Homework 8

Steve Harms

October 29, 2017

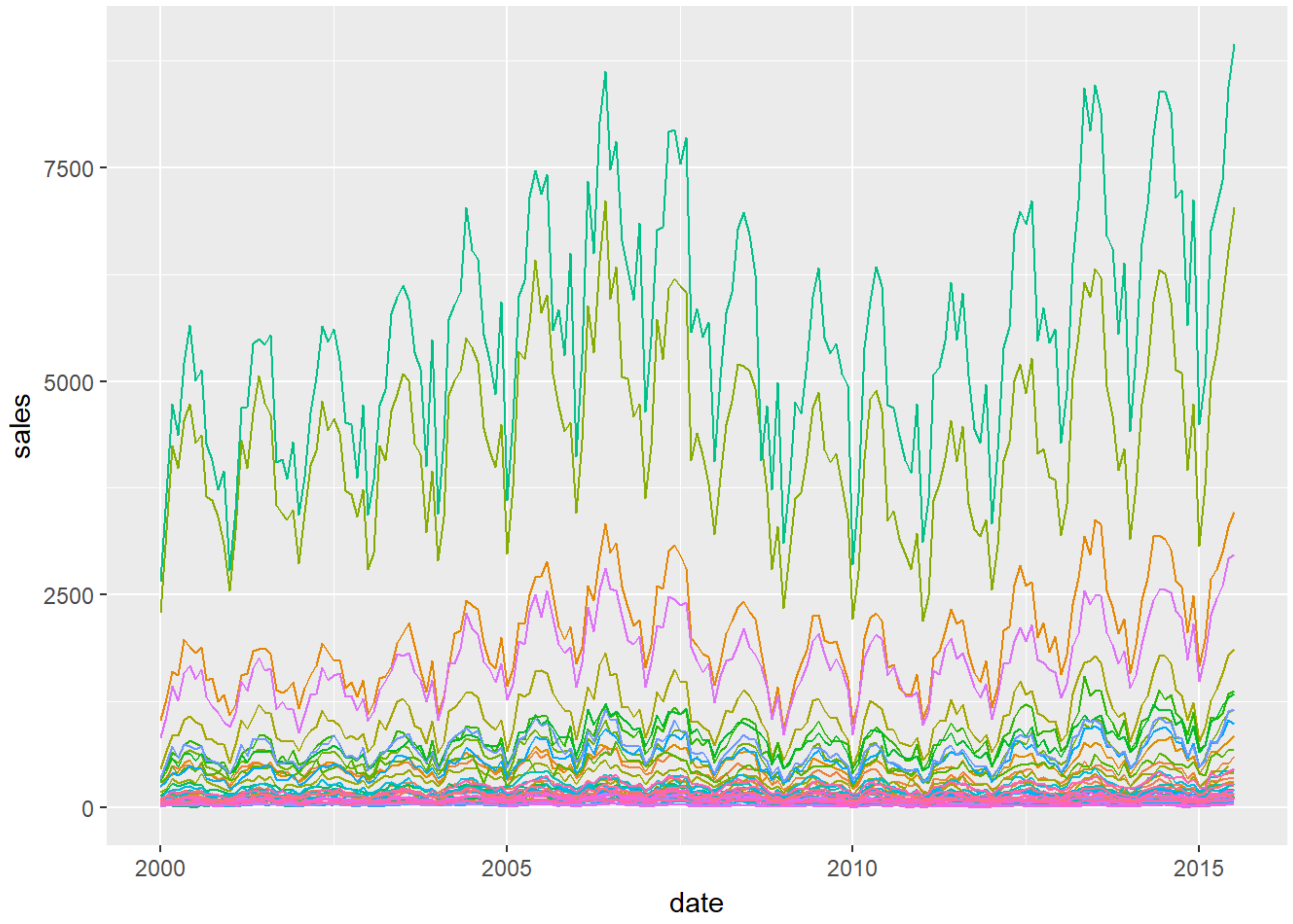
Exercise 1

a)

```
housing <- txhousing

#First, a plot with a key for cities by color. I remove the legend because it's use
less with so many cities
a <- ggplot(data = housing, aes(x=date, y = sales, col = city))
a + geom_line() + theme(legend.position = "none")
```

```
## Warning: Removed 430 rows containing missing values (geom_path).
```

