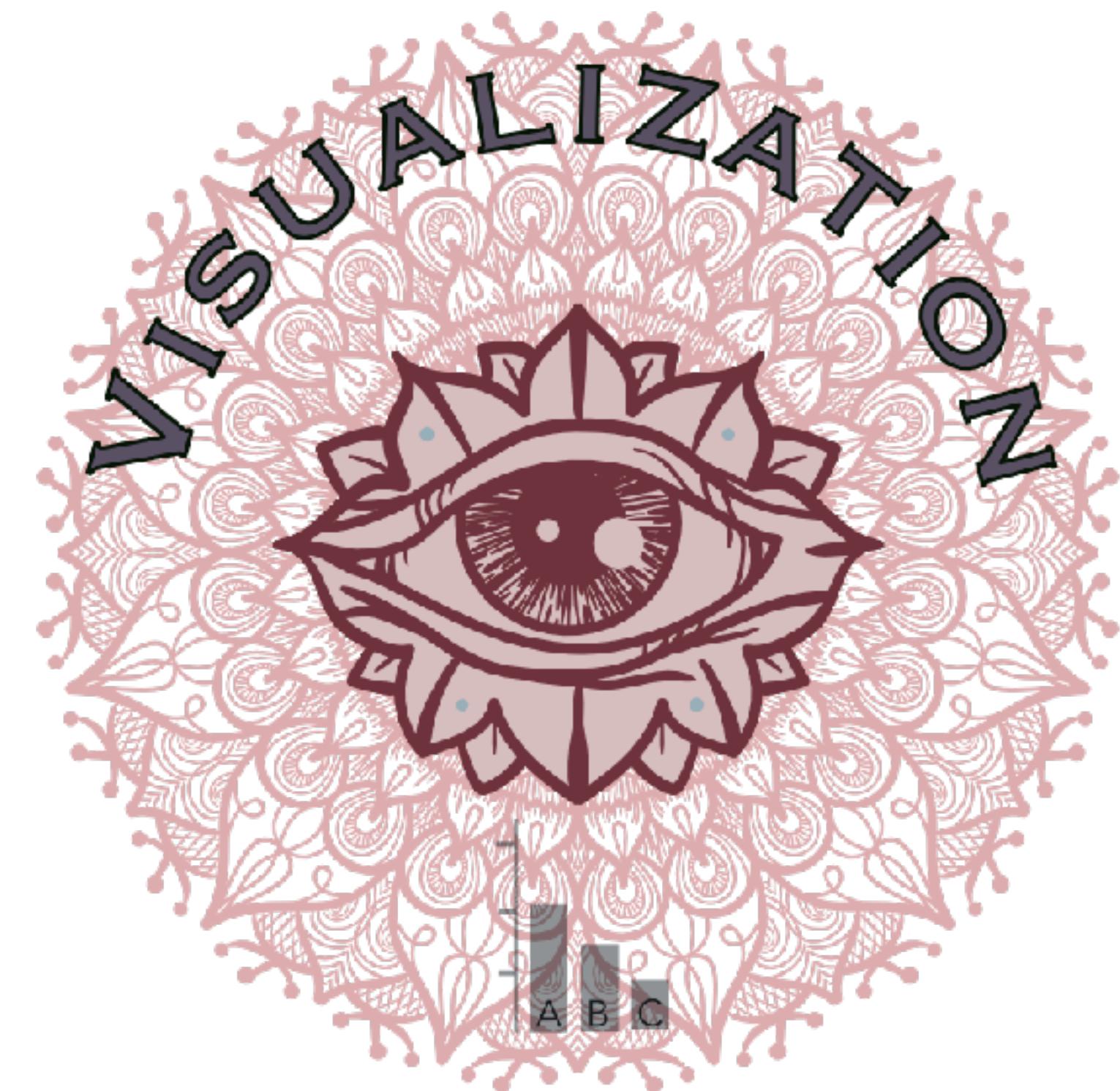


INTRODUCTION TO DATA ANALYSIS

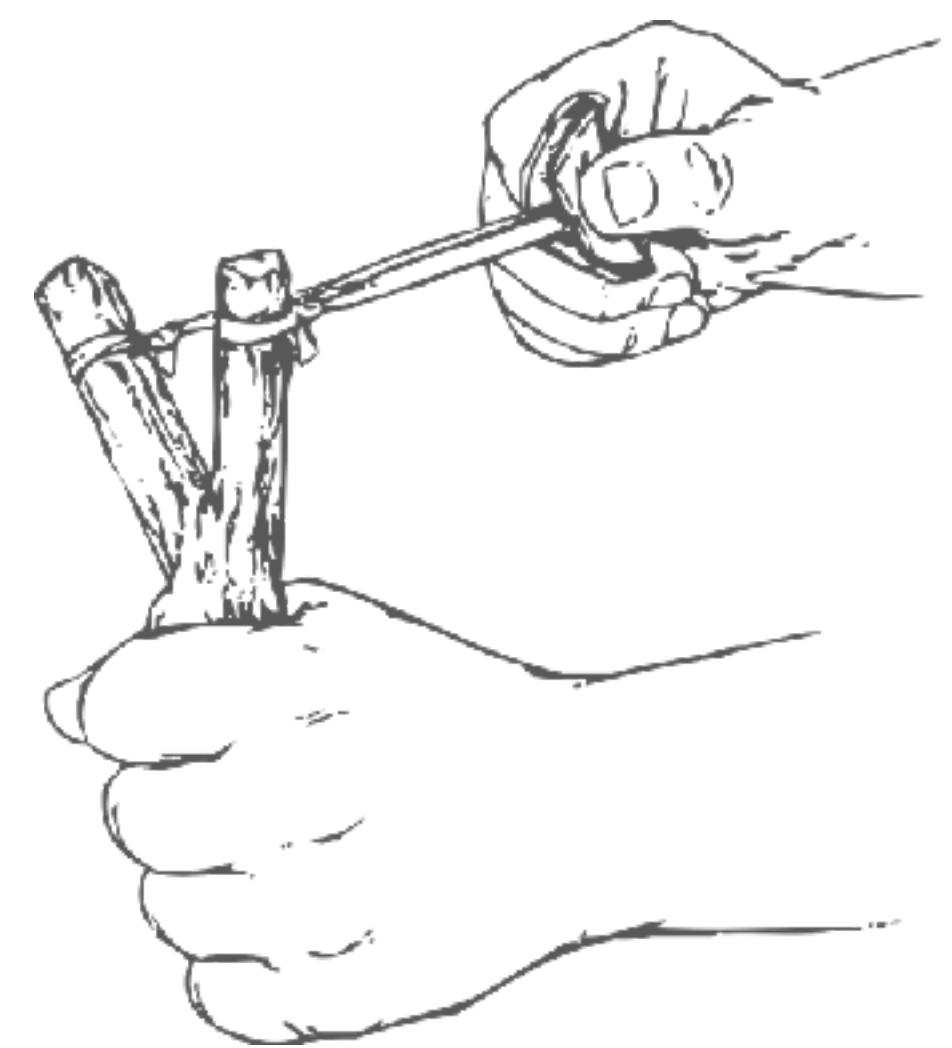
---

# DATA VISUALIZATION



### LEARNING GOALS

- ▶ obtain a basic understanding of better/worse plotting
  - ▶ understand the idea of **hypothesis-driven visualization**
- ▶ develop a basic understanding of the '**grammar of graphs**'
- ▶ get familiar with frequent visualization strategies
  - ▶ barplots, densities, violins, error bars etc.
- ▶ be able to fine-tune graphs for better visualization





# Motivation

### WHY VISUALIZE?

- ▶ a picture can be worth a million words (and numbers)
- ▶ every data analysis should start with a 'getting to know the data' phase
  - ▶ visualization of different aspects of data is key to get intimate with the data
- ▶ data visualization as a means of communication (with others)
  - ▶ **hypothesis-driven visualization**: obtain visual (suggestive) evidence regarding a research question of relevance

# WHY VISUALIZE?

- ▶ a picture can be worth a million words (and numbers)
- ▶ **summary statistics can be misleading** (because of information loss)
- ▶ every data analysis should start with a 'getting to know the data' phase
  - ▶ use extensive visualization to **get intimate with the data**
- ▶ data visualization as a means of communication (with others / with yourself)
  - ▶ **hypothesis-driven visualization**: obtain visual (suggestive) evidence regarding a research question of relevance

# BEYOND SUMMARY STATISTICS



# MOTIVATING EXAMPLE :: ANSCOMBE'S QUARTET

- ▶ famous data set, ships with core R

```
glimpse(anscombe %>% as_tibble)

## Observations: 11
## Variables: 8
## $ x1 <dbl> 10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5
## $ x2 <dbl> 10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5
## $ x3 <dbl> 10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5
## $ x4 <dbl> 8, 8, 8, 8, 8, 8, 19, 8, 8, 8
## $ y1 <dbl> 8.04, 6.95, 7.58, 8.81, 8.33, 9.96, .
## $ y2 <dbl> 9.14, 8.14, 8.74, 8.77, 9.26, 8.10, .
## $ y3 <dbl> 7.46, 6.77, 12.74, 7.11, 7.81, 8.84, .
## $ y4 <dbl> 6.58, 5.76, 7.71, 8.84, 8.47, 7.04, .
```

messy start

```
tidy_anscombe <- anscombe %>% as_tibble %>%
  pivot_longer(
    ## we want to pivot every column
    everything(),
    ## use reg-exps to capture 1st and 2nd character
    names_pattern = "(.)(.)",
    ## assign names to new cols, using 1st part of
    ## what reg-exp captures as new column names
    names_to = c(".value", "grp")
  ) %>%
  mutate(grp = paste0("Group ", grp))
tidy_anscombe
```

tidy up

```
## # A tibble: 44 x 3
##   grp      x     y
##   <chr>  <dbl> <dbl>
## 1 Group 1  10  8.04
## 2 Group 2  10  9.14
## 3 Group 3  10  7.46
## 4 Group 4   8  6.58
## 5 Group 1   8  6.95
## 6 Group 2   8  8.14
## 7 Group 3   8  6.77
## 8 Group 4   8  5.76
## 9 Group 1  13  7.58
## 10 Group 2 13  8.74
## # ... with 34 more rows
```

nice!



# MOTIVATING EXAMPLE :: ANSCOMBE'S QUARTET

```
## # A tibble: 44 x 3
##   grp      x     y
##   <chr>  <dbl> <dbl>
## 1 Group 1 10  8.04
## 2 Group 2 10  9.14
## 3 Group 3 10  7.46
## 4 Group 4  8  6.58
## 5 Group 1  8  6.95
## 6 Group 2  8  8.14
## 7 Group 3  8  6.77
## 8 Group 4  8  5.76
## 9 Group 1 13  7.58
## 10 Group 2 13  8.74
## # ... with 34 more rows
```

```
tidy_anscombe %>%
  group_by(grp) %>%
  summarise(
    mean_x = mean(x),
    mean_y = mean(y),
    min_x = min(x),
    min_y = min(y),
    max_x = max(x),
    max_y = max(y),
    corrln = cor(x,y)
  )
```

```
## # A tibble: 4 x 8
##   grp      mean_x  mean_y  min_x  min_y  max_x  max_y corrln
##   <chr>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 Group 1  9.00  7.50  4.00  4.26  14.0  10.8  0.816
## 2 Group 2  9.00  7.50  4.00  3.10  14.0  9.26  0.816
## 3 Group 3  9.00  7.50  4.00  5.39  14.0  12.7  0.816
## 4 Group 4  9.00  7.50  8.00  5.25  19.0  12.5  0.817
```

input data

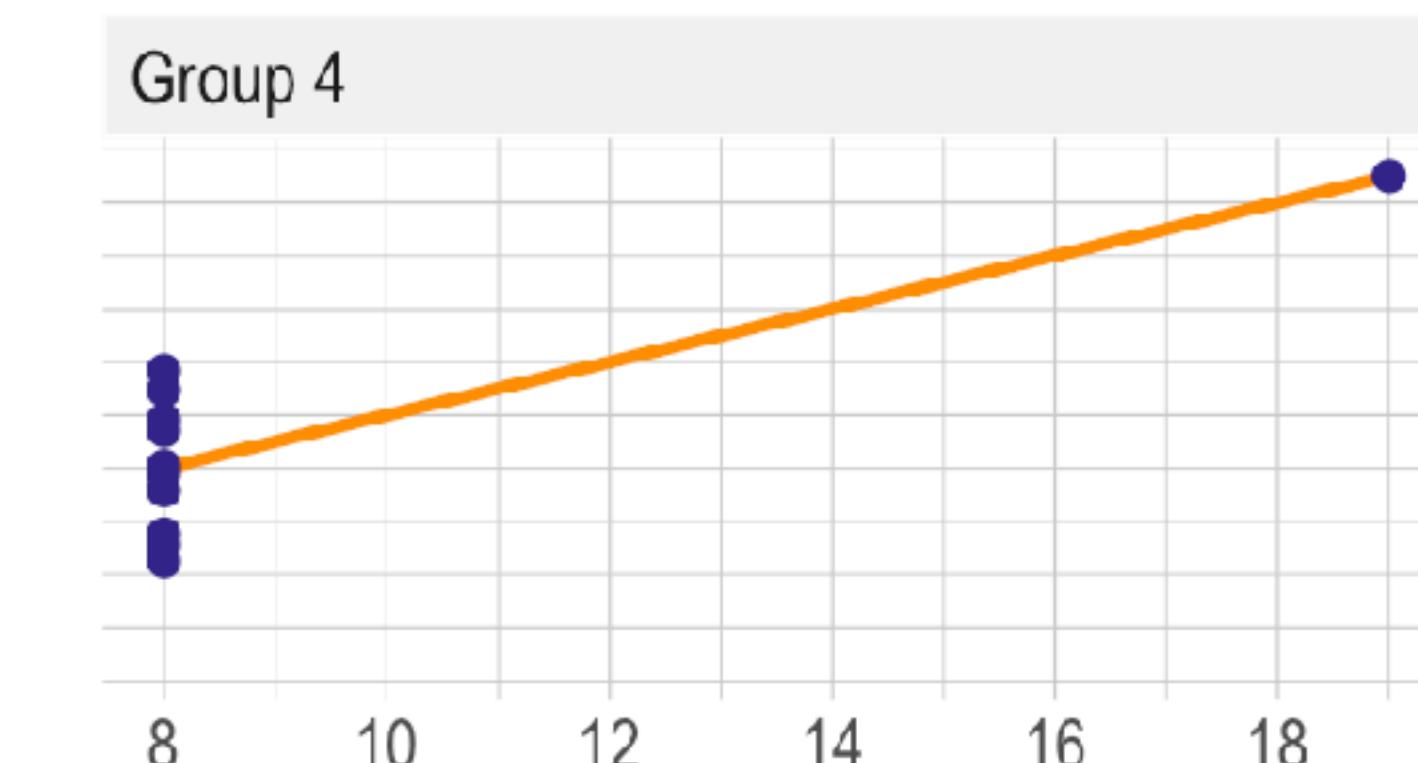
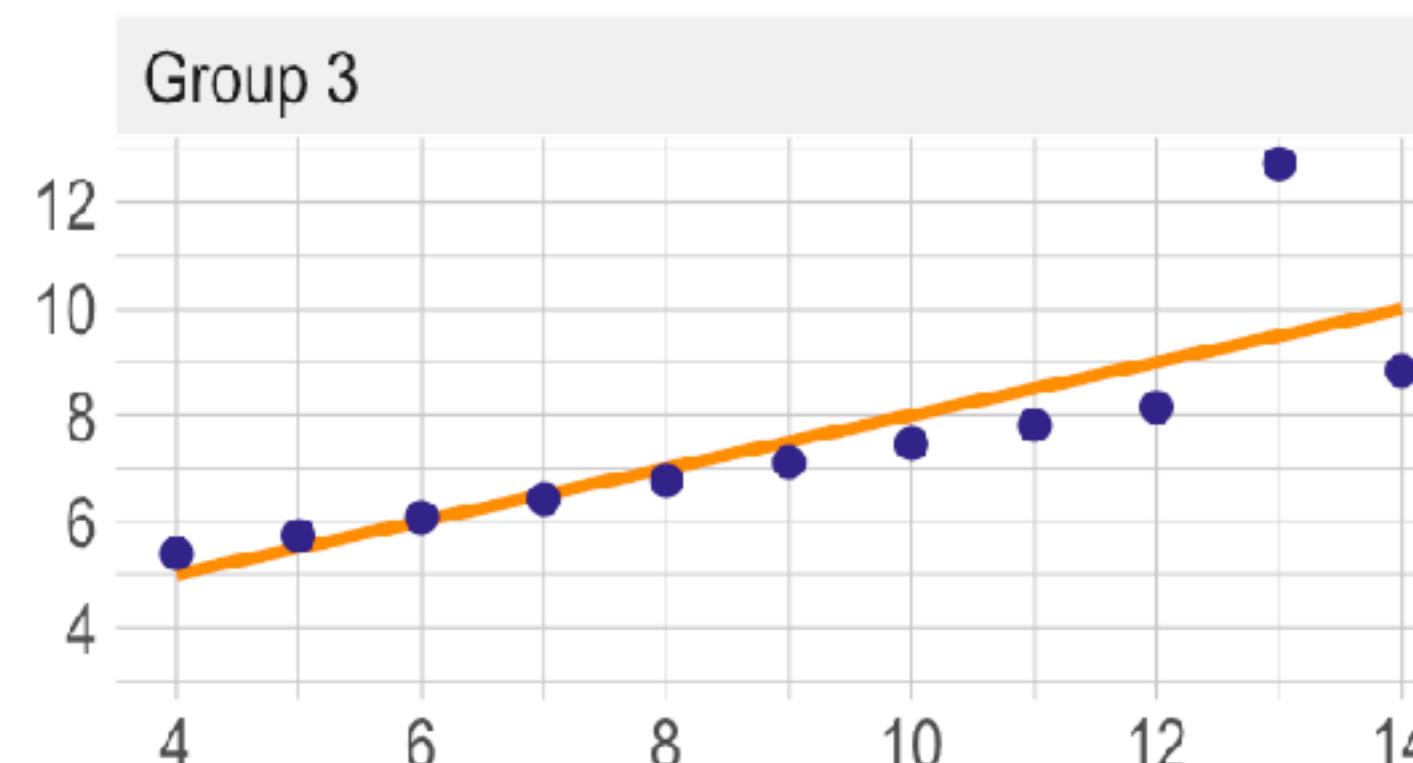
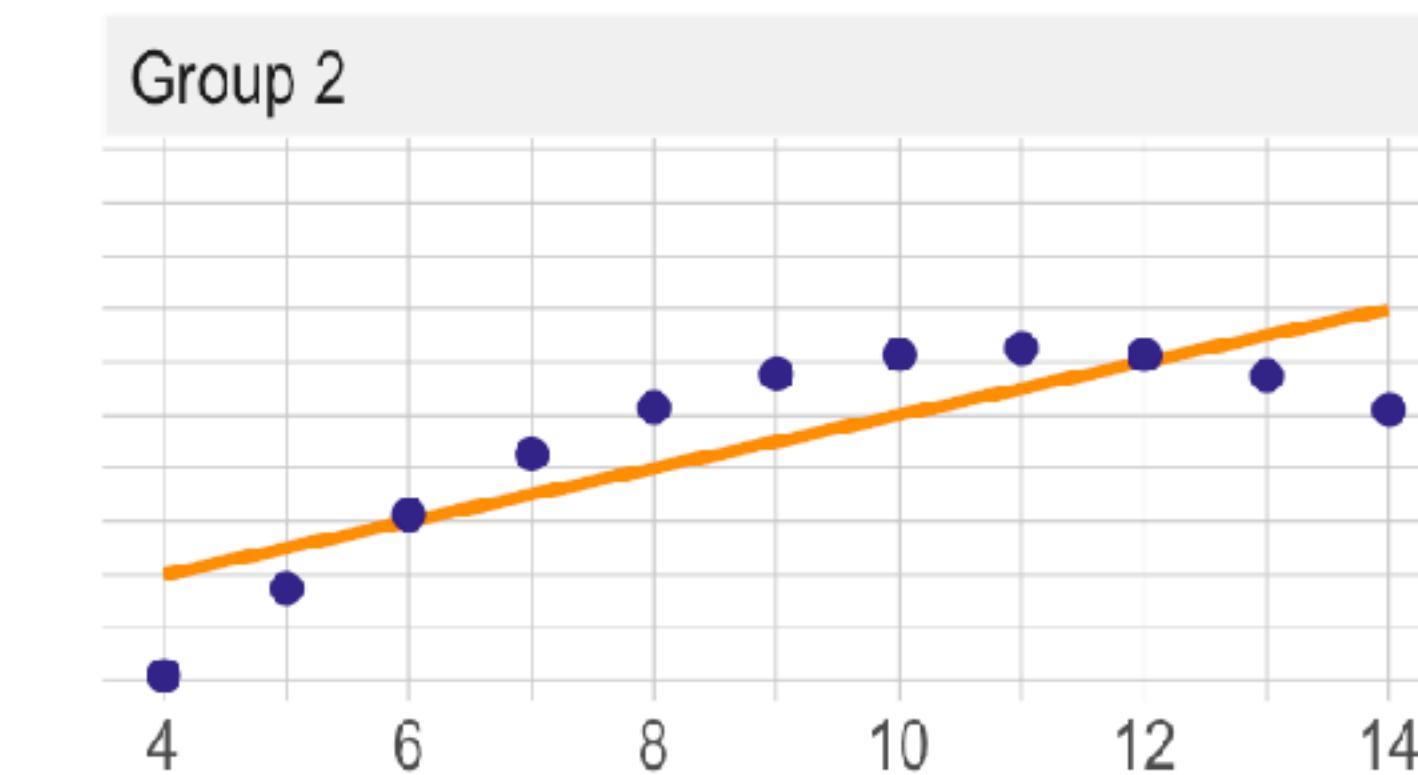
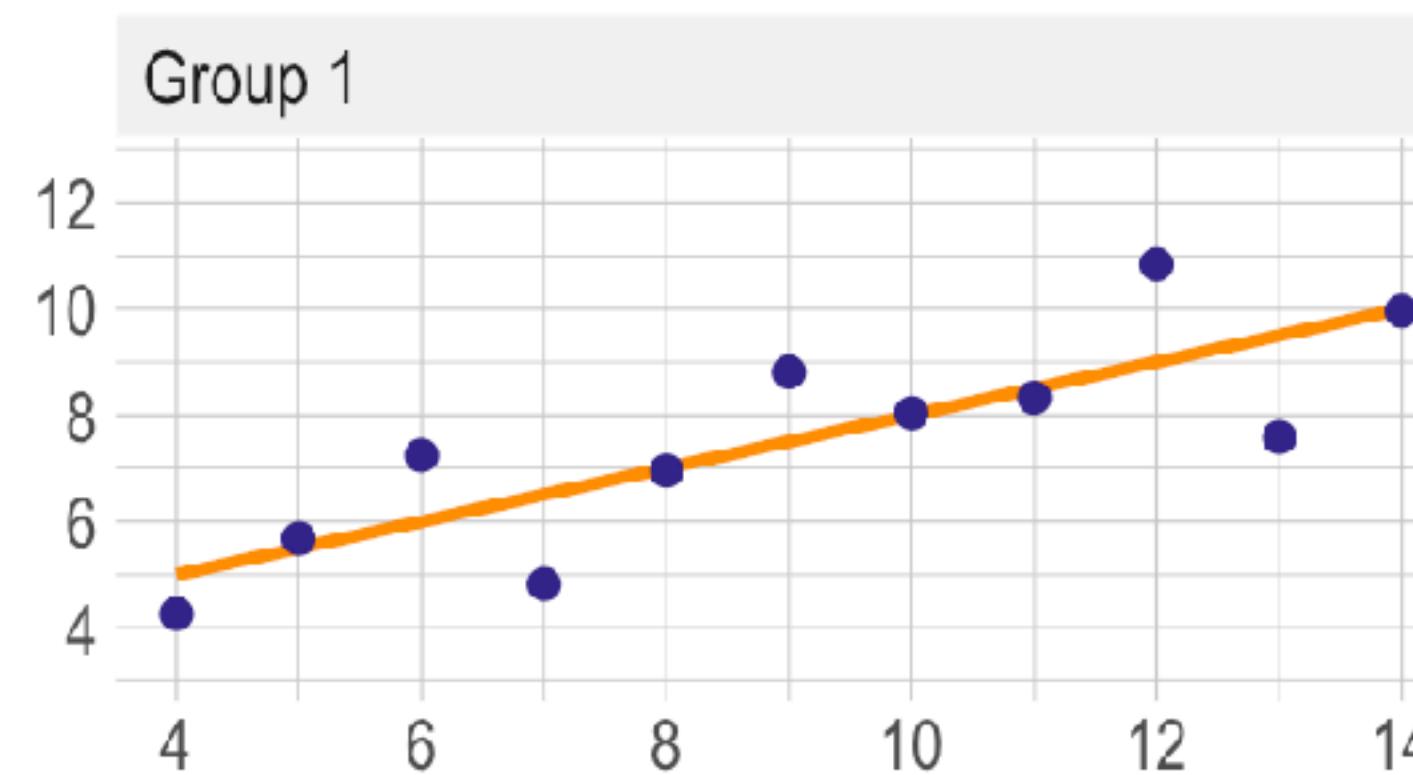
summarise

all four groups look very similar!

