Relationships

Week 10

AEM 2850 / 5850 : R for Business Analytics
Cornell Dyson
Spring 2025

Acknowledgements: Andrew Heiss

Announcements

Reminders:

- Group project due April 18 (link)
  - We will set up group-specific workspaces on posit cloud for the project to allow simultaneous collaborative editing
  - Still: save your work early and often to avoid overwriting each other's work
  - Log out and close your posit cloud browser tab between sessions
  - Instructions to be posted on posit cloud and on canvas
  - Make a plan and start early!
- No homework-10 due to spring break

Questions before we get started?

Plan for today

Prologue: The dangers of dual y-axes

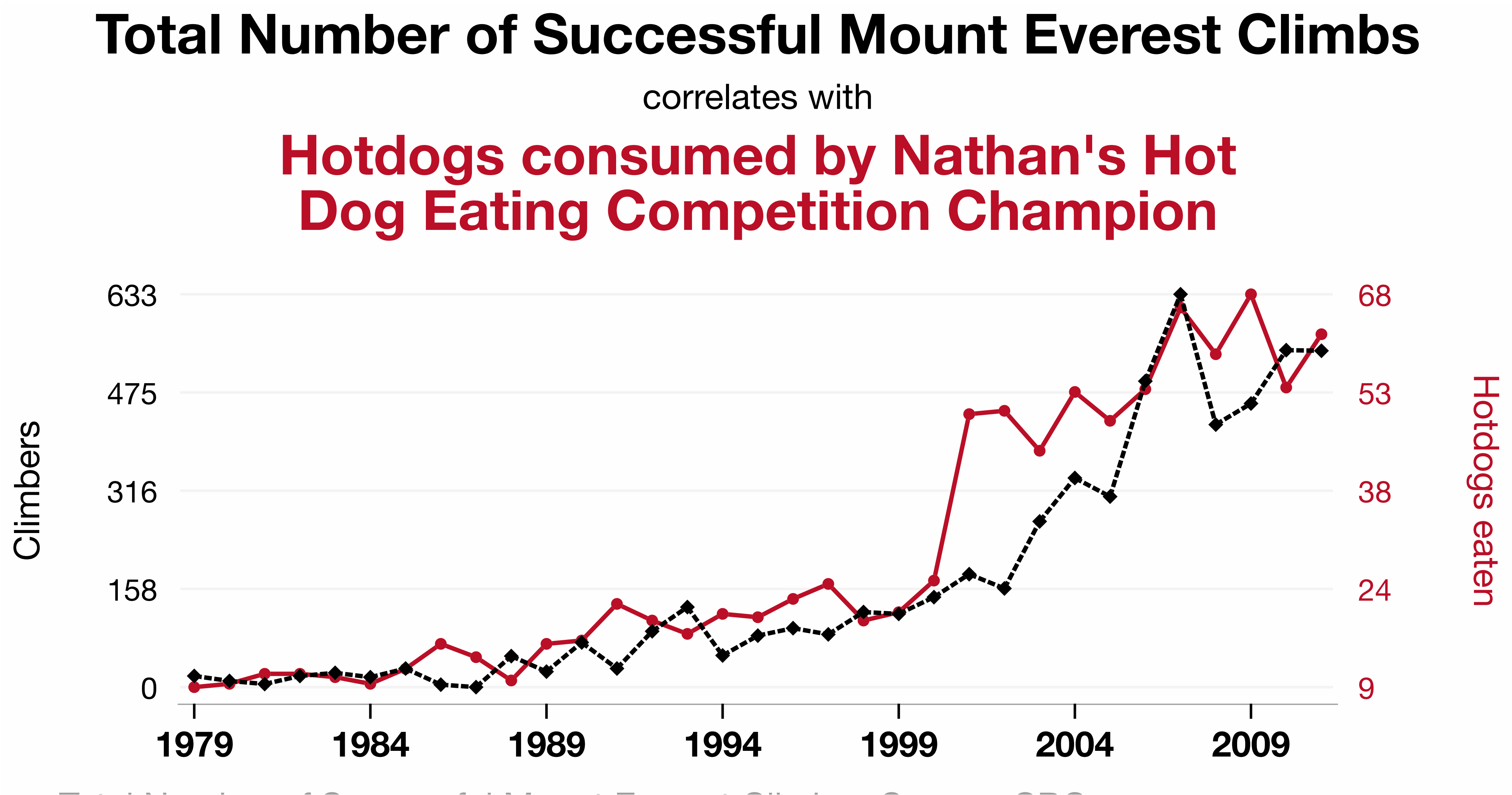
Visualizing relationships between a numerical and a categorical variable

Visualizing relationships between two numerical variables

- Visualizing correlations
- Visualizing regressions

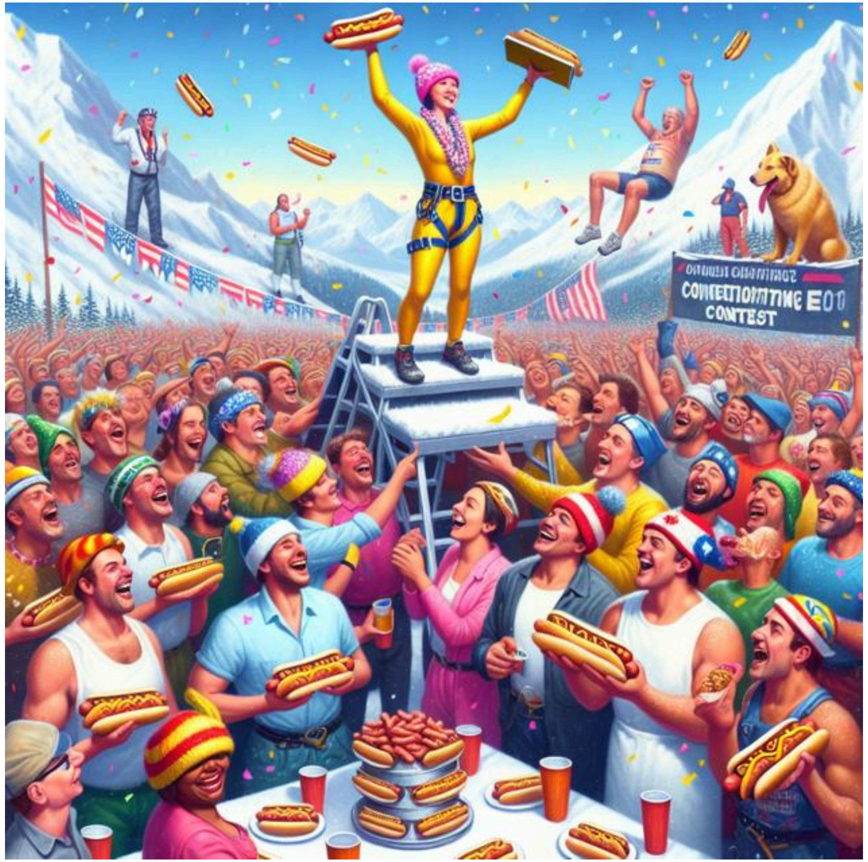
Prologue: The dangers of dual y-axes

Oh no!


Total Number of Successful Mount Everest Climbs
correlates with
Hotdogs consumed by Nathan's Hot Dog Eating Competition Champion
Climbers
633
475
316
158
0
Hotdogs eaten
68
53
38
24
9
1979
1984
1989
1994
1999
2004
2009

GPT 3.5 and DALL·E 3 explainer

"As the number of successful Mount Everest climbs rises, so does the peak appetite for adventure. This, in turn, creates a sausage-yetis-faction where competitors are relishing the thrill of the challenge like never before, and they're on a roll to claim the title. It's a summit showdown of epic proportions, where each contender is truly reaching their peak performance..."

