Statistical Rethinking

Week 2:
Linear Models

Richard McElreath

R code 4.40

```
precis( m4.3 )
```

```
          Mean StdDev   5.5%  94.5%
a       113.90   1.91 110.85 116.94
b         0.90   0.04   0.84   0.97
sigma     5.07   0.19   4.77   5.38
```

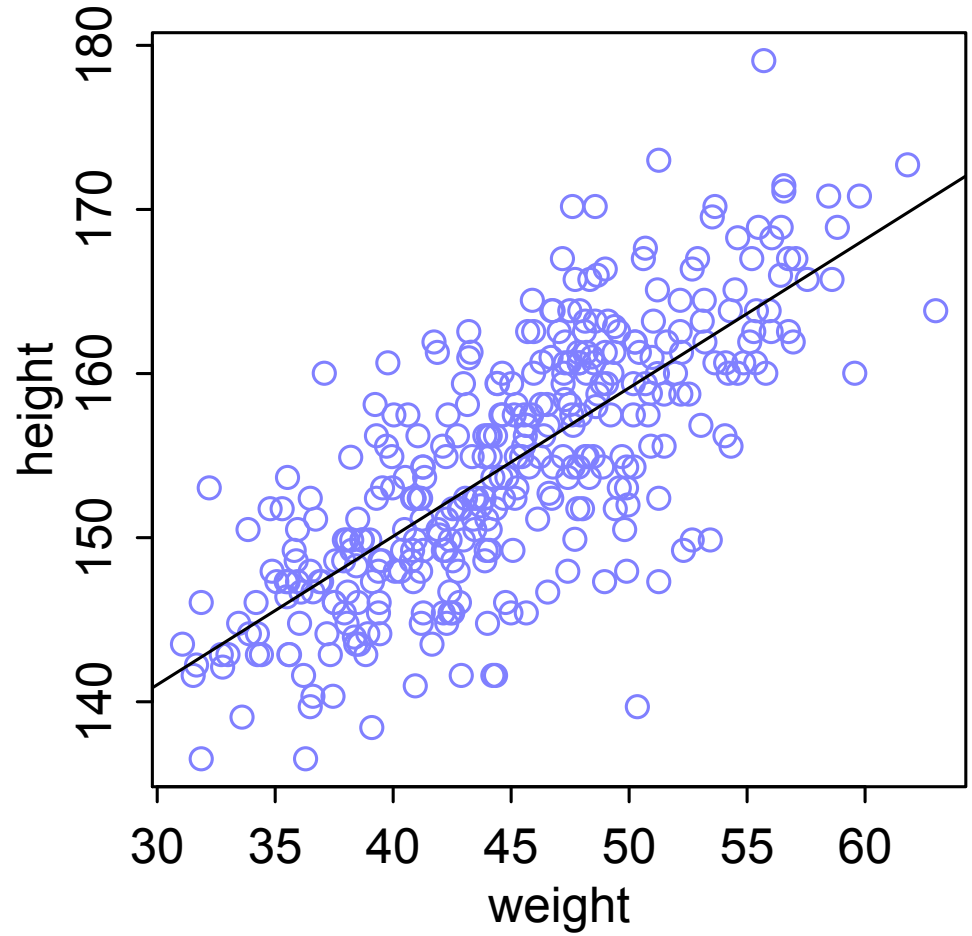


FIGURE 4.4. Height in centimeters (vertical) plotted against weight in kilograms (horizontal), with the *maximum a posteriori* line for the mean height at each weight plotted in black.

Sampling from the posterior

- Want to get uncertainty onto that graph
- Again, sample from posterior
  1. Use MAP and standard deviation to approximate posterior
  2. Sample from *multivariate normal* distribution of parameters
  3. Use samples to generate predictions that "integrate over" the uncertainty

Sampling from the posterior

```
post <- extract.samples( m4.3 )
```

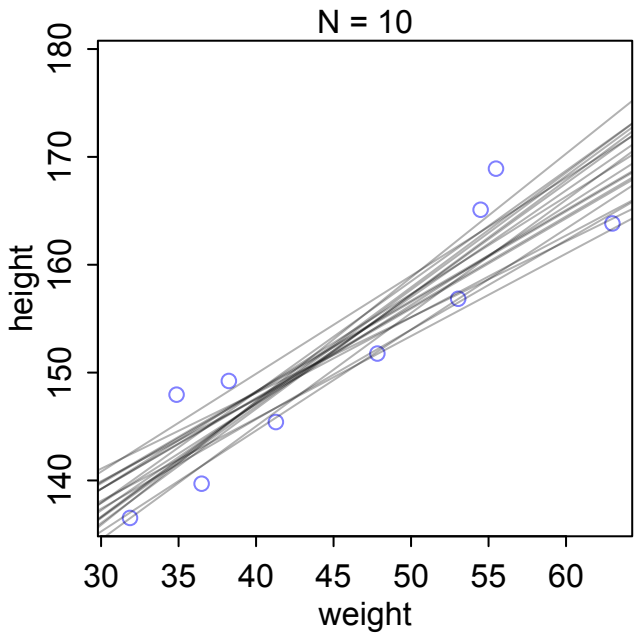
R code 4.46

```
post[1:5,]
```

R code 4.47

```
         a         b    sigma
1 114.7880 0.8822921 5.121102
2 112.7115 0.9230855 4.907987
3 114.4557 0.9018482 5.276036
4 114.7696 0.8831561 5.021958
5 112.6333 0.9383632 4.898554
```

Posterior is full of lines




```
post[1:5,]
```

```
          a    sigma         b
1  115.1964 4.992267 0.8776393
2  111.0389 5.169515 0.9758554
3  115.4833 5.133463 0.8726757
4  109.6488 5.005837 0.9812692
5  112.4637 4.678314 0.9384814
```

