Statistical Rethinking
Winter 2019

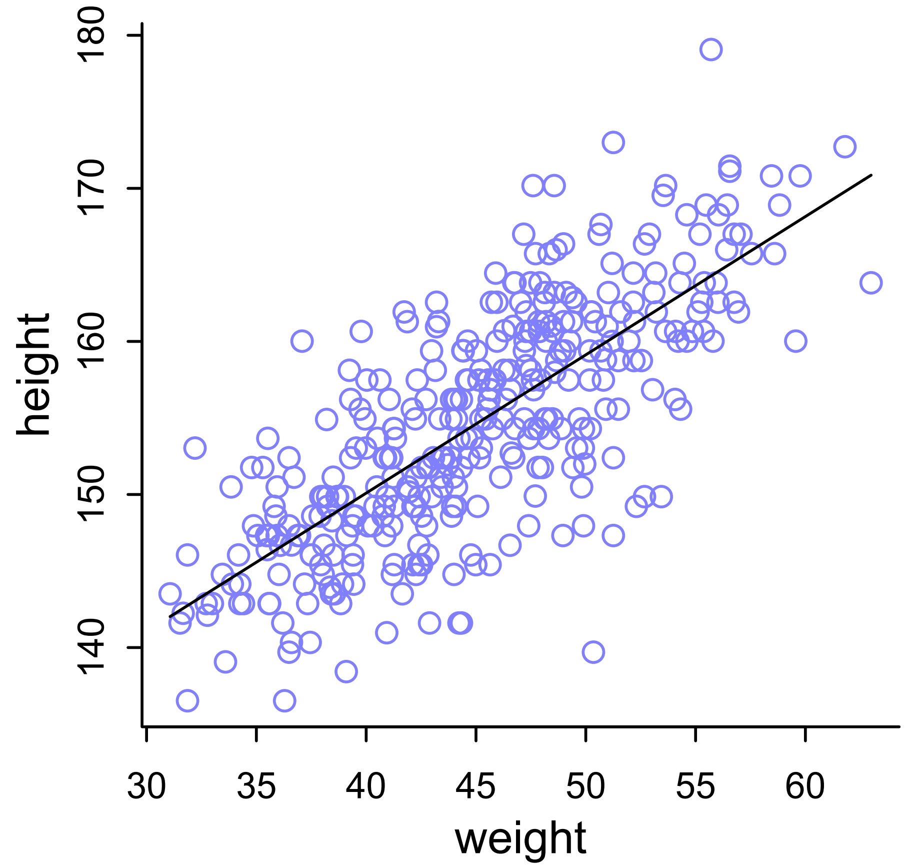
Lecture 04 / Week 2

Wiggly Orbits

R code 4.44

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

```
       mean   sd   5.5%  94.5%
a    154.60 0.27 154.17 155.03
b      0.90 0.04   0.84   0.97
sigma  5.07 0.19   4.77   5.38
```



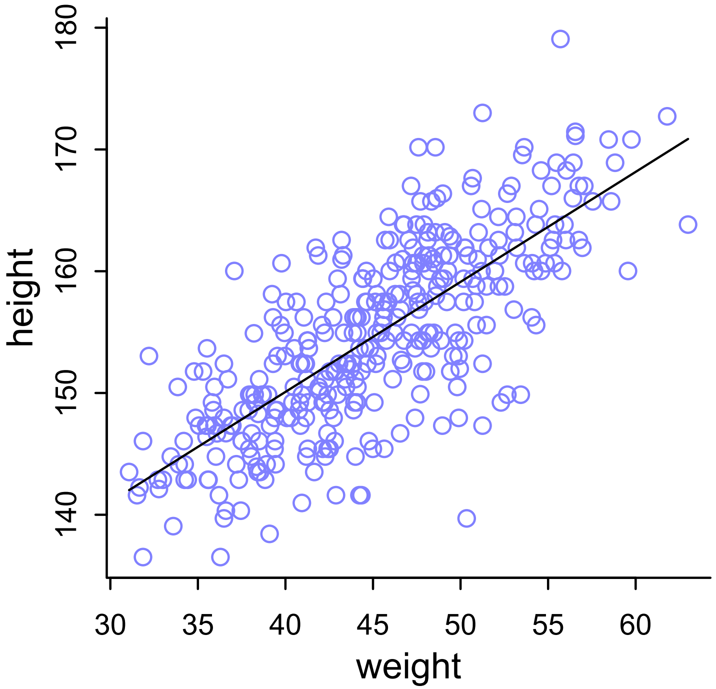
R code 4.46

```
plot( height ~ weight , data=d2 , col=rangi2 )
post <- extract.samples( m4.3 )
a_map <- mean(post$a)
b_map <- mean(post$b)
curve( a_map + b_map*(x - xbar) , add=TRUE )
```

Figure 4.6

Showing Uncertainty

- Want to get uncertainty onto that graph
- Again, sample from posterior
  1. Use mean 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

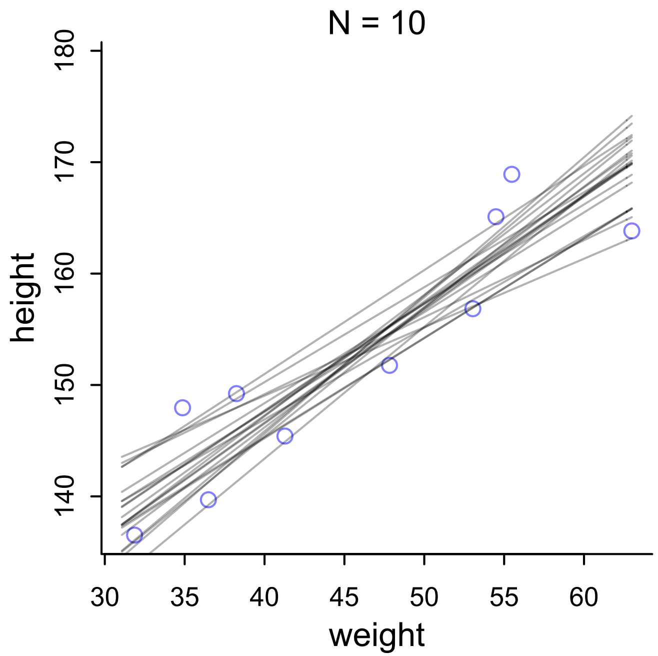
R code
4.47

```
post <- extract.samples( m4.3 )
post[1:5,]
```

```
         a         b    sigma
1 154.5505 0.9222372 5.188631
2 154.4965 0.9286227 5.278370
3 154.4794 0.9490329 4.937513
4 155.2289 0.9252048 4.869807
5 154.9545 0.8192535 5.063672
```

Each row is a line

Posterior is full of lines



R code
4.47

```
post <- extract.samples( m4.3 )
post[1:5,]
```

```
         a         b    sigma
1 154.5505 0.9222372 5.188631
2 154.4965 0.9286227 5.278370
3 154.4794 0.9490329 4.937513
4 155.2289 0.9252048 4.869807
5 154.9545 0.8192535 5.063672
```

