

# Simulation for Inference

Bret Larget

Departments of Botany and of Statistics  
University of Wisconsin—Madison

March 25, 2008

- We will use *simulation* to assess uncertainty in predictions and in estimated regression coefficients.
- This is in contrast to deriving formulas for standard errors.
- One advantage is that we can assess uncertainty in quantities for which there are not formulas.
- A second advantage is we do not need to learn how to derive formulas.
- However, we will require learning to write small programs in R.

# How many girls?

- How many girls do we expect in 500 births?
- The population probability of being a girl is 0.488.
- This is a relatively simple problem for which you learned a formula last semester.
- We will compare answers with simulation and the formula.

- We can model the number of girls with a *binomial distribution*.
- The mean is

$$\mu = np = 500(0.488) = 244$$

- The standard deviation of the binomial distribution is

$$\sigma = \sqrt{np(1-p)} = \sqrt{500(0.488)(0.512)} \doteq 11.2$$

- With a normal approximation, we expect about a 95% chance that the number of girls would be between  $244 \pm 2 \times 11$  or from 222 to 266.
- A precise calculation of this probability is  
> `sum(dbinom(222:266, 500, 0.488))`

```
[1] 0.9559977
```

# Simulation Approach

- We begin by *writing a function* in R to simulate the number of girls in 500 births from the binomial distribution.
- The function `rbinom()` simulates draws from the binomial distribution.
- The arguments are, in order, the number of draws,  $n$ , and  $p$ .
- Unfortunately, these are named `n`, `size`, and `prob`.

```
> girls.sim = function(n, p) {  
+   return(rbinom(n = 1, size = n, prob = p))  
+ }  
> girls.sim(500, 0.488)
```

```
[1] 244
```

## Simulation Approach (cont.)

- Use the `replicate()` function to repeat this many times, say 1000.
- The object `n.girls` stores the answers.

```
> n.girls = replicate(1000, girls.sim(500, 0.488))
```

```
> mean(n.girls)
```

```
[1] 244.276
```

```
> sd(n.girls)
```

```
[1] 10.58394
```

```
> sum((n.girls >= 222) & (n.girls <= 266))/1000
```

```
[1] 0.962
```

## A Simpler Version

- For this particular example, we could have done this simulation with a single R command without writing a function.
- However, the paradigm of writing a function to do one simulation and then using `replicate()` to repeat this will be helpful for more complicated examples.

```
> n.girls2 = rbinom(1000, 500, 0.488)
```

```
> mean(n.girls2)
```

```
[1] 243.681
```

```
> sd(n.girls2)
```

```
[1] 11.01583
```

## Sibling Example

- What if we wanted a more complicated example where we modeled the number of girls from a Beta-Binomial distribution with overdispersion similar to that observed?
- Finding a formula in this situation would be a lot of work, but a simulation is not too bad.
- The Beta distribution is for continuous numbers between 0 and 1 and is defined by has two parameters,  $\alpha$  and  $\beta$ .
- Setting  $\alpha = 9(0.488)$  and  $\beta = 9(0.512)$  fits the data well.

## Sibling Example (cont.)

- We use the function `rbetabin.ab()` from the VGAM library for the Beta-Binomial distribution.

```
> siblings = read.table("siblings.txt", header = T)
> family.size = with(siblings, boy + girl)
> sum(family.size)
```

```
[1] 145
```

```
> with(siblings, sum(girl))
```

```
[1] 72
```

```
> library(VGAM)
> girls.sim = function(sizes, p, M) {
+   return(sum(rbetabin.ab(length(sizes), sizes, p *
+     M, (1 - p) * M)))
+ }
> g = replicate(1000, girls.sim(sizes = family.size, p = 0.488,
+   M = 9))
```

## Sibling Example (cont.)

```
> mean(g)
```

```
[1] 70.87
```

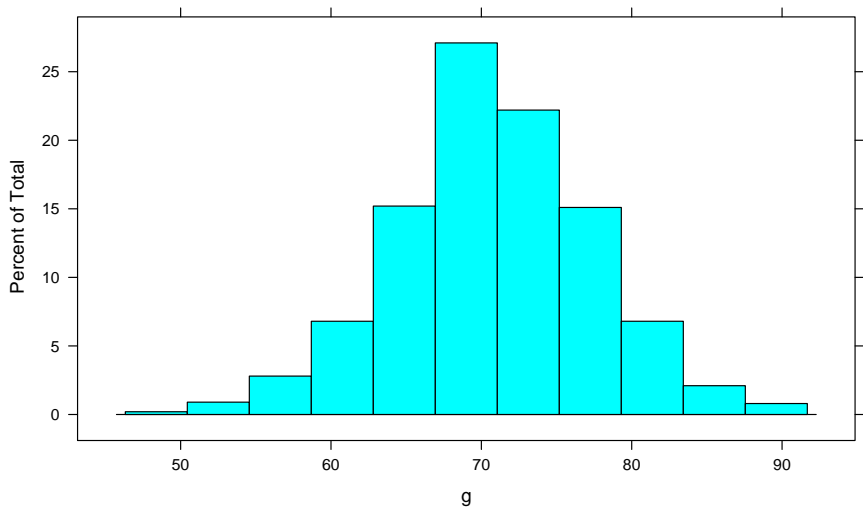
```
> sd(g)
```

```
[1] 6.801129
```

```
> quantile(g, c(0.025, 0.05, 0.5, 0.95, 0.975))
```

2.5%	5%	50%	95%	97.5%
57	60	71	82	84

## Sibling Example (cont.)



- If you use the VGAM library, it will mess up the `predict()` function.
- This command will unload a library.

```
> detach("package:VGAM")
```