06 Statistical Estimation

Confidence Intervals

- Suppose we observe data

$$X_1, X_2, X_3, \ldots, X_n$$

  and estimate the expected value using $\bar{X}$.
- There will be some estimation error between $\bar{X}$ and the estimation target $\mu = EX$.
- To provide a more complete description of the information in the data about μ, it is important to define a range of values around $\bar{X}$ that is likely to contain μ such that

$$\bar{X} \pm c$$

  for a constant c.
- This is called a confidence interval for μ.

Confidence Intervals

- The key concept behind a confidence interval is coverage.
- Suppose we devise some procedure for constructing a confidence interval leading to the interval $\bar{X} \pm c$.
- The coverage probability of this interval is

$$P(\bar{X} - c \leq \mu \leq \bar{X} + c).$$

- In words, this is the frequency over many replications that the interval contains the target value (population parameter) such as μ and σ.

Constructing a CI for μ

- Let us begin by standardizing the sample mean

$$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} = \sqrt{n}\frac{\bar{X} - \mu}{\sigma}$$

- This expected value of Z is zero and its variance is 1.
- If $\bar{X}$ is approximately normal, then Z is approximately normal.
- The central limit theorem (CLT) tells us that $\bar{X}$ will be approximately normal regardless of distribution of X if the sample size is not too small.

Constructing a CI for μ

- If

$$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$$

  is approximately normal, then

$$P\left(z_{\alpha/2} \leq \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \leq z_{1-\alpha/2}\right) = 1 - \alpha$$

- For example, if $\alpha = 0.05$, $z_{\alpha/2} = -1.96$ and $z_{1-\alpha/2} = 1.96$.
- Thus,

$$P\left(-1.96 \leq \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \leq 1.96\right) = 0.95$$

  and finally

$$P\left(\bar{X} - 1.96\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + 1.96\frac{\sigma}{\sqrt{n}}\right) = 0.95$$

Constructing a CI for μ

- Thus the rule
$$\bar{X} \pm 1.96 \frac{\sigma}{\sqrt{n}}$$
provides an approximate 95% confidence interval as long as the variance is known and the sample size is large enough for the central limit theorem to apply.
- If a coverage level other than 95% is desired, only the constant 1.96 need be changed. For example, to get 90% coverage $(\alpha = 0.1)$ use
$$\bar{X} \pm 1.64 \frac{\sigma}{\sqrt{n}}$$
since $z_{0.05} = -1.64$ and $z_{0.95} = 1.64$.
- The next simulation estimates the coverage probability of this interval for normal data.

```
set.seed(1111)

N <- c(10, 30, 50, 70, 90, 110, 130)
alpha <- c(0.01, 0.05, 0.1)
CP <- matrix(0, length(N), length(alpha))

for (i in 1:length(alpha)) {
    for (k in 1:length(N)) {
        n <- N[k]
        X <- matrix(rnorm(1e4*n), 1e4, n)
        M <- apply(X, 1, mean)
        z <- qnorm(1 - alpha[i]/2)
        C <- z/sqrt(n)
        ci <- (M-C < 0) & (M+C > 0)
        CP[k,i] <- mean(ci)
    }
}
colnames(CP) <- alpha
rownames(CP) <- N
CP
```

Different Population Distribution

- The results of coverage probability simulation indicate that the coverage is quite accurate regardless of the sample size and level.
- Suppose that the same program is run using the exponential distribution in place of the normal distribution, i.e.,

$$X_i \sim exp(1)$$

  Note that $EX = Var(X) = 1$.
- In the next simulation, the same confidence interval will be computed and investigate the coverage probability (the proportion that the intervals include $EX = 1$).

```
set.seed(1010)

N <- c(10, 30, 50, 70, 90, 110, 130)
alpha <- c(0.01, 0.05, 0.1)
CP <- matrix(0, length(N), length(alpha))

for (i in 1:length(alpha)) {
    for (k in 1:length(N)) {
        n <- N[k]
        X <- matrix(rexp(1e4*n), 1e4, n)
        M <- apply(X, 1, mean)
        z <- qnorm(1 - alpha[i]/2)
        C <- z/sqrt(n)
        ci <- (M-C < 1) & (M+C > 1)
        CP[k,i] <- mean(ci)
    }
}
colnames(CP) <- alpha
rownames(CP) <- N
CP
```

