Data-based Statistical Decision Model

Lecture 9 - Inference without assumptions using Boostrap

Sungkyu Jung

Review: Multiple regression model

In the multiple linear regression model, we assume that the response Y is a linear function of all the predictors, plus a constant, plus noise:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon.$$

We make no assumptions about the (marginal or joint) distributions of the X_i, but we assume that $E[\varepsilon|X] = 0$, $Var[\varepsilon|X] = \sigma^2$, and that ε is uncorrelated across measurements.

In a matrix form, the model is

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon.$$

- Fitted values: $\hat{\mathbf{Y}} = \hat{m}(\mathbf{X}) = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} = \mathbf{H}\mathbf{Y}$.
- Point predictions: To predict the conditional mean of Y at a point $\mathbf{X} = \mathbf{x}_*$,

$$\hat{m}(\mathbf{x}_*) = \mathbf{x}_*'\hat{\beta} = \mathbf{x}_*'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}.$$

- Confidence interval:

$$\hat{m}(\mathbf{x}_*) \pm t_{n-p,\alpha/2} s(\hat{m}(\mathbf{x}^*))$$

> - Valid only if when we “add” the Gaussian noise assumption $\varepsilon \sim N(0, \sigma^2\mathbf{I})$ independently of X.

The bootstrap

Bootstrap is used to estimate the standard error or the confidence interval

- without making the Gaussian noise assumption, or
- when the estimator is not a linear predictor (e.g. ordinary least squares, linear smoothers, etc)

Bootstrapping means

- positively: “get (oneself or something) into or out of a situation using existing resources,”
- negatively: “lift oneself by pulling his bootstrap.”

To better understanding what bootstrap aims to do, let us step back and understand the sampling distribution of a statistic (or an estimator).

Sampling distributions

We will take `nycflights13` data as an example. The dataset contains airline on-time data for all flights departing NYC in 2013. Among those, we take flights headed to San Francisco as our population, and assume that our observations consist of only 25 flights.

Below, we treat `SF` as the population, and `Sample25` as the sample (or the data at hand).

```
library(mdsr)
library(nycflights13)
SF <- flights %>%
  filter(dest == "SFO", !is.na(arr_delay))

set.seed(101)
Sample25 <- SF %>%
  sample_n(size = 25)
```

We are simply interested in estimating the *median arrival delay*, using the average of all available values of `arr_delay` (i.e. the values in the sample).

```
Sample25 %>% with(median(arr_delay))
```

```
## [1] -7
```

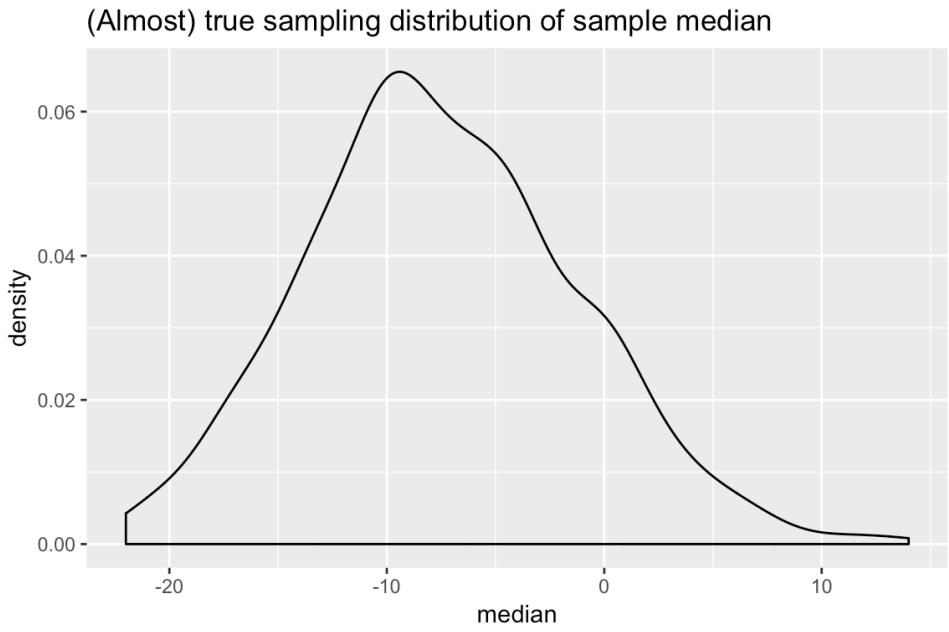
How reliable this statistic is? (Here, a *statistic* is a value computed from a sample.) To answer this, we can see how much the statistic vary from a sample to another. Since we have the population, we collect many of the sample statistics.

```
Trials <- list()
for(i in 1:500){
  Trials[[i]] <- SF %>%
    sample_n(size = 25) %>%
    summarize(median = median(arr_delay))
}
Trials_df <- bind_rows(Trials)
```

```
favstats( ~ median, data = Trials_df)
```

```
##  min  Q1 median Q3 max   mean       sd   n missing
##  -22 -12     -8 -4  14 -7.494 6.206691 500       0
```

```
Trials_df %>% ggplot(aes(x=median)) + geom_density() +
  labs(title = "(Almost) true sampling distribution of sample median")
```



