Introduction to the Practice of Statistics using R: Chapter 6

Ben Baumer Nicholas J. Horton[*]

March 10, 2013

Contents

Introduction

This document is intended to help describe how to undertake analyses introduced as examples in the Sixth Edition of *Introduction to the Practice of Statistics* (2009) by David Moore, George McCabe and Bruce Craig. More information about the book can be found at `http://bcs.whfreeman.com/ips6e/`. This file as well as the associated `knitr` reproducible analysis source file can be found at `http://www.math.smith.edu/~nhorton/ips6e`.

This work leverages initiatives undertaken by Project MOSAIC (`http://www.mosaic-web.org`), an NSF-funded effort to improve the teaching of statistics, calculus, science and computing in the undergraduate curriculum. In particular, we utilize the `mosaic` package, which was written to simplify the use of R for introductory statistics courses. A short summary of the R needed to teach introductory statistics can be found in the mosaic package vignette (`http://cran.r-project.org/web/packages/mosaic/vignettes/MinimalR.pdf`).

To use a package within R, it must be installed (one time), and loaded (each session). The package can be installed using the following command:

```
> install.packages('mosaic')              # note the quotation marks
```

The `#` character is a comment in R, and all text after that on the current line is ignored.

Once the package is installed (one time only), it can be loaded by running the command:

[*] Department of Mathematics and Statistics, Smith College, nhorton@smith.edu

```
> require(mosaic)
```

This needs to be done once per session.
We also set some options to improve legibility of graphs and output.

```
> trellis.par.set(theme=col.mosaic())  # get a better color scheme for lattice
> options(digits=3)
```

The specific goal of this document is to demonstrate how to replicate the analysis described in Chapter 6: Introduction to Inference.

1 Estimating with Confidence

First, let's generate a random sample of 500 SAT scores drawn from a normal distribution with mean 500 and standard deviation 100. To do this we use the `rnorm()` function, which draws from a normal distribution.

```
> mu = 500
> sigma = 100
> x = rnorm(500, mean=mu, sd=sigma)
> favstats(x)

 min  Q1 median  Q3 max mean   sd   n missing
 195 430    500 566 773  500 98.6 500       0
```

To compute a confidence interval for the mean, we'll use a simple function that finds a confidence interval for the mean of any vector of data x, given a specified significance level and the true (assumed known) population standard deviation. Note that 95% is the default level of confidence.

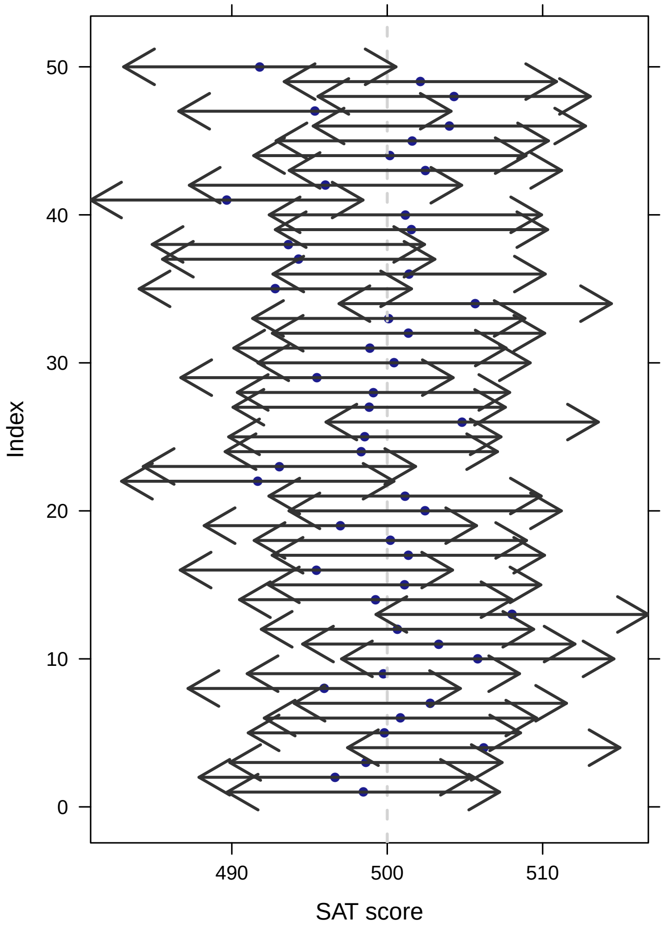
```
> meanconfint = function (x, sigma, level = 0.95, ...) {
   se = sigma / sqrt(length(x))
   mu = mean(x)
   z = qnorm(1 - (1 - level)/2)
   out = c(mu, mu - z * se, mu + z * se)
   names(out) = c("mean", "lower", "upper")
   return(out)
 }
> meanconfint(x, sigma = sigma)

 mean lower upper
  500   492   509
```