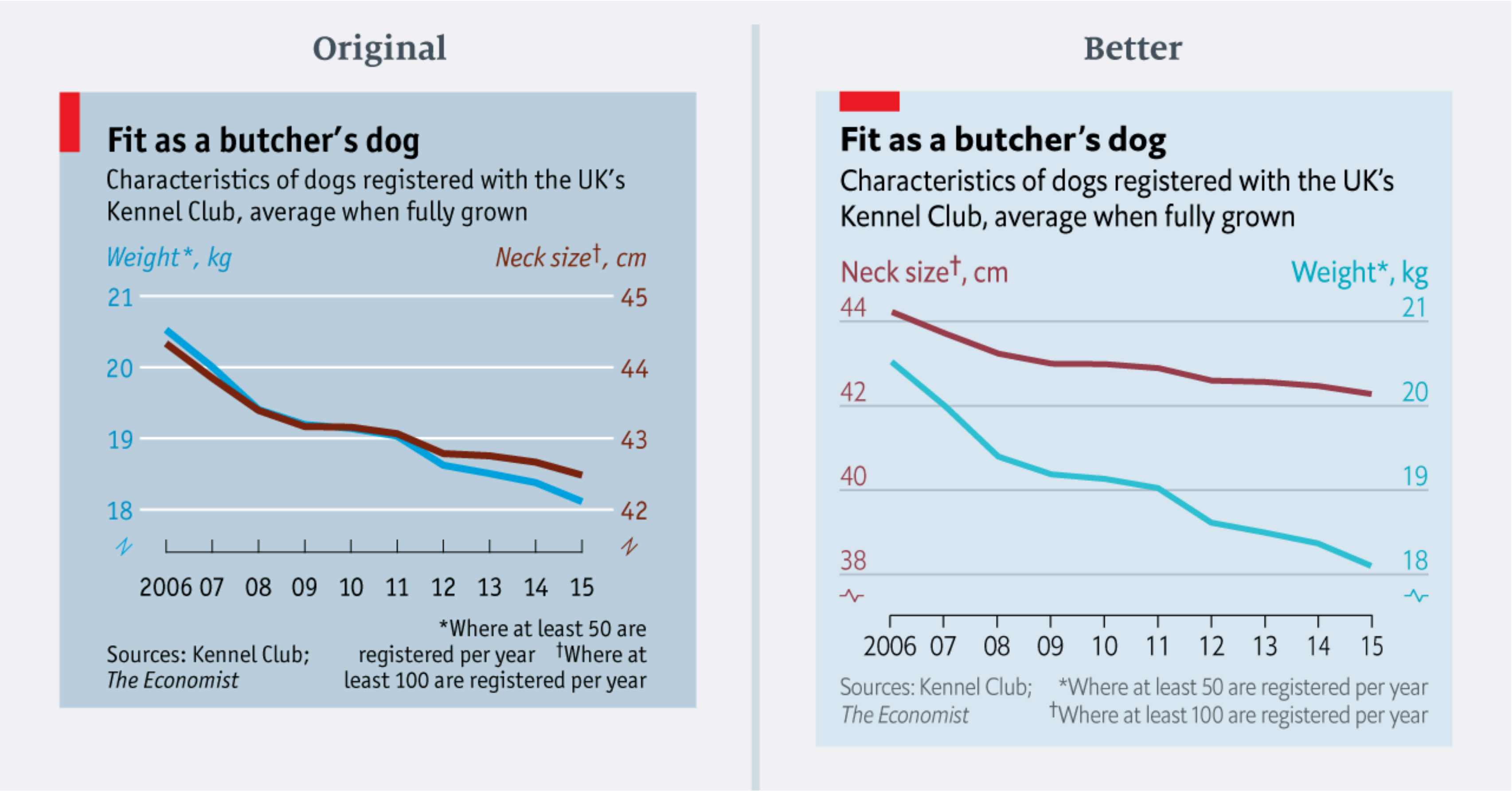


Why not use two y-axes?

You have to choose where the y-axes start and stop, which means…

…you can force the two trends to line up however you want.

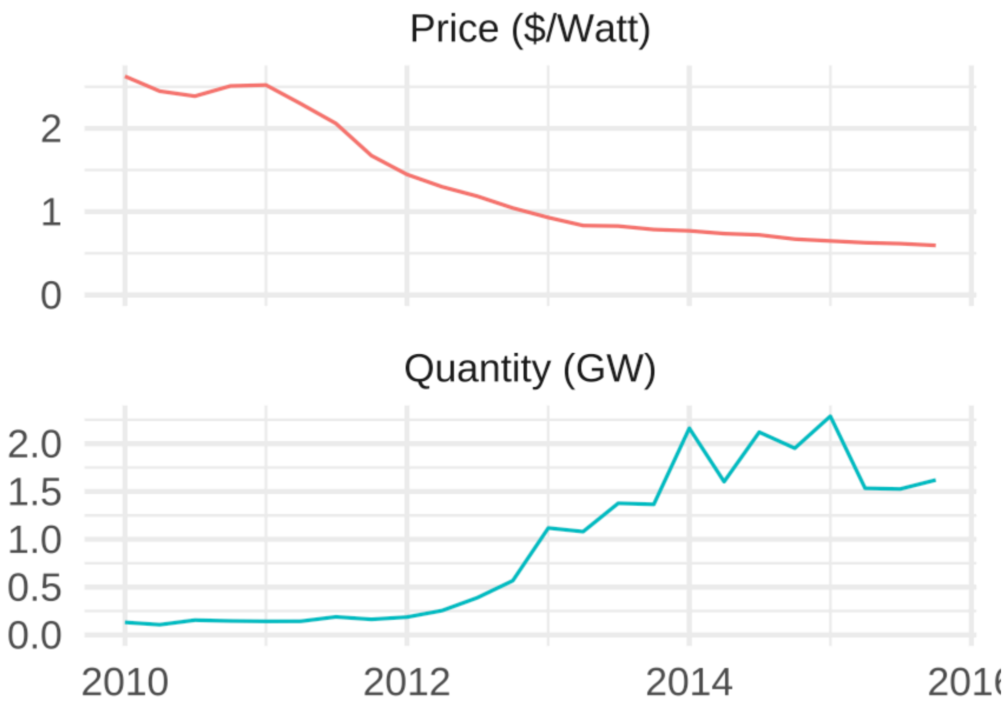
It even happens in *The Economist*!



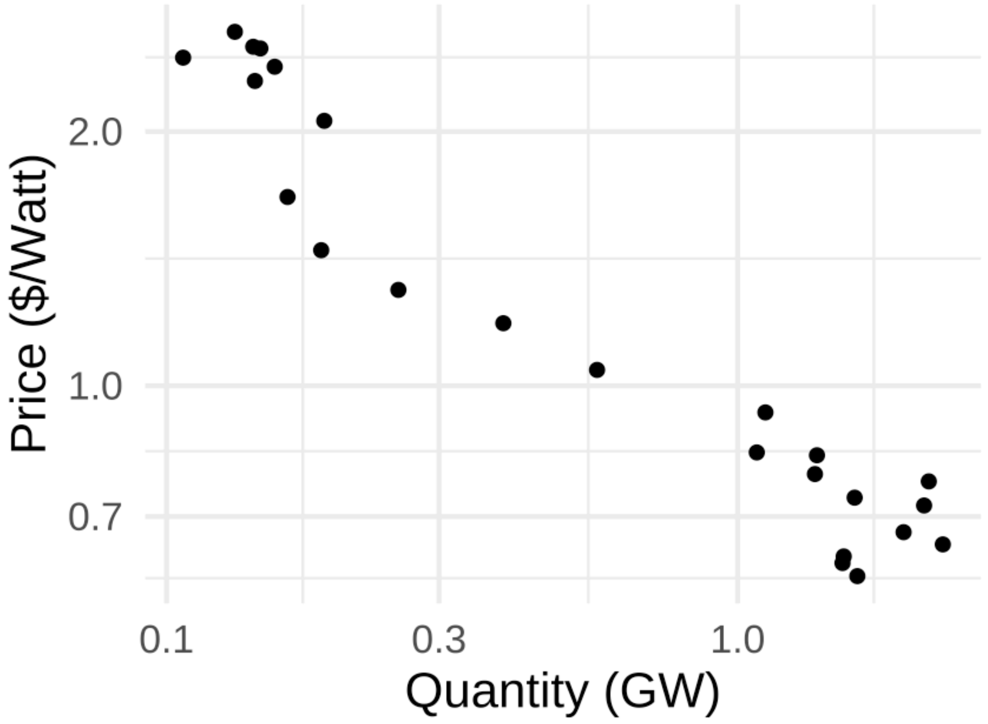
The revised axes ranges reflect a comparable proportional change

What could we do instead?

- Use multiple plots



- Use scatter plots



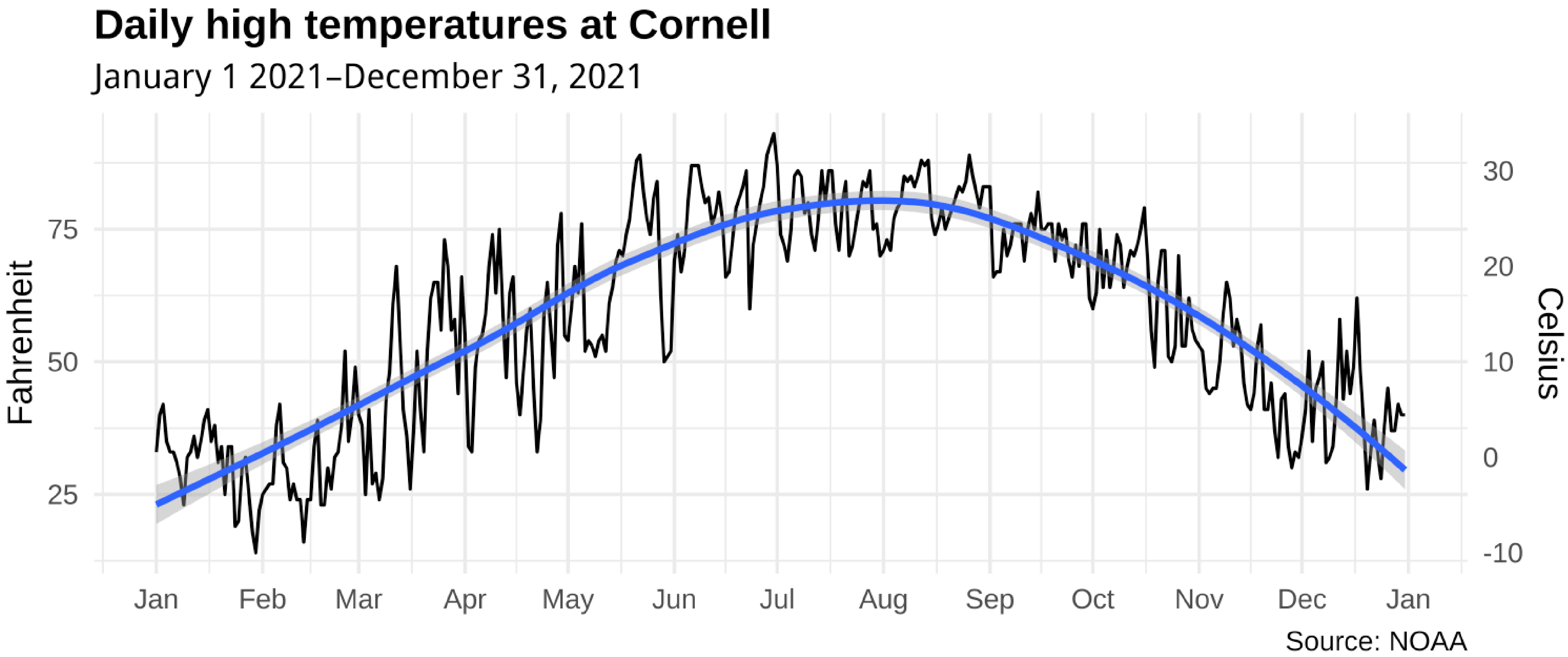
How could we make multiple plots in R?

