EDA Techniques

Group 4

2025-05-17

7.1 - Introduction

Paul

Note:

The sections (Introduction, Questions, ggplot calls, and Learning more) are narrative in nature. We include their summaries here for completeness, to retain a simple reference, and for the equal division of group work.

A note on Exploratory Data Analysis (EDA): EDA is a state of mind, feel free to explore all of the ideas that come to you whether they pan out or not. Scrubbing is just one important part, ensuring the data is in the right form for your analysis–using visualization, transformation, and modeling.

EDA workflow: question %>% visualize, transform, model %>% refine your questions and generate new ones.

We'll be using dplyr and ggplot2 in the tidyverse.

7.2 - Questions

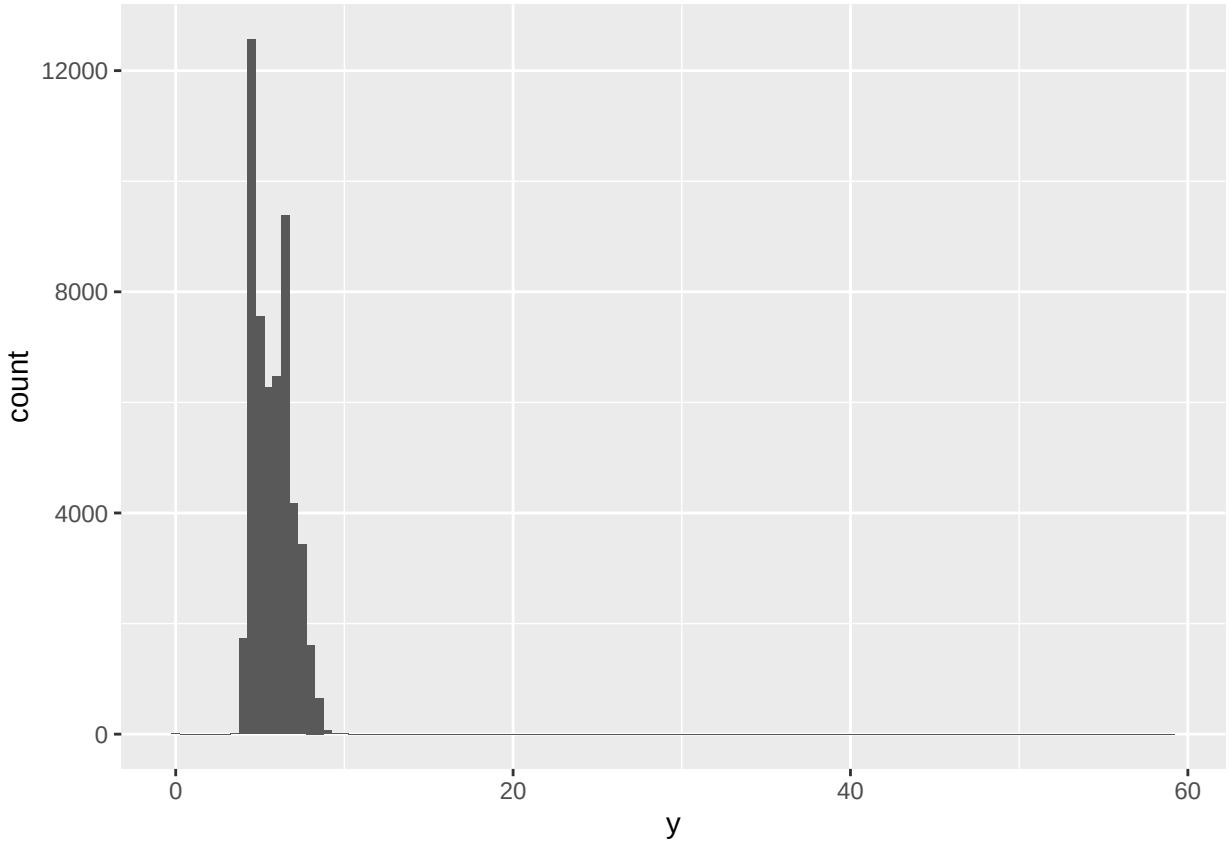
Paul

Creatively understand your data. Generate many questions and select the right one. In Lean Six Sigma, we call this process, the 5 W's. Continue to ask an additional question based on the answer to your last question, until you reach the 5th, at which point you can begin to make relevant discoveries. Here the questions are about variation, and covariation. (Know these terms: variable, value, observation, Tabular data, tidy tabular is when cell = value, column = variable, row = observation).

