HW7

2024-10-27

1.

Consider the dataset txhousing which is in the ggplot2 R package. This dataset contains information about 46 Texas cities, recording the number of house sales (sales), the total volume of sales (volume), the average and median sale prices, the number of houses listed for sale (listing) and the number of months inventory (inventory). Data is recorded monthly from Jan 2000 to Apr 2015, 187 entries for each city.

(a)

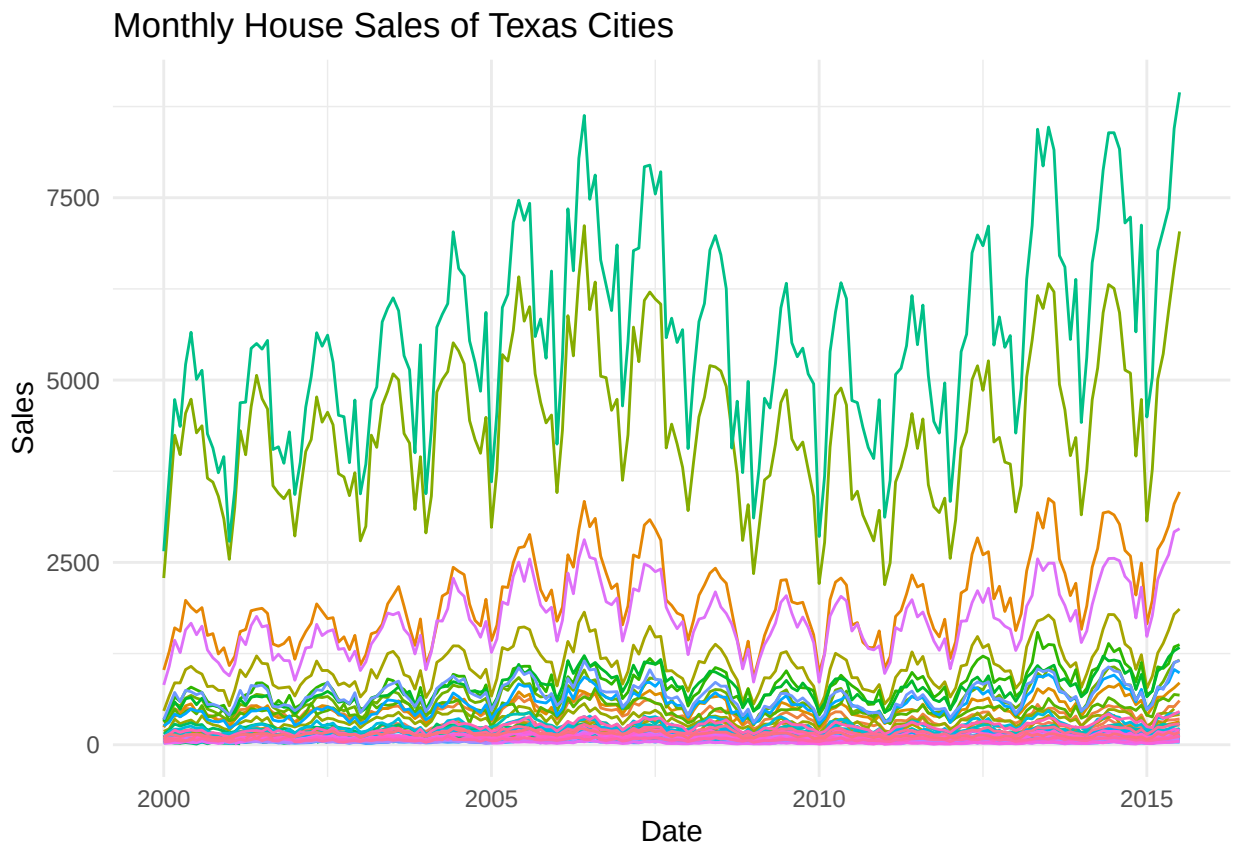
Plot a time series of sales against date for each city on the same plot using ggplot2 package. Describe problems that make it hard to see the long-term trend in this plot.

```
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
```

```
attach(txhousing)

# txhousing <- na.omit(txhousing, cols = c("sales", "city"))

ggplot(txhousing, aes(x = date, y = sales, color = city)) +
  geom_line() +
  labs(title = "Monthly House Sales of Texas Cities",
       x = "Date",
       y = "Sales",
       color = "City") +
  theme_minimal() +
  theme(legend.position = "none")
```

```
## Warning: Removed 430 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
length(unique(txhousing$city))
```

```
## [1] 46
```

```
lowCities <- txhousing |>
  group_by(city) |>
  summarize(avgSales = mean(sales, na.rm = TRUE)) |>
  filter(avgSales < 2500) |>
  nrow()

lowCities
```

```
## [1] 44
```