Figure 4.5

Posterior is full of lines

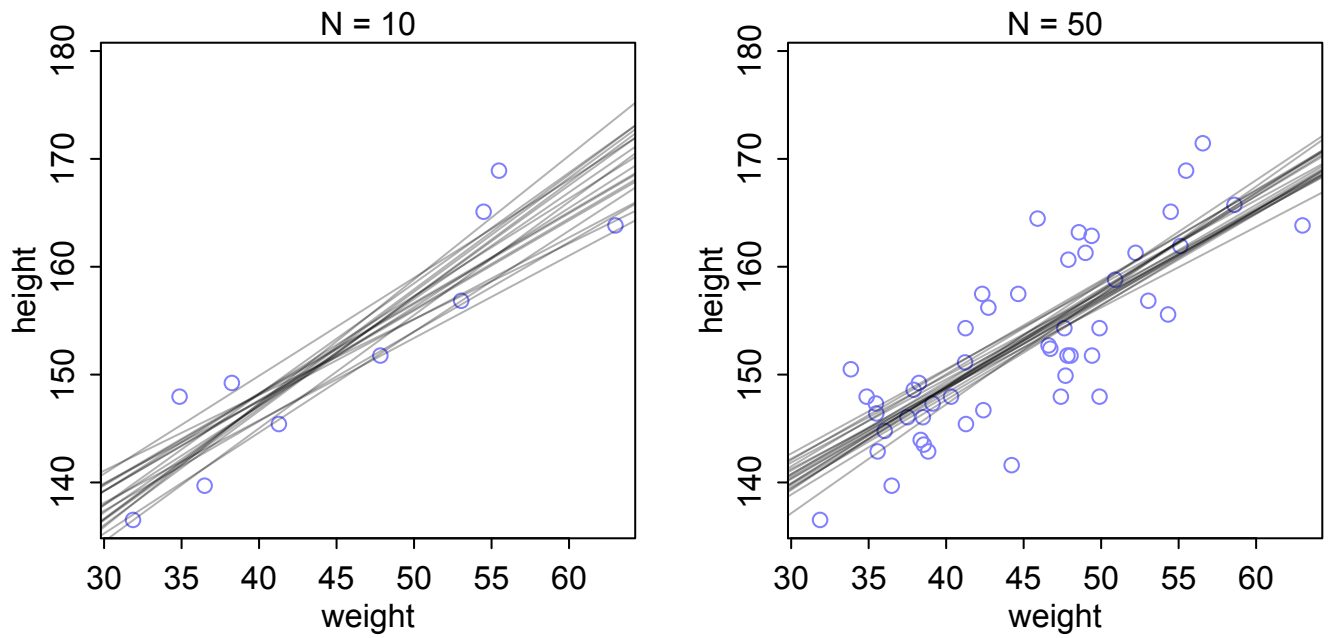


Figure 4.5

Posterior is full of lines

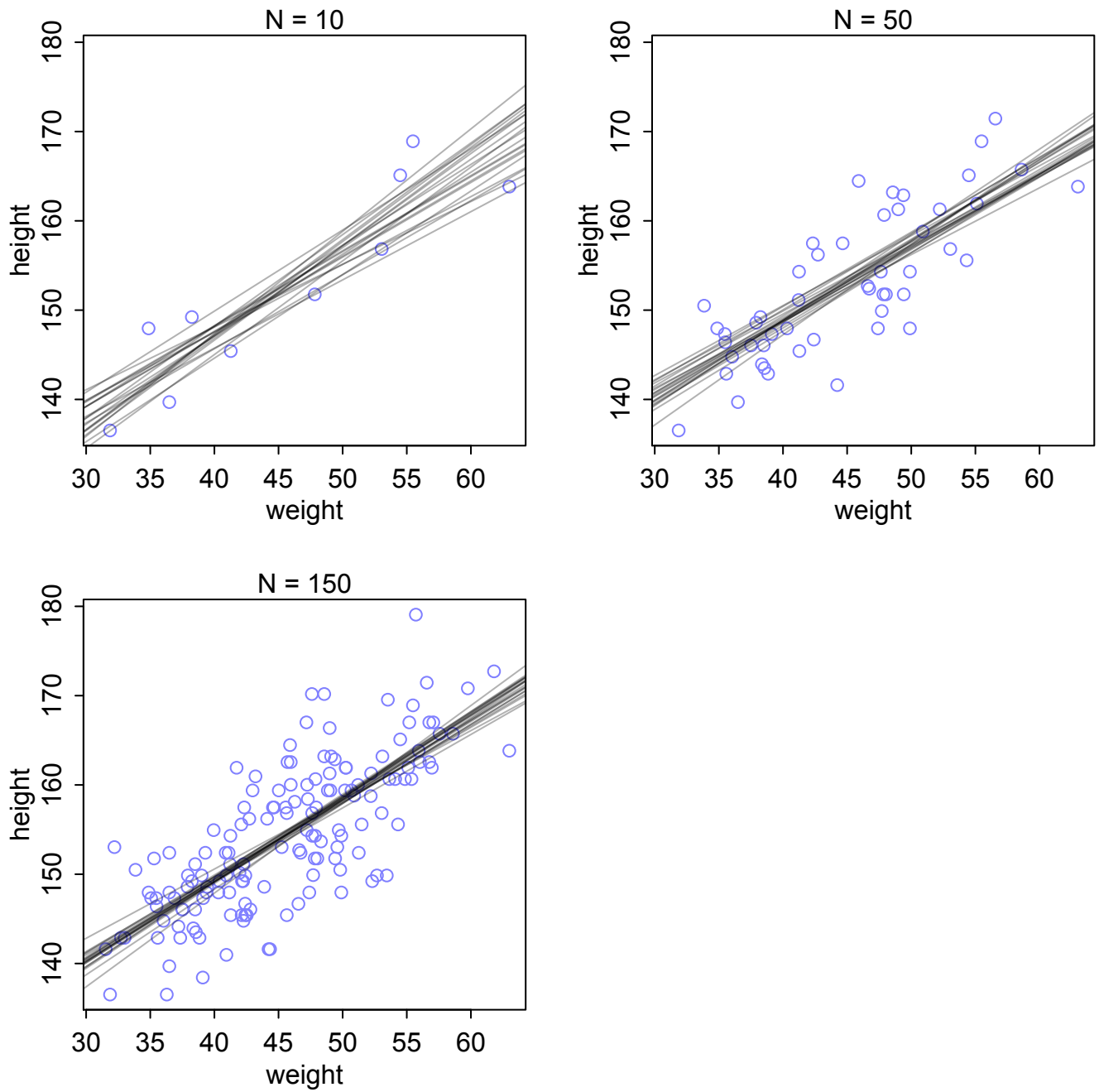


Figure 4.5

Posterior is full of lines

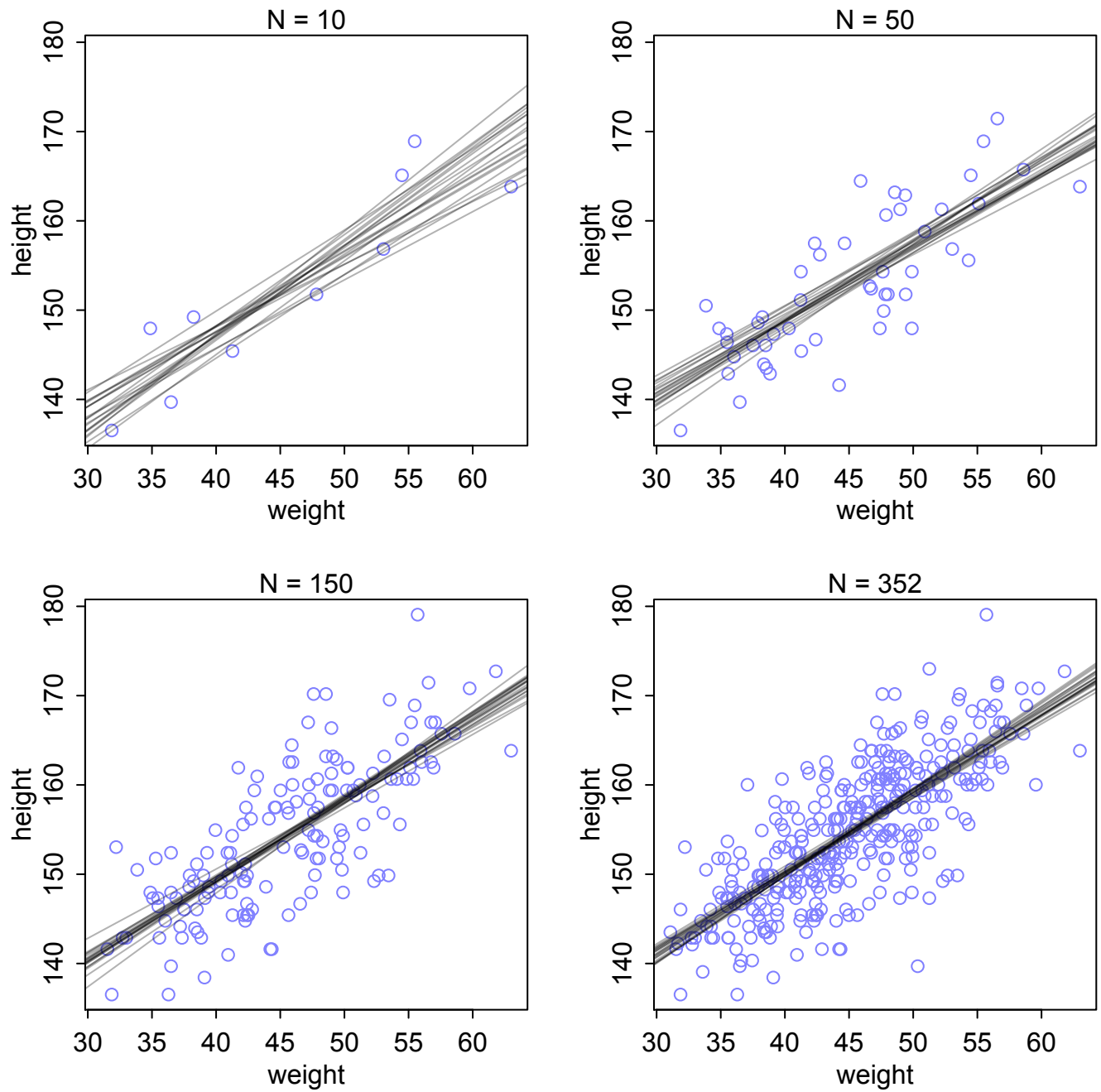


Figure 4.5

Predict mu

```
post[1:5,]
```

R code 4.47

```
         a         b    sigma
1 114.7880 0.8822921 5.121102
2 112.7115 0.9230855 4.907987
3 114.4557 0.9018482 5.276036
4 114.7696 0.8831561 5.021958
5 112.6333 0.9383632 4.898554
```

```
mu_at_50 <- post$a + post$b * 50
```

R code 4.50

$$\mu_i = \alpha + \beta x_i$$

Predict mu

```
mu_at_50 <- post$a + post$b * 50
```

R code 4.50

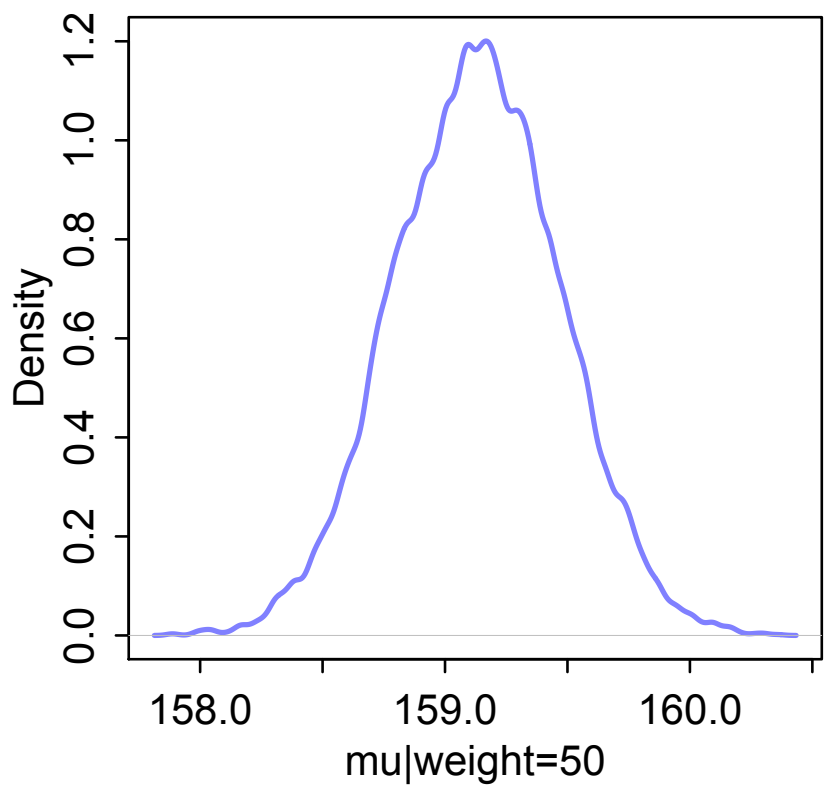


Figure 4.6

Predict every mu

R code 4.54

```
# define sequence of weights to compute predictions for
# these values will be on the horizontal axis
weight.seq <- seq( from=25 , to=70 , by=1 )

# use link to compute mu
# for each sample from posterior
# and for each weight in weight.seq
mu <- link( m4.3 , data=data.frame(weight=weight.seq) )
str(mu)
```

```
 num [1:1000, 1:46] 137 136 137 137 136 ...
```

How `link` works

- Sample from posterior
- Define series of predictor (weight) values
- For each predictor value
  - For each sample from posterior
    - Compute mu: a + b*weight
- Summarize

```
post <- extract.samples(m4.3)
mu.link <- function(weight) post$a + post$b*weight
weight.seq <- seq( from=25 , to=70 , by=1 )
mu <- sapply( weight.seq , mu.link )
mu.mean <- apply( mu , 2 , mean )
mu.HPDI <- apply( mu , 2 , HPDI )
```

R code 4.57

```
> post <- extract.samples(m4.3)
> str(post)
'data.frame':  10000 obs. of  3 variables:
 $ a    : num  115 116 114 111 111 ...
 $ b    : num  0.885 0.871 0.9 0.989 0.981 ...
 $ sigma: num  5.07 5.23 5.3 4.84 5.07 ...
```

1. sample from posterior

```
> post <- extract.samples(m4.3)
> str(post)
```

```
'data.frame': 10000 obs. of  3 variables:
 $ a    : num  115 116 114 111 111 ...
 $ b    : num  0.885 0.871 0.9 0.989 0.981 ...
 $ sigma: num  5.07 5.23 5.3 4.84 5.07 ...
```

1. sample from posterior

```
> mu.link <- function(weight) post$a + post$b*weight
> str( mu.link(50) )
```

```
 num [1:10000] 159 159 159 160 160 ...
```