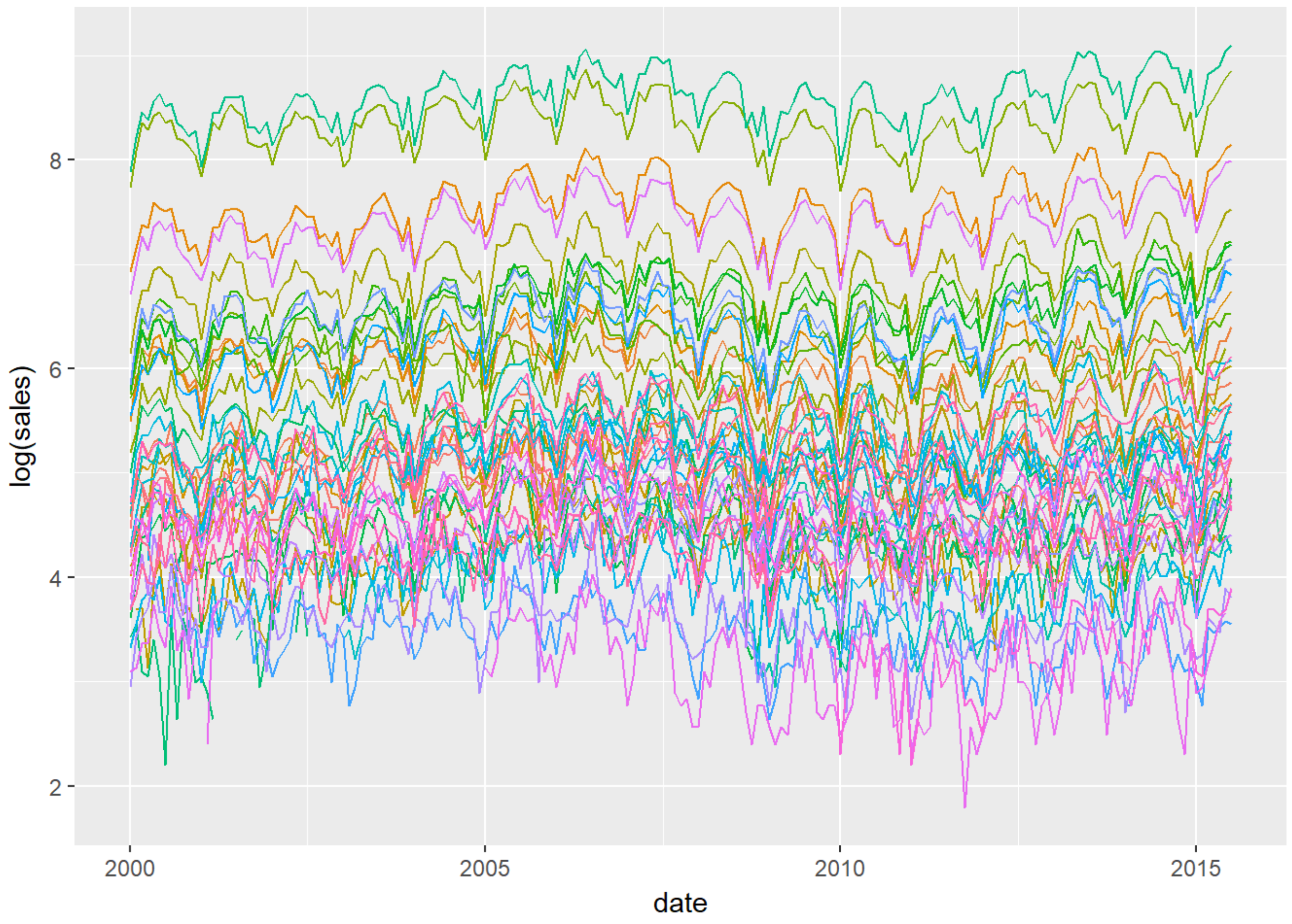


Some of the larger cities had far more sales than the smaller cities, so the scale of the plot makes it difficult to see anything but the large cities. In addition, there appears to be a large seasonal effect of the sales that makes it difficult to examine the underlying trend.

b)