To discuss reliability, it helps to have some standardized vocabulary.

- *Sample size* is the number of cases in the sample, usually denoted with n. In the above, the sample size is $n = 25$.
- *Sampling distribution* is the collection of the sample statistic from all potential samples of the same size from the population. Since there are zillons of potential samples, we approximated it using 500 trials here.
- *Shape of the sampling distribution* is worth noting. Here it is a little skewed to the right, but centered around -8, which is close to the truth.
- **Standard error** is the standard deviation of the sampling distribution. It describes the variability of the sampling distribution. For the trials calculating the sample mean in samples with $n = 25$, the standard error is 6.2066907 minutes.
- *95% confidence interval* is another way of summarizing the sampling distribution. From the smoothed histogram you can see it is about -20 to +10 minutes. If $n = 25$ were thought of as a larger number, then by the central limit theorem, the interval is calculated from the mean and standard error of the sampling distribution (of the median):

```
mean(Trials_df$median) + sd(Trials_df$median) * 1.96 * c(-1,1)
```

```
## [1] -19.659114   4.671114
```

The reliability of a sample statistic is typically measured by

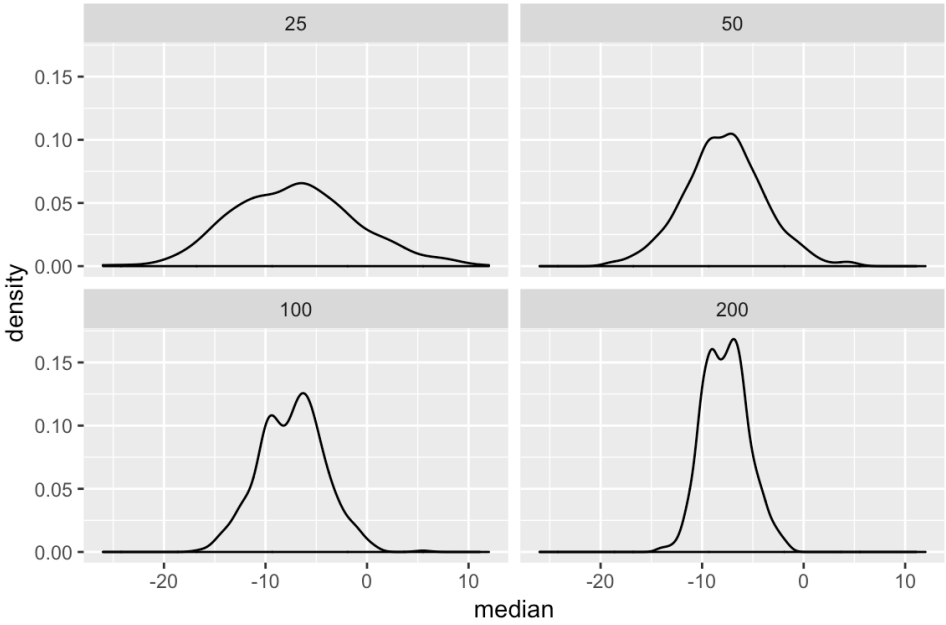
1. the mean of the statistic (mean of the sampling distribution). Better if it is closer to the truth.
2. the standard error of the statistic (standard of the sampling distribution). Better if it is small.

In above example, the standard error 6.2066907 seems to be too large to be acceptable. Equivalently, the confidence interval -19.6591139, 4.6711139 seems to be too wide. (The mean of the statistic is close to the truth.)