At the bottom of page 358, many such confidence intervals are calculated. We can simulate this using our function. The `do()` function will repeat any operation a specified number of times, and return a data frame of the results. The `apply()` family of functions provide a powerful way to apply an operation to the rows or columns of a data frame. Here it lets us repeat an operation for each of the 50 sets of 500 random numbers.

```
> randomx = do(50) * rnorm(500, mean=mu, sd=sigma)
> ci = data.frame(t(apply(randomx, 1, meanconfint, sigma=sigma)))
> head(ci, 3)

  mean lower upper
1  498   490   507
2  497   488   505
3  499   490   507
```

Let's try to visualize these intervals in a manner analogous to the plot on the bottom of page 358.

```
> xyplot(1:nrow(ci) ~ mean, data=ci, xlim=range(ci), xlab="SAT score", ylab="Index")
> ladd(panel.abline(v=500, col="lightgray", lty=2))
> ladd(with(ci, panel.arrows(x0 = lower, y0=1:nrow(ci), y1=1:nrow(ci), cex=0.5,
   x1=upper, code=3)))
```



We see that sometimes (e.g. simulation 41) the confidence interval does *not* cover the true population mean (500 points).

Note that we can consider confidence levels other than 0.95 by specifying the `level` argument. Here's how we compute a 90% confidence interval.

```
> head(t(apply(randomx, 1, meanconfint, sigma=sigma, level=0.9)), 3)

     mean lower upper
[1,]  498   491   506
[2,]  497   489   504
[3,]  499   491   506
```

The 90% confidence intervals are narrower than the 95% confidence intervals, since we sacrifice some accuracy in exchange for increased confidence that the interval will contain the true mean.

In Example 6.4 (page 361), we are asked to compute a 95% confidence interval for a sample mean of $18,900 in undergraduate debt, computed from a sample of 1280 borrowers. The standard deviation of the population is known to be $49,000. Since we want a 95% confidence interval, we need to find the z-score that corresponds to 0.025 (or equivalently 0.0975), since 95% of the standard normal distribution lies between these two values.

```
> z.star = qnorm(0.975)
> z.star

[1] 1.96
```

Then we compute the margin or error and the confidence interval by:

```
> se = z.star * (49000) / sqrt(1280)
> se

[1] 2684

> 18900 + c(-se, se)

[1] 16216 21584
```

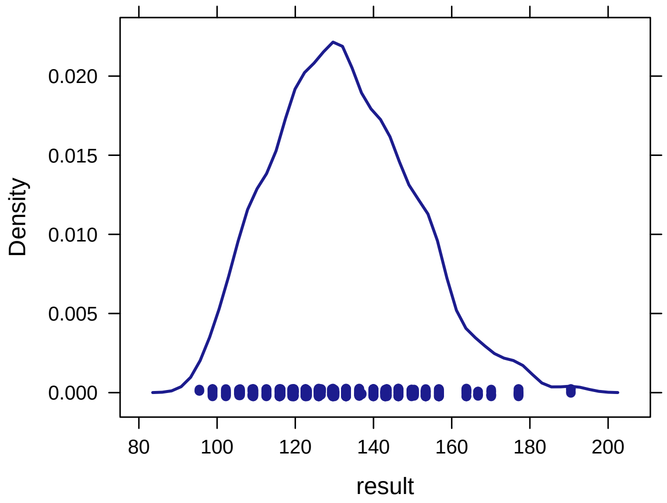
In Example 6.6 (page 364), we change the confidence level to 99%. Thus, we need to compute a different value of z^*.

```
> z.star2 = qnorm(0.995)
> z.star2

[1] 2.58

> se2 = z.star2 * (49000) / sqrt(1280)
> se2

[1] 3528

> 18900 + c(-se2, se2)

[1] 15372 22428
```

1.1 Beyond the Basics

We'll discuss the bootstrap in much greater detail in Chapter 16. Here, we can use the `resample()` function from `mosaic` to quickly compute a bootstrap sample.

```
> time = c(190.5, 109, 95.5, 137)
> resample(time)

[1] 190.5  95.5 190.5 109.0

> bootstrap = do(1000) * mean(resample(time))
> densityplot(~result, data=bootstrap)
```



2 Tests of Significance

In Example 6.12 (page 378), we compute a p-value for the observed difference of $4,100. Note that we need to multiply the cumulative probability in the right-hand tail by 2 for a two-sided test.