2. define link function

```
> post <- extract.samples(m4.3)
> str(post)
```

```
'data.frame': 10000 obs. of  3 variables:
 $ a    : num  115 116 114 111 111 ...
 $ b    : num  0.885 0.871 0.9 0.989 0.981 ...
 $ sigma: num  5.07 5.23 5.3 4.84 5.07 ...
```

1. sample from posterior

```
> mu.link <- function(weight) post$a + post$b*weight
> str( mu.link(50) )
```

```
 num [1:10000] 159 159 159 160 160 ...
```

2. define link function

```
> weight.seq <- seq( from=25 , to=70 , by=1 )
> str( weight.seq )
```

```
 num [1:46] 25 26 27 28 29 30 31 32 33 34 ...
```

3. define weight values to compute predictions for

```
> post <- extract.samples(m4.3)
> str(post)
'data.frame':  10000 obs. of  3 variables:
 $ a    : num  115 116 114 111 111 ...
 $ b    : num  0.885 0.871 0.9 0.989 0.981 ...
 $ sigma: num  5.07 5.23 5.3 4.84 5.07 ...
```

1. sample from posterior

```
> mu.link <- function(weight) post$a + post$b*weight
> str( mu.link(50) )
 num [1:10000] 159 159 159 160 160 ...
```

2. define link function

```
> weight.seq <- seq( from=25 , to=70 , by=1 )
> str( weight.seq )
 num [1:46] 25 26 27 28 29 30 31 32 33 34 ...
```

3. define weight values to compute predictions for

```
> mu <- sapply( weight.seq , mu.link )
> str( mu )
 num [1:10000, 1:46] 137 138 136 135 135 ...
```

4. compute prediction for each sample in posterior, for each weight value

`sapply` (simplified apply)

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

`sapply` (simplified apply)

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
sapply( 1:10 , function(z) z^2 )
```

```
[1]   1   4   9  16  25  36  49  64  81 100
```

`sapply` (simplified apply)

```
1:10
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
sapply( 1:10 , function(z) z^2 )
```

```
[1]   1   4   9  16  25  36  49  64  81 100
```

```
sapply( 1:10 , function(z) prod(1:z)^(1/z) )
```

```
[1] 1.000000 1.414214 1.817121 2.213364 2.605171 2.993795
[7] 3.380015 3.764351 4.147166 4.528729
```

```
> post <- extract.samples(m4.3)
> str(post)
'data.frame':  10000 obs. of  3 variables:
 $ a    : num  115 116 114 111 111 ...
 $ b    : num  0.885 0.871 0.9 0.989 0.981 ...
 $ sigma: num  5.07 5.23 5.3 4.84 5.07 ...
```

1. sample from posterior

```
> mu.link <- function(weight) post$a + post$b*weight
> str( mu.link(50) )
 num [1:10000] 159 159 159 160 160 ...
```

2. define link function

```
> weight.seq <- seq( from=25 , to=70 , by=1 )
> str( weight.seq )
 num [1:46] 25 26 27 28 29 30 31 32 33 34 ...
```

3. define weight values to compute predictions for

```
> mu <- sapply( weight.seq , mu.link )
> str( mu )
 num [1:10000, 1:46] 137 138 136 135 135 ...
```

4. compute prediction for each sample in posterior, for each weight value

```
> post <- extract.samples(m4.3)
> str(post)
'data.frame':  10000 obs. of  3 variables:
 $ a    : num  115 116 114 111 111 ...
 $ b    : num  0.885 0.871 0.9 0.989 0.981 ...
 $ sigma: num  5.07 5.23 5.3 4.84 5.07 ...
```

1. sample from posterior

```
> mu.link <- function(weight) post$a + post$b*weight
> str( mu.link(50) )
 num [1:10000] 159 159 159 160 160 ...
```

2. define link function

```
> weight.seq <- seq( from=25 , to=70 , by=1 )
> str( weight.seq )
 num [1:46] 25 26 27 28 29 30 31 32 33 34 ...
```

3. define weight values to compute predictions for

```
> mu <- sapply( weight.seq , mu.link )
> str( mu )
 num [1:10000, 1:46] 137 138 136 135 135 ...
```

4. compute prediction for each sample in posterior, for each weight value

```
> mu.mean <- apply( mu , 2 , mean )
> str( mu.mean )
 num [1:46] 137 137 138 139 140 ...
```

5. summarize

Predict every mu

R code 4.46

```
weight.seq <- seq( from=30 , to=63 , by=1 )
mu.ci <- sapply( weight.seq , function(x) HPDI( post$a + post$b*x ) )
```

Predict every mu

R code 4.46

```
weight.seq <- seq( from=30 , to=63 , by=1 )
mu.ci <- sapply( weight.seq , function(x) HPDI( post$a + post$b*x ) )
```

Predict every mu

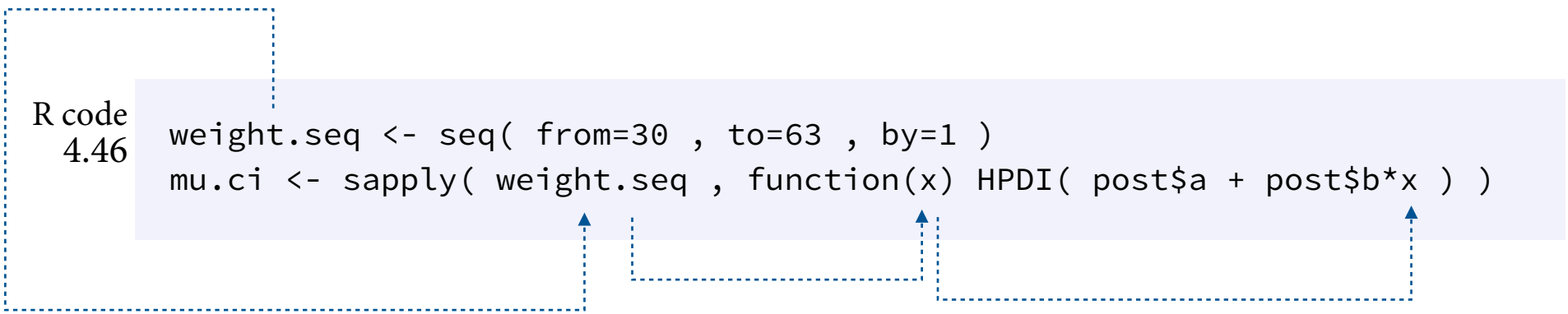
R code 4.46

```
weight.seq <- seq( from=30 , to=63 , by=1 )
mu.ci <- sapply( weight.seq , function(x) HPDI( post$a + post$b*x ) )
```

Predict every mu

R code 4.46

```
weight.seq <- seq( from=30 , to=63 , by=1 )
mu.ci <- sapply( weight.seq , function(x) HPDI( post$a + post$b*x ) )
```



Predict every mu

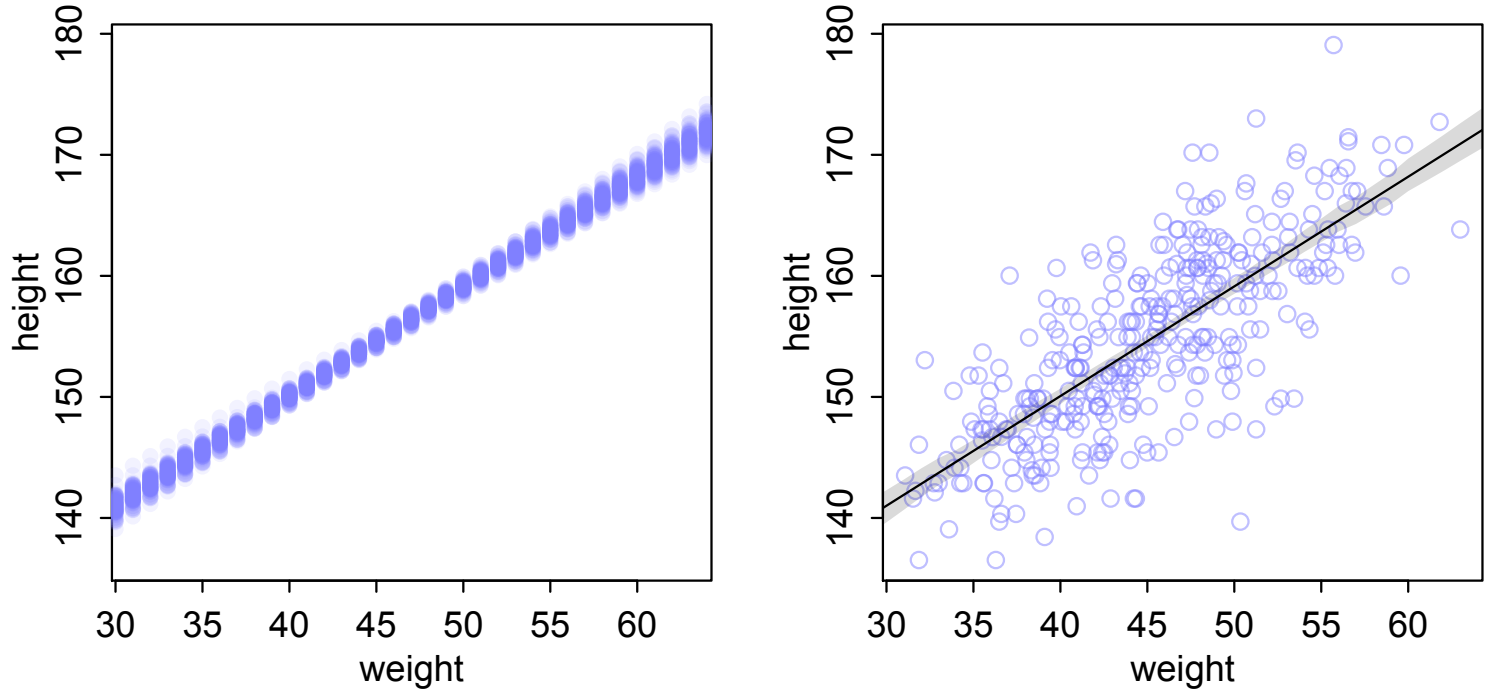
R code 4.46

```
weight.seq <- seq( from=30 , to=63 , by=1 )
mu.ci <- sapply( weight.seq , function(x) HPDI( post$a + post$b*x ) )
```

