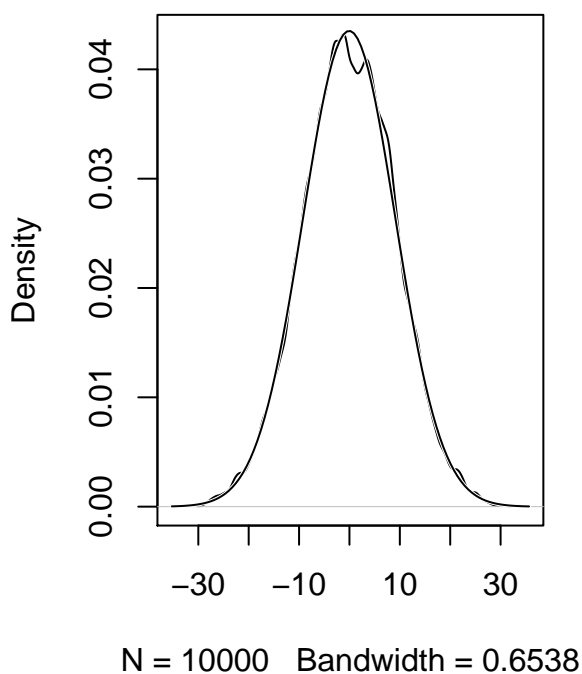
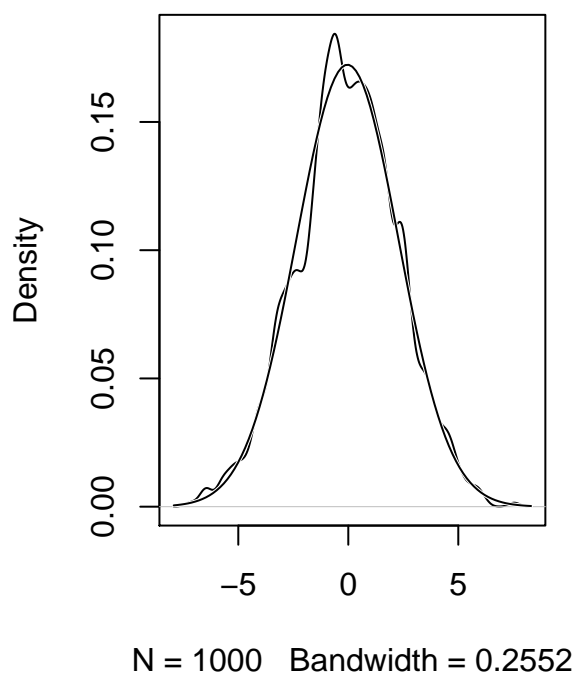


4 - Linear Models

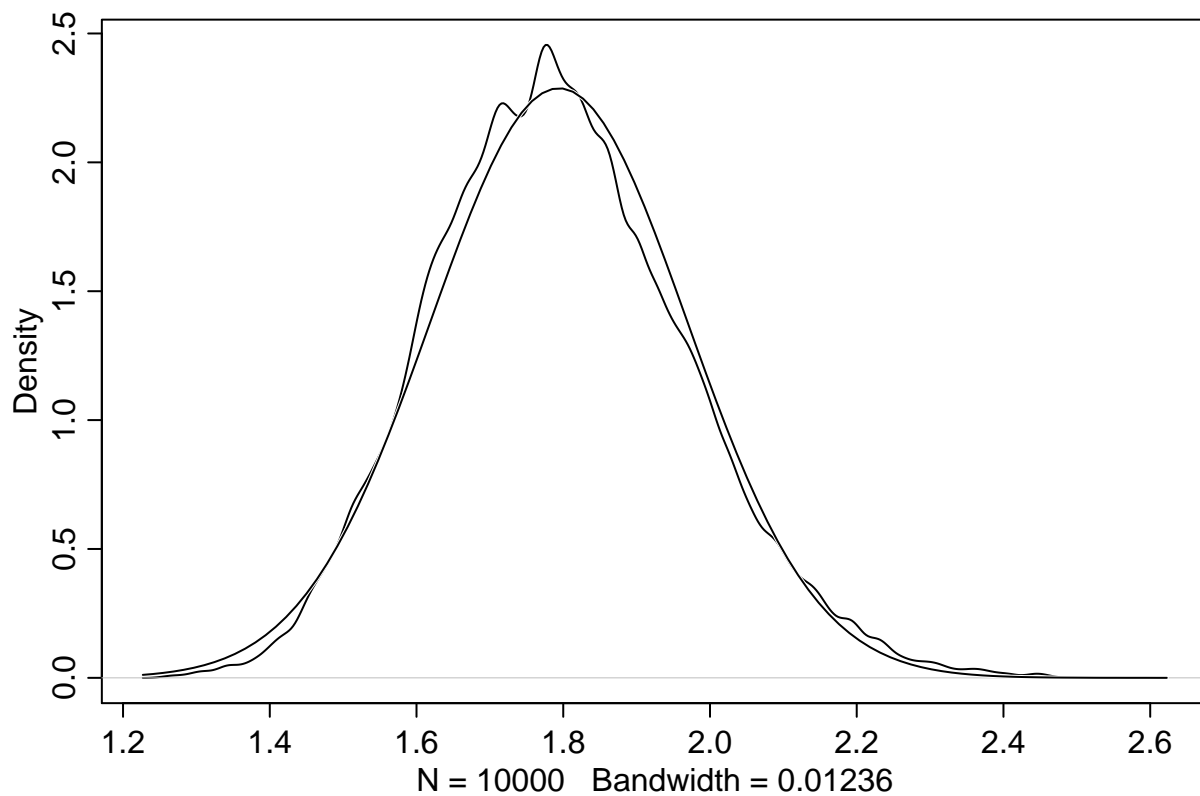
4.1.1. Normal by addition

```
# 4.1
pos <- replicate(1000, sum(runif(16, -1, 1)))
par(mfrow=c(1, 2))
dens(pos, norm.comp = T)
dens(replicate(10000, sum(runif(256, -1, 1))), norm.comp = T)
```

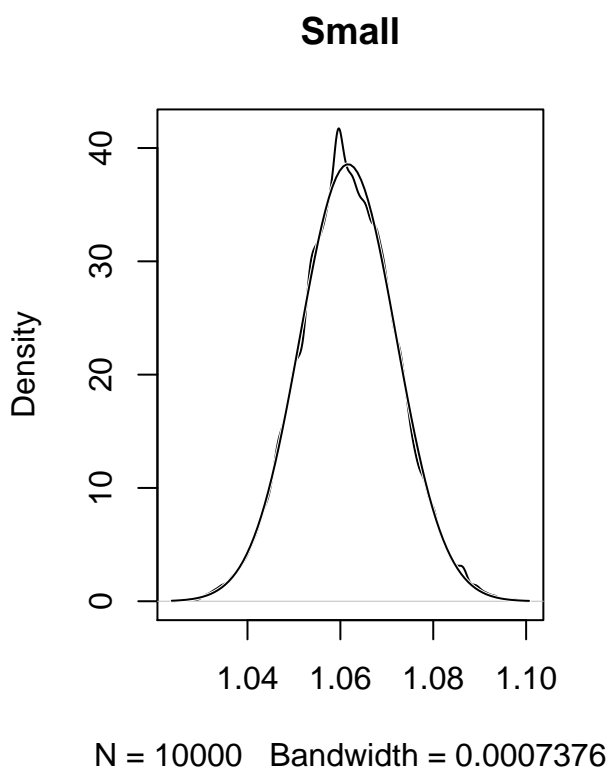
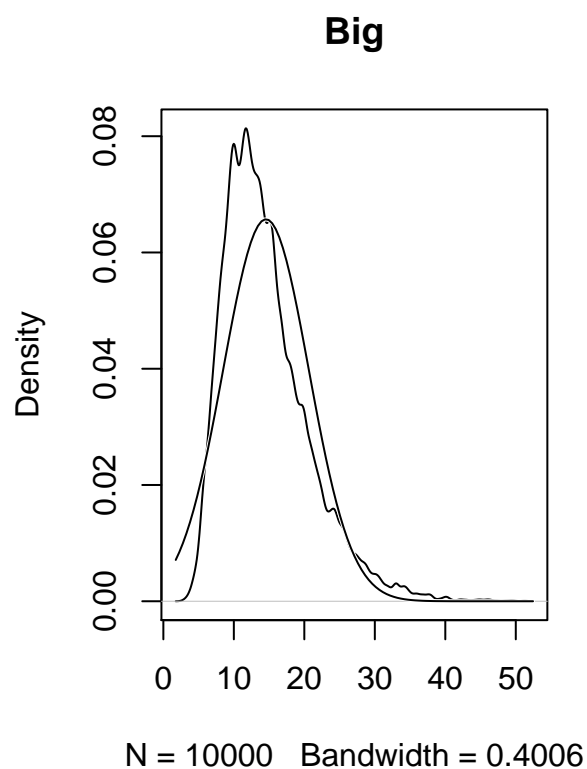


