# Bootstrapping

Jasmine Zhang

2023-11-16

## Generate a relevant sample

```
n_samp = 250
#constant variance
sim_df_const =
  tibble(
    x = rnorm(n_samp, 1, 1),
    error = rnorm(n_samp, 0, 1),
    y = 2 + 3 * x + error)

#non constant variance
sim_df_nonconst = sim_df_const |>
  mutate(
  error = error * .75 * x,
  y = 2 + 3 * x + error)
```

fit some linear models: lm assumes constant variance

```
sim_df_const |>
  lm(y~x, data = _) |>
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)     1.98    0.0981      20.2 3.65e- 54
## 2 x               3.04    0.0699      43.5 3.84e-118
```

```
sim_df_nonconst |>
  lm(y~x, data = _) |>
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)     1.93    0.105       18.5 1.88e- 48
## 2 x               3.11    0.0747      41.7 5.76e-114
```

solve the case when the variance is not constant

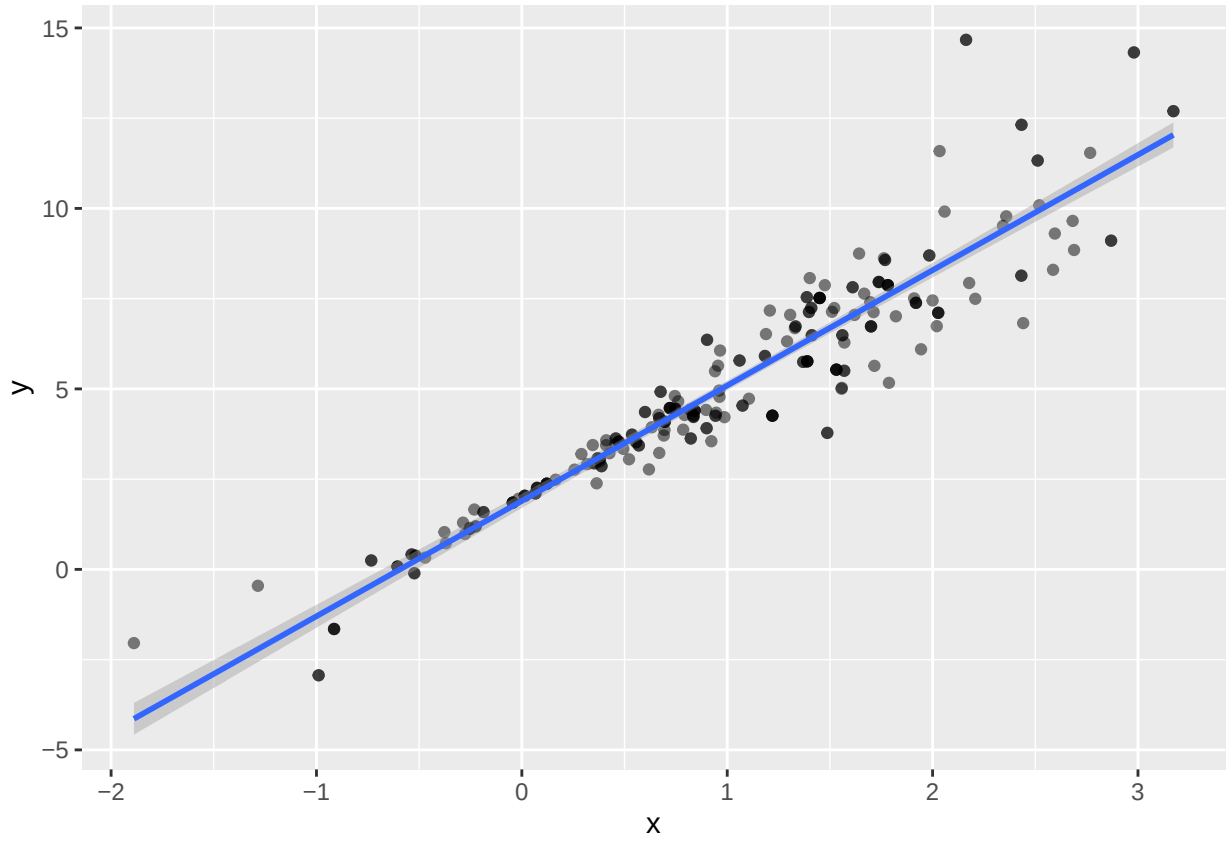
## Draw and analyze a bootstrap sample

```
boot_sample = function(df){
  sample_frac(df, replace = TRUE)
}
```