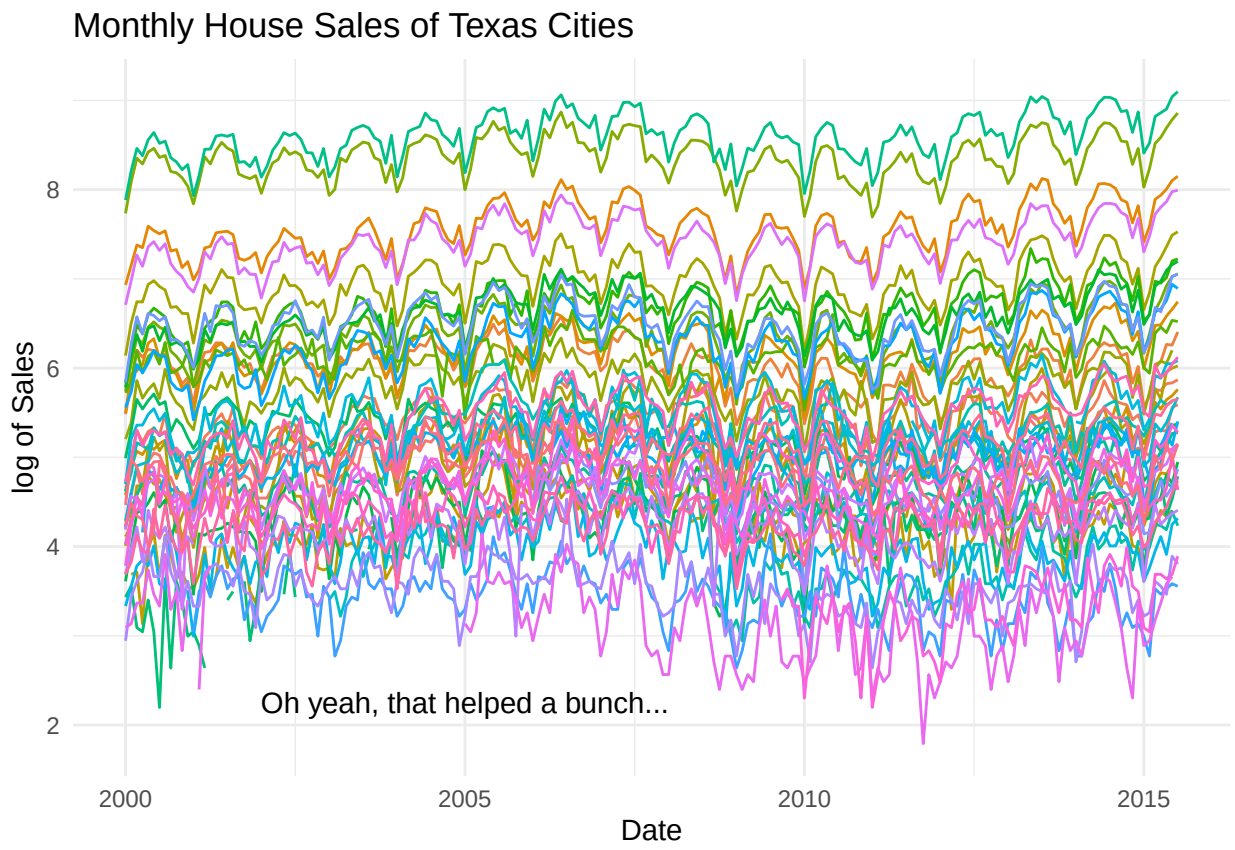
There are a lot of cities in this dataset (46 to be exact)! The fact that so many cluster around 0 to 2500 in "Sales" (44 of 46 average below 2500) means there's a lot of difficulty distinguishing between trends for most of the cities in the above display.

(b)

We can fix one problem described in (a) by plotting sales in the log-scale. Plot a time series of sales in the log-scale against date for each city on the same plot using ggplot2.

```
ggplot(txhousing, aes(x = date, y = log(sales), color = city)) +
  geom_line() +
  labs(title = "Monthly House Sales of Texas Cities",
       x = "Date",
       y = "log of Sales",
       color = "City") +
  theme_minimal() +
  theme(legend.position = "none") +
  annotate("text", x = 2005, y = 2.25, label = "Oh yeah, that helped a bunch...", size = 4, hjust = 0.5)
```

```
## Warning: Removed 430 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

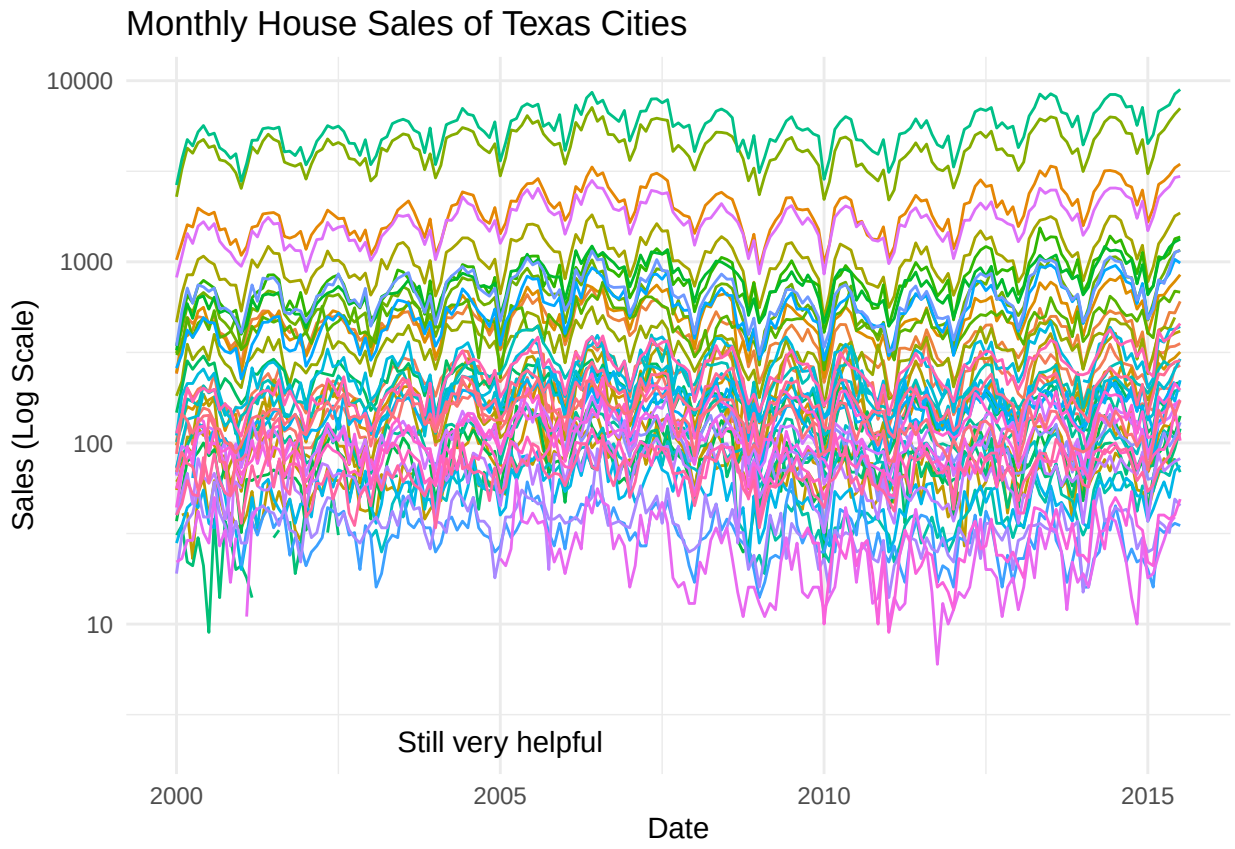


Because of what Prof. Maitra noted about the exam, I also did it without directly converting y into log(Sales). This was done by doing the following:

```
ggplot(txhousing, aes(x = date, y = sales, color = city)) +
  geom_line() +
  scale_y_log10() +
  labs(title = "Monthly House Sales of Texas Cities",
       x = "Date",
       y = "Sales (Log Scale)",
       color = "City") +
  theme_minimal() +
```

```
  theme(legend.position = "none") +
  annotate("text", x = 2005, y = 2.25, label = "Still very helpful", size = 4, hjust = 0.5)
```

```
## Warning: Removed 430 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



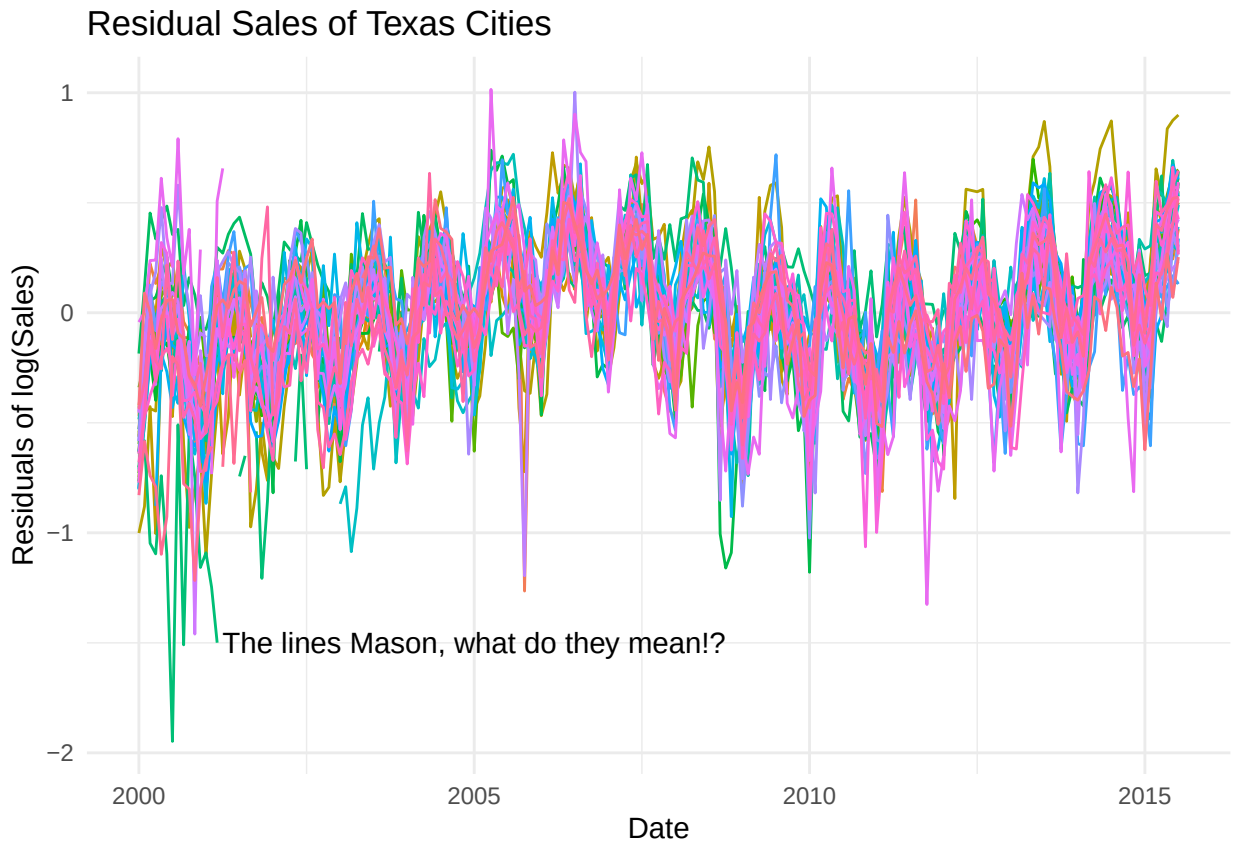
(c)

