2025 CCADD Winter Internship - Book Reading Session

Chapter 10: Exploratory Data Analysis

- https://r4ds.hadley.nz/eda

> "You ask question about whether your data meets your expectation."

EDA highlights **creativity** to generate a ***large quantity of questions in good quality***. You have to keep asking **"why"** throughout the whole process.

The tools of EDA are as follows:

- **visualization**,
- **transformation**,
- **modeling**

There is no specific rule for making a question, but there are some routines that data scientists usually do.

In this chapter, we use the `diamond` dataset to learn how to explore the data and extract useful discoveries within it.

In [1]:
```r
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.
0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts
() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all c
onflicts to become errors
```

In [2]:
```r
head(diamonds)
```

A tibble: 6 × 10

carat	cut	color	clarity	depth	table	price	x	y	z
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

1. Variation

: differences *within* a variable's values

Variation is the tendency of the values of a variable to change from measurement to measurement.

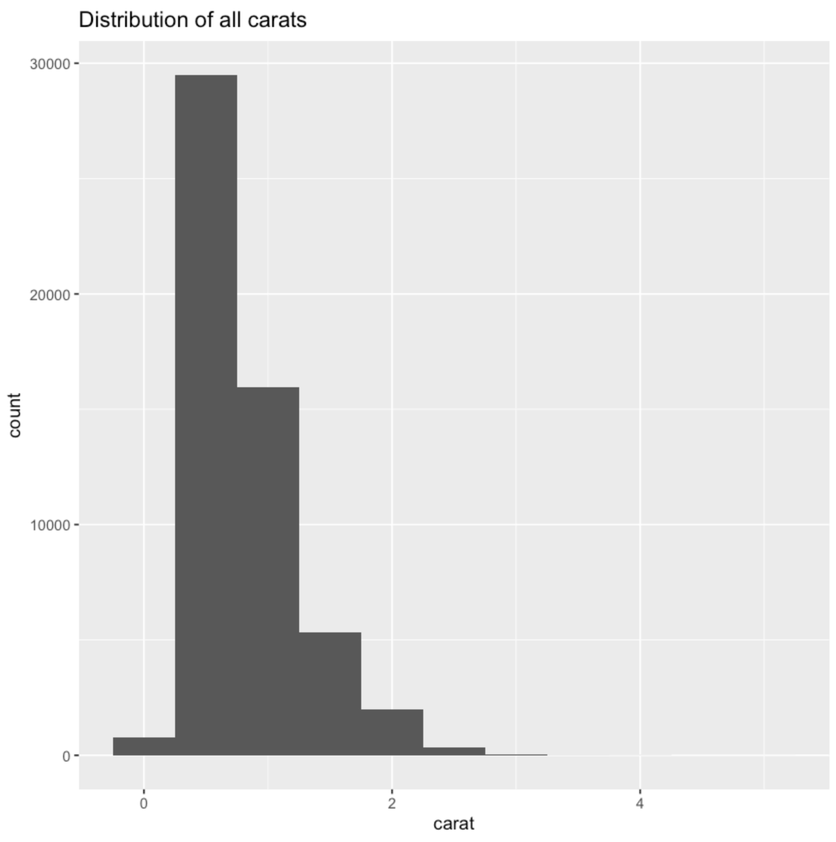
Understanding variation is crucial for data analysis, as it helps **identify patterns**, **anomalies**, and the **overall distribution** of data.

Key Points:

- **Distribution Analysis**: Examining the distribution of a variable's values is essential to understand its variation. This can be visualized using histograms, boxplots, or density plots.
- **Identifying Anomalies**: By analyzing variation, one can detect outliers or unusual observations that may require further investigation.

- **Data Transformation**: Understanding variation aids in deciding whether data transformations are necessary to meet the assumptions of statistical models.

```
In [3]: ggplot(diamonds) +
            geom_histogram(aes(x = carat), binwidth = 0.5) +
            ggtitle("Distribution of all carats")
```



#Right skewed

Follow-up questions:

> - Which values are **the most common?** + WHY?
> - Which values are **rare**? + Does that match your expectations?
> - Are there any **unusual patterns**? + What might explain them?

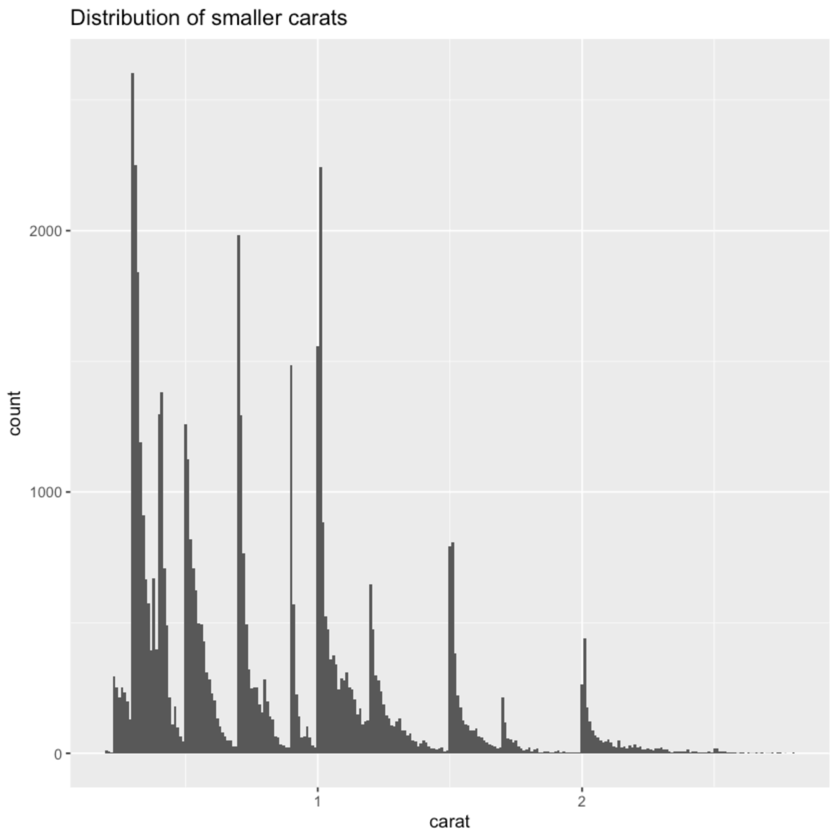
```
In [4]: summary(diamonds['carat']) # or summary(diamonds$carat)
```

```
     carat
Min.   :0.2000
1st Qu.:0.4000
Median :0.7000
Mean   :0.7979
3rd Qu.:1.0400
Max.   :5.0100
```

1.1. Typical Values

In [5]:
```
# Visualization of `carat` distribution for smaller diamonds
smaller <- diamonds |>
    filter(carat < 3)

ggplot(smaller, aes(x = carat)) +
    geom_histogram(binwidth = 0.01) +
    ggtitle("Distribution of smaller carats")
```



- *#Right_skewed_in_each_group*
- *0.5, 1, 1.5, 2*
- *#Clustered -> #Subgroups*

In [6]:
```
summary(smaller['carat'])
```

```
    carat
Min.   :0.2000
1st Qu.:0.4000
Median :0.7000
Mean   :0.7961
3rd Qu.:1.0400
Max.   :2.8000
```

> - Why are there more diamonds at whole carats (i.e., 1.0, 2.0) and common fractions of carats (i.e., 0.25, 0.5, 1.5)
> - Why are there more diamonds slightly to the right of each peak?