Lets see how it works

```
sim_df_nonconst |>
  boot_sample() |>
  ggplot(aes(x=x, y=y))+
  geom_point(alpha = 0.5)+
  stat_smooth(method = "lm") #change when bootstrap sample changes
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




## draw multiple bootstrap samples and analyze them

```
boot_straps = tibble(strap_number = 1:100) |>
  mutate(strap_sample = map(strap_number, \(i) boot_sample(sim_df_nonconst)))
boot_straps |>
```

```
  pull(strap_sample) |>
  nth(1) |>
  arrange(x)
```

```
## # A tibble: 250 x 3
##         x   error       y
##     <dbl>   <dbl>   <dbl>
##  1 -1.89   1.62   -2.04
##  2 -1.89   1.62   -2.04
##  3 -1.21  -0.781  -2.43
##  4 -1.21  -0.781  -2.43
##  5 -1.00   0.832  -0.169
##  6 -0.989 -1.97   -2.93
##  7 -0.914 -0.908  -1.65
##  8 -0.606 -0.106   0.0774
##  9 -0.536  0.0227  0.413
## 10 -0.524 -0.536  -0.106
## # i 240 more rows
```

Now do the `lm` fit

```
boot_results = boot_straps |>
  mutate(models = map(strap_sample, \(df) lm(y~x, data = df)),
         results = map(models, broom::tidy)) |>
  select(strap_number, results) |>
  unnest(results)
```

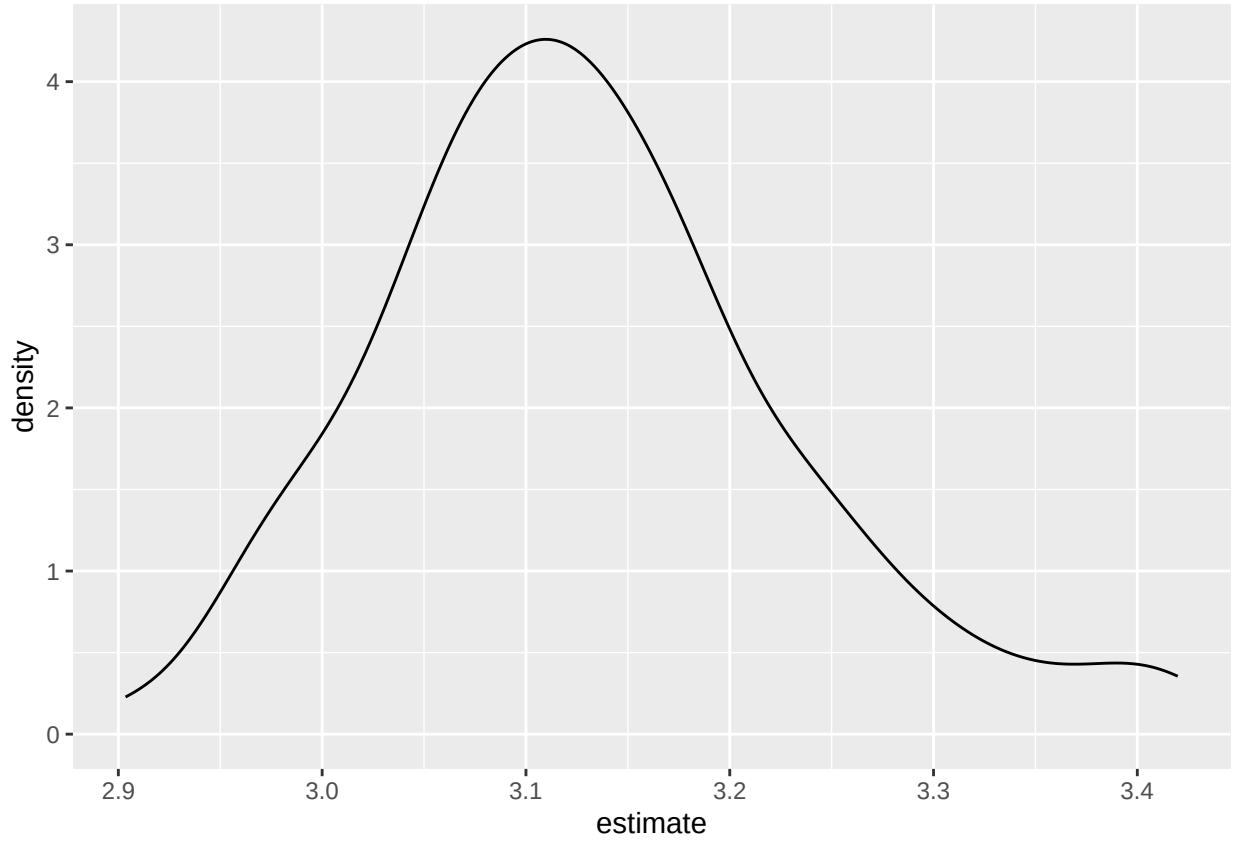
try to summarize these results: get a bootstrap SE