We can fix another problem described in (a) by removing the strong seasonal trend present in the dataset. Fit a linear model with log(sales) against month for each city, then create a new column in txhousing, call it "rel_sales", which contains the residuals of the linear fits. Then plot a time series of rel sales against date for each city (on the same plot) using ggplot2

```
# use seq_along to keep order to the index used in the udf
# but effectively the index = INDEX = "city"
residualsList <- tapply(X = seq_along(txhousing$sales),
                        INDEX = txhousing$city,
                        FUN = function(index) {
                          lm_fit <- lm(log(txhousing$sales[index]) ~ txhousing$month[index], na.action =
                          residuals(lm_fit)
                          }
                        )
```

```r
txhousing$rel_sales <- unlist(residualsList)

ggplot(txhousing, aes(x = date, y = rel_sales, color = city)) +
  geom_line() +
  labs(title = "Residual Sales of Texas Cities",
       x = "Date",
       y = "Residuals of log(Sales)",
       color = "City") +
  theme_minimal() +
  theme(legend.position = "none") +
  annotate("text", x = 2005, y = -1.5, label = "The lines Mason, what do they mean!?", size = 4, hjust =
```

```
## Warning: Removed 430 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



(d)

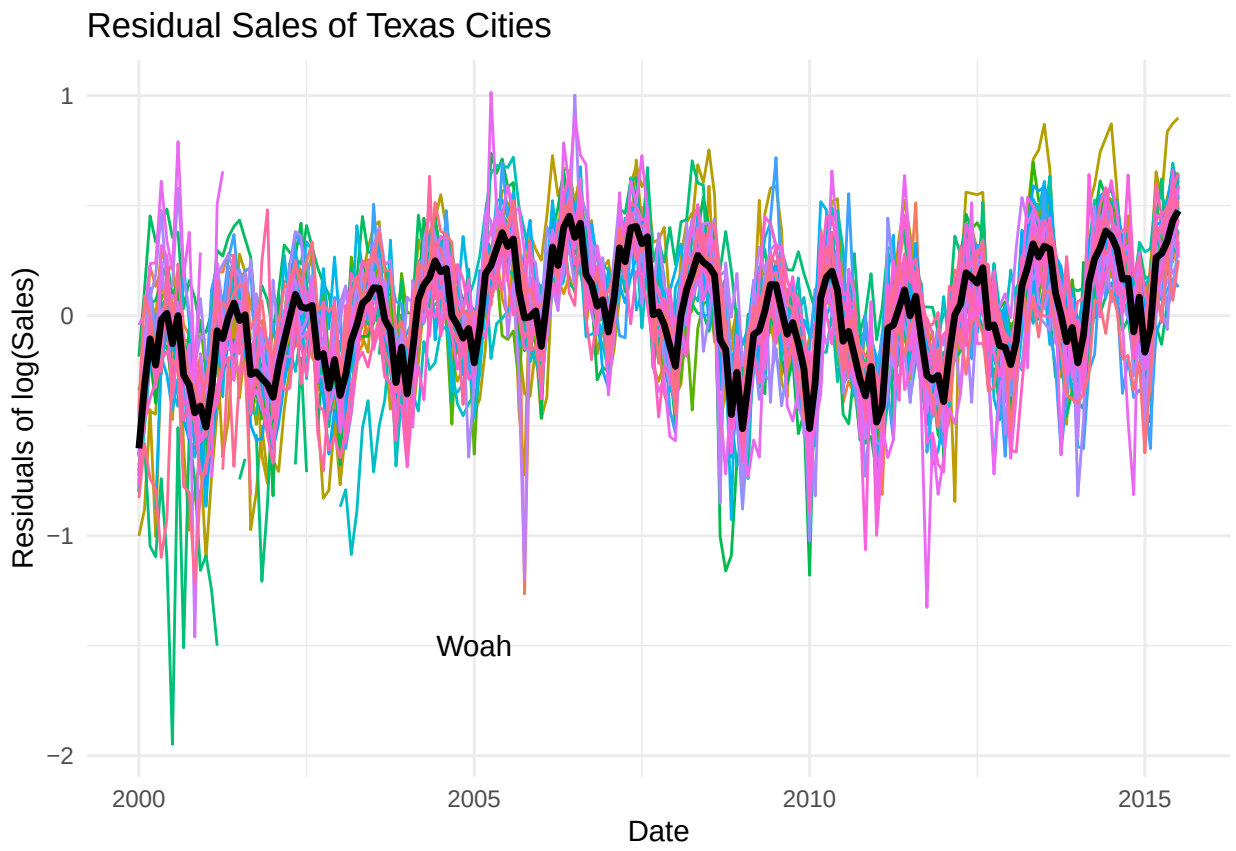
On the plot for (c), put in the mean curve using the function geom line() on the R code in (c).

```r
# Fighting the urge to use DB language and group_by...
# mean_rel_sales <- txhousing |>
#   group_by(date) |>
#   summarise(mean_rel_sales = mean(rel_sales, na.rm = TRUE))
```

```r
# I'll give in to the underscore instead of Camel case...
# since it must be called rel_sales
mean_rel_sales <- data.frame(
  date = unique(txhousing$date),
  mean_rel_sales = tapply(txhousing$rel_sales, txhousing$date, mean, na.rm = TRUE)
)