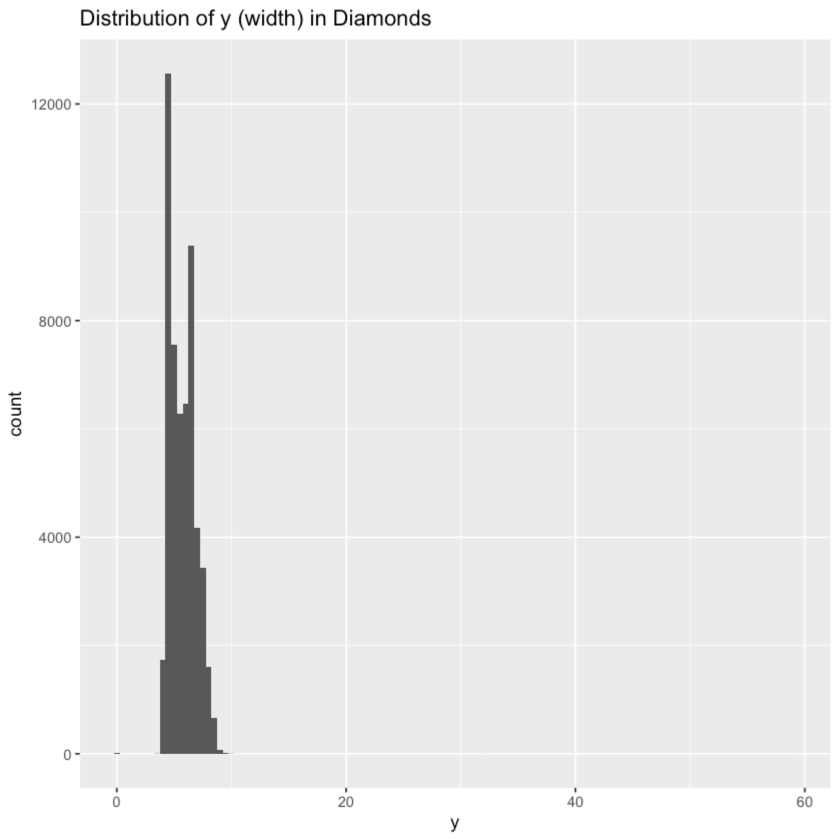
Questions to understand the subgroups:

> - How are the observations within each subgroup similar to each group similar to each other?
> - How are the observations in separate clusters different from each other?
> - How can you explain or describe the clusters?
> - Why might the appearance of clusters be misleading?

1.2. Unusual Values

- **Outliers**: unusual observations; data points that don't seem to fit the pattern
  - They might be due to data **entry errors**, **extremes** of the data collection, or **new discoveries**, etc.

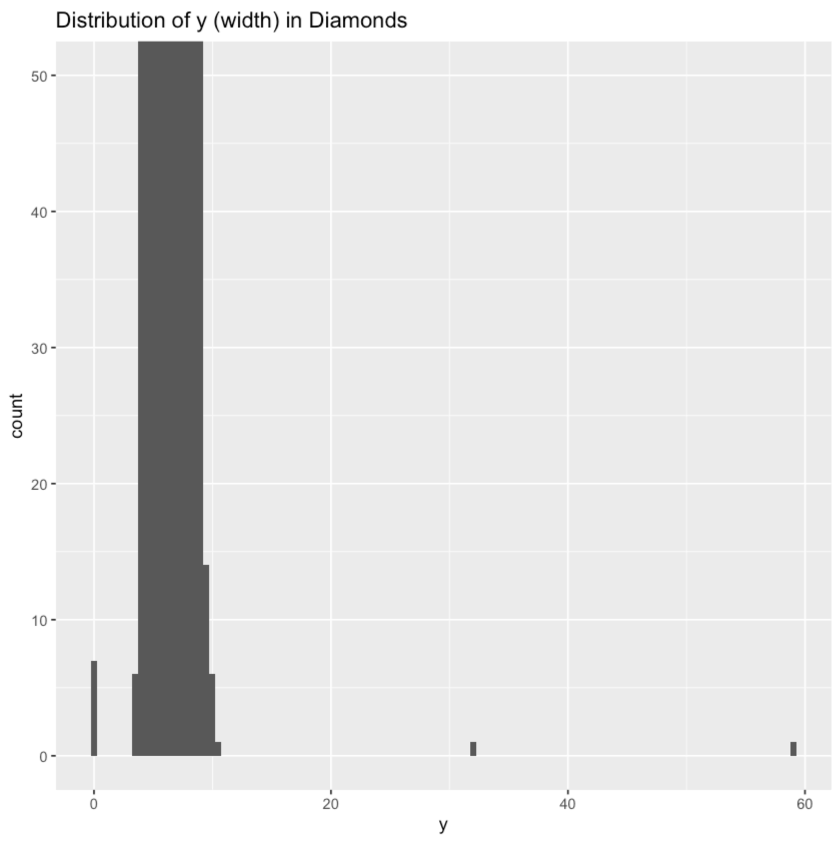
In [7]:
```r
# Distribution of 'y' variable
ggplot(diamonds, aes(x = y)) +
  geom_histogram(binwidth = 0.5) +
  labs(title = "Distribution of y (width) in Diamonds")
```



In this plot, you cannot see values larger than 10.0, but you know their existence because R automatically considers all the values, meaning their counts are so small. Thus, to make it easy to see the count of larger `y` values, we need to zoom.

- `coord_cartesian()` has `xlim` & `ylim` arguments to zoom in

In [8]:
```
# Distribution of 'y' variable
ggplot(diamonds, aes(x = y)) +
    geom_histogram(binwidth = 0.5) +
    coord_cartesian(ylim = c(0, 50)) +
    labs(title = "Distribution of y (width) in Diamonds")
```



In [9]:
```
unusual <- diamonds |>
    filter(y <3 | y > 20) |>
    select(price, x, y, z) |>
    arrange(y)

unusual
```

A tibble: 9 × 4

price	x	y	z
<int>	<dbl>	<dbl>	<dbl>
5139	0.00	0.0	0.00
6381	0.00	0.0	0.00
12800	0.00	0.0	0.00
15686	0.00	0.0	0.00
18034	0.00	0.0	0.00
2130	0.00	0.0	0.00
2130	0.00	0.0	0.00
2075	5.15	31.8	5.12
12210	8.09	58.9	8.06

In [10]:
```
summary(diamonds["price"])
```

```
    price
Min.   :  326
1st Qu.:  950
Median : 2401
Mean   : 3933
3rd Qu.: 5324
Max.   :18823
```

1. We know that diamonds cannot have a width of 0mm, so these values must be incorrect.
   - That means the `NA` s or missing data were coded as 0.
   - We can re-code these values as `NA` s to prevent misleading calculation
2. We can suspect the prices of width 32mm & 59mm diamonds: they are more than an inch long but too cheap!!