# MOTIVATING EXAMPLE :: ANSCOMBE'S QUARTET

- ▶ quite different patterns despite similar correlation

$$y = 0.5x + 3 \quad (R^2 \approx 0.82) \text{ for all datasets}$$





The good, the  
bad and the  
info-graphic

## PRINCIPLES OF GOOD VISUALIZATION

- ▶ maximize data-ink ratio (Tufte 1983)
  - ▶ maximize information, minimize ink
  - ▶ contra **chart junk**
  - ▶ ink vs. processing effort
- ▶ analogy to language
  - ▶ information flow
  - ▶ ease of processing
  - ▶ bound by conventional rules
- ▶ **hypothesis-driven visualization**
  - ▶ relevance of information

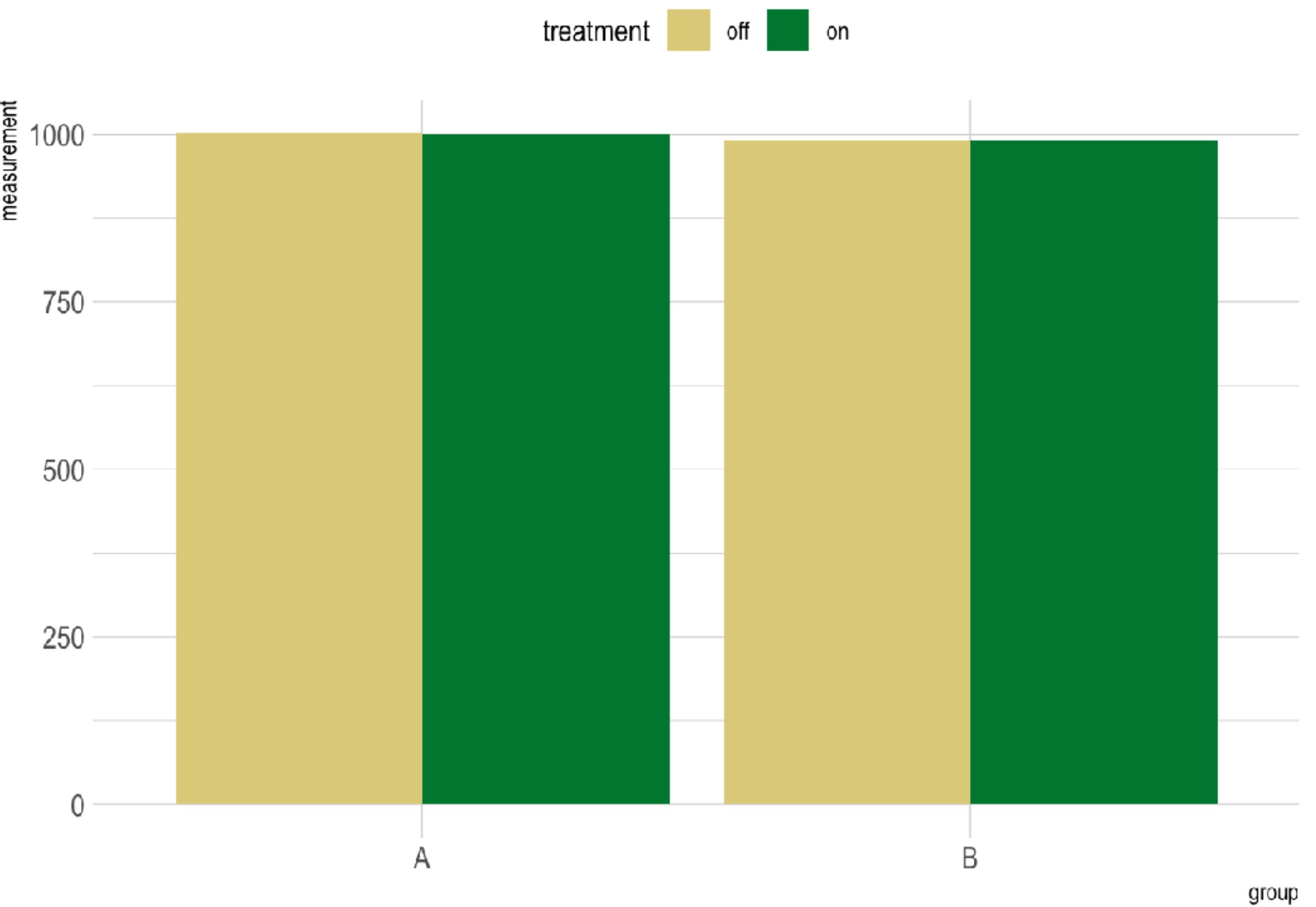


**The vague & defeasible rule of thumb of good data visualization (according to the author).**

“Communicate a maximal degree of relevant true information in a way that minimizes the recipient’s effort of retrieving this information.”

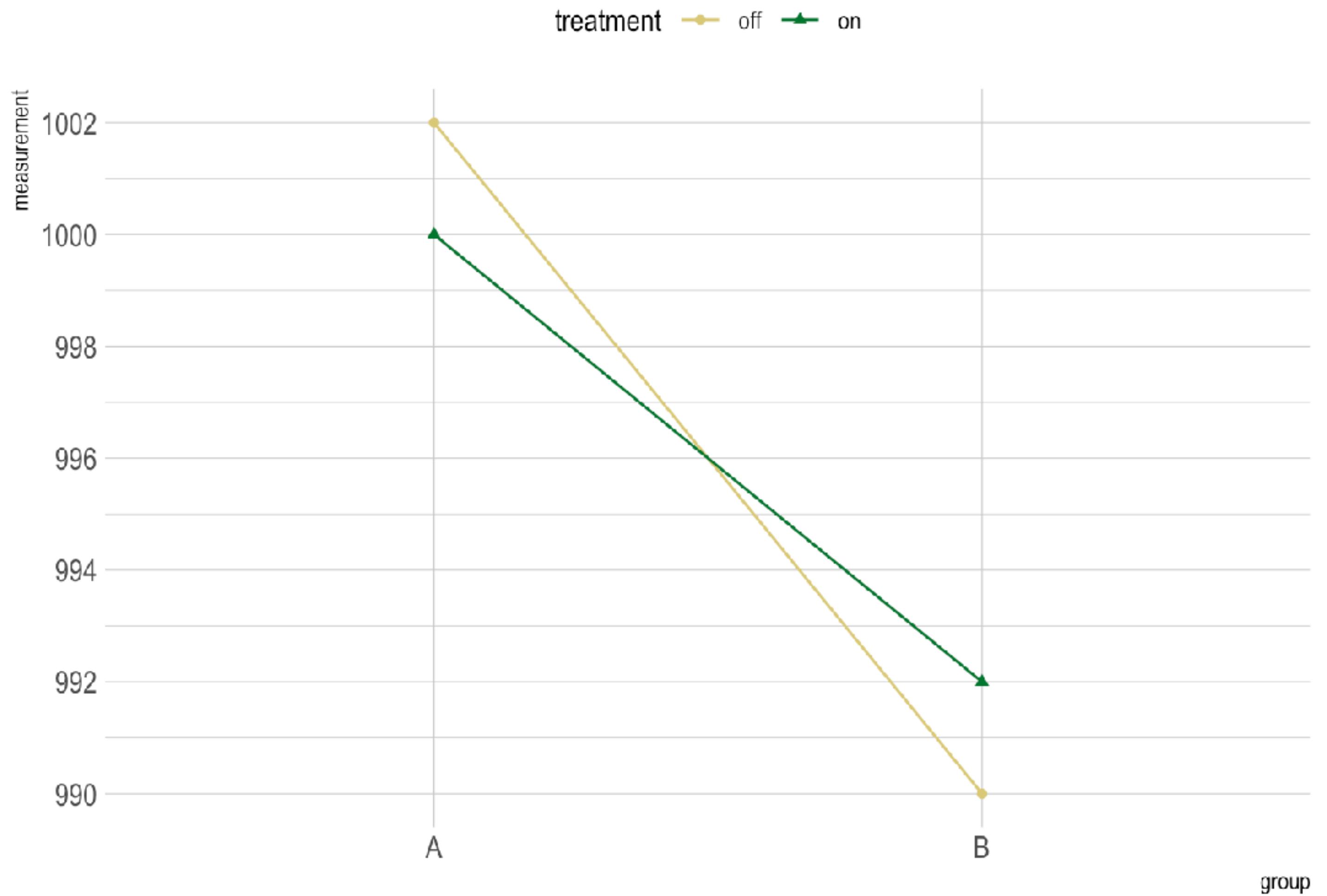
# EXAMPLE OF UNINFORMATIVE PLOTTING

```
large_contrast_data <- tribble(  
  ~group, ~treatment, ~measurement,  
  "A", "on", 1000,  
  "A", "off", 1002,  
  "B", "on", 992,  
  "B", "off", 990  
)
```



# EXAMPLE OF INFORMATIVE HYPOTHESIS-DRIVEN PLOTTING

```
large_contrast_data <- tribble(  
  ~group, ~treatment, ~measurement,  
  "A", "on", 1000,  
  "A", "off", 1002,  
  "B", "on", 992,  
  "B", "off", 990  
)
```

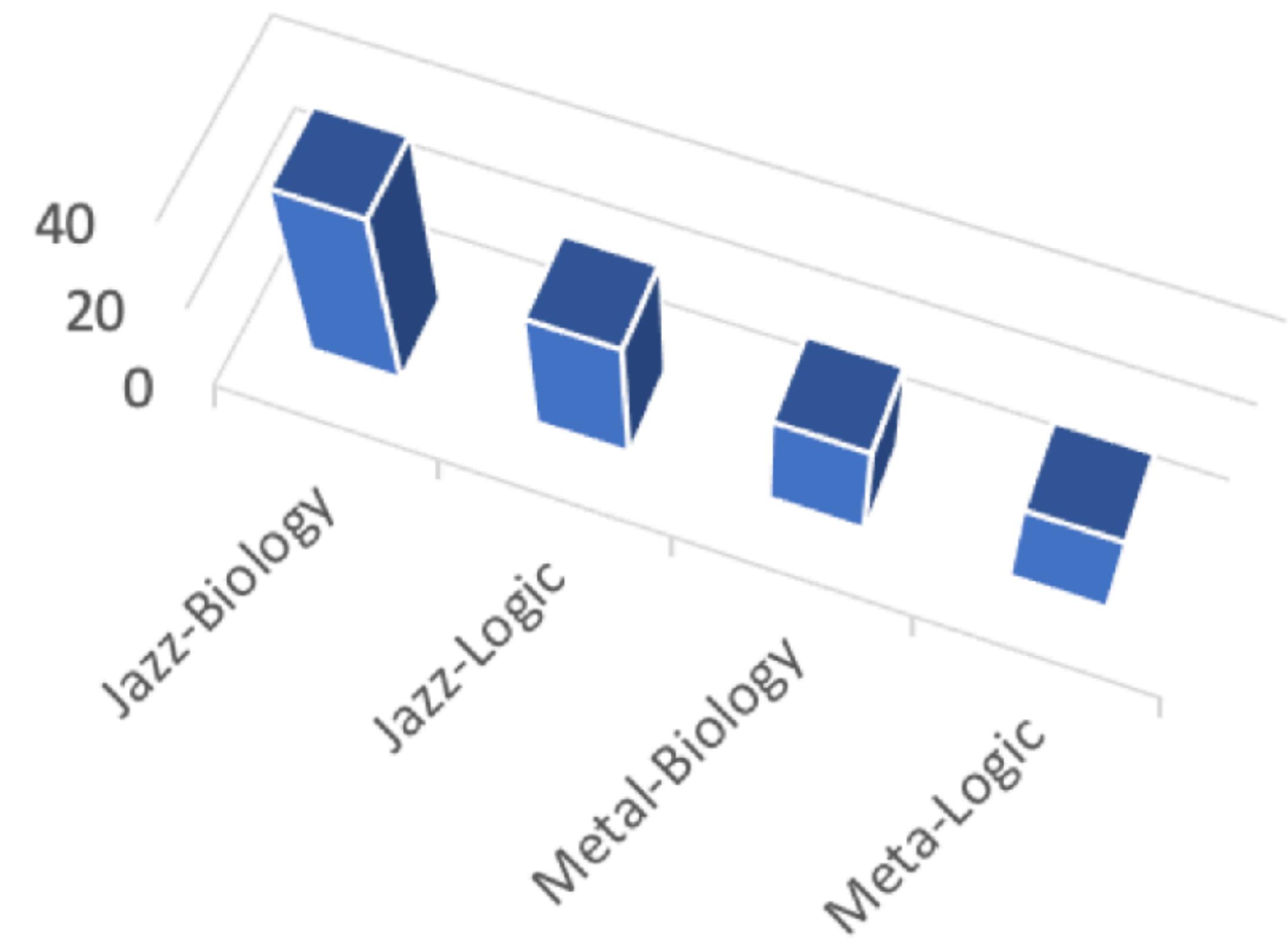




## EXAMPLE OF UNINFORMATIVE PLOTTING

Counts of music-subject choice pairs

```
## # A tibble: 4 x 3
##   JM     LB     n
##   <chr> <chr> <int>
## 1 Jazz   Biology 38
## 2 Jazz   Logic    26
## 3 Metal  Biology 20
## 4 Metal  Logic   18
```

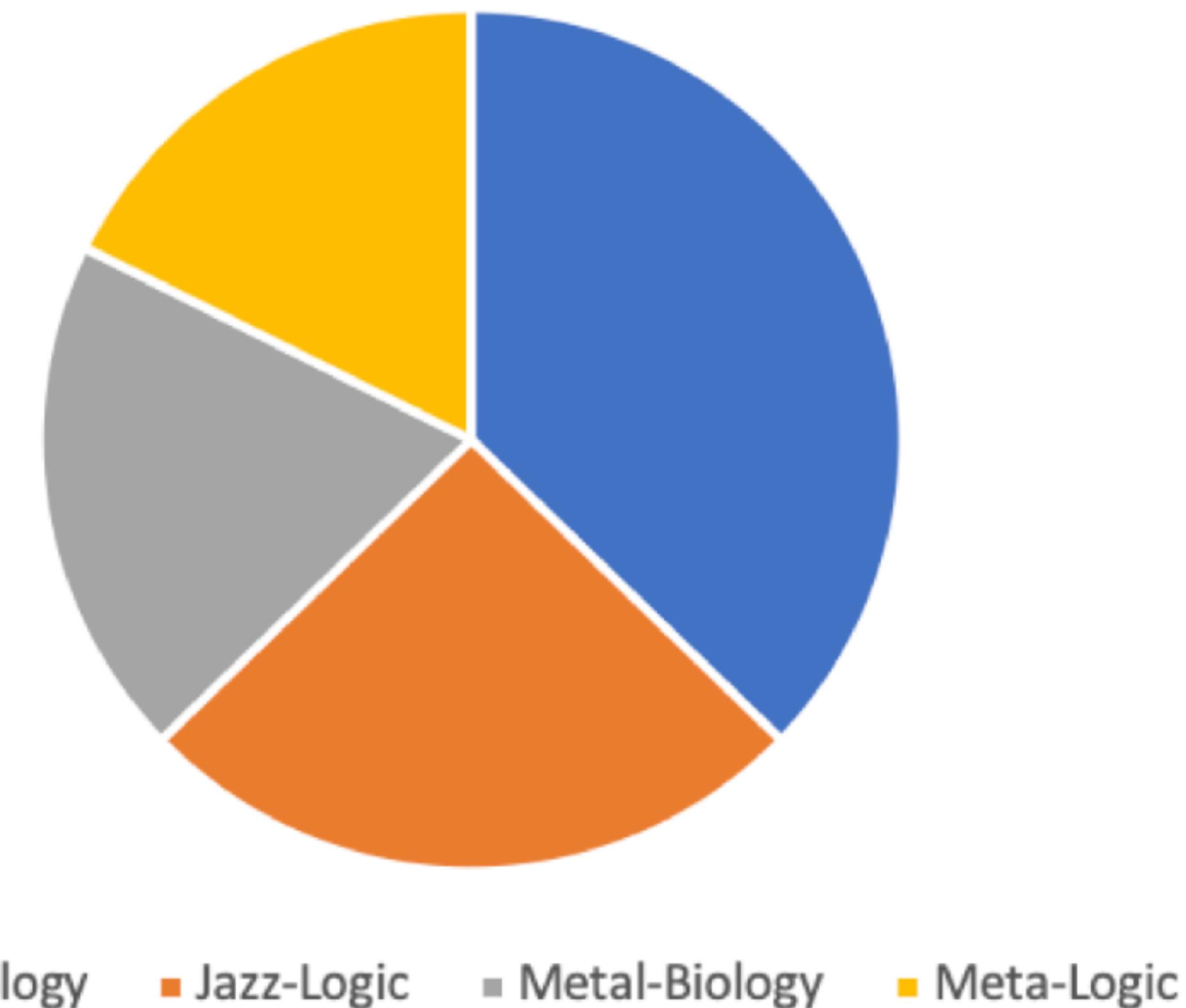


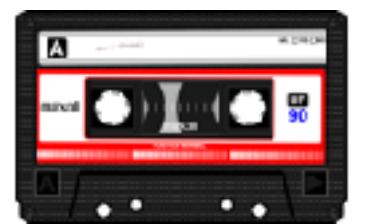


## EXAMPLE OF (STILL) UNINFORMATIVE PLOTTING

```
## # A tibble: 4 x 3
##   JM     LB     n
##   <chr> <chr> <int>
## 1 Jazz   Biology 38
## 2 Jazz   Logic    26
## 3 Metal  Biology 20
## 4 Metal  Logic   18
```

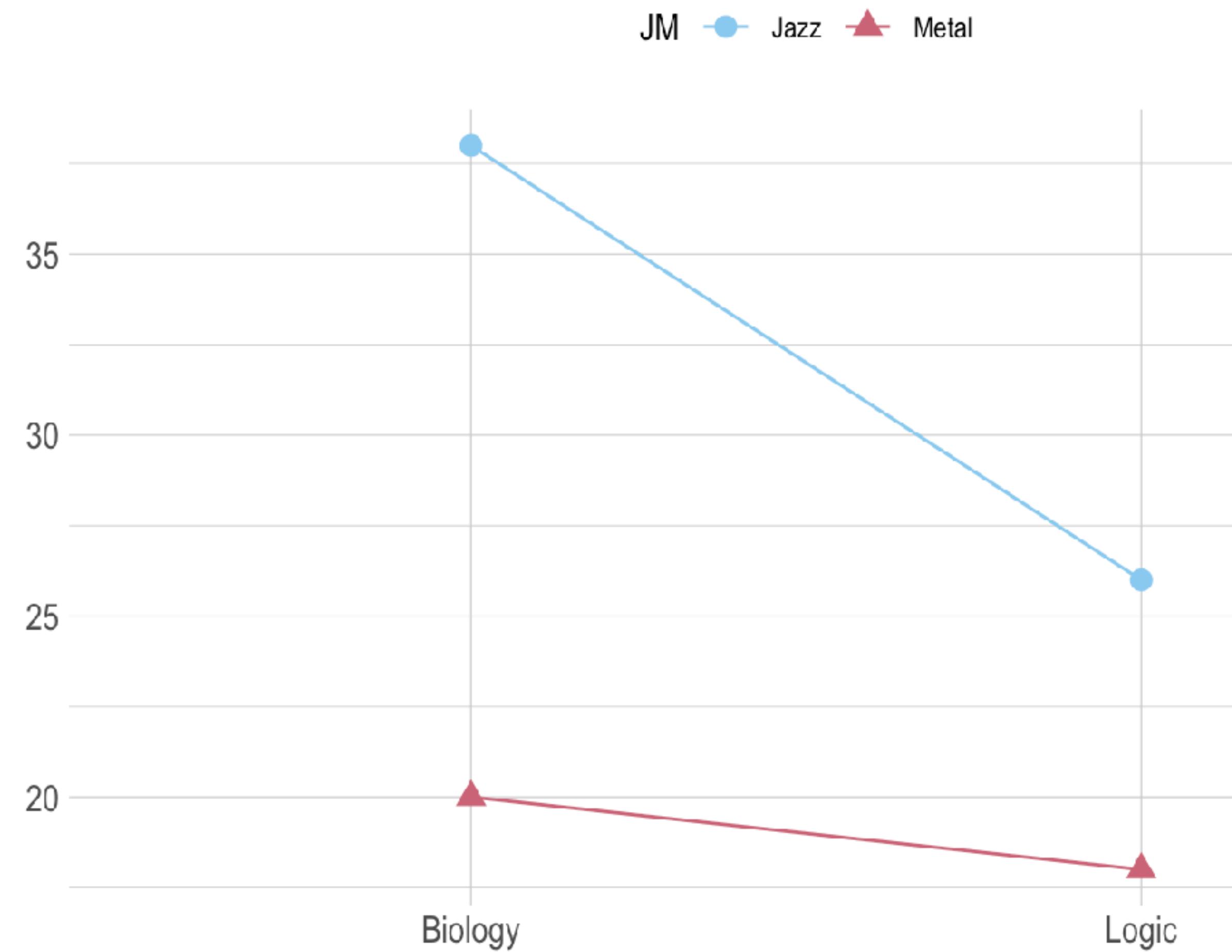
Proportions of music-subject choice pairs





## EXAMPLE OF INFORMATIVE HYPOTHESIS-DRIVEN PLOTTING

```
## # A tibble: 4 x 3
##   JM     LB     n
##   <chr> <chr> <int>
## 1 Jazz   Biology 38
## 2 Jazz   Logic   26
## 3 Metal  Biology 20
## 4 Metal  Logic   18
```



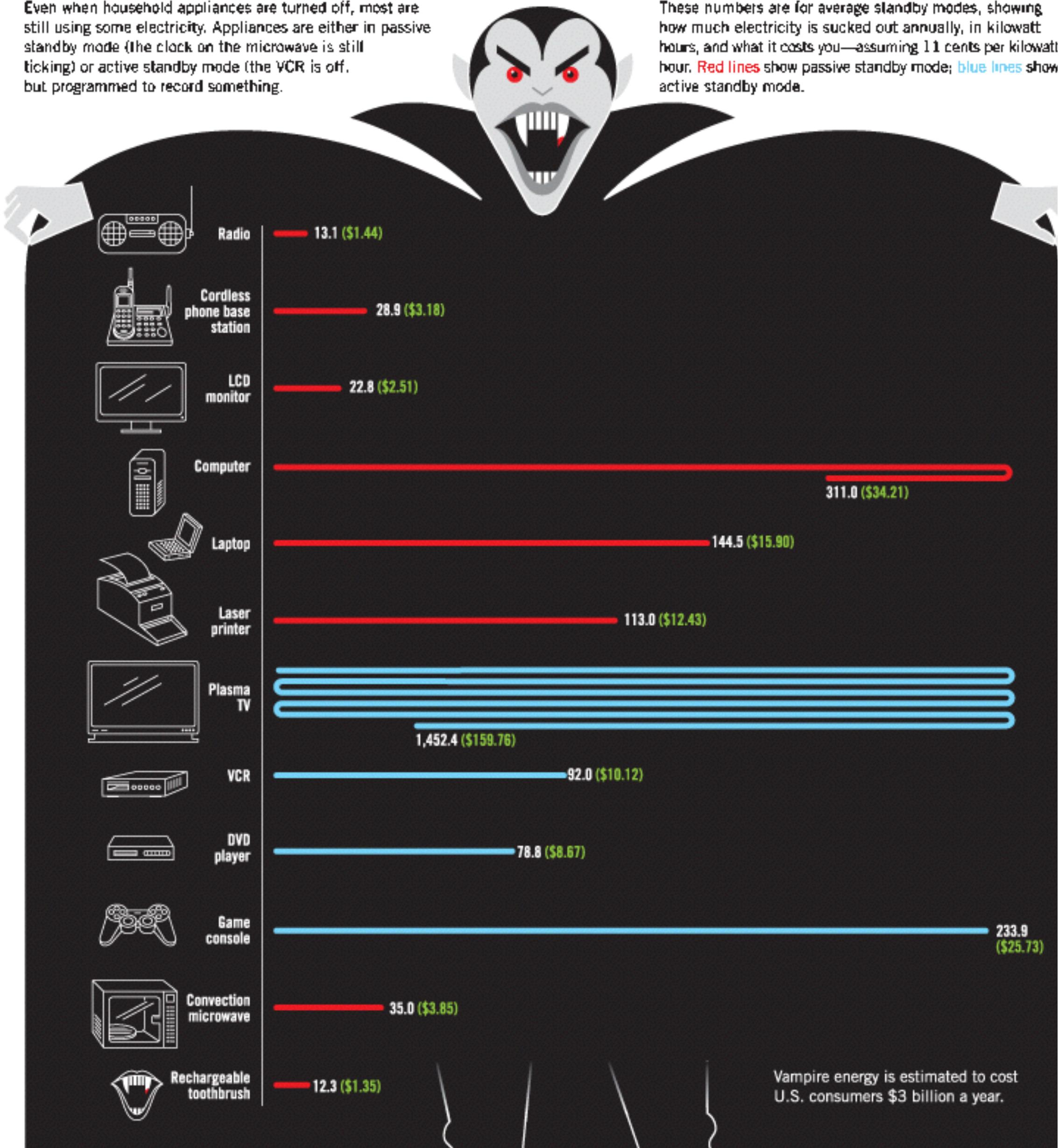
# INTRODUCTION TO DATA ANALYSIS

## INFOGRAPHICS

- ▶ ≠ hypothesis-driven visualization
- ▶ purposes:
  - ▶ memorability
  - ▶ eye-catchiness
  - ▶ persuasion
  - ▶ ...

