Bootstrapping

Adam J Sullivan, PhD

04/30/2018

Boostrapping

Goal of Statistics

- Make inferences about a population based on data.
- How??
  - Traditional statistical inference is based on the assumption of drawing repeated samples from a population
  - A statistic (e.g. the mean or a regression coefficient) is assumed to be fixed in the population.

What happens?

- If a researcher were to draw multiple samples, the statistic would be different each time.
- The term sampling distribution refers to the distribution that would result from taking multiple samples from a population and re-estimating a statistic on each new set of observations.
- Many times the sampling distribution is assumed to be normal, and a statistic is declared to be “significant” if the 95% confidence interval (the area under the curve excluding the smallest and largest 2.5% of values) does not include zero.

What if Assumptions are violated?

- Many times we do not have normally distributed variables.
- For example, consider a linear regression, we assume the residuals are normally distributed.
- If we do not have normally distributed residuals then our standard errors are off and therefore our individual significance tests for the estimates in our regression are off.

In Comes Bootstrapping!

- Bootstrapping is a method of resampling.
- Researchers use this to help us understand the variance of different estimates.

How does it work?

- We just discussed using a sample to make inferences about a population.
- In Bootstrapping, we will treat our sample data as a population.
- Then we will draw many samples from this new "population."

How does it work?

- For each sample, we caclulate our test statistic and save the information.
- If we do this say 500 times, we have 500 estimates of our test statistic.

Example

```
truth <- function(x){
  2.34 + 2.68*x - 3.42*x^2
}

noise <- function(x){
  rnorm(length(x), sd=0.1)
}
```

Example Data

```
set.seed(124)
data <- data_frame(
  x = runif(n=100, min=0, max = 1),
  y = truth(x) + noise(x)
)
```

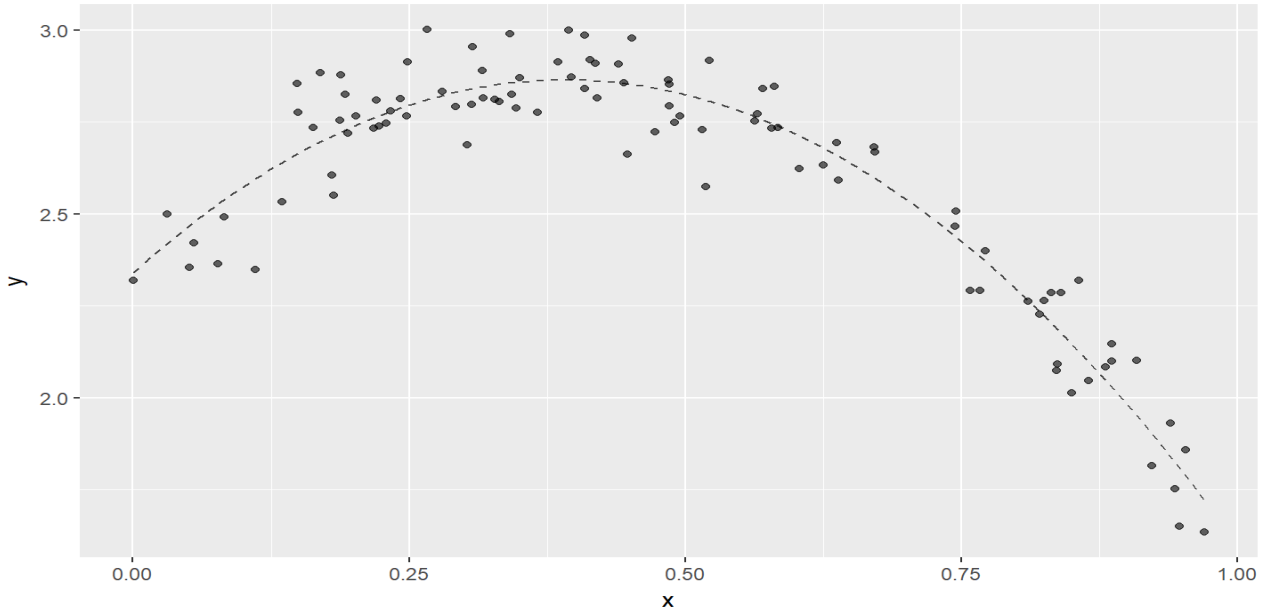
Example Data

```
data %>%
  DT::datatable()
```

Graphing the Data

```
ggplot(data, aes(x = x, y = y)) +
  stat_function(fun = truth, color = "black", alpha = 0.7, linetype = "dashed") +
  geom_point(alpha = 0.6)
```