Incorrect Confidence Interval

- Suppose we use

$$\bar{X} \pm \frac{\text{range}(X_1, \ldots, X_n)}{2\sqrt{n}}$$

as a confidence interval for μ, where

$$\text{range}(X_1, \ldots, X_n) = \max(X_i) - \min(X_i)$$

- The following simulation estimates the coverage probability of this interval for standard exponential data.

```
set.seed(1234)
N <- c(10, 30, 50, 70, 90, 110, 130)
CP <- rep(0, length(N))

for (k in 1:length(N)) {
    n <- N[k]
    X <- matrix(rexp(n*1000), c(1000, n))
    M <- apply(X, 1, mean)
    MX <- apply(X, 1, max)
    MN <- apply(X, 1, min)
    C <- (MX - MN)/(2*sqrt(n))
    ci <- (M-C < 1) & (M+C > 1)
    CP[k] <- mean(ci)
}
CP
```

Incorrect Confidence Interval

- The results indicate that the coverage is around 0.78 for sample size 10, and increases to almost 0.98 for sample size 130.
- A confidence interval should have the same coverage probability regardless of the sample size, so this procedure is not working well.
- That is,
$$\bar{X} \pm \frac{\text{range}(X_1, \ldots, X_n)}{2\sqrt{n}}$$
is not a correct confidence interval for μ.

Unknown Population Variance

- The confidence interval

$$\bar{X} \pm z_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

  requires the population variance σ^2, which is usually unknown.
- A parameter such as σ^2 necessary for constructing the CI but not the value being estimated is called a nuisance parameter.
- A standard approach for handling nuisance parameters is to choose a reasonable estimate, then substitute the estimate in place of the population value, called the "plug-in" approach.

Unknown Population Variance

- For example, we could plug-in the estimate $\hat{\sigma}$ in place of σ.

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{2} (x_i - \bar{x})^2$$

- Plug-in CI's work well if the sample size is large.
- If the sample size is small or moderate, they tend to cover at less than the "nominal level." For example a CI with nominal 95% coverage may have 88% coverage when the sample size is small if the plug-in method is used.
- The next simulation calculates coverage probabilities for the the plug-in confidence interval.

```
set.seed(111)

N <- c(5, 10, 30, 50, 70, 90, 110, 130)
alpha <- c(0.01, 0.05, 0.1)
CP <- matrix(0, length(N), length(alpha))

for (i in 1:length(alpha)) {
    for (k in 1:length(N)) {
        n <- N[k]
        X <- matrix(rnorm(1e4*n), 1e4, n)
        M <- apply(X, 1, mean)
        SD <- apply(X, 1, sd)
        C <- qnorm(1 - alpha[i]/2)*SD/sqrt(n)
        ci <- (M-C < 0) & (M+C > 0)
        CP[k,i] <- mean(ci)
    }
}
colnames(CP) <- alpha
rownames(CP) <- N
CP
```

Unknown Population Variance

- Based on the above simulation, you will find that the coverage probabilities for sample size $n = 5$ and $n = 10$ are much lower than the nominal level. For larger sample sizes, the coverage is about right.
- What can be done to improve the performance for small sample sizes?
- In the special case where the data are approximately normal, it is possible to mathematically compensate for the effects of plugging in the estimated value $\hat{\sigma}$.

Confidence Interval with t-distribution

- For approximately normal data, it turns out that

$$\frac{\bar{X} - \mu}{\hat{\sigma}/\sqrt{n}} \sim t_{n-1}$$

  has a t-distribution with $n-1$ degrees of freedom.

- Thus,

$$P\left(\bar{X} - t^*_{\alpha/2,n-1}\frac{\hat{\sigma}}{\sqrt{n}} \leq \mu \leq \bar{X} + t^*_{1-\alpha/2,n-1}\frac{\hat{\sigma}}{\sqrt{n}}\right) = 1 - \alpha,$$

  where $t^*_{\alpha/2,n-1}$ means a critical value such as the $\alpha/2$ percentile of the t_{n-1} distribution.