If we **increase the sample size n, the standard error will decrease**. To see this, repeat the experiment with $n = 25, 50, 100$ and 200, and see if the standard error decreases.

```
Trials.all <-list()
nvec = c(25,50,100,200)
for(j in seq_along(nvec)){
  Trials_n <- list()
  n <- nvec[j]
  for(i in 1:500){
    Trials_n[[i]] <- SF %>%
    sample_n(size = n) %>%
    summarize(median = median(arr_delay), n = n)
  }
  Trials.all[[j]] <- bind_rows(Trials_n)
}
Trials.all <- bind_rows(Trials.all)
```

```
Trials.all %>% ggplot(aes(x=median)) + geom_density() + facet_wrap(~n)
```



대수의법칙은 성립하나, 중심극한정리는 성립하지 않는다. 그 이유중 하나는 mean이 아닌 median으로 했기 때문.

```
Trials.all %>% group_by(n) %>% summarize(error = sd(median))
```

```
## # A tibble: 4 x 2
##       n error
##   <dbl> <dbl>
## 1    25  6.06
## 2    50  3.95
## 3   100  3.19
## 4   200  2.17
```

The figure above also demonstrates that Central Limit Theorem may fail for statistics whose form is not an average. In this case, the approximate confidence interval using the normal quantiles (or t-distribution quantiles) is less accurate, even when we know the population.

Bootstrap sampling distribution

In the previous examples, we had access to the population data and so we could find the sampling distribution by repeatedly sampling from the population. In practice, however, we have only one sample and not the entire population. The bootstrap is a statistical method that allows us to approximate the sampling distribution even

without access to the population.

The logical leap involved in the bootstrap is to **think of our sample itself as if it were the population**. Just as in the previous examples we drew many samples from the population, now we will draw many new samples from our original sample. This process is called *resampling*: drawing a new sample from an existing sample.

Bootstrap procedure

Basic setup: suppose that we have independent samples $z_1, \ldots, z_n \sim P_\theta$. The subscript θ emphasizes the fact that θ is some parameter of interest, defined at the population level. E.g., this could be the mean of the distribution, the variance, or something more complicated. Let $\hat{\theta} = \hat{\theta}(z_1, \ldots, z_n)$ be an estimate for θ that we compute from the samples $z_1, \ldots, z_n$.

Step 0: Treat $\hat{P}_\theta = \{z_1, \ldots, z_n\}$ as if it was the population

Step 1: Randomly draw n i.i.d. sample from $\hat{P}_\theta$;

$$z_i^* \sim \text{Unif}\{z_1, \ldots, z_n\}, \quad i.i.d., \ i = 1, \ldots, n.$$

n개를 뽑고, 복원추출을 한다.
모집단이 샘플인과정

(Draw n observations from $\{z_1, \ldots, z_n\}$ *with replacement*.)

Step 2: Compute $\hat{\theta}^* = \hat{\theta}(z_1^*, \ldots, z_n^*)$

Step 3: Repeat Steps 1 and 2, to get the *bootstrap sample* $\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$, for a large B.

Approximating the sampling distribution of $\hat{\theta}$

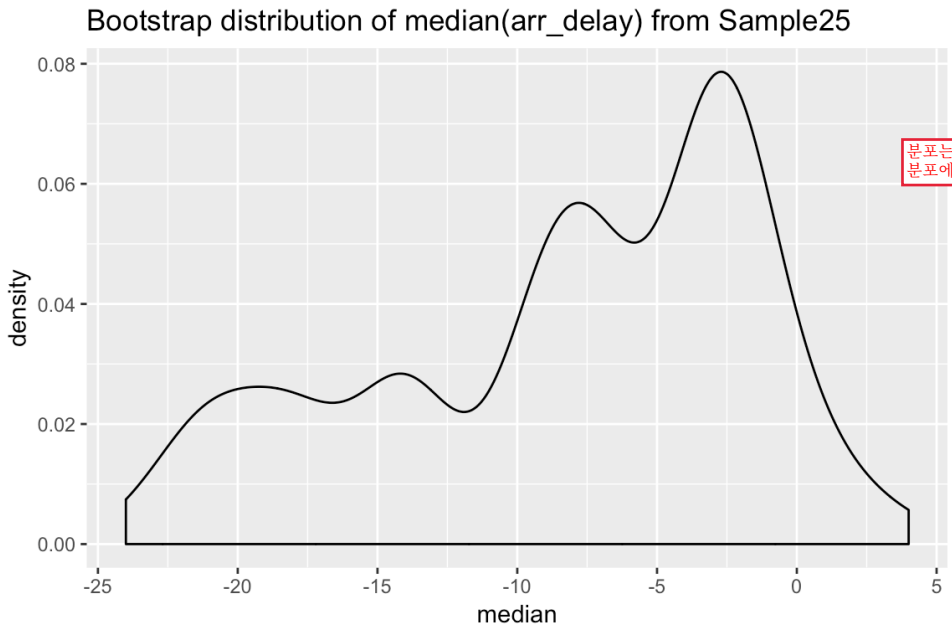
The empirical distribution of bootstrap sample is a crude estimate of the sampling distribution of $\hat{\theta}$ (in this example, $\hat{\theta}$ = median of the sample).