```
mu.ci[,1:5]
```

```
                [,1]     [,2]     [,3]     [,4]     [,5]
lower 0.95 139.6944 140.6860 141.6516 142.6507 143.6348
upper 0.95 142.3601 143.2046 144.0292 144.8874 145.7358
```

Figure 4.7

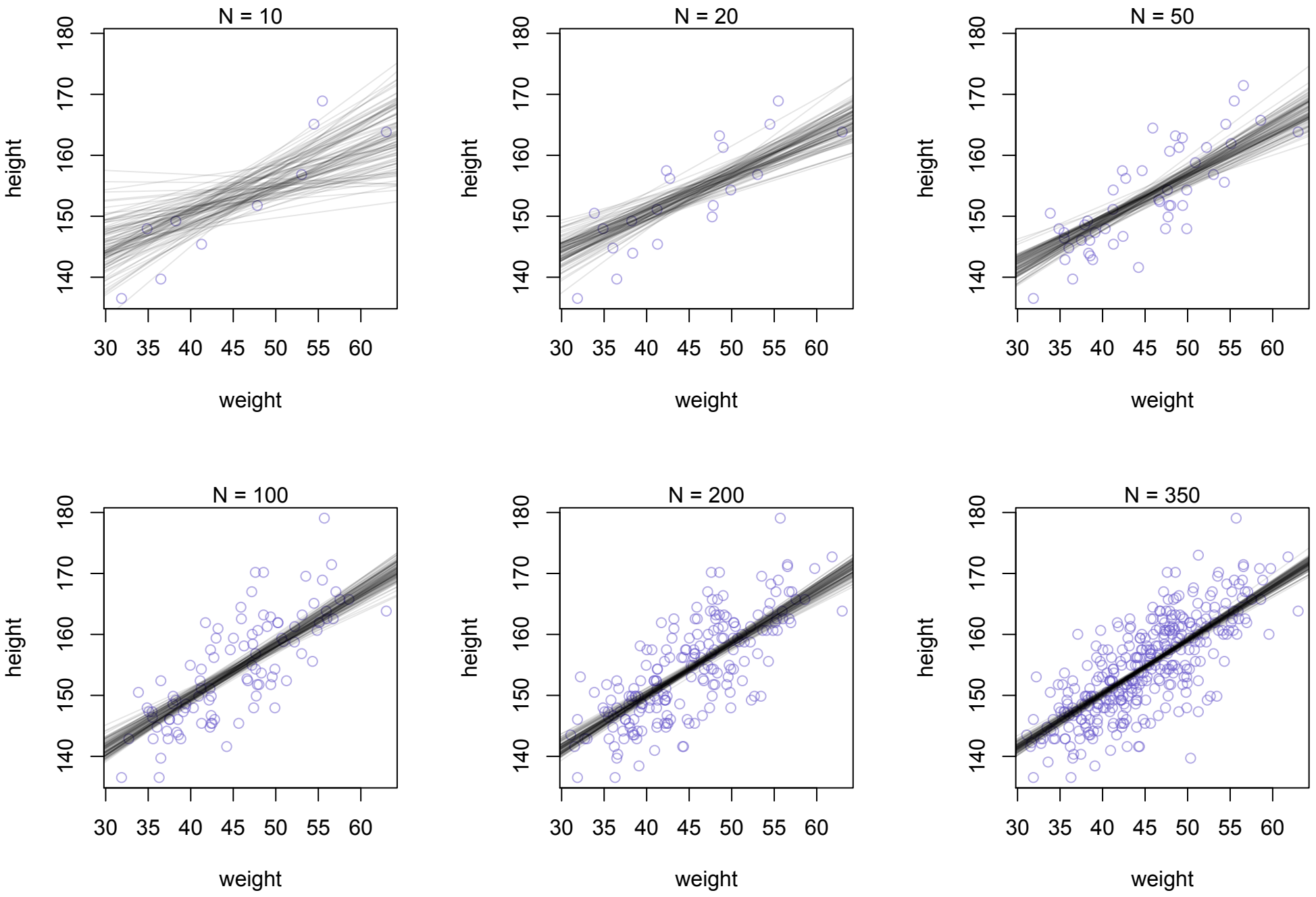


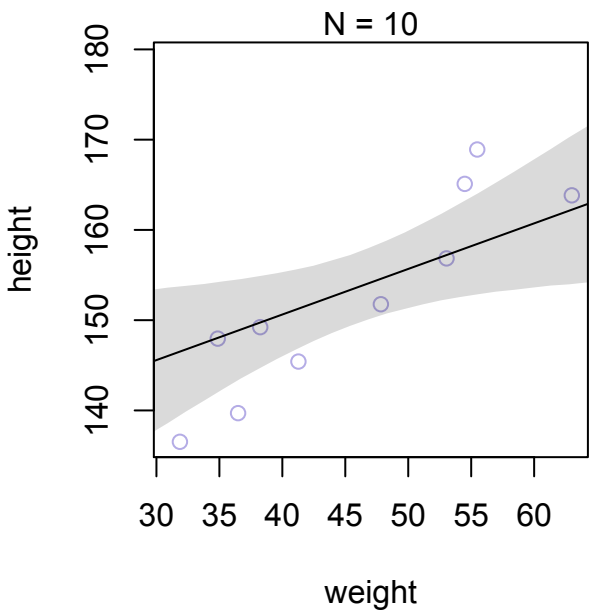
R code 4.56

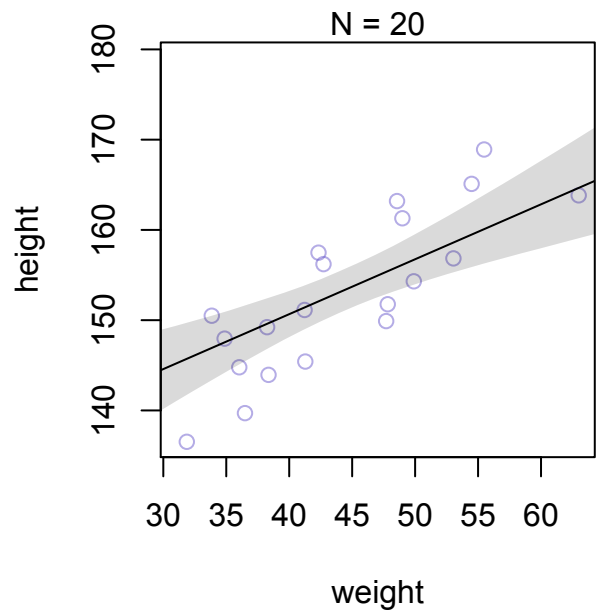
```
# summarize the distribution of mu
mu.mean <- apply( mu , 2 , mean )
mu.HPDI <- apply( mu , 2 , HPDI , prob=0.89 )
```

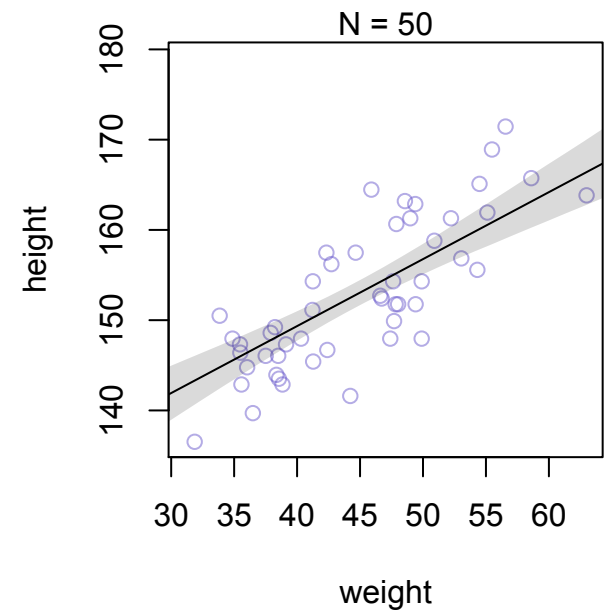
R code 4.57

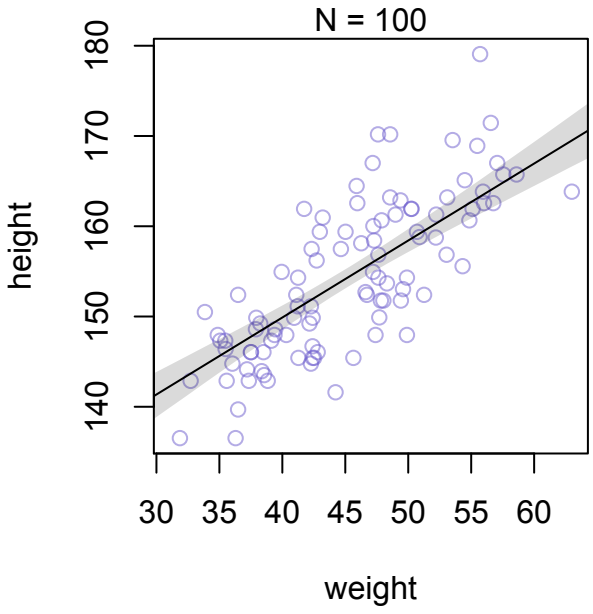
```
# plot raw data
# fading out points to make line and interval more visible
plot( height ~ weight , data=d2 , col=col.alpha(rangi2,0.5) )