2 Options to move on:

> 1. Drop the entire row with strange values
> 2. Replace the unusual values with missing values (Recommended)

# 1. Drop the entire row with strange values diamonds2 <- diamonds |> filter(between(y,3,20)) show(diamonds2) #A tibble: 53,931 × 10

`if_else(<CONDITION>, true, false, missing = NULL)` :

- `true` : return value if the value is `TRUE` in the `CONDITION`
- `false` : return value if the value is `FALSE` in the `CONDITION`

In [11]:
```
# 2. Replace the unusual values with missing values (Recommended)
diamonds2 <- diamonds |>
```

```
    mutate(y = if_else(y < 3 | y > 20, NA, y))
diamonds2 |>
    filter(is.na(y))
```

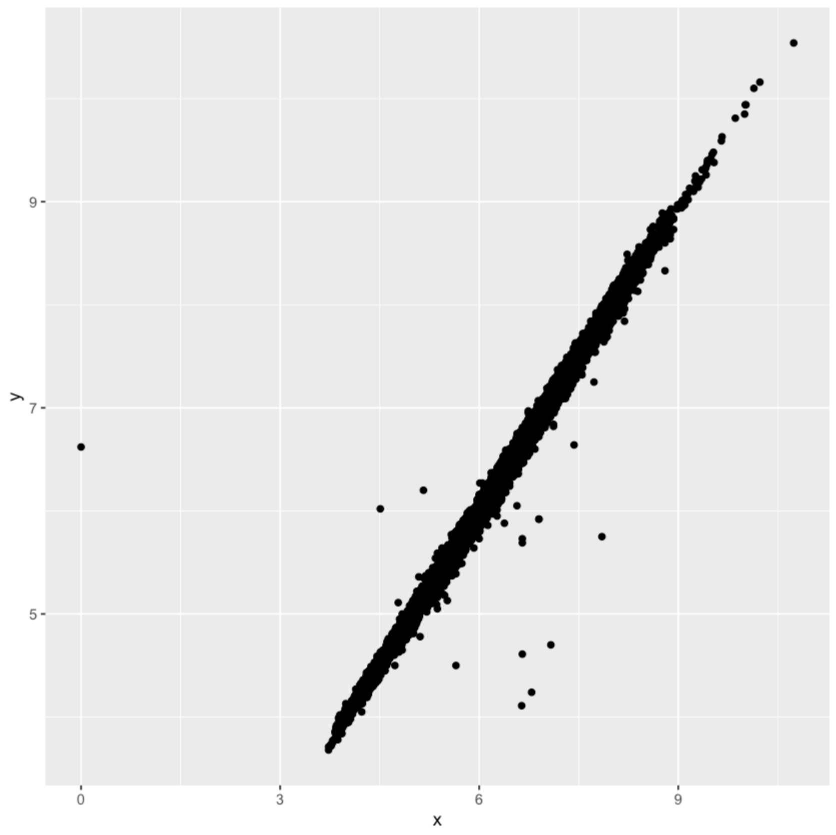
A tibble: 9 × 10

carat	cut	color	clarity	depth	table	price	x	y	z
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
1.00	Very Good	H	VS2	63.3	53	5139	0.00	NA	0.00
1.14	Fair	G	VS1	57.5	67	6381	0.00	NA	0.00
2.00	Premium	H	SI2	58.9	57	12210	8.09	NA	8.06
1.56	Ideal	G	VS2	62.2	54	12800	0.00	NA	0.00
1.20	Premium	D	VVS1	62.1	59	15686	0.00	NA	0.00
2.25	Premium	H	SI2	62.8	59	18034	0.00	NA	0.00
0.51	Ideal	E	VS1	61.8	55	2075	5.15	NA	5.12
0.71	Good	F	SI2	64.1	60	2130	0.00	NA	0.00
0.71	Good	F	SI2	64.1	60	2130	0.00	NA	0.00

```
In [12]: sum(is.na(diamonds2$y))
```

9

```
In [13]: ggplot(diamonds2, aes(x = x, y = y)) +
             geom_point(na.rm = TRUE) # `na.rm = TRUE` is set to suppress the warning
```



```
In [14]: # install.packages("nycflights13")
         library(nycflights13)
```

```
In [15]: View(flights)
```

A tibb

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
2013	1	1	517	515	2	830	819
2013	1	1	533	529	4	850	830
2013	1	1	542	540	2	923	850
2013	1	1	544	545	-1	1004	1022
2013	1	1	554	600	-6	812	837
2013	1	1	554	558	-4	740	728
2013	1	1	555	600	-5	913	854
2013	1	1	557	600	-3	709	723
2013	1	1	557	600	-3	838	846
2013	1	1	558	600	-2	753	745
2013	1	1	558	600	-2	849	851
2013	1	1	558	600	-2	853	856
2013	1	1	558	600	-2	924	917
2013	1	1	558	600	-2	923	937

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
2013	1	1	559	600	-1	941	910
2013	1	1	559	559	0	702	706
2013	1	1	559	600	-1	854	902
2013	1	1	600	600	0	851	858
2013	1	1	600	600	0	837	825
2013	1	1	601	600	1	844	850
2013	1	1	602	610	-8	812	820
2013	1	1	602	605	-3	821	805
2013	1	1	606	610	-4	858	910
2013	1	1	606	610	-4	837	845
2013	1	1	607	607	0	858	915
2013	1	1	608	600	8	807	735
2013	1	1	611	600	11	945	931
2013	1	1	613	610	3	925	921
2013	1	1	615	615	0	1039	1100

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
2013	1	1	615	615	0	833	842
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2013	9	30	2123	2125	-2	2223	2247
2013	9	30	2127	2129	-2	2314	2323
2013	9	30	2128	2130	-2	2328	2359
2013	9	30	2129	2059	30	2230	2232
2013	9	30	2131	2140	-9	2225	2255
2013	9	30	2140	2140	0	10	40
2013	9	30	2142	2129	13	2250	2239
2013	9	30	2145	2145	0	115	140
2013	9	30	2147	2137	10	30	27
2013	9	30	2149	2156	-7	2245	2308
2013	9	30	2150	2159	-9	2250	2306
2013	9	30	2159	1845	194	2344	2030

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
2013	9	30	2203	2205	-2	2339	2331
2013	9	30	2207	2140	27	2257	2250
2013	9	30	2211	2059	72	2339	2242
2013	9	30	2231	2245	-14	2335	2356
2013	9	30	2233	2113	80	112	30
2013	9	30	2235	2001	154	59	2249
2013	9	30	2237	2245	-8	2345	2353
2013	9	30	2240	2245	-5	2334	2351
2013	9	30	2240	2250	-10	2347	7
2013	9	30	2241	2246	-5	2345	1
2013	9	30	2307	2255	12	2359	2358
2013	9	30	2349	2359	-10	325	350
2013	9	30	NA	1842	NA	NA	2019
2013	9	30	NA	1455	NA	NA	1634
2013	9	30	NA	2200	NA	NA	2312

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>
2013	9	30	NA	1210	NA	NA	1330
2013	9	30	NA	1159	NA	NA	1344
2013	9	30	NA	840	NA	NA	1020

In [16]:
```r
# Sometimes NA is critical to understand the data
flights_mutated <- nycflights13::flights |>
    mutate(
        cancelled = is.na(dep_time), # For this dataset, NA in dep_time indi
        sched_hour = sched_dep_time %/% 100,
        sched_min = sched_dep_time %% 100,
        sched_dep_time = sched_hour + (sched_min / 60)
    )

head(flights_mutated)
```

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time
<int>	<int>	<int>	<int>	<dbl>	<dbl>	<int>	<int>
2013	1	1	517	5.250000	2	830	819
2013	1	1	533	5.483333	4	850	830
2013	1	1	542	5.666667	2	923	850
2013	1	1	544	5.750000	-1	1004	1022
2013	1	1	554	6.000000	-6	812	837
2013	1	1	554	5.966667	-4	740	728

- `%/%` (Modulo): an operator gives the remainder of the division of two numbers.
- `%%` (Integer Division): an operator performs integer division, giving the quotient without the fractional part.