The boostrap procedure of creating $B = 500$ bootstrap statistics from `Sample25` in R is shown below.

```
n <- nrow(Sample25)
boot <- list()
for(i in 1:500){
    boot[[i]] <- Sample25 %>%
      sample_n(size = n, replace = TRUE) %>%
      summarize(median = median(arr_delay))
  }
boot_df <- bind_rows(boot)
```

sample25를 모집단인 것처럼 다시 부스트랩 샘플을 또 한다.

```
boot_df %>% ggplot(aes(median)) +
  geom_density() +
  labs(title = "Bootstrap distribution of median(arr_delay) from Sample25")
```



분포는 다르지만 range는 잘 나왔다.
분포에 대한 범위를 잘 보아야 하는 듯 하다.

The distribution of values in the bootstrap trials is called the bootstrap sampling distribution. It's not exactly the same as the sampling distribution, but for moderate to large sample sizes it has been proven to approximate those aspects of the sampling distribution that we care most about, such as the *standard error*.

Approximating the standard error

To approximate the variance of $\hat{\theta}$, compute the sample variance of bootstrap sample;

$$\hat{Var}^B(\hat{\theta}^*) = \frac{1}{B}\sum_{j=1}^{B}(\hat{\theta}_j^* - \bar{\hat{\theta}}^*)^2.$$

Then,

$$\text{Standard error}(\hat{\theta}) \approx \sqrt{\hat{Var}^B(\hat{\theta}^*)}$$

The bootstrap estimate of the standard error is the standard deviation of the bootstrap distribution:

```
sd(boot_df$mean)
```

```
## [1] NA
```

Bootstrap standard error estimates for large sample

The following code sample for population `SF` only once for each of the sample sizes 25–200, and use the sample (without reference to the population) to estimate the standard error.

```
Trials.bootstrap <-list()
nvec = c(25,50,100,200)
for(j in seq_along(nvec)){
  n <- nvec[j]
  sample_df <- SF %>% sample_n(size = n)
  Trials_n <- list()
  for(i in 1:500){
    Trials_n[[i]] <- sample_df %>%
      sample_n(size = n, replace = TRUE) %>%
      summarize(median = median(arr_delay), n = n)
  }
  Trials.bootstrap[[j]] <- bind_rows(Trials_n)
}
bind_rows(Trials.bootstrap) %>%
  group_by(n) %>%
  summarize(error = sd(median))
```

```
## # A tibble: 4 x 2
##       n error
##   <dbl> <dbl>
## 1    25  5.04
## 2    50  4.55
## 3   100  3.00
## 4   200  2.47
```

샘플사이즈가 커질 수록 error가 작아진다.

These standard error estimates, computed by bootstrapping the sample, are indeed quite close to the standard error, computed from the population.

In this example, the sample statistic is the mean. In practice, the statistics of interest are more complex, e.g. the coefficient estimate in linear regression model. The bootsrapping procedure can be applied to almost all situations, in order to quantify the uncertainty in a statistic.