# plot the MAP line, aka the mean mu for each weight
lines( weight.seq , mu.mean )

# plot a shaded region for 89% HPDI
shade( mu.HPDI , weight.seq )
```

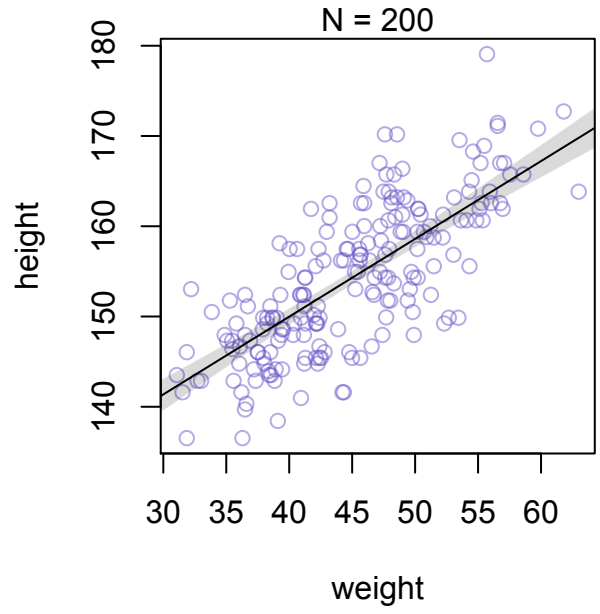

N = 10
N = 20
N = 50
N = 100
N = 200
N = 350
height
weight
140
150
160
170
180
30
35
40
45
50
55
60

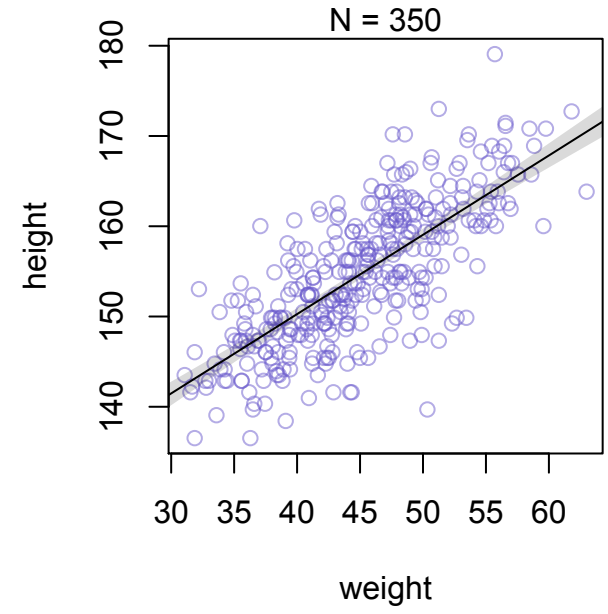

N = 10
height
weight
140
150
160
170
180
30
35
40
45
50
55
60


N = 20
height
weight
140
150
160
170
180
30
35
40
45
50
55
60


N = 50
height
weight
140
150
160
170
180
30
35
40
45
50
55
60


N = 100
height
weight
140
150
160
170
180
30
35
40
45
50
55
60


N = 200
height
weight
140
150
160
170
180
30
35
40
45
50
55
60


N = 350
height
weight
140
150
160
170
180
30
35
40
45
50
55
60

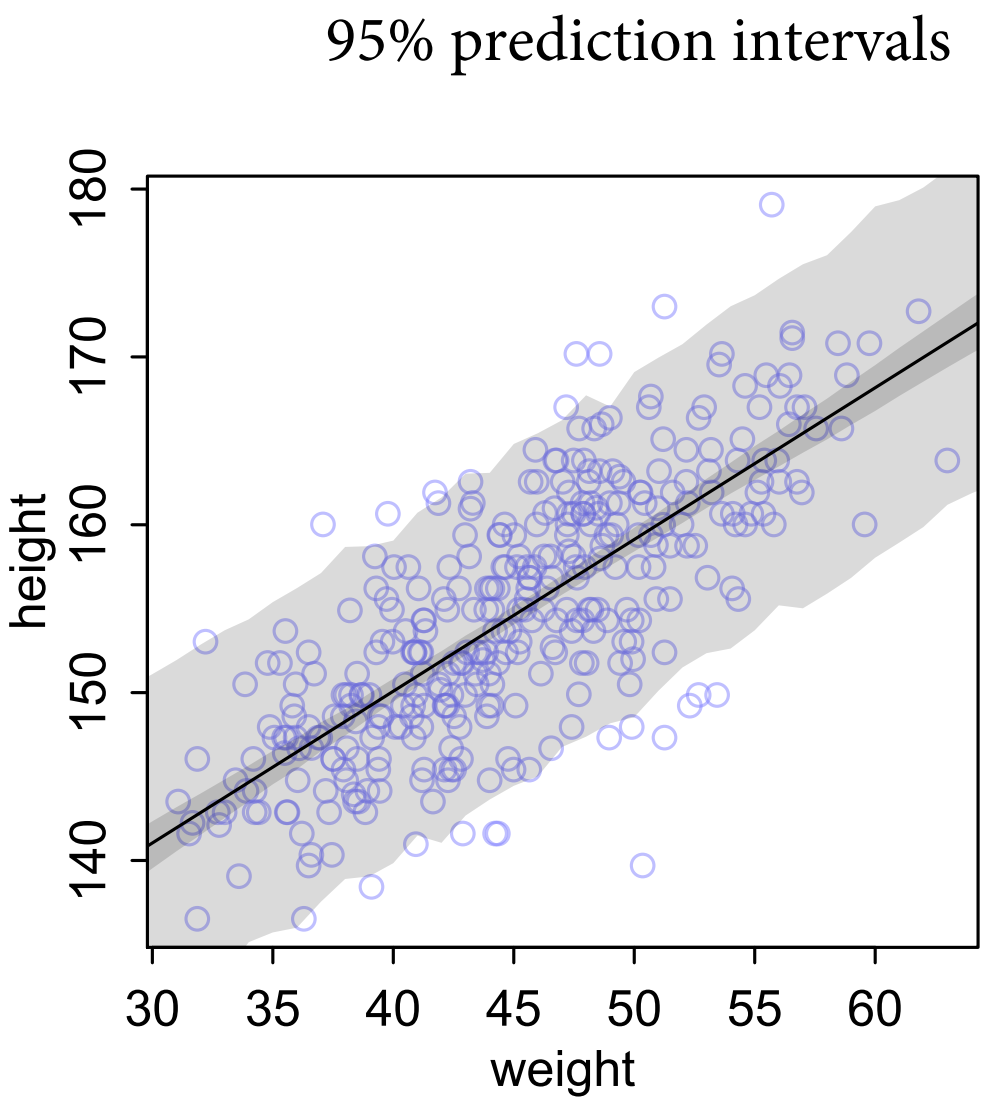
Prediction intervals, too

- What about predicted heights, not just mean height?
  - Uncertainty from posterior *and* uncertainty from likelihood => distribution of predicted height
- Could use `rnorm` to simulate sampling — like your Chapter 3 homework
- Can automate with `sim`

R code 4.58

```
sim.height <- sim( m4.3 , data=list(weight=weight.seq) )
str(sim.height)
```

```
 num [1:1000, 1:46] 139 144 141 140 130 ...
```



Nothing special about 95%

Try 50%, 80%, 99%

Interested in *shape*,
not *boundaries*

Figure 4.9

How `sim` works

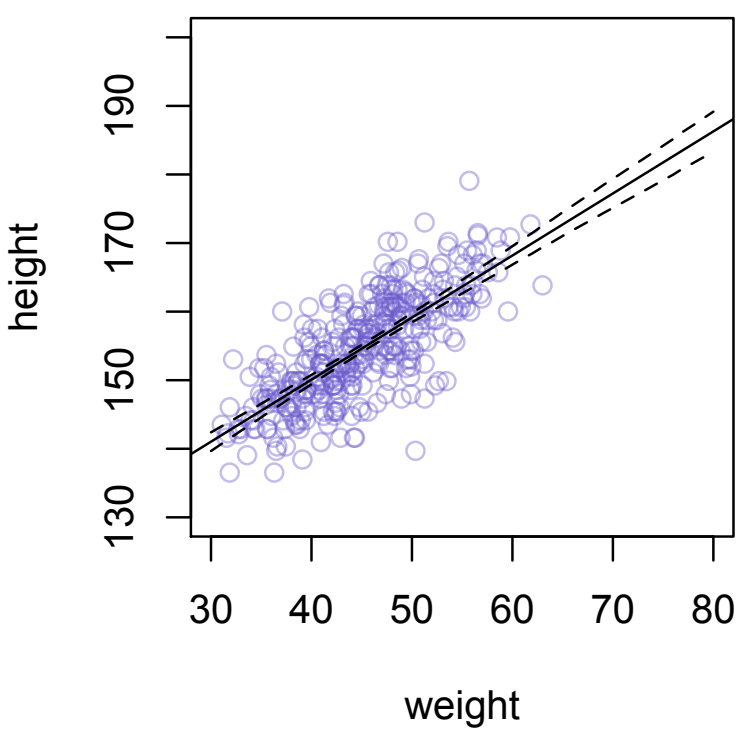
- For each weight
  - For each sample from posterior
    - Simulate a height: `rnorm(n,mu,sigma)`

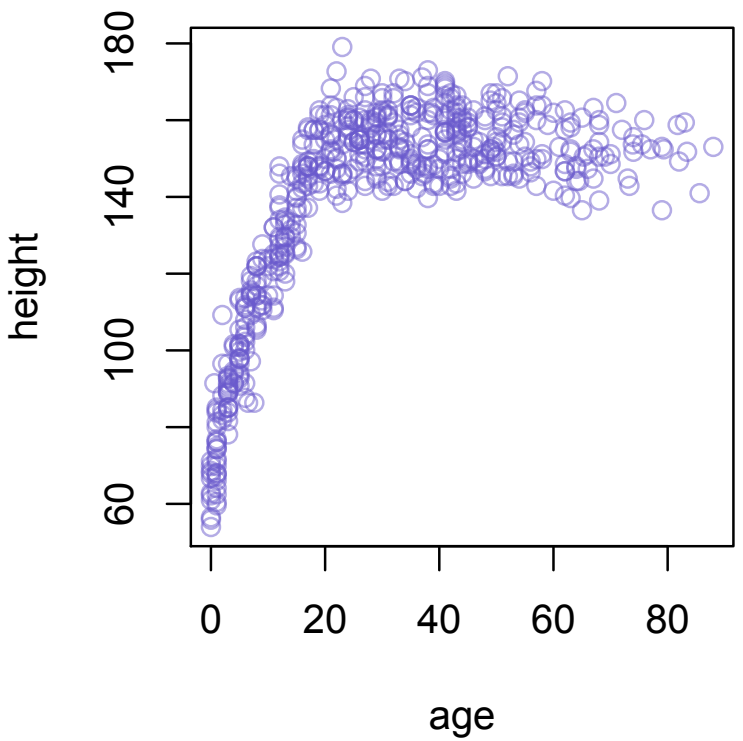
R code 4.62