4.1.2. Normal by multiplication

```
# 4.2
dens(replicate(1e4, prod(1 + runif(12, 0, 0.1))), norm.comp = T)
```

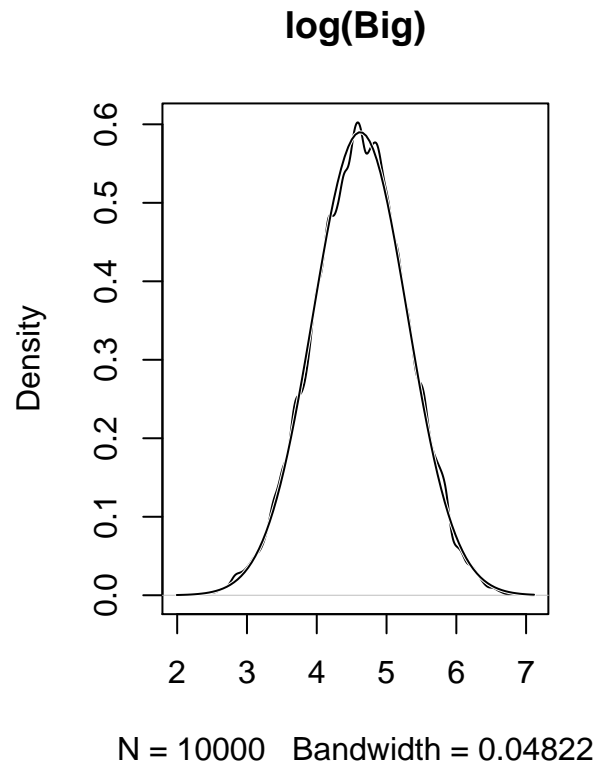
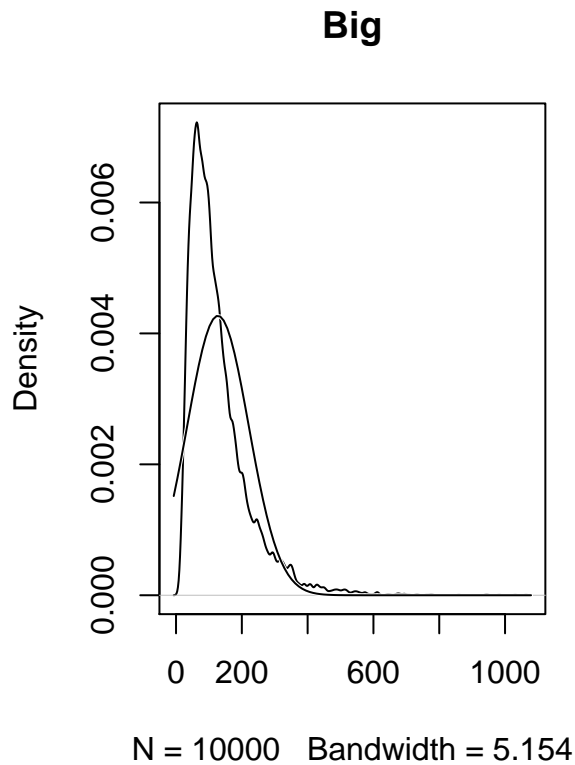


```
# 4.4
big <- replicate(1e4, prod(1 + runif(12, 0, 0.5)))
small <- replicate(1e4, prod(1 + runif(12, 0, 0.01)))
par(mfrow=c(1, 2))
dens(big, norm.comp = T, main = "Big")
dens(small, norm.comp = T, main = "Small")
```



Normal by log-multiplication

```
# 4.5
big <- replicate(1e4, prod(1 + runif(12, 0, 1)))
log_big <- log(big)
par(mfrow=c(1, 2))
dens(big, norm.comp = T, main = "Big")
dens(log_big, norm.comp = T, main = "log(Big)")
```



4.3 A Gaussian model of height

```
# 4.7
library(rethinking)
data(Howell1)
d <- Howell1
```

```
# 4.8
str(d)
```

```
## '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 ...
```

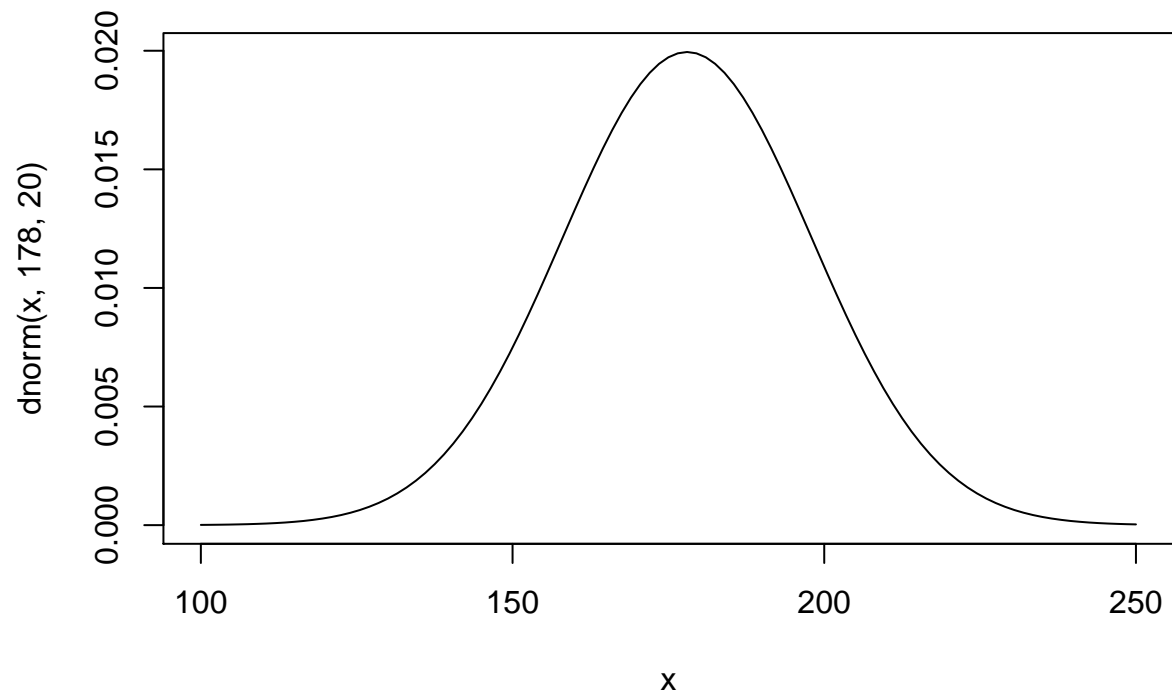
We want heights of adults only (352 rows):

```
# 4.10
d2 <- d[d$age >= 18, ]
```

4.3.2 The model

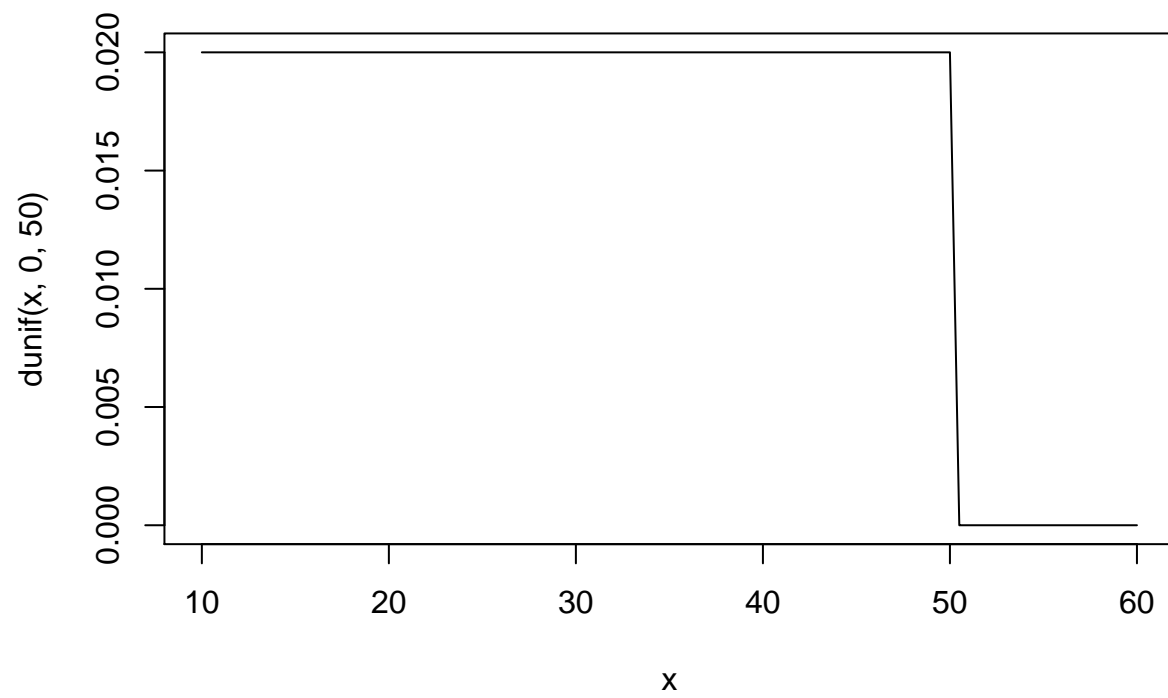
Height mean:

```
# 4.11  
curve(dnorm(x, 178, 20), from=100, to=250)
```