7.3 - Variation

Matthew

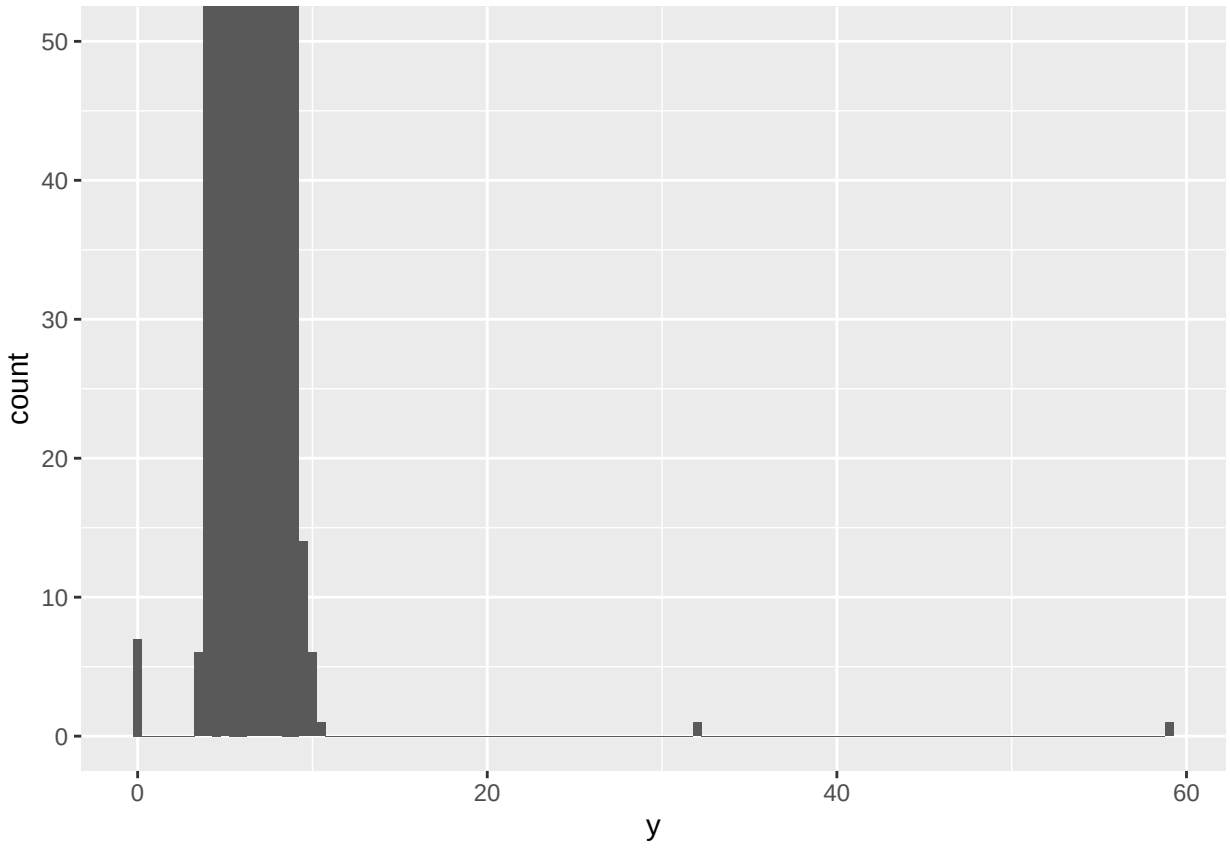
```
ggplot(diamonds, aes(x = y)) +
  geom_histogram(binwidth = 0.5)
```



While the graph as is shows the general distribution and trends as a whole, but the far right tail values are essentially illegible due to their small value overall. Adding in cood_cartesian() funcion allows you to zoom in on this specific difficult to see range.

It's also very useful when you want to see a specific range without distorting the overall data and keeping the axis scaling consistent.

```
ggplot(diamonds, aes(x = y)) +
  geom_histogram(binwidth = 0.5) +
  coord_cartesian(ylim = c(0, 50))
```



Here you can see it is used to zoom in on the bars that have less than 50, without removing the other bars. Ideally you will want to use this to view a specific subset of data among a larger dataset that is difficult to visualize within the base graphic

7.4 - Missing Values

Miles

Frequently you will encounter wonky or erroneous values in your dataset. There are two things you can do to handle them. The first is to drop the whole row of data:

```
diamonds2 <- diamonds %>%
  filter(between(y, 3, 20))
```

This usually isn't recommended, unless there is reason to believe one invalid measurement is evidence of other invalid measurements.

The other option is to replace the individual problem values with NAs. This is most easily done using mutate() and ifelse():

```
diamonds2 <- diamonds %>%
  mutate(y = ifelse(y < 3 | y > 20, NA, y))
```

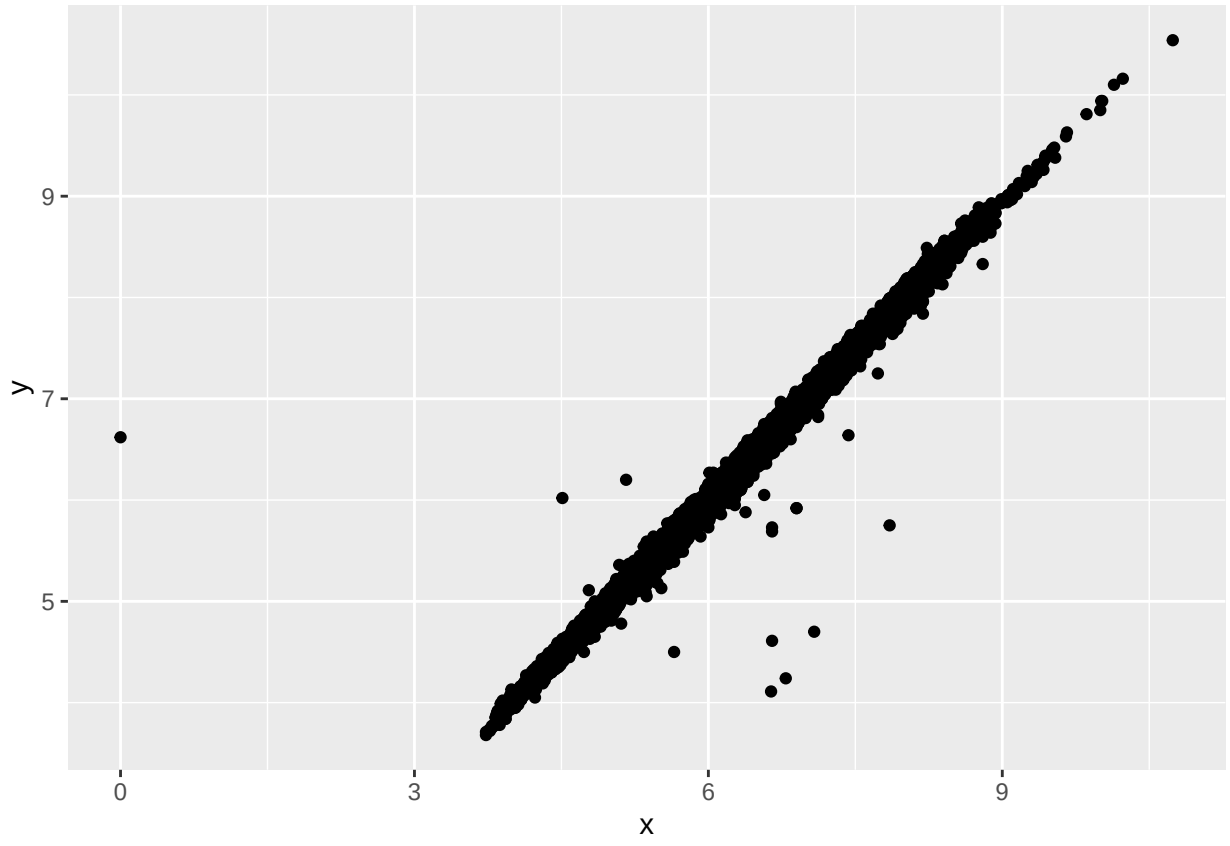
The three arguments of ifelse() are 1. the logical condition, 2. the value when the condition is true, and 3. the value when the condition is false. For more complex combinations of existing variables, case_when() is a better option. For example, I used it to create bins of years when I was plotting the baseball data in Assignments 1 and 2:

```r
if (!requireNamespace("Lahman", quietly = TRUE)) {
  install.packages("Lahman")
}

library(Lahman)

# Binning years
battingYearBin <- Batting %>%
  mutate(year_bin = case_when(
    yearID >= 1986 & yearID < 1991 ~ "86-90",
    yearID >= 1991 & yearID < 1996 ~ "91-95",
    yearID >= 1996 & yearID < 2001 ~ "96-00",
    yearID >= 2001 & yearID < 2006 ~ "01-05",
    yearID >= 2006 & yearID < 2011 ~ "06-10",
    yearID >= 2011 & yearID < 2016 ~ "11-15"
  )) %>%
  filter(!is.na(year_bin))
```