```
post <- extract.samples(m4.3)
weight.seq <- 25:70
sim.height <- sapply( weight.seq , function(weight)
    rnorm(
        n=nrow(post) ,
        mean=post$a + post$b*weight ,
        sd=post$sigma ) )
height.PI <- apply( sim.height , 2 , PI )
```

Polynomial regression

- Linear trends can make absurd predictions
- Some relationships obviously not linear





Polynomial regression

- Purely descriptive (geocentric) strategy: use *polynomial* of predictor variable

$$\text{1st order (line):} \quad \mu_i = \alpha + \beta_1 x_i$$

$$\text{2nd order (parabola):} \quad \mu_i = \alpha + \beta_1 x_i + \beta_2 x_i^2$$

Polynomial models suck, but are very common

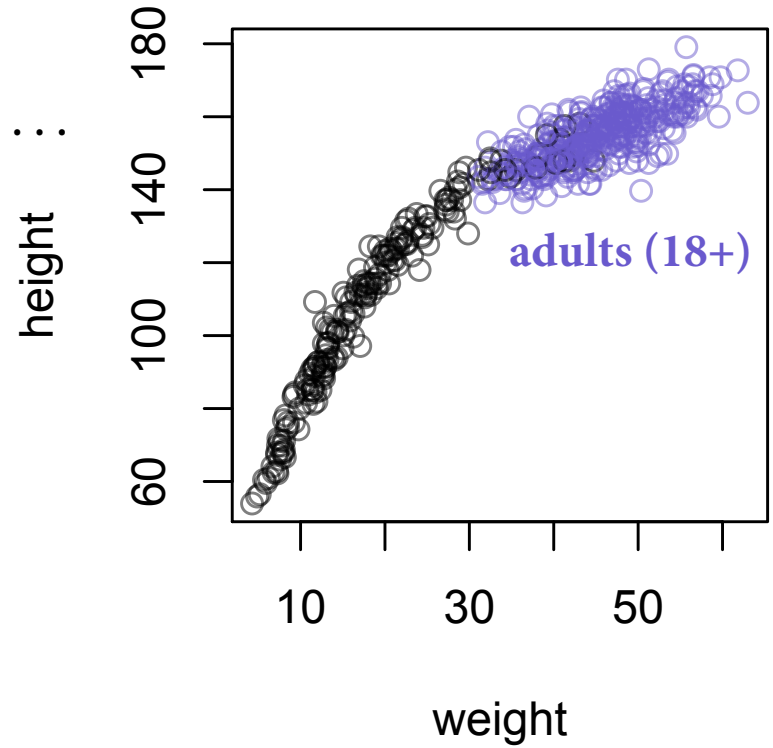
Polynomial regression

- We'll use full !Kung height/weight data