- To get this critical value in `R`, use `qt(0.025, n-1)` if $\alpha = 0.05$.

```r
set.seed(111)

N <- c(5, 10, 30, 50, 70, 90, 110, 130)
alpha <- c(0.01, 0.05, 0.1)
CP <- matrix(0, length(N), length(alpha))

for (i in 1:length(alpha)) {
    for (k in 1:length(N)) {
        n <- N[k]
        X <- matrix(rnorm(1e4*n), 1e4, n)
        M <- apply(X, 1, mean)
        SD <- apply(X, 1, sd)
        C <- qt(1 - alpha[i]/2, n-1)*SD/sqrt(n)
        ci <- (M-C < 0) & (M+C > 0)
        CP[k,i] <- mean(ci)
    }
}
colnames(CP) <- alpha
rownames(CP) <- N
CP
```

Quantiles of t-distribution

- As the degrees of freedom gets larger, the $(1-\alpha)^{\text{th}}$ quantile of the t-distribution gets closer to the $(1-\alpha)^{\text{th}}$ quantile of the standard normal.
- That is to say, t-distribution is equivalent to the standard normal distribution as the sample size n goes infinity.
- In the next simulation, the quantile values of t-distribution are computed for each sample size which ranges from 2 to 10^8.
- Theoretical quantile of the standard normal distribution is then compared with that of t-distribution.

```
N <- c(2, 5, 10, 15, 20, 30, 50, 1e2, 1e3, 1e4,
      1e5, 1e6, 1e7, 1e8)
alpha <- c(0.01, 0.05, 0.1)
out <- matrix(0, length(N)+1, length(alpha))

for (k in 1:length(alpha)) {
    for (i in 1:length(N)) {
        out[i, k] <- qt(1-alpha[k]/2, N[i]-1)
    }
out[length(N)+1, k] <- qnorm(1-alpha[k]/2)
}

rownames(out) <- c(N, "qnorm")
colnames(out) <- alpha
out
```

Confidence Interval Widths

- The CI $\bar{X} \pm c$ has width $2c$.
- The width of a confidence interval is related to its coverage probability
  - Wider confidence intervals have higher coverage probabilities.
  - Narrower confidence intervals have lower coverage probabilities.
- In case that two different confidence intervals are compared with each other, the narrower confidence interval is always preferred as long as both confidence intervals have correct coverage probabilities.

Exercise

- Suppose that the random variable $x_i \sim B(n,p)$, and $x_i \in \{0,1\}$. If the sample proportion is defined as

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

The 95% confidence interval for the success probability p is defined as

$$\left[\max\left(0,\ \hat{p} - 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right), \min\left(1,\ \hat{p} + 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right)\right]$$

Compute the coverage probability when $n = 10, 20, 50$ and 100.

Bootstrap Confidence Interval

- In statistics, bootstrap is one of random resampling techniques with replacement.
- The idea behind the bootstrap is to use the data we have to produce artificial data sets that are similar to what we would get under replication.
- If we want a CI for $\mu = EX$, then for the k-th artificial data set $X^{(k)}$ we can compute the sample mean $\bar{X}^{(k)}$.
- The artificial data sets must be constructed so that the variation in $\bar{X}^{(k)}$ approximates the sampling distribution of $\bar{X}$ under actual replication.
- Once we have achieved this, we can use the sample $\alpha/2$ and $(1 - \alpha/2)$ quantiles of the $\bar{X}^{(k)}$ as the lower and upper bounds of a confidence interval, respectively.