Figure 4.7

Posterior is full of lines

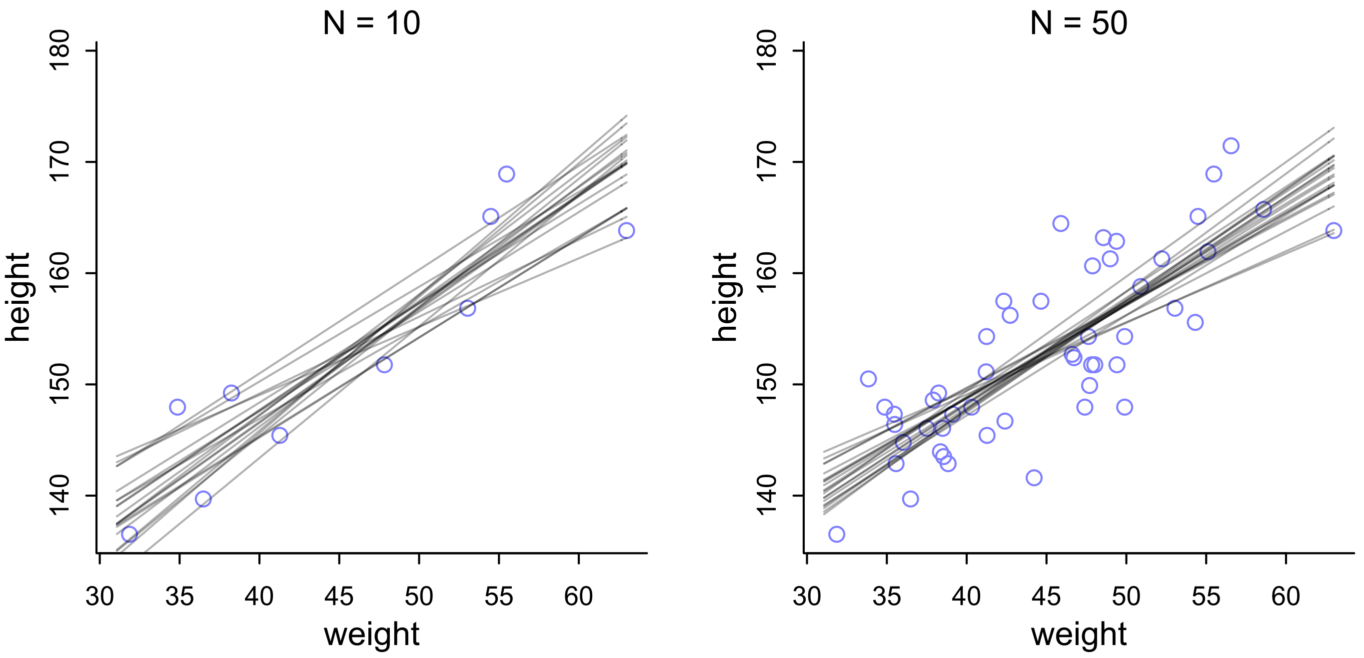


Figure 4.7

Posterior is full of lines

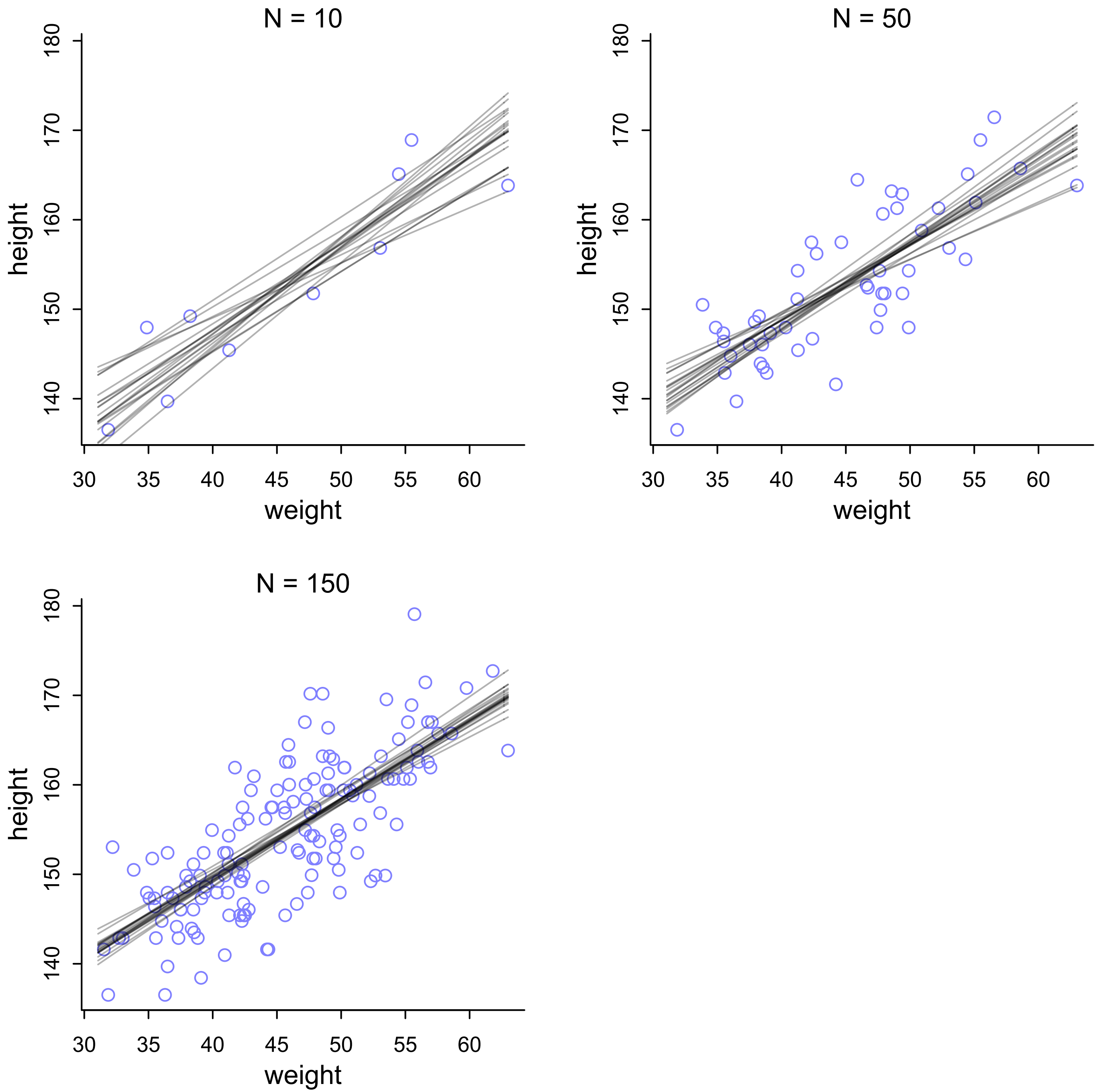


Figure 4.7

Posterior is full of lines

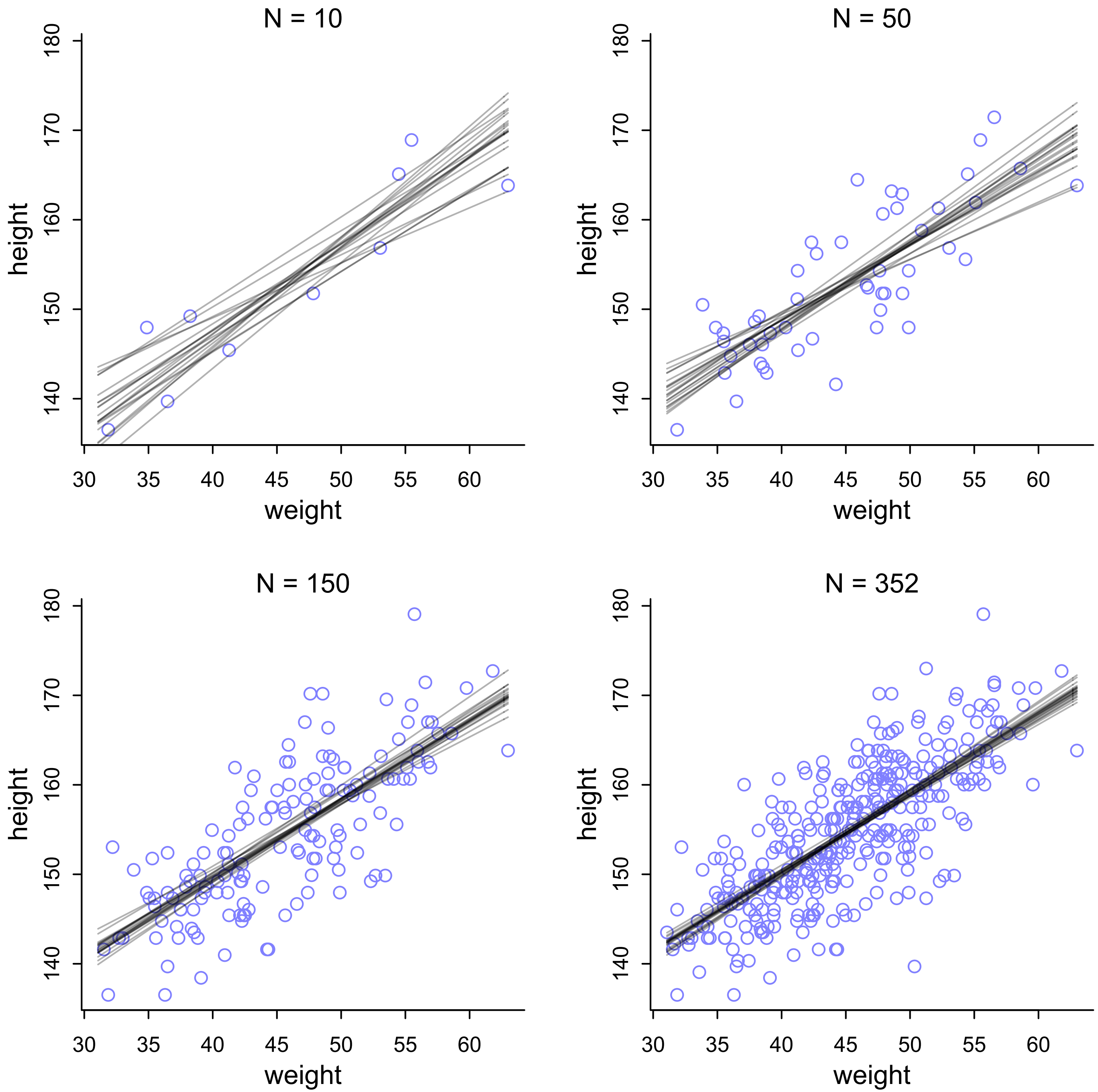


Figure 4.7

Predict mu

$$\mu_i = \alpha + \beta(x_i - \bar{x})$$

R code 4.50

```
post <- extract.samples( m4.3 )
mu_at_50 <- post$a + post$b * ( 50 - xbar )
```

Predict mu

$$\mu_i = \alpha + \beta(x_i - \bar{x})$$

R code 4.50

```
post <- extract.samples( m4.3 )
mu_at_50 <- post$a + post$b * ( 50 - xbar )
```

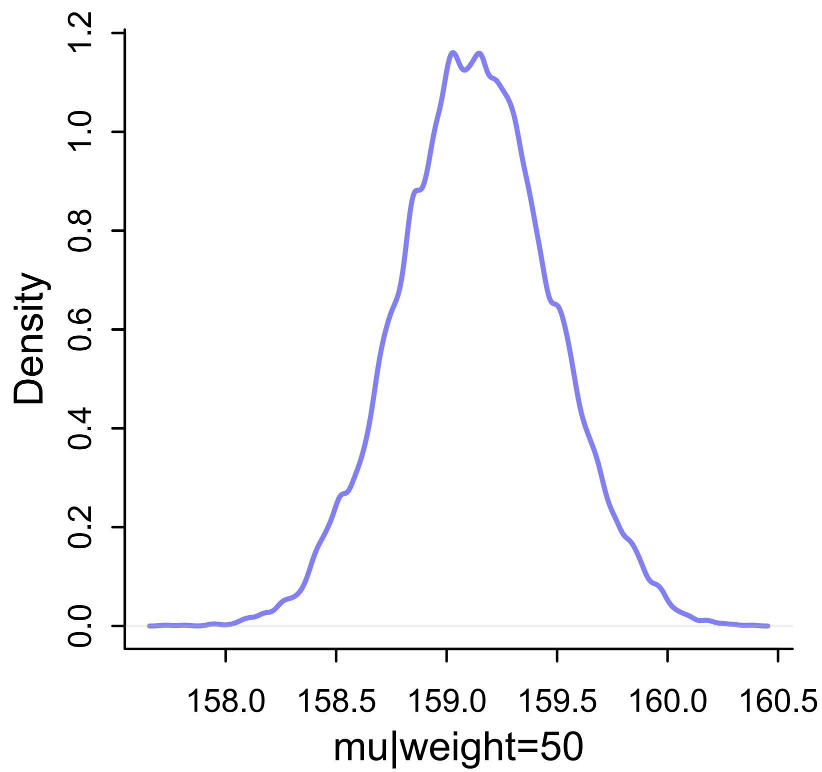


Figure 4.8

Predict every mu

We want a distribution for every value of x

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] 136 136 138 136 137 ...
```

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 – xbar)
- Summarize

R code
4.58