ggplot2 doesn't include missing values in plots, but will give a warning that they were excluded. This warning can be suppressed with na.rm = TRUE:

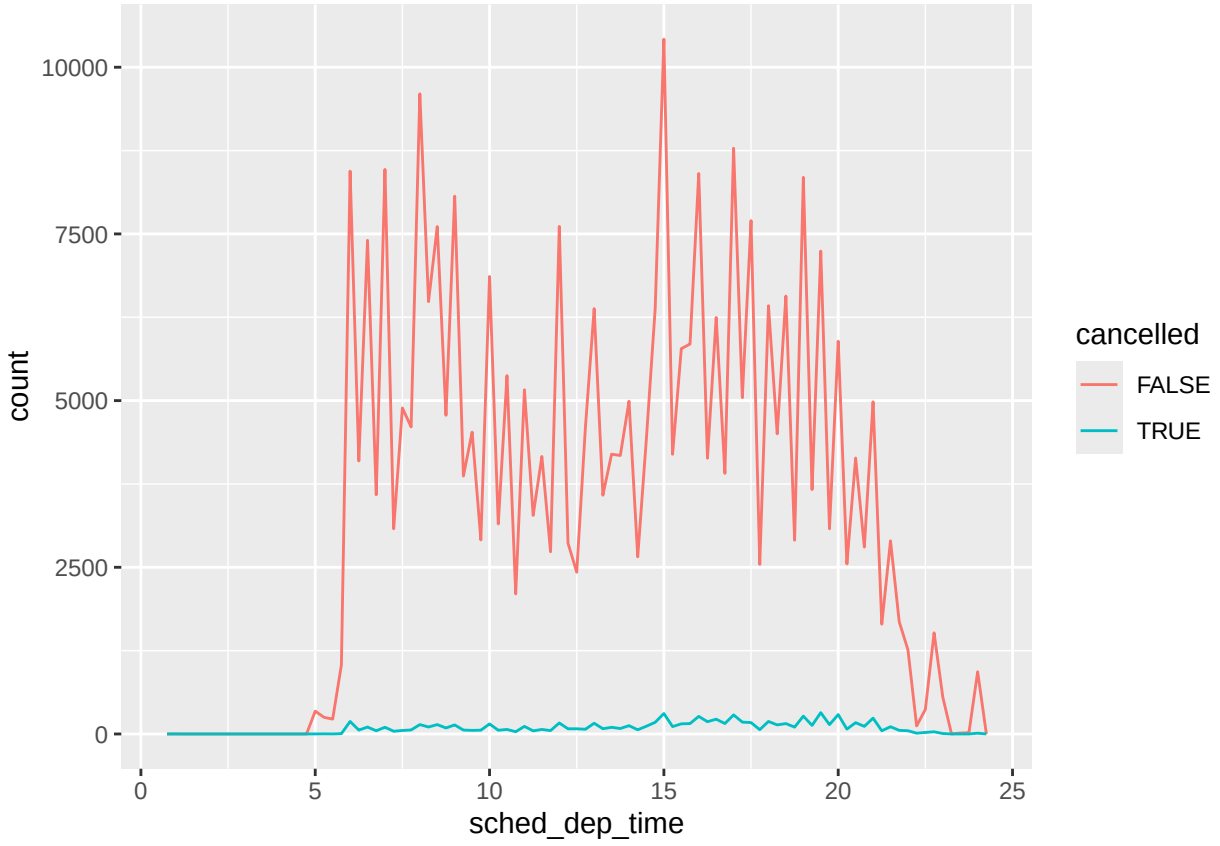
```r
ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
  geom_point()
```

```
## Warning: Removed 9 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



In some cases, the absence of data may still be encoding something. In nycflights13::flights, NAs in the dep_time variable mean a flight was cancelled. You could use is.na() to build a new variable to compare cancellations at different times. Note, that the include plot is good at comparing number of non-cancelled to cancelled flights, but not understanding the dynamics of just cancelled flights