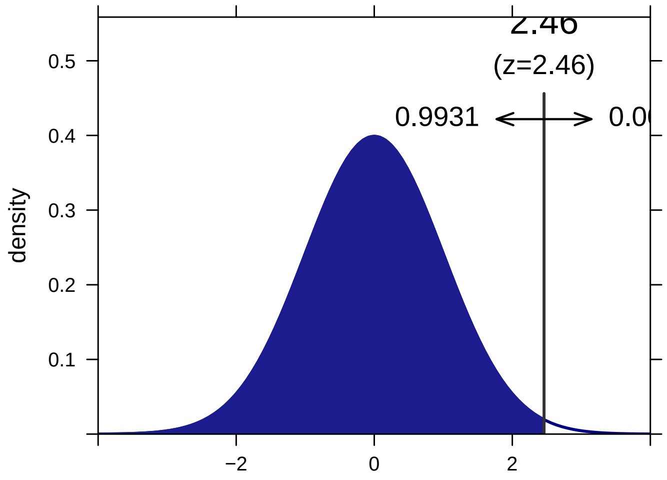
```
> z = (4100 - 0) / 3000
> z

[1] 1.37

> 2 * (1 - pnorm(z))

[1] 0.172
```

The z-test for a population mean on page 383 can be computed using the `pbinom()`.

```
> # one-sided test for right tail probability
> pnorm(2, lower.tail=FALSE)

[1] 0.0228

> # one-sided test for left tail probability
> pnorm(-2)

[1] 0.0228

> # two-sided test
> 2 * pnorm(2, lower.tail=FALSE)

[1] 0.0455
```

In Example 6.16 (page 385), we find the right-hand tail probability.

```
> pnorm(461, mean=450, sd=100 / sqrt(500), lower.tail=FALSE)

[1] 0.00695

> xpnorm(2.46)


If X ~ N(0,1), then

P(X <= 2.46) = P(Z <= 2.46) = 0.9931
P(X >  2.46) = P(Z >  2.46) = 0.0069
[1] 0.993
```



3 Use and Abuse of Tests

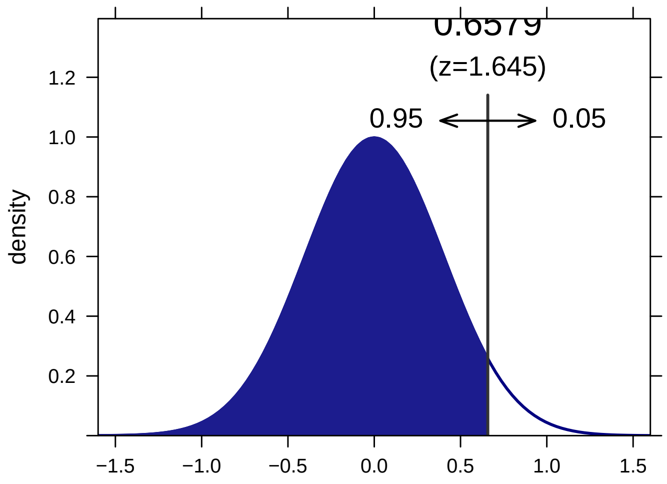
In Example 6.84 (page 396), we test for significance. Note the use of a one-sided test.

```
> z1 = (541.4 - 525) / (100 / sqrt(100))
> pnorm(z1, lower.tail=FALSE)

[1] 0.0505

> z2 = (541.5 - 525) / (100 / sqrt(100))
> pnorm(z2, lower.tail=FALSE)

[1] 0.0495
```

4 Power and Inference as a Decision

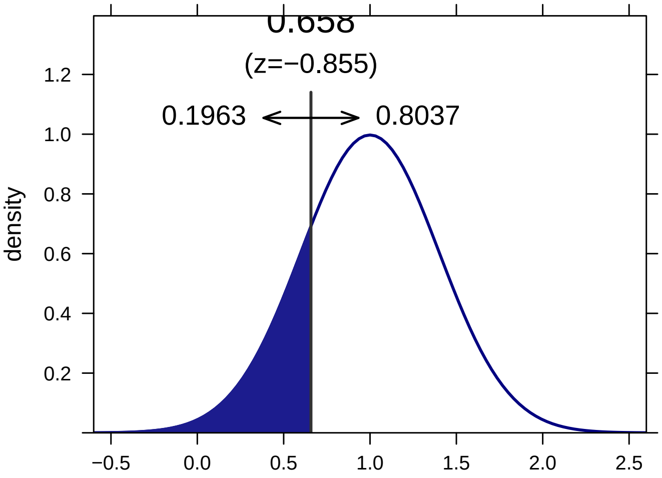
Example 6.29 (page 402) considers the power for a study with n=25 subjects, where a one-sided alternative is tested at an α level of 0.05 and the population standard deviation is assumed known and equals $\sigma = 2$.

```
> xqnorm(.95, mean=0, sd=2/sqrt(25))

P(X <= 0.657941450780589) = 0.95
P(X >  0.657941450780589) = 0.05
[1] 0.658
```



We can now compare this to the distribution when the alternative is true ($\mu = 1$).

```
> xpnorm(0.658, mean=1, sd=2/sqrt(25))


If X ~ N(1,0.4), then

P(X <= 0.658) = P(Z <= -0.855) = 0.1963
P(X >  0.658) = P(Z >  -0.855) = 0.8037
[1] 0.196
```



We see that the power is 0.80.