Even when household appliances are turned off, most are still using some electricity. Appliances are either in passive standby mode (the clock on the microwave is still ticking) or active standby mode (the VCR is off, but programmed to record something).

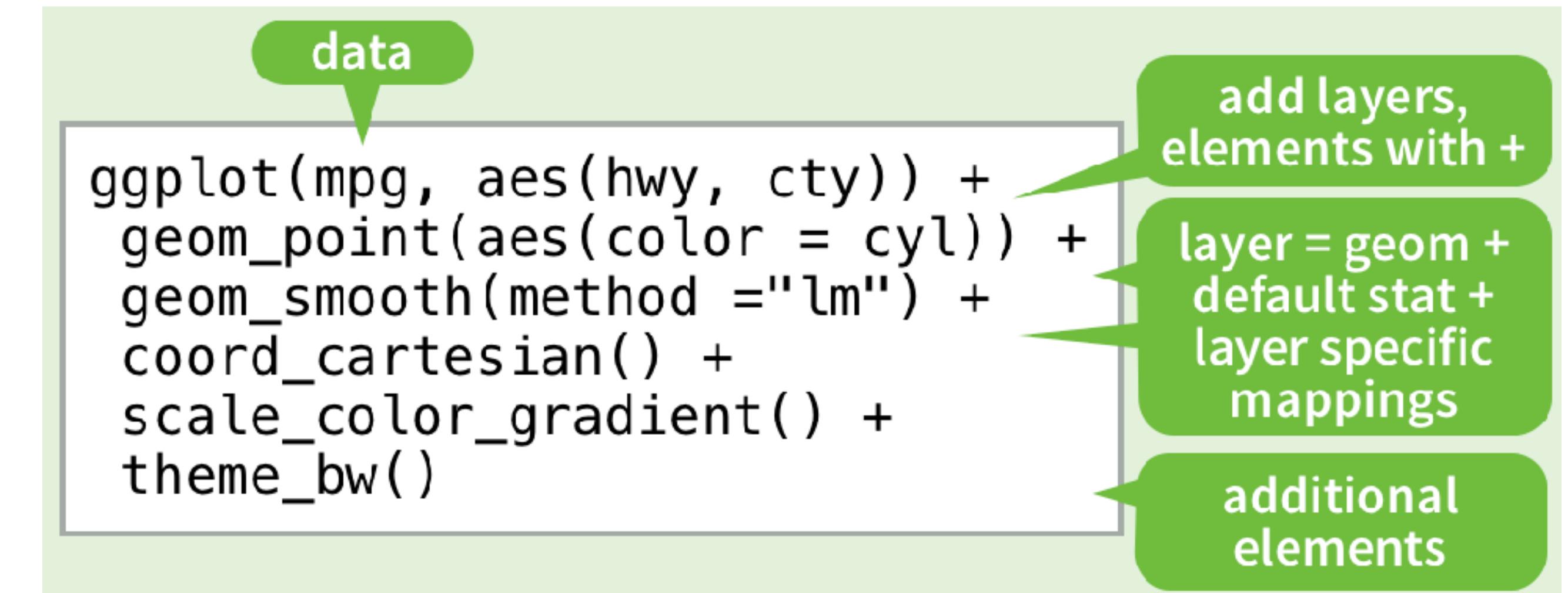
These numbers are for average standby modes, showing how much electricity is sucked out annually, in kilowatt hours, and what it costs you—assuming 11 cents per kilowatt hour. **Red lines** show passive standby mode; **blue lines** show active standby mode.



# Basics of ggplot

## BASICS OF GGPLOT

- ▶ “grammar of layered graphs”
- ▶ incremental composition
- ▶ layers
- ▶ system of rich convenience functions & defaults
- ▶ grouping
- ▶ multiple ways of customization



# INCREMENTAL COMPOSITION

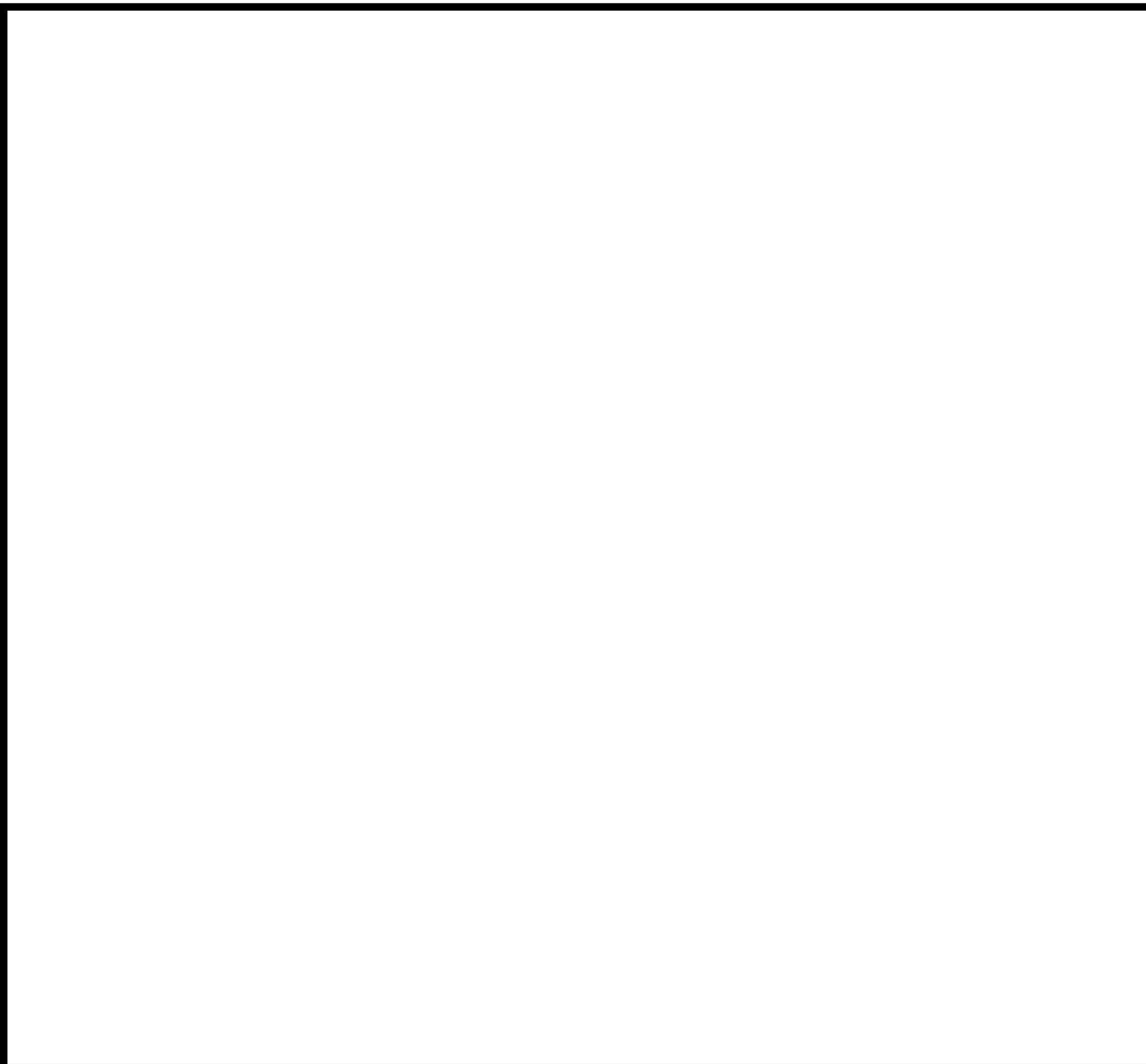
create a plot

```
incrementally_built_plot <- ggplot()
```

display the plot

```
incrementally_built_plot
```

output 😊

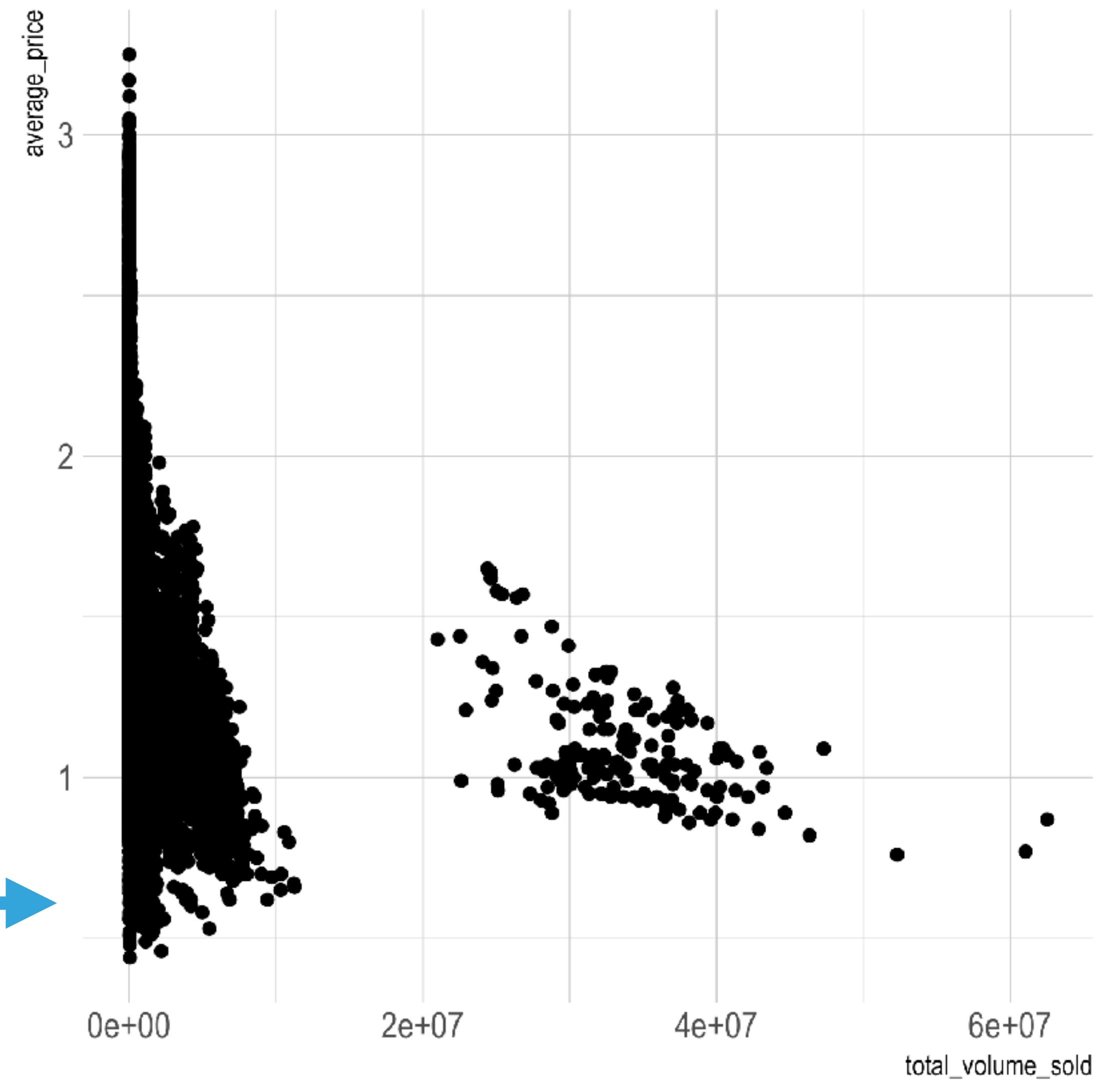


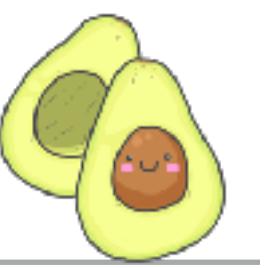


# INCREMENTAL COMPOSITION

```
incrementally_built_plot +  
  # add a geom of type `point` (=> scatter plot)  
  geom_point(  
    # what data to use  
    data = avocado_data,  
    # supply a mapping (in the form of an 'aesthetic' (see below))  
    mapping = aes(  
      # which variable to map onto the x-axis  
      x = total_volume_sold,  
      # which variable to map onto the y-axis  
      y = average_price  
    )  
  )
```

output →



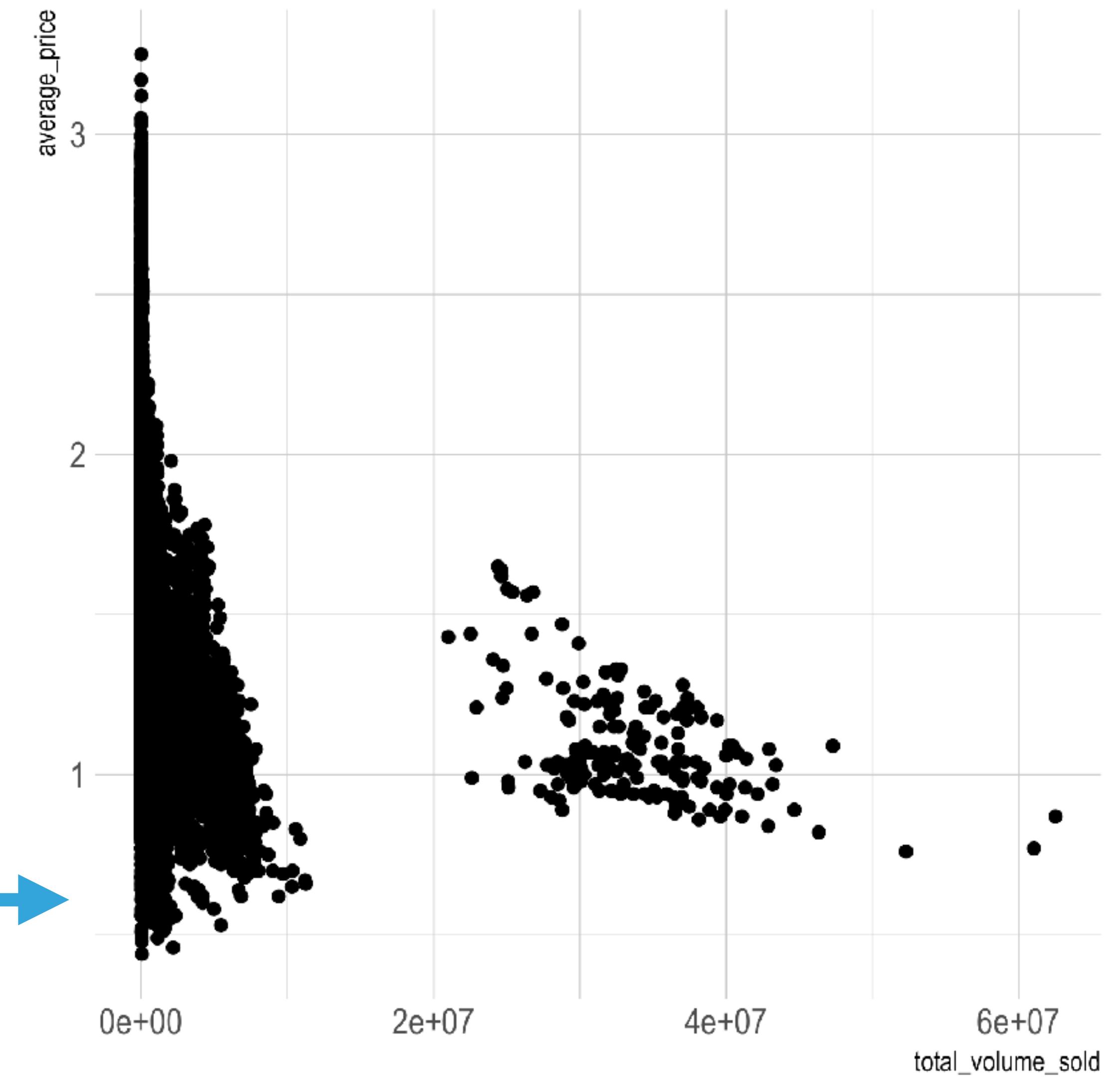


## INCREMENTAL COMPOSITION

- ▶ piping data into 1<sup>st</sup> argument slot
- ▶ declaring mapping globally for all subsequent calls to `geom\_` functions

```
avocado_data %>%  
  ggplot(aes(x = total_volume_sold, y = average_price)) +  
  geom_point()
```

output

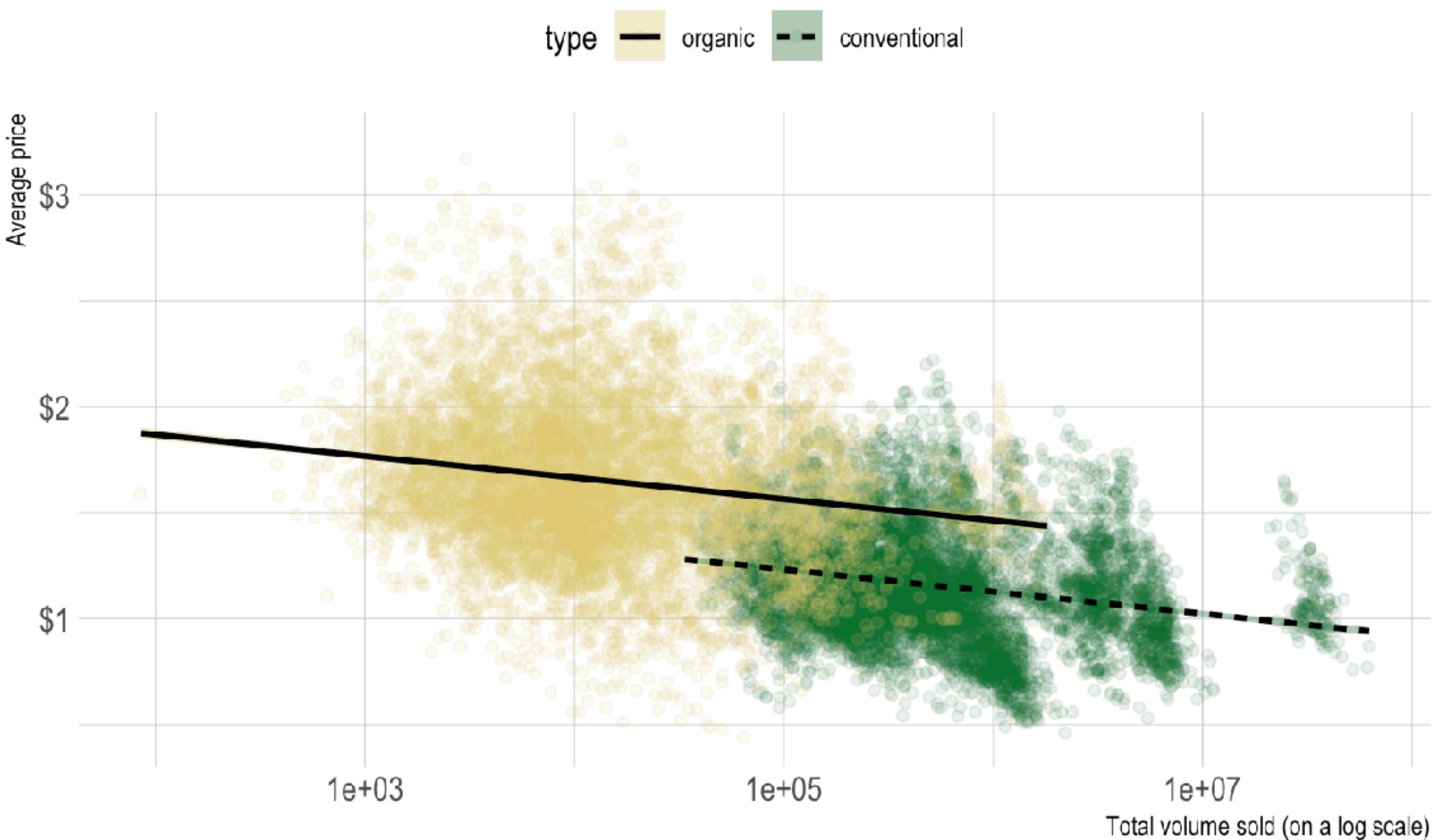


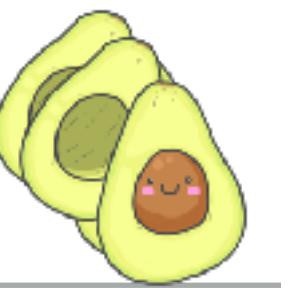


## FULL EXAMPLE

### Avocado prices plotted against the amount sold per type

With linear regression lines

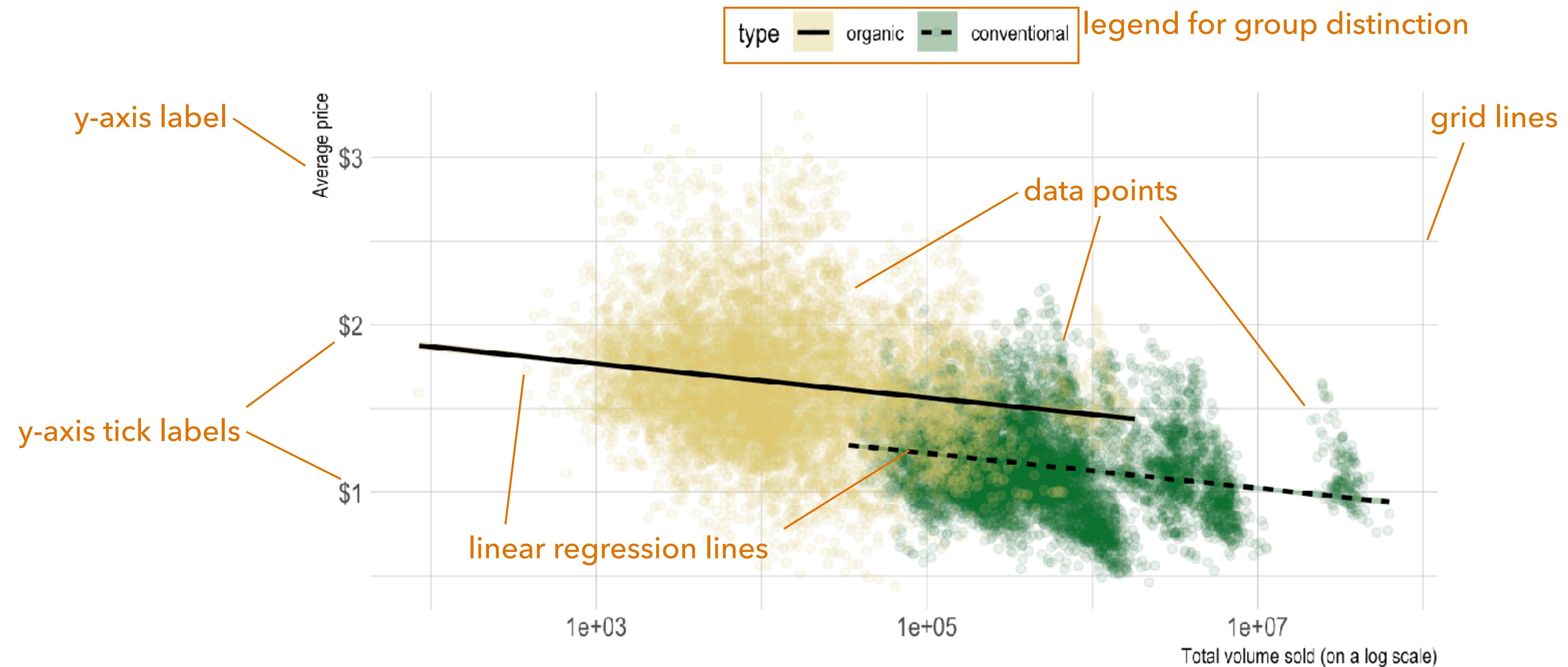


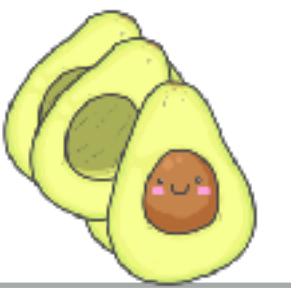


## FULL EXAMPLE

**title** Avocado prices plotted against the amount sold per type

**subtitle** With linear regression lines

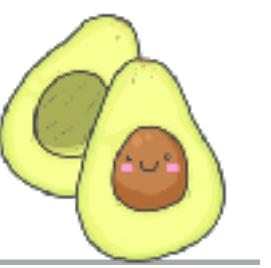




## FULL EXAMPLE :: CODE

```
avocado_data %>%
  # reverse factor level so that horizontal legend entries align with
  # the majority of observations of each group in the plot
  mutate(
    type = fct_rev(type)
  ) %>%
  # initialize the plot
  ggplot(
    # defined mapping
    mapping = aes(
      # which variable goes on the x-axis
      x = total_volume_sold,
      # which variable goes on the y-axis
      y = average_price,
      # which groups of variables to distinguish
      group = type,
      # color and fill to change by grouping variable
      fill = type,
      linetype = type,
      color = type
    )
  ) +
```

```
# declare that we want a scatter plot
  geom_point(
    # set low opacity for each point
    alpha = 0.1
  ) +
  # add a linear model fit (for each group)
  geom_smooth(
    color = "black",
    method = "lm"
  ) +
  # change the default (normal) of x-axis to log-scale
  scale_x_log10() +
  # add dollar signs to y-axis labels
  scale_y_continuous(labels = scales::dollar) +
  # change axis labels and plot title & subtitle
  labs(
    x = 'Total volume sold (on a log scale)',
    y = 'Average price',
    title = "Avocado prices plotted against the amount sold per type",
    subtitle = "With linear regression lines"
  )
```