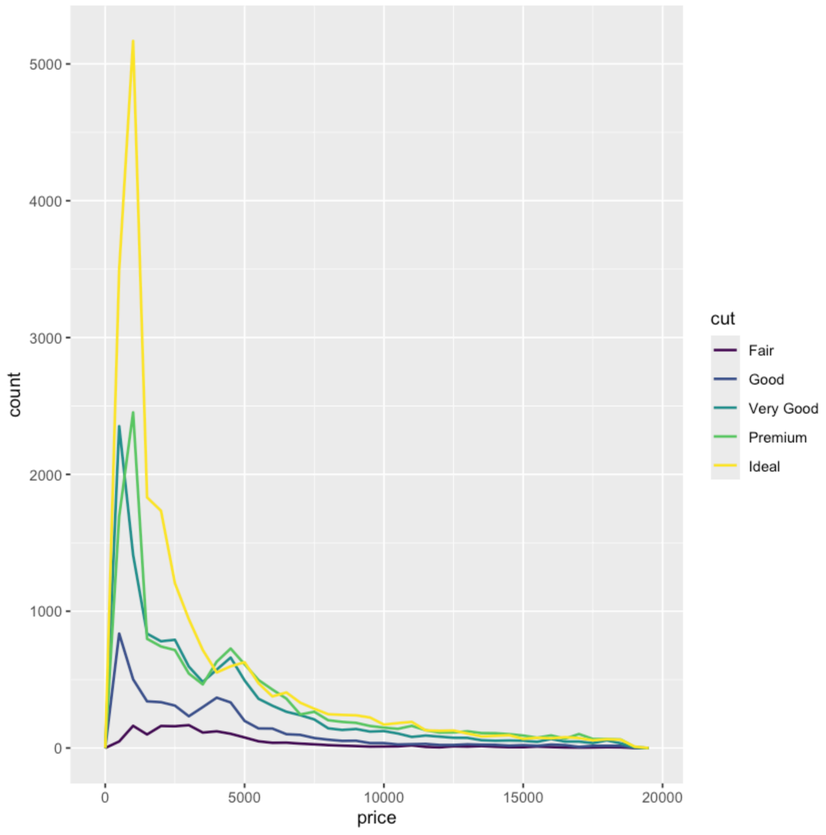
2. Covariation

- Covariation describes the behavior *between* variables.
- Covariation is the tendency for the values of two or more variables to vary together in a related way.

2.1. A Categorical and a Numerical Variable

- `price` (numerical variable) & `cut` (or quality; categorial variable)
  - `cut` is defined as an ordered factor variable.

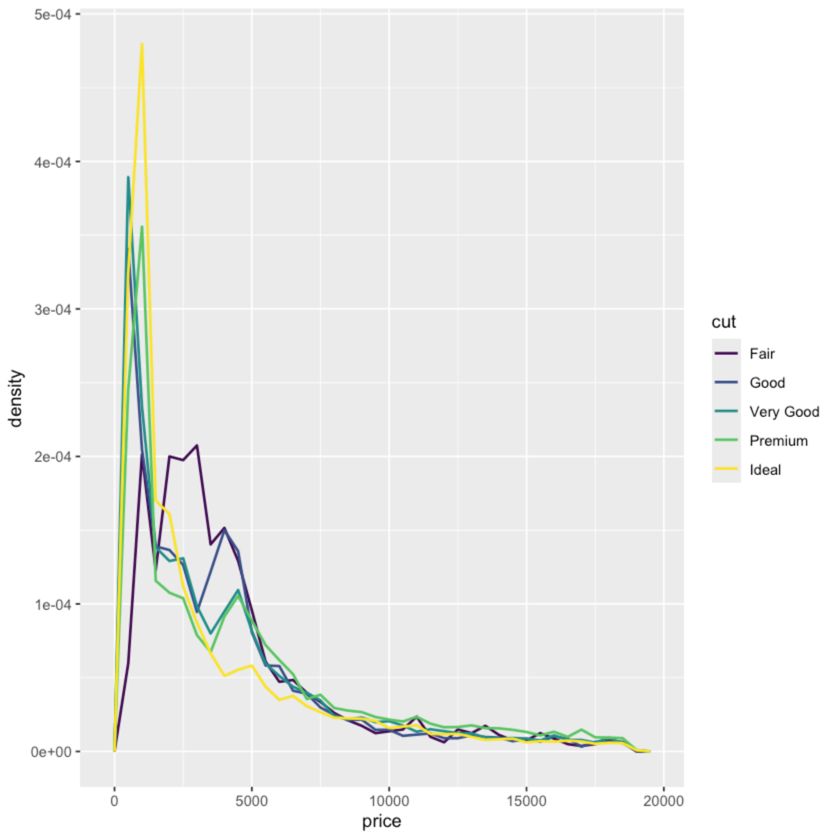
In [17]:
```
ggplot(diamonds) +
    geom_freqpoly(aes(x = price, color = cut), binwidth = 500, linewidth=0.7
```



- To compare the group, count might not be the best option.
- Normalization by density is required.
  - `y = after_stat(density)`: Normalized value by dividing the frequency of each section by the total number of data

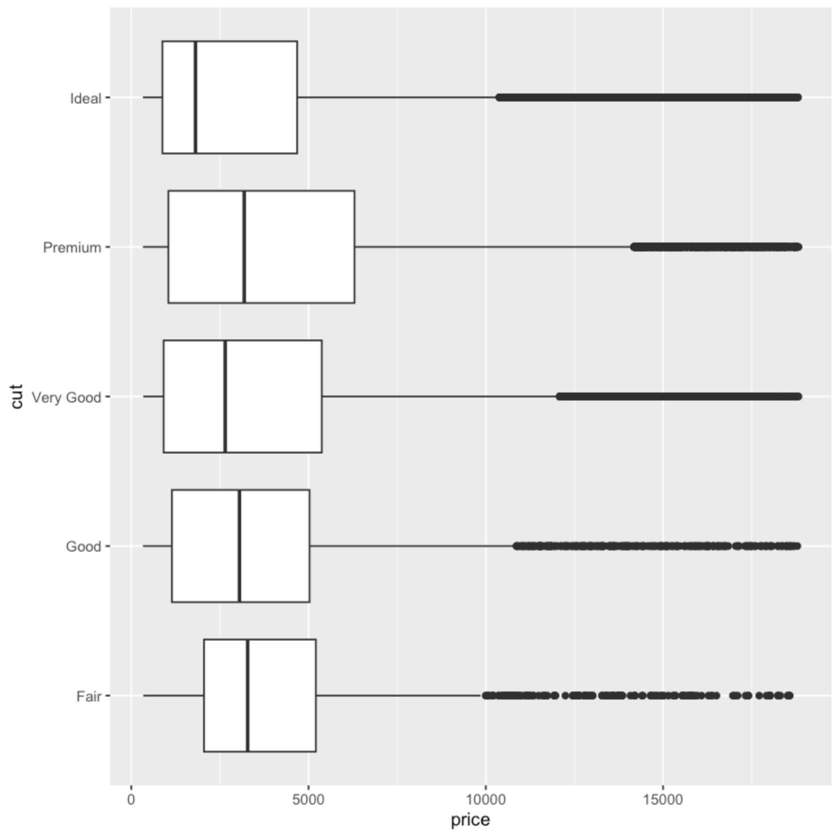
- Summation of `density` is 1, indicating the relative distribution of the data.