Bootstrap Data

- For the standard bootstrap, the data values are sampled with replacement from the actual data (this is called “resampling”).
- In the bootstrap, the artificial data sets are the same size as the actual data set. In any given data set, some of the actual data values may be repeated, and others may be missing
- For example, suppose that we have the actual data $1, 2, 3, \ldots, 10$. Then, 100 bootstrap data sets can be obtained in the following way.

```
set.seed(12345)
x <- seq(10)
boot <- NULL
for (j in 1:100)
boot <- rbind(boot, sample(x, length(x), replace=TRUE))
```

Bootstrap Confidence Interval

- The following codes generate 10,000 bootstrap data sets using `ceiling()` and `runif()` functions.

  ```
  nrep <- 1e4
  p <- length(x)
  boot <- matrix(0, nrep, p)
  for (i in 1:nrep) {
      boot[i, ] <- x[ceiling(p*runif(p))]
  }
  ```

- The following code gives you a 95% confidence interval from the bootstrap data sets

  ```
  M <- apply(boot, 1, mean)
  hist(M, nclass=50, col="cyan")

  M <- sort(M)
  c(M[250], M[9750])
  abline(v=c(M[250], M[9750]), lty=2, lwd=2, col="red")
  ```

```
set.seed(1010)

N <- c(10, 20, 50, 100, 200)
nrep <- 1000; nboot <- 1000
CP <- NULL
for (j in 1:length(N)) {
    nc <- 0
    n <- N[j]
    for (k in 1:nrep) {
        X <- rnorm(n)
        B <- X[ceiling(n*runif(n*nboot))]
        B <- array(B, c(nboot, n))
        M <- apply(B, 1, mean)
        M <- sort(M)
        C <- c(M[25], M[975])
        if ((C[1] < 0) & (C[2] > 0)) nc <- nc+1
    }
CP[j] <- nc/nrep
}
CP
```

Bootstrap Confidence Interval

- Although the bootstrap is easy to apply, it does not necessarily have good coverage properties.
- A major advantage of the bootstrap is that it can be applied to any estimation problem, not just estimation of the expected value.
- $(1-\alpha)\times 100\%$ bootstrap confidence interval for a population parameter θ holds

  $$P(\hat{\theta}_{[\alpha/2]} < \theta < \hat{\theta}_{[1-\alpha/2]}) = 1-\alpha$$

  if n is large enough.
- The following simulation computes the coverage probability of bootstrap confidence intervals for the population standard deviation based on the sample standard deviation.

```
set.seed(123)

N <- c(10, 20, 50, 100, 200)
nrep <- 1000; nboot <- 1000
CP <- NULL
for (j in 1:length(N)) {
    nc <- 0
    n <- N[j]
    for (k in 1:nrep) {
        X <- rnorm(n)
        B <- X[ceiling(n*runif(n*nboot))]
        B <- array(B, c(nboot, n))
        M <- apply(B, 1, sd)
        M <- sort(M)
        if ((M[25] < 1) & (M[975] > 1)) nc <- nc+1
    }
CP[j] <- nc/nrep
}
CP
```

The Parametric Bootstrap

- The parametric bootstrap is a variant of the "non-parametric" bootstrap presented above.
- The parametric bootstrap is used if the distributional family of the data is considered known (e.g. normal, exponential).
- Like the non-parametric bootstrap, the parametric bootstrap is most useful when a statistic other than the expected value is of interest. In this case, no simple formula such as σ^2/n exists for producing a confidence interval.
- The parametric bootstrap requires some distribution assumption of data. If this assumption is true, parametric bootstrap is better than the non-parametric bootstrap. However, if the distribution assumption is not satisfied, the non-parametric bootstrap is often better than the parametric bootstrap.

Parametric Bootstrap Steps

- There are three main steps to the parametric bootstrap:
  1. Use the observed data to estimate the parameters of the parametric distribution. For example, if the data are thought to be normally distributed, then the mean (μ) and variance (σ^2) must be estimated. e.g., $\bar{X}$ and $\hat{\sigma}^2$.
  2. A large number of artificial data are drawn from the estimated parametric distribution.

$$x_1^*, x_2^*, \ldots, x_n^* \sim N(\bar{X}, \hat{\sigma}^2)$$

  3. Calculate your statistic of interest on each artificial data set. Then sort the values and use the appropriate quantiles to define your CI.
- The following simulation computes the coverage probability of bootstrap confidence intervals for the median based on the sample median.

```
set.seed(54321)
N <- c(5, 10, 20, 50, 100, 200)
nrep <- 1000; nboot <- 1000
CP <- NULL
for (j in 1:length(N)) {
    n <- N[j]
    nc <- 0
    for (k in 1:nrep)  {
        X <- rnorm(n)
        xbar <- mean(X)
        sighat <- sd(X)
        B <- rnorm(nboot*n, mean=xbar, sd=sighat)
        B <- array(B, c(nboot, n))
        M <- apply(B, 1, median)
        M <- sort(M)
        if ((M[25] < 0) & (M[975] > 0)) nc <- nc+1
    }
CP[j] <- nc/nrep
}
CP
```