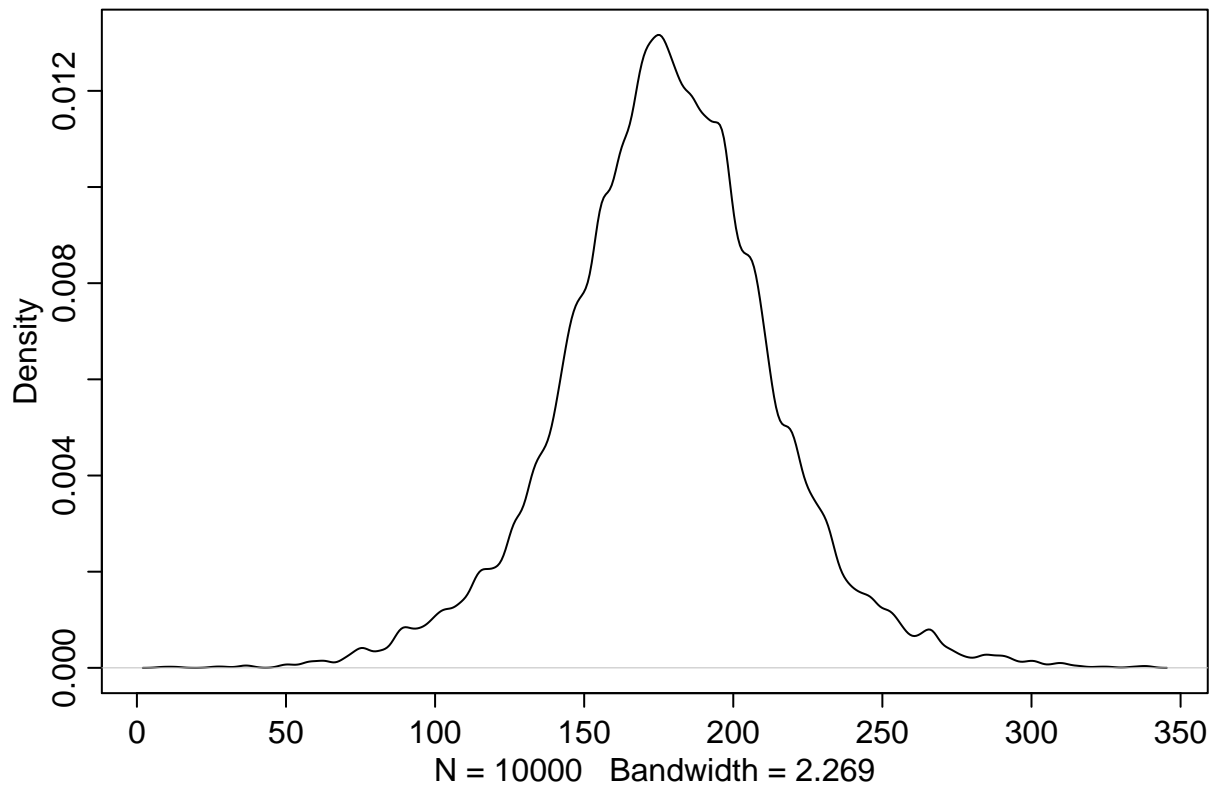


Height standard deviation:

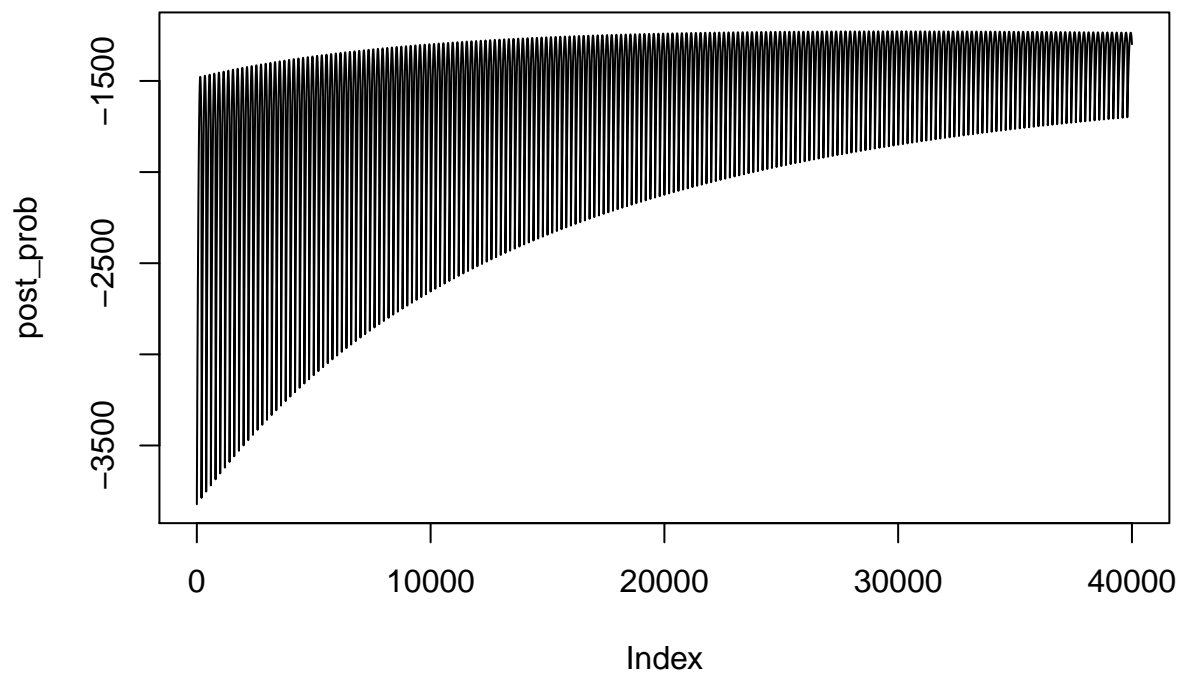
```
# 4.12  
curve(dunif(x, 0, 50), from=10, to=60)
```



```
# 4.13
sample_mu <- rnorm(1e4, 178, 20)
sample_sigma <- runif(1e4, 0, 50)
prior_h <- rnorm(1e4, sample_mu, sample_sigma)
dens(prior_h)
```



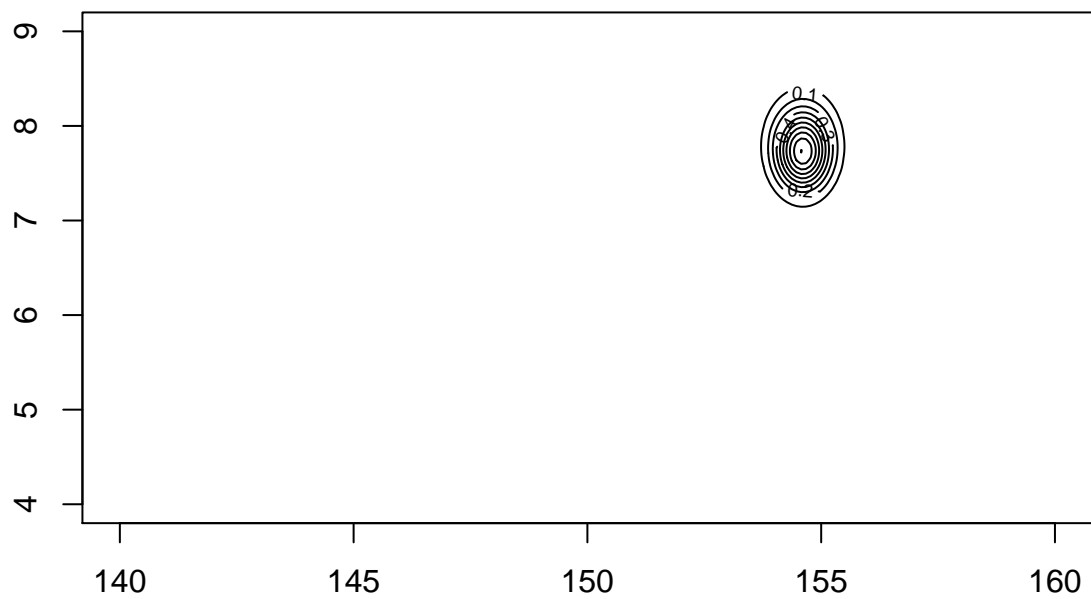
```
# 4.14
mu_list <- seq(from=140, to=160, length.out=200)
sigma_list <- seq(from=4, to=9, length.out=200)
post <- expand.grid(mu=mu_list, sigma=sigma_list)
post_ll <- sapply(1:nrow(post), function(i) sum(dnorm(
  d2$height,
  mean=post$mu[i],
  sd=post$sigma[i],
  log=T
)))
post_prob <- post_ll + dnorm(post$mu, 178, 20, T) + dunif(post$sigma, 0, 50, T)
plot(post_prob, type="l")
```