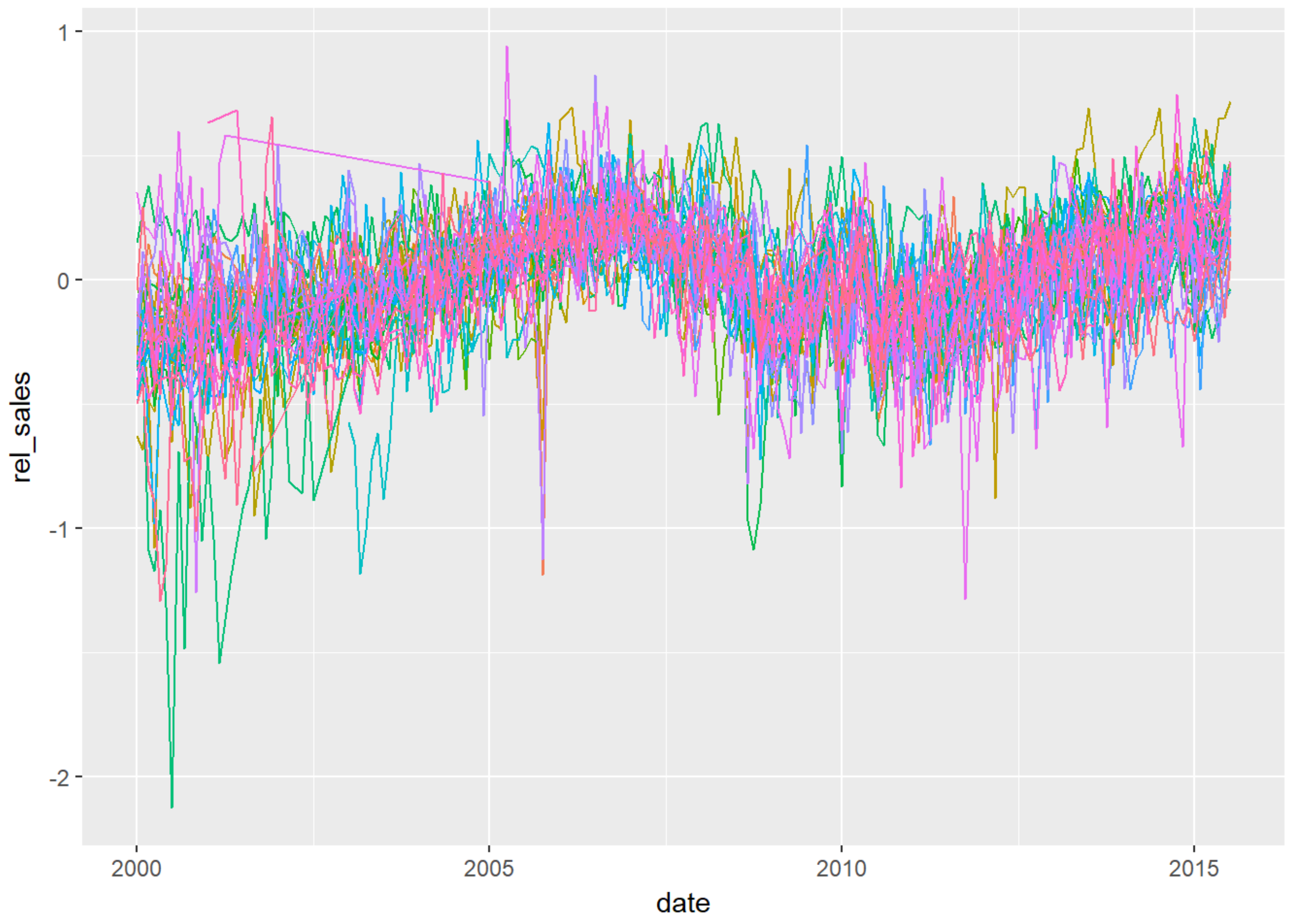
```r
#now a plot using log transformed sales
b <- ggplot(data = housing, aes(x=date, y = log(sales)))
b + geom_line(aes(col = city)) + theme(legend.position = "none")
```

```
## Warning: Removed 430 rows containing missing values (geom_path).
```



c)

```r
#Remove the na values from the housing set
housing <- filter(housing,!is.na(sales))
#fit the model using log(sales), with date and factor city as explanatory
model <- lm(log(sales)~factor(month) + city, data = housing)
#Save the residuals
housing$rel_sales <- model$residuals
#Look at the new plot(s)
g  <- ggplot(data = housing, aes(x=date))
h  <-g + geom_line(aes(y= rel_sales, col = city))
h + theme(legend.position = "none")
```