# Plot the time series of 'rel_sales' for each city and add the mean curve
ggplot(txhousing, aes(x = date, y = rel_sales, color = city)) +
  geom_line() +
  geom_line(data = mean_rel_sales, aes(x = date, y = mean_rel_sales), color = "black", size = 1.2, line
  labs(title = "Residual Sales of Texas Cities",
       x = "Date",
       y = "Residuals of log(Sales)",
       color = "City") +
  theme_minimal() +
  theme(legend.position = "none") +
  annotate("text", x = 2005, y = -1.5, label = "Woah", size = 4, hjust = 0.5)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## Warning: Removed 430 rows containing missing values or values outside the scale range
## (‘geom_line()‘).
```



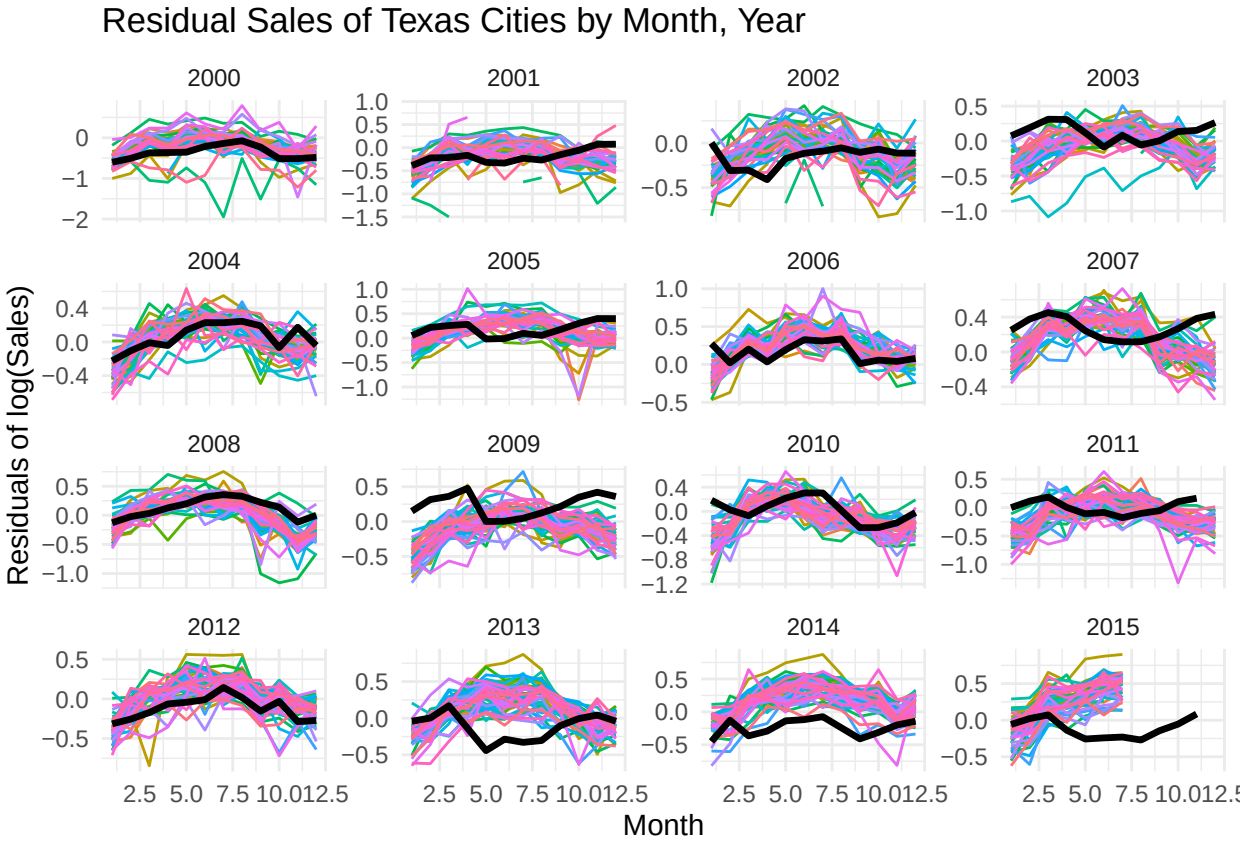
(e)

Provide a page of plots of rel sales against month by year using the facet option on ggplot2 package. On each plot, put in the mean curve using geom line() options.

```
# could also dplyr it, but I don't want to annoy the grading gods
# Oh Gautham, I lament: Have mercy!
# mean_curve <- txhousing |>
#   group_by(year, month) |>
#   summarize(mean_rel_sales = mean(rel_sales, na.rm = TRUE), .groups = "drop")

mean_rel_sales <- data.frame(
  year = rep(unique(txhousing$year), each = 12),
  month = 1:12,
  mean_rel_sales = as.numeric(tapply(X = txhousing$rel_sales,
                                     INDEX = list(txhousing$year, txhousing$month),
                                     FUN = mean,
                                     na.rm = TRUE))
)

mean_rel_sales <- na.omit(mean_rel_sales)

ggplot(txhousing, aes(x = month, y = rel_sales, color = city)) +
  geom_line() +
  geom_line(data = mean_rel_sales, aes(x = month, y = mean_rel_sales),
```

```
            color = "black", size = 1.2, linetype = "solid") +
  labs(title = "Residual Sales of Texas Cities by Month, Year",
       x = "Month",
       y = "Residuals of log(Sales)",
       color = "City") +
  facet_wrap(~ year, scales = "free_y") +
  theme_minimal() +
  theme(legend.position = "none")
```

```
## Warning: Removed 430 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



Note: The above plot had breaks in the fit line until I added the following code `mean_rel_sales <- na.omit(mean_rel_sales)`

2.

When a cell phone relay tower is not working properly, wireless providers can lose great deal of money so it is important for them to be aware of and to be able to fix problems expeditiously. A first step toward understanding the problems involved is to collect data from designed experiments involving three factors, as was done in the experiment that gave rise to the dataset available in the file breakdown.dat. In this case, a reported problem was initially classified as low or high severity, simple or complex, and the engineer assigned was rated as relatively new (novice) or expert(guru). Two times were observed. The dataset has measurements on severity level of the problem (low/high, column 1), level of complexity of the problem (simple/complex, column 2), experience level of engineer (novice/guru, column 3), and time to assess problem (column 4), implement solution (column 5) and total resolution (column 6).

(a)