```
data(Howell1)
d <- Howell1
str(d)
```

R code 4.49

```
'data.frame':    544 obs. of  4 variables:
 $ height: num  152 140 137 157 145 ...
 $ weight: num  47.8 36.5 31.9 53 41.3 ...
 $ age   : num  63 63 65 41 51 35 32 27 19 54 ...
 $ male  : int  1 0 0 1 0 1 0 1 0 1 ...
```



Standardized predictors

- Very helpful to *standardize* predictor variables before fitting
  - Makes estimation easier
  - Makes interpretation maybe easier
- To standardize:
  - subtract mean
  - divide by standard deviation
  - result: mean of zero and standard deviation of 1

R code 4.65

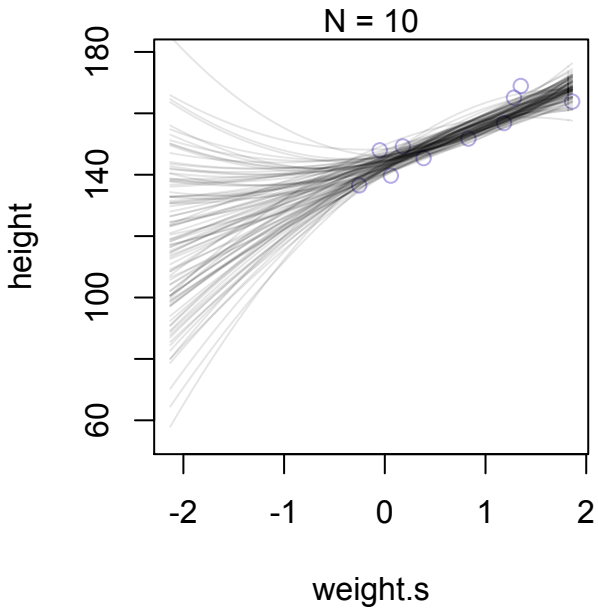
```
d$weight.s <- ( d$weight - mean(d$weight) )/sd(d$weight)
```

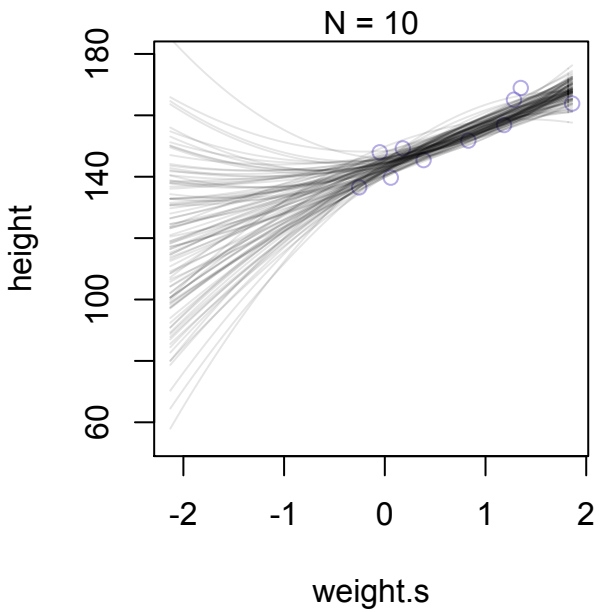
Parabolic regression

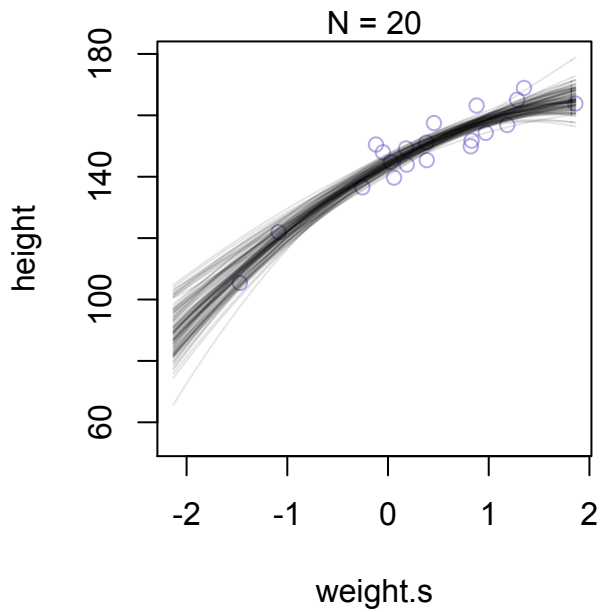
- Parabolic model of height as function of weight:

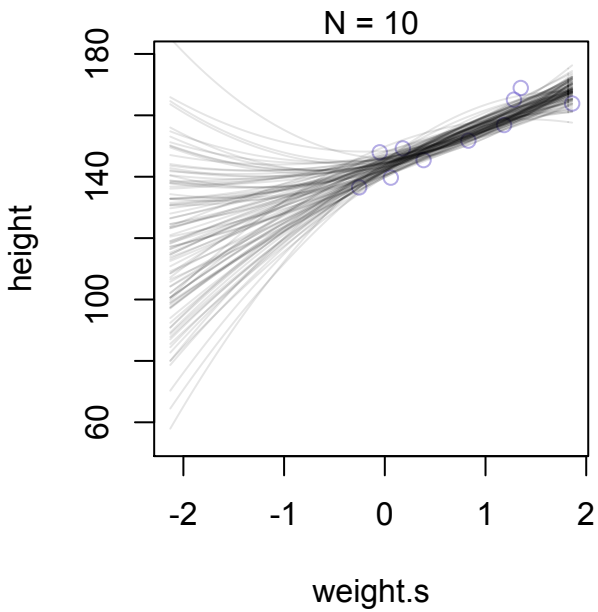
$$h_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta_1 x_i + \beta_2 x_i^2$$
$$\alpha \sim \text{Normal}(178, 100)$$
$$\beta_1 \sim \text{Normal}(0, 10)$$
$$\beta_2 \sim \text{Normal}(0, 10)$$
$$\sigma \sim \text{Uniform}(0, 50)$$

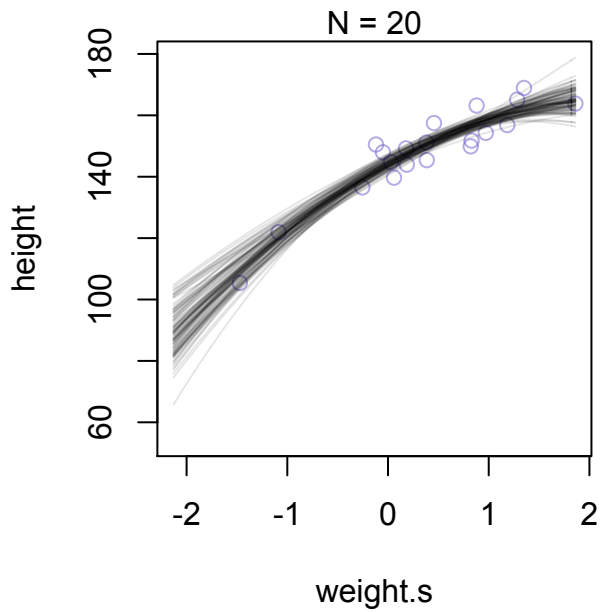
```
height ~ dnorm(mu,sigma)
mu <- a + b1*weight.s + b2*weight.s^2
a ~ dnorm(178,100)
b1 ~ dnorm(0,10)
b2 ~ dnorm(0,10)
sigma ~ dunif(0,50)
```

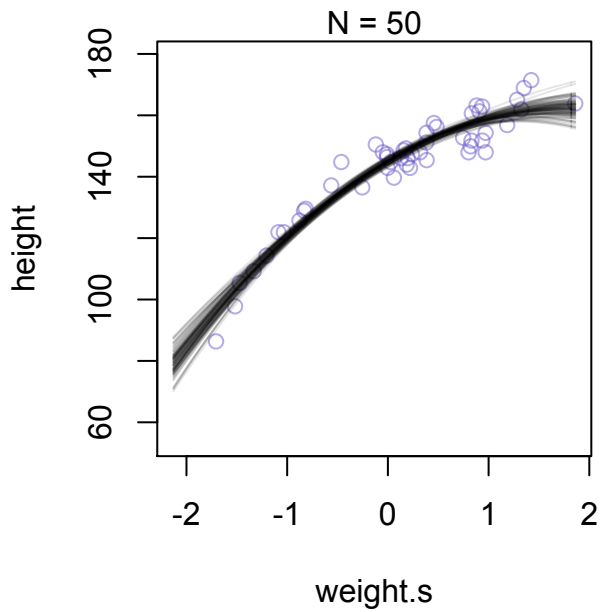

N = 10
height
180
140
100
60
-2
-1
0
1
2
weight.s

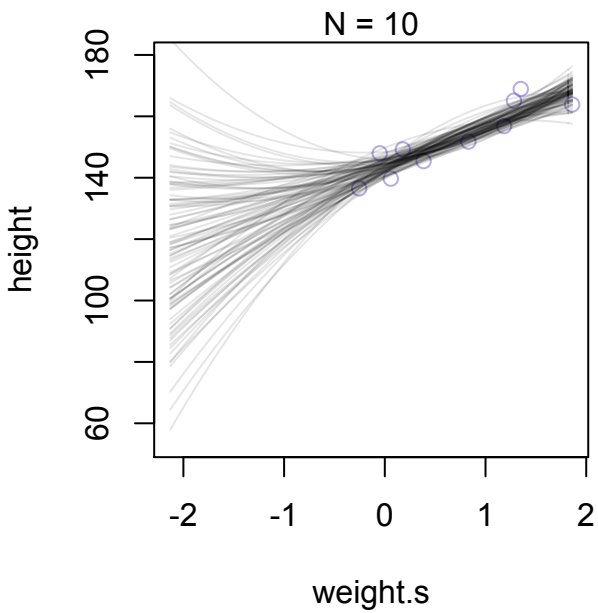

N = 10
height
180
140
100
60
-2
-1
0
1
2
weight.s

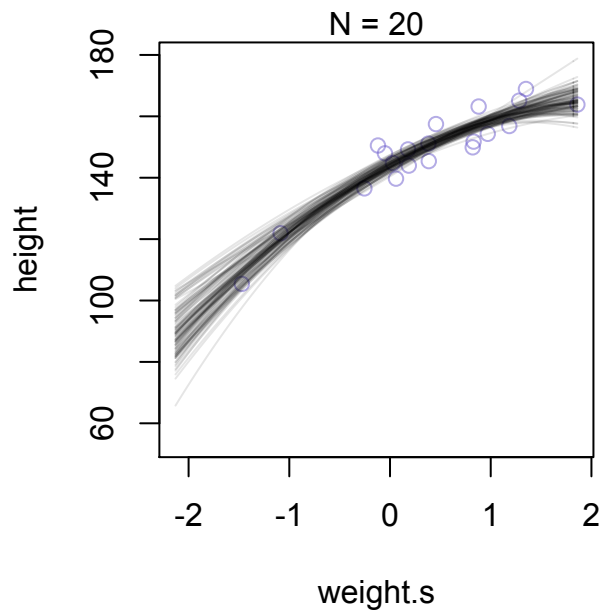

N = 20
height
180
140
100
60
-2
-1
0
1
2
weight.s

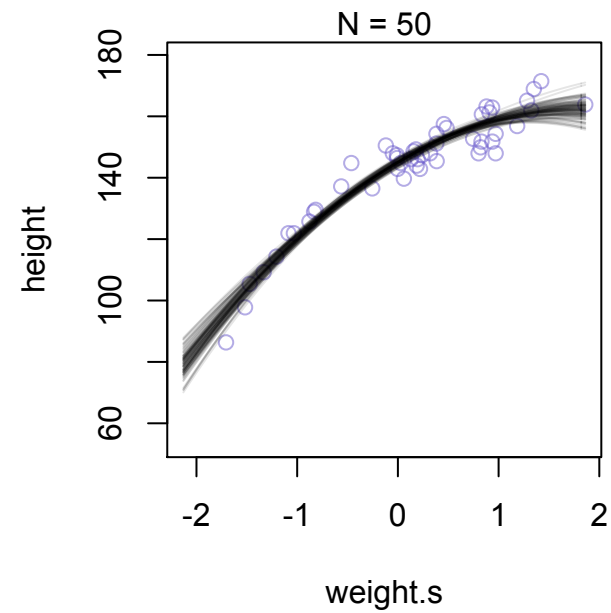

N = 10
height
180
140
100
60
-2
-1
0
1
2
weight.s