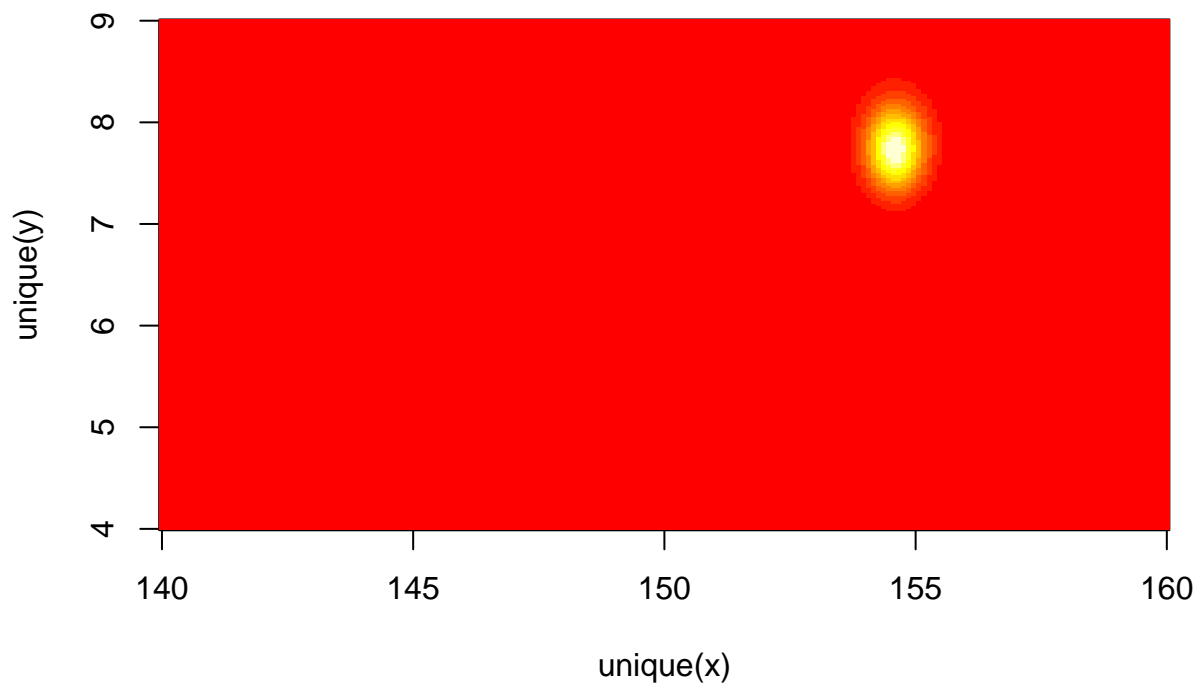
```
post_prob <- exp(post_prob - max(post_prob))
```

```
# 4.15
```

```
contour_xyz(post$mu, post$sigma, post_prob)
```

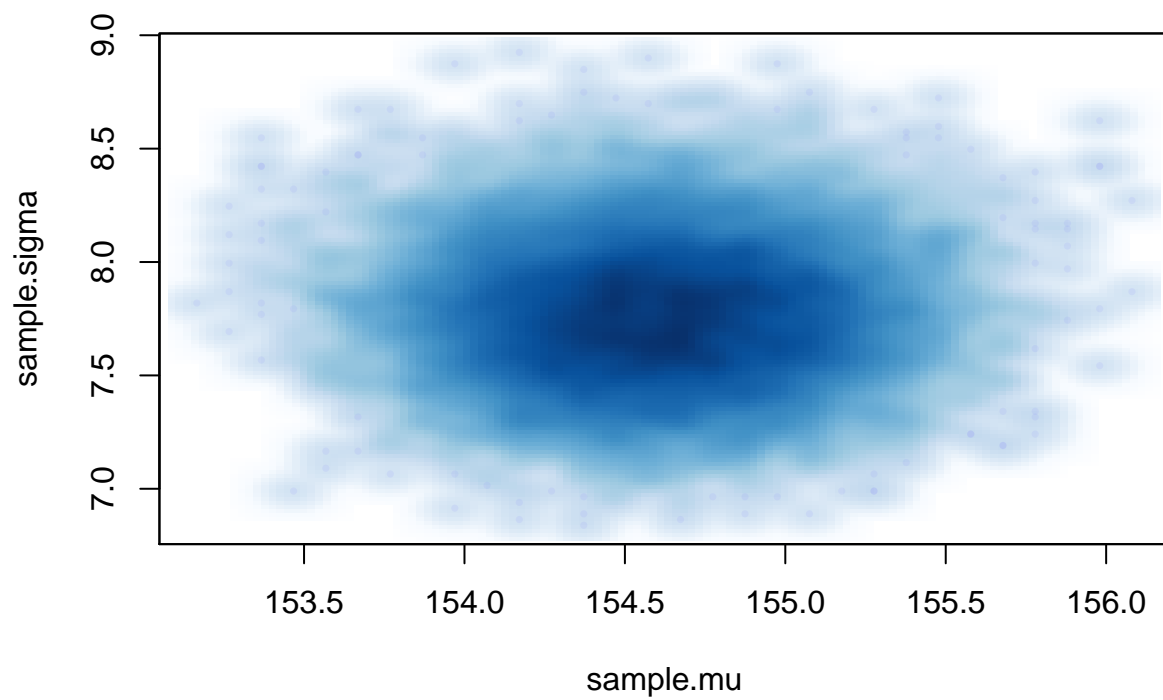
```
# 4.16  
image_xyz(post$mu, post$sigma, post_prob)
```