In [18]:
```
ggplot(diamonds) +
    geom_freqpoly(aes(x = price, y = after_stat(density), color = cut), binw
```



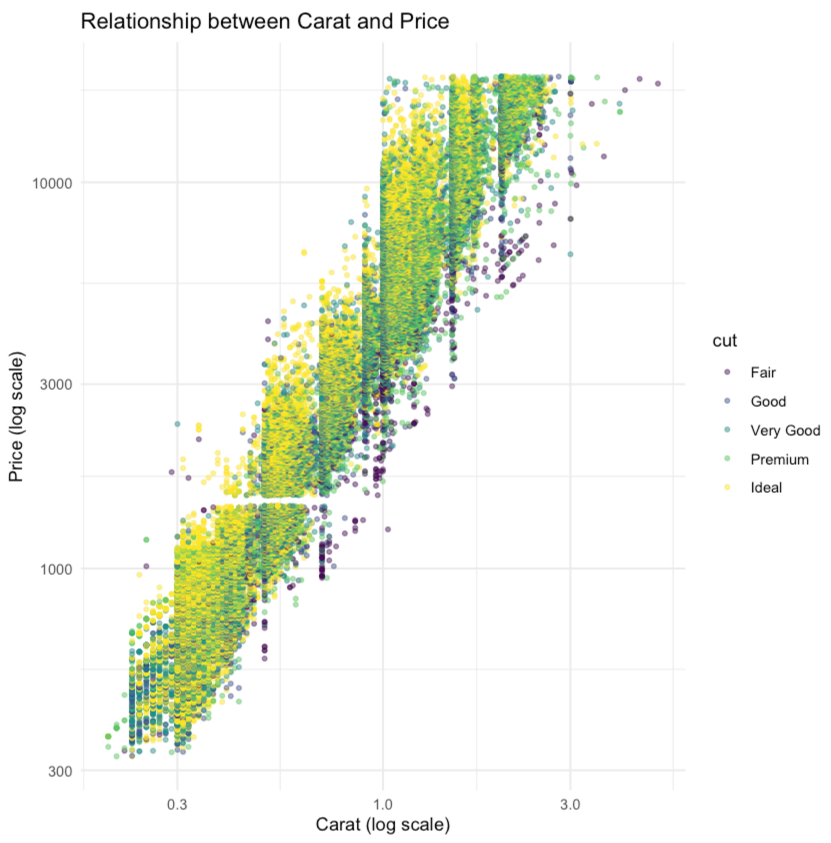
- Does it mean fair diamonds (the lowest quality) have the highest average price?

In [19]:
```
ggplot(diamonds) +
    geom_boxplot(aes(x = cut, y = price)) +
    coord_flip()
```



Better quality diamonds are typically cheaper

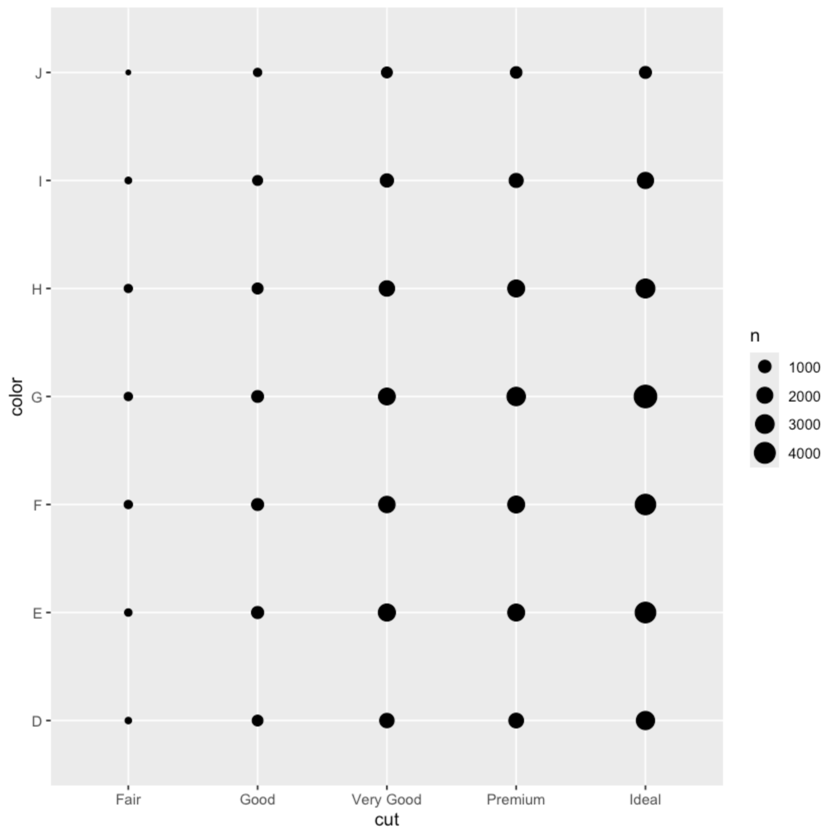
In [20]:
```
ggplot(diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point(alpha = 0.5, size = 1) +
  scale_x_log10() + # Log scale for x-axis (Carat)
  scale_y_log10() + # Log scale for y-axis (Price)
  labs(
    title = "Relationship between Carat and Price",
    x = "Carat (log scale)",
    y = "Price (log scale)"
  ) +
  theme_minimal()
```



2.2. Two Categorical Variables

Count the number of observations for each combination of levels of these categorical variables.

```
In [21]: ## ggplot2's `geom_count()`
ggplot(diamonds, aes(x = cut, y = color)) +
    geom_count()
```