1. **Facets** are great when using a common geometry (we've already seen that)

2. **Combining multiple plot objects** can be more flexible (see, e.g., `patchwork`)

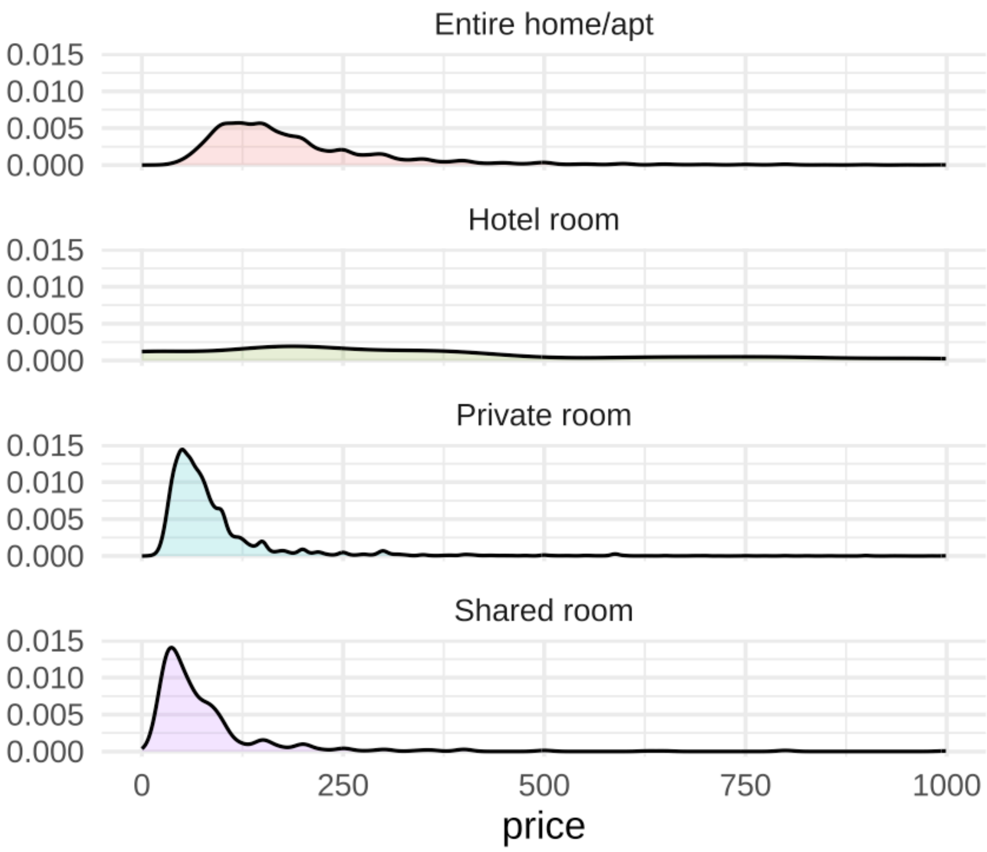
When are dual y-axes defensible?

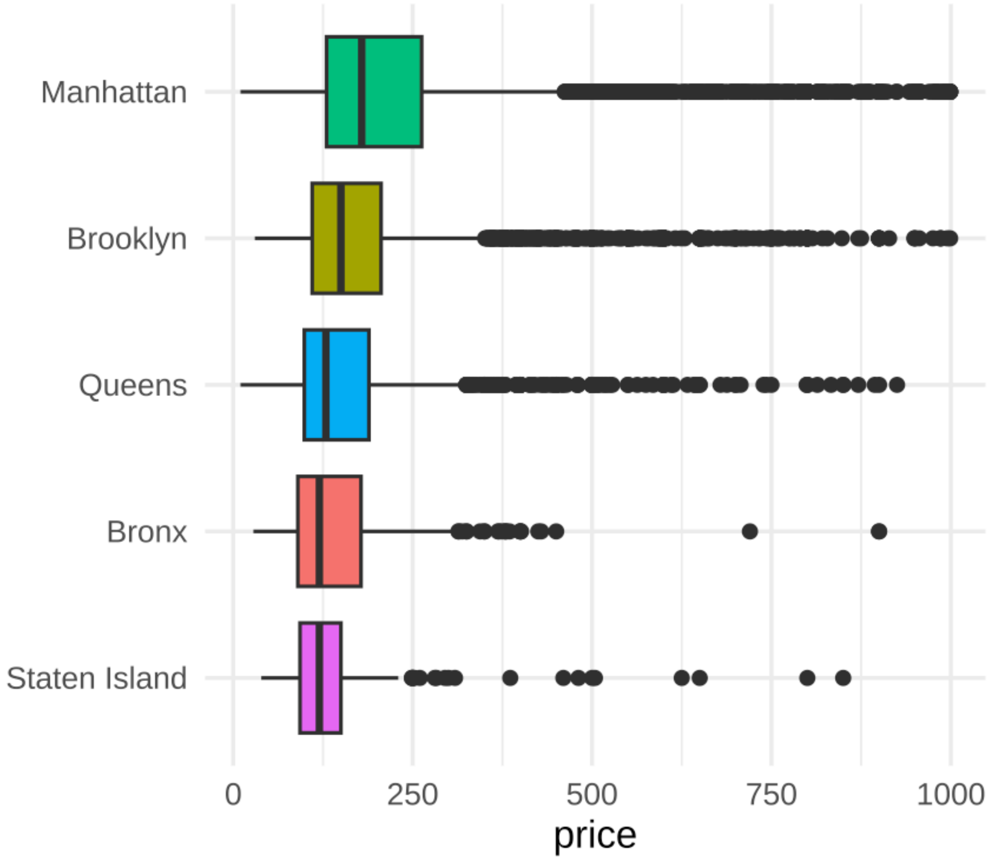
When the two axes measure the same thing (e.g., indexing, conversion, etc.)



Visualizing relationships between a numerical and a categorical variable

We already did this! When?





Visualizing relationships between two numerical variables

Visualizing correlations

What does "correlation" mean to you?

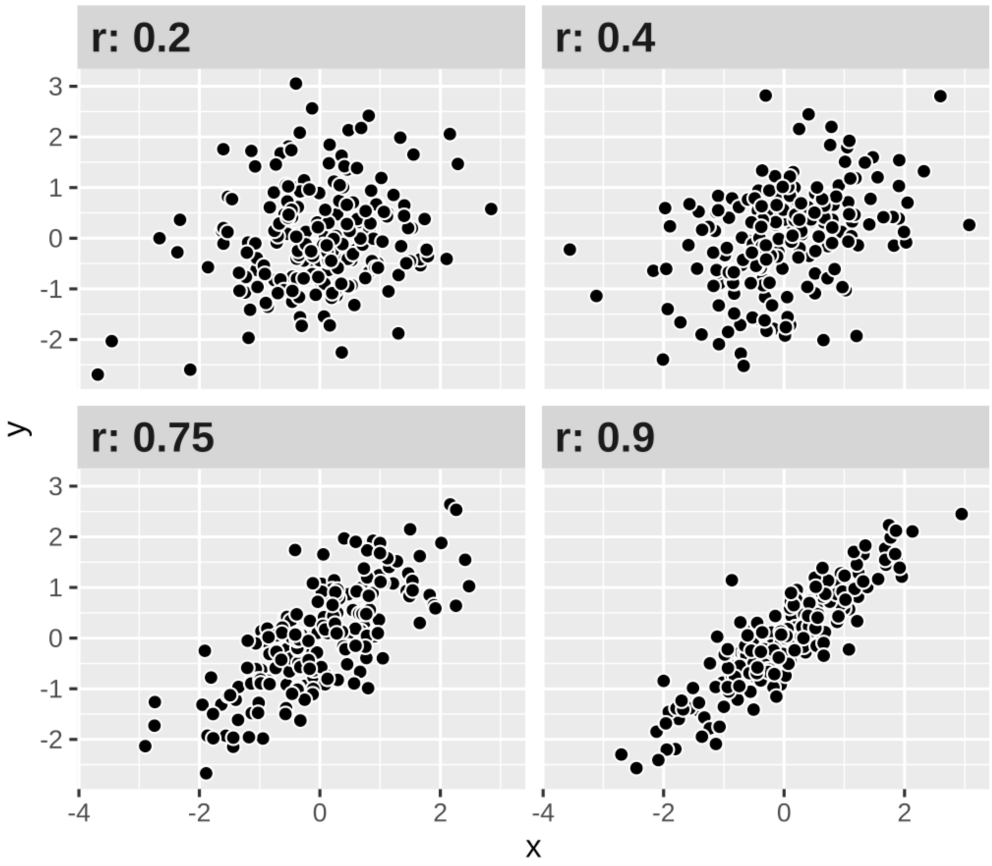
As the value of X goes up, Y is very / a little / not at all likely to go up (down)

