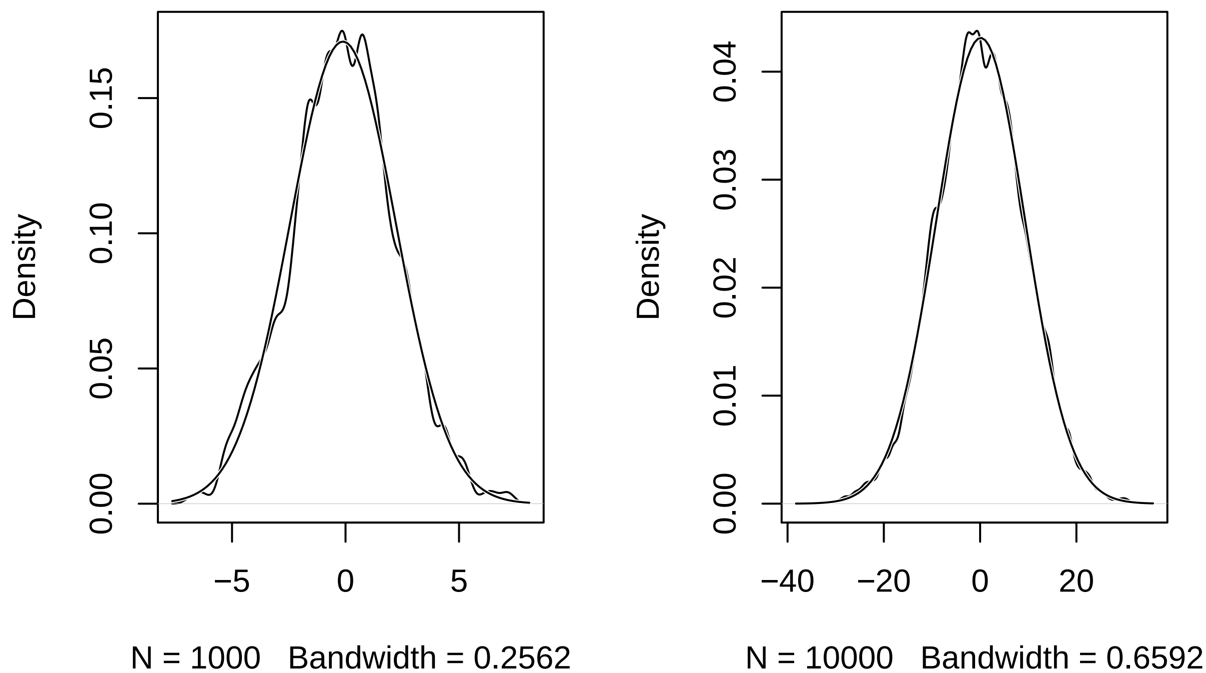
4 - Linear Models

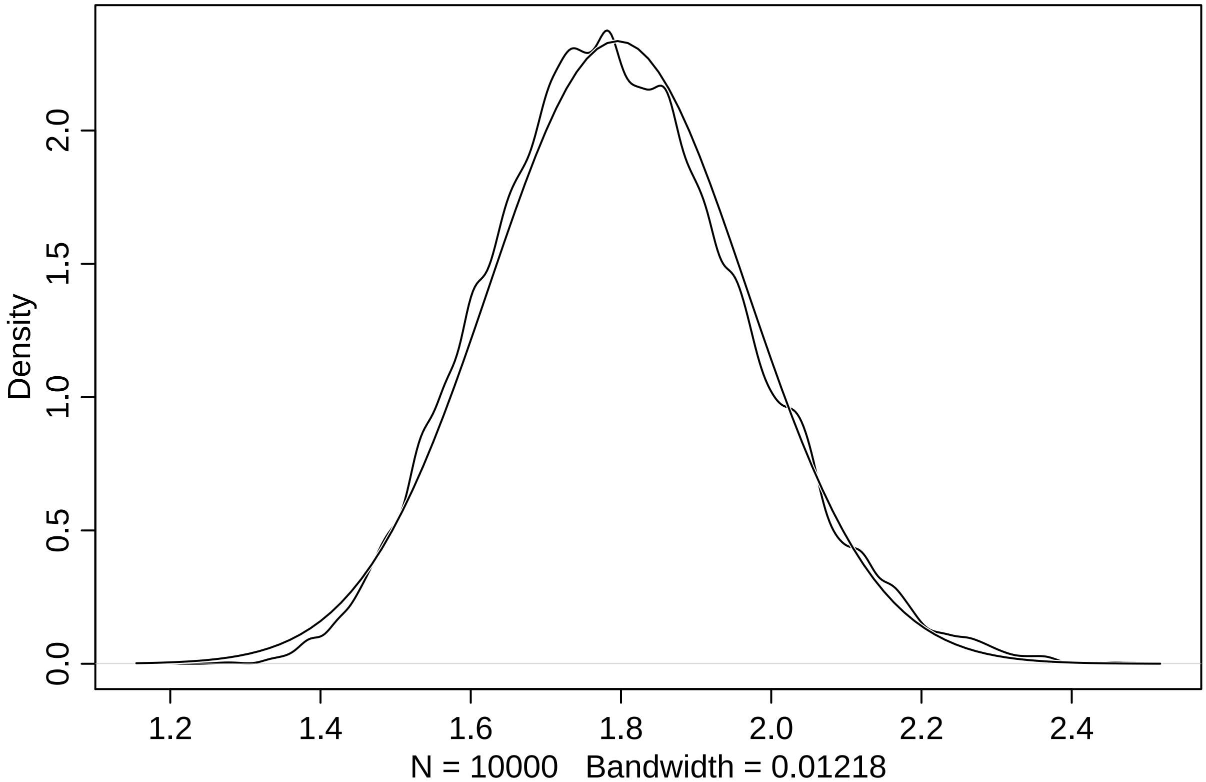
4.1.1. Normal by addition

```
# 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)
```