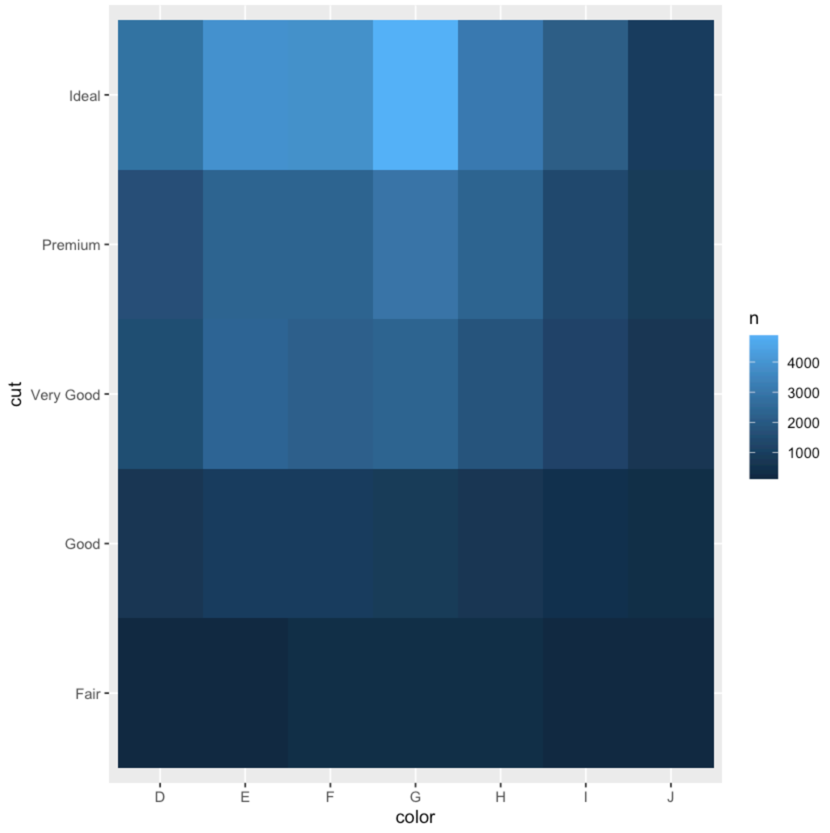
The size of each circle in the plot displays how many observations occurred at each combination of values.

In [22]:
```
# dplyr
diamonds |>
    count(color, cut) |>
    arrange(desc(n)) |>
    head(10)
```

A tibble: 10 × 3

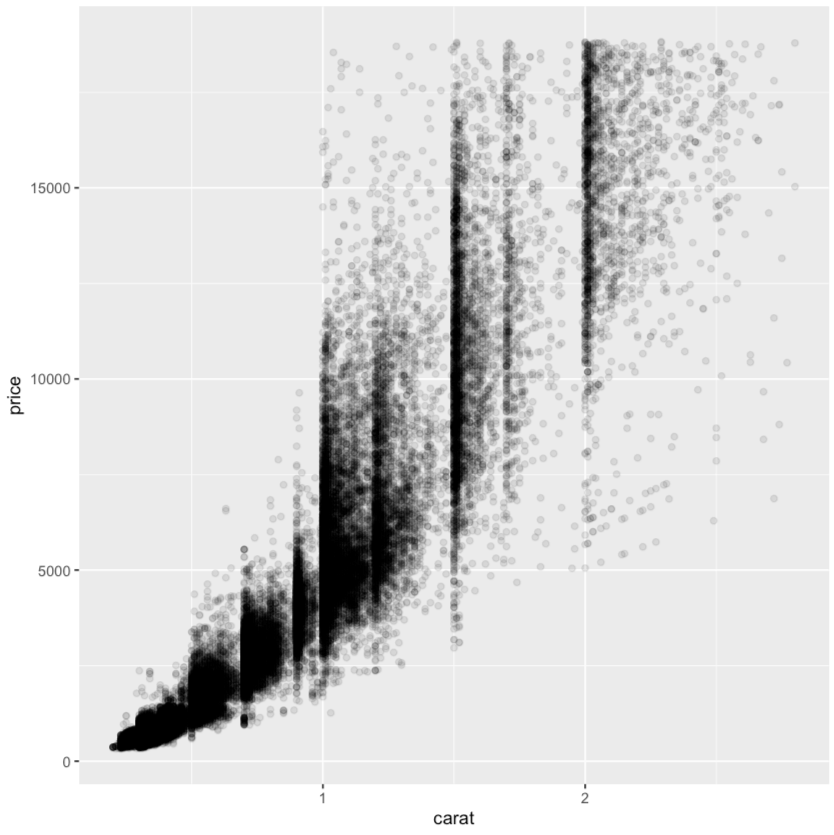
color	cut	n
<ord>	<ord>	<int>
G	Ideal	4884
E	Ideal	3903
F	Ideal	3826
H	Ideal	3115
G	Premium	2924
D	Ideal	2834
E	Very Good	2400
H	Premium	2360
E	Premium	2337
F	Premium	2331

```
In [23]: # `geom_tile`
         diamonds |>
           count(color, cut) |>
           ggplot() +
           geom_tile(aes(x = color, y = cut, fill = n))
```



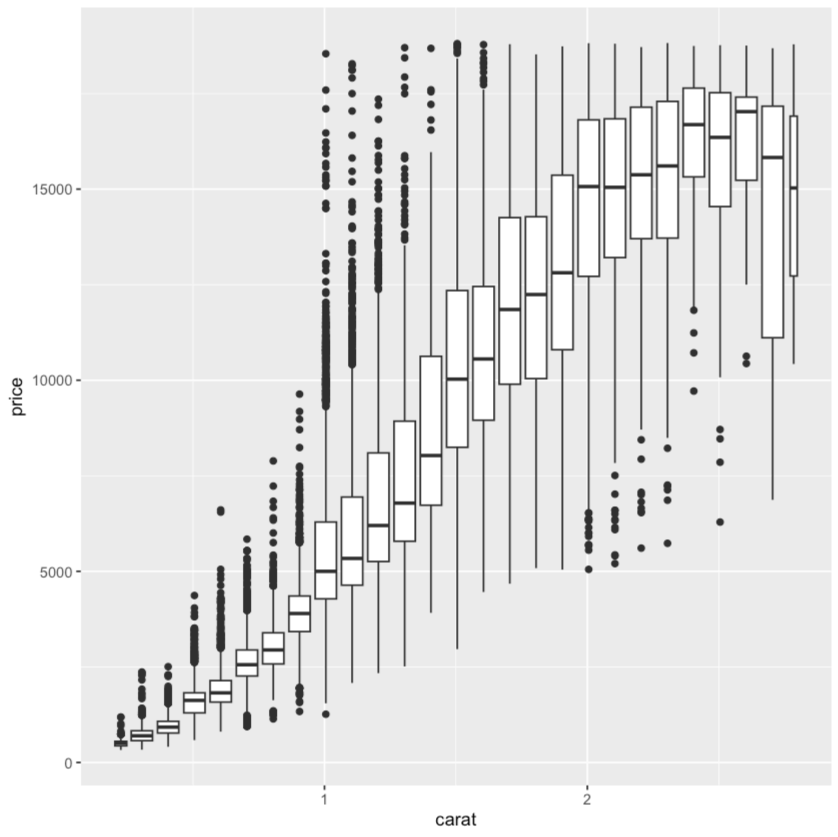
2.3. Two Numerical Variables

In [24]:
```
ggplot(smaller) +
  geom_point(aes(x = carat, y = price), alpha = 0.1)
```



Bin one continuous variable (`carat`) so it acts like a categorical variable

In [25]:
```
ggplot(smaller) +
    geom_boxplot(aes(x = carat, y = price,
                     group = cut_width(carat, 0.1))) # cut_width(x, width)
```



2.4. Patterns & Models

Patterns:

If a systematic relationship exists between two variables it will appear as a pattern in the data.