```
nycflights13::flights %>%
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/% 100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + sched_min / 60
  ) %>%
  ggplot(mapping = aes(sched_dep_time)) +
    geom_freqpoly(mapping = aes(colour = cancelled), binwidth = 1/4)
```



7.5 - Covariation

John

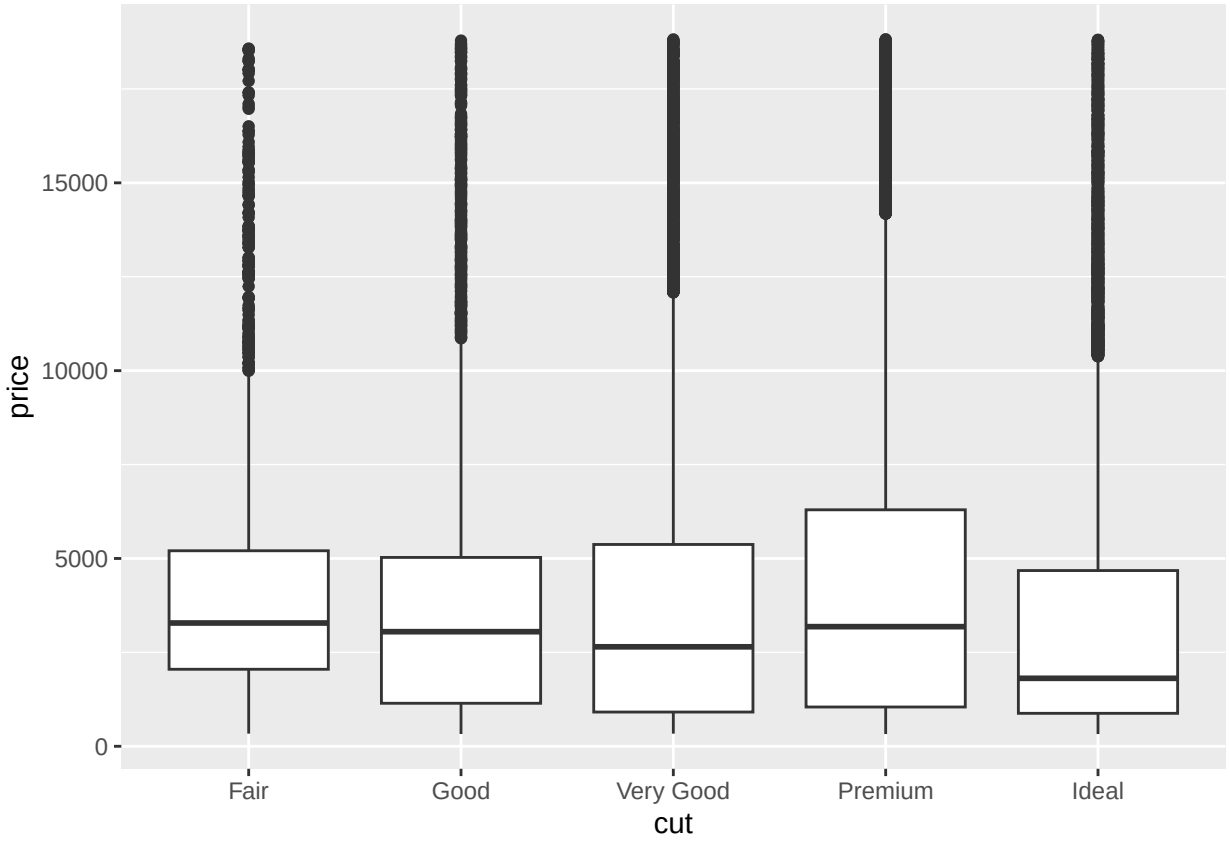
Covariation is when two or more variables have similar variations. It describes behavior between two or more variables.

It is important to recognize covariation between variables while performing EDA in order to explore the relationships between the variables, and identify trends and patterns.

The way you can recognize covariation is by plotting the relationship between these variables. This can be done between a continuous and categorical variable, two continuous variables, or two categorical variables.

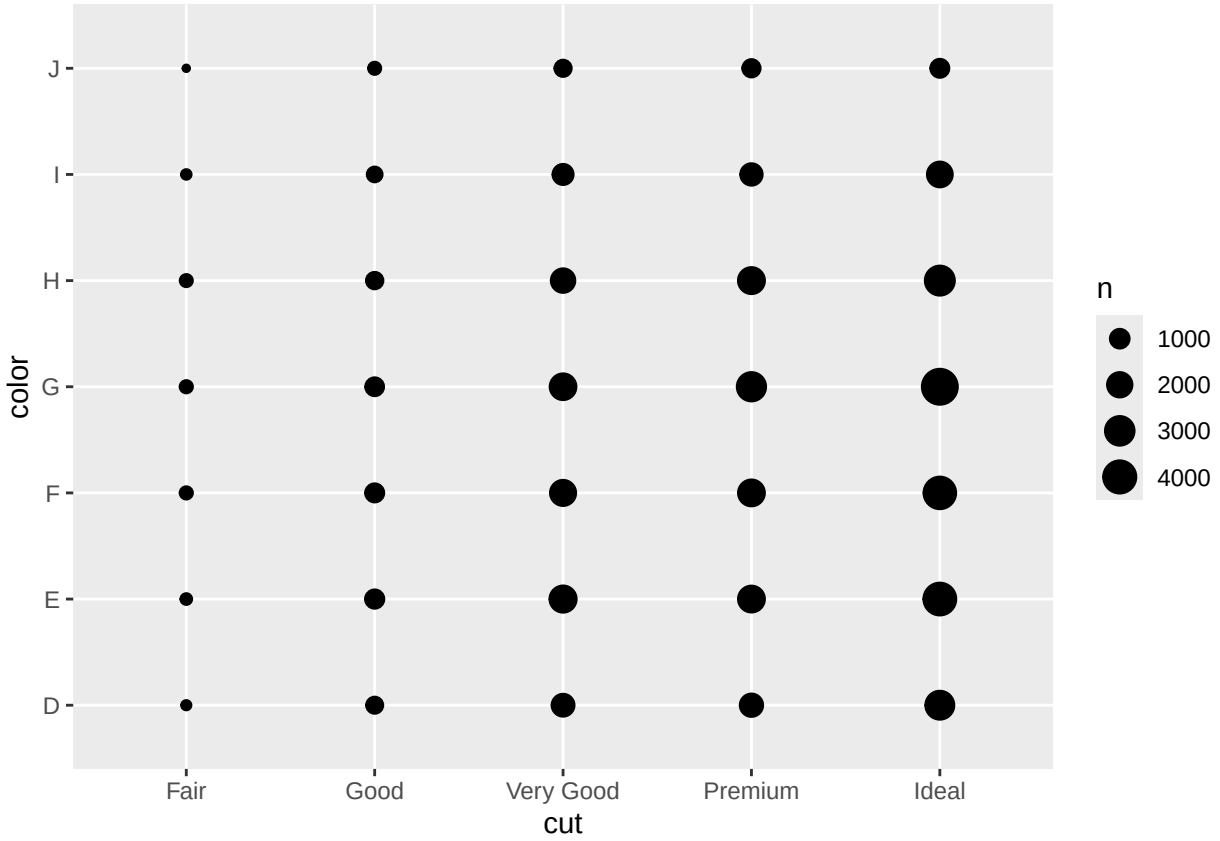
For a categorical and continuous variable, the best way to display covariation is through plotting boxplots. In the graph below, you can see that the diamonds that are "fair" in quality actually have the highest price, and the best quality diamonds are less expensive.