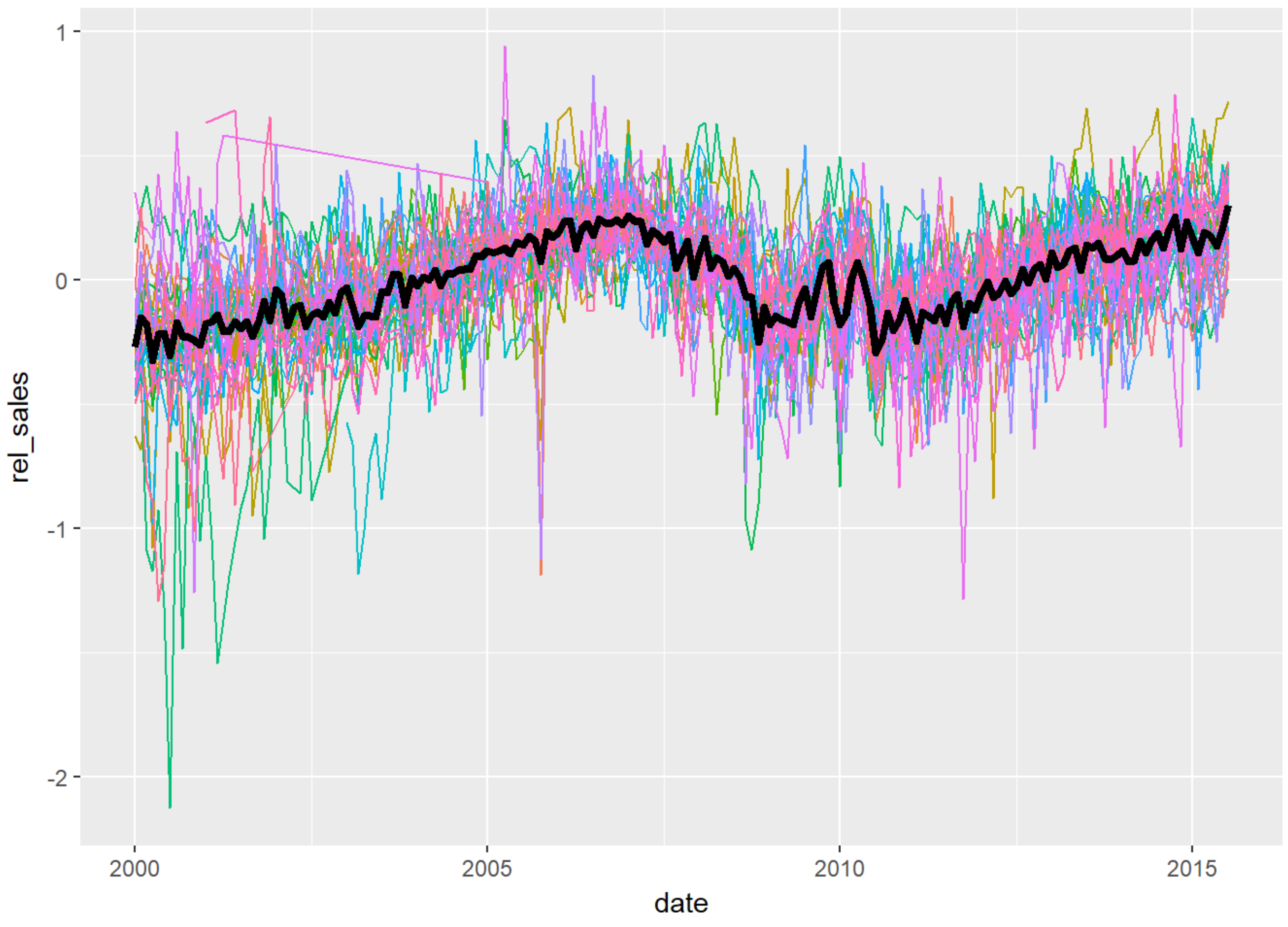


d)

```r
#Get our monthly average residuals
dateavg <- housing %>% group_by(date) %>% summarize(mean(rel_sales))
names(dateavg) <- c("dates", "averagesales")

#Create columns of months and dates for easier plotting later
dateavg$year <- floor(dateavg$dates)
dateavg$month <- round(12*(dateavg$dates-dateavg$year), 0) + 1

#Add it to our previous plot with a different color so we know which one it is
i <- h + geom_line(data = dateavg, aes(x = dates, y = averagesales),col = "black",
size = 1.5)
i + theme(legend.position = "none")