Point Estimation

- In statistical inference problem, we have a sample

$$X_1, X_2, X_3, \ldots, X_n$$

  generated from a distribution with a parameter θ.
  - $\theta = \mu$ in a normal distribution $N(\mu, \sigma)$.
  - $\theta = \sigma$ in a normal distribution $N(\mu, \sigma)$.
  - $\theta = p$ in a binomial distribution $B(n, p)$.
- The objective of point estimation is to use $X_1, \ldots, X_n$ to find an appropriate estimate for the true and unknown value of θ.
- A point estimator can be obtained by selecting a suitable statistic and computing its value from the given sample data $X_1, \ldots, X_n$

Bias

- The bias is the expected value of the estimator minus the population parameter. That is,

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

- Bias measures whether over many replications, the estimator yields results that are correct on average.
  - Positive bias means that the estimator is too large on average compared to the true value.
  - Negative bias means that the estimator is too small on average compared to the true value.
- If $E(\hat{\theta}) = \theta$, $\hat{\theta}$ is called an unbiased estimator.
  - A sample mean $\bar{X}$ is an unbiased estimator for μ.
  - A sample standard deviation $\hat{\sigma}$ is an unbiased estimator for σ.

Other Estimators of Population Mean

- Given data $X_1, \ldots, X_n$, the sample mean $\bar{X}$ is the most common estimator of the population mean μ.
- However, there are alternative estimators of μ that may work better in certain situations.
- For example, if the population distribution is known to be symmetric, the sample median can be used to estimate μ (which is also the population median in that case).
  - Sample mean $\bar{X}$ vs. sample median $\tilde{X}$.
  - $E(\bar{X}) = \mu$ and $E(\tilde{X}) = \mu$ if the distribution is symmetric.
  - What if the population distribution is not symmetric?
- Next we compares the expected values and variances of the sample mean and sample median for standard normal data, and for t-distributed data with various degrees of freedom.

```
set.seed(101010)

nrep <- 1e4; n <- 10
Z <- matrix(rnorm(nrep*n), nrep, n)
Z_mean <- apply(Z, 1, mean)
Z_med <- apply(Z, 1, median)
E <- c(mean(Z_mean), mean(Z_med))
V <- c(var(Z_mean), var(Z_med))
df <- c(2, 4, 6, 8, 10)
for (i in 1:length(df)) {
    T <- rt(nrep*n, df[i])
    T <- matrix(T, nrep, n)
    T_mean <- apply(T, 1, mean)
    T_med <- apply(T, 1, median)
    E <- rbind(E, c(mean(T_mean), mean(T_med)))
    V <- rbind(V, c(var(T_mean), var(T_med)))
}
colnames(E) <- colnames(V) <- c("mean", "median")
rownames(E) <- rownames(V) <- c("normal", df)
```

Example

- Suppose
$$\hat{\theta} = \hat{\theta}(X_1, \ldots, X_n)$$
is an estimator of a population parameter θ.
- The mean squared error is
$$\text{MSE} = E(\hat{\theta} - \theta)^2.$$
- The MSE is a direct measure of the amount of error in the estimator.
- The square in the MSE prevents positive and negative errors from canceling each other out. However, it changes the scale. To get an error estimate on the same scale as the data, use the root mean squared error (RMSE):
$$\text{RMSE} = \sqrt{E(\hat{\theta} - \theta)^2}$$