```
post <- extract.samples(m4.3)
mu.link <- function(weight) post$a + post$b*( weight - xbar )
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 , prob=0.89 )
```

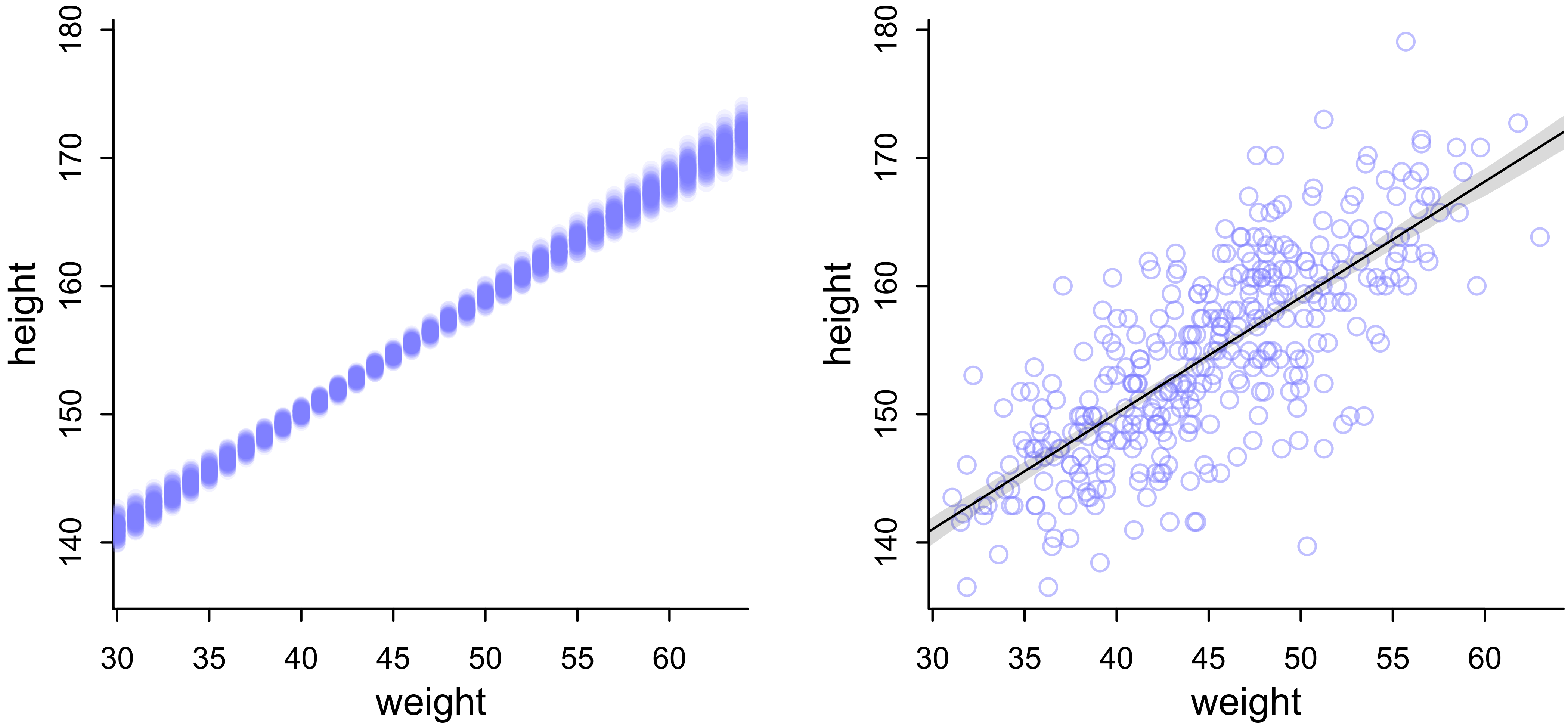
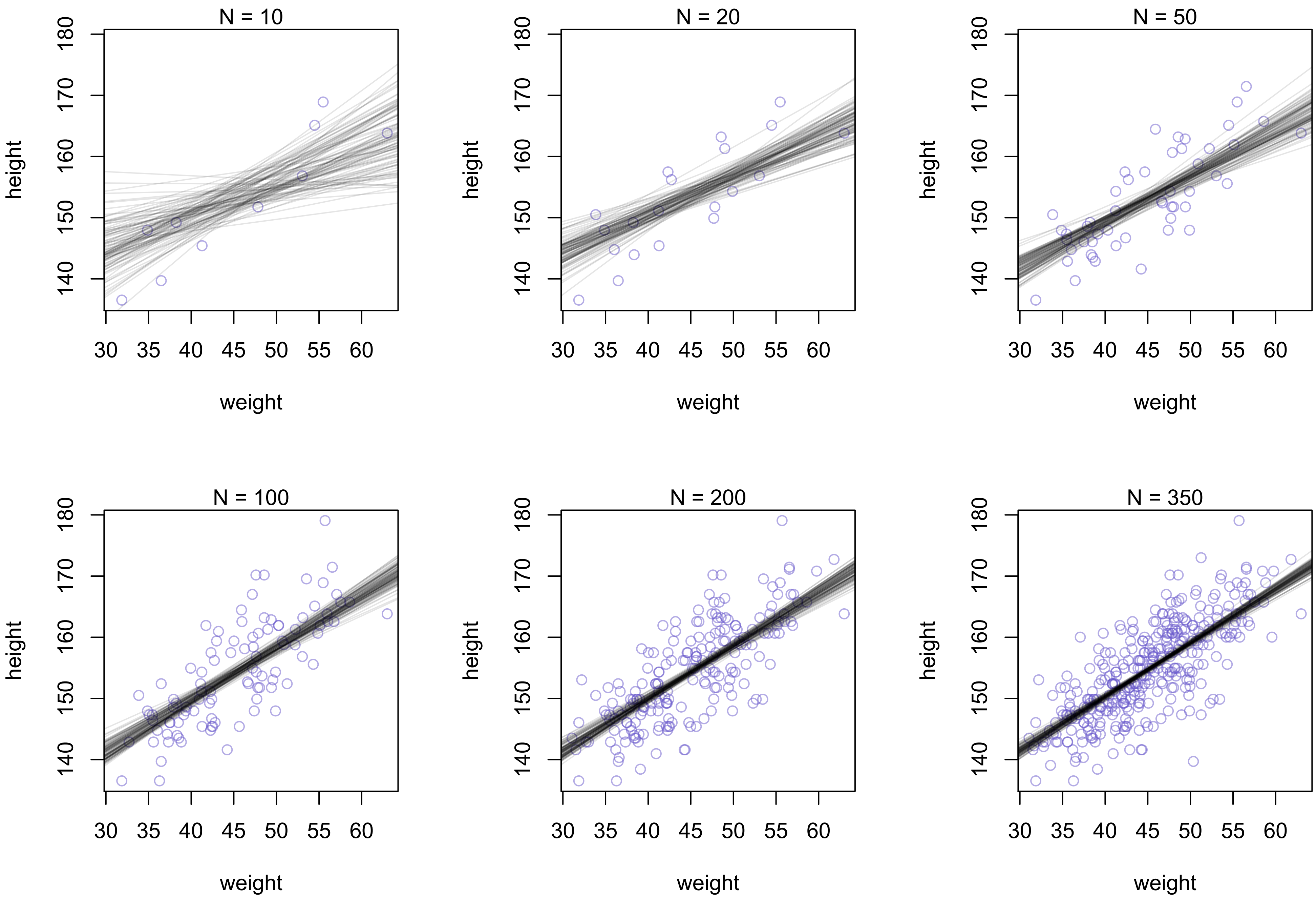
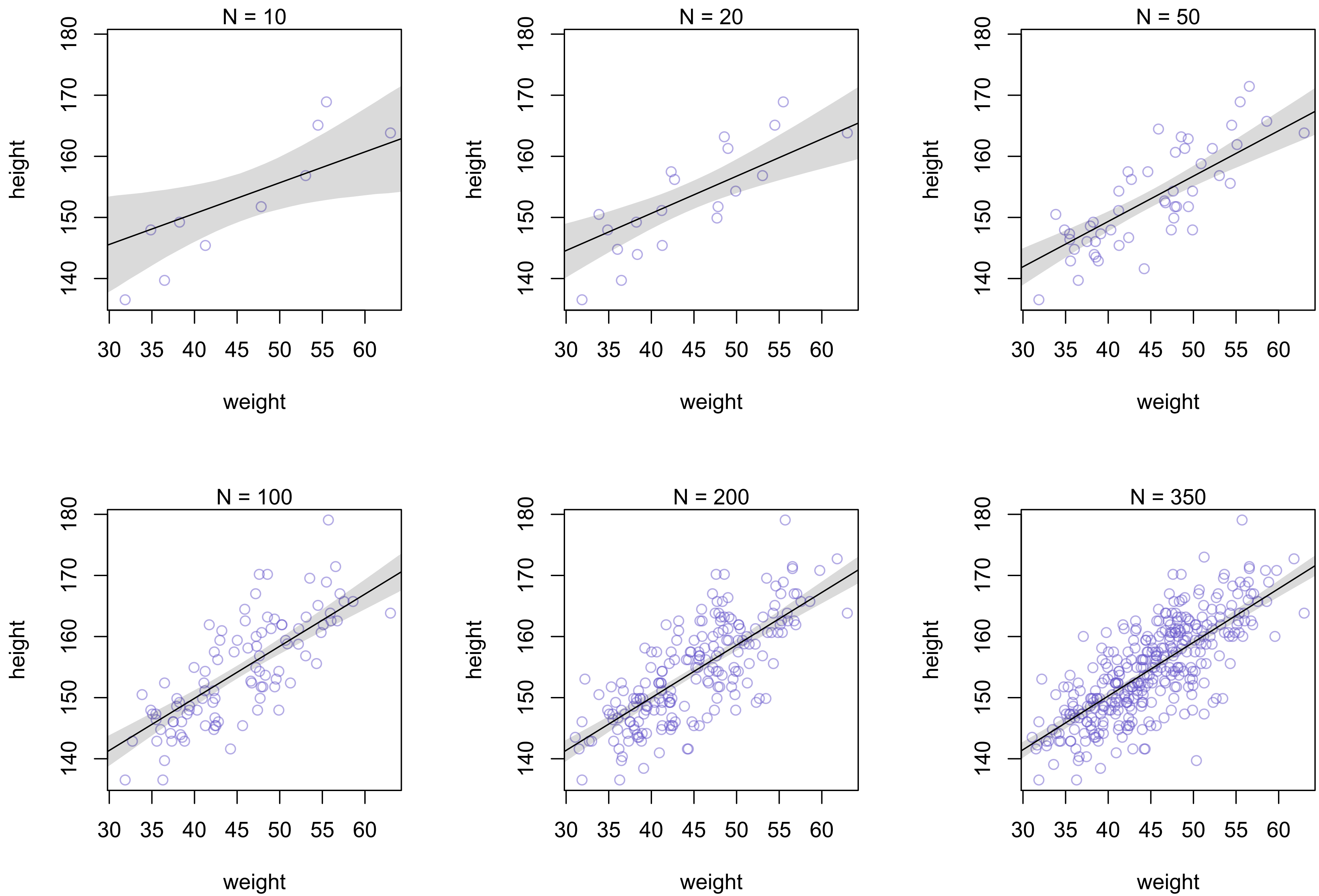
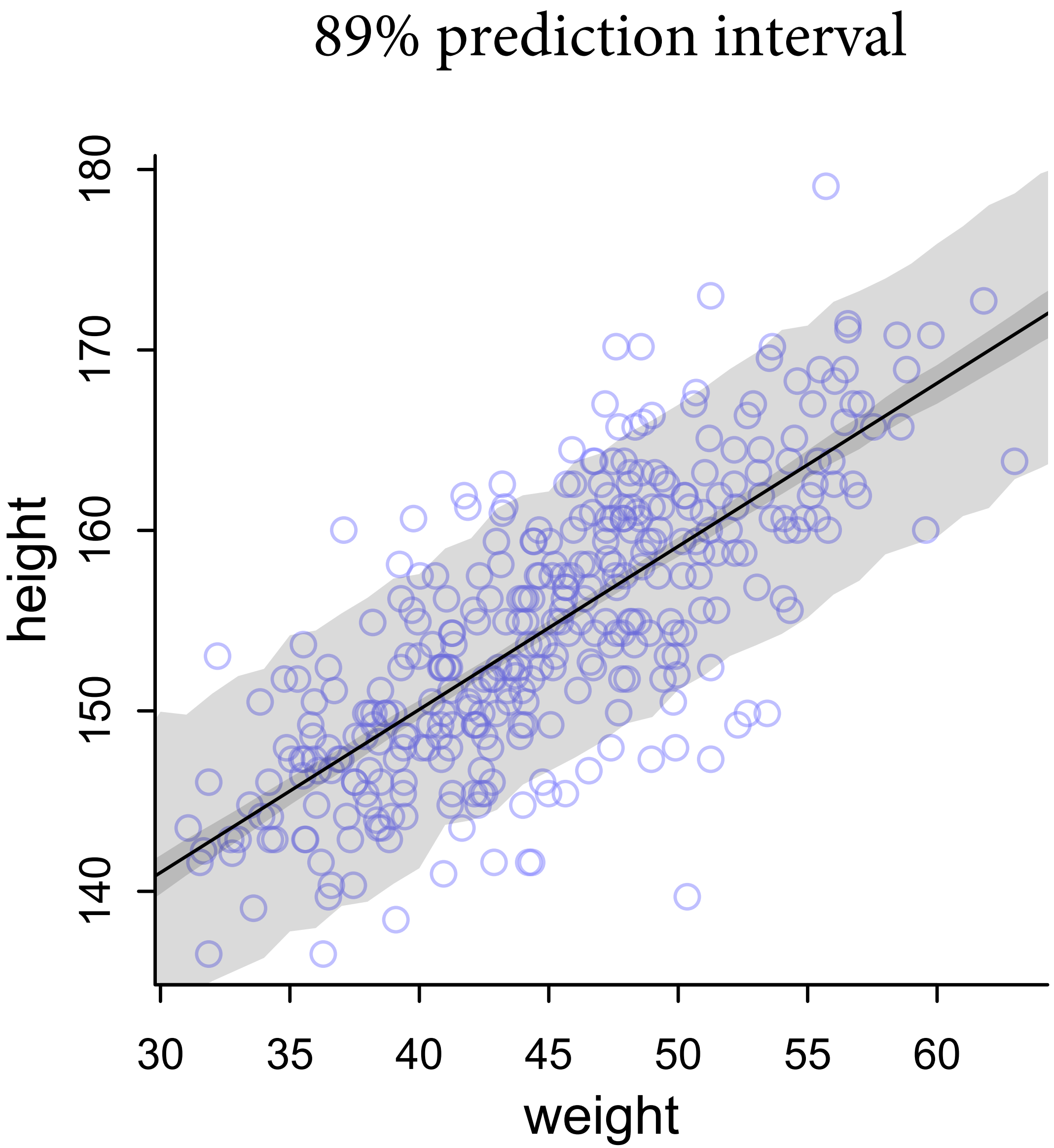

180
170
160
150
140
height
30
35
40
45
50
55
60
weight
180
170
160
150
140
height
30
35
40
45
50
55
60
weight

Figure 4.9


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


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



Nothing special about 95%

Try 50%, 80%, 99%

Interested in *shape*, not *boundaries*

Figure 4.10

Curves From Lines

- “Linear” models can make curves
- Polynomial regression
  - Common
  - Badly behaved
- Splines
  - Very flexible
  - Highly geocentric

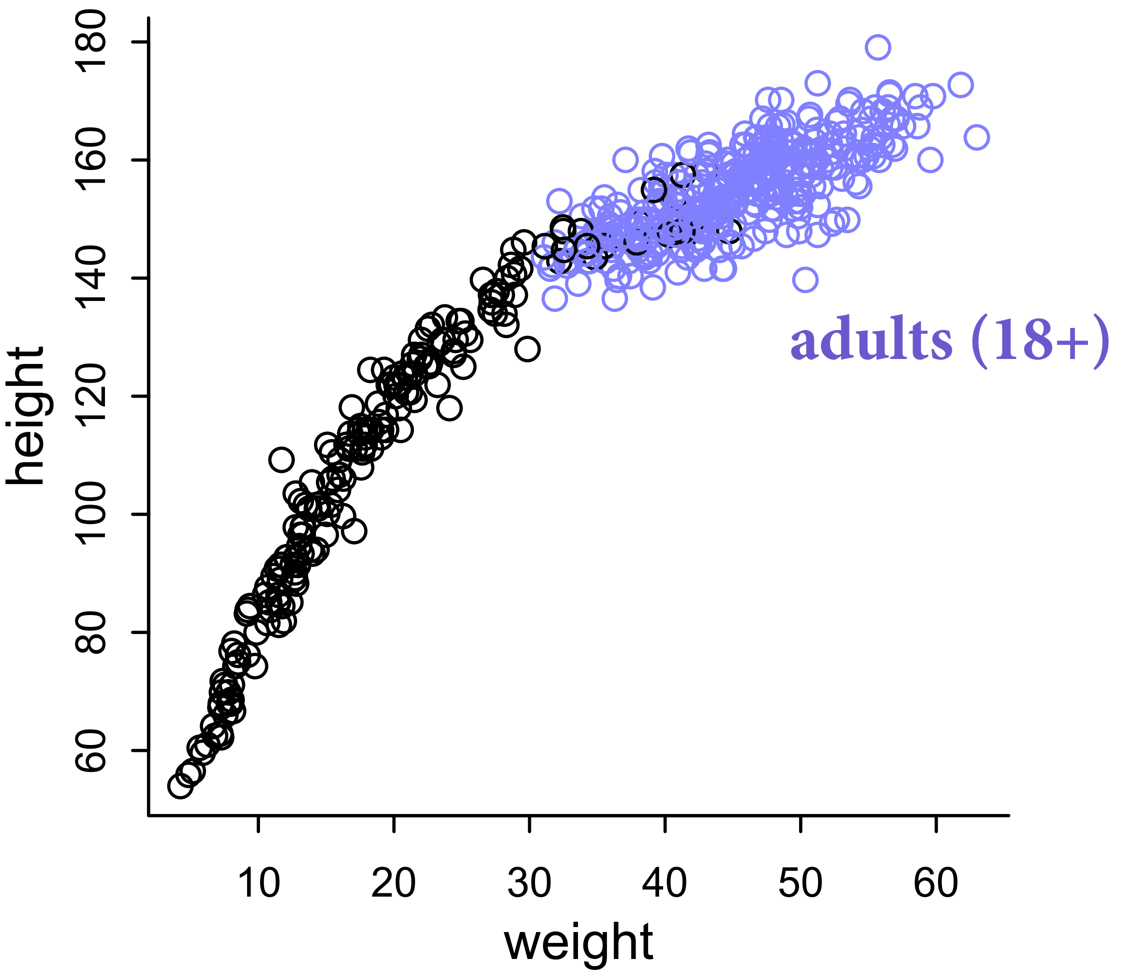
Polynomial regression

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

$$\begin{aligned}\text{1st order (line):} \quad & \mu_i = \alpha + \beta_1 x_i \\ \text{2nd order (parabola):} \quad & \mu_i = \alpha + \beta_1 x_i + \beta_2 x_i^2\end{aligned}$$

Polynomial regression

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



Parabolic model of height

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 \\
\alpha &\sim \text{Normal}(178, 20) \\
\beta_1 &\sim \text{Log-Normal}(0, 1) \\
\beta_2 &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

Standardized predictors

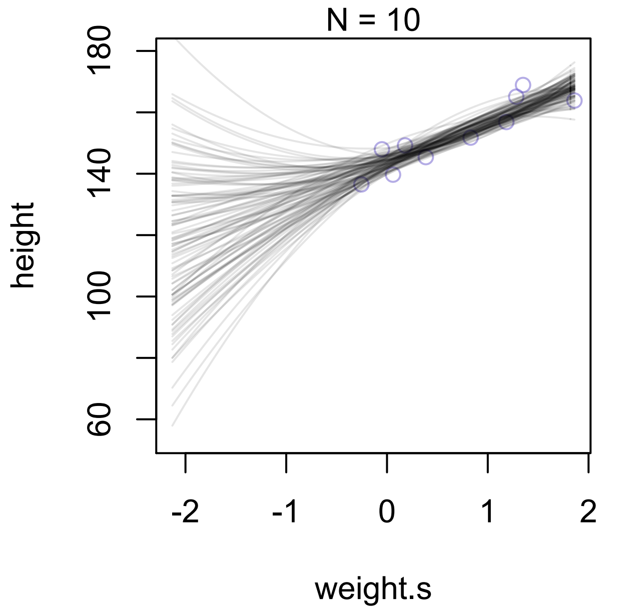
- Very helpful to *standardize* predictor variables before fitting
  - Makes estimation easier
  - Helps interpretation (sometimes)
- To standardize:
  - subtract mean
  - divide by standard deviation
  - result: mean of zero and standard deviation of 1

Parabolic model of height

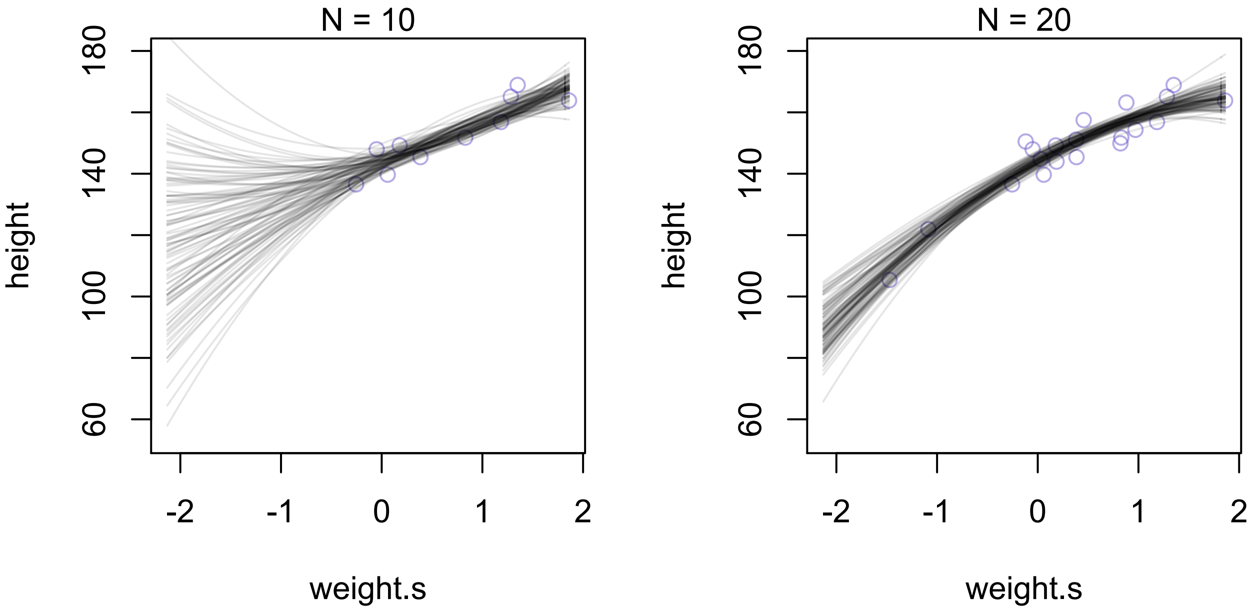
R code 4.65

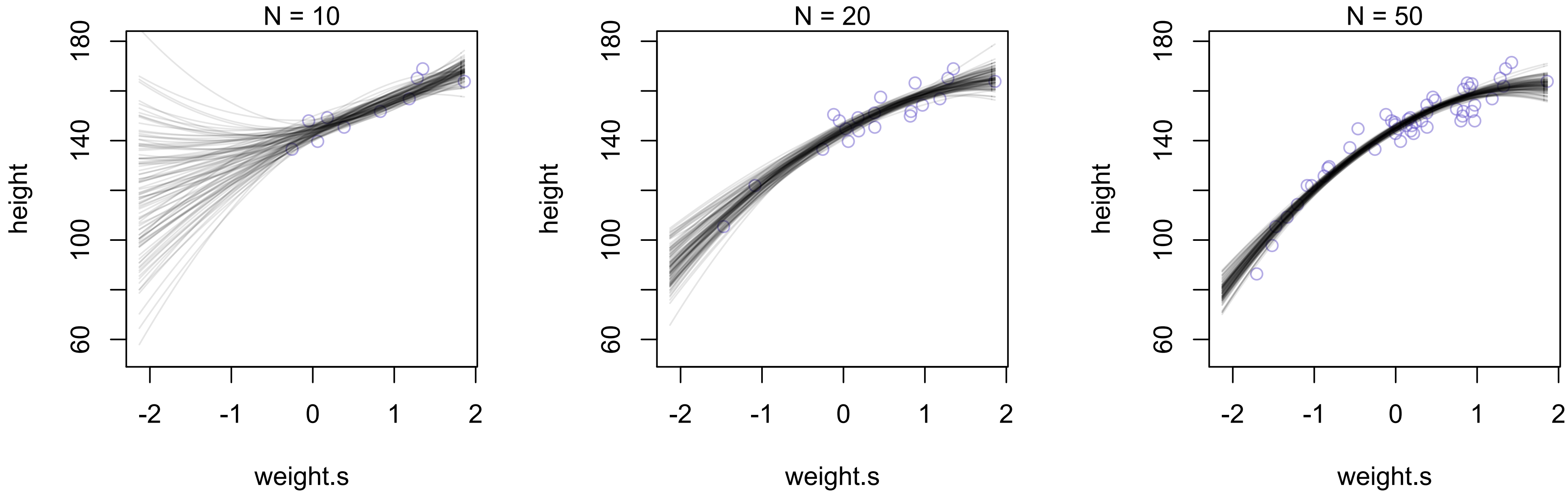
```
d$weight_s <- ( d$weight - mean(d$weight) )/sd(d$weight)
d$weight_s2 <- d$weight_s^2
m4.5 <- quap(
    alist(
        height ~ dnorm( mu , sigma ) ,