```
ggplot(data = diamonds, mapping = aes(x = cut, y = price)) +
  geom_boxplot()
```



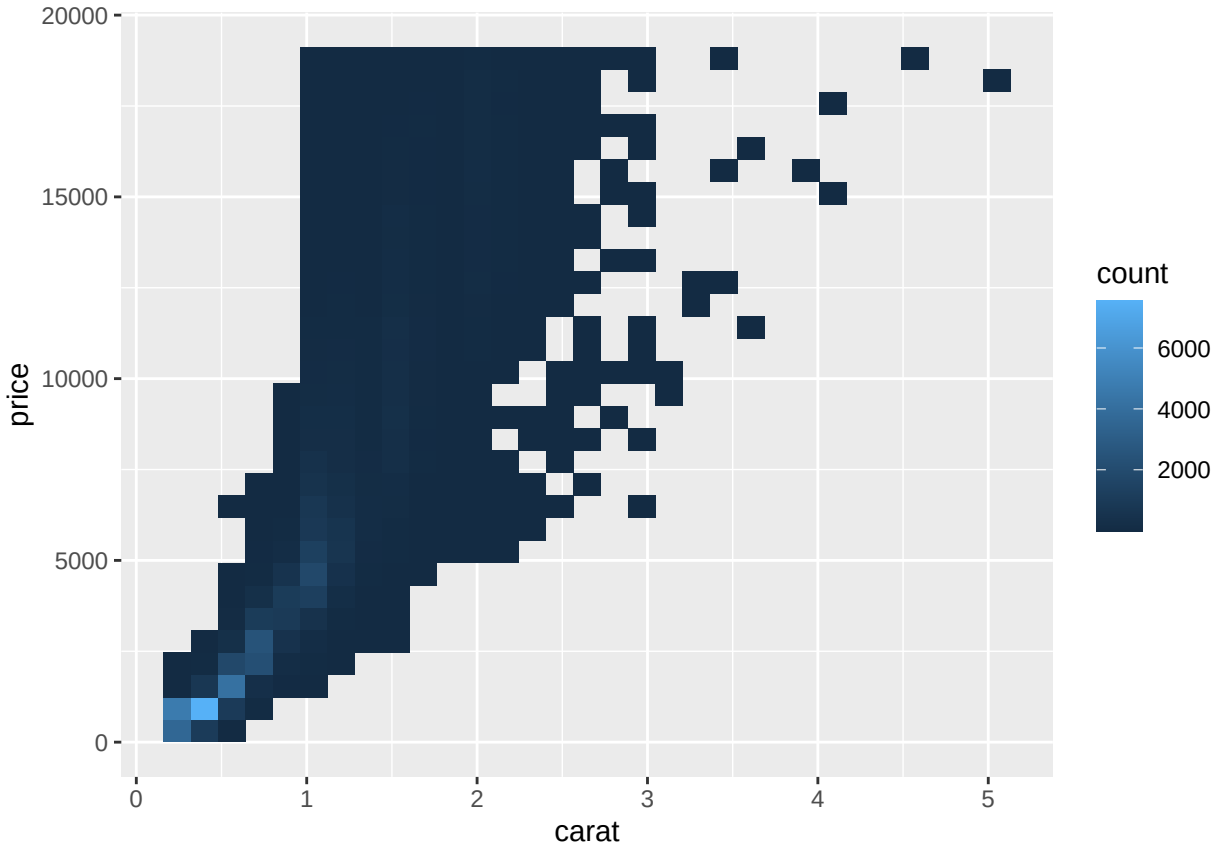
For two categorical variables, you have to count the observations between the two variables, and plot the counts.

```
ggplot(data = diamonds) +
  geom_count(mapping = aes(x = cut, y = color))
```



And for two continuous variables, a scatterplot or bin plot will show you the relationships between the variables.