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$$

Says nothing about *how much* Y changes when X changes

Correlation values

ρ	**Rough meaning**
±0.1–0.3	Weak
±0.3–0.5	Moderate
±0.5–0.8	Strong
±0.8–0.9	Very strong



Scatter plots

The humble scatter plot is often the best place to start when studying the association between two variables

Example: max and min temperature in Ithaca each day of the year

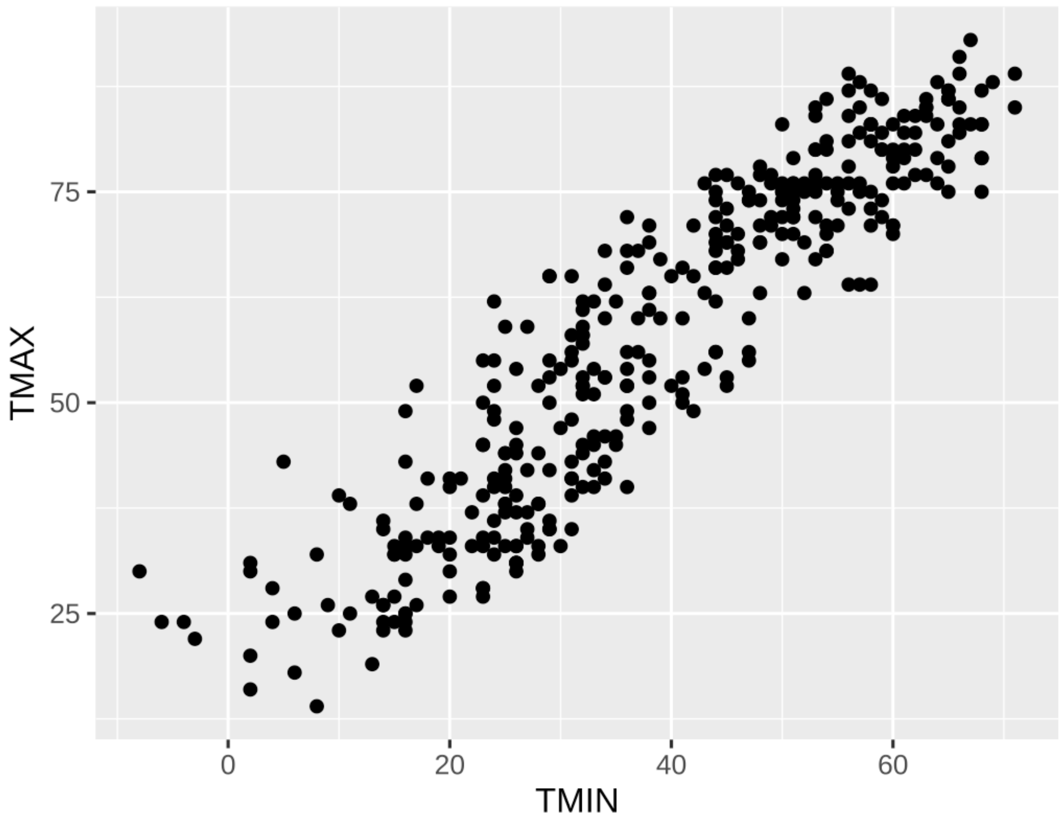
- Do you think they are highly correlated, somewhat correlated, or not at all correlated?
- What sign do you think this correlation has?
- How would you make a scatter plot of these data in R?

Scatter plots

```
ithaca_weather |>
  ggplot(aes(x = TMIN, y = TMAX)) +
  geom_point()
```

```
ithaca_weather |>
  summarize(cor(TMIN, TMAX))
```

```
## # A tibble: 1 × 1
##   `cor(TMIN, TMAX)`
##               <dbl>
## 1             0.919
```



Strong positive correlation

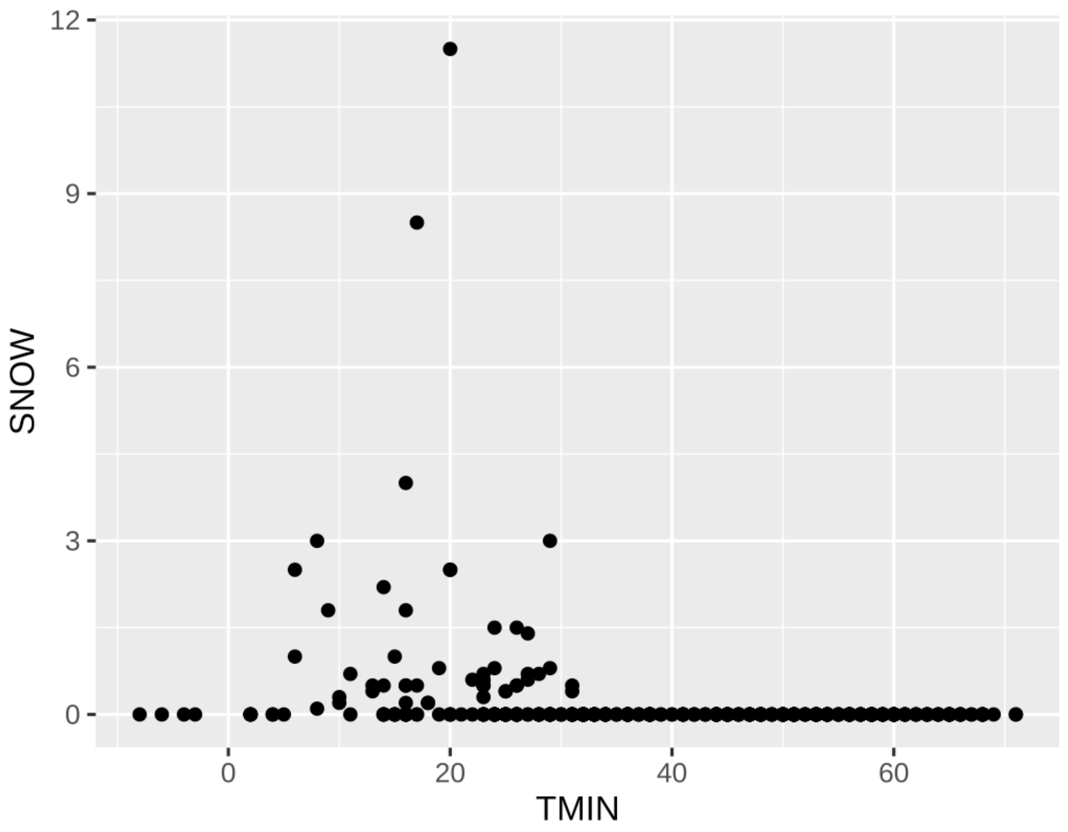
What about min temp and snowfall?

```
ithaca_weather |>
  ggplot(aes(x = TMIN, y = SNOW)) +
  geom_point()
```

```
ithaca_weather |>
  summarize(cor(TMIN, SNOW))
```

```
## # A tibble: 1 × 1
##   `cor(TMIN, SNOW)`
##               <dbl>
## 1            -0.239
```