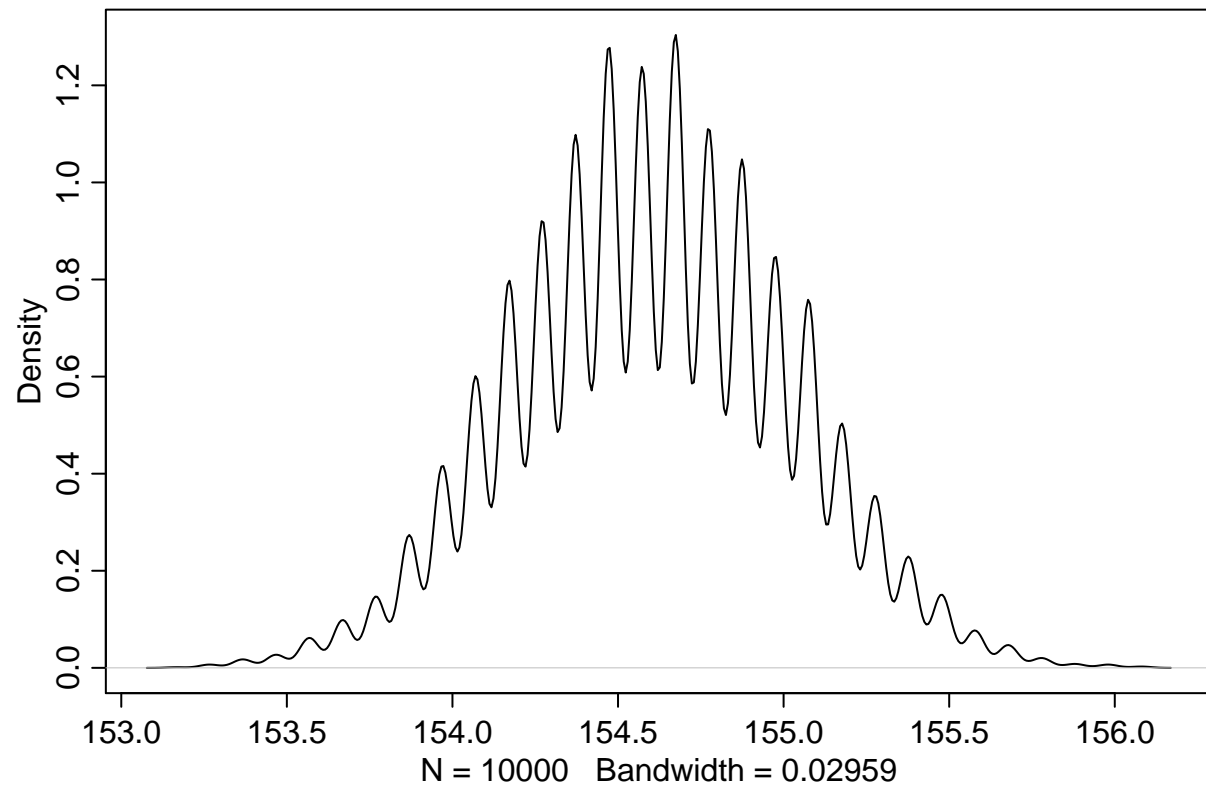
4.3.4 Sampling from the posterior

```
# 4.17
sample.rows <- sample(1:nrow(post), size=1e4, replace = T, prob = post_prob)
sample.mu <- post$mu[sample.rows]
sample.sigma <- post$sigma[sample.rows]

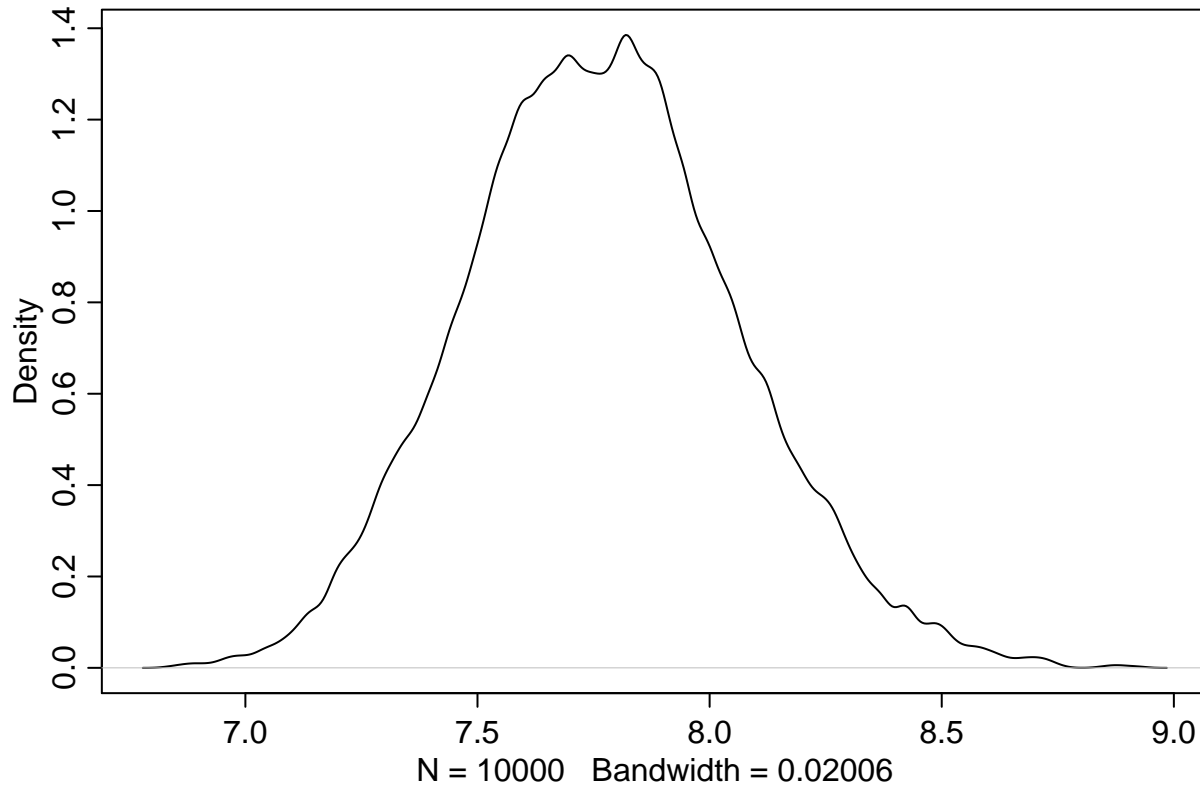
# 4.18
smoothScatter(sample.mu, sample.sigma, cex=0.5, pch=16, col=col.alpha(rangi2, 0.1))
```



```
# 4.19  
dens(sample.mu)
```



```
dens(sample.sigma)
```



```
# 4.20
HPDI(sample.mu)
```

```
## |0.89 0.89|
## 153.8693 155.1759
```

```
HPDI(sample.sigma)
```

```
## |0.89 0.89|
## 7.266332 8.195980
```

Smaller Sample

To illustrate the posterior is not always Gaussian in shape.

```
# 4.22
d3 <- sample(d2$height, size=10)

small.post_ll <- sapply(1:nrow(post), function(i) sum(dnorm(d3, mean=post$mu[i], sd=post$sigma[i], log=
small.post_product <- small.post_ll + dnorm(post$mu, 178, 20, T) + dunif(post$sigma, 0, 50, T)

small.post_proba <- exp(small.post_product - max(small.post_product))

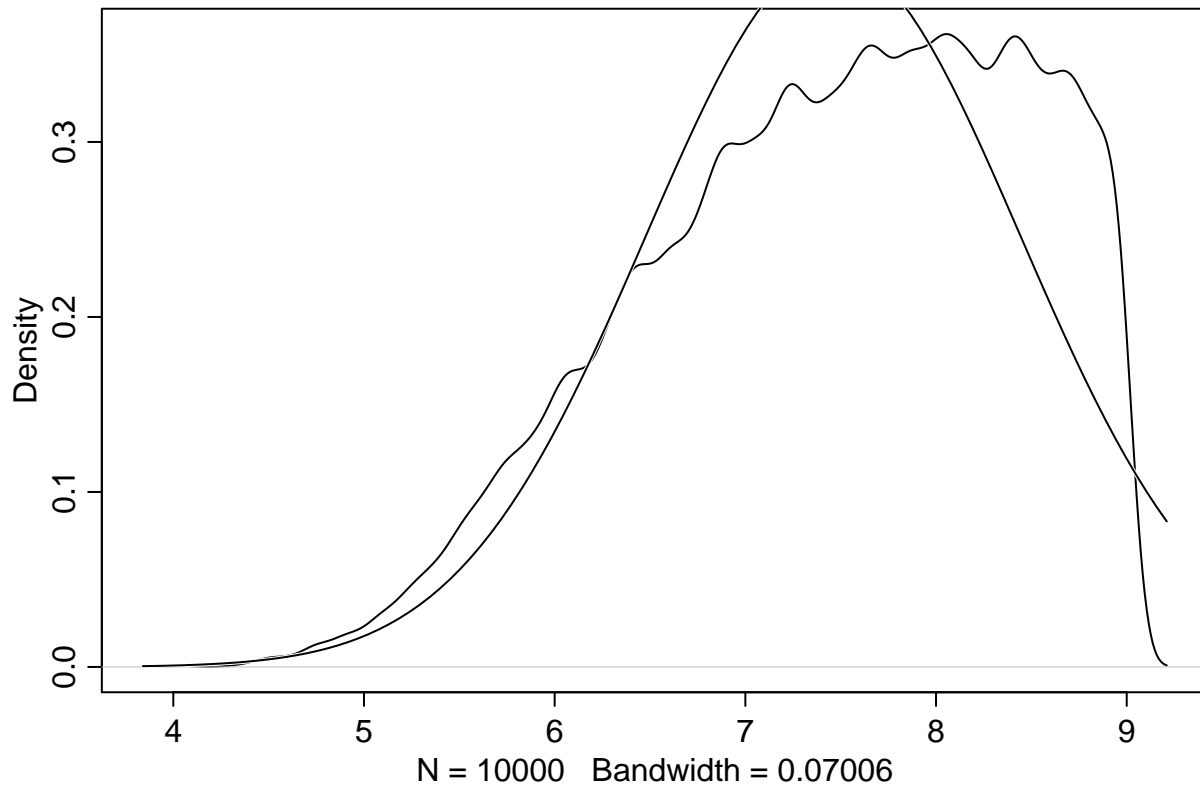
small.sample.rows <- sample(1:nrow(post), size=1e4, replace = T, prob=small.post_proba)

small.sample.mu <- post$mu[small.sample.rows]
```

```
small.sample.sigma <- post$sigma[small.sample.rows]
```

```
# 4.23
```

```
dens(small.sample.sigma, norm.comp = T)
```