Bias, Variance and MSE

- Suppose $\hat{\theta}$ is an estimator for θ
  - The bias is $E(\hat{\theta}) - \theta$.
  - The variance is $\mathrm{Var}(\hat{\theta}) = E\left(\hat{\theta} - E(\hat{\theta})\right)^2$.
- It is a fact that

$$\begin{aligned}\text{MSE} &= E\left(\hat{\theta} - \theta\right)^2 \\ &= E\left(\hat{\theta} - E(\hat{\theta}) + E(\hat{\theta}) - \theta\right)^2 \\ &= E\left(\hat{\theta} - E(\hat{\theta})\right)^2 + \left(E(\hat{\theta}) - \theta\right)^2 \\ &= \text{Variance} + \text{Bias}^2\end{aligned}$$

- That is,

$$\text{MSE}(\hat{\theta}) = \text{Bias}(\hat{\theta})^2 + \mathrm{Var}(\hat{\theta})$$

```
set.seed(101010)

nrep <- 1e4; n <- 10
Z <- matrix(rnorm(nrep*n), nrep, n)
Z_mean <- apply(Z, 1, mean)
Z_med <- apply(Z, 1, median)
MSE <- c(mean(Z_mean)^2+var(Z_mean), mean(Z_med)^2+var(Z_med))
df <- c(2, 4, 6, 8, 10)
for (i in 1:length(df)) {
    T <- rt(nrep*n, df[i])
    T <- matrix(T, nrep, n)
    T_mean <- apply(T, 1, mean)
    T_med <- apply(T, 1, median)
    MSE <- rbind(MSE, c(mean(T_mean)^2+var(T_mean),
                 mean(T_med)^2+var(T_med)))
}
colnames(MSE) <- c("mean", "median")
rownames(MSE) <- c("normal", df)
MSE
```

Exercise

- For estimation of population variance σ^2, compare two estimators in terms of bias, variance and MSE.

$$\hat{\theta}_1 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 , \quad \text{and} \quad \hat{\theta}_1 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Fix the sample size as $n = 50$.

1. Assume that $x_i \sim N(0, 1)$, so $\sigma^2 = 1$.
2. Assume that $x_i \sim U(0, 1)$, so $\sigma^2 = 1/12$.
3. Assume that $x_i \sim \exp(1)$, so $\sigma^2 = 1$.

```r
set.seed(1234)
n <- 50
out1 <- out2 <- matrix(0, 3, 3)
colnames(out1) <- colnames(out2) <- c("norm", "unif", "exp")
rownames(out1) <- rownames(out2) <- c("bias", "var", "MSE")
for (j in 1:3) {
    sig <- 1
    if (j==1) x <- matrix(rnorm(n*1e4), 1e4, n)
    else if (j==2) {
        x <- matrix(runif(n*1e4), 1e4, n)
        sig <- 1/12
    }
    else x <- matrix(rexp(n*1e4), 1e4, n)
    sd1 <- apply(x, 1, var)
    sd2 <- apply(x, 1, function(t) var(t)*(n-1)/n)
    out1[,j] <- c((mean(sd1)-sig), var(sd1), mean((sd1-sig)^2))
    out2[,j] <- c((mean(sd2)-sig), var(sd2), mean((sd2-sig)^2))
}
```