Weak negative correlation



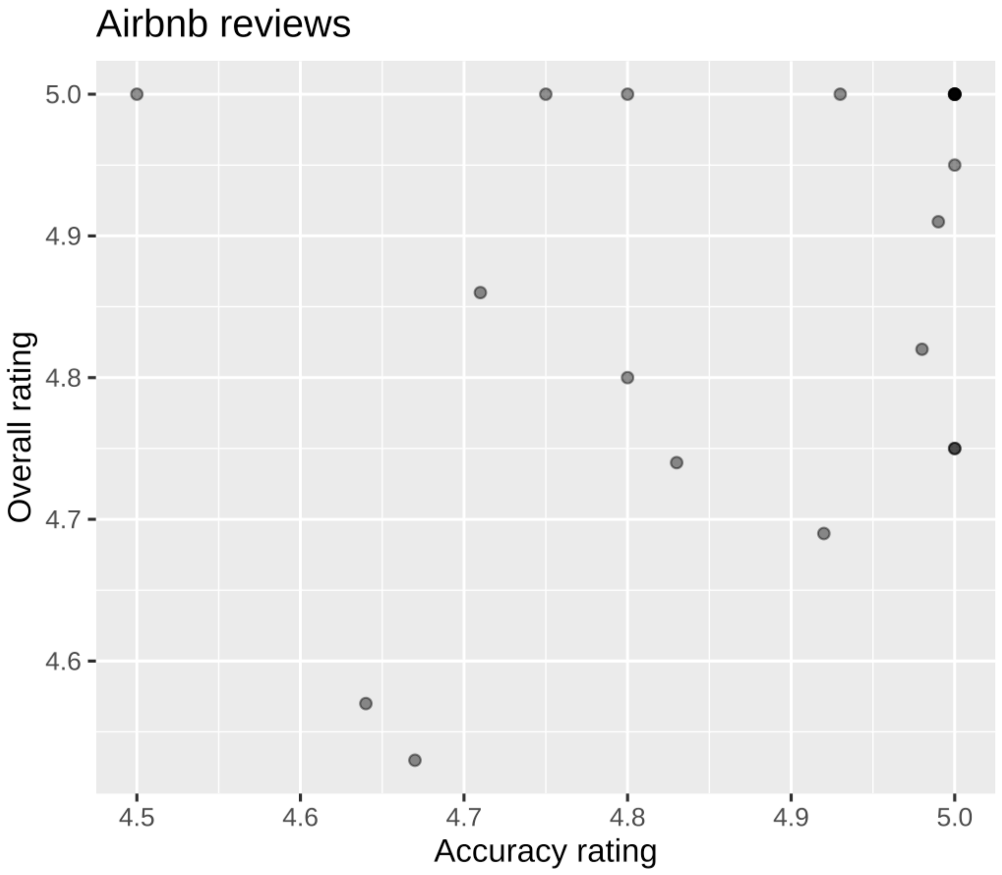
Visualizing regressions

Linear regression reminder

$$y = \beta_0 + \beta_1 x_1 + \varepsilon$$

y	Outcome variable (DV)
x_1	Explanatory variable (IV)
β_1	Slope
β_0	y-intercept
ε	Error (residuals)

Linear regression is just drawing lines



Building models in R

Base R has some basic modeling tools:

```r
<MODEL> <- lm(<Y> ~ <X>, data = <DATA>) # use lm to fit simple linear models

summary(<MODEL>) # see model details
```

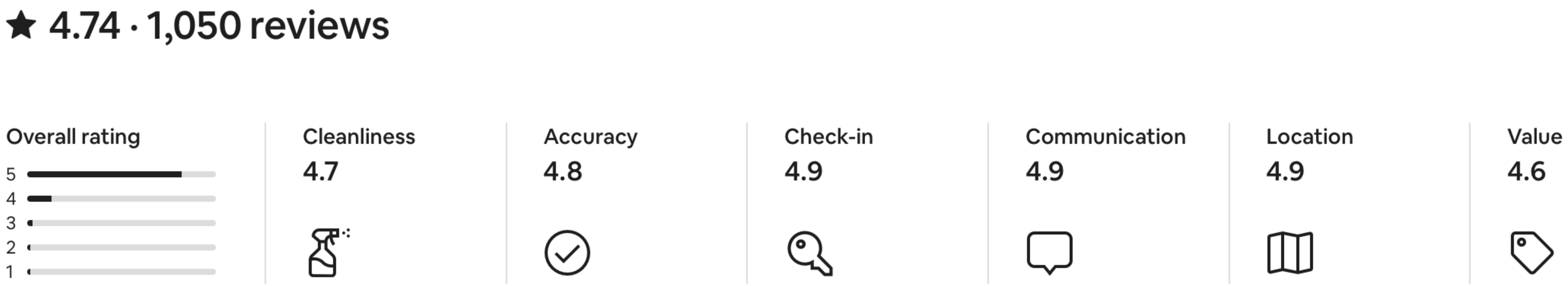
The `broom` package provides helpful tools for tidying model output:

```r
library(broom)

# convert model estimates to a data frame for plotting
tidy(<MODEL>)

# return a data frame that includes predictions, residuals, etc.
augment(<MODEL>)
```

Modeling Airbnb reviews

Let's use some real-world data to explore linear regression

Put yourself in the shoes of an Airbnb host trying to decide how much to invest in improvements across these categories:



Let's see how well "accuracy" reviews predict an Airbnb's overall rating

Modeling Airbnb reviews

$$\text{rating} = \beta_0 + \beta_1 \text{accuracy} + \varepsilon$$

```r
review_model <- lm(
  rating ~ accuracy,
  data = reviews
  )
```

Note how we didn't write anything for the β_0 or ε terms

What do you think the sign on β_1 is?

How large do you think β_1 is?

```r
review_model
```

```
##
## Call:
## lm(formula = rating ~ accuracy, data = reviews)
##
## Coefficients:
## (Intercept)     accuracy
##      0.7590       0.8271
```

Modeling Airbnb reviews

```r
summary(review_model)
```

```
##
## Call:
## lm(formula = rating ~ accuracy, data = reviews)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.8943 -0.0648  0.0608  0.1057  4.2410
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.758952   0.017156   44.24   <2e-16 ***
## accuracy    0.827067   0.003597  229.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2996 on 28159 degrees of freedom
##   (10116 observations deleted due to missingness)
## Multiple R-squared:  0.6525,    Adjusted R-squared:  0.6525
## F-statistic: 5.287e+04 on 1 and 28159 DF,  p-value: < 2.2e-16
```

Modeling Airbnb reviews

```r
tidy(review_model, conf.int = TRUE)
```

```
## # A tibble: 2 × 7
##   term        estimate std.error statistic p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 (Intercept)    0.759   0.0172       44.2       0    0.725     0.793
## 2 accuracy       0.827   0.00360     230.        0    0.820     0.834
```

Interpretation for a continuous variable

$$y = \beta_0 + \beta_1 x_1 + \varepsilon$$

On average, a one unit increase in x_1 is *associated* with a β_1 change in y

$$\text{rating} = \beta_0 + \beta_1 \text{accuracy} + \varepsilon$$

$$\widehat{\text{rating}} = 0.76 + 0.83 \times \text{accuracy}$$

On average, a one unit increase in accuracy rating is associated with 0.83 higher overall rating

This is easy to visualize: it's a line!

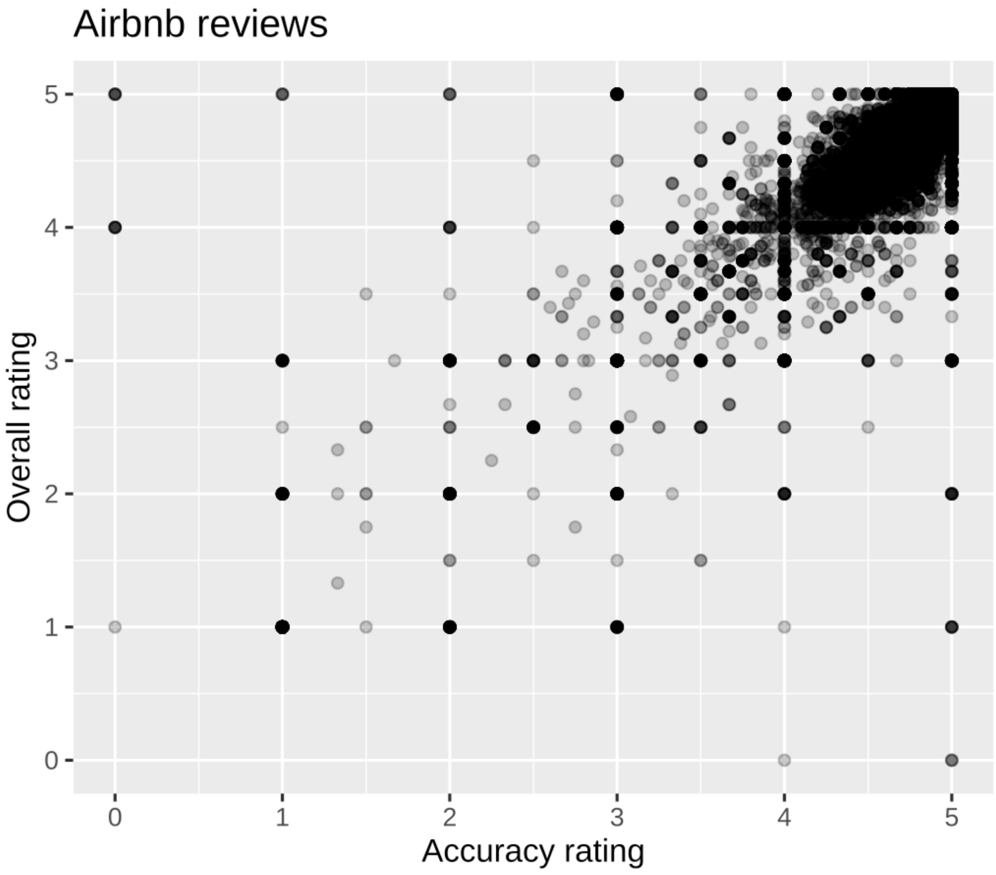
Visualization of a continuous variable

```
tidy(review_model) |>
  select(term, estimate)
```

```
## # A tibble: 2 × 2
##   term        estimate
##   <chr>          <dbl>
## 1 (Intercept)    0.759
## 2 accuracy       0.827
```

$$\widehat{\text{rating}} = 0.76 + 0.83 \times \text{accuracy}$$

Note: this is an example where `alpha` helps with overplotting



Visualization of a continuous variable

```
tidy(review_model) |>
  select(term, estimate)
```

```
## # A tibble: 2 × 2
##   term        estimate
##   <chr>          <dbl>
## 1 (Intercept)    0.759
## 2 accuracy       0.827
```

$$\widehat{\text{rating}} = 0.76 + 0.83 \times \text{accuracy}$$

Note: this is an example where `alpha` helps with overplotting

Visualization of a continuous variable

Recall: `geom_smooth(method = "lm")` allows us to skip the estimation step!

```
reviews |>
  ggplot(aes(x = accuracy, y = rating)) +
  geom_point(alpha = 0.25) +
  geom_smooth(
    method = "lm",    # smoothing function
    se = FALSE        # omit confidence bands
  )
```

Multiple regression

We're not limited to just one explanatory variable!