Estimating bias

- $\text{Bias}(\hat{\theta}) = E(\hat{\theta}) - \theta$ (mean of estimator - population parameter)
- Treating $\{z_1, \ldots, z_n\}$ as if it is the population, above relation is translated to

$$\hat{\text{Bias}}(\hat{\theta}) = E(\hat{\theta}^*) - \hat{\theta}$$

$$E(\hat{\theta}) - \theta \approx E(\hat{\theta}^*) - \hat{\theta} \approx \frac{1}{B}\sum_{j=1}^{B}\hat{\theta}_j^* - \hat{\theta}$$

In our toy data example, there is no bias as $\theta = \bar{x}$ is known to be unbiased in estimation of $E(X)$. We can check the bootstrap estimate of the bias is close to zero.

```
mean(boot_df$median) - median(Sample25$arr_delay)
```

```
## [1] -1.276
```

Bootstrap confidence interval

Level $1-\alpha$ confidence interval for $\hat{\theta}$

1. Normal approximation confidence interval

By believing that $\hat{\theta}$ is approximately normal,

$$\theta \in (\hat{\theta} - \hat{\text{Bias}}(\hat{\theta})) \pm z_{1-\alpha/2} s^*(\hat{\theta}),$$

where $s^*(\hat{\theta}) = \sqrt{\hat{Var}^B(\hat{\theta}^*)}$ is the bootsrap approximation of standard error.

- If the normality is true, then the number of bootstrap sample can be of order 100.

---

2. Use percentiles of bootstrap sample

Directly using the bootstrap sample to approximate the sampling distribution, find the $\alpha/2$ and $1-\alpha/2$ quantiles of $\hat{\theta}^*_1, \ldots, \hat{\theta}^*_B$.

When $\hat{\theta}^*_j$ are ordered (from the smallest to the largest), set $(lower) = [(B+1)\alpha/2]$, $(upper) = [(B+1)(1-\alpha/2)]$. Then

$$\hat{\theta}^*_{(lower)} < \theta < \hat{\theta}^*_{(upper)}$$

E.g. If $B = 999$ bootstrap replicates were sampled, for $\alpha = 0.05$, $(lower) = 25$, $(upper) = 975$.

- To be accurate, the number of bootstrap sample should be of order 1000 or more.

---

3. Bias-corrected, accelerated percentile interval

Similar to the percentile interval above, but with a jackknife correction of bias:

$$\hat{\theta}^*_{(lower^*)} < \theta < \hat{\theta}^*_{(upper^*)}$$

- To be accurate, the number of bootstrap sample should be of order 1000 or more.

To use in R; Package `boot`

It is more convenient to use the `boot` function in the `boot` library. This function takes several arguments, the first three of which are required:

- data: A data vector, matrix, or data frame to which bootstrap resampling is to be applied. Each element of the data vector, or each row of the matrix or data frame, is treated as an observation.
- statistic: A function that returns the (possibly vector-valued) statistic to be bootstrapped. The first two arguments of this function must specify the original (sample) data set and an index vector, giving the indices of the observations included in the current bootstrap sample.
- R: the number of bootstrap replications.

To use boot, we therefore have to write a function that returns the statistic to be bootstrapped. For example, suppose now we want to use the median to estimate the *median arrival delay*.

```
boot.median <- function(data, indices){
  data <- data[indices,] # select obs. in bootstrap sample
  median(data$arr_delay) # return the median of bootstrap replicates
}
```

The `boot` function takes at least three arguments (data, statistics and R) and returns an object. In the printout below, “original” refers to the original $\hat{\theta}$, “bias” is the estimated bias, and “std. error” is the bootstrap standard error estimate.