Graphing the Data



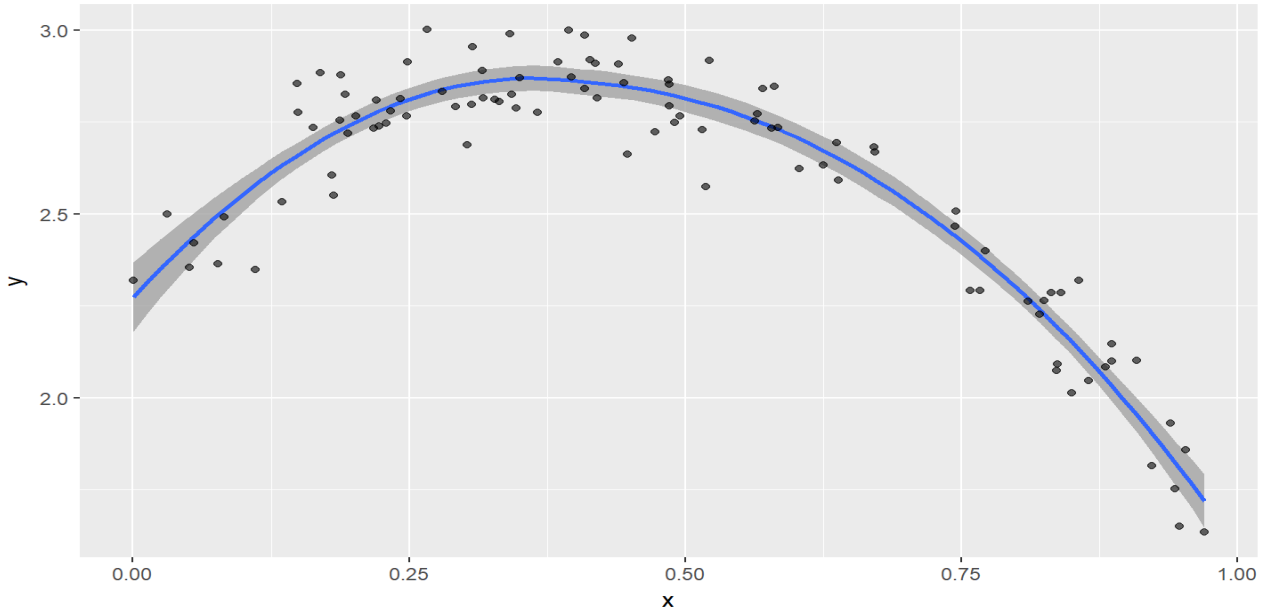
Our Linear Model

```
mod <- lm(y~poly(x,2,raw=TRUE), data)
tidy(mod, conf.int=T)[,-c(3:4)]
```

Our Linear Model

```
##                     term estimate  p.value conf.low conf.high
## 1             (Intercept)     2.32 3.04e-84     2.26      2.39
## 2 poly(x, 2, raw = TRUE)1     2.76 1.10e-32     2.45      3.06
## 3 poly(x, 2, raw = TRUE)2    -3.49 1.58e-42    -3.77     -3.20
```

Our Linear Model



Bootstrapping

- First we create 10,000 new datasets

```
data_bootstrap <-
  data %>%
  modelr::bootstrap(10000)
```

Bootstrapping

Running the `lm()` on our Bootstrap

- We first need to create a function in which we wish to run on our data.

```
fn_mod <- function(data){
  lm(y~poly(x, 2, raw=TRUE), data = data)
}
```

Running the `lm()` on our Bootstrap

- Then we run the `fn_mod()` over out bootstraps

```
data_bootstrap_model <-
  data_bootstrap %>%
  mutate(model=map(strap, fn_mod))
```

Running the `lm()` on our Bootstrap

```
data_bootstrap_model %>%
  head() %>%
  DT::datatable()
```

Getting the Parameters

- We use the `tidy()` function from the `broom` package.

```
data_bootstrap_param <-
  data_bootstrap_model %>%
  mutate(param = map(model, tidy)) %>%
  select(.id, param) %>%
  unnest()
```