        mu <- a + b1*weight_s + b2*weight_s2 ,
        a ~ dnorm( 178 , 20 ) ,
        b1 ~ dlnorm( 0 , 1 ) ,
        b2 ~ dnorm( 0 , 1 ) ,
        sigma ~ dunif( 0 , 50 )
    ) ,
    data=d )
```

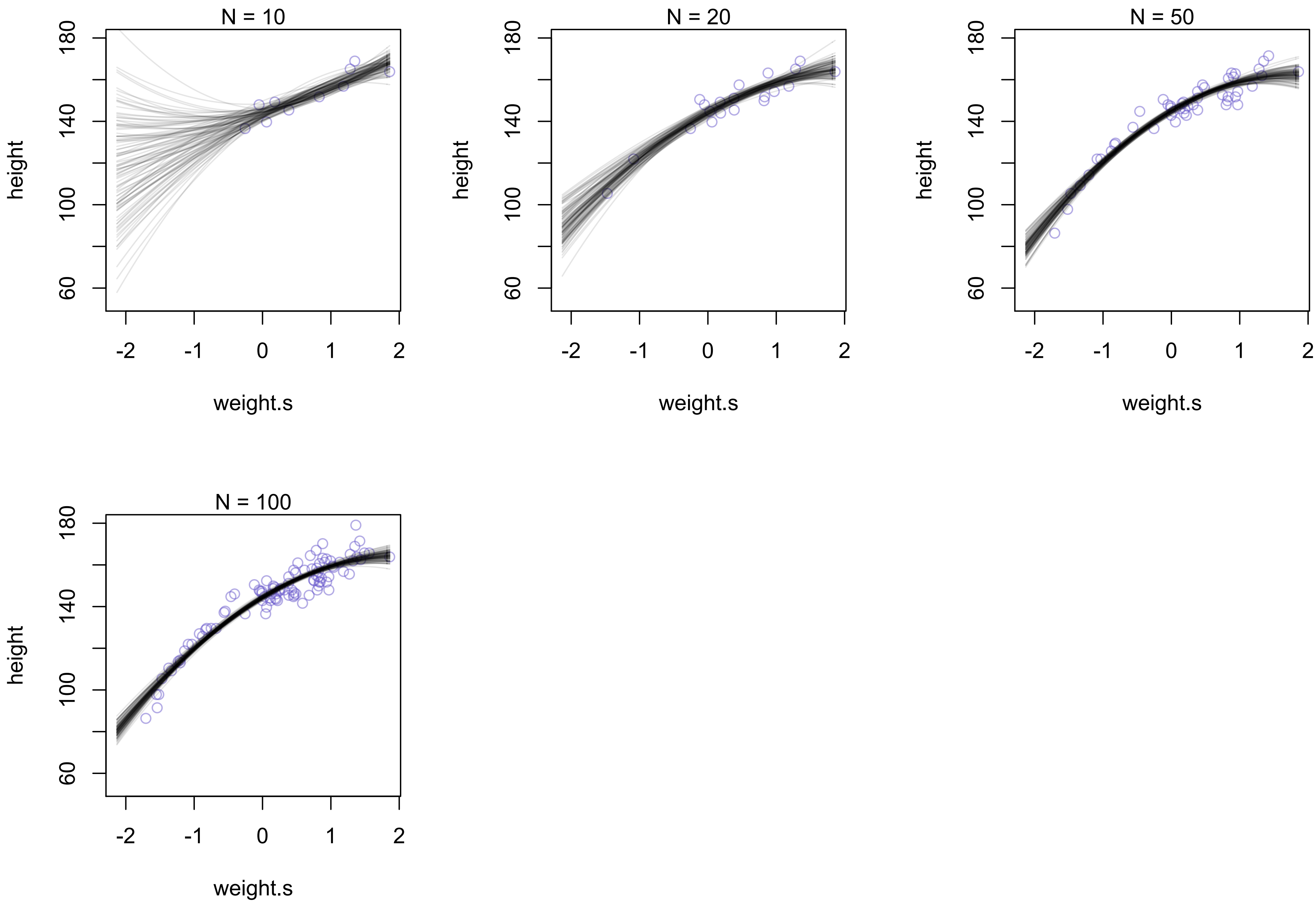
$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 \\
\alpha &\sim \text{Normal}(178, 20) \\
\beta_1 &\sim \text{Log-Normal}(0, 1) \\
\beta_2 &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

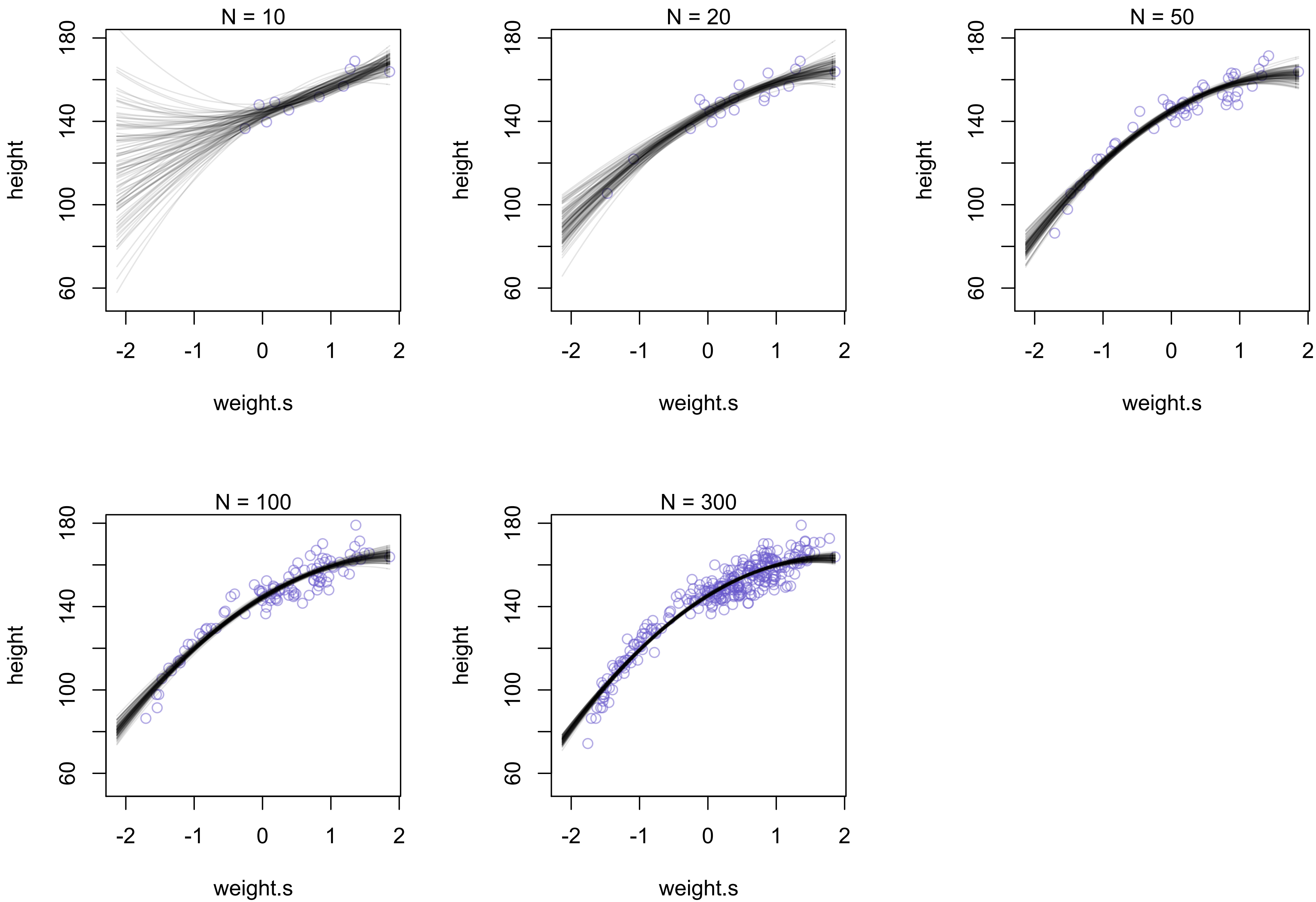


N = 10
height
weight.s
180
140
100
60
-2
-1
0
1
2

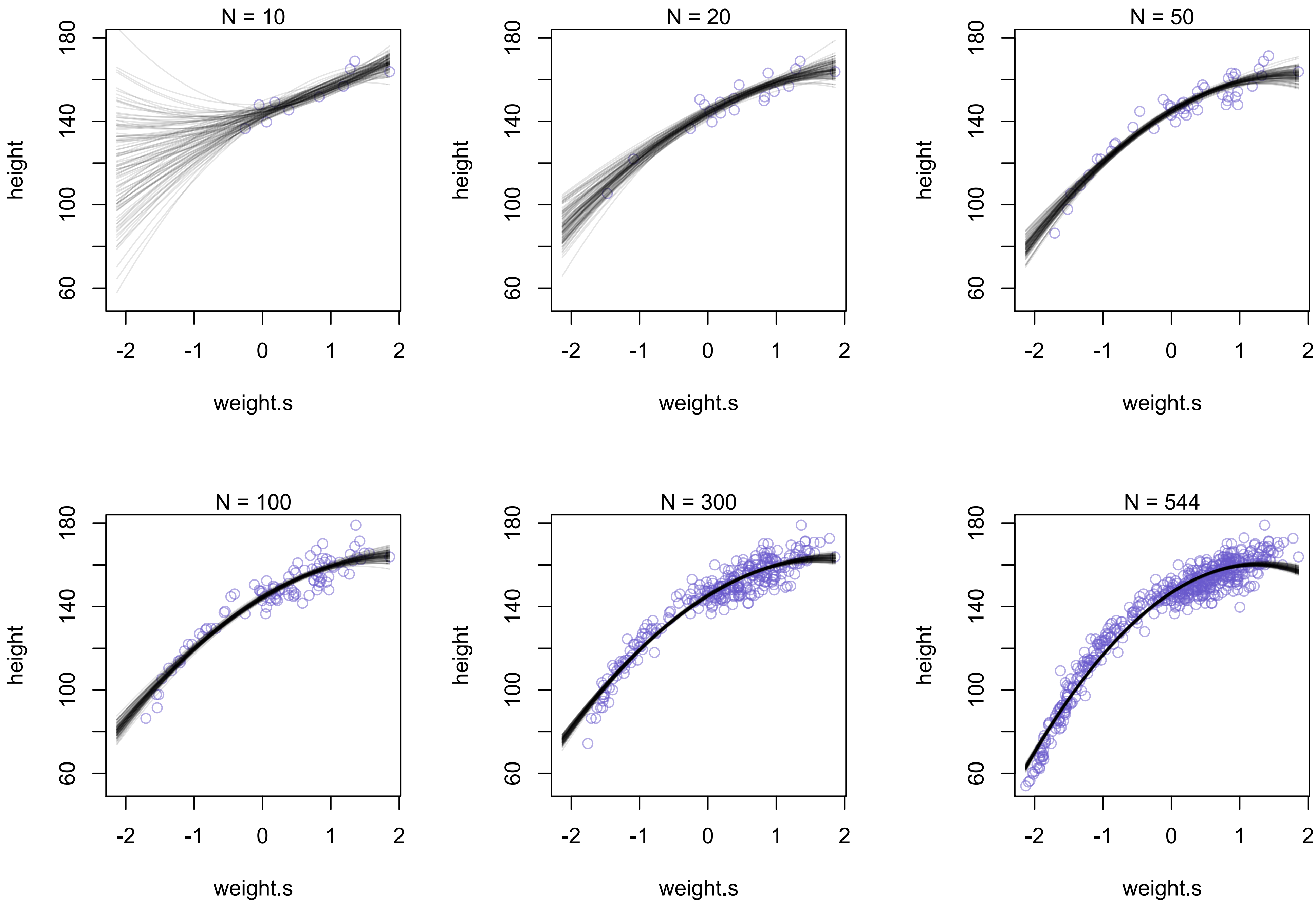

N = 10
N = 20
height
weight.s
60
100
140
180
-2
-1
0
1
2


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


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



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


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

Cubic model

- Can go further down the rabbit hole:

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

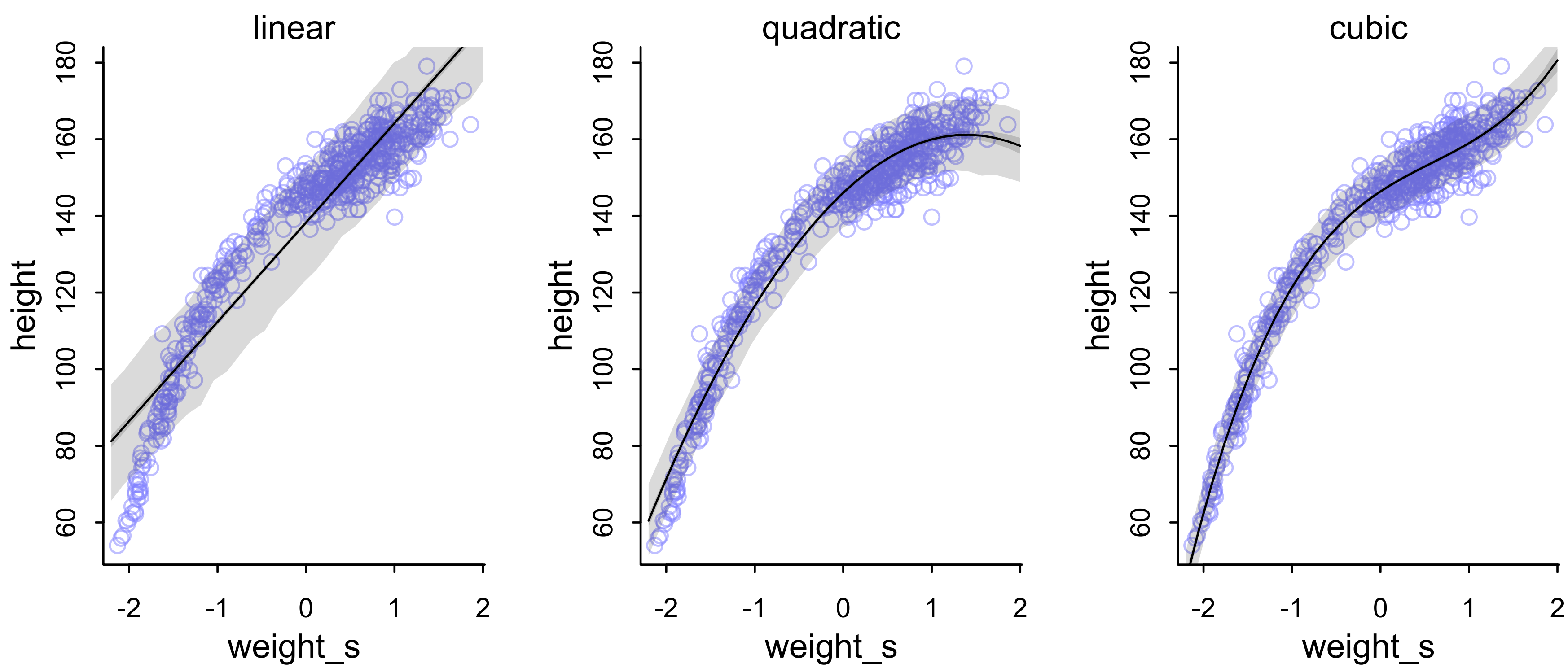
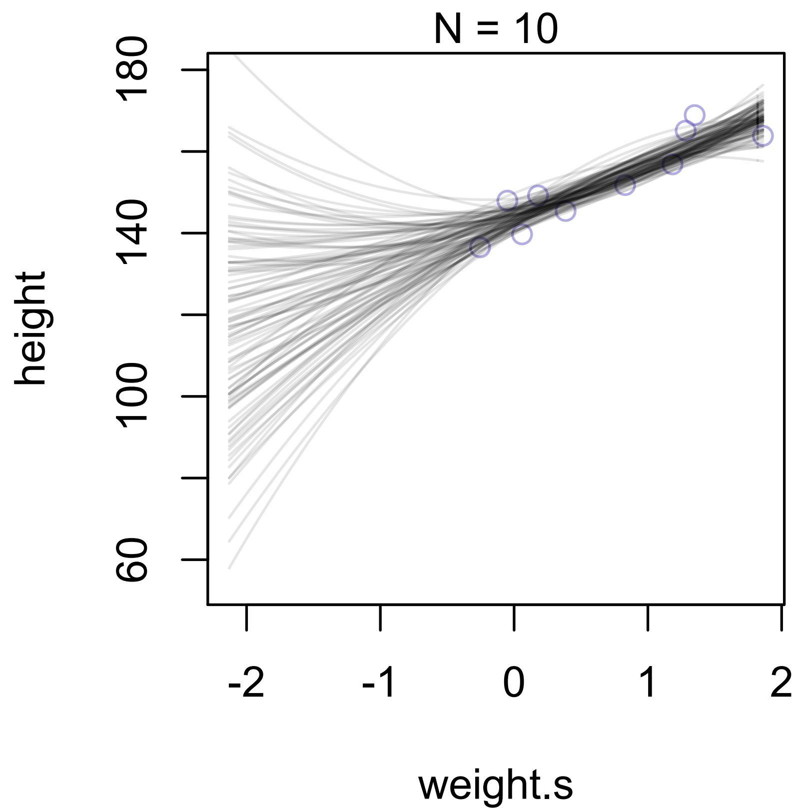
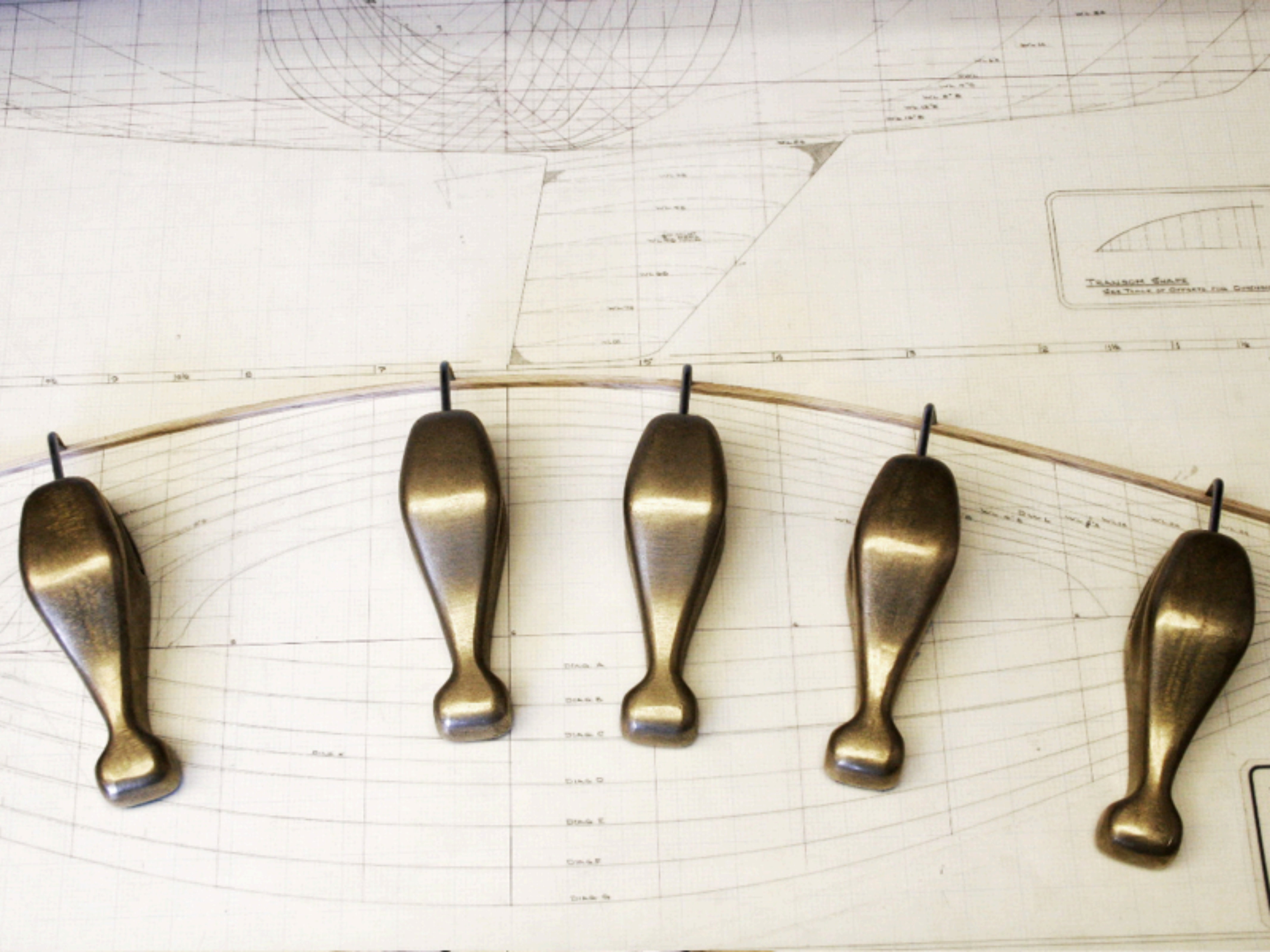


Figure 4.11

Polynomial grief

- Polynomials make absurd predictions outside range of data
- Parameters influence every part of curve, so hard to understand
- Not actually very flexible — can't have a monotonic curve!





Going Local — B-Splines

- Basis-Splines: Wiggly function built from many local, less wiggly functions
- Basis function: A local function
- Better than polynomials, but equally geocentric
- Bayesian B-splines often called P-splines.

think global act local

Going Local — B-Splines

- B-Splines are just linear models, but with some weird synthetic variables:

$$\mu_i = \alpha + w_1 B_{i,1} + w_2 B_{i,2} + w_3 B_{i,3} + \ldots$$

- Weights w are like slopes
- Basis functions B are synthetic variables
    - In spirit like a squared or cubed terms
    - But observed data not used to build B
    - B values turn on weights in different regions of x variable

B-Spline of Climate