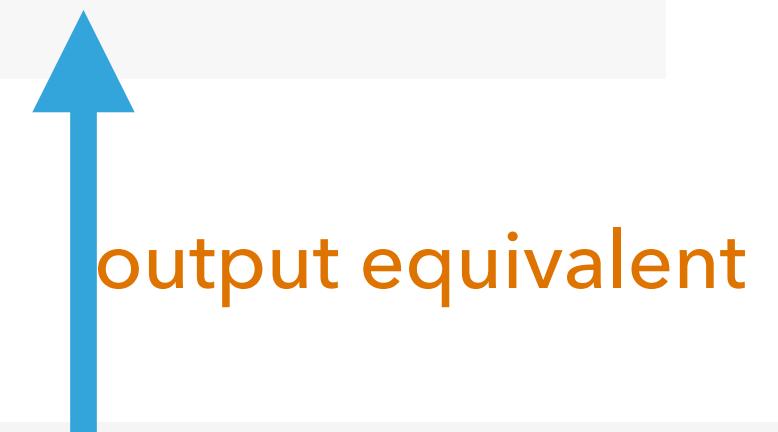


## LAYERED GRAMMAR OF GRAPHS

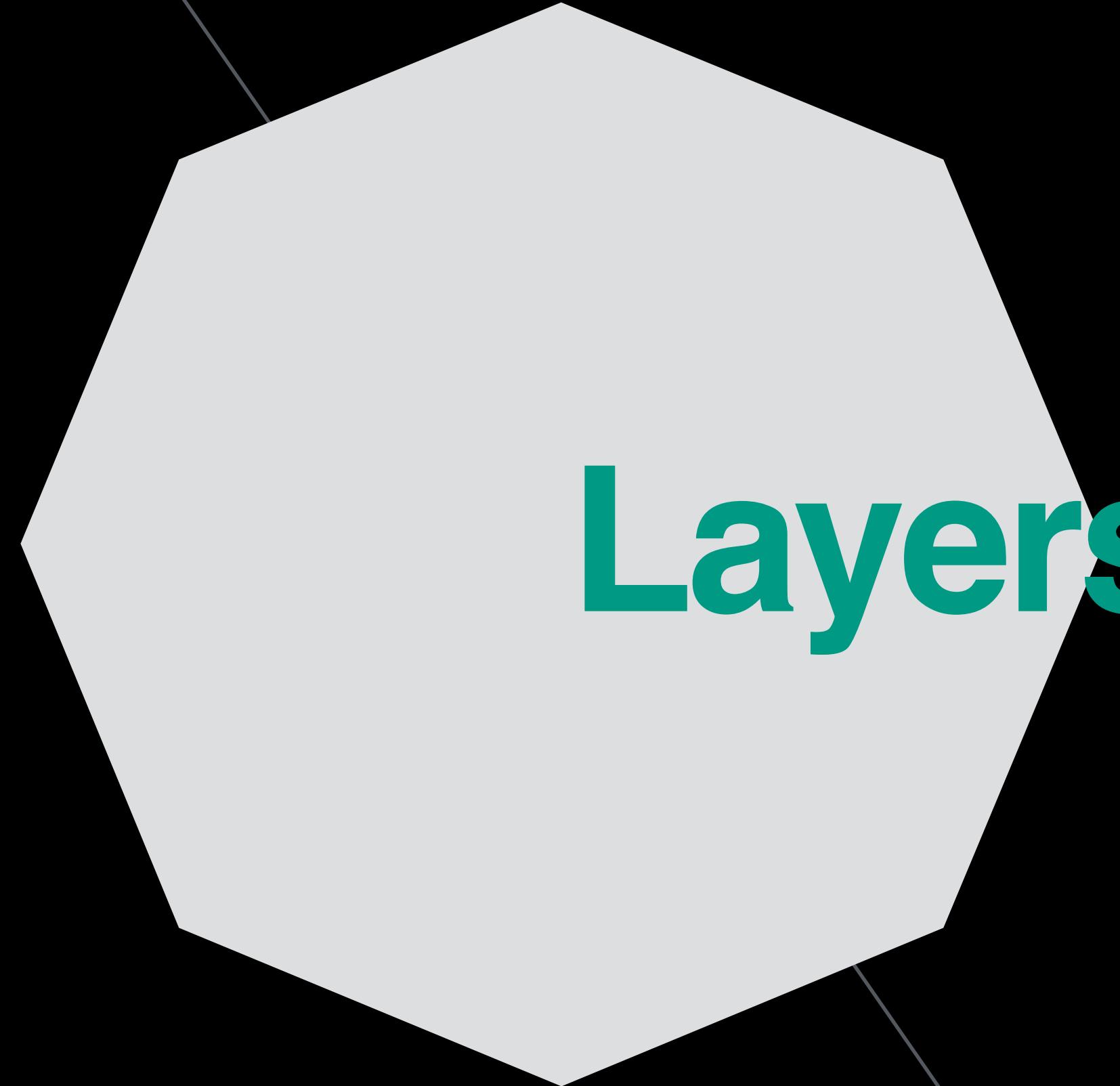
data -> statistical transformation ->  
geom. object -> aesthetics

- ▶ `geom\_` functions are wrappers
- ▶ default stat. transform, position, axis type etc.
- ▶ defaults can be overwritten

```
avocado_data %>%  
  ggplot(aes(x = total_volume_sold, y = average_price)) +  
  geom_point()
```



```
ggplot() +  
  layer(  
    data = avocado_data,  
    mapping = aes(x = total_volume_sold, y = average_price),  
    geom = "point",  
    stat = "identity",  
    position = "identity"  
  ) +  
  scale_x_continuous() +  
  scale_y_continuous() +  
  coord_cartesian()
```

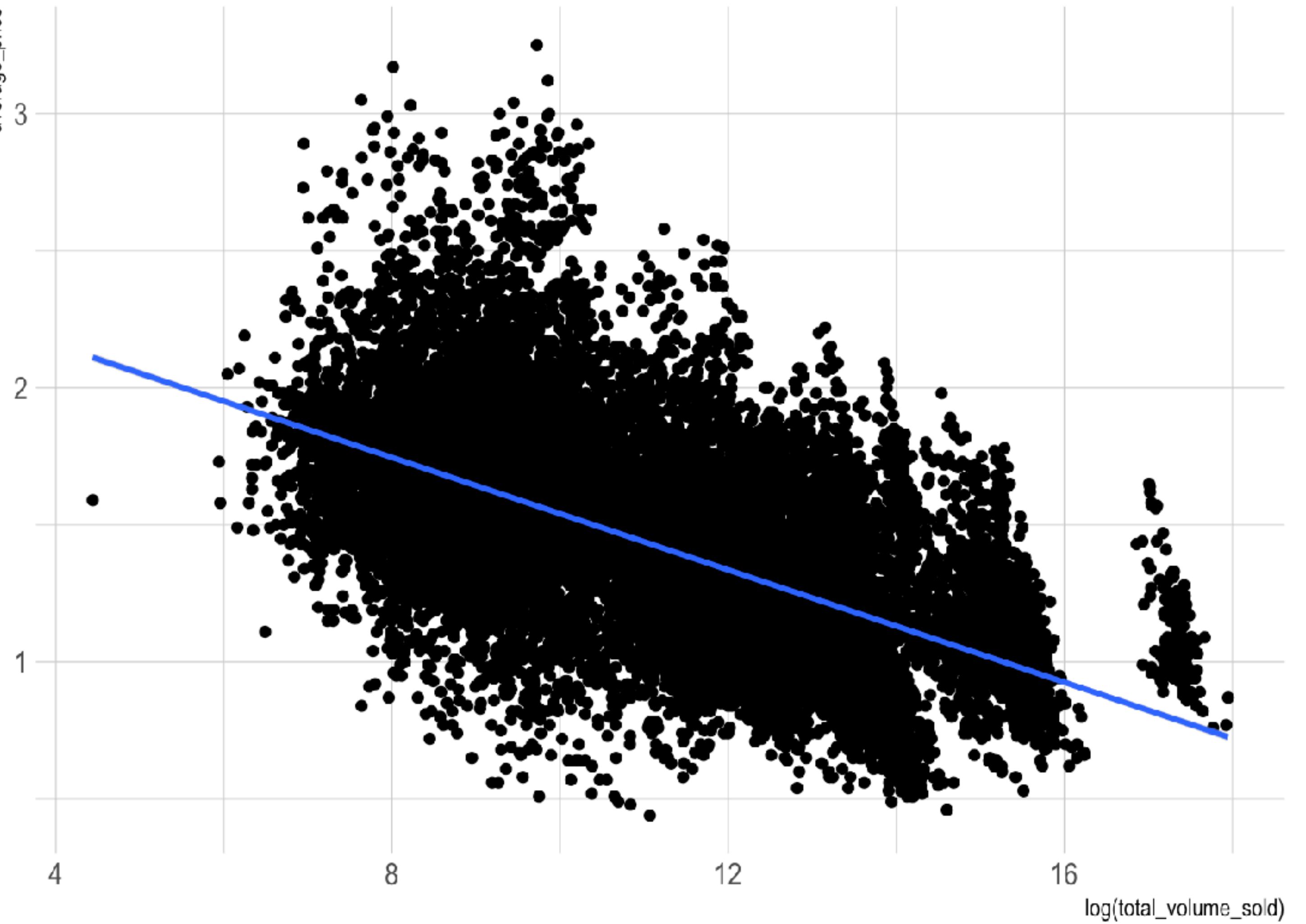


**Layers**



# LAYERS

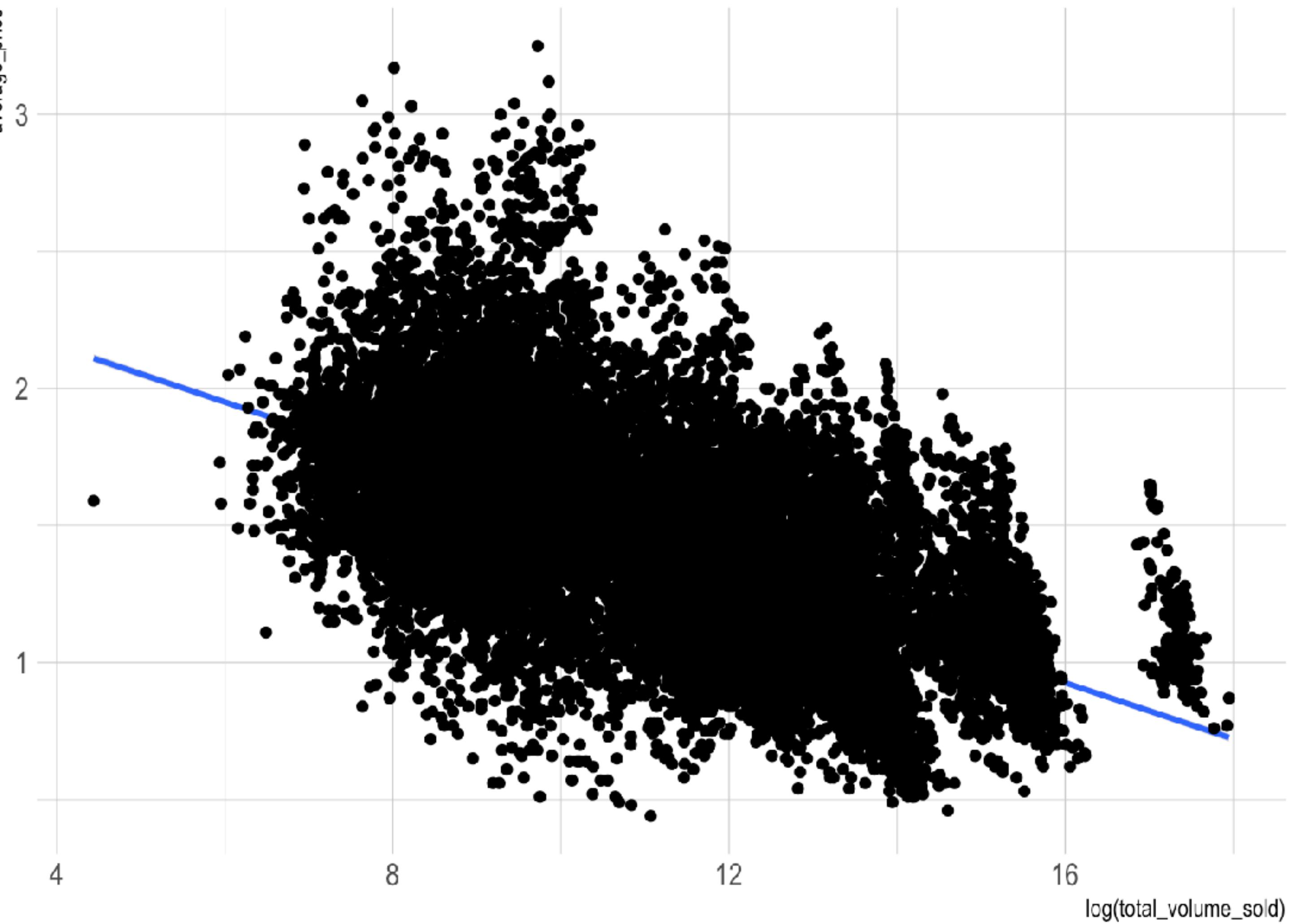
```
avocado_data %>%
  ggplot(
    mapping = aes(
      x = log(total_volume_sold),
      y = average_price
    )
  ) +
  # add a scatter plot
  geom_point() +
  # add a linear regression line
  geom_smooth(method = "lm")
```





# LAYER ORDER

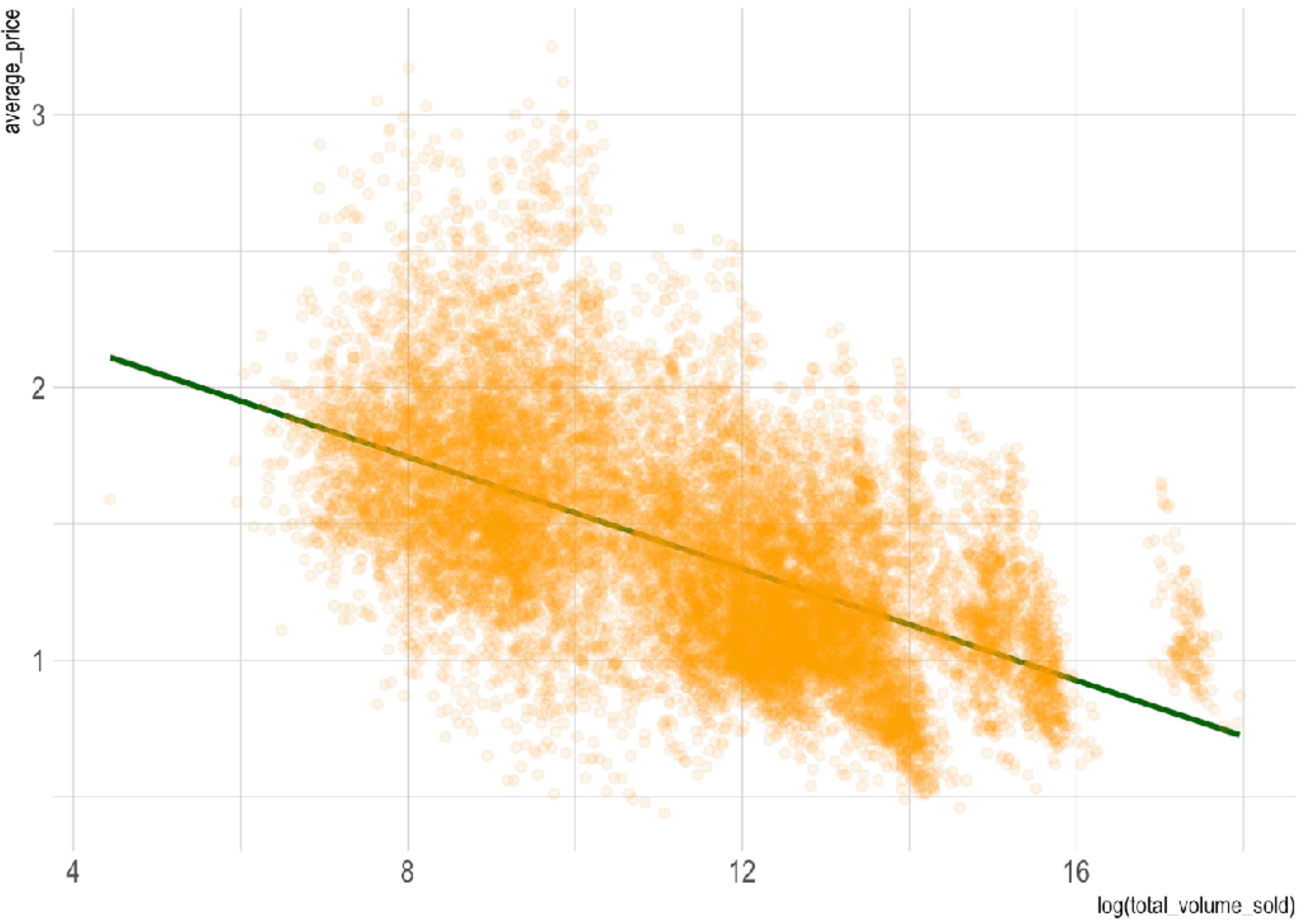
```
avocado_data %>%
  ggplot(
    mapping = aes(
      x = log(total_volume_sold),
      y = average_price
    )
  ) +
  # FIRST: add a linear regression line
  geom_smooth(method = "lm") +
  # THEN: add a scatter plot
  geom_point()
```





# OPACITY

```
avocado_data %>%
  ggplot(
    mapping = aes(
      x = log(total_volume_sold),
      y = average_price
    )
  ) +
  # FIRST: add a linear regression line
  geom_smooth(method = "lm", color = "darkgreen") +
  # THEN: add a scatter plot
  geom_point(alpha = 0.1, color = "orange")
```



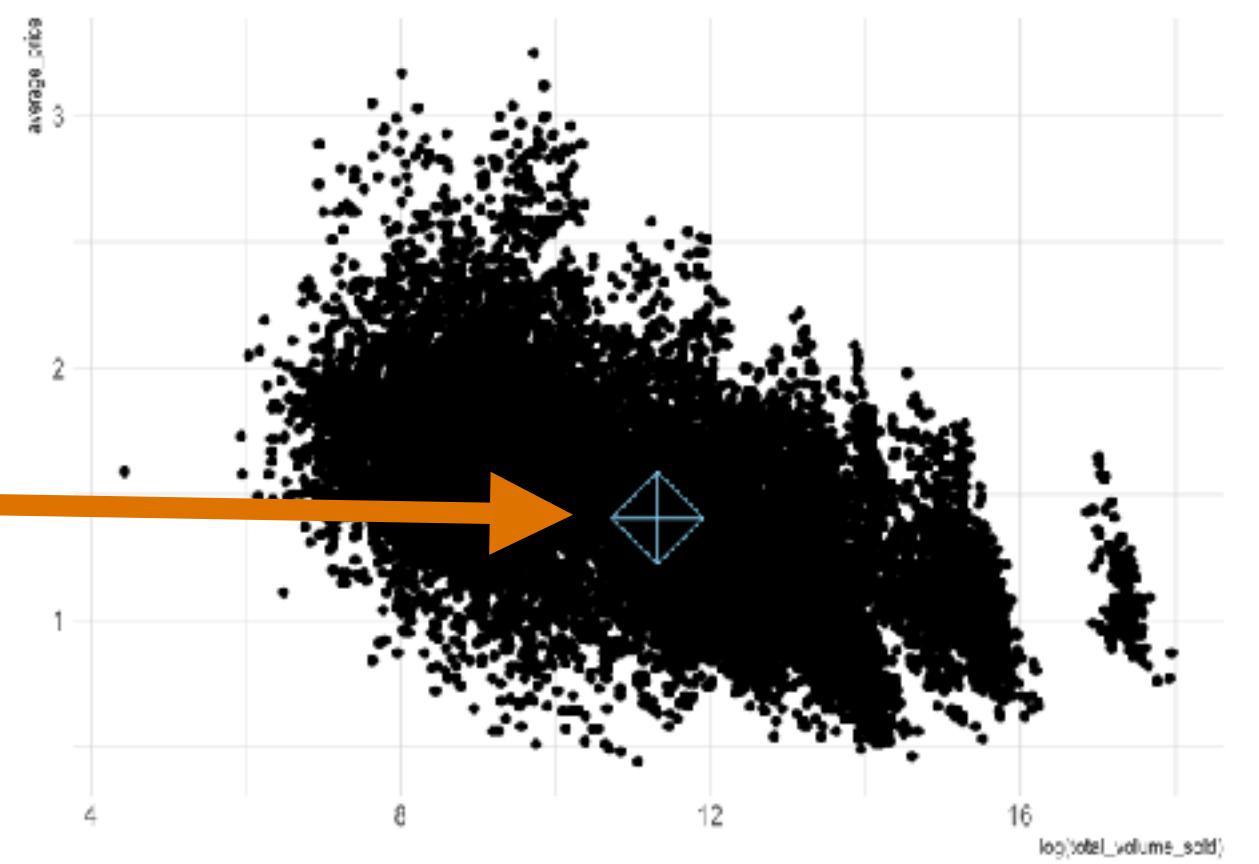


# DIFFERENT DATA FOR DIFFERENT LAYERS

```
# create a small tibble with the means of both
#  variables of interest
avocado_data_means <-
  avocado_data %>%
  summarize(
    mean_volume = mean(log(total_volume_sold)),
    mean_price  = mean(average_price)
  )
avocado_data_means
```

```
## # A tibble: 1 x 2
##   mean_volume mean_price
##       <dbl>      <dbl>
## 1       11.3      1.41
```

```
avocado_data %>%
  ggplot(
    aes(x = log(total_volume_sold),
        y = average_price)
  ) +
  # first layer uses globally declared data & mapping
  geom_point() +
  # second layer uses different data set & mapping
  geom_point(
    data = avocado_data_means,
    mapping = aes(
      x = mean_volume,
      y = mean_price
    ),
    # change shape of element to display (see below)
    shape = 9,
    # change size of element to display
    size = 12,
    color = "skyblue"
  )
```