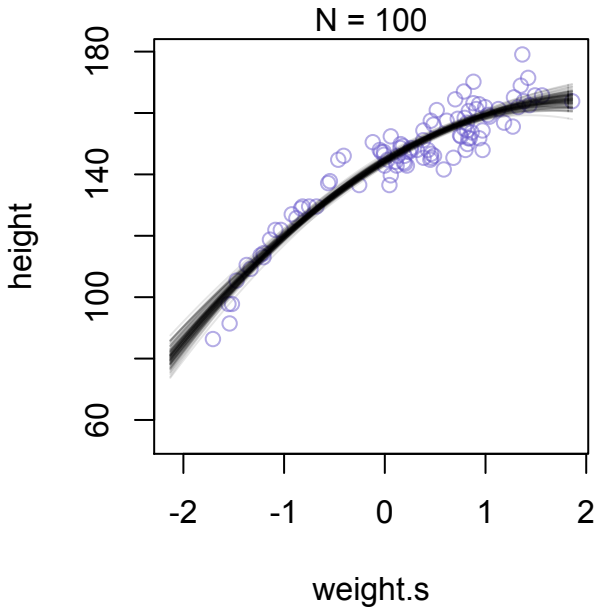

N = 20
height
180
140
100
60
-2
-1
0
1
2
weight.s

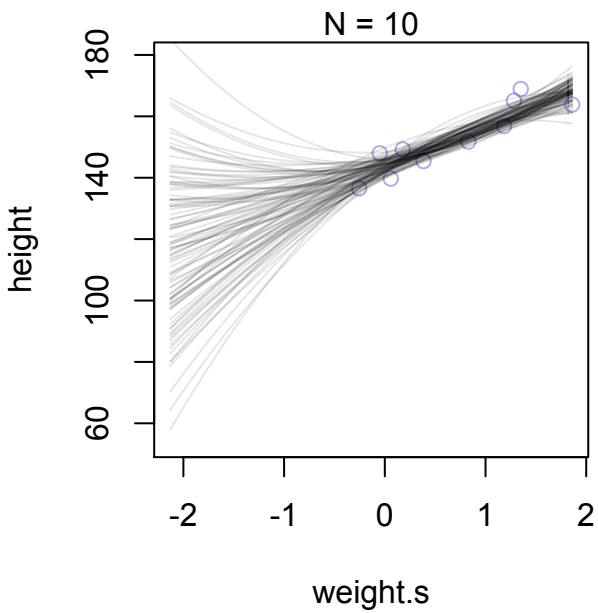

N = 50
height
180
140
100
60
-2
-1
0
1
2
weight.s

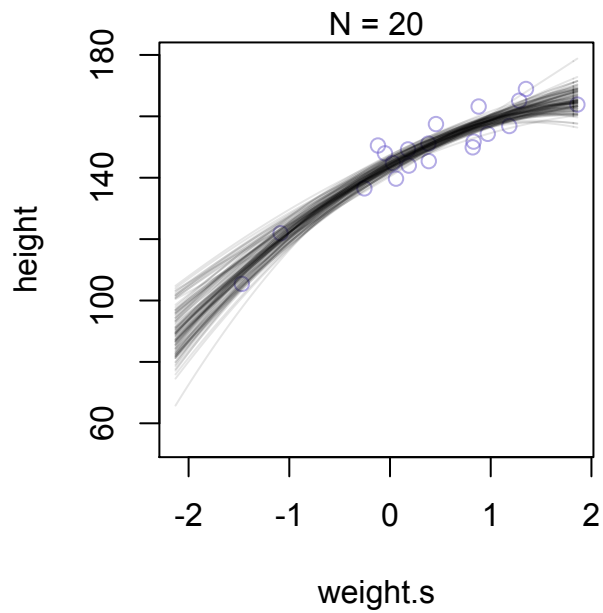

N = 10
height
weight.s
60
100
140
180
-2
-1
0
1
2

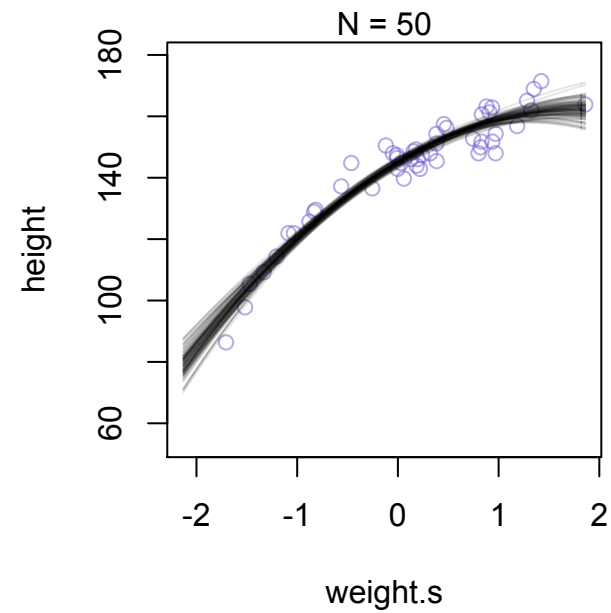

N = 20
height
weight.s
60
100
140
180
-2
-1
0
1
2

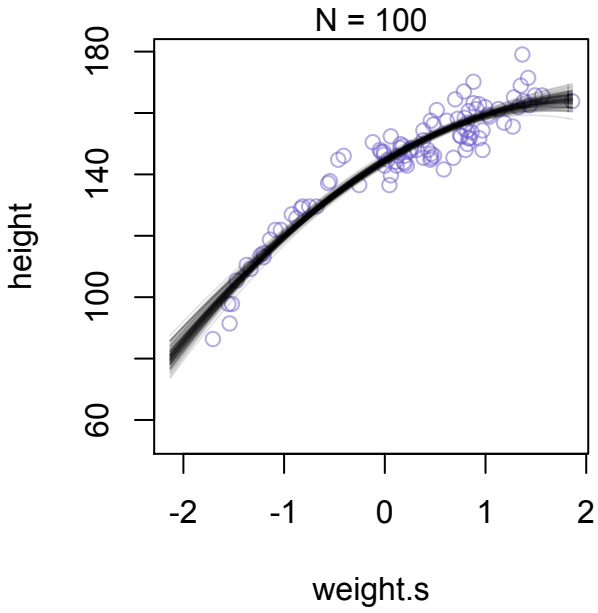

N = 50
height
weight.s
60
100
140
180
-2
-1
0
1
2

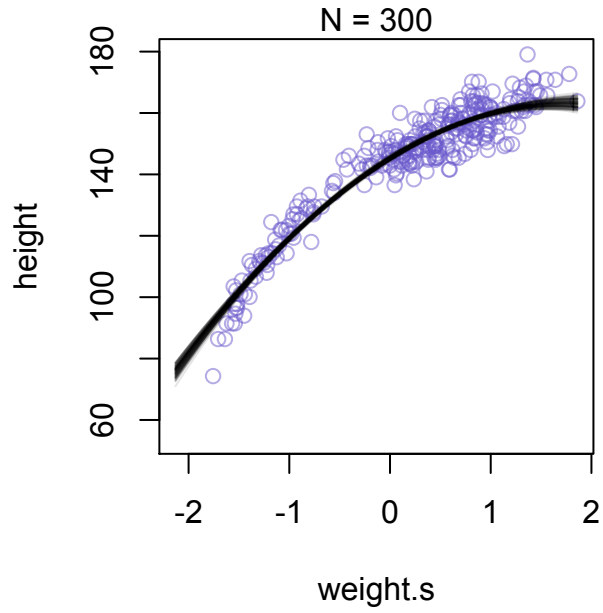

N = 100
height
weight.s
60
100
140
180
-2
-1
0
1
2

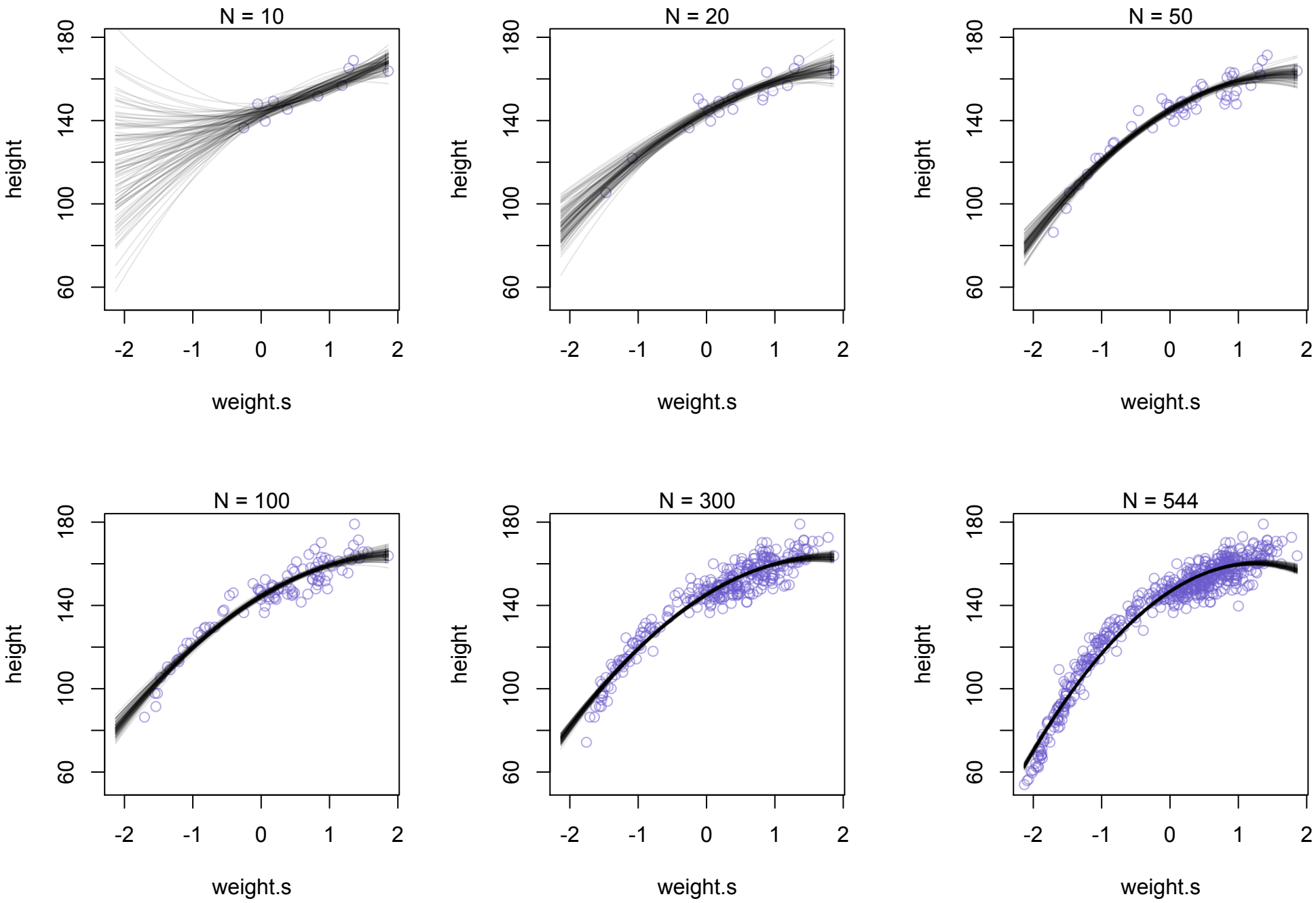

N = 10
height
weight.s
60
100
140
180
-2
-1
0
1
2


N = 20
height
weight.s
60
100
140
180
-2
-1
0
1
2


N = 50
height
weight.s
60
100
140
180
-2
-1
0
1
2


N = 100
height
weight.s
60
100
140
180
-2
-1
0
1
2


N = 300
height
weight.s
60
100
140
180
-2
-1
0
1
2


N = 10
N = 20
N = 50
N = 100
N = 300
N = 544
height
weight.s
60
100
140
180
-2
-1
0
1
2

Cubic model

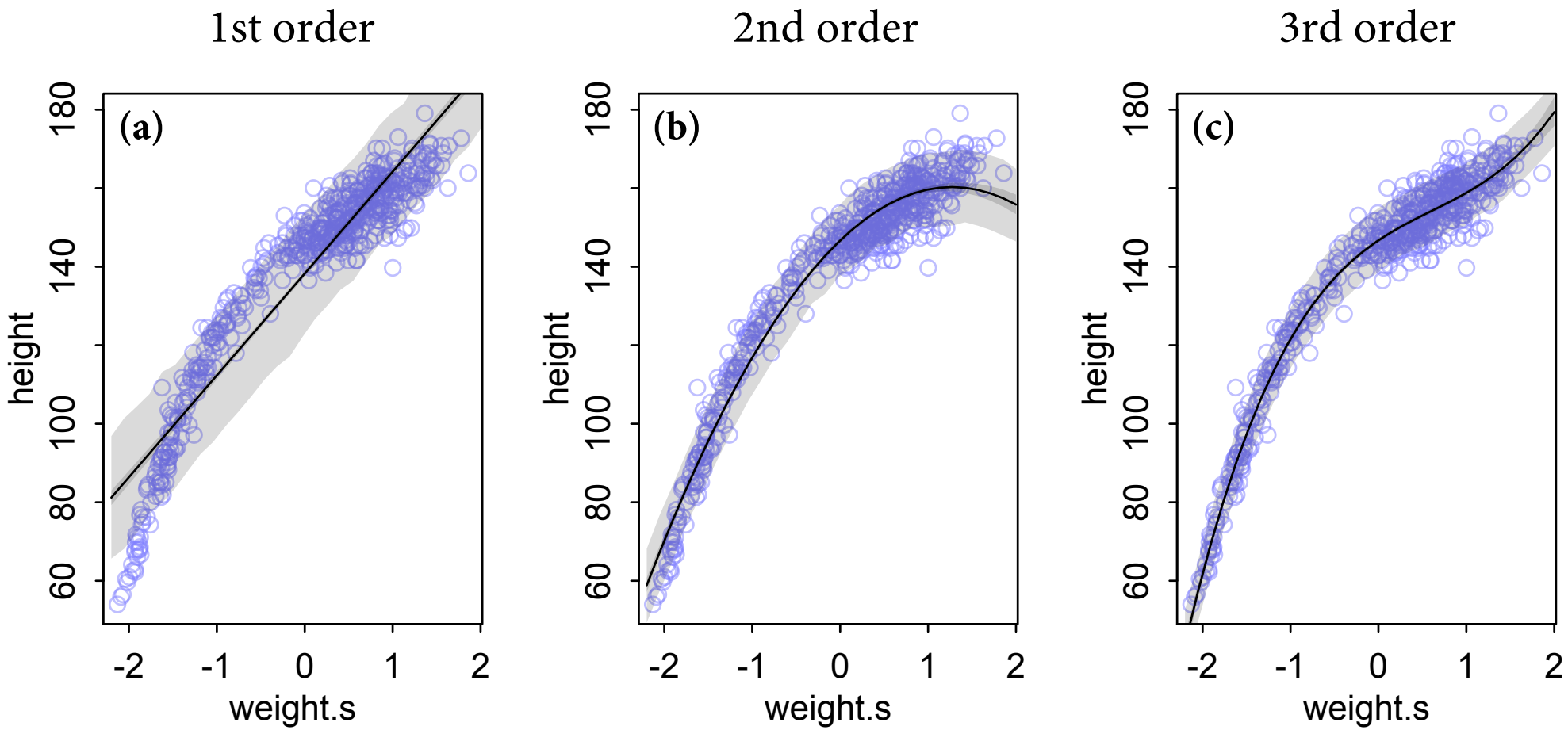
- Can go further down the rabbit hole:

$$h_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3$$

R code 4.70

```
d$weight.s3 <- d$weight.s^3
m4.6 <- map(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu <- a + b1*weight.s + b2*weight.s2 + b3*weight.s3 ,
        a ~ dnorm( 178 , 100 ) ,
        b1 ~ dnorm( 0 , 10 ) ,
        b2 ~ dnorm( 0 , 10 ) ,
        b3 ~ dnorm( 0 , 10 ) ,
        sigma ~ dunif( 0 , 50 )
    ) ,
    data=d )
```

Polynomial regression



Work

- Chapter 4 homework: 4H1, 4H2, 4H3

- Next week:
  - Multiple regression
  - Categorical data
  - Steady practice with these same tools
    - quadratic approximation
    - plotting implied predictions
    - model criticism