```



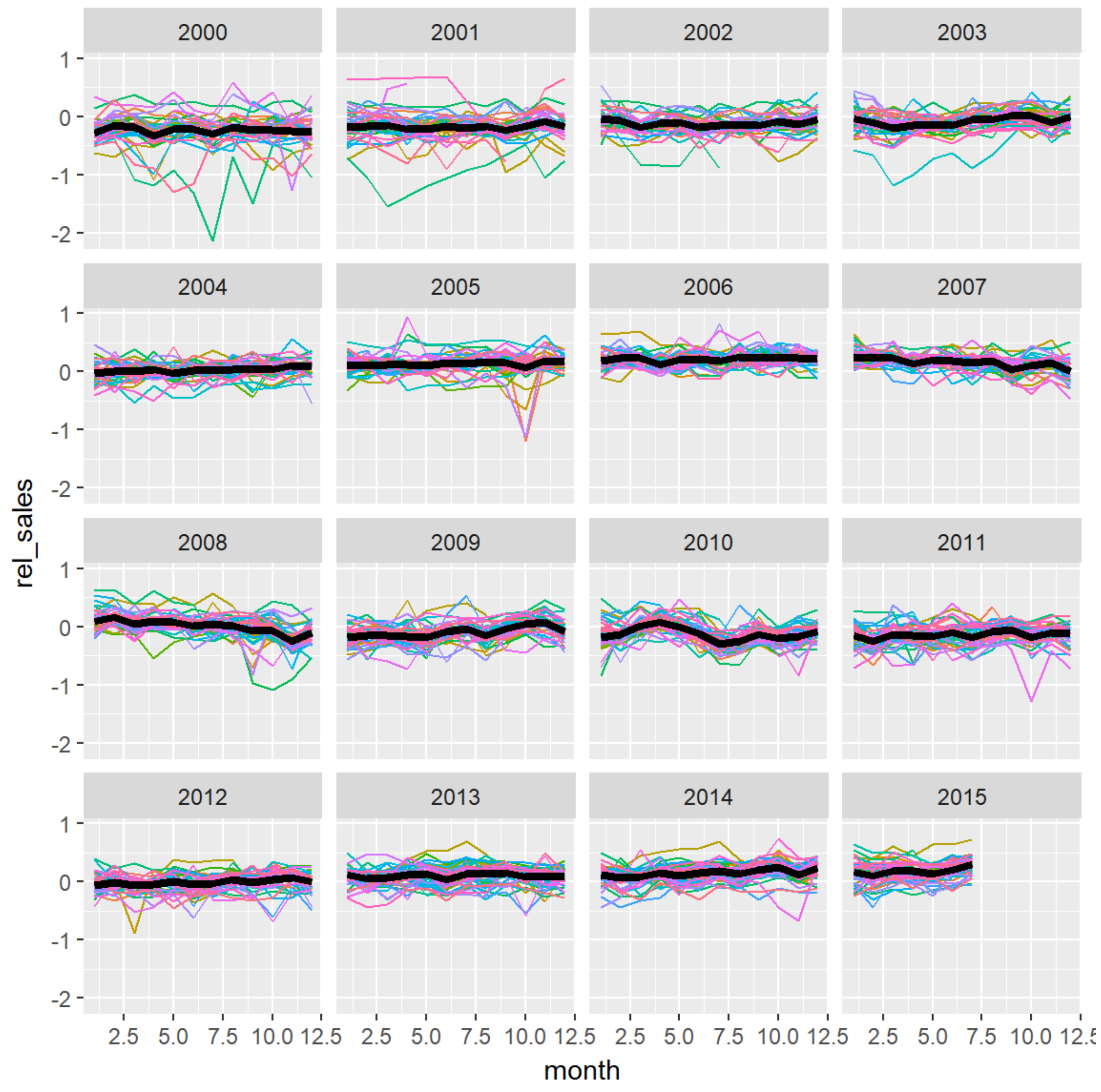
e)

```r
#Create a set of plots, one for each year
j <- ggplot(data = housing, aes(x = month, y = rel_sales))

#Make sure to add a black line with the mean values
k <- j + geom_line(aes(col = city)) + geom_line(data = dateavg, aes(x = month, y =
averagesales), col = "black", size = 1.5)

#legend isn't really useful and just takes up space with so many different cities o
n the same plote
m <- k + theme(legend.position = "none")

#Facet by year
l <- m +facet_wrap(~year, nrow = 4, ncol = 4)
l
```