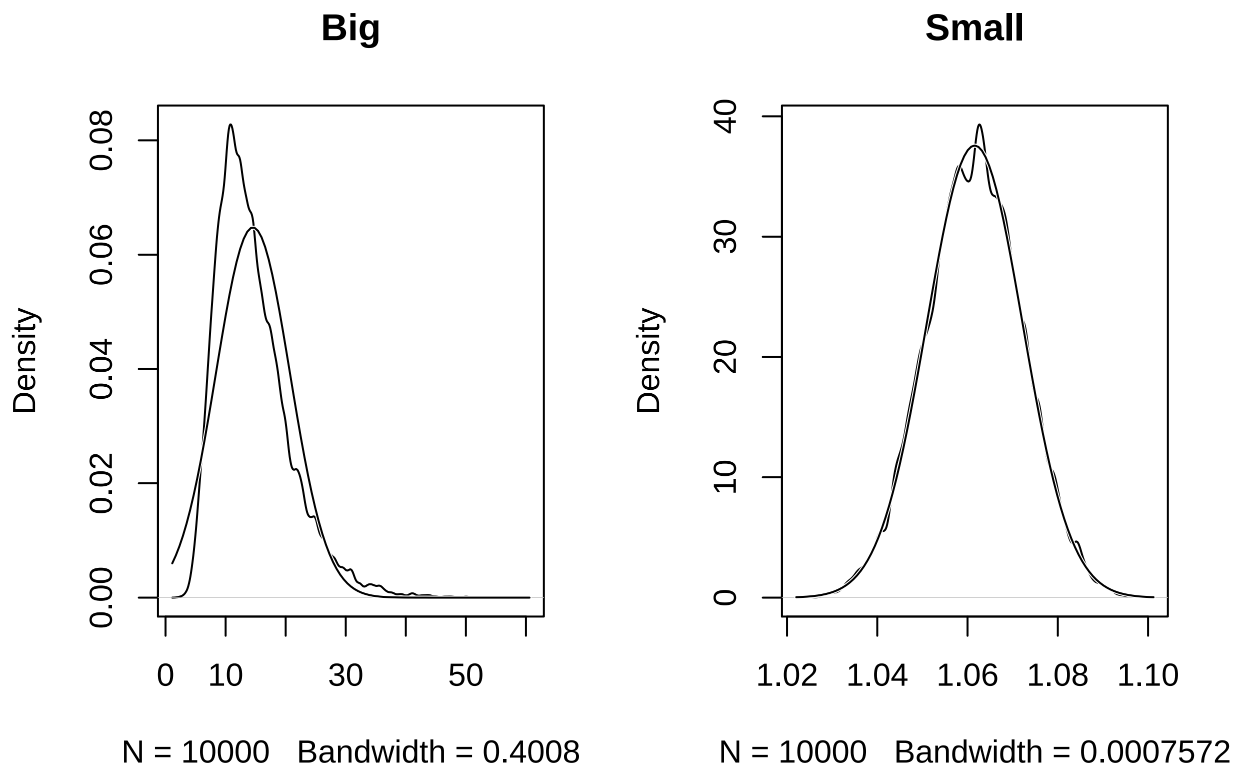


4.1.2. Normal by multiplication

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

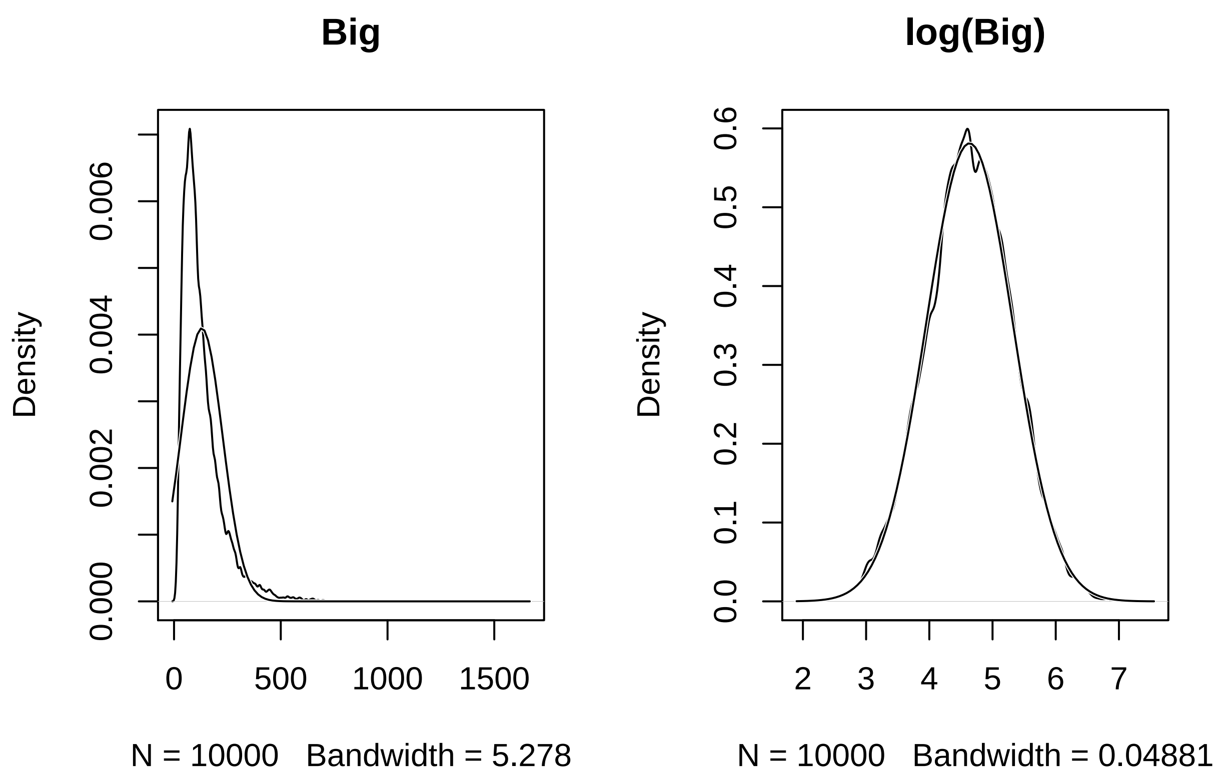



```r
# 4.4
big <- replicate(1e4, prod(1 + runif(12, 0, 0.5)))
small <- replicate(1e4, prod(1 + runif(12, 0, 0.01)))
par(mfrow=c(1, 2))