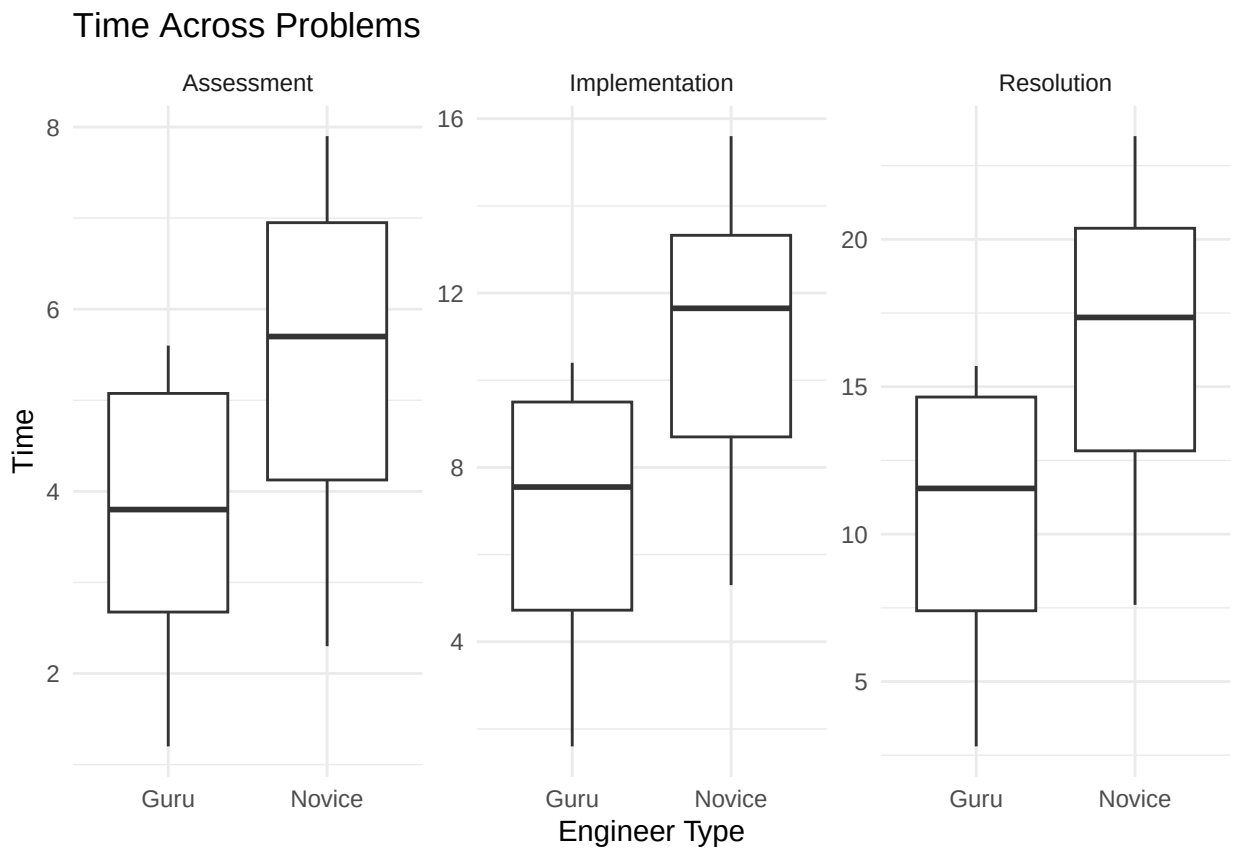
Read the data set and name each column, the variables corresponding names, problem, severity and engineer, i.e., problem for low and high, severity for simple and complex and engineer for new or expert. The names for the last three columns are given by "Assessment","Implementation","Resolution". Note that the resolution is the sum of the time assessment of the problem and the implementation.

```
breakdown <- read.table("breakdown.dat")
names(breakdown) <- c("Problem", "Severity", "Engineer", "Assessment","Implementation","Resolution")
```

(b)

For the engineer type as a categorical variable on the x axis, create a facet with boxplots for the variables. You may need to use the reshape2 package. Also rename the y-axis label. What difference can you tell for the engineer for each time?

```
library(reshape2)
library(ggplot2)

breakdown_long <- melt(data = breakdown,
                       id.vars = c("Problem", "Severity", "Engineer"),
                       measure.vars = c("Assessment", "Implementation", "Resolution"),
                       variable.name = "Thing",
                       value.name = "Time")

ggplot(breakdown_long, aes(x = Engineer, y = Time)) +
  geom_boxplot() +
  facet_wrap(~ Thing, scales = "free_y") +
  labs(y = "Time", x = "Engineer Type", title = "Time Across Problems") +
  theme_minimal()
```