> - Could this pattern be due to randomness
> - How can you describe the relationship implied by the pattern?
> - How strong does the pattern imply the relationship?
> - Does the relationship change if you look at individual subgroups of the data or other variables?

If you think of variation as a phenomenon that creates uncertainty, covariation is a phenomenon that reduces it.

Models:

Models are a tool for extracting patterns out of data. It's possible to use a model to remove the very strong relationship between price and carat so we can explore the subtleties that remain.

install.packages("tidymodels")

```
In [26]: library(tidymodels)
```

```
── Attaching packages ──────────────────────────────────────── tidymodels 1.2.
0 ──

✔ broom        1.0.7     ✔ rsample      1.2.1
✔ dials        1.3.0     ✔ tune         1.2.1
✔ infer        1.0.7     ✔ workflows    1.1.4
✔ modeldata    1.4.0     ✔ workflowsets 1.1.0
✔ parsnip      1.2.1     ✔ yardstick    1.3.1
✔ recipes      1.1.0

── Conflicts ───────────────────────────────────────── tidymodels_conflicts
() ──
✖ scales::discard() masks purrr::discard()
✖ dplyr::filter()   masks stats::filter()
✖ recipes::fixed()  masks stringr::fixed()
✖ dplyr::lag()      masks stats::lag()
✖ yardstick::spec() masks readr::spec()
✖ recipes::step()   masks stats::step()
• Dig deeper into tidy modeling with R at https://www.tmwr.org
```

Log transform the values of `carat` & `price` first, and fit a model to the log-transformed values.

In [27]:
```
diamonds <- diamonds |>
  mutate(
    log_price = log(price),
    log_carat = log(carat)
  )

View(diamonds)
```

A tibble: 53940 × 12

carat	cut	color	clarity	depth	table	price	x	y	z	log_price
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43	5.786897
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31	5.786897
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31	5.789960
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63	5.811141
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75	5.814131
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48	5.817111
0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47	5.817111
0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53	5.820083
0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49	5.820083
0.23	Very Good	H	VS1	59.4	61	338	4.00	4.05	2.39	5.823046
0.30	Good	J	SI1	64.0	55	339	4.25	4.28	2.73	5.826000
0.23	Ideal	J	VS1	62.8	56	340	3.93	3.90	2.46	5.828946
0.22	Premium	F	SI1	60.4	61	342	3.88	3.84	2.33	5.834811
0.31	Ideal	J	SI2	62.2	54	344	4.35	4.37	2.71	5.840642
0.20	Premium	E	SI2	60.2	62	345	3.79	3.75	2.27	5.843544
0.32	Premium	E	I1	60.9	58	345	4.38	4.42	2.68	5.843544
0.30	Ideal	I	SI2	62.0	54	348	4.31	4.34	2.68	5.852202
0.30	Good	J	SI1	63.4	54	351	4.23	4.29	2.70	5.860786
0.30	Good	J	SI1	63.8	56	351	4.23	4.26	2.71	5.860786
0.30	Very Good	J	SI1	62.7	59	351	4.21	4.27	2.66	5.860786
0.30	Good	I	SI2	63.3	56	351	4.26	4.30	2.71	5.860786
0.23	Very Good	E	VS2	63.8	55	352	3.85	3.92	2.48	5.863631
0.23	Very Good	H	VS1	61.0	57	353	3.94	3.96	2.41	5.866468
0.31	Very Good	J	SI1	59.4	62	353	4.39	4.43	2.62	5.866468
0.31	Very Good	J	SI1	58.1	62	353	4.44	4.47	2.59	5.866468

carat	cut	color	clarity	depth	table	price	x	y	z	log_price
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
0.23	Very Good	G	VVS2	60.4	58	354	3.97	4.01	2.41	5.869297
0.24	Premium	I	VS1	62.5	57	355	3.97	3.94	2.47	5.872118
0.30	Very Good	J	VS2	62.2	57	357	4.28	4.30	2.67	5.877736
0.23	Very Good	D	VS2	60.5	61	357	3.96	3.97	2.40	5.877736
0.23	Very Good	F	VS1	60.9	57	357	3.96	3.99	2.42	5.877736
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.70	Premium	E	SI1	60.5	58	2753	5.74	5.77	3.48	7.920447
0.57	Premium	E	IF	59.8	60	2753	5.43	5.38	3.23	7.920447
0.61	Premium	F	VVS1	61.8	59	2753	5.48	5.40	3.36	7.920447
0.80	Good	G	VS2	64.2	58	2753	5.84	5.81	3.74	7.920447
0.84	Good	I	VS1	63.7	59	2753	5.94	5.90	3.77	7.920447
0.77	Ideal	E	SI2	62.1	56	2753	5.84	5.86	3.63	7.920447
0.74	Good	D	SI1	63.1	59	2753	5.71	5.74	3.61	7.920447
0.90	Very Good	J	SI1	63.2	60	2753	6.12	6.09	3.86	7.920447
0.76	Premium	I	VS1	59.3	62	2753	5.93	5.85	3.49	7.920447
0.76	Ideal	I	VVS1	62.2	55	2753	5.89	5.87	3.66	7.920447
0.70	Very Good	E	VS2	62.4	60	2755	5.57	5.61	3.49	7.921173
0.70	Very Good	E	VS2	62.8	60	2755	5.59	5.65	3.53	7.921173
0.70	Very Good	D	VS1	63.1	59	2755	5.67	5.58	3.55	7.921173
0.73	Ideal	I	VS2	61.3	56	2756	5.80	5.84	3.57	7.921536
0.73	Ideal	I	VS2	61.6	55	2756	5.82	5.84	3.59	7.921536
0.79	Ideal	I	SI1	61.6	56	2756	5.95	5.97	3.67	7.921536
0.71	Ideal	E	SI1	61.9	56	2756	5.71	5.73	3.54	7.921536
0.79	Good	F	SI1	58.1	59	2756	6.06	6.13	3.54	7.921536
0.79	Premium	E	SI2	61.4	58	2756	6.03	5.96	3.68	7.921536
0.71	Ideal	G	VS1	61.4	56	2756	5.76	5.73	3.53	7.921536

carat	cut	color	clarity	depth	table	price	x	y	z	log_price
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
0.71	Premium	E	SI1	60.5	55	2756	5.79	5.74	3.49	7.921536
0.71	Premium	F	SI1	59.8	62	2756	5.74	5.73	3.43	7.921536
0.70	Very Good	E	VS2	60.5	59	2757	5.71	5.76	3.47	7.921898
0.70	Very Good	E	VS2	61.2	59	2757	5.69	5.72	3.49	7.921898
0.72	Premium	D	SI1	62.7	59	2757	5.69	5.73	3.58	7.921898
0.72	Ideal	D	SI1	60.8	57	2757	5.75	5.76	3.50	7.921898
0.72	Good	D	SI1	63.1	55	2757	5.69	5.75	3.61	7.921898
0.70	Very Good	D	SI1	62.8	60	2757	5.66	5.68	3.56	7.921898
0.86	Premium	H	SI2	61.0	58	2757	6.15	6.12	3.74	7.921898
0.75	Ideal	D	SI2	62.2	55	2757	5.83	5.87	3.64	7.921898