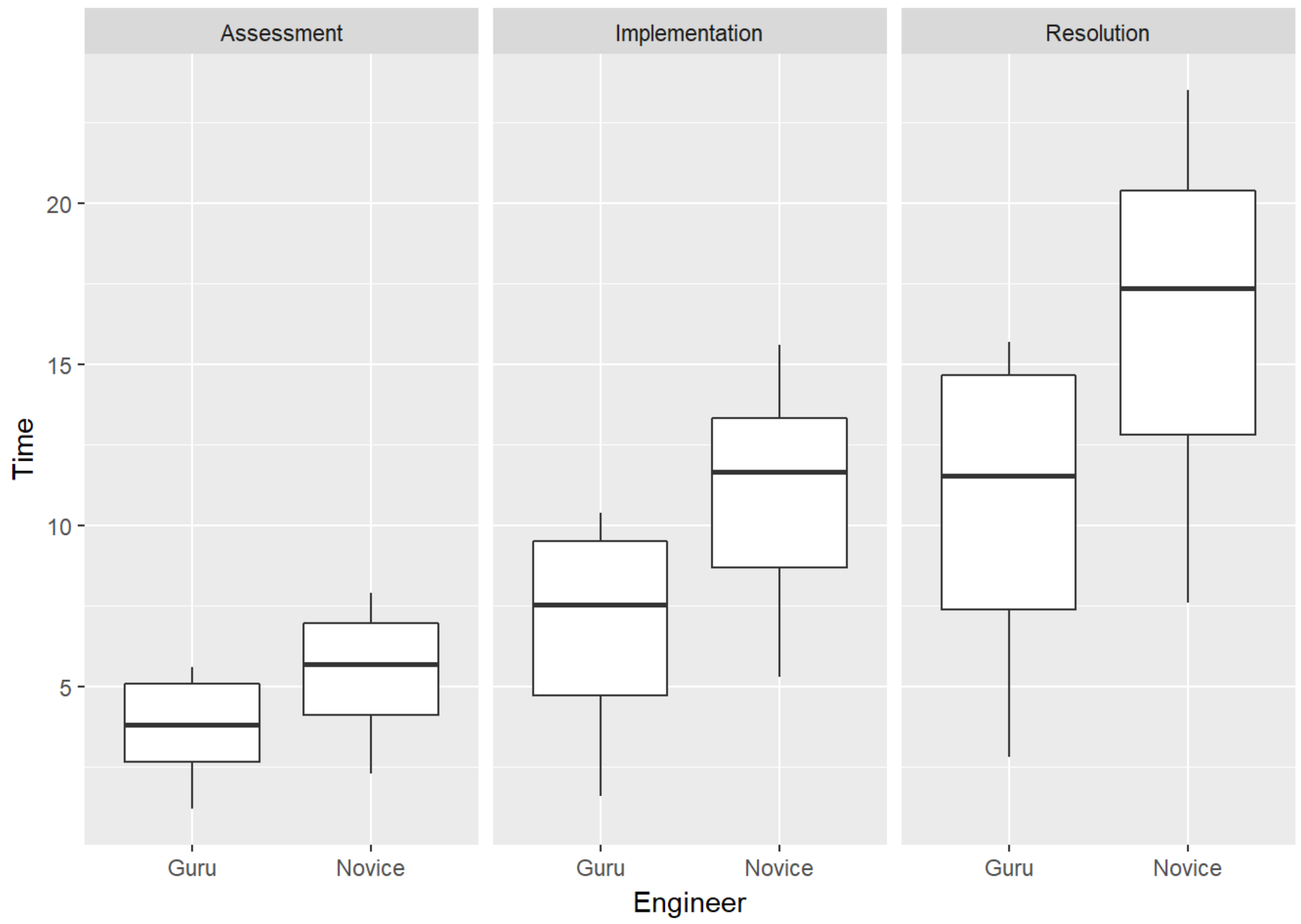
Exercise 2

a)

```
breakdown <- read.table(file = "http://maitra.public.iastate.edu/stat501/datasets/b
reakdown.dat", header = F)
names(breakdown) <- c("Severity", "Problem", "Engineer", "Assessment", "Implementa
tion","Resolution")
```

b)

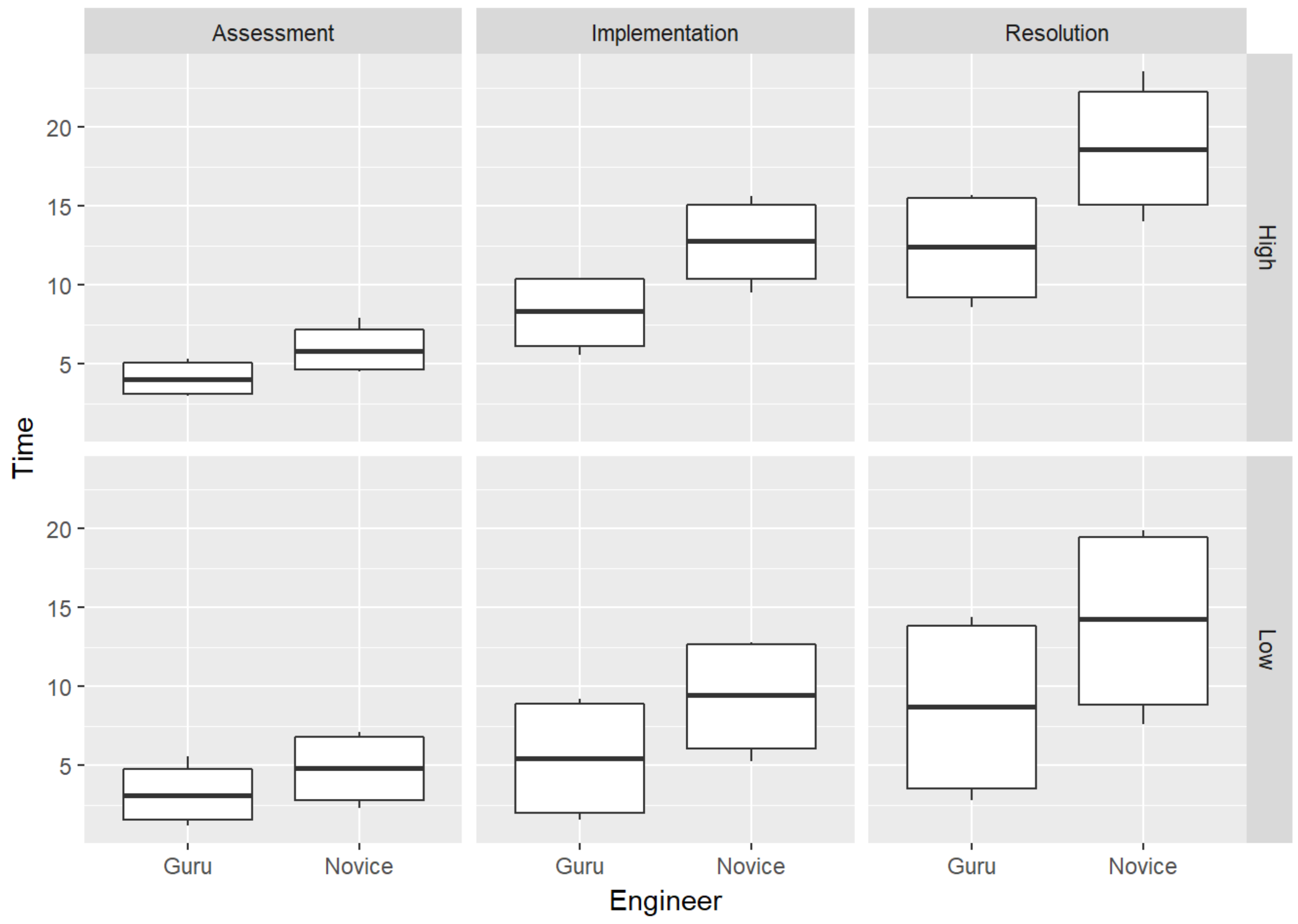
```
#melt the variables into one column for easy faceting
breakmelt <- melt(breakdown, id.vars = c("Severity", "Engineer", "Problem"))
#plot the 3 variables side-by-side for each engineer type
bb <- ggplot(data = breakmelt, aes(x = Engineer, y = value))
bbb <- bb + geom_boxplot() + facet_grid(.~variable) + ylab("Time")
bbb
```