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

```
review_model_big <- lm(rating ~ accuracy + cleanliness +
                         communication + location +
                         checkin + value,
                       data = reviews)
```

$$\begin{aligned}
\widehat{\text{rating}} = & \hat{\beta}_0 + \hat{\beta}_1 \text{accuracy} + \hat{\beta}_2 \text{cleanliness} + \\
& \hat{\beta}_3 \text{communication} + \hat{\beta}_4 \text{location} + \\
& \hat{\beta}_5 \text{checkin} + \hat{\beta}_6 \text{value}
\end{aligned}$$

Multiple regression

We started by estimating this **univariate** (aka **bivariate**) regression model:

$$\text{rating} = \beta_0 + \beta_1 \text{accuracy} + \varepsilon$$

Now we are estimating this **multivariate** regression model:

$$\begin{aligned}\text{rating} = \beta_0 &+ \beta_1 \text{accuracy} + \beta_2 \text{cleanliness} + \\ &\beta_3 \text{communication} + \beta_4 \text{location} + \\ &\beta_5 \text{checkin} + \beta_6 \text{value} + \varepsilon\end{aligned}$$

Why are we doing this? Wasn't it complicated enough already?!

We want to use these data to inform our Airbnb hosting strategy. What are the pros and cons of the two models for this purpose?

Multiple regression

Will the coefficient on `accuracy` will be smaller, larger, or the same? Why?

```
tidy(review_model_big, conf.int = TRUE)
```

```
## # A tibble: 7 × 7
##   term          estimate std.error statistic   p.value conf.low conf.high
##   <chr>            <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)    -0.124    0.0178      -6.96 3.43e- 12  -0.159    -0.0892
## 2 accuracy        0.217    0.00531     40.8  0            0.206     0.227
## 3 cleanliness     0.227    0.00356     63.9  0            0.220     0.234
## 4 communication   0.169    0.00507     33.4  1.45e-239    0.159     0.179
## 5 location        0.0384   0.00428      8.97 3.25e- 19    0.0300    0.0468
## 6 checkin         0.0578   0.00521     11.1  1.37e- 28    0.0476    0.0680
## 7 value           0.313    0.00476     65.8  0            0.304     0.323
```

$$
\begin{aligned}
\widehat{\text{rating}} = &-0.12 + 0.22 \times \text{accuracy} + 0.23 \times \text{cleanliness} + \\
&0.17 \times \text{communication} + 0.04 \times \text{location} + \\
&0.06 \times \text{checkin} + 0.31 \times \text{value}
\end{aligned}
$$

Interpretation for continuous variables

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

Holding everything else constant, a one unit increase in x_n is *associated* with a β_n change in y, on average

$$\begin{aligned}\widehat{\text{rating}} = & -0.12 + 0.22 \times \text{accuracy} + 0.23 \times \text{cleanliness} + \\ & 0.17 \times \text{communication} + 0.04 \times \text{location} + \\ & 0.06 \times \text{checkin} + 0.31 \times \text{value}\end{aligned}$$

On average, a one unit increase in accuracy rating is associated with 0.22 higher overall rating, holding everything else constant

For the earlier model we had said

> On average, a one unit increase in accuracy rating is associated with 0.83 higher overall rating

Good luck visualizing all this!

You can't just draw a single line! There are too many moving parts!

Main challenges

Each coefficient has its own estimate and standard errors

Solution: Plot the coefficients and their errors with a *coefficient plot*

The results change as you move sliders (continuous variables) up and down or flip switches (categorical variables) on and off

Solution: Plot the *marginal effects* for the coefficients you're interested in

Coefficient plots

Convert the model results to a data frame with `tidy()`

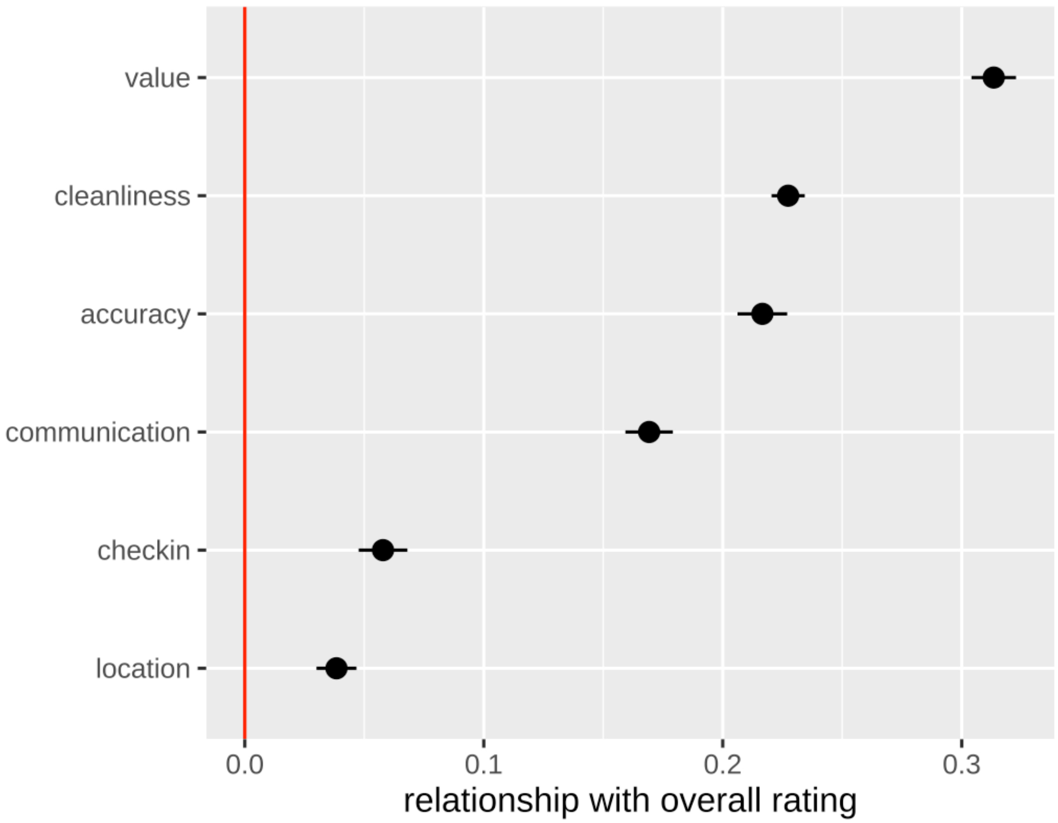
```
# tidy the estimates (reformatting names is not required)
review_coefs <- tidy(
  review_model_big, # get the model's coefficients
  conf.int = TRUE   # include confidence intervals
) |>
  filter(term!="(Intercept)")

review_coefs
```

```
## # A tibble: 6 × 7
##   term          estimate std.error statistic   p.value conf.low conf.high
##   <chr>            <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 accuracy        0.217    0.00531     40.8  0            0.206     0.227
## 2 cleanliness     0.227    0.00356     63.9  0            0.220     0.234
## 3 communication   0.169    0.00507     33.4  1.45e-239    0.159     0.179
## 4 location        0.0384   0.00428      8.97 3.25e- 19    0.0300    0.0468
## 5 checkin         0.0578   0.00521     11.1  1.37e- 28    0.0476    0.0680
## 6 value           0.313    0.00476     65.8  0            0.304     0.323
```

Coefficient plots

Plot the point estimate and confidence intervals with `geom_pointrange()`

```
review_coefs |>
  ggplot(aes(x = estimate,
             y = fct_reorder(term, estimate))) +
  geom_pointrange(aes(xmin = conf.low,
                      xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red") +
  labs(x = "relationship with overall rating",
       y = NULL)
```



What do you take away from this?

Should this inform where you decide to focus your investment as a host?