4.3.5. Fitting the model with *map*

map finds the values of μ and σ that maximize the posterior probability.

```
# 4.25
```

```
model.list <- alist(  
  height ~ dnorm(mu, sigma),  
  mu ~ dnorm(178, 20),  
  sigma ~ dunif(0, 50)  
)
```

```
# 4.26
```

```
model.solved <- map(model.list, data=d2)
```

```
# 4.27
```

```
precis(model.solved)
```

```
##      Mean StdDev  5.5% 94.5%  
## mu   154.61   0.41 153.95 155.27  
## sigma 7.73   0.29  7.27  8.20
```

Compare to HPDI intervals from above.

```
HPDI(sample.mu)
```

```
##      |0.89      0.89|  
## 153.8693 155.1759
```

```
HPDI(sample.sigma)
```

```
##      |0.89      0.89|  
## 7.266332 8.195980
```

We've calculated the HPDI intervals using the grid approximation. The model is solved via a quadratic approximation. The quadratic approximation does a very good in identifying the 89% intervals.

It works because the posterior is approximately Gaussian.

The priors we used so far are very weak. We'll splice in a more informative prior for μ .

```
#4.29
```

```
model.solved_narrow_mu <- map (  
  alist(  
    height ~ dnorm(mu, sigma),  
    mu ~ dnorm(178, 0.1),  
    sigma ~ dunif(0, 50)  
  ),  
  data=d2)  
precis(model.solved_narrow_mu)
```

```
##           Mean StdDev   5.5%  94.5%  
## mu      177.86   0.10 177.70 178.02  
## sigma   24.52   0.93  23.03  26.00
```

The estimate for μ has hardly moved off the prior. The estimate for σ has changed a lot, even though we didn't change the prior at all. Our machine had to make μ and σ fit out data. Since μ is very concentrated around 178, the machine had to change σ to accomodate the data.

4.3.6. Sampling from a *map* fit.

Variance-covariance matrix:

```
# 4.30
```

```
vcov(model.solved)
```

```
##           mu      sigma  
## mu  0.1697533860 0.0002219694  
## sigma 0.0002219694 0.0849230296
```

We can split it into (1) vector of variances, and (2) the correlation matrix:

```
# 4.31
```

```
diag(vcov(model.solved))
```

```
##           mu      sigma  
## 0.16975339 0.08492303
```

```
cov2cor(vcov(model.solved))
```

```
##           mu      sigma  
## mu  1.00000000 0.00184872  
## sigma 0.00184872 1.00000000
```

Sampling from the posterior:

```
# 4.34
coef(model.solved)

##           mu           sigma
## 154.607201    7.731646

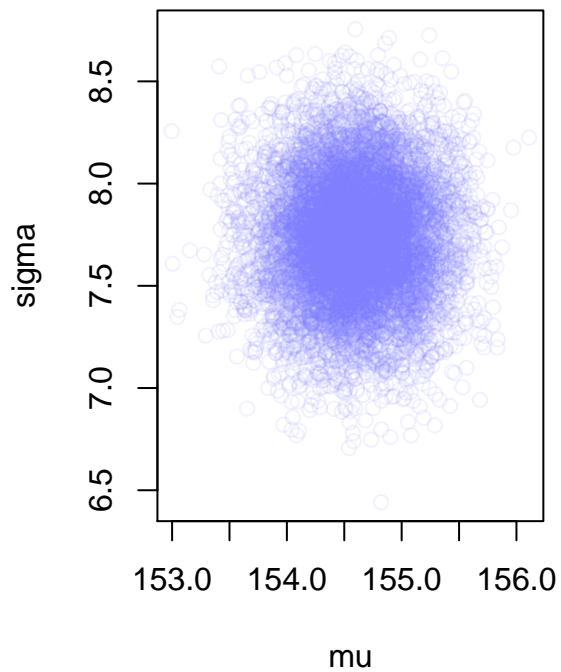
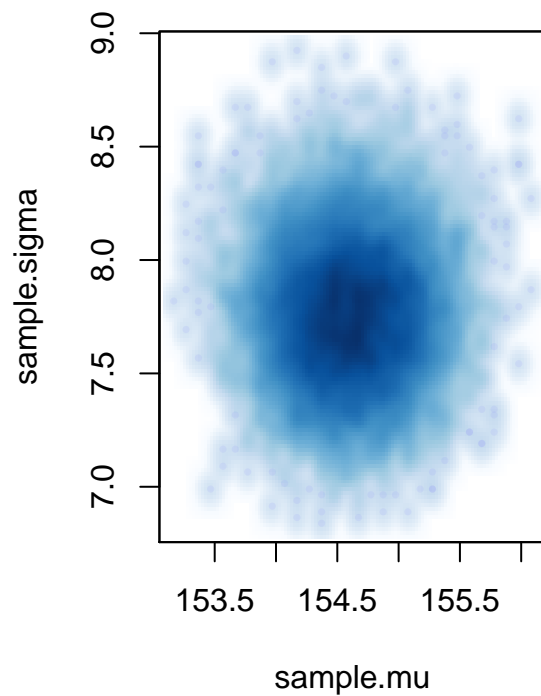
library(MASS)
post <- mvrnorm(n=1e4, mu=coef(model.solved), Sigma=vcov(model.solved))
post = data.frame(post)
head(post)

##           mu           sigma
## 1 154.7097    7.549941
## 2 154.8767    8.291259
## 3 155.0473    7.971083
## 4 155.0242    8.124977
## 5 153.8408    7.748980
## 6 154.9519    7.709578

# 4.33
precis(post)

##           Mean StdDev |0.89 0.89|
## mu      154.60    0.41 153.95 155.25
## sigma    7.73    0.29  7.30  8.23

par(mfrow=c(1, 2))
smoothScatter(sample.mu, sample.sigma, cex=0.5, pch=16, col=col.alpha(rangi2, 0.1))
plot(post, col=col.alpha(rangi2, 0.1))
```

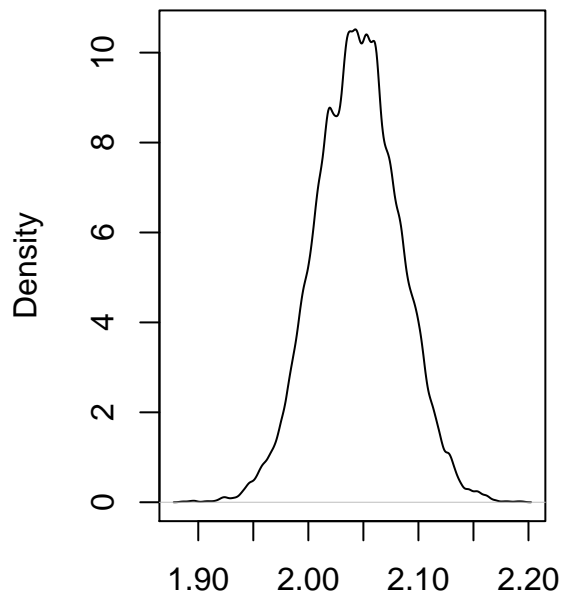



Getting sigma right

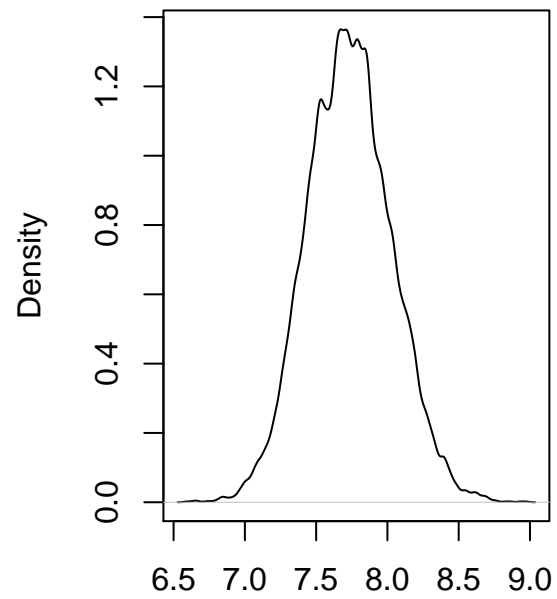
The quadratic assumption for σ may be not correct. In this case it's better to estimate $\log(\sigma)$ instead, because the distribution of \log will be much closer to Gaussian.

```
# 4.35
model.solved_log_sigma <- map(
  alist(
    height ~ dnorm(mu, exp(log_sigma)),
    mu ~ dnorm(178, 20),
    log_sigma ~ dnorm(2, 10)
  ),
  data = d2
)

# 4.36
post <- mvnrm(n=1e4, mu=coef(model.solved_log_sigma), Sigma=vcov(model.solved_log_sigma))
post <- data.frame(post)
par(mfrow=c(1, 2))
dens(post$log_sigma)
dens(exp(post$log_sigma))
```