```
library(rethinking)
data(cherry_blossoms)
d <- cherry_blossoms
precis(d)
```

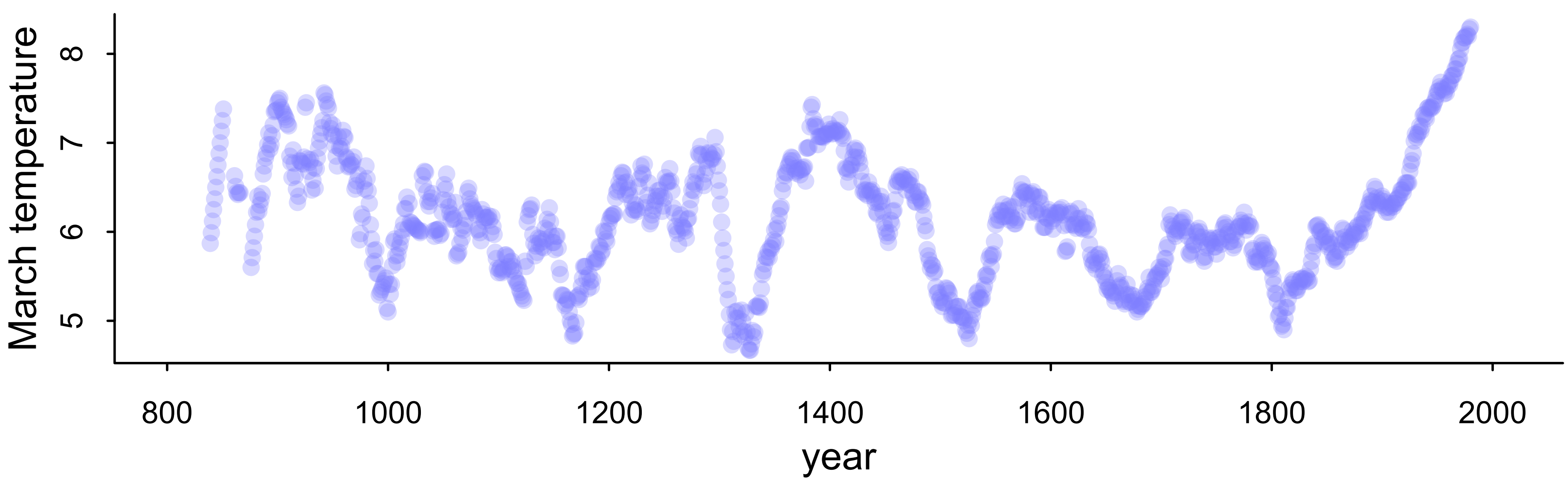
R code 4.72

B-Spline of Climate