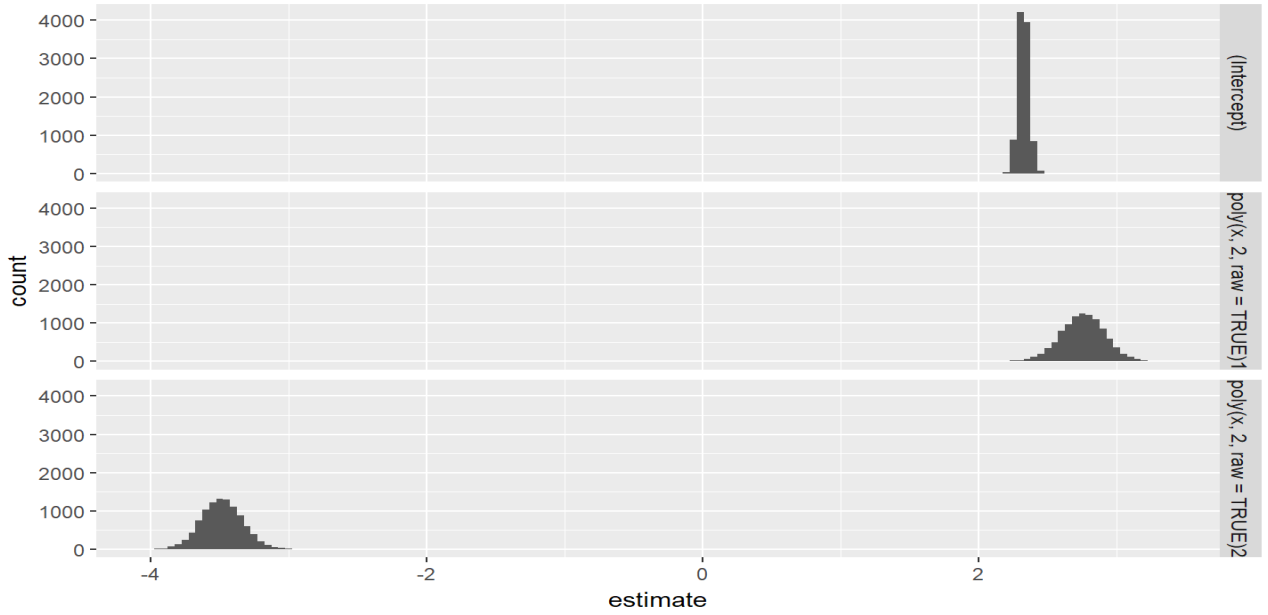
Getting the Parameters

```
data_bootstrap_param %>%
  head() %>%
  DT::datatable()
```

Distribution Plots

```
data_bootstrap_param %>%
  ggplot(aes(x = estimate)) +
  geom_histogram(binwidth = 0.05) +
  facet_grid(term ~ ., scales = "free_x")
```

Distribution Plots



Summarizing the Results

```
data_bootstrap_param_mean <-
  data_bootstrap_param %>%
  group_by(term) %>%
  summarise(estimate_mean = mean(estimate))
```

Summarizing the Results

```
data_bootstrap_param_mean %>%
  DT::datatable()
```

Interquartile Range of Estimates

```
data_bootstrap_param %>%
  group_by(term) %>%
  summarise(
    q_25 = quantile(estimate, 0.25),
    median = quantile(estimate, 0.50),
    q_75 = quantile(estimate, 0.75)
  )
```

Interquartile Range of Estimates

```
## # A tibble: 3 x 4
##   term                     q_25 median  q_75
##   <chr>                   <dbl>  <dbl> <dbl>
## 1 (Intercept)              2.30   2.32  2.35
## 2 poly(x, 2, raw = TRUE)1  2.65   2.76  2.86
## 3 poly(x, 2, raw = TRUE)2 -3.58  -3.48 -3.38
```

Confidence Intervals

```
data_bootstrap_param %>%
  group_by(term) %>%
  summarise(
    lower_95 = quantile(estimate, 0.025),
    upper_95 = quantile(estimate, 0.975)
  )
```

Confidence Intervals

```
## # A tibble: 3 x 3
##   term                    lower_95 upper_95
##   <chr>                      <dbl>    <dbl>
## 1 (Intercept)                 2.25     2.40
## 2 poly(x, 2, raw = TRUE)1     2.42     3.06
## 3 poly(x, 2, raw = TRUE)2    -3.77    -3.18
```

Add Predictions to Plot of Original Data

```
grid <-
  data %>%
  expand(x=seq_range(x,20))

boot_pred <-
  data_bootstrap_model %>%
  transmute(
    .id,
    data = map2(list(grid), model, add_predictions, var = "y")
  ) %>%
  unnest()
```

Add Predictions to Plot of Original Data

Add Predictions to Plot of Original Data

Plot Data and Model Estimates

```
ggplot(data = data, mapping = aes(x = x, y = y)) +
  geom_line(
    data = boot_pred %>% filter(as.numeric(.id) < 3000),
    aes(group = .id),
    color = "blue",
    alpha = 0.002
  ) +
  stat_function(fun = truth, color = "black", linetype = "dashed", size = 1) +
  geom_point(data = data, alpha = 0.5)
```

Plot Data and Model Estimates