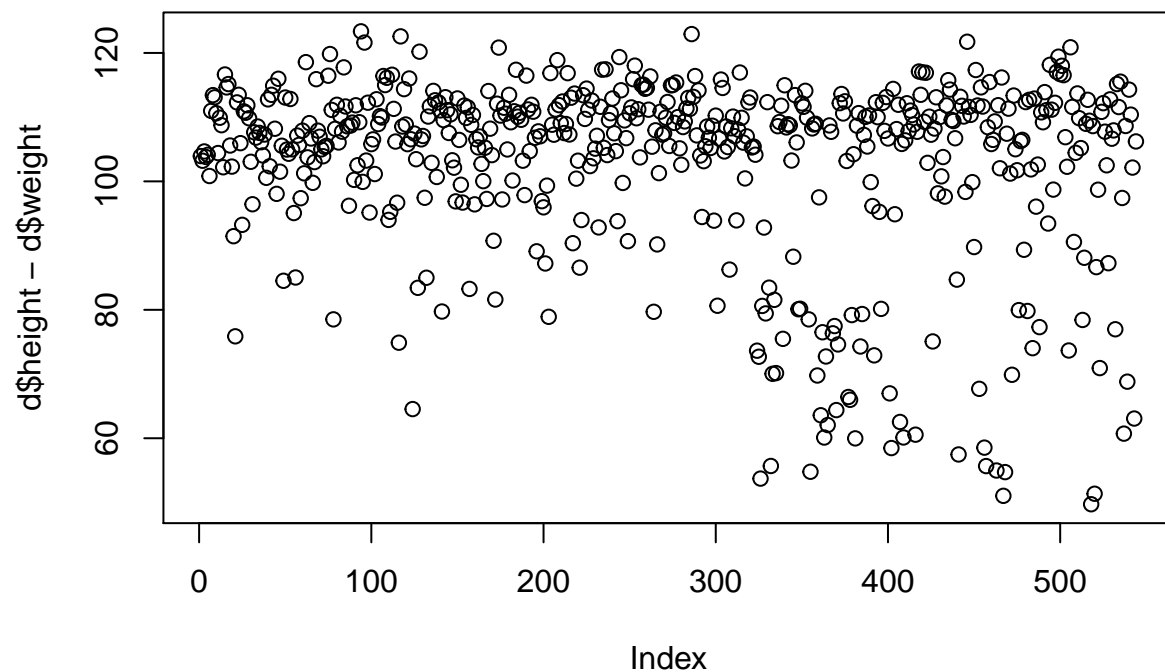
N = 10000 Bandwidth = 0.002725



N = 10000 Bandwidth = 0.02108

4.4. Adding a predictor

```
#4.37
plot(d$height ~ d$weight)
```



4.4.2. Fitting the model

```
# 4.38
model.linear_m43 <- map(
  alist(
    height ~ dnorm(mu, sigma),
    mu <- a + b * weight,
    a ~ dnorm(178, 100),
    b ~ dnorm(0, 10),
    sigma ~ dunif(0, 50)
  ),
  data=d2
)
```

```
# 4.40
precis(model.linear_m43, corr=T)
```

##	Mean	StdDev	5.5%	94.5%	a	b	sigma
## a	113.90	1.91	110.86	116.95	1.00	-0.99	0
## b	0.90	0.04	0.84	0.97	-0.99	1.00	0
## sigma	5.07	0.19	4.77	5.38	0.00	0.00	1

Centering

```
# 4.42
d2$weight_centered <- d2$weight - mean(d2$weight)
```

```
# 4.43
model.linear_m44 <- map(
  alist(
    height ~ dnorm(a + b * weight_centered, sigma),
    a ~ dnorm(178, 100),
    b ~ dnorm(0, 10),
    sigma ~ dunif(0, 50)
  )
  , data=d2
)
```

```
# 4.44
precis(model.linear_m44, corr=T)
```

```
##           Mean StdDev   5.5%  94.5% a b sigma
## a       154.60   0.27 154.17 155.03 1 0      0
## b         0.91   0.04   0.84   0.97 0 1      0
## sigma    5.07   0.19   4.77   5.38 0 0      1
```

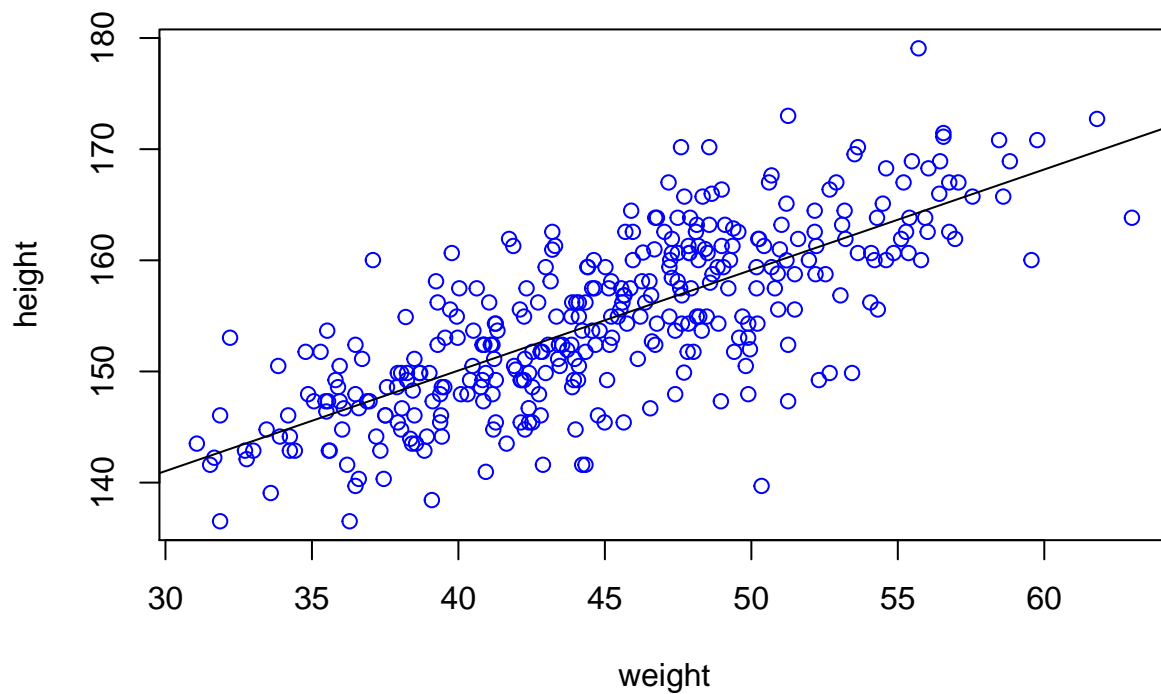
The new estimate for α is now the same as mean:

```
mean(d2$height)
```

```
## [1] 154.5971
```

Let's plot the posterior against the data:

```
# 4.45
plot(height ~ weight, data=d2, col="blue")
abline(a=coef(model.linear_m43)["a"], b = coef(model.linear_m43)["b"])
```



This line is just the posterior mean, the most plausible line. There are infinite regression lines from the posterior.