```
library(rethinking)
data(cherry_blossoms)
d <- cherry_blossoms
precis(d)
```

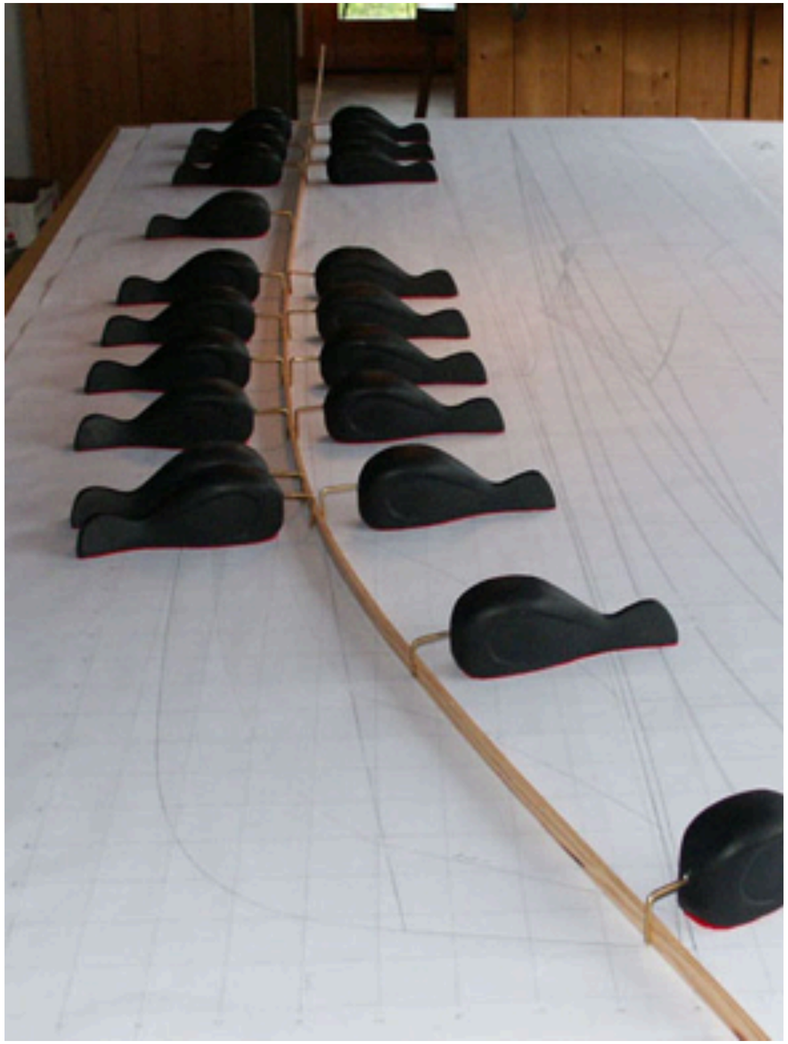
R code 4.72

