Getting a "Bad" Sample

Steve Swann

07 September 2015

Introduction

We take a random sample S from a population P. Though pure bad luck we draw a "bad" sample which does not really reflect the underlying population. How likely is this to happen and what can we do about it.

Population

We create a normally distributed population P.

```
# population P normal distribution of size n=100,000
P = rnorm(n=100000, mean=25, sd=5)
```

Generalised function for population mean interval estimate

```
# returns population mean confidence interval
# pass sample of values and confidence level
pop.mean.ci = function(sample, level) {
  s.mean = mean(sample)
  s.sd = sd(sample)
  s.size = length(sample)
  se = s.sd/sqrt(s.size)
  lower.bound = qnorm(p=(1 - level)/2, mean=s.mean, sd=se)
  upper.bound = s.mean + (s.mean - lb)
  return (c(lower.bound, upper.bound))
}

# True if value on interval
# Pass value, bounds=vector(lower.bound, upper.bound)
is.on.interval = function(value, bounds) {
  if ((value < bounds[1]) | (value > bounds[2])) {
    FALSE
  } else {TRUE}
}
```

"Bad sample"" from the population

```
# contrived "bad" sample
sample.low.draw = P[which(P<12.5)]
mean.sample.low.draw = mean(sample.low.draw)
sd.sample.low.draw = mean(sample.low.draw)
# standard error of the mean
se.mean = sd.sample.low.draw/sqrt(length(sample.low.draw))
```

```r
# confidence level
cl = .95
lb = qnorm(p=(1 - cl)/2, mean=mean.sample.low.draw, sd=se.mean)
ub = mean.sample.low.draw + (mean.sample.low.draw - lb)
print (lb)
```

```
## [1] 9.906209
```

```r
print (ub)
```

```
## [1] 11.6436
```

```r
result = pop.mean.ci(sample.low.draw, 0.95)
print (result)
```

```
## [1] 10.65864 11.64360
```

Randomly generated bad samples

```r
no.of.random.samples = 1000
size.of.sample = 1000
# sample.means will contain each sample.mean
bad.samples.count = 0
i = 1
while (i <= no.of.random.samples) {
  s = sample(P, size=size.of.sample, replace=FALSE)
  if (!is.on.interval(25, pop.mean.ci(s, 0.95))) {
    bad.samples.count = bad.samples.count + 1
  }
  i = i + 1
}

print (bad.samples.count)
```

```
## [1] 15
```