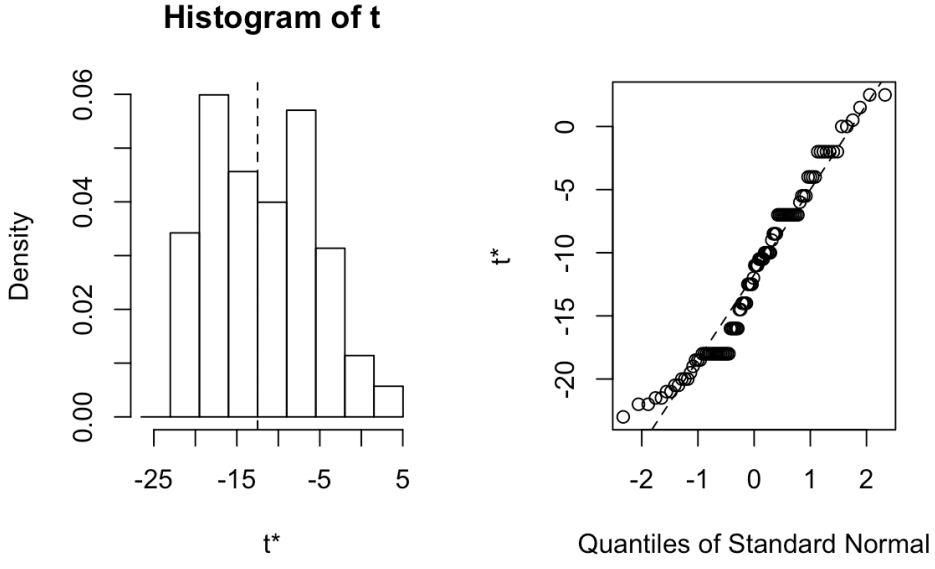
```
Sample50 <- SF %>% sample_n(size = 50)

library(boot)
boot.out <- boot(Sample50, boot.median, 100)
boot.out
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Sample50, statistic = boot.median, R = 100)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    -12.5    0.76    6.738432
```

`plot(boot.out)' provides two important graphics: the bootstrap distribution of $\hat{\theta}$ (as a histogram) and a QQ plot to see if the sampling distribution is indeed normal.

```
plot(boot.out)
```



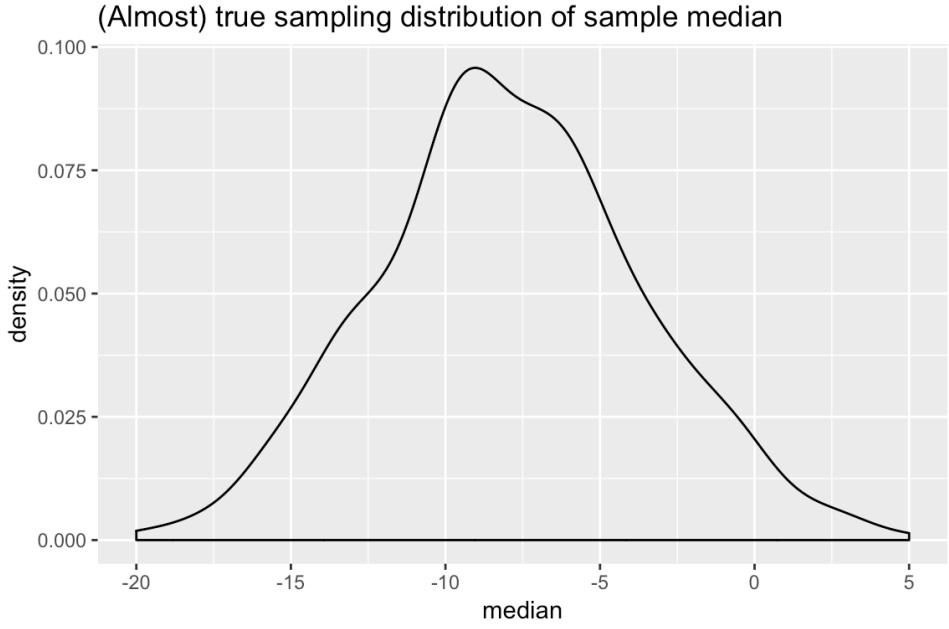
The `boot.ci` function calculates several kinds of bootstrap confidence intervals, including the bias-corrected normal-theory, percentile, and BCa intervals.

```
boot.ci(boot.out, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.out, type = c("norm", "perc", "bca"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   (-26.47,  -0.05 )   (-22.00,   1.93 )   (-22.52,   0.33 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
## Some BCa intervals may be unstable
```

Compare the intevals with true 95% coverage interval:

```
Trials_df %>% ggplot(aes(x=median)) + geom_density() +
  labs(title = "(Almost) true sampling distribution of sample median")
```




```
quantile(Trials_df$median, probs = c(0.025, 0.975))
```

```
##  2.5% 97.5%
## -16.0   0.5
```

Bootstrapping regressions

Assume that we want to fit a regression model with response variable y and predictors $X_1, \ldots, X_p$. We have a sample of n observations $\mathbf{z}_i = (y_i, x_{i1}, \ldots, x_{ip})$. We simply select B bootstrap samples of the $\mathbf{z}_i$, fitting the model and saving the coefficients from each bootstrap sample. We can then construct confidence intervals for the regression coefficients (or predictions) using the `boot` function.