Contaminated Normal Distribution

- Real data often are not as clean as simulated normal data.
- The contaminated normal distribution is a model for data in which some extreme outliers are present.
- To simulated a contaminated normal draw, we select one of two normal populations at random, then generate a value from the selected population.
  - First, generate a *Bernoulli* trial B with probability α of observing $B = 1$, and probability $1 - \alpha$ of observing $B = 0$.
  - Next, if $B = 1$ generate $Z \sim N(0, \tau^2)$. If $B = 0$ generate $Z \sim N(0, \sigma^2)$.
  - Typically, $\sigma^2 < \tau^2$, and the value of $\alpha = 0.01$ or 0.05.
  - The Z values drawn when $B = 1$ are the outliers. But we don't observe B, only Z.

```
set.seed(1010)
alpha <- 0.05       ## Prob. of the population 1
tau2 <- 100         ## Variance of the population 1
sig2 <- 1           ## Variance of the population 2

## A 1000 x 50 array of iid true/false values in which
## each entry has probability alpha of being true.
A <- matrix(runif(1000*50) < alpha, 1000, 50)

## The population 1 draws.
B <- matrix(rnorm(50*1000, sd=sqrt(tau2)), 1000, 50)

## The population 2 draws.
C <- matrix(rnorm(50*1000, sd=sqrt(sig2)), 1000, 50)

## These are the contaminated normal draws.
X <- A*B + (1-A)*C
```

```
set.seed(1234)
alpha <- 0.05   ## Proportion of outliers.
tau2 <- 100     ## Variance of the outlier component.
sig2 <- 1       ## Variance of the non-outlier component.
nrep <- 1e3     ## Number of replications.
n <- 50         ## Sample size
mu <- 0         ## Population mean (parameter)

A <- matrix(runif(n*nrep) < alpha, nrep, n)
B <- matrix(rnorm(n*nrep, mu, sqrt(tau2)), nrep, n)
C <- matrix(rnorm(n*nrep, mu, sqrt(sig2)), nrep, n)
X <- A*B + (1-A)*C
X_mean <- apply(X, 1, mean)
X_med <- apply(X, 1, median)
c(mean((X_mean-mu)^2), mean((X_med-mu)^2))
```

```
set.seed(12345)
RE1 <- RE2 <- matrix(0, 100, 3)

for (i in 1:100) {
    A <- matrix(runif(n*nrep) < alpha, nrep, n)
    B <- matrix(rnorm(n*nrep, mu, sqrt(tau2)), nrep, n)
    C <- matrix(rnorm(n*nrep, mu, sqrt(sig2)), nrep, n)
    X <- A*B + (1-A)*C
    X1 <- apply(X, 1, mean)
    X2 <- apply(X, 1, median)
    RE1[i, ] <- c((mean(X1)-mu)^2, var(X1), mean((X1-mu)^2))
    RE2[i, ] <- c((mean(X2)-mu)^2, var(X2), mean((X2-mu)^2))
}
apply(RE1, 2, mean)
apply(RE2, 2, mean)
```

```
col <- c("orange", "green", "purple")
par(mfrow=c(1,2))
boxplot(RE1, boxwex=0.5, col=col, names=c("Bias2", "Var",
        "MSE"), main="Mean", ylab="Error")
boxplot(RE2, boxwex=0.5, col=col, names=c("Bias2", "Var",
        "MSE"), main="Median", ylab="Error")

## The same range of y-axis
boxplot(RE1, boxwex=0.5, col=col, names=c("Bias2", "Var",
        "MSE"), main="Mean", ylab="Error",
        ylim=c(0, 0.15))
boxplot(RE2, boxwex=0.5, col=col, names=c("Bias2", "Var",
        "MSE"), main="Median", ylab="Error",
        ylim=c(0, 0.15))
```

Bias and Variance Trade-off

- Suppose we collect n independent measurements using two different populations.
- The "good" population has a mean μ_1, which is the value we aim to estimate. Let $\bar{X}_G$ be the average of these data.
- The "bad" population has a mean μ_2. Let $\bar{X}_B$ be the average of these data.
- Assume that two populations have the same variance.
- Rather than throwing out the "bad" data, suppose we consider estimates of the form

$$\hat{\theta} = p\bar{X}_G + (1-p)\bar{X}_B \quad \text{for} \quad 0 \le p \le 1,$$

  trying to salvage some information from the bad data.
- We want to find the best value of p that minimizes MSE.

```
## The number of replications.
nrep <- 1e4

## The sample size from each population.
n <- 10

## The mean value of the good data.
mu1 <- 0

## The mean value of the bad data.
mu2 <- 0.5

## A grid of proportions "p" to consider.
p <- seq(0, 1, 0.05)
V <- matrix(0, length(p), 3)
colnames(V) <- c("Bias", "Variance", "MSE")
```

```
set.seed(111111)
for (k in 1:length(p)) {
    X1 <- matrix(rnorm(nrep*n, mean=mu1), nrep, n)
    M1 <- apply(X1, 1, mean)

    X2 <- matrix(rnorm(nrep*n, mean=mu2), nrep, n)
    M2 <- apply(X2, 1, mean)

    M <- p[k]*M1 + (1 - p[k])*M2

    bias <- mean(M) - mu1
    variance <- var(M)
    mse <- mean((M - mu1)^2)
    V[k,] <- c(bias, variance, mse)
}
cbind(p, V)
```