```
'data.frame': 1215 obs. of 5 variables:
               mean     sd    5.5%   94.5%      histogram
year        1408.00 350.88  867.77 1948.23
doy          104.54   6.41   94.43  115.00
temp           6.14   0.66    5.15    7.29
temp_upper     7.19   0.99    5.90    8.90
temp_lower     5.10   0.85    3.79    6.37
```



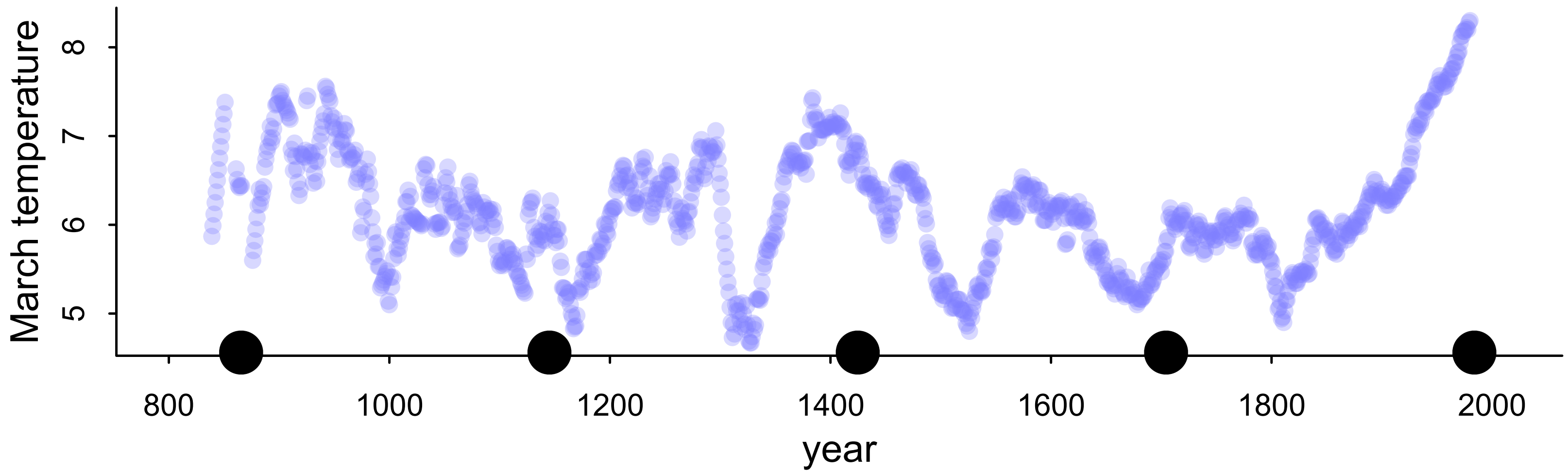
B-Spline of Climate

- B-Spline recipe:
  - Choose some knots — locations on predictor variable where the spline is anchored
  - Choose degree of basis functions — how wiggly
  - Find posterior distribution of weights



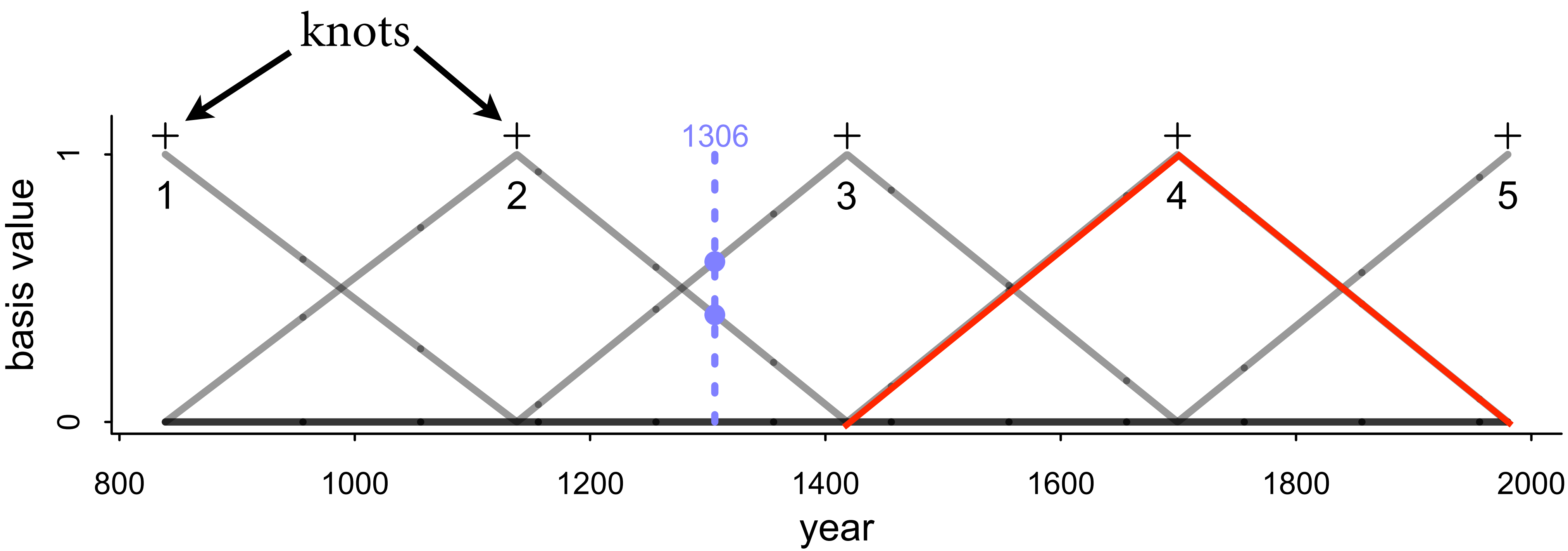
Knots

- More knots means more wiggle in global function



Basis functions

- Starter example: Linear basis functions



Each basis defines the local region where it influences the spline

Figure 4.12

Weights

- Just an ordinary linear model now
- Basis functions in a matrix **B**

$$\mu_i = \alpha + w_1 B_{i,1} + w_2 B_{i,2} + w_3 B_{i,3} + \ldots$$

R code
4.76