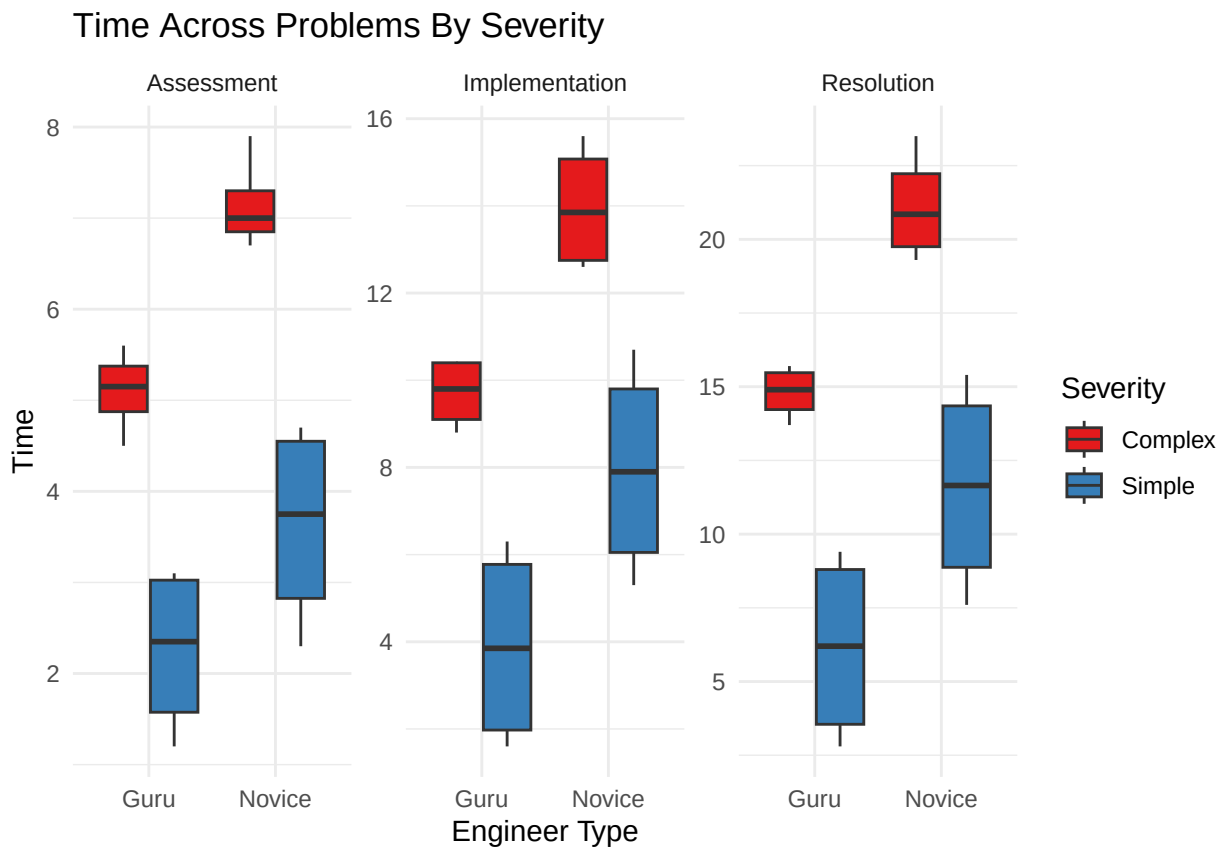


From the above plot, we see that on average the "Guru" tends to take less time than the "Novice" across the three types of problems in our dataset.

(c)

The above boxplot may be hard to interpret for some. For the above, separate the levels severity and create side-by-side boxplots for each level of engineer. What conclusions can you draw from each of these plots?

```
ggplot(breakdown_long, aes(x = Engineer, y = Time, fill = Severity)) +
  geom_boxplot(position = position_dodge(width = 0.8)) +
  facet_wrap(~ Thing, scales = "free_y") +
  labs(y = "Time", x = "Engineer Type", title = "Time Across Problems By Severity") +
  scale_fill_brewer(palette = "Set1", name = "Severity") +
  theme_minimal()
```

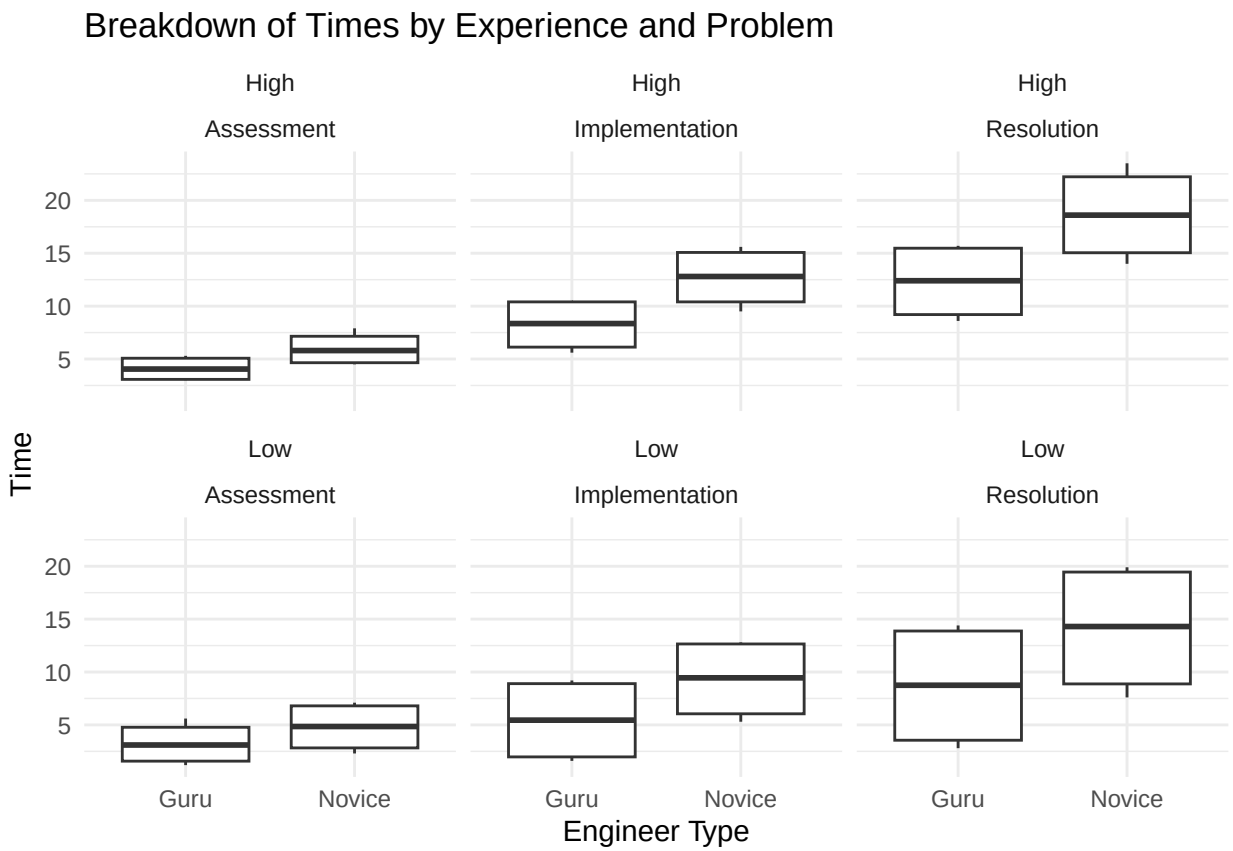


We see that "Complex" problems (regardless of which type of Problem) tend to take more time than "Simple" problems. Furthermore, we notice the above trend from part (b) of Guru's being more efficient with their time than Novices, still holds across the severity types. What's more, the difference appears more pronounced in the "Implementation" stage of the problem, though this generally also holds and is noticeable when looking at "Complex" problems by stage of the problem and looking at the lack of overlap between Novice and Guru.