Let's extract some examples from the model:

```
# 4.46
post <- mvnrm(n=1e4, mu=coef(model.linear_m43), Sigma=vcov(model.linear_m43))
post <- data.frame(post)
post[1:6, ]
```

```
##           a           b      sigma
## 1 111.5940 0.9508733 5.339035
## 2 111.9771 0.9430517 5.282626
## 3 113.6294 0.9159170 4.829141
## 4 111.2441 0.9459894 5.296611
## 5 115.9471 0.8610052 4.972675
## 6 112.7264 0.9460034 4.965899
```

Let's try on the small data set first to see how the regression lines vary:

```
# 4.48

ablines_N = function (N_) {
  library(rethinking)
  data(Howell1)
  d <- Howell1
  d2 <- d[d$age >= 18, ]

  dN <- d2[1:N_,]
```

```

mN <- map(
  alist(
    height ~ dnorm(a + b * weight, sigma),
    a ~ dnorm(178, 100),
    b ~ dnorm(0, 10),
    sigma ~ dunif(0, 50)
  ),
  data = dN
)
post <- mvrnorm(n=40, mu=coef(mN), Sigma=vcov(mN))
post <- data.frame(post)

plot(dN$weight, dN$height, xlim=range(d2$weight), ylim=range(d2$height),
     col=range(2, xlab="weight", ylab="height")
mtext(concat("N = ", N_))

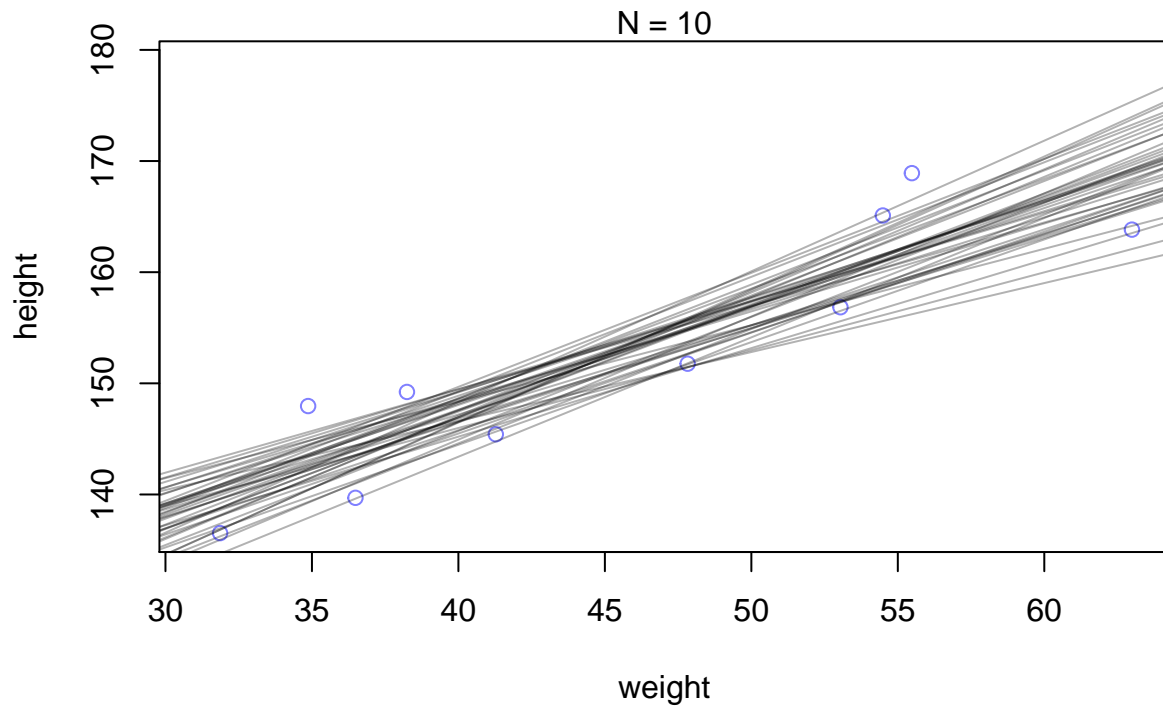
for (i in 1:nrow(post))
  abline(a=post$a[i], b=post$b[i], col=col.alpha("black", 0.3))
}

ablines_N(10)

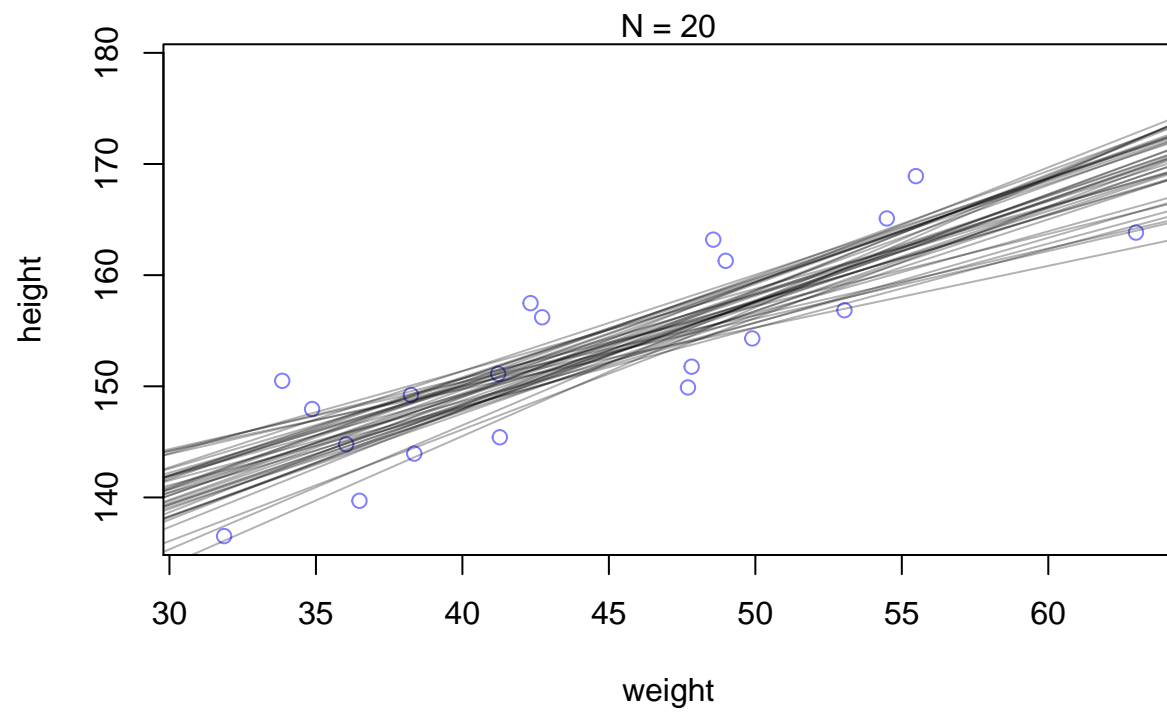
```

Caution, model may not have converged.

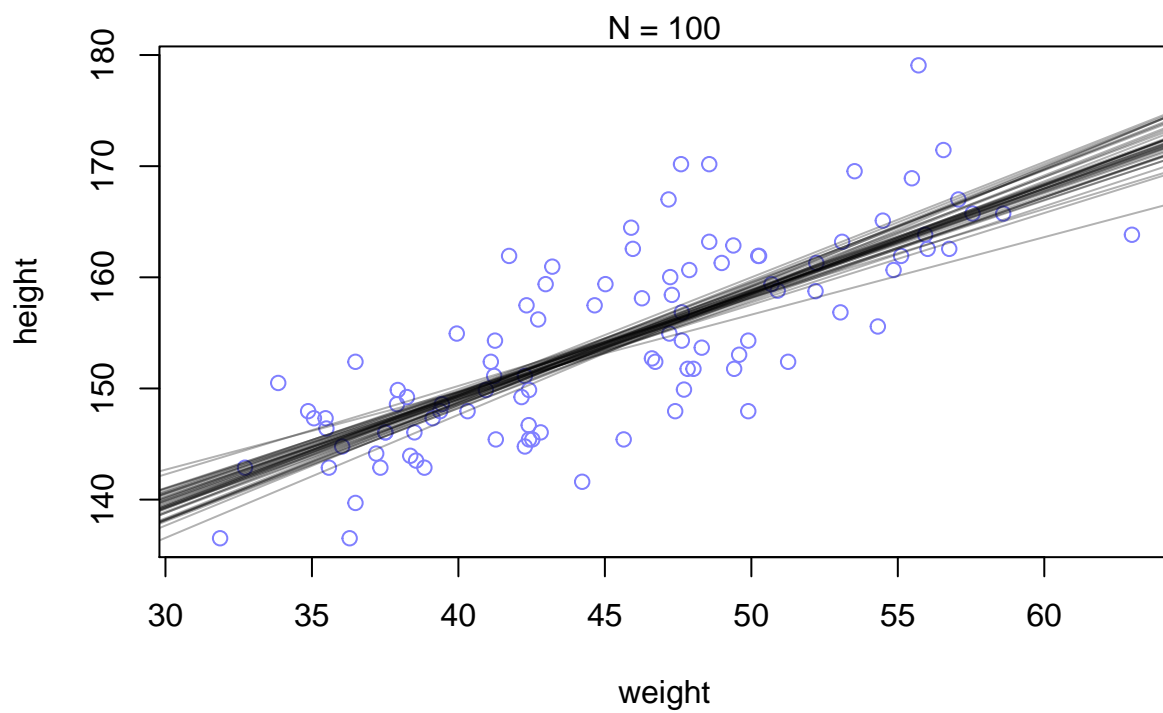
Code 1: Maximum iterations reached.



```
ablines_N(20)
```



```
ablines_N(100)
```



```
ablines_N(352)
```