```
boot_results |>
  group_by(term) |>
  summarize(
    se = sd(estimate)
  )
```

```
## # A tibble: 2 x 2
##   term            se
##   <chr>        <dbl>
## 1 (Intercept) 0.0752
## 2 x           0.102
```

Look at the distribution

```
boot_results |>
  filter(term == "x") |>
  ggplot(aes(x = estimate)) +
  geom_density()
```

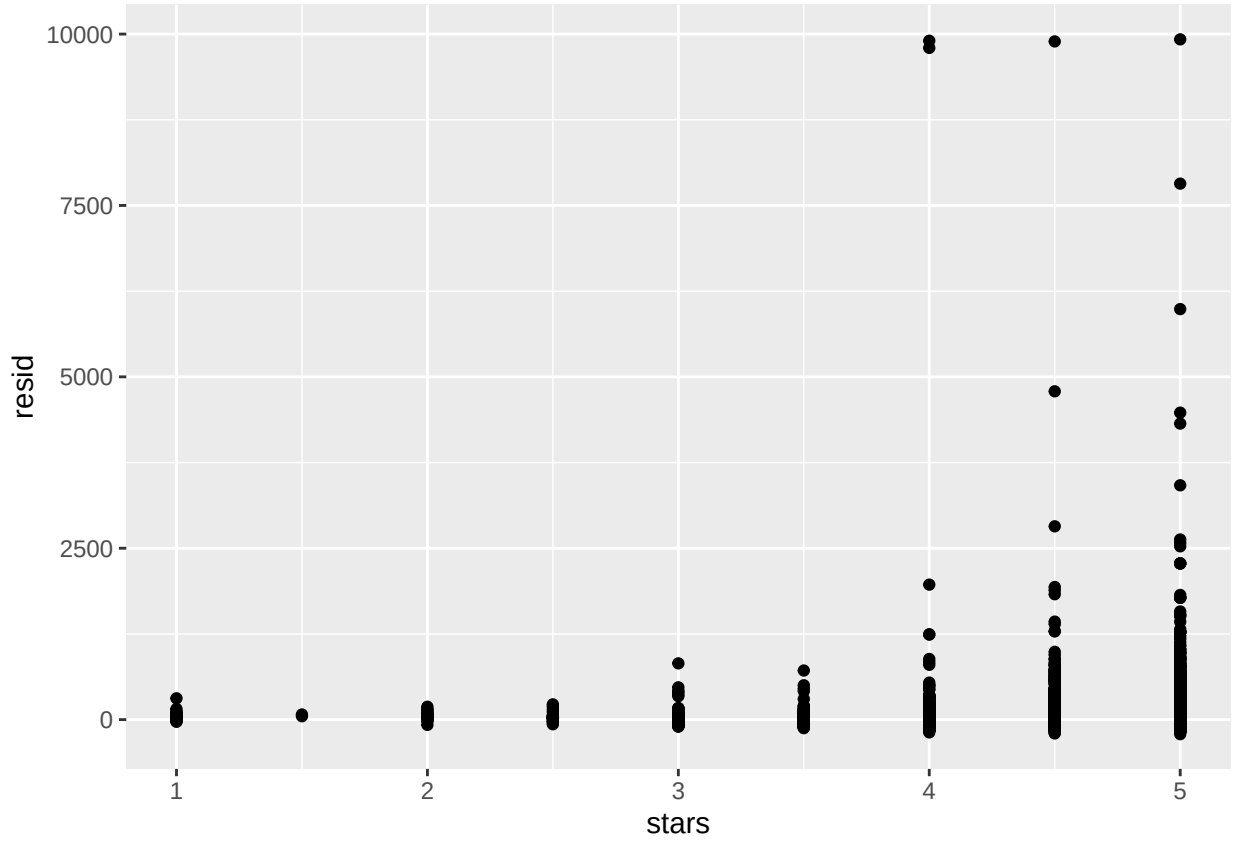



### Airbnb dataset