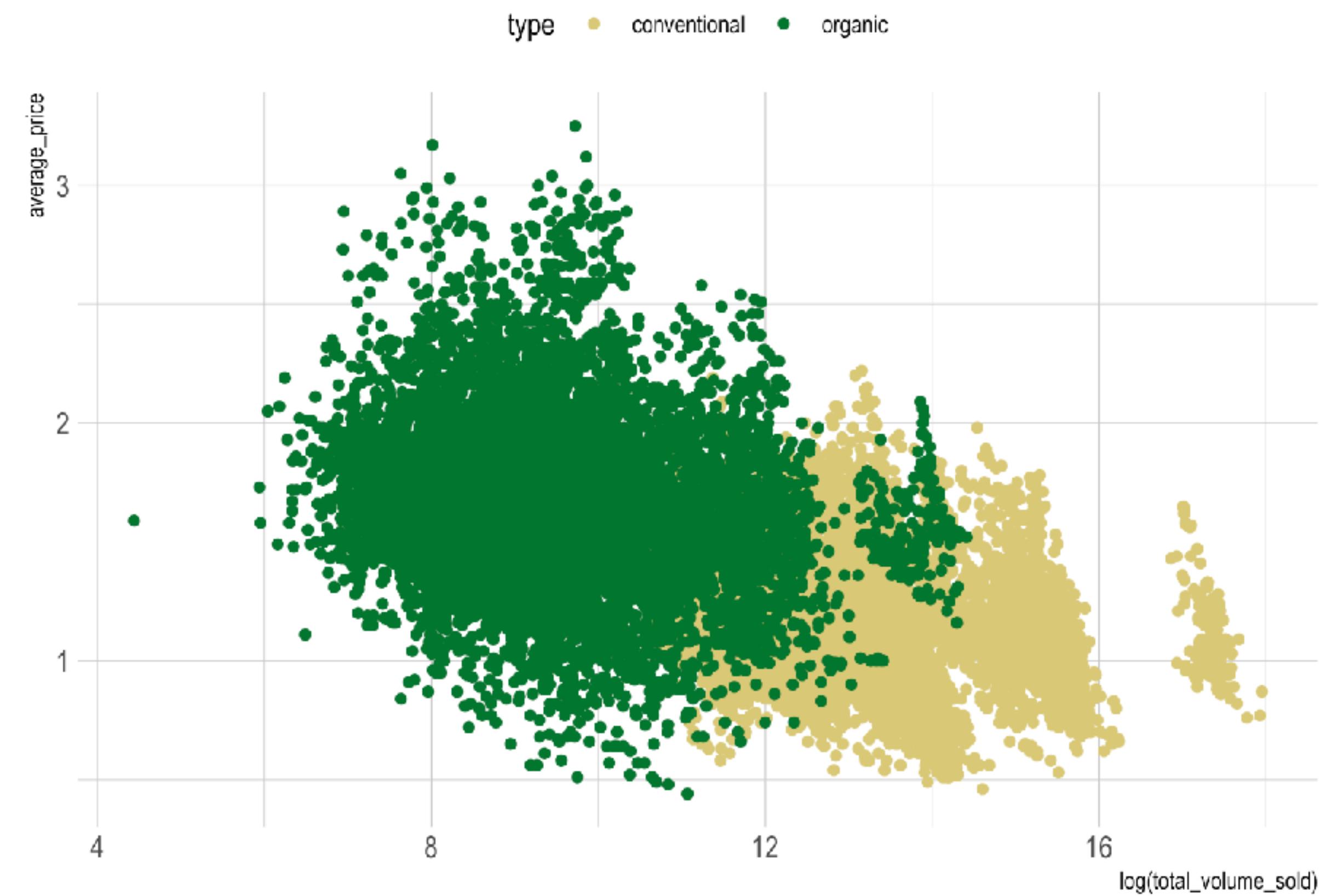
# Grouping



## GROUPING

- ▶ group information for uniform display in terms of color, shape, etc.

```
avocado_data %>%
  ggplot(
    aes(
      x = log(total_volume_sold),
      y = average_price,
      # use a different color for each type of avocado
      color = type
    )
  ) +
  geom_point(aes(color = type))
```

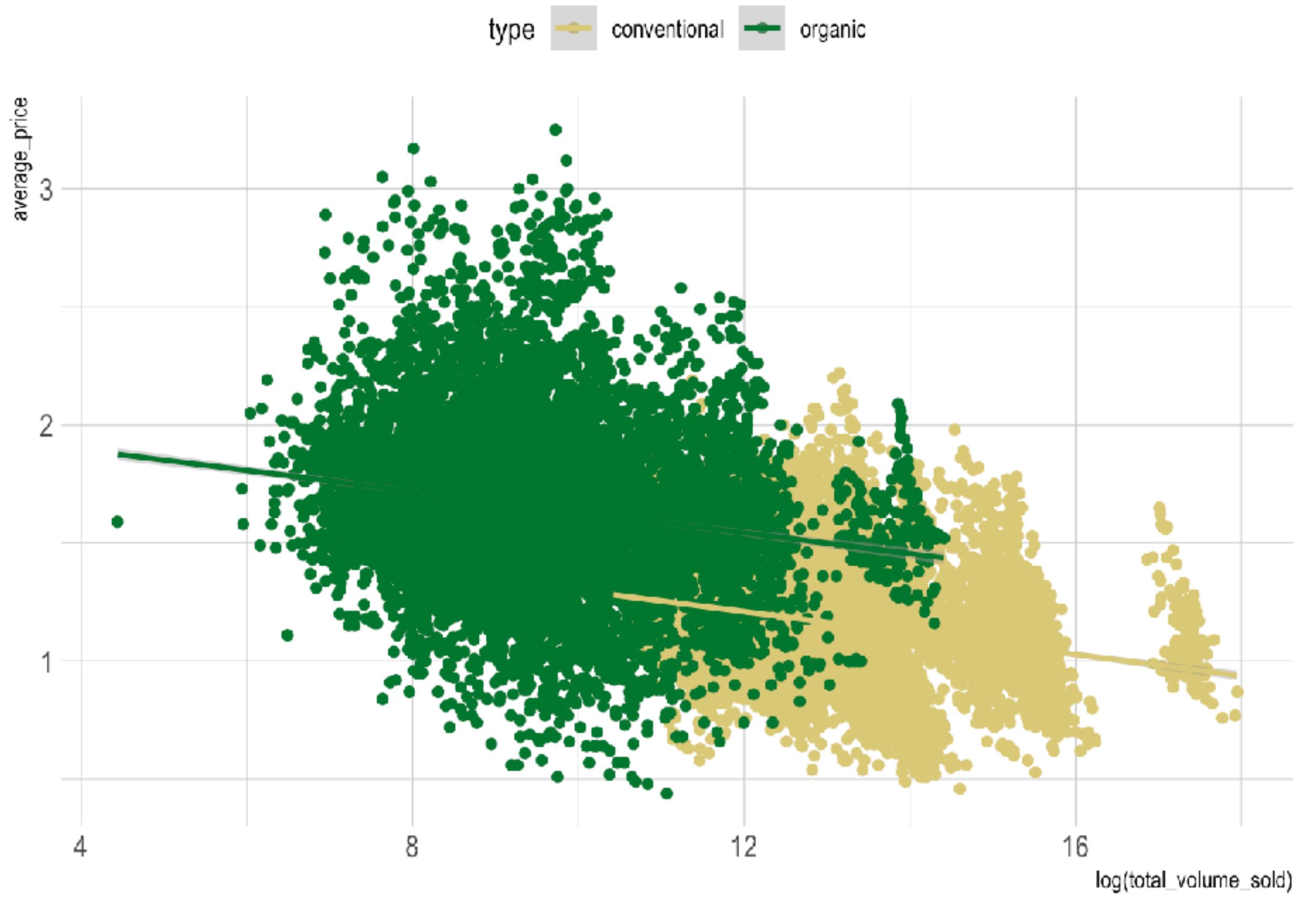




## GLOBAL GROUPING

- ▶ global grouping applies to all subsequent layers

```
avocado_data %>%  
  ggplot(  
    aes(  
      x = log(total_volume_sold),  
      y = average_price,  
      # use a different color for each type of avocado  
      color = type  
    )  
  ) +  
  geom_point(aes(color = type)) +  
  geom_smooth(method = "lm")
```

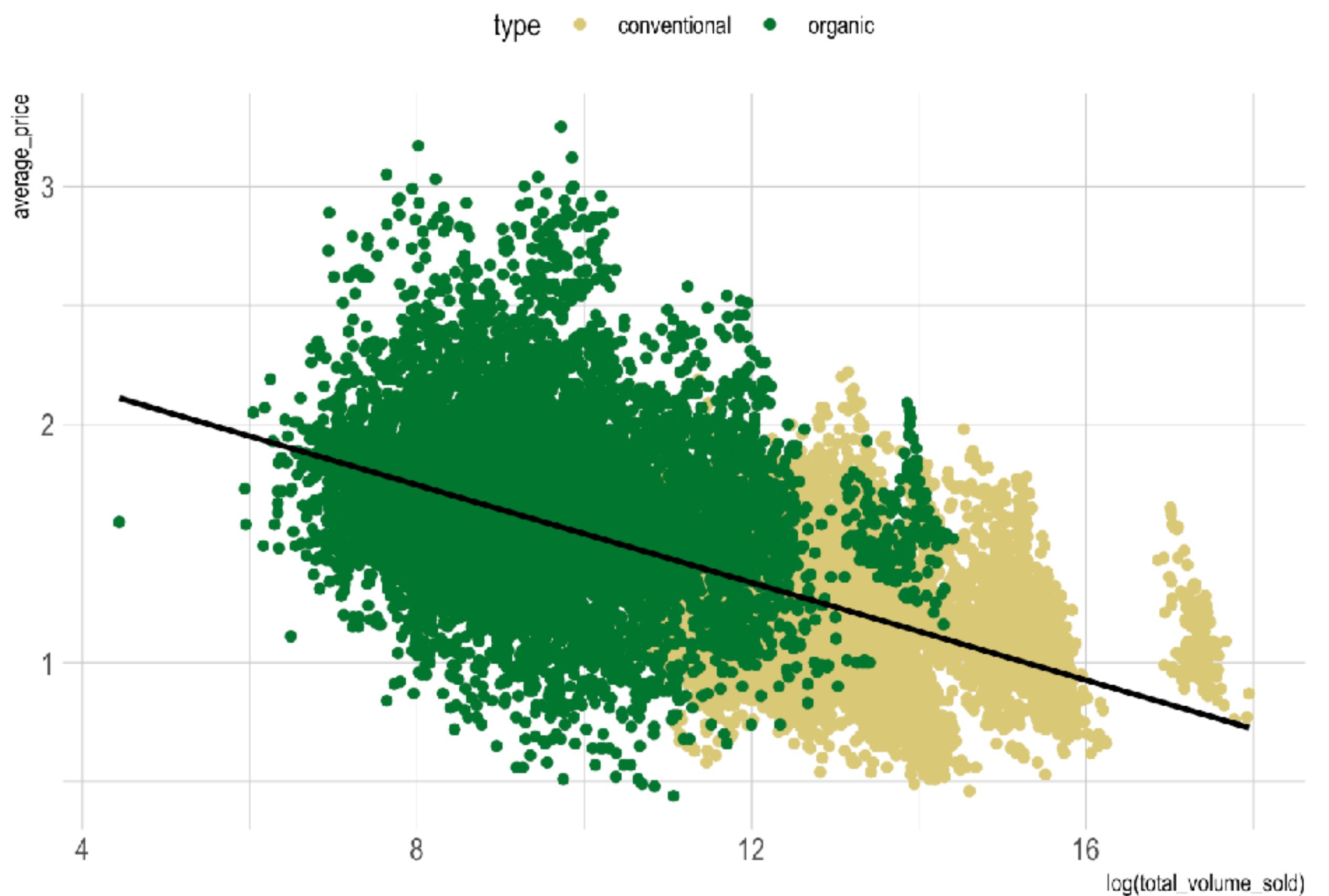




# OVERWRITING GROUPING INFORMATION

- ▶ overwriting grouping information locally

```
avocado_data %>%
  ggplot(
    aes(
      x = log(total_volume_sold),
      y = average_price,
      # use a different color for each type of avocado
      color = type
    )
  ) +
  geom_point(aes(color = type)) +
  geom_smooth(method = "lm", color = "black")
```

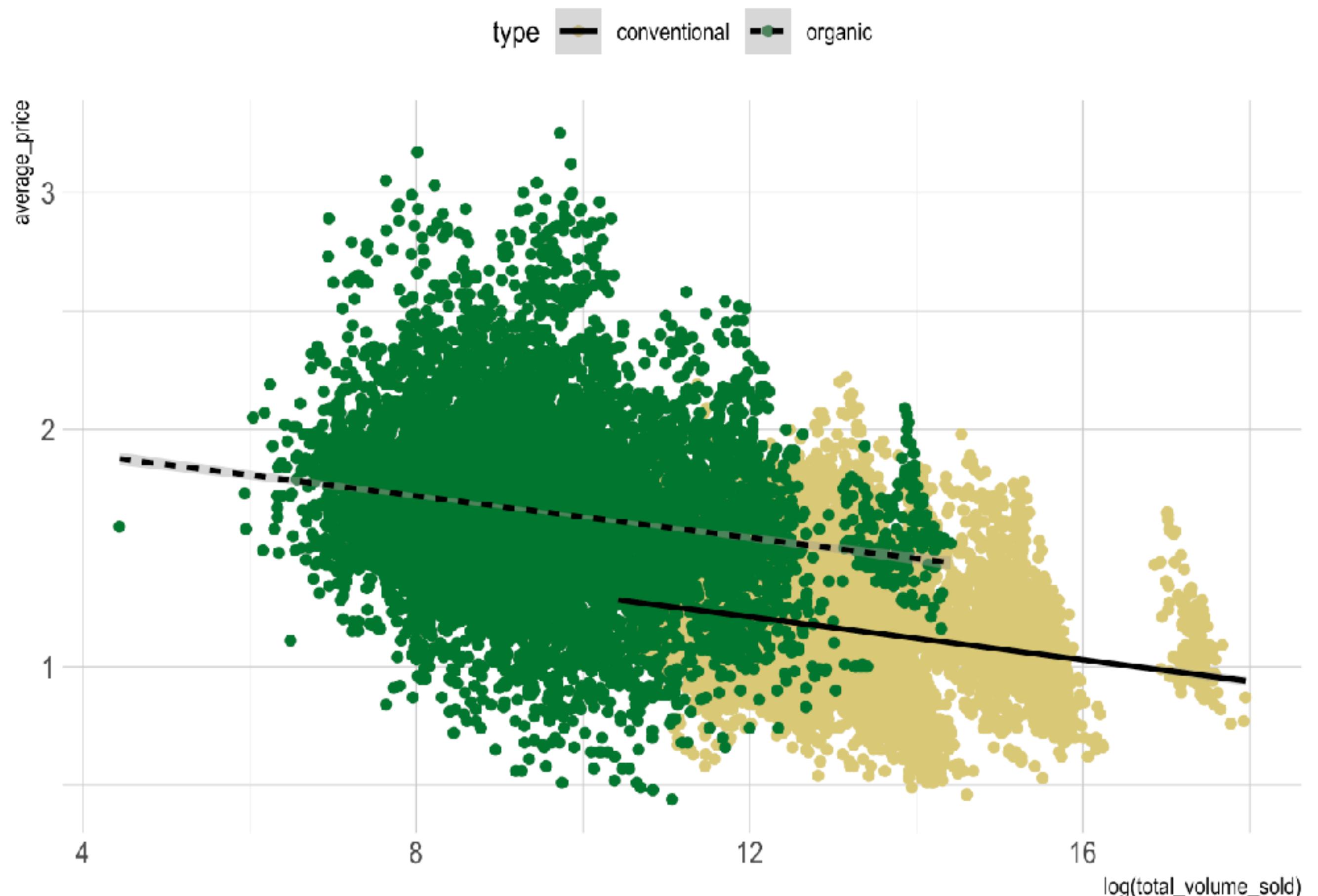




# DIFFERENT GROUPING IN DIFFERENT LAYERS

- ▶ each layer has its own grouping information

```
avocado_data %>%
  ggplot(
    aes(
      x = log(total_volume_sold),
      y = average_price,
      # use a different color for each type of avocado
      color = type
    )
  ) +
  geom_point(aes(color = type)) +
  geom_smooth(
    # tell the smoother to deal with avocados types separately
    aes(group = type, linetype = type),
    method = "lm",
    color = "black"
  )
```

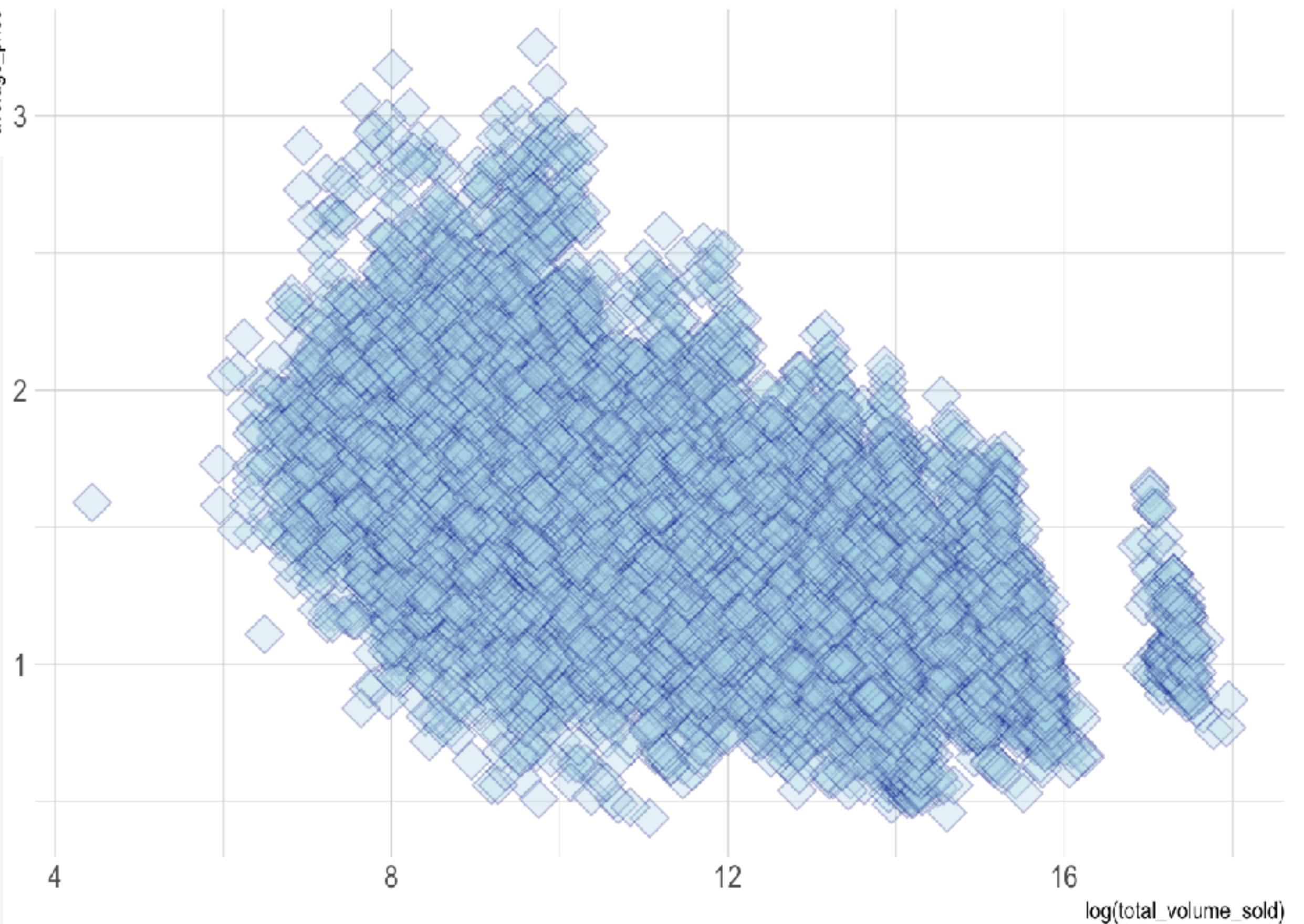


# Geoms & plot types



# SCATTER PLOTS

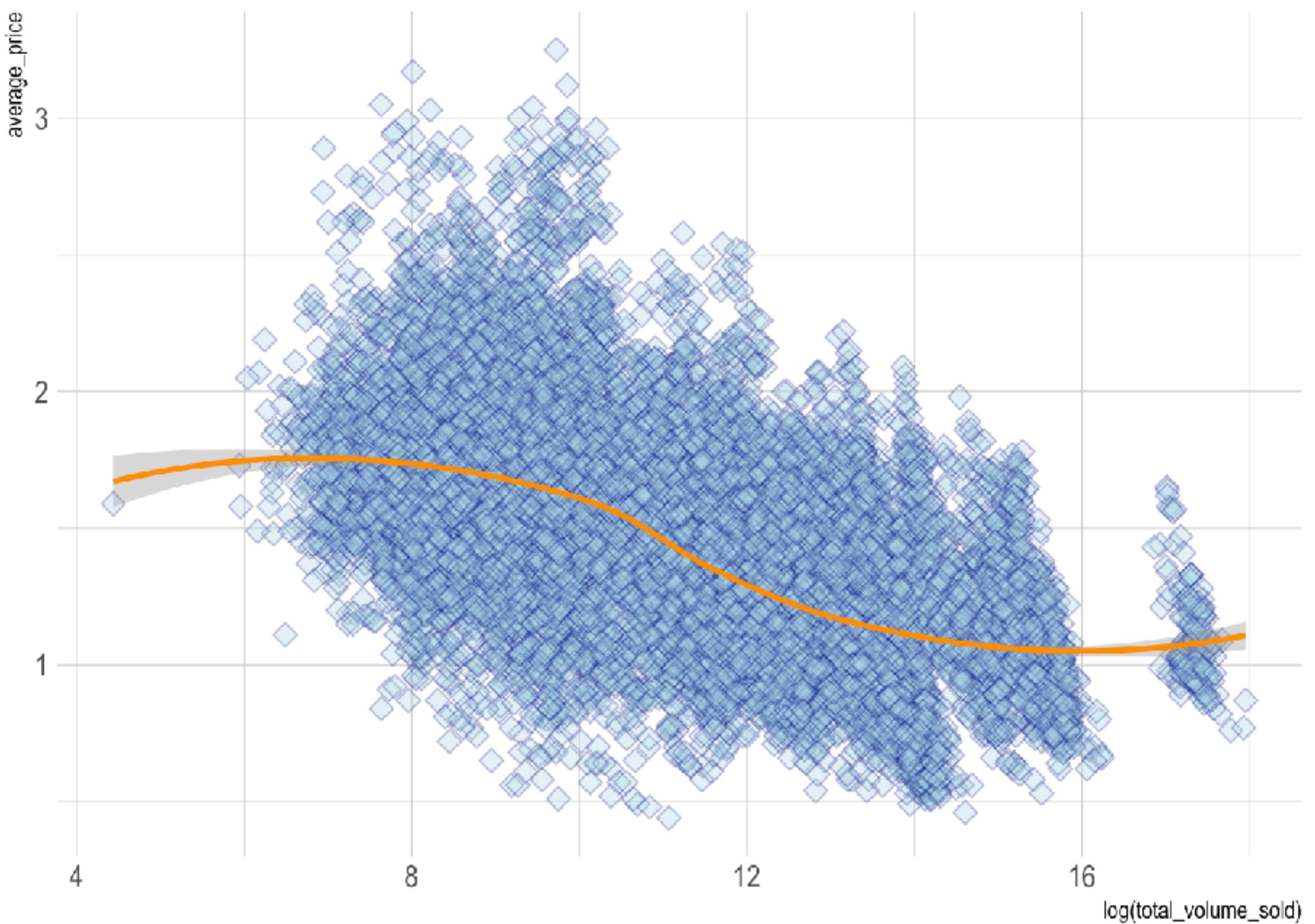
```
avocado_data %>%
  ggplot(aes(x = log(total_volume_sold), y = average_price)) +
  geom_point(
    # shape to display is number 23 (tilted rectangle, see below)
    shape = 23,
    # color of the surrounding line of the shape (for shapes 21-24)
    color = "darkblue",
    # color of the interior of each shape
    fill = "lightblue",
    # size of each shape (default is 1)
    size = 5,
    # level of opacity for each shape
    alpha = 0.3
  )
```





# CURVE AND LINE FITS

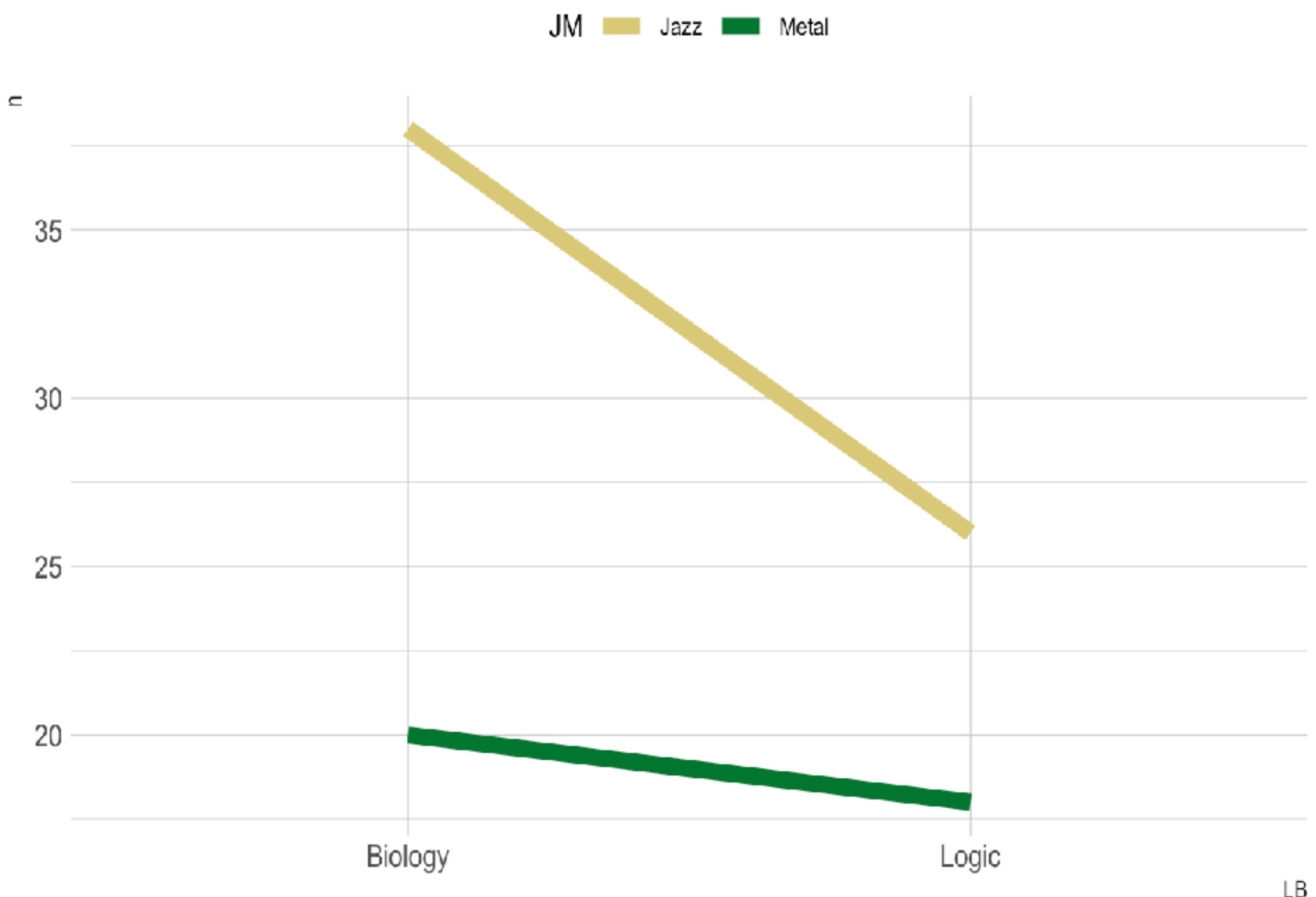
```
avocado_data %>%  
  
  ggplot(aes(x = log(total_volume_sold), y = average_price))  
  
  geom_point(  
    shape = 23,  
    color = "darkblue",  
    fill = "lightblue",  
    size = 3,  
    alpha = 0.3  
  ) +  
  
  geom_smooth(  
    # fitting a smoothed curve to the data  
    method = "loess",  
    # display standard error around smoothing curve  
    se = T,  
    color = "darkorange"  
  )
```