```
ggplot(data = diamonds) +
  geom_bin2d(mapping = aes(x = carat, y = price))
```



All of these techniques help you visualize the relationships between variables, and help in identifying covariation.

7.6 Patterns and models

Andrea

People are really good at finding patterns, whether the truly exist or not. Finding patterns in data is an important part of research and can reveal previously unknown relationships, even ones that were not explicitly tested for.

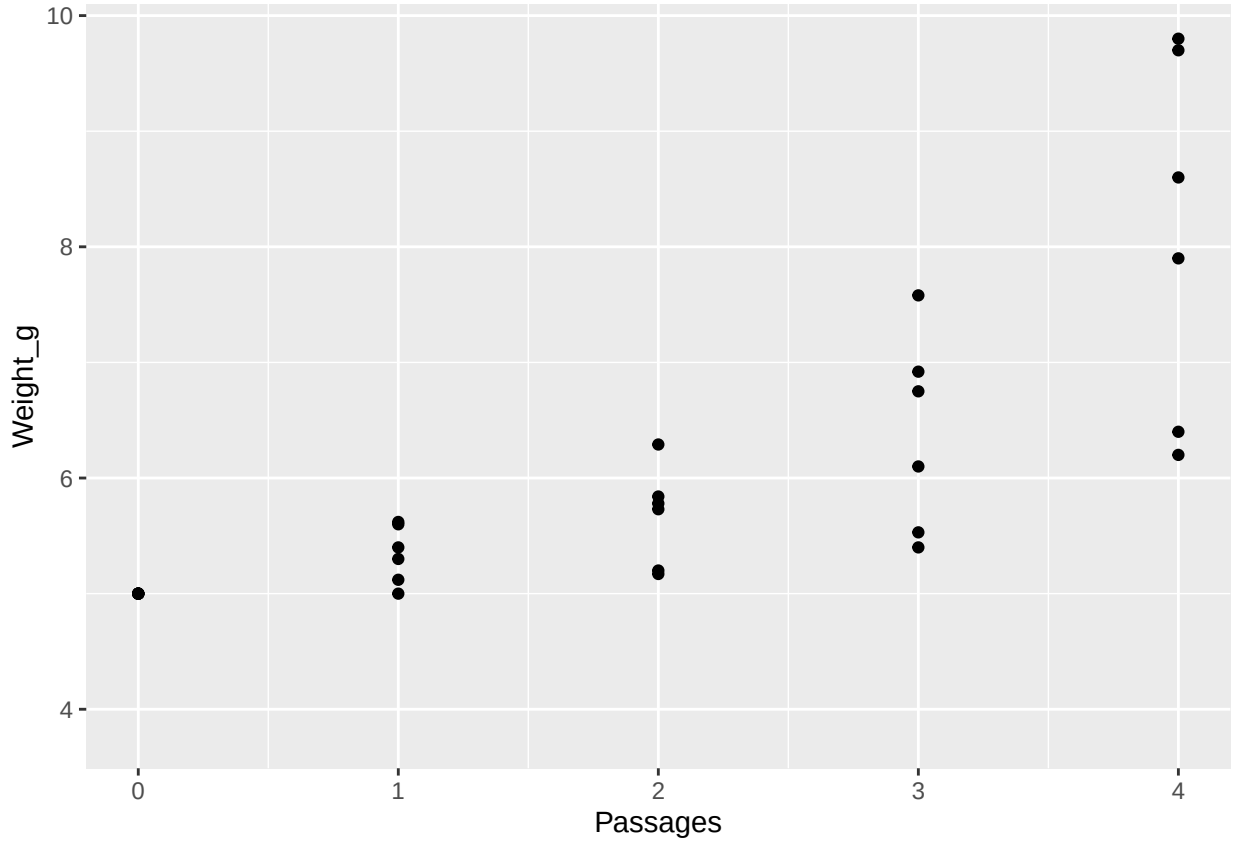
If a pattern is found, some questions should be asked: - How strong is the effect? - What variables influence the pattern? - Is it actually random chance?

Fitting a model can help answer these questions. For example, here's some data from an experiment I did with making Kefir, a fermented milk beverage. We started with 5 grams of grains in jars and "fed" them with milk. Every 48 hours we changed out the milk and weighed the grains.

```
colors <- c("#F80C66", "#1F7F80", "#40037F", "#AA66FF", "#66CCFE")

kefir.url <- "https://raw.githubusercontent.com/anddomen/Sourching_data_ST537/refs/heads/main/Data/Kefi

kefir.raw <- read.csv(kefir.url) |>
  filter(!is.na(Weight_g)) |>
  mutate(Date = as.Date(Date, format = "%m/%d/%y"),
         Day = as.numeric(Date - min(Date)),
         Passages  = case_match(Day,
```

```
                                0 ~ 0,
                                2 ~ 1,
                                4 ~ 2,
                                6 ~ 3,
                                8 ~ 4))

kefir.raw |>
  ggplot(aes(x = Passages,
             y = Weight_g))+
  geom_point()
```

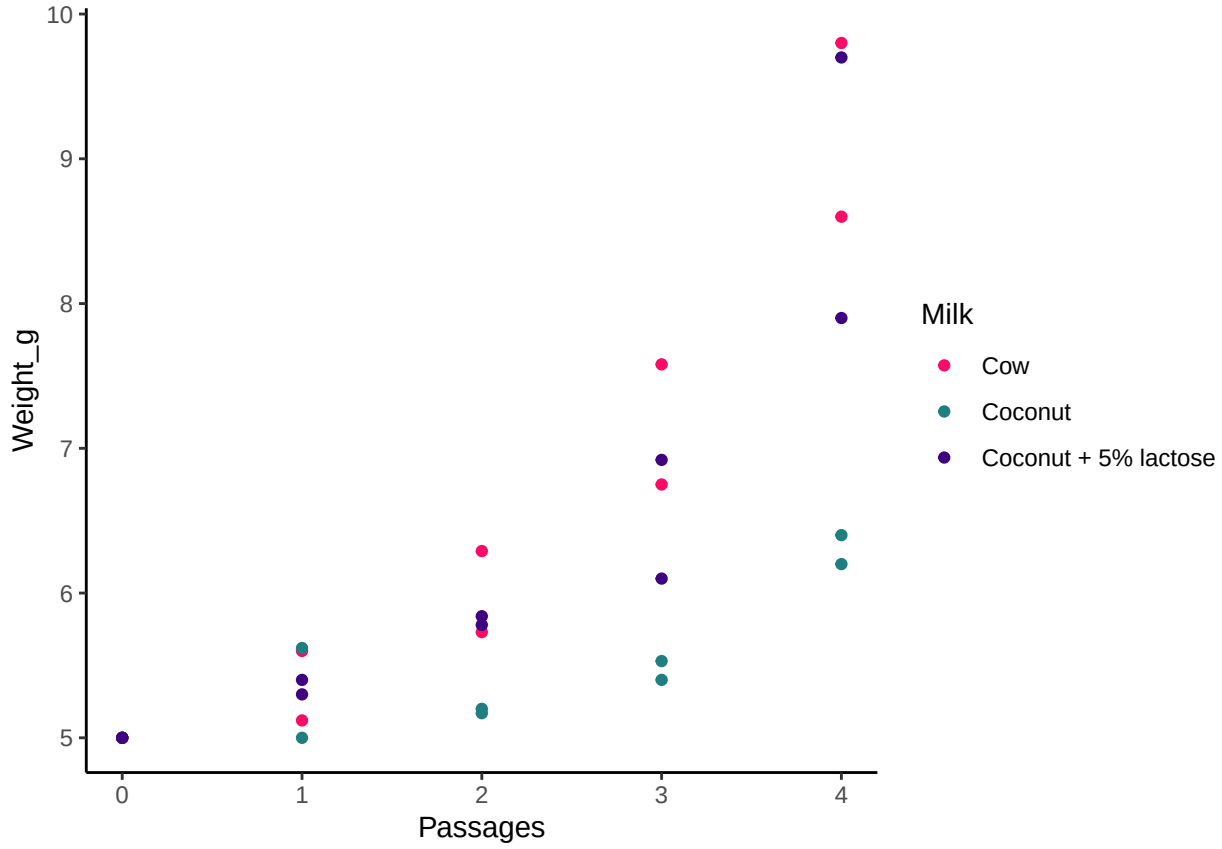
```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