For this, we will need a new function to compute and save the coefficients. Suppose that we regress `arr_delay` on `hour + dep_time` . Then the new statistic is a vector of $\hat{\beta}_1$ and $\hat{\beta}_2$.

```
boot.coef <- function(data, indices){
 data <- data[indices,] # select obs. in bootstrap sample
 mod <- lm(arr_delay ~ hour + dep_time, data = data)
 coefficients(mod)[-1] # return coefficient vector, except beta0
}
```

```
boot.out <- boot(Sample50, boot.coef, 500)
boot.out
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Sample50, statistic = boot.coef, R = 500)
##
##
## Bootstrap Statistics :
##        original       bias    std. error
## t1* -29.6395945  0.341181600  9.39072562
## t2*   0.3015349 -0.002301274  0.09139843
```

```
boot.ci(boot.out, index = 1, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 500 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.out, type = c("norm", "perc", "bca"),
##     index = 1)
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   (-48.39, -11.58 )   (-47.27,  -9.80 )   (-48.37, -10.73 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```
boot.ci(boot.out, index = 2, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 500 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.out, type = c("norm", "perc", "bca"),
##     index = 2)
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   ( 0.1247,  0.4830 )   ( 0.1157,  0.4762 )   ( 0.1226,  0.4826 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

If we wanted the confidence intervals for fitted values, then simply adjust the function for statistics.

```
boot.pred <- function(data, indices){
 data.boot <- data[indices,]
 mod <- lm(arr_delay ~ hour + dep_time, data = data.boot)
 predict(mod, data)
}
```

```
boot.out <- boot(Sample50, boot.pred, 500)
boot.out
```

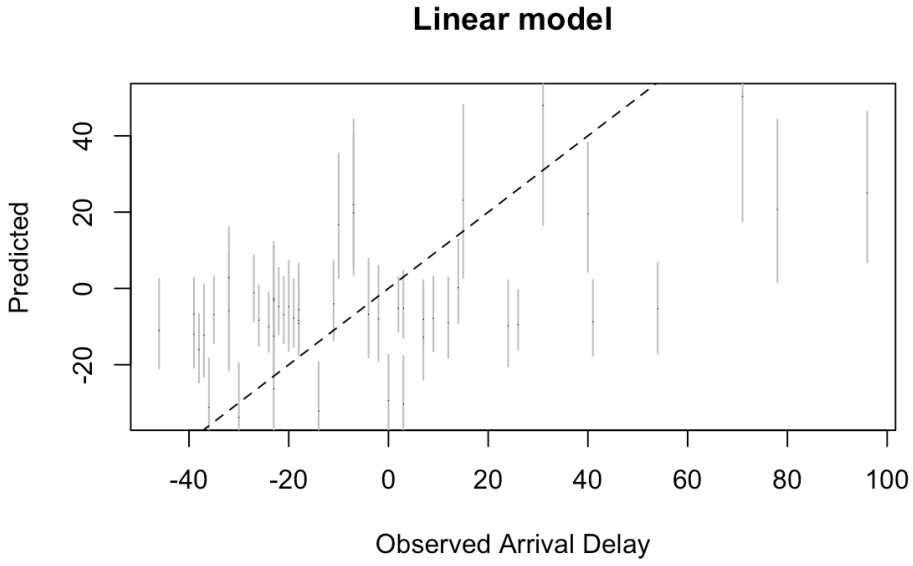
```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Sample50, statistic = boot.pred, R = 500)
##
##
## Bootstrap Statistics :
##         original       bias    std. error
## t1*   -7.8777002 -0.024642653    5.130714
## t2*   -8.0900567 -0.117584050    4.504431
## t3*  -32.1024877 -0.489798656    7.351776
## t4*  -29.3546743 -0.246099004    6.556681
## t5*   -4.7391736  0.116080018    6.387798
## t6*   50.3185274 -0.993076424   16.560152
## t7*   -7.7885218 -0.122601842    4.483853
## t8*   48.0294265 -0.762033505   15.556132
## t9*   -9.4533737 -0.577273247    4.072260
## t10*  -6.7395600 -0.622433371    4.287849
## t11*  19.5619719 -0.481261685    8.301536
## t12*  -2.6412500 -0.883583039    6.193485
## t13*  -3.1551413 -0.971506645    6.881612
## t14* -12.1883667  0.143565618    6.391456
## t15*  19.8295071 -0.775139252    8.877611
## t16*  -9.1518388 -0.582291038    4.088951
## t17*  16.7249801 -0.627002148    7.794588
## t18*  -8.7823048 -0.009589278    5.213203
## t19*  20.7001121 -1.079052403   10.462005
## t20*  -9.7760878  0.103423286    6.067875
## t21*  -5.3422434  0.126115601    6.429492
## t22*  -4.0469255  0.008085246    5.599155
## t23* -31.1978831 -0.504852030    7.183741
## t24*  -7.9796990 -0.891221980    6.182695
## t25* -30.2932785 -0.519905405    7.021947
## t26* -10.0564434 -0.567237664    4.044612
## t27*  -6.7735596 -0.911293147    6.334366
## t28* -15.9979622 -0.564841022    4.532481
## t29* -11.9888306 -0.728031501    5.097524
## t30*  -8.3024131 -0.210525448    4.003623
## t31* -33.8225184 -0.557651095    7.939670
## t32*  -6.8839172 -0.137655217    4.432394
## t33*  -5.1978861 -0.358662553    3.724394
## t34*  23.1123910 -1.119194735   11.126516
## t35*  -1.1887545 -0.521853032    4.250200
## t36* -12.4899015  0.148583410    6.436358
## t37*  -4.7731732 -0.172779758    4.374455
## t38*  21.9190719 -0.134585052    9.543216
## t39* -10.9950476 -0.841044065    5.882192
## t40* -12.7914364  0.153601201    6.482184
## t41*  25.0787777 -0.669541122    9.793991
## t42*  -6.8839172 -0.137655217    4.432394
## t43*  -5.6437782  0.131133393    6.452100
## t44*   2.7991978 -0.009364771    6.287059
## t45*  -5.2318857 -0.647522329    4.466240
## t46* -26.3605050  0.379401822    9.231952
## t47*  -8.9734820 -0.778209417    5.319352
```