For all three times, the guru is (on average) faster than the novice. Also, there is a wider variance for resoultion time for both engineer types, indicating that some problems might be easy to fix while some are hard.

c)

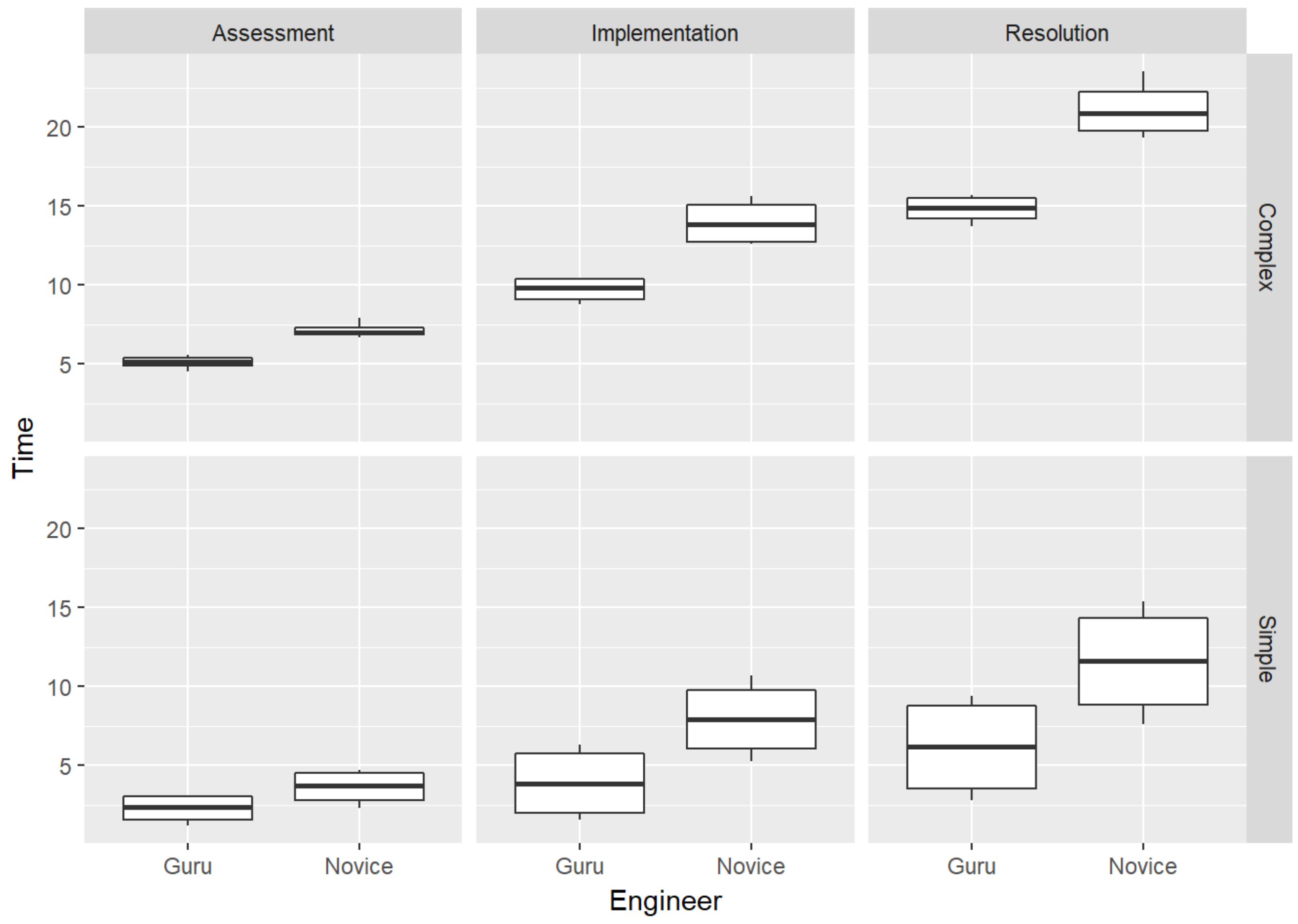
```
#now add another facet level for severity
cc <- bbb + facet_grid(Severity ~ variable)
cc
```