The scatterplot here is a helpful beginning graph as a pattern can be seen here, without adding complexity of additional variables. This graph shows the mass of the grains increasing over time. This is to be expected. However, it does show that as the number of passages is increasing, there is a certain divergence and increase in variability. When we add the fact that some of the milk wasn't bovine based, but there was coconut milk and coconut milk with 5% lactose (a similar lactose content to whole cow's milk).

```
kefir.milk <- kefir.raw |>
  filter(!is.na(Passages)) |>
  mutate(Milk = case_when(
    grepl("^COW", Sample)  == TRUE ~ "Cow",
    grepl("^CCO", Sample)  == TRUE ~ "Coconut",
    grepl("^CLA", Sample)  == TRUE ~ "Coconut + 5% lactose",
    grepl("^GOAT", Sample) == TRUE ~ "Commercial goat",
```

```
    grepl("^LIFE", Sample) == TRUE ~ "Commercial cow"),
    Milk = factor(Milk,
                  levels = c("Cow", "Coconut", "Coconut + 5% lactose")))

kefir.milk |>
  ggplot(aes(x = Passages,
             y = Weight_g,
             color = Milk))+
  geom_point() +
  theme_classic()+
  scale_color_manual(values = colors)
```

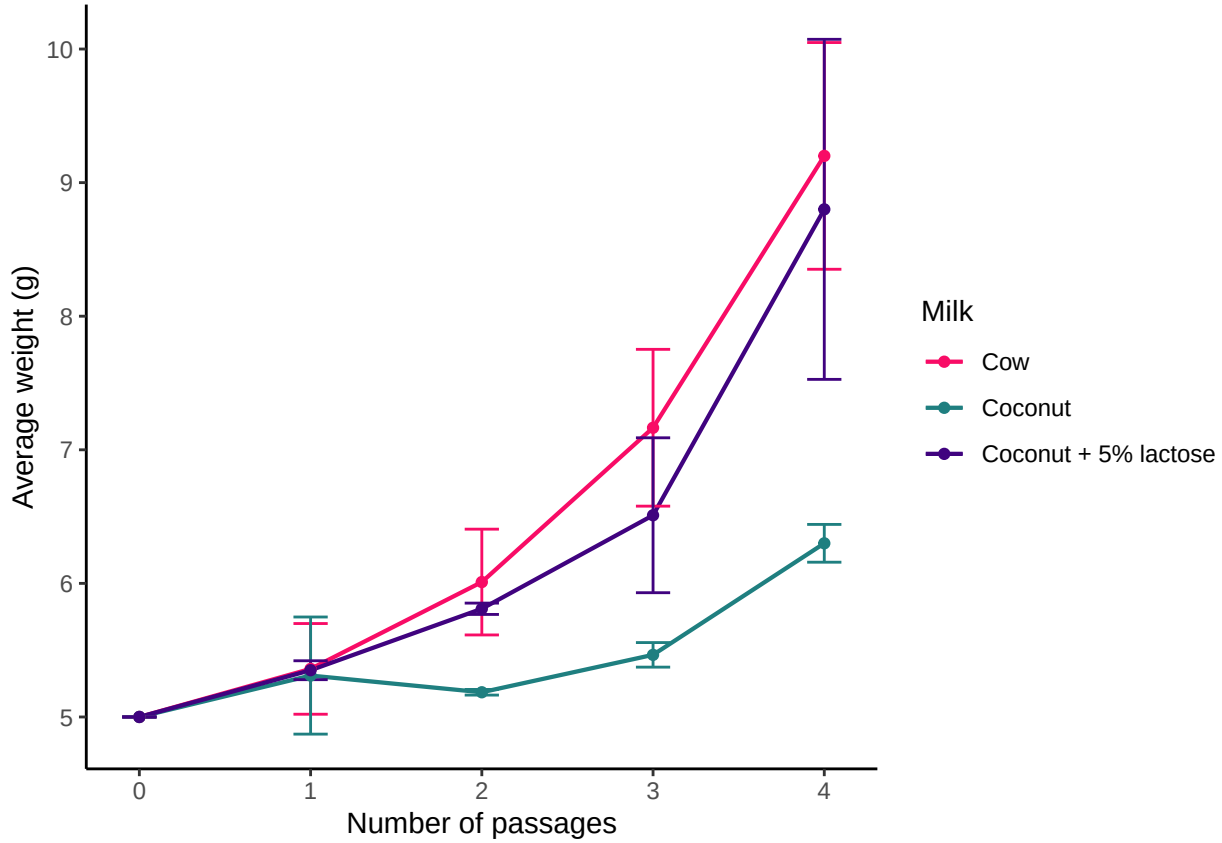


Now things are starting to make sense. The pattern here seems to be that cow and lactose-supplemented coconut milk have similar weights at the end of 4 passages while regular coconut milk lags behind. If we clean up the graph a bit more, the pattern is even clearer.

```
kefir.milk |>
  group_by(Milk, Passages) |>
  summarize(mean_wt = mean(Weight_g),
            stdev   = sd(Weight_g)) |>
  ggplot(aes(x = Passages,
             y = mean_wt,
             color = Milk)) +
  geom_line(linewidth = 0.75) +
  geom_errorbar(aes(ymin = mean_wt - stdev,
                    ymax = mean_wt + stdev), width = 0.2)+
```

```
  geom_point()+
  theme_classic() +
  labs(x = "Number of passages",
       y = "Average weight (g)") +
  scale_color_manual(values = colors)
```

```
## `summarise()` has grouped output by 'Milk'. You can override using the
## `.groups` argument.
```