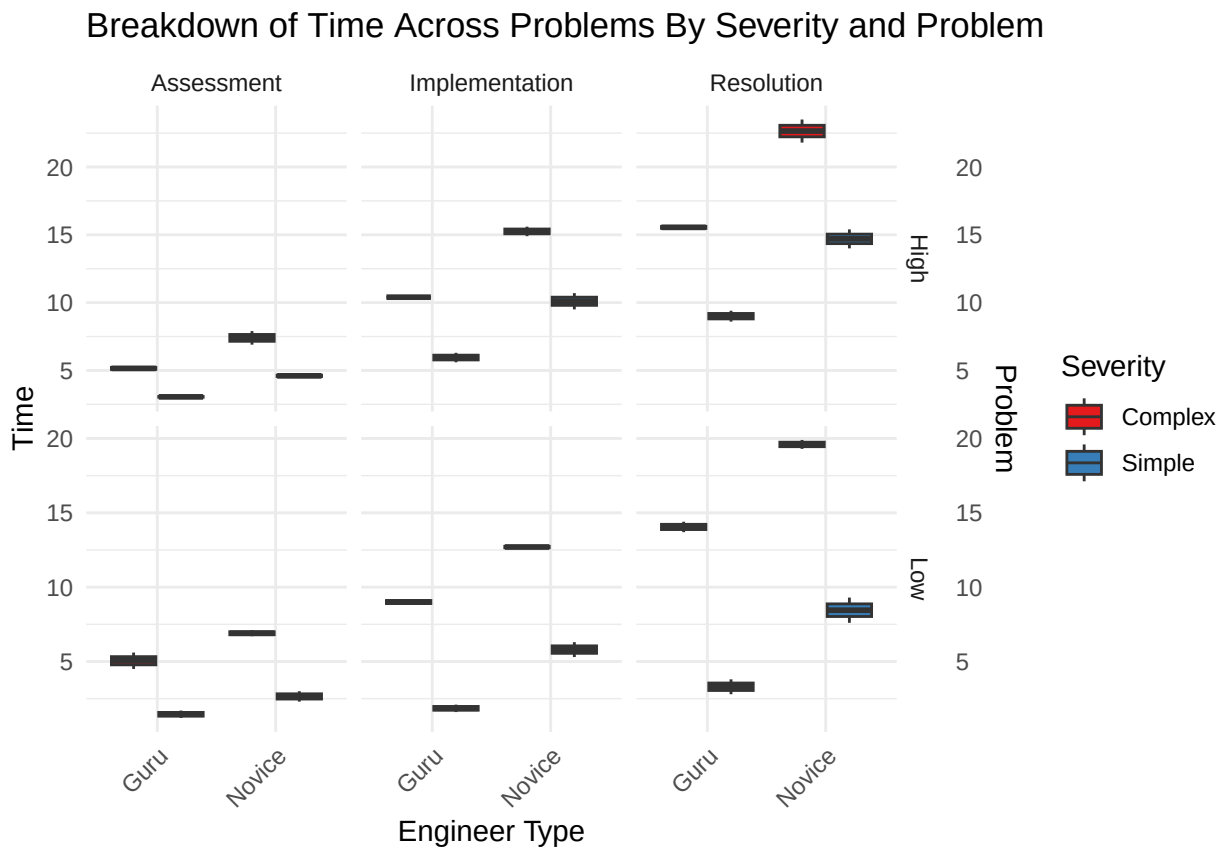
(d)

Now create a new faceting plot, by faceting with "Problem" and the three variables. Discuss the plot and draw appropriate conclusions, if any.

```
ggplot(breakdown_long, aes(x = Engineer, y = Time)) +
  geom_boxplot() +
  facet_wrap(Problem ~ Thing, scales = "fixed") +
  labs(title = "Breakdown of Times by Experience and Problem",
       x = "Engineer Type",
       y = "Time"
       ) +
  theme_minimal()
```




```r
ggplot(breakdown_long, aes(x = Engineer, y = Time, fill = Severity)) +
  geom_boxplot(position = position_dodge(width = 0.8)) +
  facet_grid(Problem ~ Thing, scales = "free_y") +
  scale_y_continuous(
    name = "Time",
    sec.axis = sec_axis(~ ., name = "Problem",)
  ) +
  labs(x = "Engineer Type", title = "Breakdown of Time Across Problems By Severity and Problem") +
  scale_fill_brewer(palette = "Set1", name = "Severity") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Hello again, Gautham. For this problem, I felt there was a bit of ambiguity as to whether this part also needed the breakdown (fill) by Severity, i.e. what "and the three variables" was referring to (stages of a problem each initially as their own column or for Engineer Type, Time, and Severity). So I included both plots. The overall conclusions remain the same though, just the added complication of Severity in the assessment. That being said:

In general, I find the above difficult to see (so tiny!) That point notwithstanding there tend to be three comparisons I find myself making: Guru vs. Novice comparisons, "High" vs. "Low" Severity comparisons, and "Complex" vs. "Simple" comparisons, all across the (secret fourth comparison) of Assessment, Implementation, and Resolution. As Resolution is a composite of Assessment and Implementation, results tended to be most consistent with observations for Assessment or Implementation individually.

Guru v. Novice: The Guru tends to have lower variability (greater precision) in time taken across all stages, and tends to take less time than the Novice regardless of task or complexity. By contrast the "Novice" shows more variability (less precision) particularly during the Implementation and Resolution. Though a bit difficult to see, Novice has significant outliers and longer times for resolving "Complex" problems, particularly in Implementation.

Complex v. Simple: For "Complex" problems, the Resolution stage takes longer for the Novice compared to the Guru (largest disparity, visually, from the two plots), which seems in large part due to differences in Implementation (this point isn't especially poignant since Resolution is a composite of the two components).

High v. Low: Assessment times do not vary much between "High" and "Low" severity (top vs. bottom graphs) though there are slight differences between Guru and Novice for Assessments.

Let's get down to Business: I need Gurus, more Gurus. But short of that, maybe there is something to be said about Novices difficulty in Implementation (since Resolution is a composite of Implementation and Assessment).

Figure 1: Spiderman