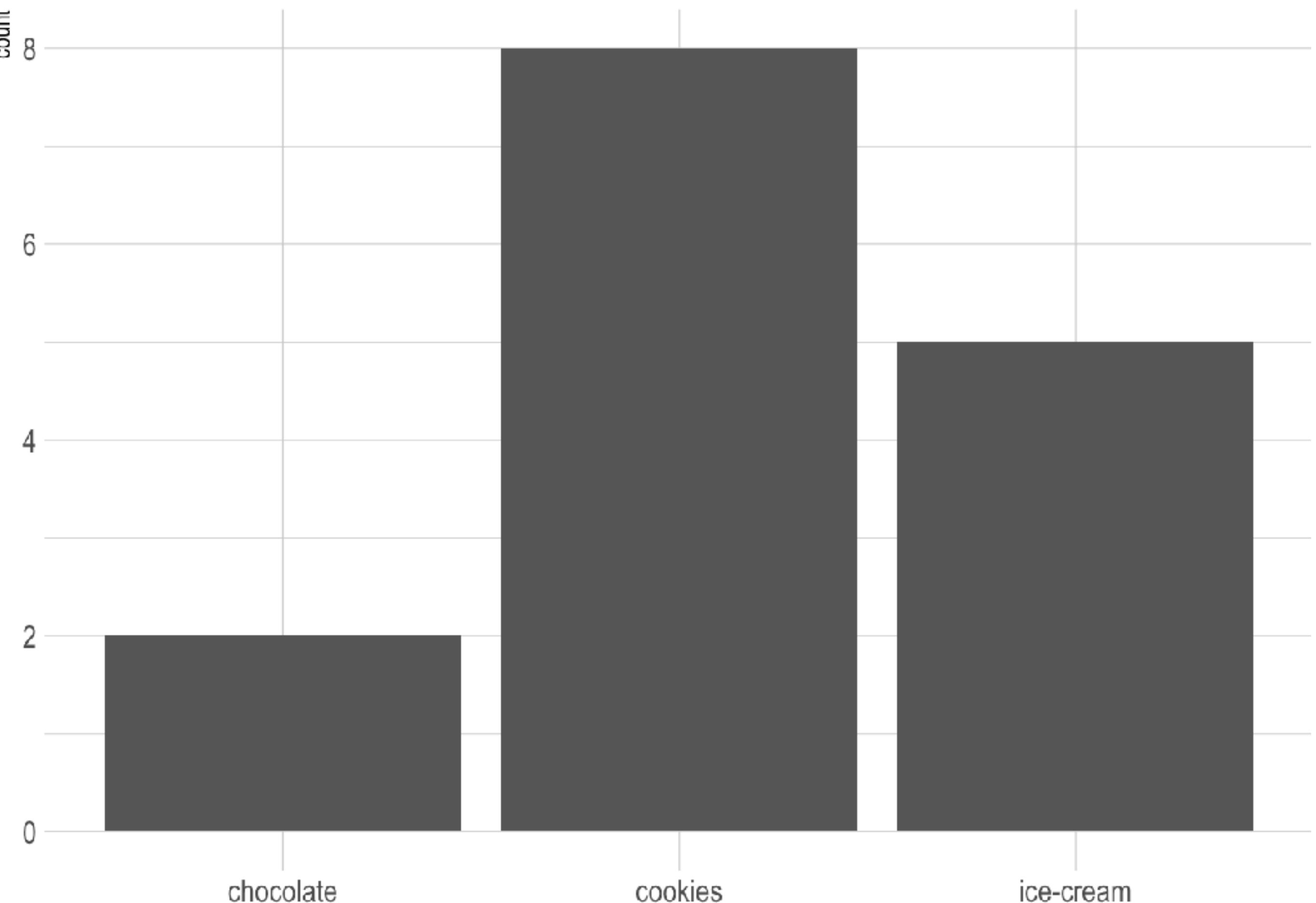
## LINE PLOTS

```
BLJM_associated_counts %>%
  ggplot(
    aes(
      x = LB,
      y = n,
      color = JM,
      group = JM
    )
  ) +
  geom_line(size = 3)
```



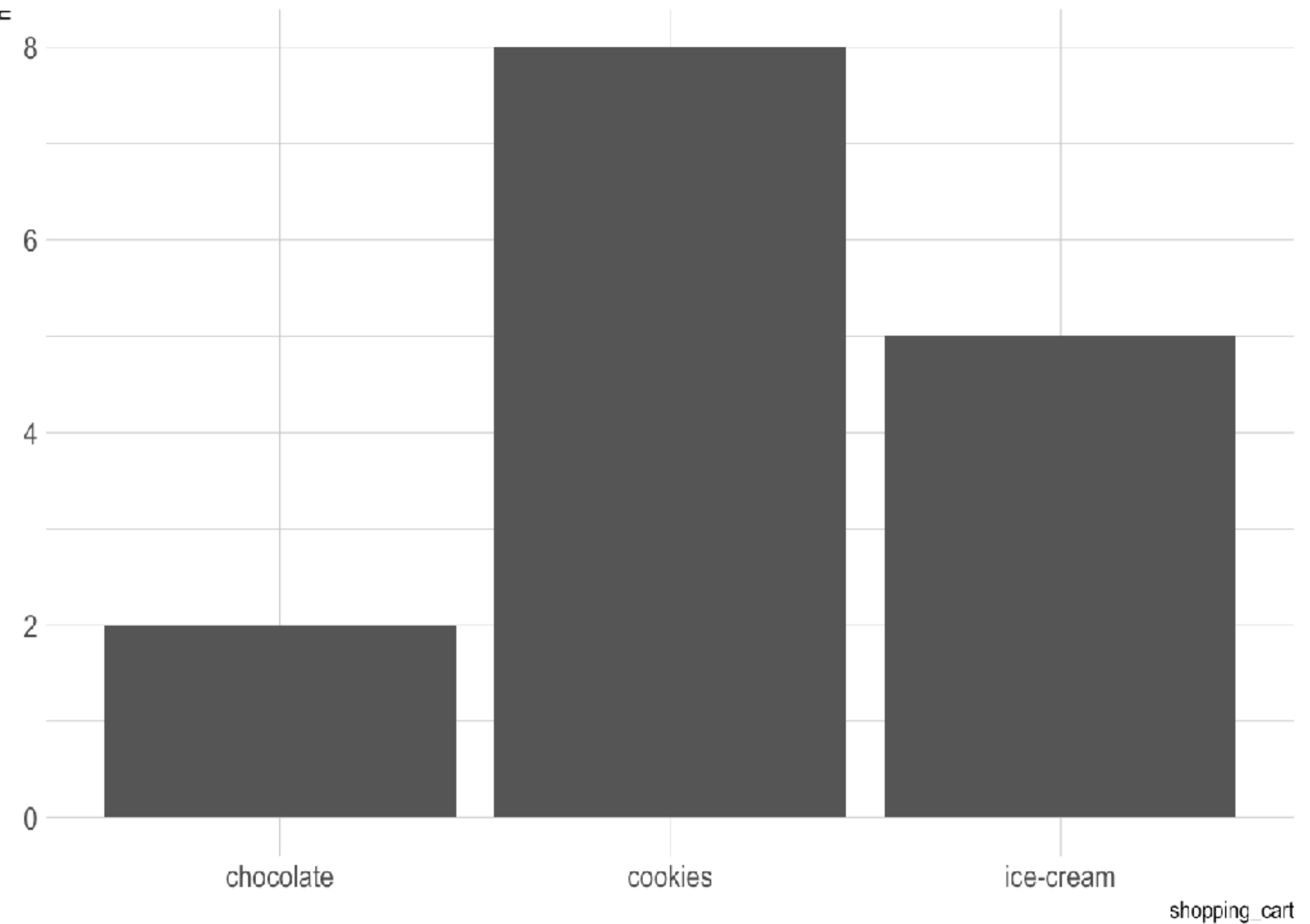
## BAR PLOTS

```
tibble(  
  shopping_cart = c(  
    rep("chocolate", 2),  
    rep("ice-cream", 5),  
    rep("cookies", 8)  
  )  
  ) %>%  
  ggplot(aes(x = shopping_cart)) +  
  geom_bar()
```



## BAR PLOTS

```
tibble(  
  shopping_cart = c(  
    rep("chocolate", 2),  
    rep("ice-cream", 5),  
    rep("cookies", 8)  
  )  
) %>%  
  dplyr::count(shopping_cart) %>%  
  ggplot(aes (x = shopping_cart, y = n) ) +  
  geom_col()
```

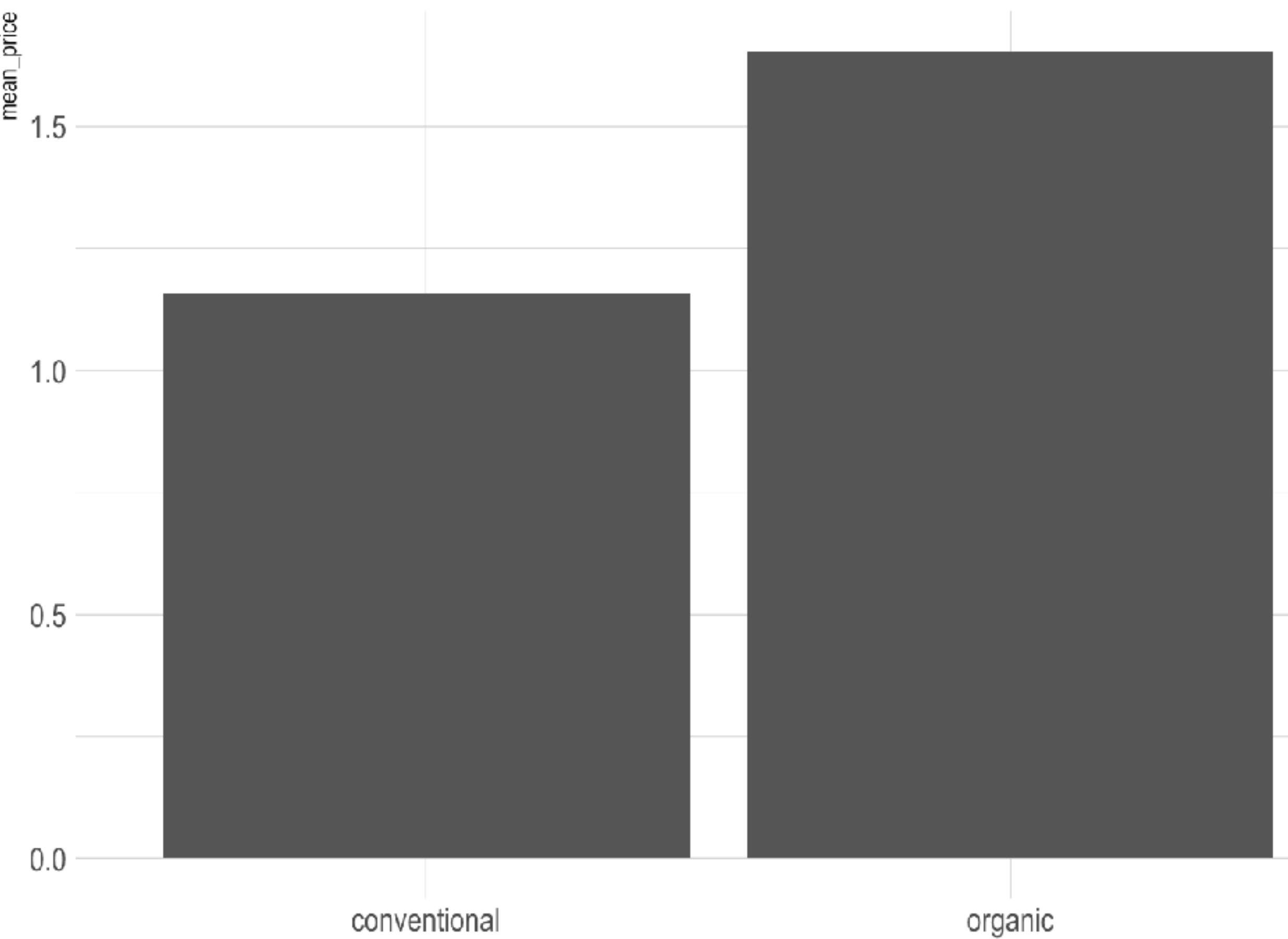




## BAR PLOTS CAN BE UNDERINFORMATIVE

- ▶ suboptimal data-ink ratio
- ▶ lacks distributional information

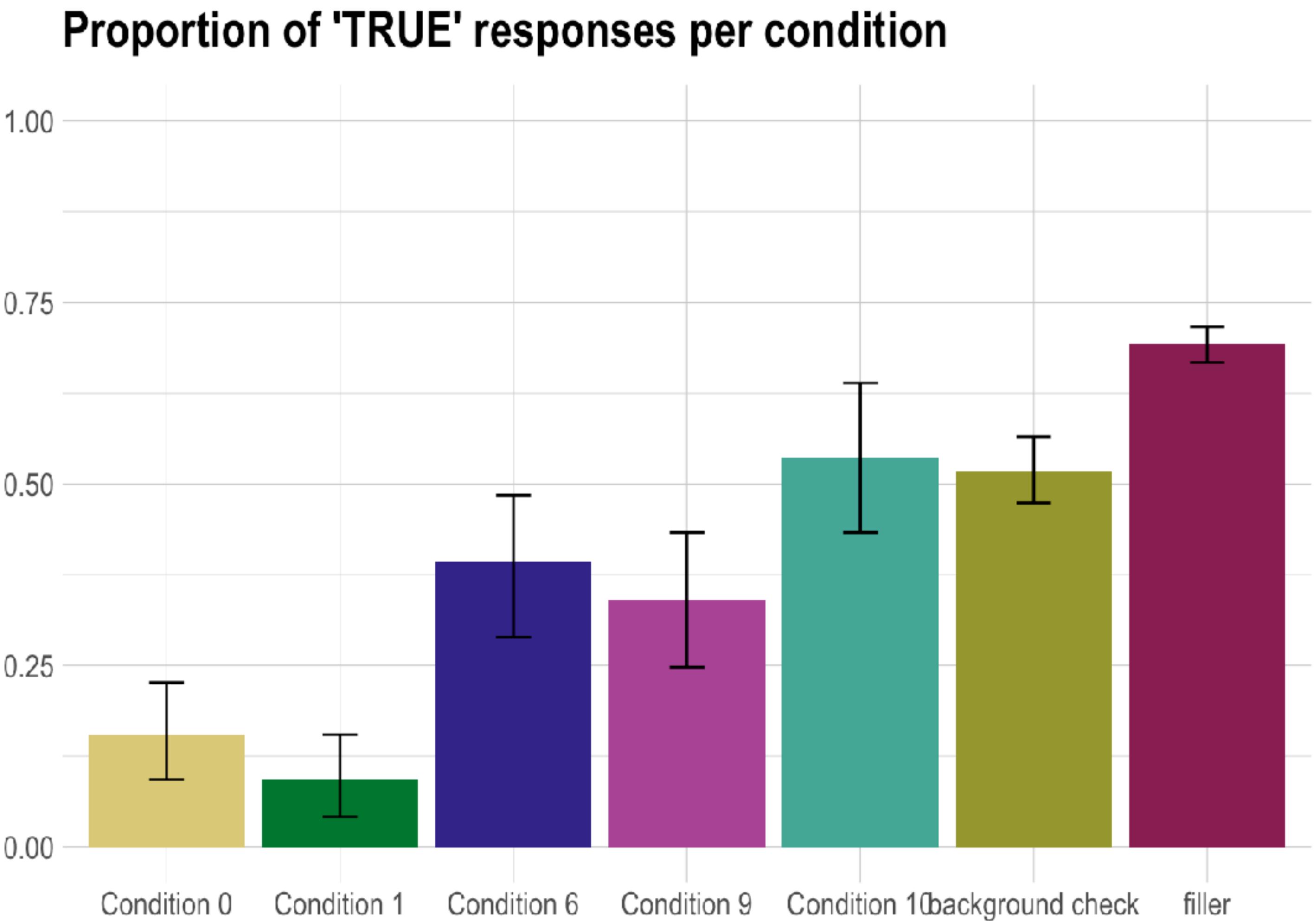
```
avocado_data %>%  
  group_by(type) %>%  
  summarise(  
    mean_price = mean(average_price)  
  ) %>%  
  ggplot(aes(x = type, y = mean_price)) +  
  geom_col()
```





## BAR PLOTS CAN OKAY

- ▶ choice proportions
- ▶ with 95% bootstrapped CIs

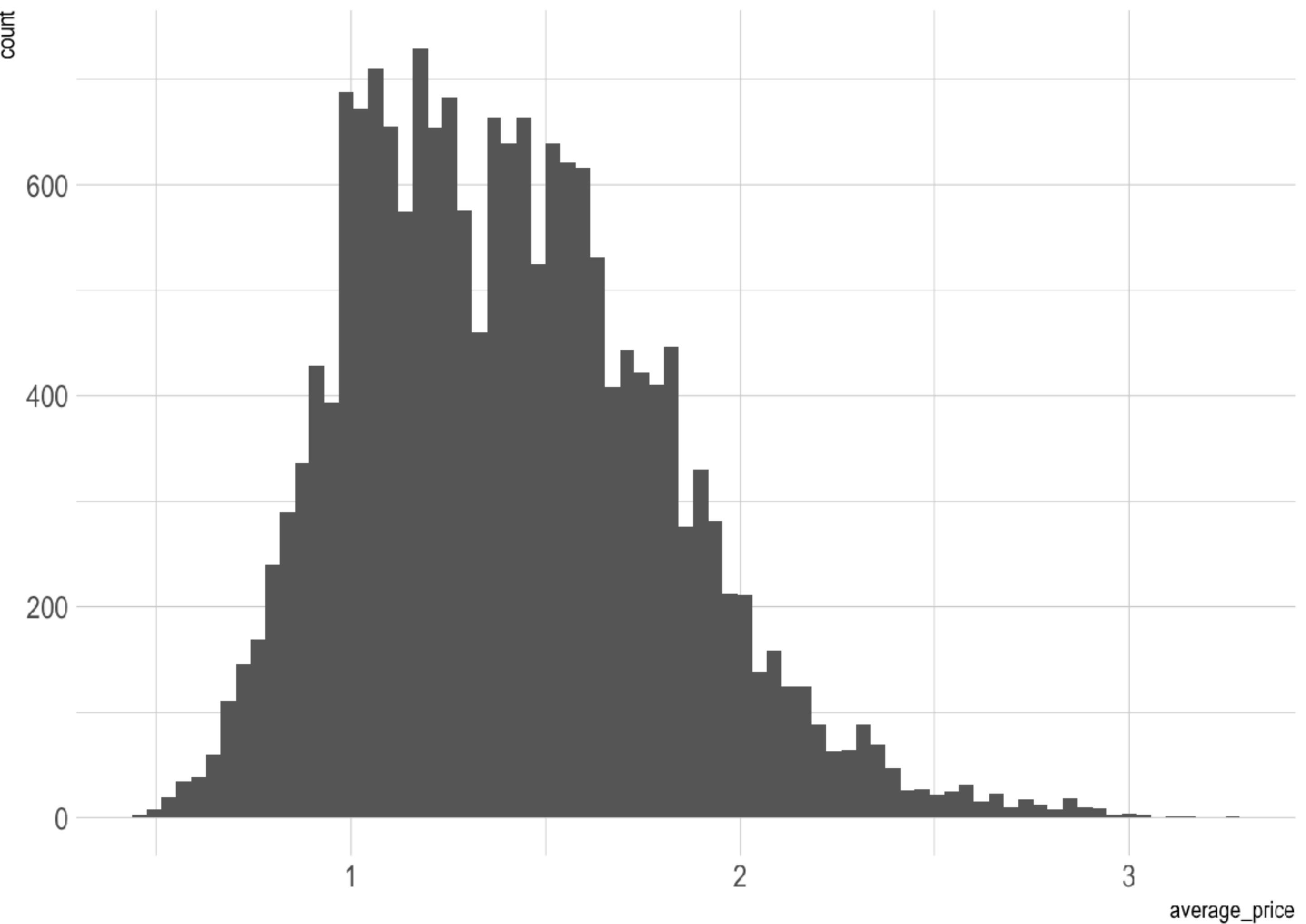




## HISTOGRAMS

- ▶ fix bins
- ▶ count number of data points in each bin
- ▶ plot as bar

```
avocado_data %>%  
  ggplot(aes(x = average_price)) +  
  geom_histogram(bins = 75)
```

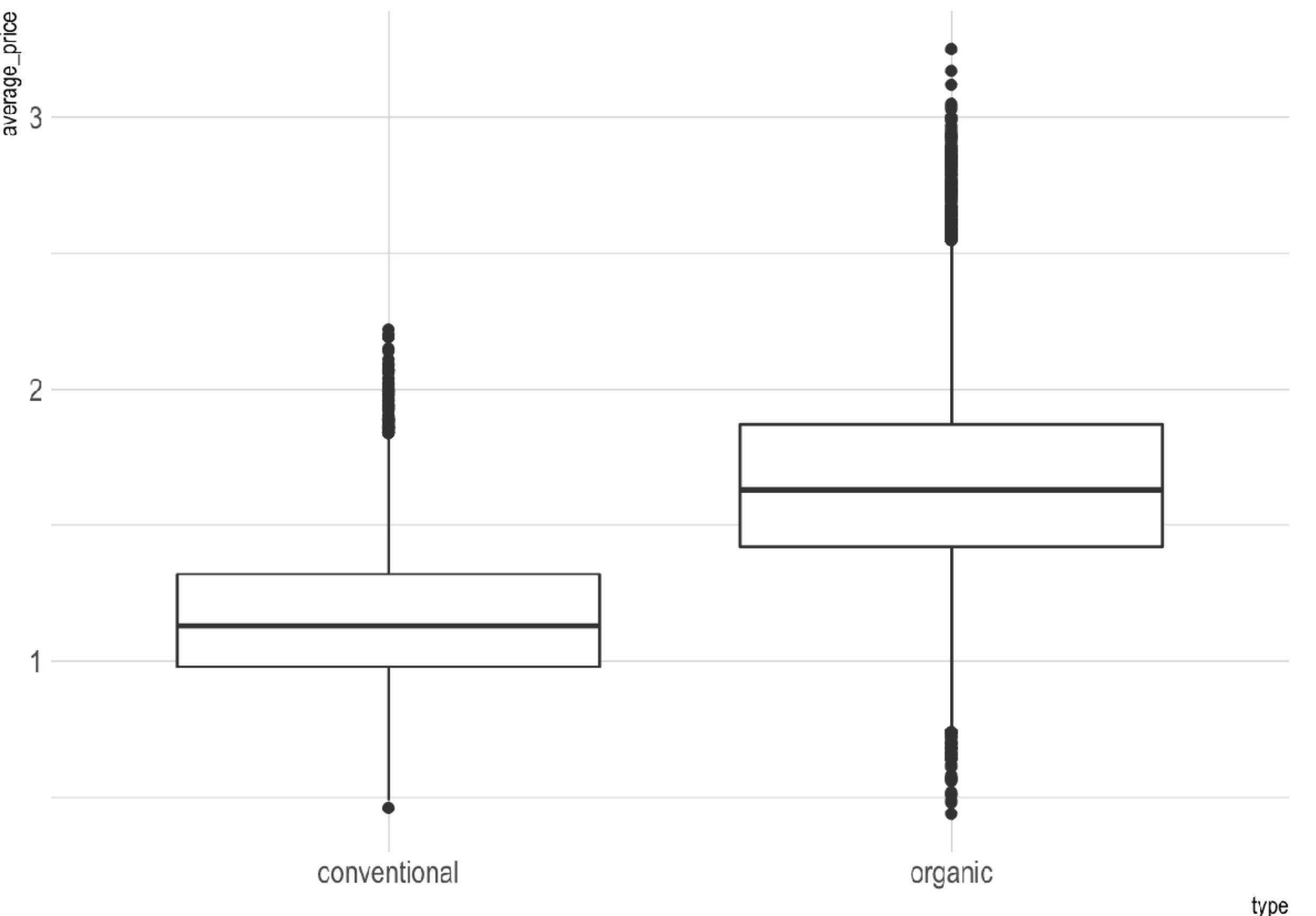




## BOX PLOTS

- ▶ visualize common summary statistics
  - ▶ mean
  - ▶ 25% & 75% quantile
  - ▶ ...

```
avocado_data %>%
  ggplot(aes(x = type , y = average_price)) +
  geom_boxplot()
```