Marginal effects plots

Remember we interpret individual coefficients while holding others constant

We move one slider while leaving all the other sliders and switches alone

The same principle applies to visualizing a variable's effect

Plug a bunch of values into the model and find the predicted outcome

Plot the values and predicted outcome

Marginal effects plots

Create a data frame of values you want to manipulate and values you want to hold constant

The data frame must include all the explanatory variables in the model

Marginal effects plots

```
reviews_new_data <- reviews |>
  select(rating, accuracy, cleanliness, checkin, communication, location, value) |>
  mutate(   # use mutate to modify existing variables
    across( # transform multiple columns the same way
      c(cleanliness, checkin, communication, location, value), # specify columns
      \(x) mean(x, na.rm = TRUE)                               # specify transformation
    )
  )
```

`\(x) mean(x, na.rm = TRUE)` is a small anonymous "lambda function" that operates on inputs just like a named function. This function:

1. takes in a column of the data, `x`
2. applies the `mean` function to `x` (ignoring missing values)
3. returns the result

This function is applied to each column, with its output populating each column

Marginal effects plots

Here is what the resulting data frame looks like:

```
reviews_new_data
```

```
## # A tibble: 38,277 × 7
##    rating accuracy cleanliness checkin communication location value
##     <dbl>    <dbl>       <dbl>   <dbl>         <dbl>    <dbl> <dbl>
##  1   4.7      4.72        4.61    4.81          4.81     4.75  4.65
##  2   4.45     4.58        4.61    4.81          4.81     4.75  4.65
##  3   4.52     4.22        4.61    4.81          4.81     4.75  4.65
##  4   5        5           4.61    4.81          4.81     4.75  4.65
##  5   4.21     4.21        4.61    4.81          4.81     4.75  4.65
##  6   4.91     4.83        4.61    4.81          4.81     4.75  4.65
##  7   4.7      4.71        4.61    4.81          4.81     4.75  4.65
##  8   4.56     4.59        4.61    4.81          4.81     4.75  4.65
##  9  NA       NA           4.61    4.81          4.81     4.75  4.65
## 10   4.88     4.81        4.61    4.81          4.81     4.75  4.65
## # ℹ 38,267 more rows
```

Marginal effects plots

Plug each of those rows of data into the model with `augment()`

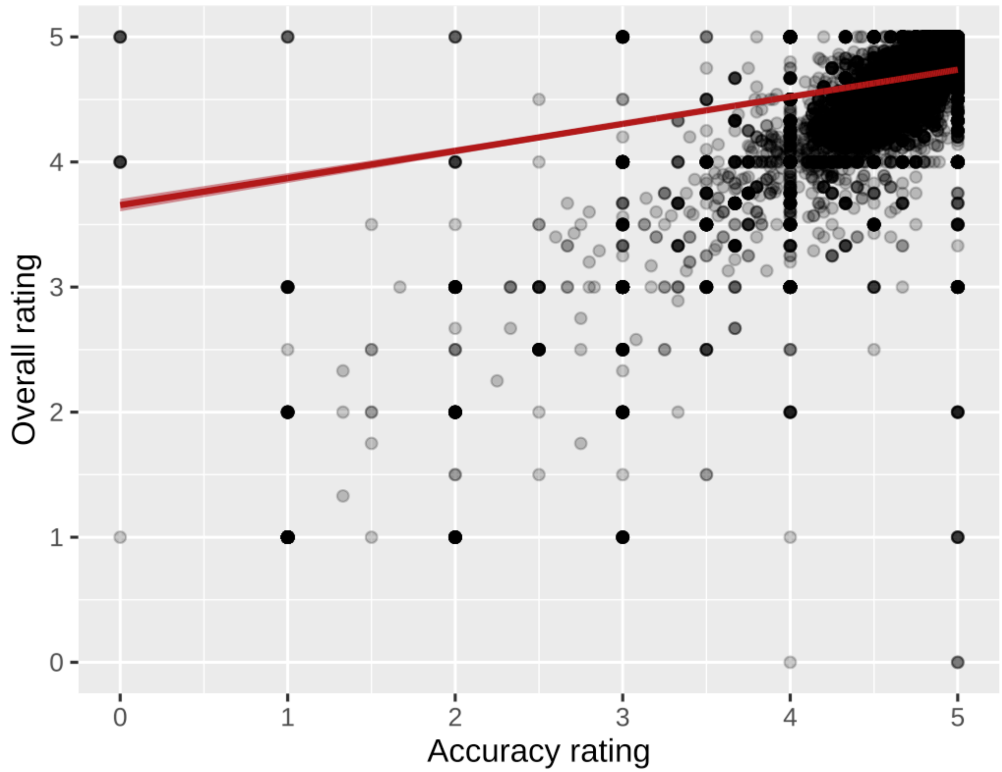
```
predicted_reviews <- augment(
  review_model_big,              # our estimated model
  newdata = reviews_new_data,    # our new data for plotting
  interval = "confidence"        # add confidence intervals
  )

head(predicted_reviews)
```

```
## # A tibble: 6 × 11
##   rating accuracy cleanliness checkin communication location value .fitted
##    <dbl>    <dbl>       <dbl>   <dbl>         <dbl>    <dbl> <dbl>   <dbl>
## 1   4.7      4.72        4.61    4.81          4.81     4.75  4.65    4.68
## 2   4.45     4.58        4.61    4.81          4.81     4.75  4.65    4.65
## 3   4.52     4.22        4.61    4.81          4.81     4.75  4.65    4.57
## 4   5        5           4.61    4.81          4.81     4.75  4.65    4.74
## 5   4.21     4.21        4.61    4.81          4.81     4.75  4.65    4.57
## 6   4.91     4.83        4.61    4.81          4.81     4.75  4.65    4.70
## # ℹ 3 more variables: .lower <dbl>, .upper <dbl>, .resid <dbl>
```

Marginal effects plots

Plot the fitted values for each row

```
mfx_plot <- predicted_reviews |>
  ggplot(aes(x = accuracy, y = rating)) +
  geom_point(alpha = 0.25) +
  geom_line( # multivariate predictions
    aes(y = .fitted),
    color = "#B31B1B",
    linewidth = 1
  ) +
  geom_ribbon( # confidence intervals
    aes(ymin = .lower, ymax = .upper),
    fill = "#B31B1B",
    alpha = 0.5
  ) +
  labs(x = "Accuracy rating",
       y = "Overall rating")
mfx_plot
```



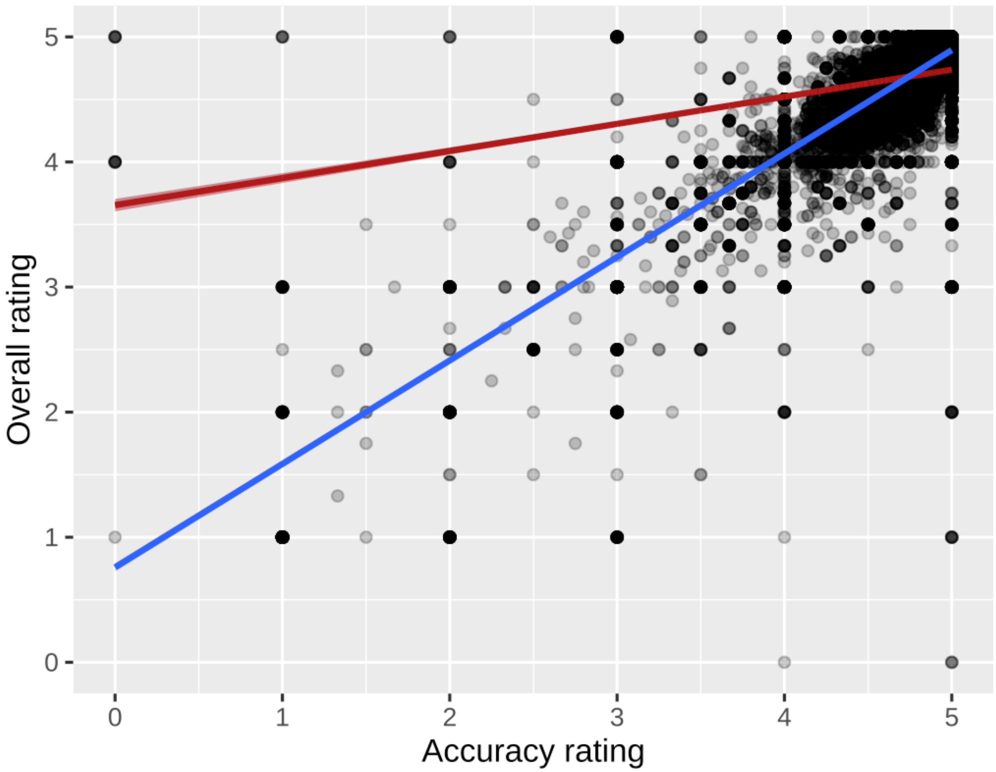
Marginal effects plots

How does this regression line compare to our univariate regression line?

```r
mfx_plot +
  # add the univariate regression line
  geom_smooth(method = "lm")
```

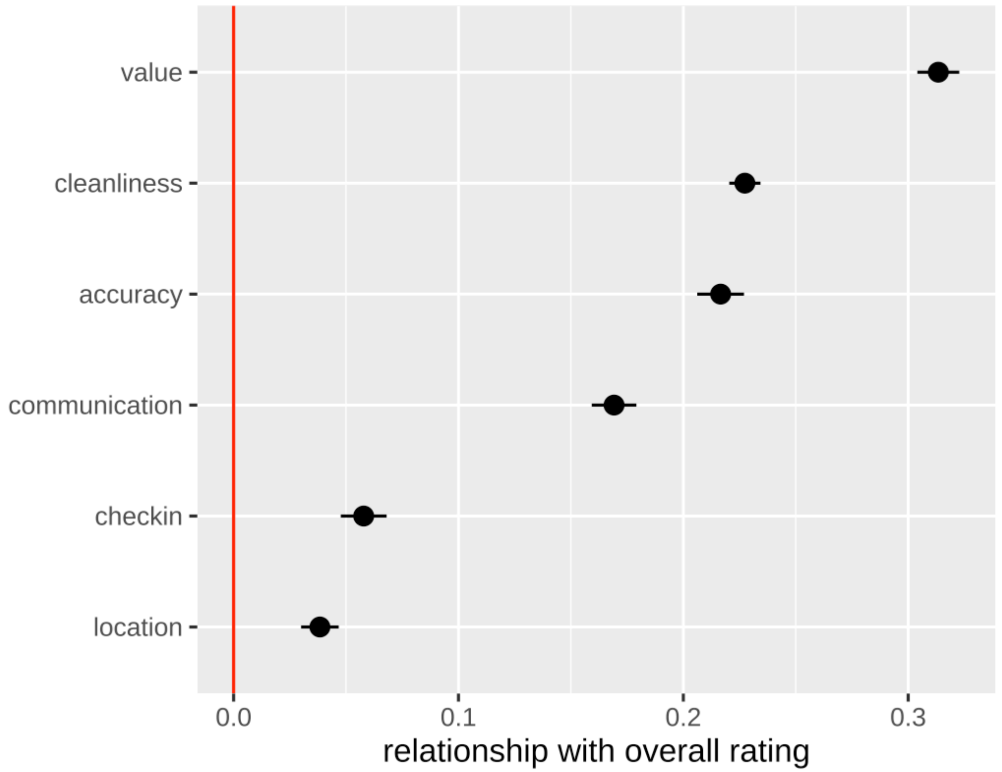
What do you take away from this?