```
data("nyc_airbnb")

nyc_airbnb =
  nyc_airbnb |>
  mutate(stars = review_scores_location / 2) |>
  rename(
    borough = neighbourhood_group,
    neighborhood = neighbourhood) |>
  filter(borough != "Staten Island") |>
  drop_na(price, stars) |>
  select(price, stars, borough, neighborhood, room_type)
```

lets fit a regression of `price` on other variables and look at residuals

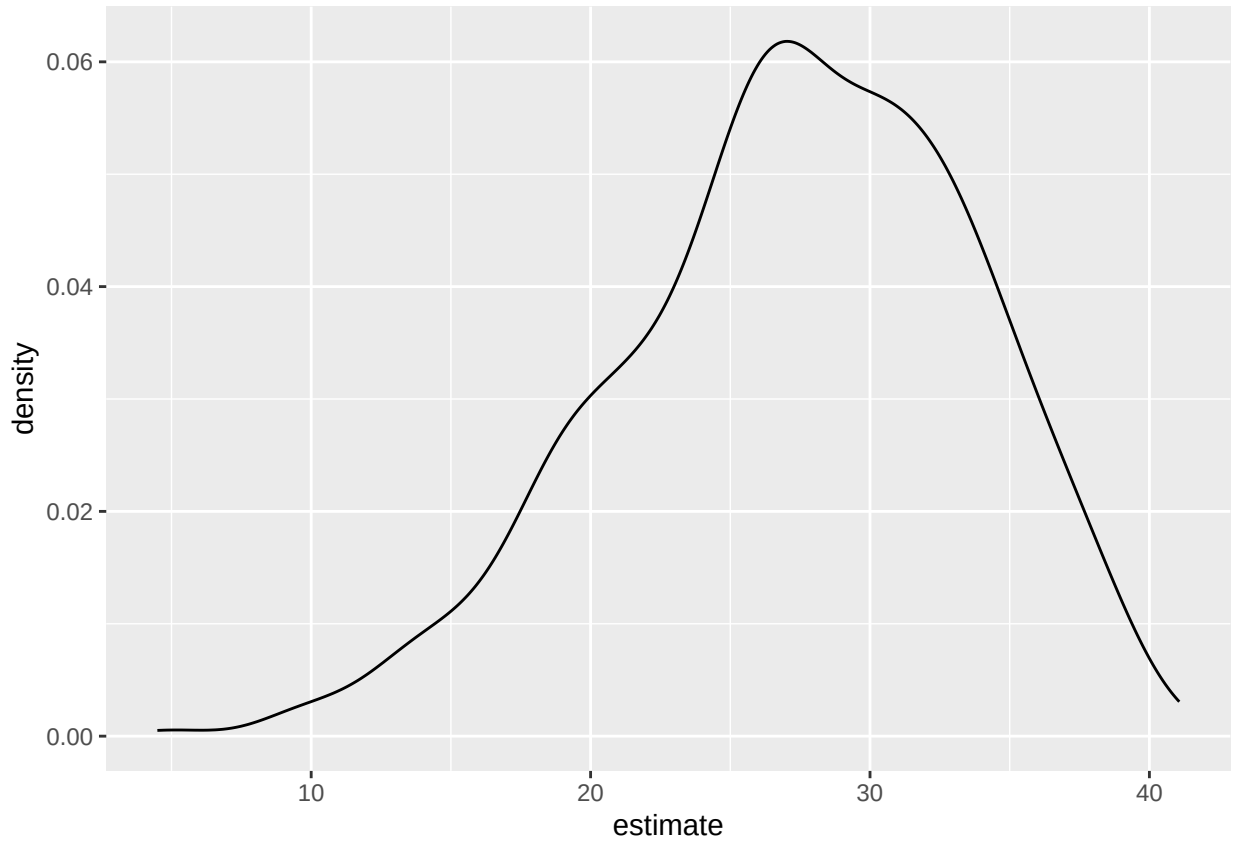
```
airbnb_fit = nyc_airbnb |>
  lm(price ~ stars+room_type+borough, data = _)

#residuals
nyc_airbnb |>
  modelr::add_residuals(airbnb_fit) |>
  ggplot(aes(x = stars, y = resid)) + geom_point() #residual very skewed
```




run a bootstrap on this whole thing to get estimates for the effect of `stars` on `price`

```
manhattan_df = nyc_airbnb |>
  filter(borough == "Manhattan")
boot_results = tibble(strap_number = 1:1000) |>
  mutate(strap_sample = map(strap_number, \(i) boot_sample(manhattan_df)),
         models = map(strap_sample, \(df) lm(price~ stars+room_type, data = df)),
         results = map(models, broom::tidy)) |>
  select(strap_number, results) |>
  unnest(results)
```

```
boot_results |>
  filter(term == "stars") |>
  ggplot(aes(x = estimate)) + geom_density()
```




```
#show dist of estimate of stars as a predictor of price
```