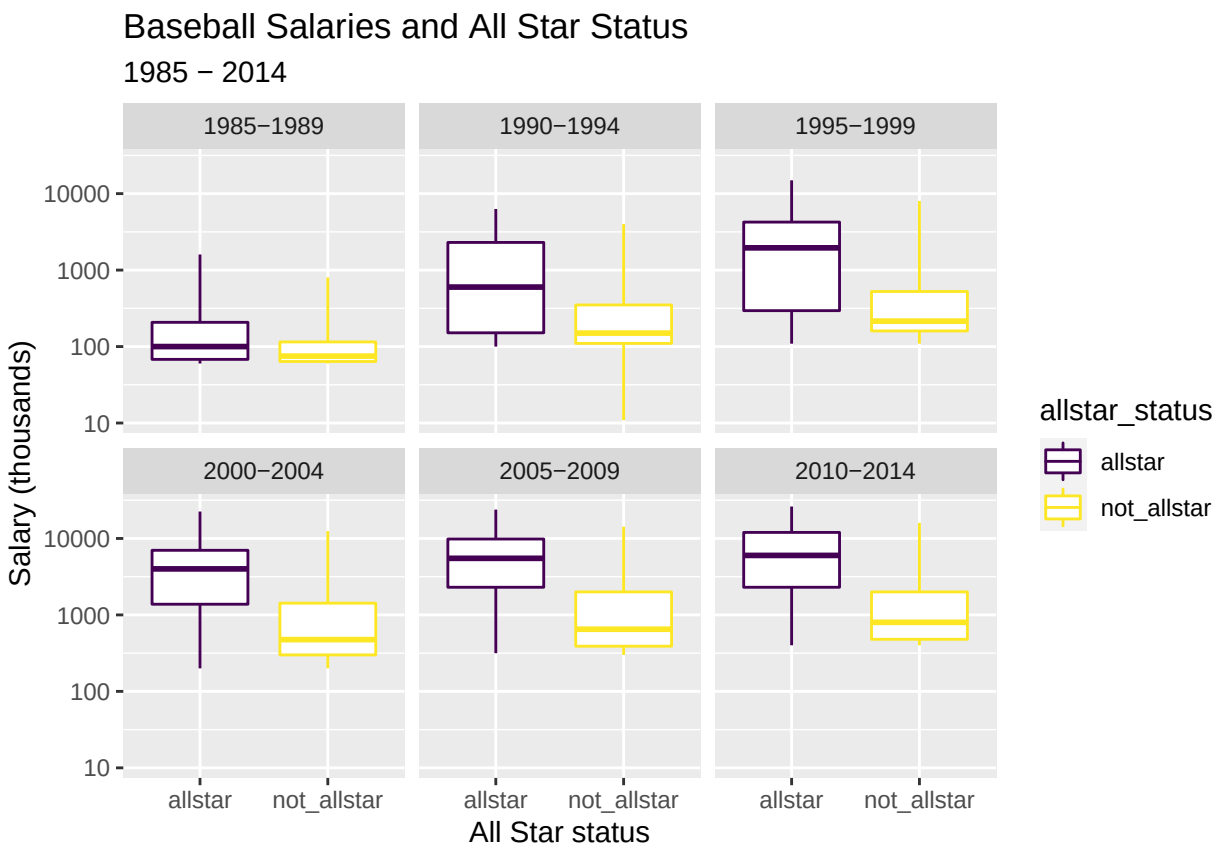
```
                            yearID >= 2005 & yearID <= 2009 ~ labels[5],
                            yearID >= 2010 & yearID <= 2014 ~ labels[6]))

head(df_1)
```

```
## # A tibble: 6 x 7
##   playerID  debut      finalGame  yearID salary allstar_status period
##   <chr>     <date>     <date>      <dbl>  <dbl> <chr>          <chr>
## 1 abadan01  2001-09-10 2006-04-13   2006 327000 not_allstar    2005-2009
## 2 abbotje01 1997-06-10 2001-09-29   1998 175000 not_allstar    1995-1999
## 3 abbotje01 1997-06-10 2001-09-29   1999 255000 not_allstar    1995-1999
## 4 abbotje01 1997-06-10 2001-09-29   2000 255000 not_allstar    2000-2004
## 5 abbotje01 1997-06-10 2001-09-29   2001 300000 not_allstar    2000-2004
## 6 abbotji01 1989-04-08 1999-07-21   1989  68000 not_allstar    1985-1989
```

**(d)** (4 points) Make side-by-side box plots that have `allstar_status` on the x-axis and `salary` (in units of $1000 dollars) on the y-axis. Adjust the y-axis to be on the `log10` scale. Color the box plots according to `allstar_status` and facet over `period` so that there are two rows of side-by-side box plots. Also, set it so the whiskers extend to the minimum and maximum values rather than having the points plotted. (Hint: consider the box plot argument `coef`.) You can modify the existing code below.

```
ggplot(df_1, aes(x=allstar_status, y=salary/1000)) +
  geom_boxplot(aes(color = allstar_status), coef = 5) +
  facet_wrap(~period, nrow=2) +
  xlab("All Star status") +
  ylab("Salary (thousands)") +
  scale_y_continuous(trans='log10') +
  ggtitle("Baseball Salaries and All Star Status",
          subtitle = "1985 - 2014")
```




**(e)** (2 points) Are there any apparent differences between All Star and non-All Star players' salaries? Is it appropriate to compare the salaries between different periods? Explain.

Allstar players tend to have higher salaries and it is appropriate to compare the salaries between different periods since there is a pattern that as period goes, the overall salaries for both non-all star players and all-star players increases.