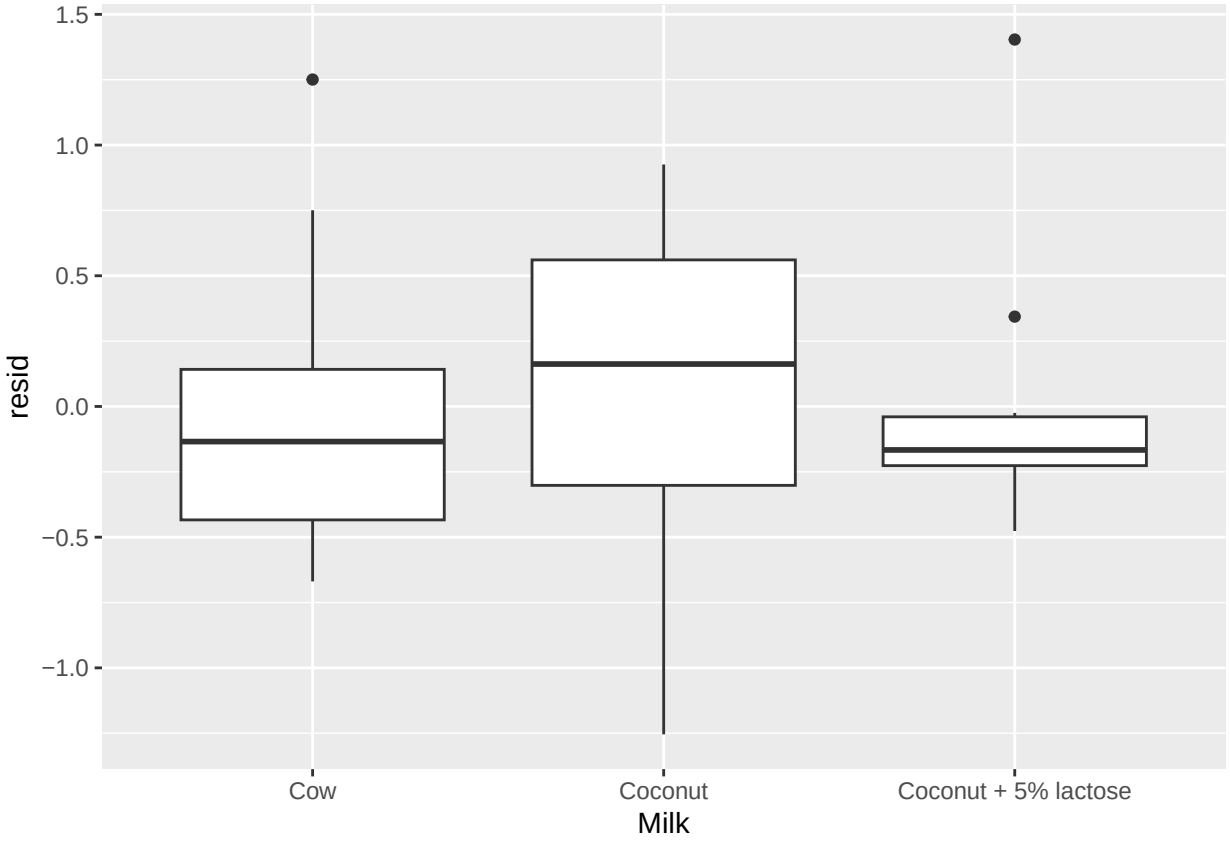
And when we fit a model to this data, it also shows that coconut milk was significantly different than the others, particularly on the last day.

```
milk.mod <- lm(Weight_g ~ Milk + factor(Passages), kefir.milk)
summary(milk.mod)
```

```
##
## Call:
## lm(formula = Weight_g ~ Milk + factor(Passages), data = kefir.milk)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2543 -0.3743 -0.1105  0.2736  1.4037
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 5.4493     0.3266  16.687 2.40e-14 ***
## MilkCoconut                -1.0950     0.3023  -3.622  0.00143 **
## MilkCoconut + 5% lactose   -0.2530     0.3023  -0.837  0.41130
```

```
## factor(Passages)1          0.3400     0.3903   0.871  0.39270
## factor(Passages)2          0.6683     0.3903   1.712  0.10029
## factor(Passages)3          1.3800     0.3903   3.536  0.00177 **
## factor(Passages)4          3.1000     0.3903   7.942 4.85e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.676 on 23 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7518
## F-statistic: 15.64 on 6 and 23 DF,  p-value: 4.319e-07
```

```
kefir.resid <- kefir.milk |>
  add_residuals(milk.mod)

kefir.resid |>
  ggplot(aes(x = Milk,
             y = resid)) +
  geom_boxplot()
```



. . . Okay the residuals could be nicer, but it's also real world data so this isn't terrible. This plot helps to compare the means and variability across the different milk types.

7.7 - New ggplot2 syntax

Paul

A concise expression of ggplot2 code will be used going forward.

Old form: ggplot(data = turnaround, mapping = aes(x = turn3, y = brighteyes)) + geom_barplot(binwidth = 0.25)

New form: ggplot(turnaround, aes(turn3, brighteyes)) + geom_barplot(binwidth = 0.25)

A reminder on adding a ggplot to the end of a pipeline: diamonds %>% count(are, fornever) %>% ggplot(aes(fornever, are, fill = n)) + # note the move to + after ggplot starts–at least until ggvis is released! geom_tile()

7.8 - Additional Resources

Paul

R for graphics cookbook (updated by Winston Change 2025-05-15) https://r-graphics.org/

ggplot2 book: 3rd edition https://ggplot2-book.org/

Springer link https://link.springer.com/book/10.1007/978-3-319-24277-4