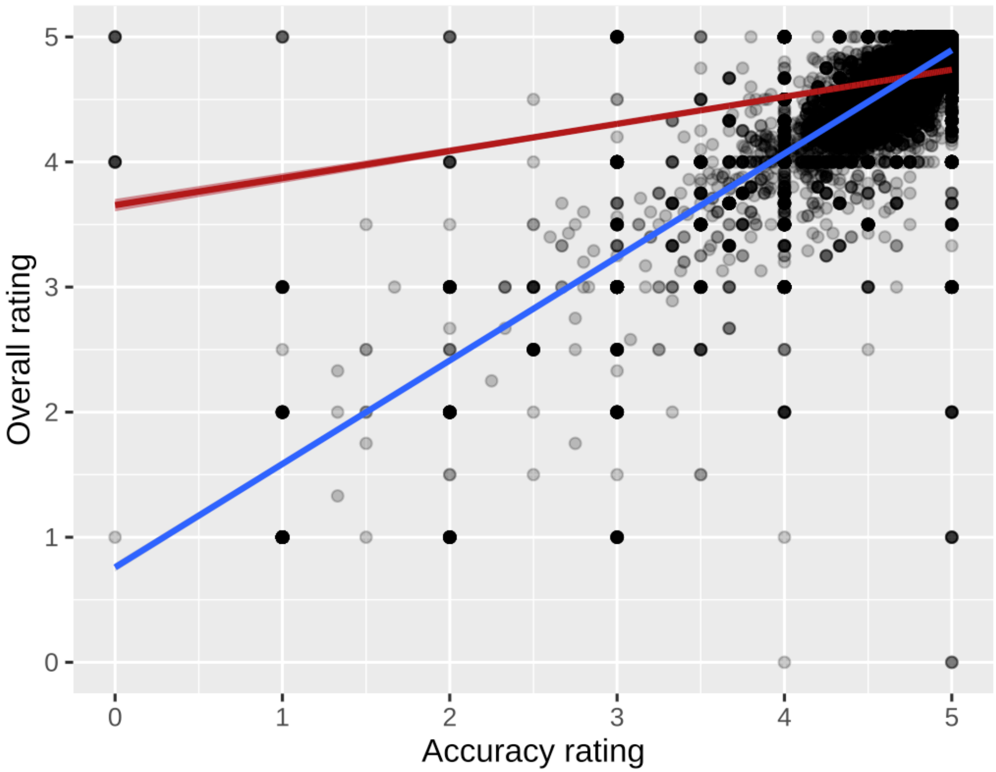
Should this affect how much you invest in accuracy?



Stepping back

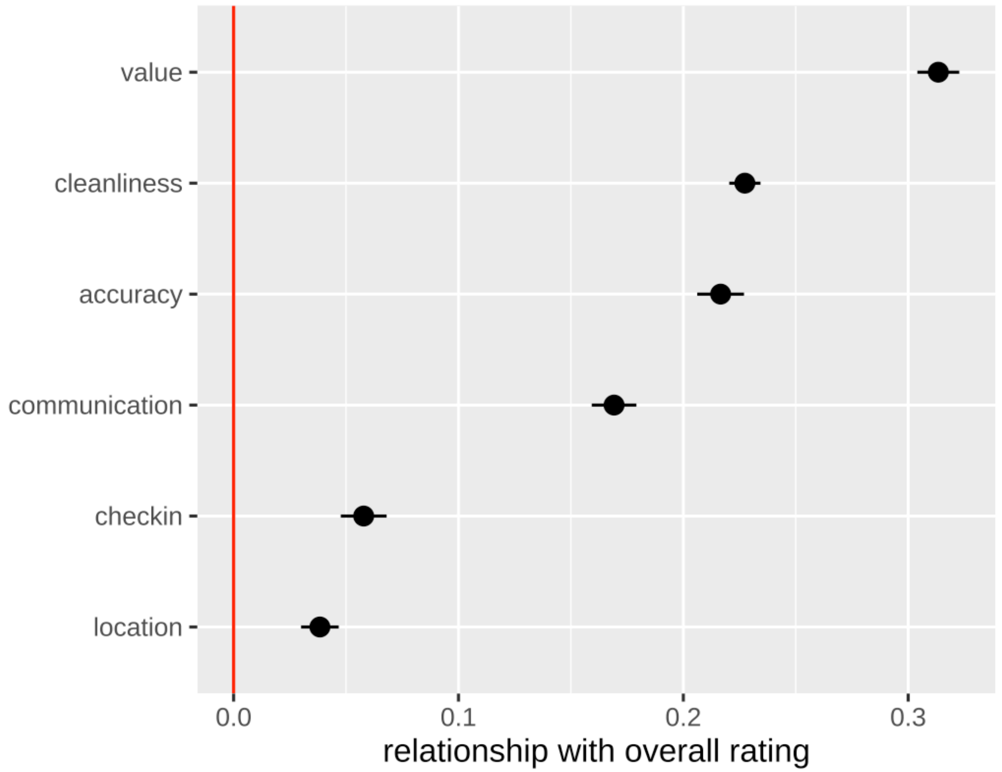
Which of these plots would be more useful to Airbnb hosts? Why?

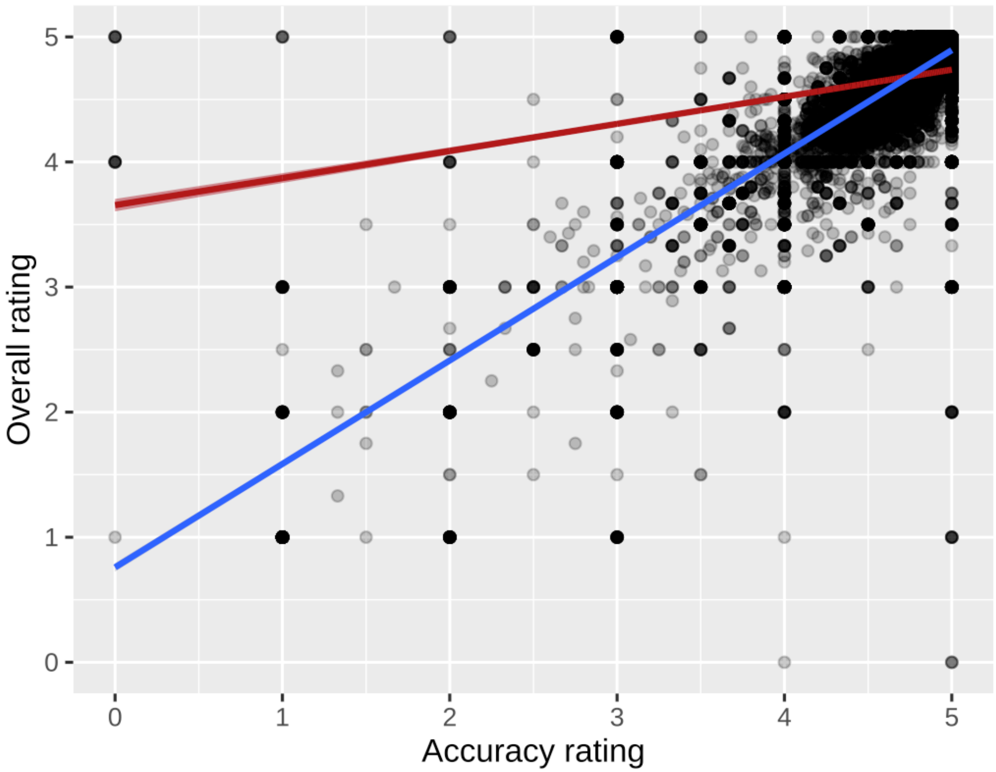




Not just OLS!

These plots are for an OLS model built with `lm()`





Any type of statistical model

The same techniques work for pretty much any model R can run

- OLS with high-dimensional fixed effects
- Logistic, probit, and multinomial regression (ordered and unordered)
- Multilevel (i.e., mixed and random effects) regression
- Bayesian models
- Machine learning models

If it has coefficients and/or makes predictions, you can (and should) visualize it!