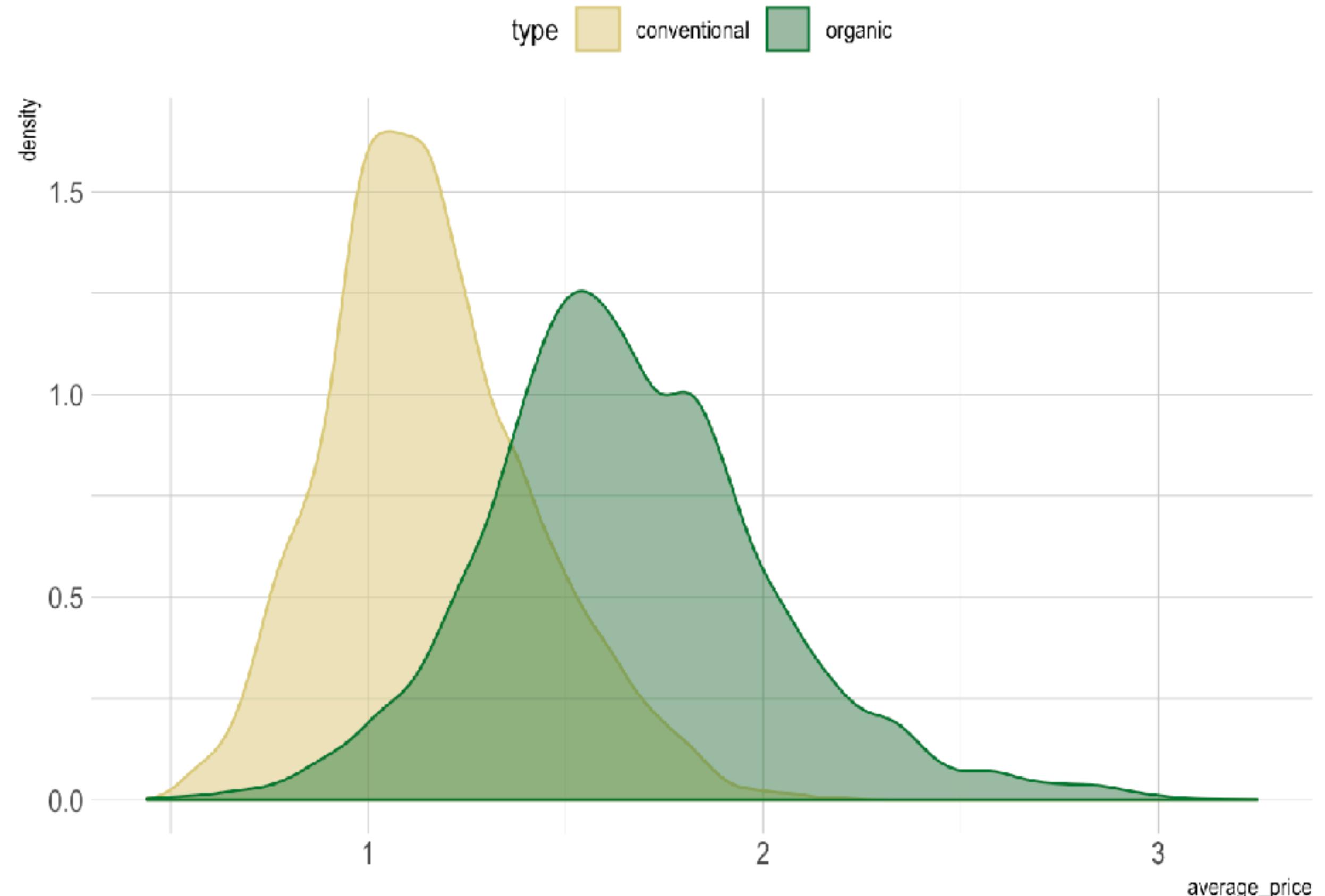




## DENSITY PLOTS

- ▶ “generalized histogram”
- ▶ uses kernel estimation to predict smoothed curves

```
avocado_data %>%
  ggplot(aes(x = average_price, color = type, fill = type)) +
  geom_density(alpha = 0.5)
```

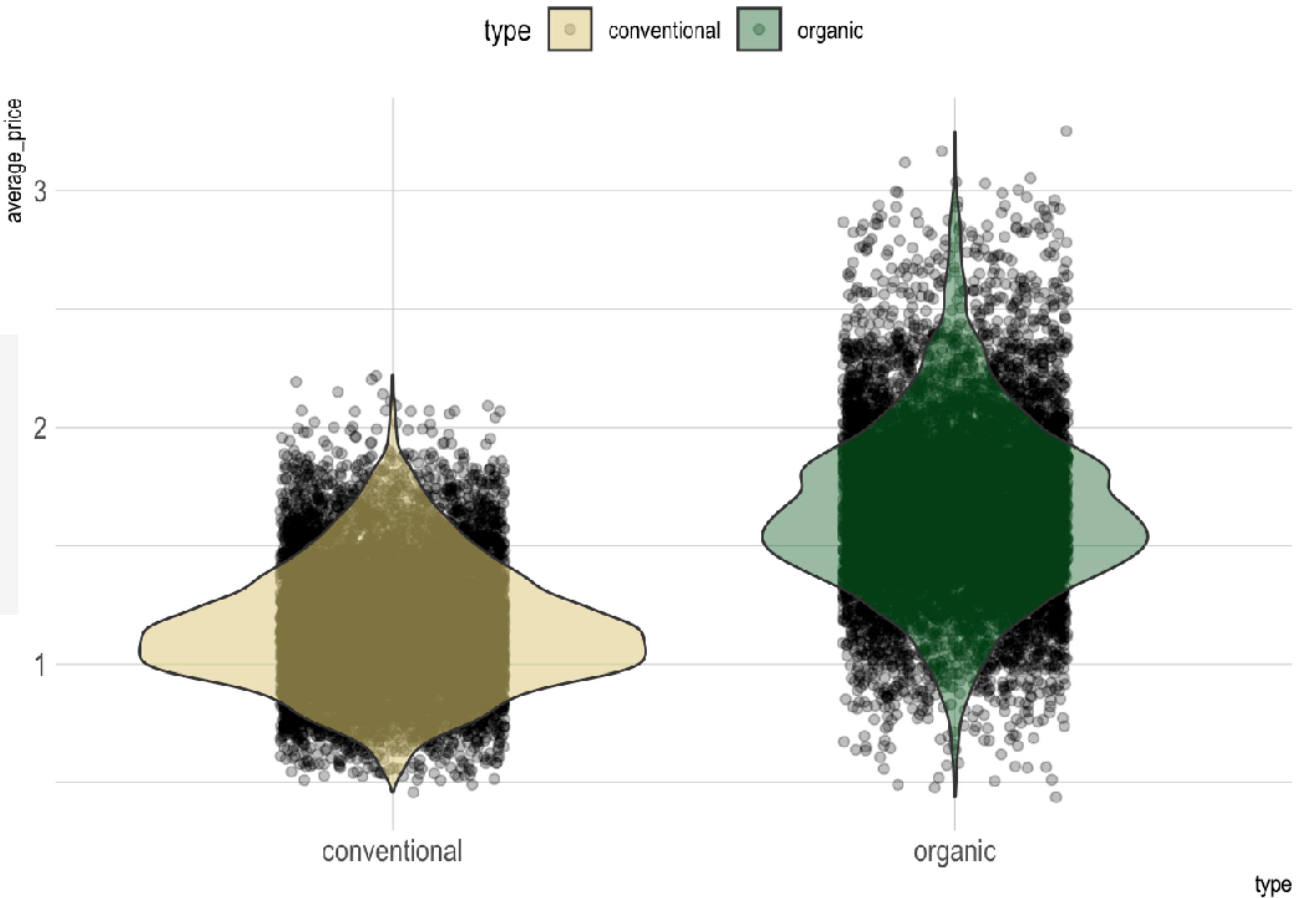




## VIOLIN PLOTS

- ▶ “mirrored density plots”
- ▶ good for multi-group comparisons

```
avocado_data %>%
  ggplot(aes(x = type, y= average_price, fill = type)) +
  geom_jitter(alpha = 0.3, width = 0.2) +
  geom_violin(alpha = 0.5)
```

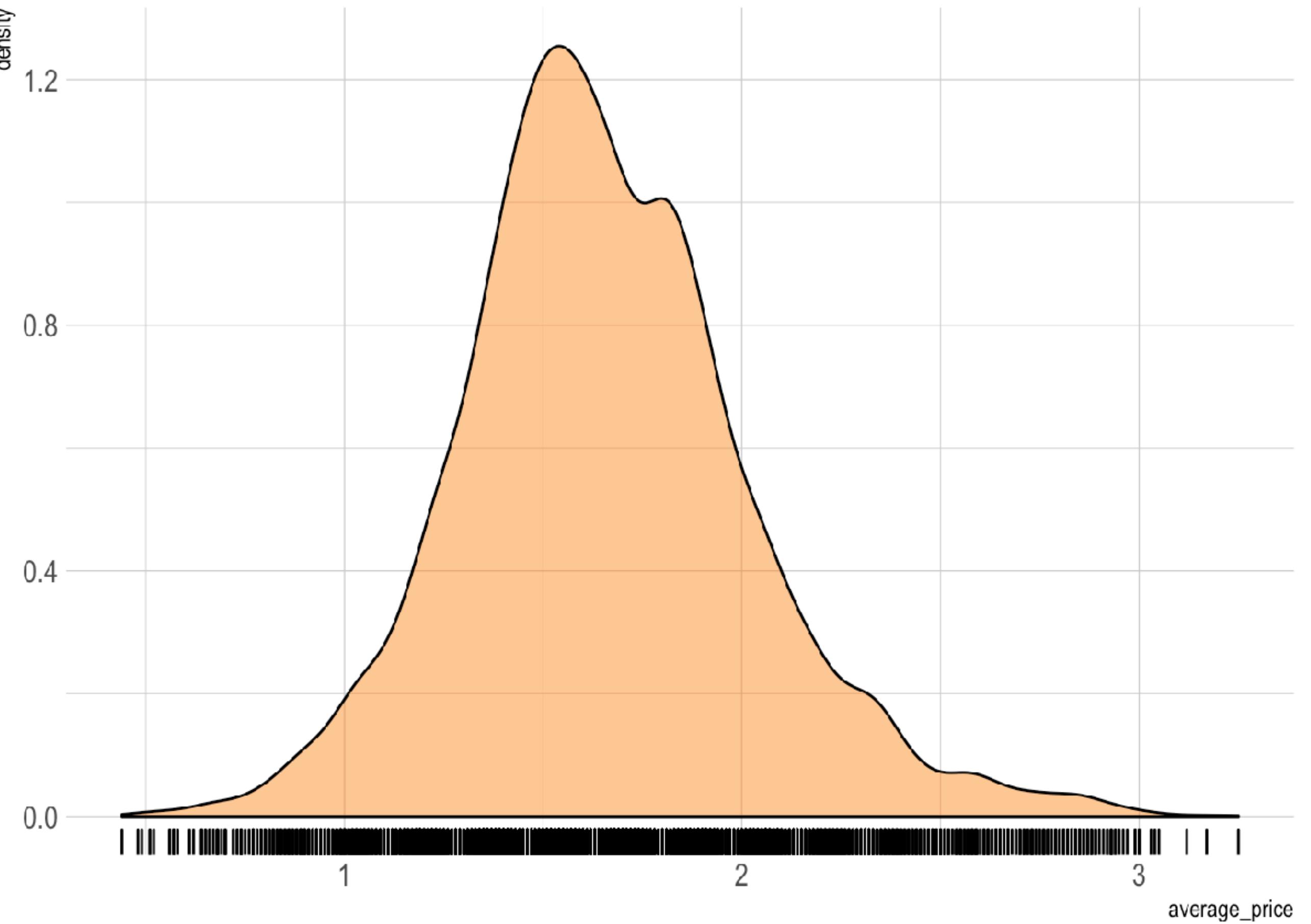




## RUG PLOTS

- ▶ show data points near axis

```
avocado_data %>%
  filter(type == "organic") %>%
  ggplot(aes(x = average_price)) +
  geom_density(fill = "darkorange", alpha = 0.5) +
  geom_rug()
```

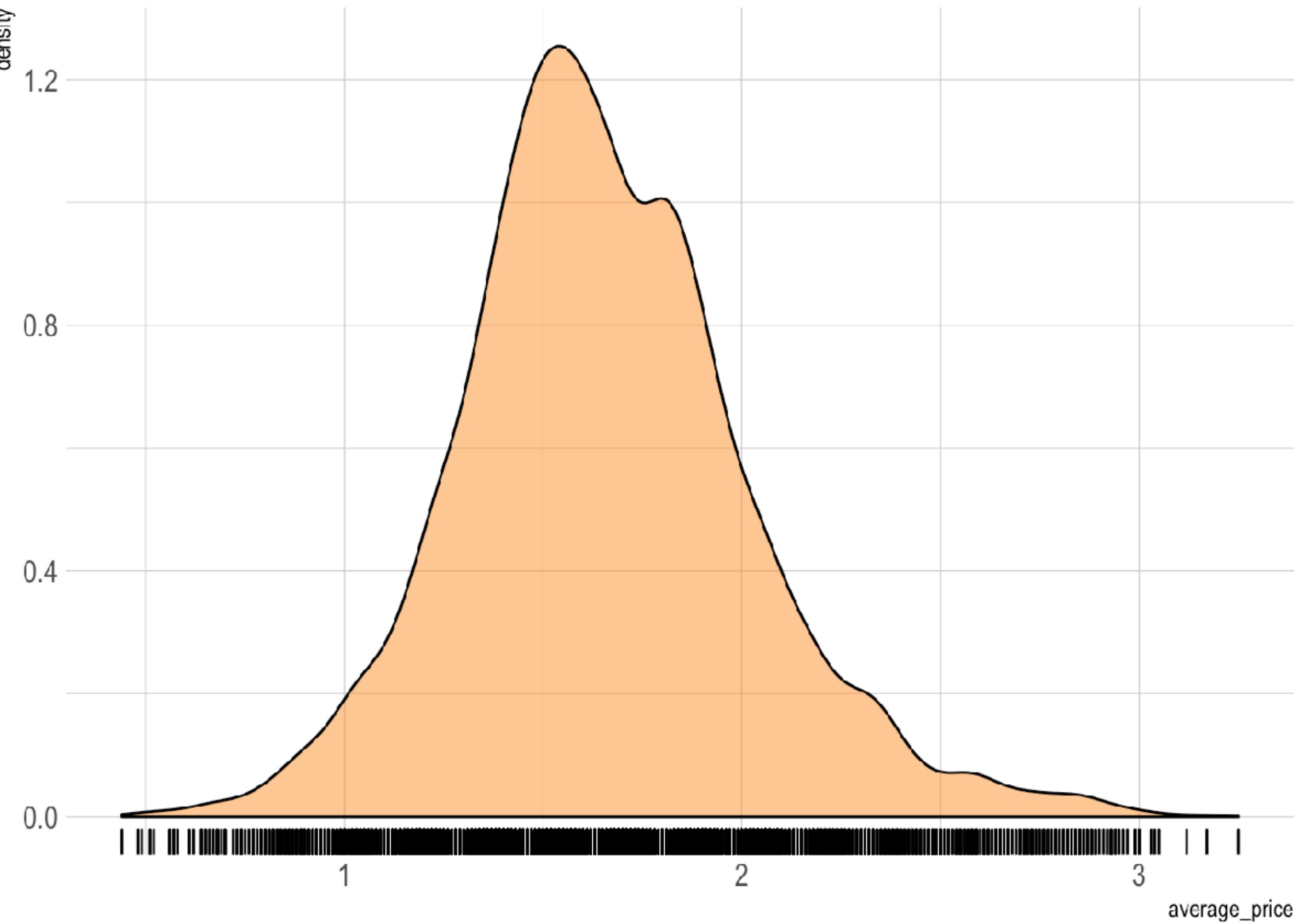


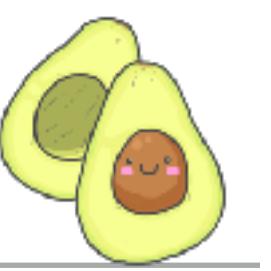


## RUG PLOTS

- ▶ show data points near axis

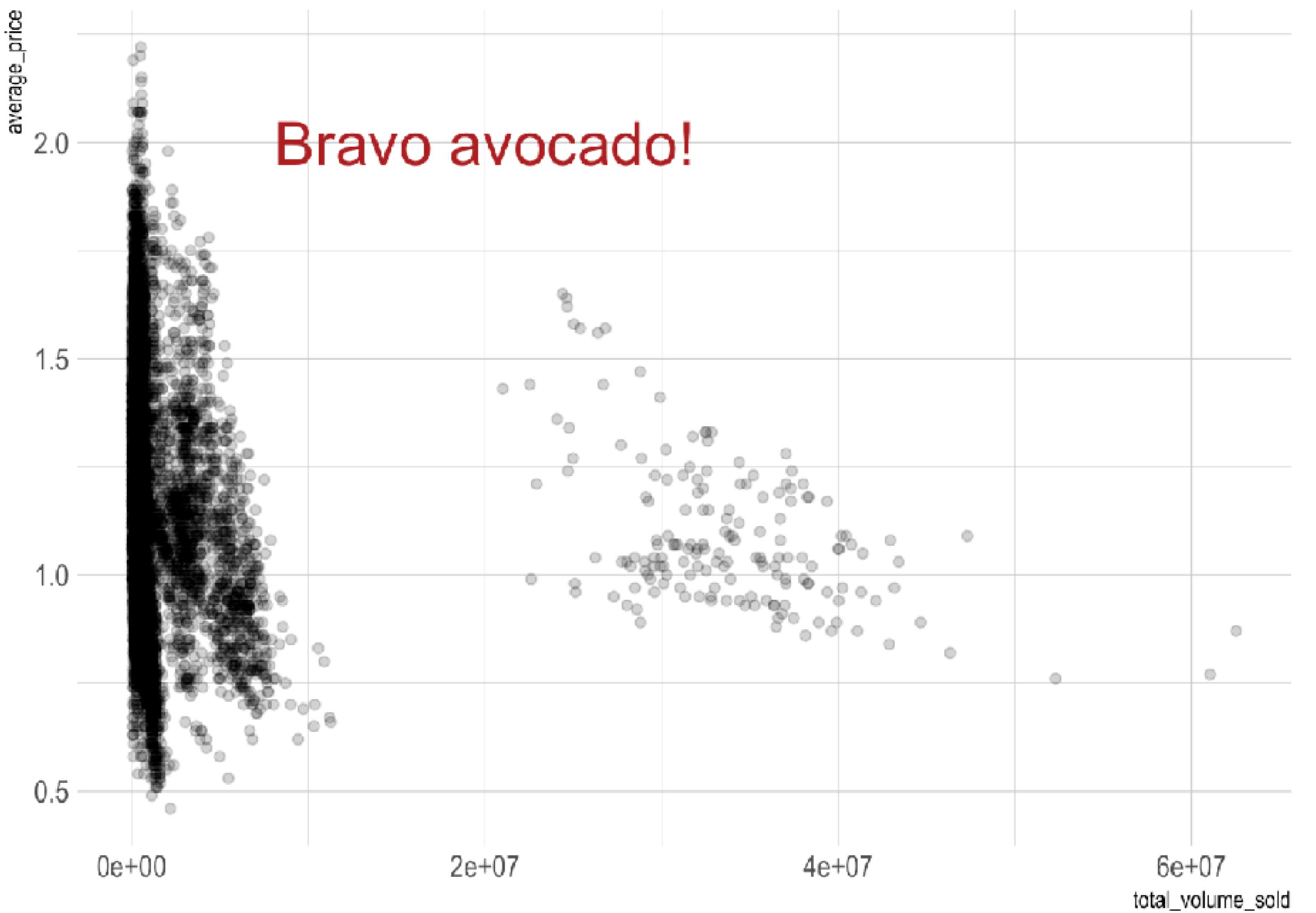
```
avocado_data %>%
  filter(type == "organic") %>%
  ggplot(aes(x = average_price)) +
  geom_density(fill = "darkorange", alpha = 0.5) +
  geom_rug()
```





# ANNOTATION

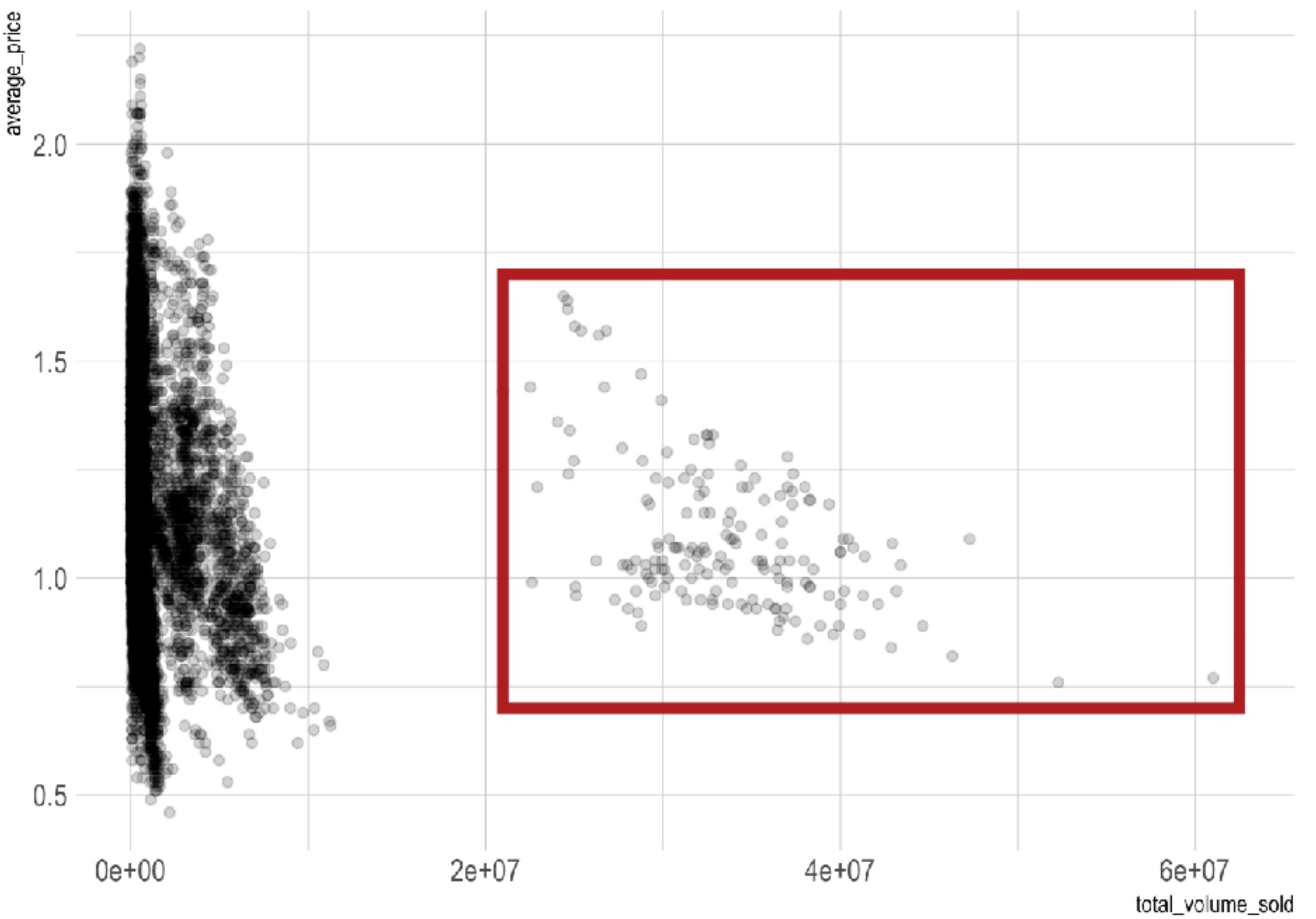
```
avocado_data %>%
  filter(type == "conventional") %>%
  ggplot(aes(x = total_volume_sold, y = average_price)) +
  geom_point(alpha = 0.2) +
  annotate(
    geom = "text",
    # x and y coordinates for the text
    x = 2e7,
    y = 2,
    # text to be displayed
    label = "Bravo avocado!",
    color = "firebrick",
    size = 8
  )
```