```
## t48*   0.1745625 -0.062163836     5.569497
## t49*  -8.3704123 -0.788245000     5.380499
## t50*  -5.8221351  0.327051771     8.136529
```

```
ci_fit <- matrix(ncol = 2,nrow=50)
for(i in 1:50){
  ci_fit[i,] <- boot.ci(boot.out, index = i, type="bca")$bca[4:5]
}
```

```
mod <- lm(arr_delay ~ hour + dep_time, data = Sample50)
plot(Sample50$arr_delay, mod$fitted.values, type = "n", pch = 22,xlab = "Observed Arrival Delay",
    ylab = "Predicted", main = "Linear model")

segments(Sample50$arr_delay, ci_fit[,1],
         Sample50$arr_delay, ci_fit[,2], col = "grey")
abline(a = 0, b = 1, lty = "dashed")
points(Sample50$arr_delay, mod$fitted.values, pch = 16, cex = 0.1)
```



For smoothers and additive models, bootstrap confidence intervals for conditional means can be obtained just as the same.

```
library(mgcv)
Sample <- SF %>% sample_n(500)
boot.pred <- function(data, indices){
 data.boot <- data[indices,]
 mod <- gam(arr_delay ~ s(hour) + dep_time, data = data.boot)
 predict(mod, data)
}
```

```
system.time( boot.out <- boot(Sample, boot.pred, 500) ) # This takes some time!
```

```
##    user  system elapsed
##   5.508   0.158   5.705
```

```
ci_fit <- matrix(ncol = 2,nrow=500)
for(i in 1:500){
  ci_fit[i,] <- boot.ci(boot.out, index = i, type="bca")$bca[4:5]
}
```

```
plot(Sample$arr_delay, boot.out$t0, type = "n", pch = 22,xlab = "Observed Arrival Delay",
    ylab = "Predicted", main = "Linear model")

segments(Sample$arr_delay, ci_fit[,1],
         Sample$arr_delay, ci_fit[,2], col = "grey")
abline(a = 0, b = 1, lty = "dashed")
points(Sample50$arr_delay, mod$fitted.values, pch = 16, cex = 0.1)
```