In [28]:
```
diamonds_fit <- linear_reg() |>
  fit(log_price ~ log_carat, data = diamonds)
```

fit(<y> ~ <x>, data =)

In [29]:
```
diamonds_fit
```

```
parsnip model object


Call:
stats::lm(formula = log_price ~ log_carat, data = data)

Coefficients:
(Intercept)    log_carat
      8.449        1.676
```

log(price) = 8.449 + 1.676 × log(carat)

---

1. `augment()` : a function calculating predicted value and residual by dataset, `diamonds_fit` and `diamonds`
2. Exponentiate the residuals to put them back on the scale of raw prices.
   - To express residuals in actual diamond price units.
   - `.resid = exp(.resid)` : Generate the column `.resid`
     - Convert previous `.resid` value to exponential function

Example

log_carat	log_price	.fitted	.resid
1.1	7.8	7.7	1.01
1.2	8.1	8.0	0.99
1.3	8.5	8.4	1.02

- `.fitted` : model predicted log_price
- `.resid` : By exponentiating the residual, return to the original scale.

In [30]:
```
diamonds_aug <- augment(diamonds_fit, new_data = diamonds) |>
  mutate(.resid = exp(.resid))
```

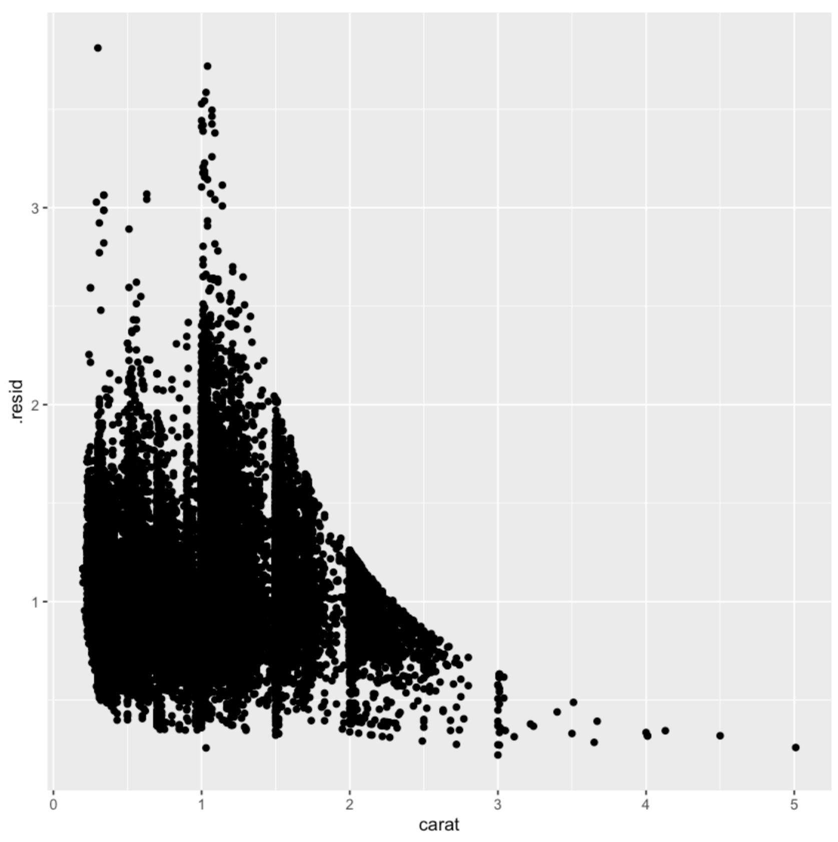
In [31]:
```
head(diamonds_aug)
```

A tibble: 6 × 14

.pred	.resid	carat	cut	color	clarity	depth	table	price	x	
<dbl>	<dbl>	<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl
5.985753	0.8196681	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.9
5.833301	0.9546565	0.21	Premium	E	SI1	59.8	61	326	3.89	3.8
5.985753	0.8221824	0.23	Good	E	VS1	56.9	65	327	4.05	4.0
6.374210	0.5694586	0.29	Premium	I	VS2	62.4	58	334	4.20	4.2
6.485973	0.5107668	0.31	Good	J	SI2	63.3	58	335	4.34	4.3
6.057075	0.7866561	0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.9

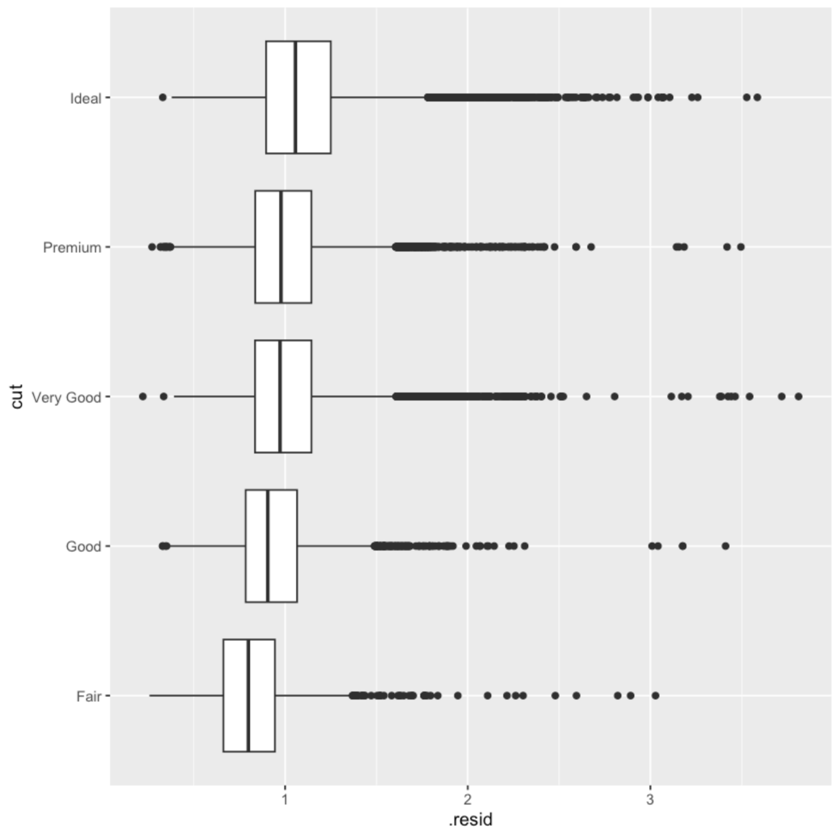
Residuals only deal with the values remaining after removing the predicted relationships (the part explained by the model).

In [32]:
```
ggplot(diamonds_aug) +
  geom_point(aes(x = carat, y = .resid))
```



- After removing the relationship between carat and price, we can see the **relationship is relative to their size**.

In [33]:
```
ggplot(diamonds_aug) +
    geom_boxplot(aes(x = cut, y = .resid)) +
    coord_flip()
```



After removing the relationship between carat and price, we can see that **better quality diamonds are more expensive.**