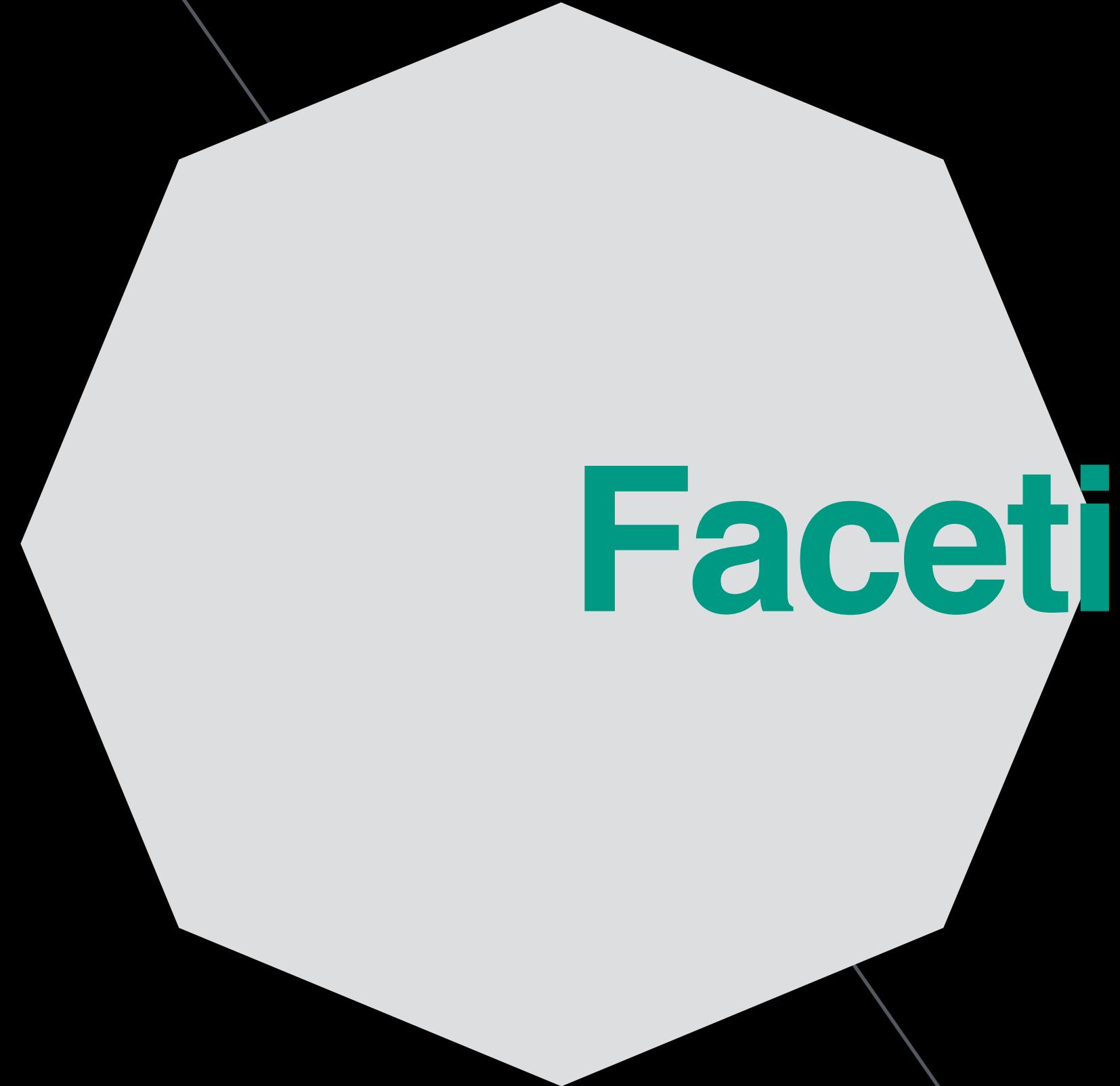




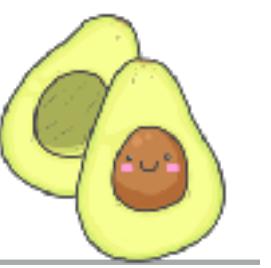
## ANNOTATION

```
avocado_data %>%
  filter(type == "conventional") %>%
  ggplot(aes(x = total_volume_sold, y = average_price)) +
  geom_point(alpha = 0.2) +
  annotate(
    geom = "rect",
    # coordinates for the rectangle
    xmin = 2.1e7,
    xmax = max(avocado_data$total_volume_sold) + 100,
    ymin = 0.7,
    ymax = 1.7,
    color = "firebrick",
    fill = "transparent",
    size = 2
  )
```

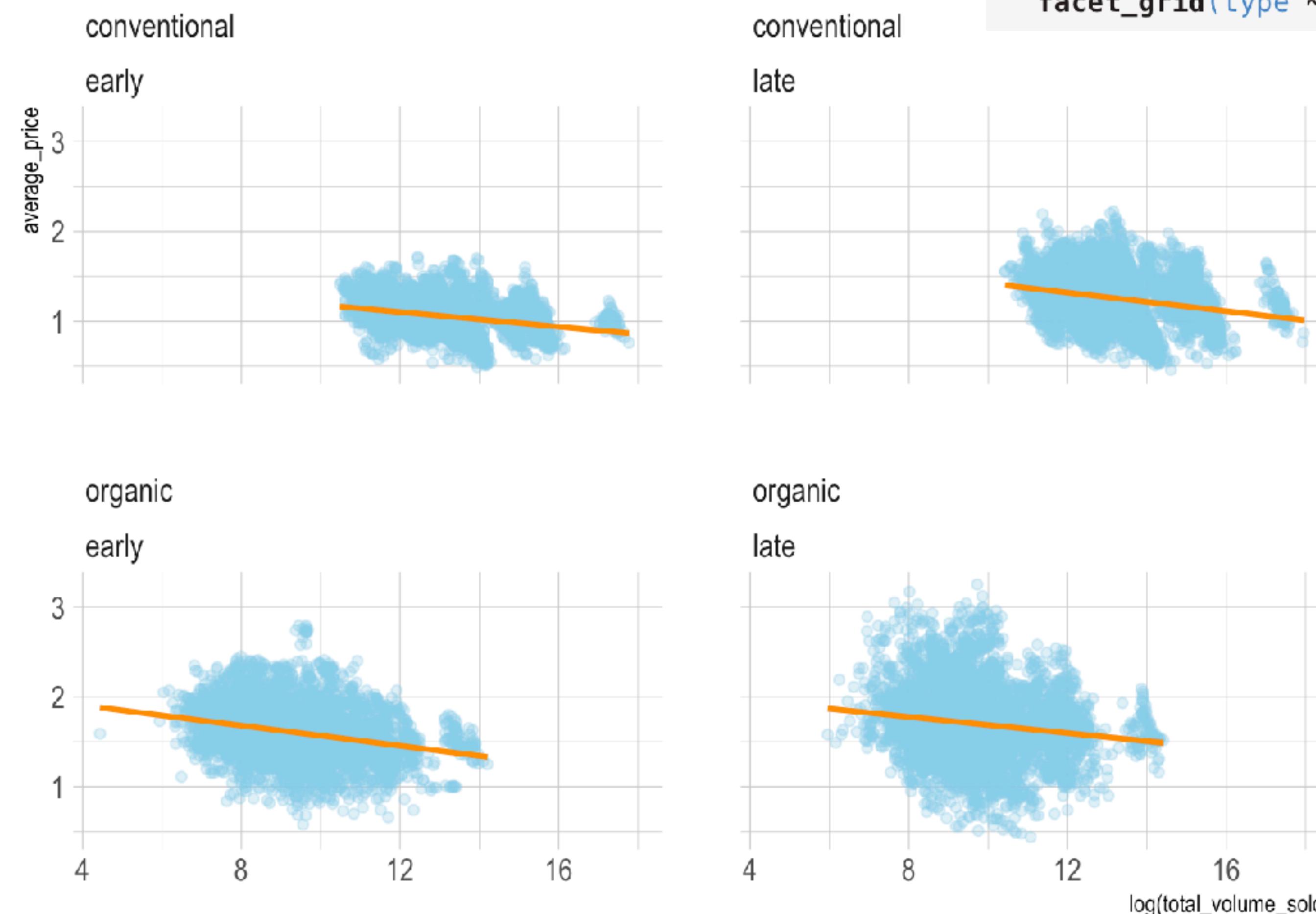




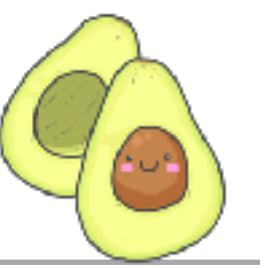
# Faceting



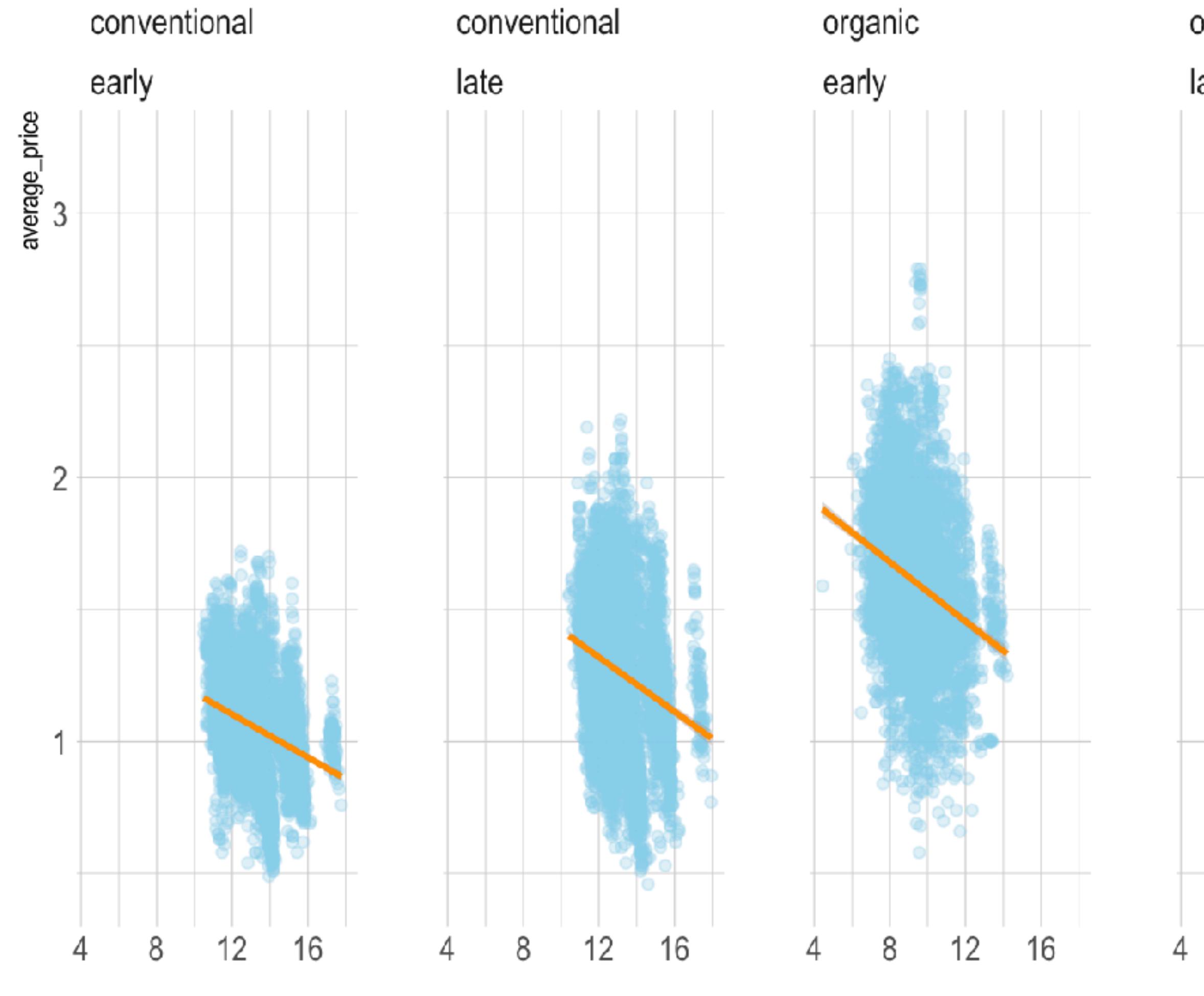
## FACET GRID



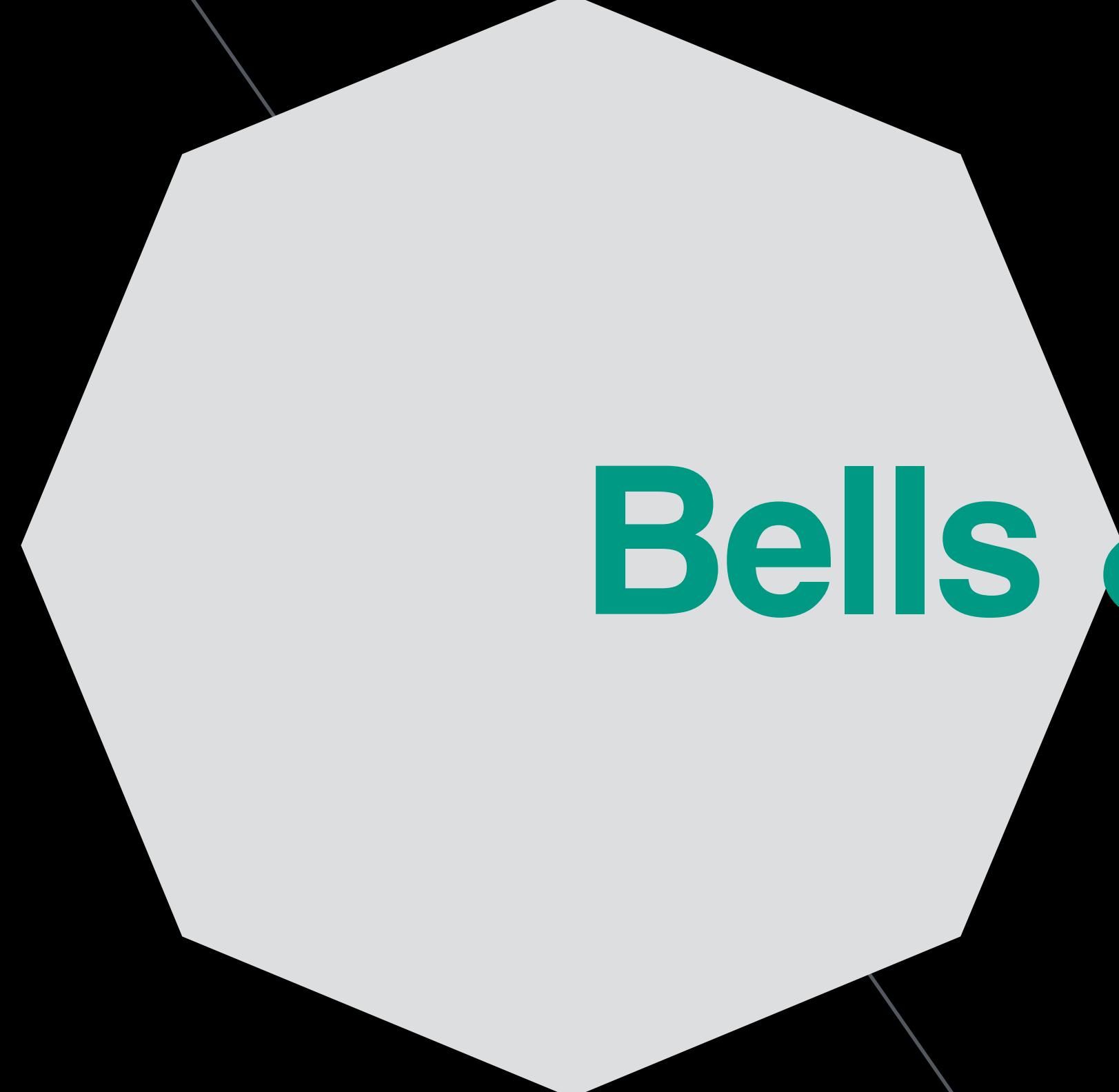
```
avocado_data_early_late %>%
  ggplot(aes(x = log(total_volume_sold), y = average_price)) +
  geom_point(alpha = 0.3, color = "skyblue") +
  geom_smooth(method = "lm", color = "darkorange") +
  facet_grid(type ~ early)
```



## Facet Wrap



```
avocado_data_early_late %>%
  ggplot(aes(x = log(total_volume_sold), y = average_price)) +
  geom_point(alpha = 0.3, color = "skyblue") +
  geom_smooth(method = "lm", color = "darkorange") +
  facet_wrap(type ~ early, nrow = 1)
```

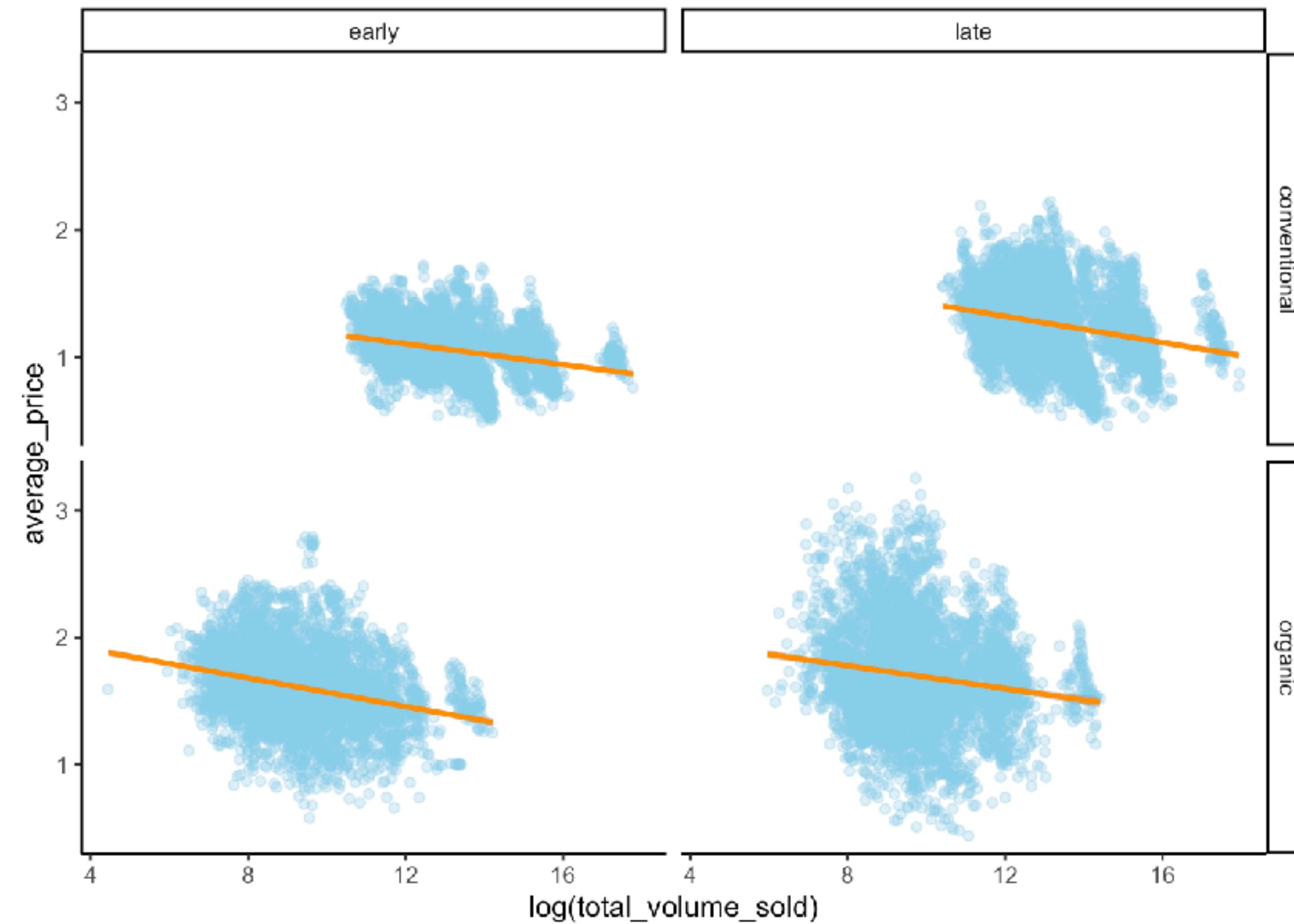


**Bells & whistles**

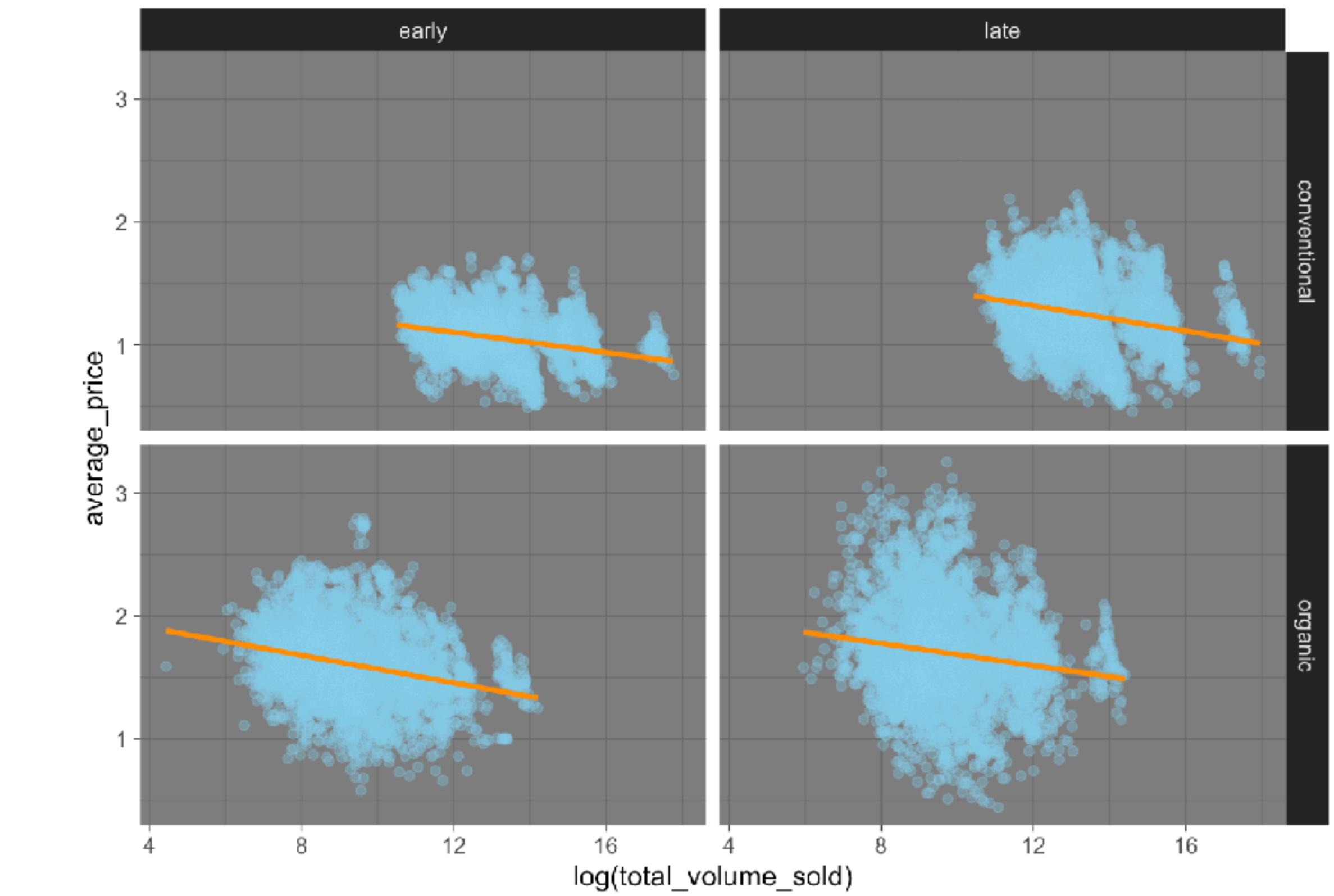


## READY-MADE THEMES

```
avocado_grid_plot + theme_classic()
```



```
avocado_grid_plot + theme_dark()
```





## TWEAKING AN EXISTING THEME

```
avocado_grid_plot + theme(plot.background = element_rect(fill = "darkgreen"))
```