```
m4.7 <- quap(
    alist(
        T ~ dnorm( mu , sigma ) ,
        mu <- a + B %*% w ,
        a ~ dnorm(6,10),
        w ~ dnorm(0,1),
        sigma ~ dexp(1)
    ),
    data=list( T=d2$temp , B=B ) ,
    start=list( w=rep( 0 , ncol(B) ) ) )
```

Weights

$$\mu_i = \alpha + w_1 B_{i,1} + w_2 B_{i,2} + w_3 B_{i,3} + \ldots$$

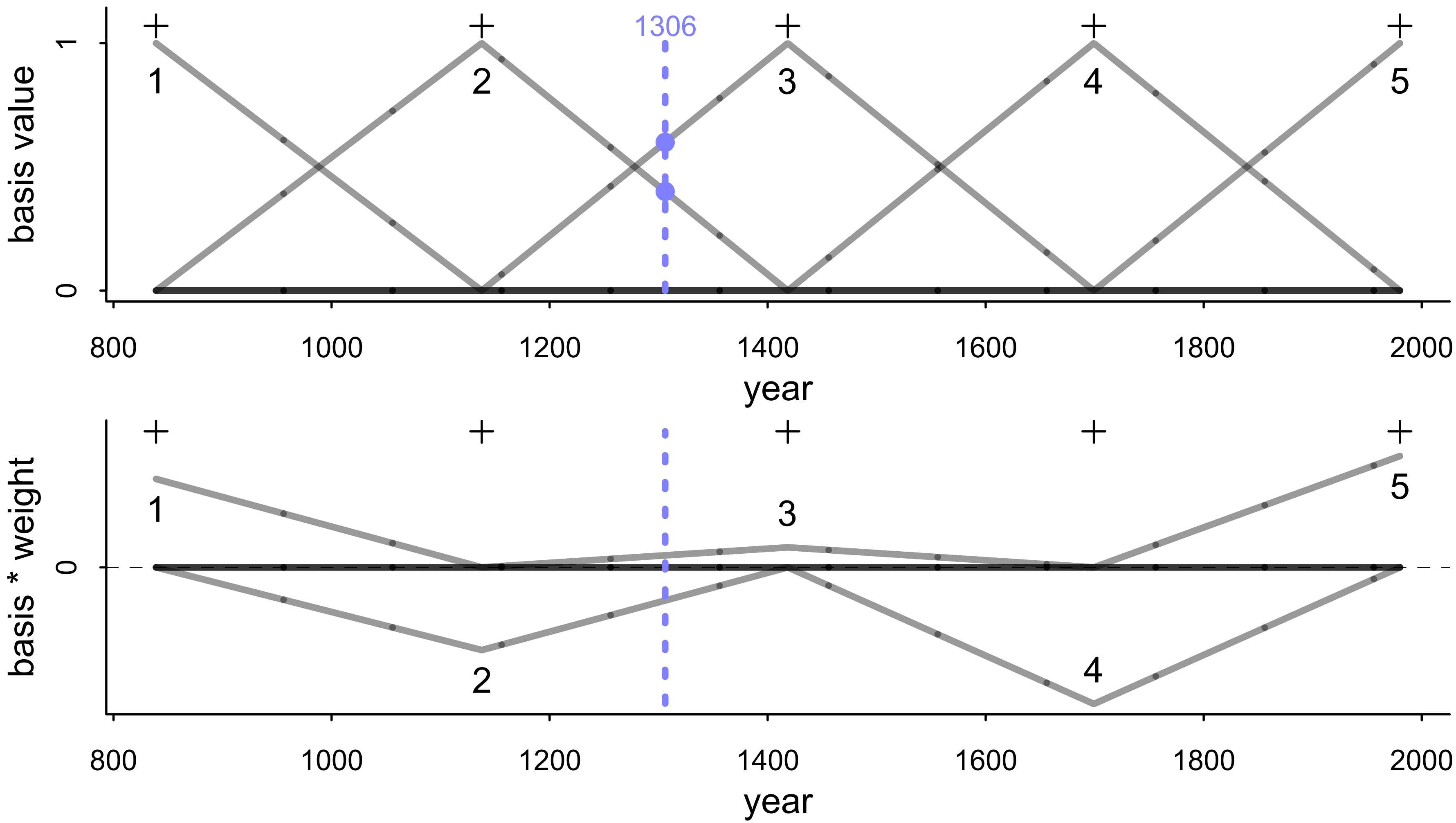
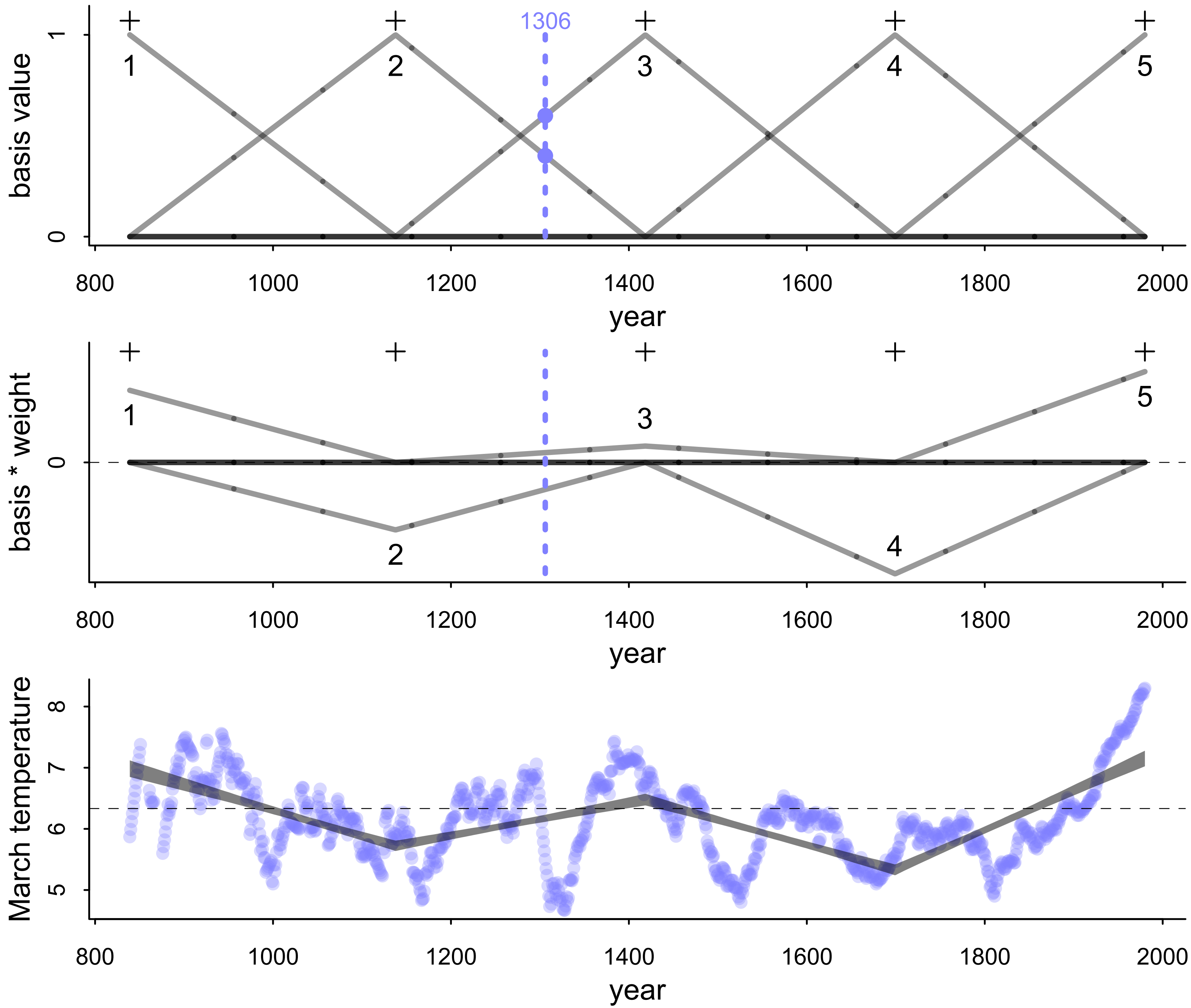
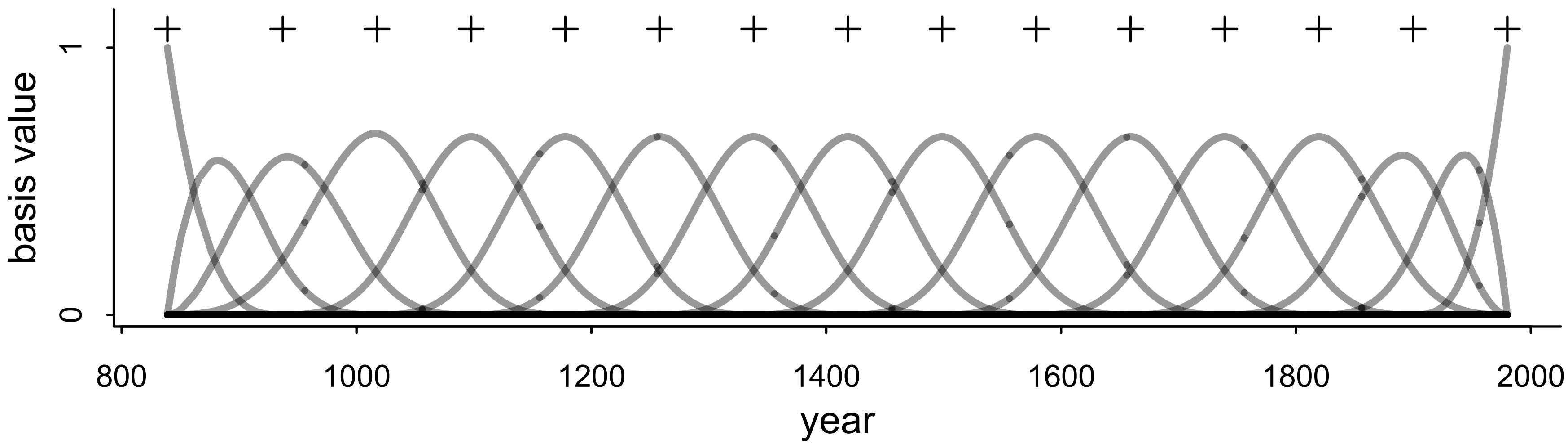


Figure 4.12



basis value
1306
1
2
3
4
5
0
1
800
1000
1200
1400
1600
1800
2000
year
basis * weight
1
2
3
4
5
0
800
1000
1200
1400
1600
1800
2000
year
March temperature
5
6
7
8
800
1000
1200
1400
1600
1800
2000
year

Figure 4.12



15 knots
3rd degree basis functions
(all code in text)

Figure 4.13

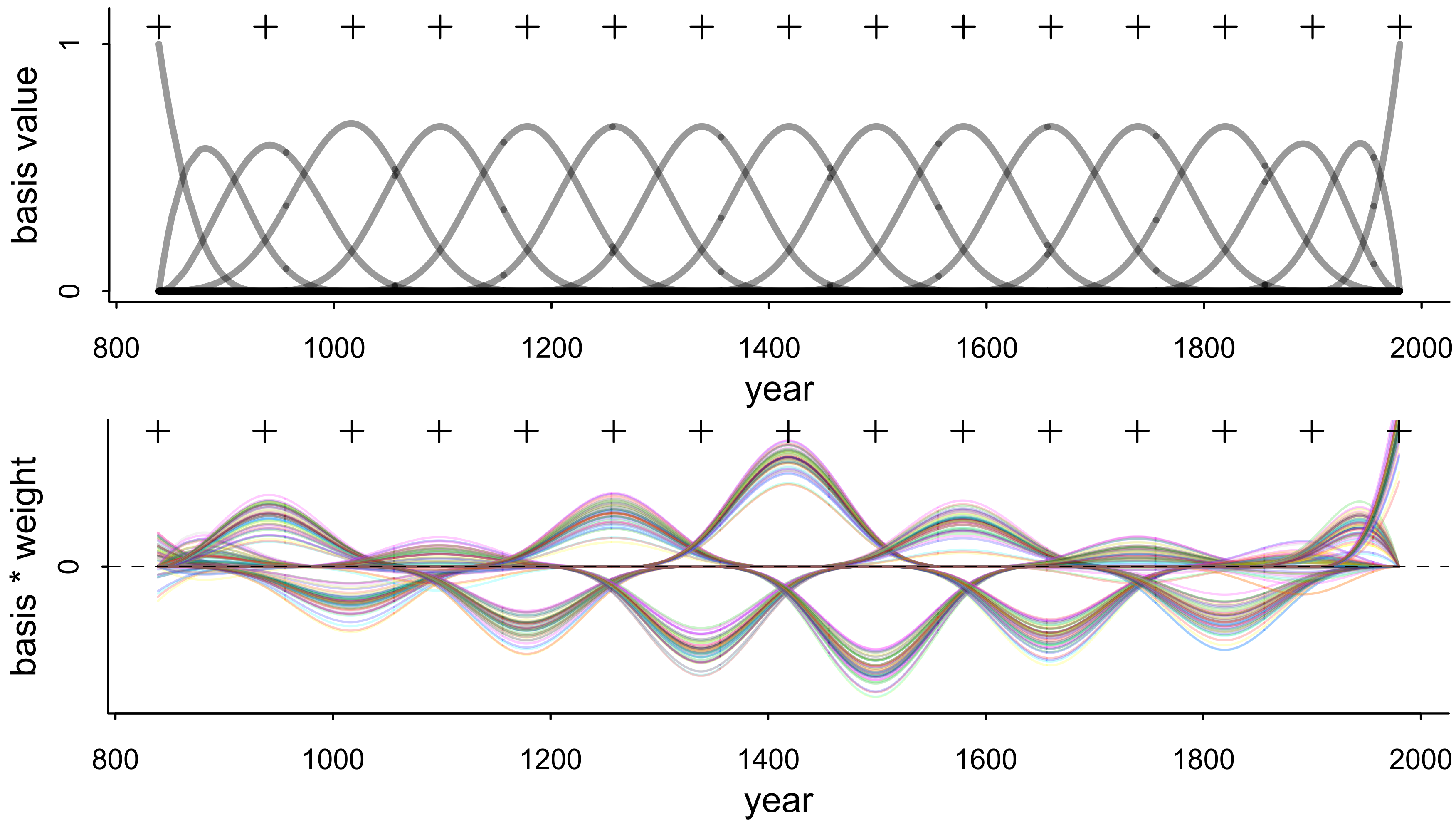

basis value
1
0
800
1000
1200
1400
1600
1800
2000
year
basis * weight
0
800
1000
1200
1400
1600
1800
2000
year

Figure 4.13

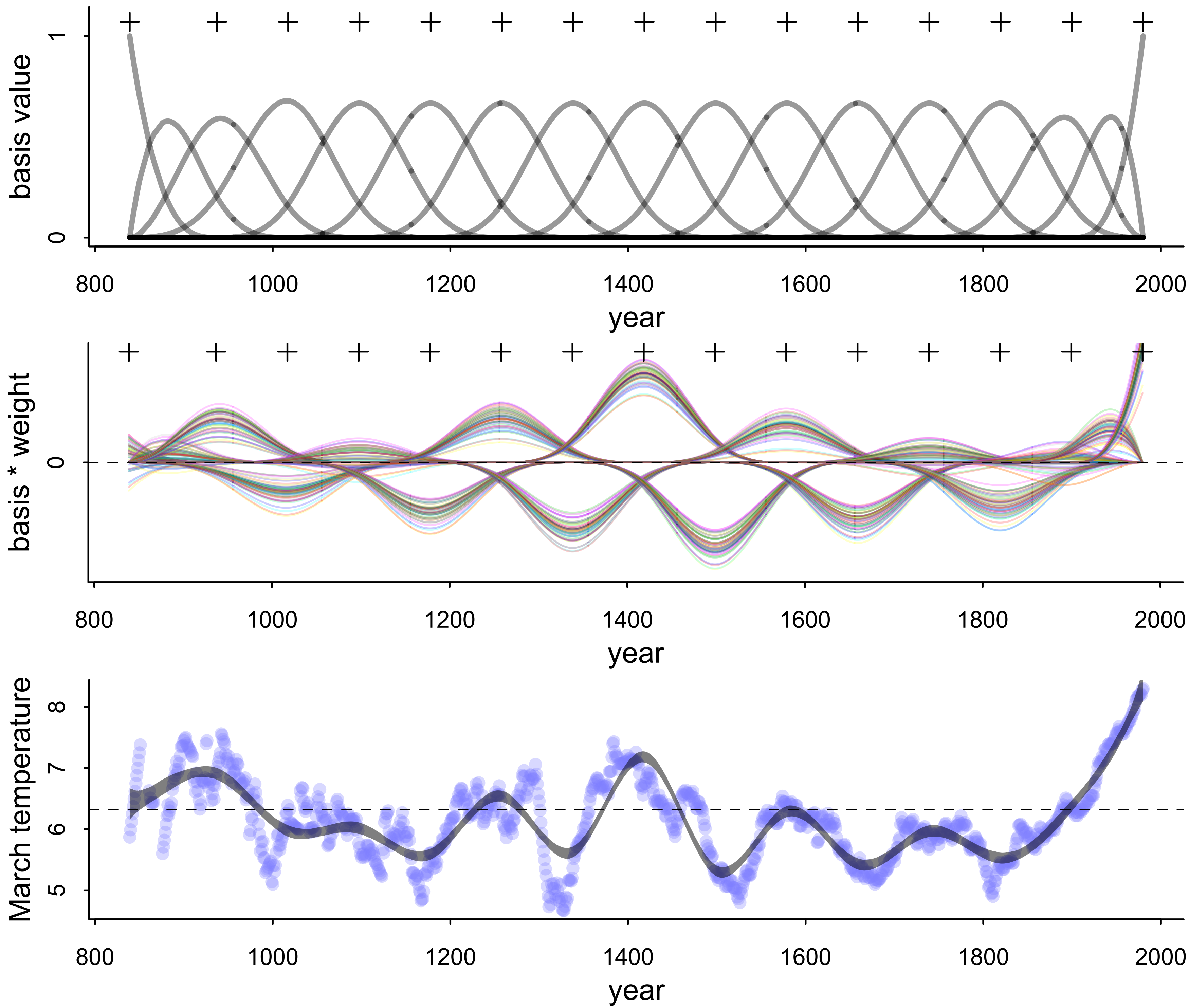


Figure 4.13

Spline possibilities

- Knots and basis degree are choices
- Must worry about *overfitting* data (Chapter 7)
- Other types of splines don't require knots
- Another idea: Gaussian Processes (Chapter 14)
- All splines are descriptive, not mechanistic

Work

- Homework online later today
- In the New Year:
  - Multiple regression
  - Causal graphs
  - Colliders
  - Overfitting
  - Multilevel models
  - Adventures in covariance