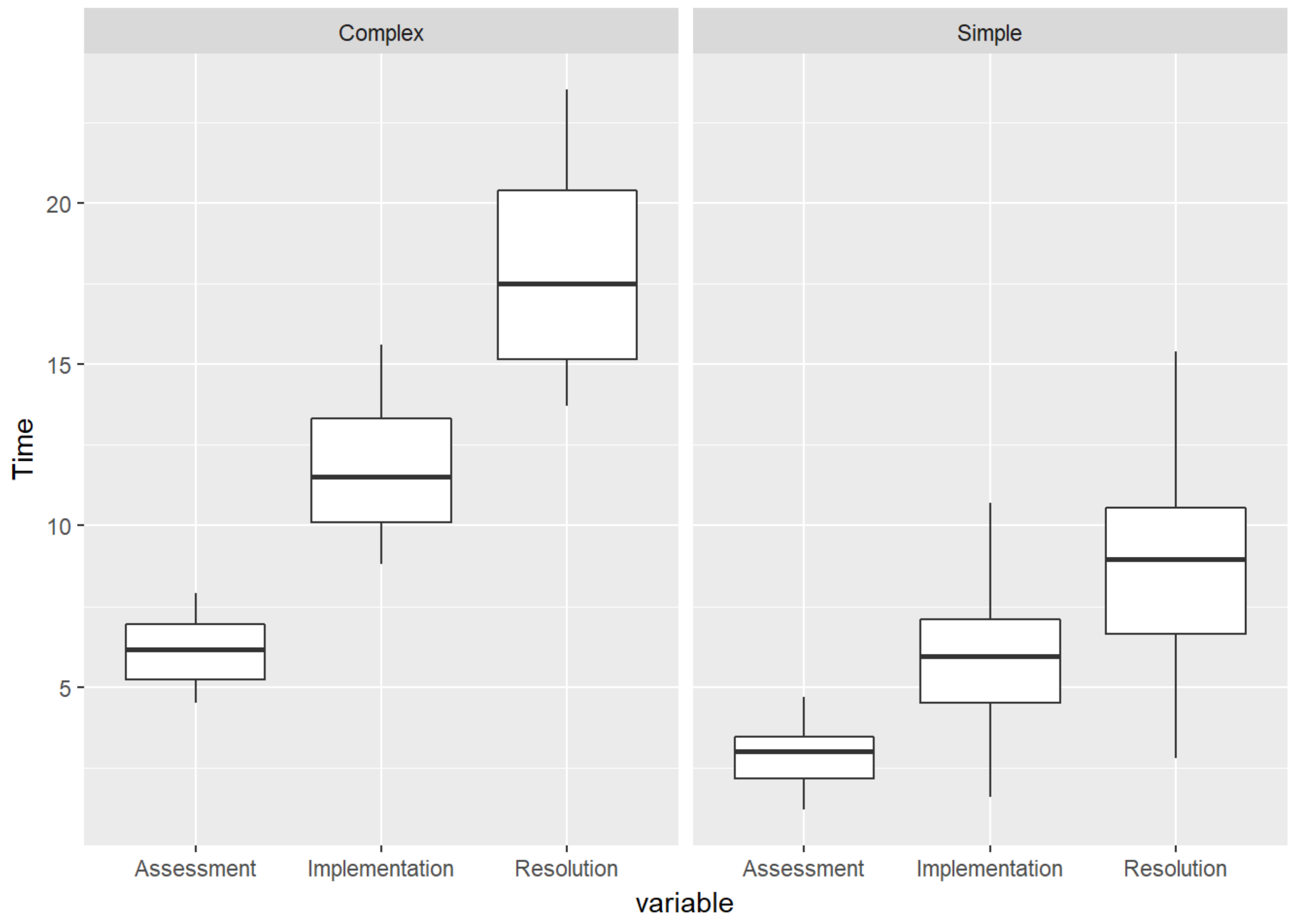
From the new plots, we can see that there is very little overlap between the “guru” and “novice” engineers. In almost every case, the slowest “guru” engineer will still be faster than most of the faster “novice” engineers for any problem, at any time. It appears that occaisonally the gurus are a bit slower on implementation, but are almost always faster on assessment. Variances are much smaller when we separate by severity, as well.

d)

```
#a plot with a facet for each problem type, each variable, with engineer type side-
by-side within
dd <- bb + geom_boxplot() + facet_grid(Problem ~ variable) + ylab("Time")
dd
```



```
#alternatively, a plot with a facet for each problem type, with variables side-by-s
ide within the facets
dddd <- ggplot(data = breakmelt, aes(x = variable, y = value)) + geom_boxplot() +
facet_grid(.~Problem) + ylab("Time")
dddd
```



It seems pretty obvious from the plot that more complex problems will always take more time both for assessment and implementation of the solution, leading to higher overall resolution time. However, note that some of the resolution times for simple problems are actually quite large, so some simple problems may not be so simple after all. Note that this conclusion could be convoluted due to the differences in engineer type for each problem.