dens(big, norm.comp = T, main = "Big")
dens(small, norm.comp = T, main = "Small")
```



Normal by log-multiplication

```r
# 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 ...
```

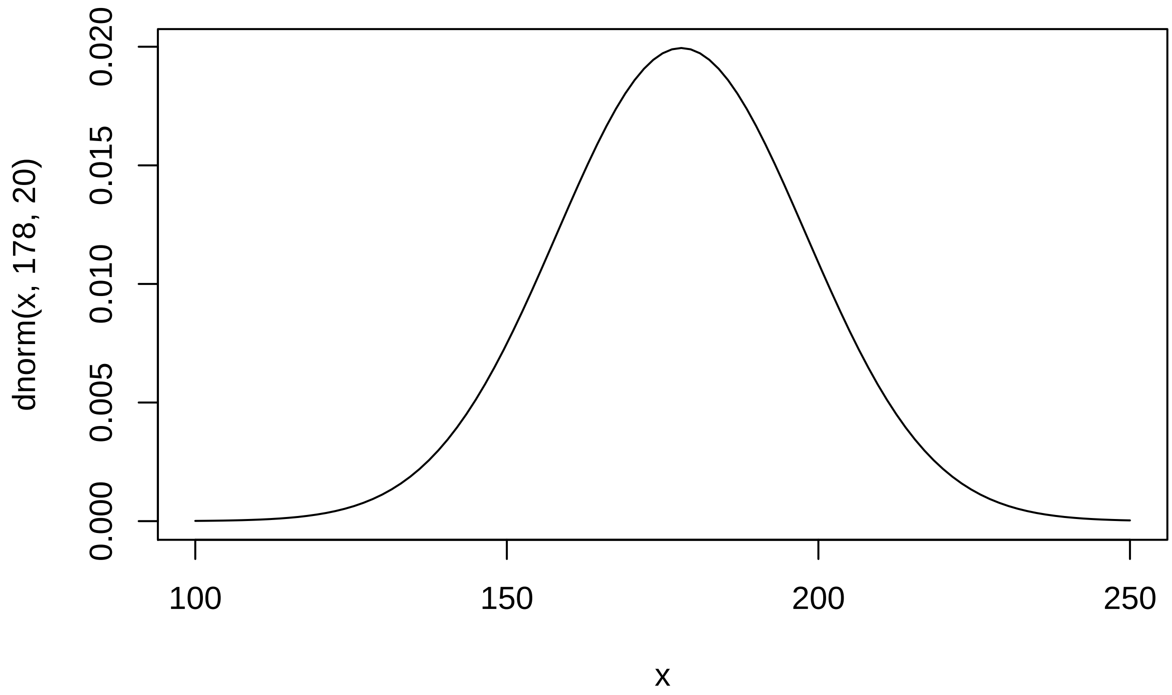
We want heights of adults only (352 rows):

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

4.3.2 The model

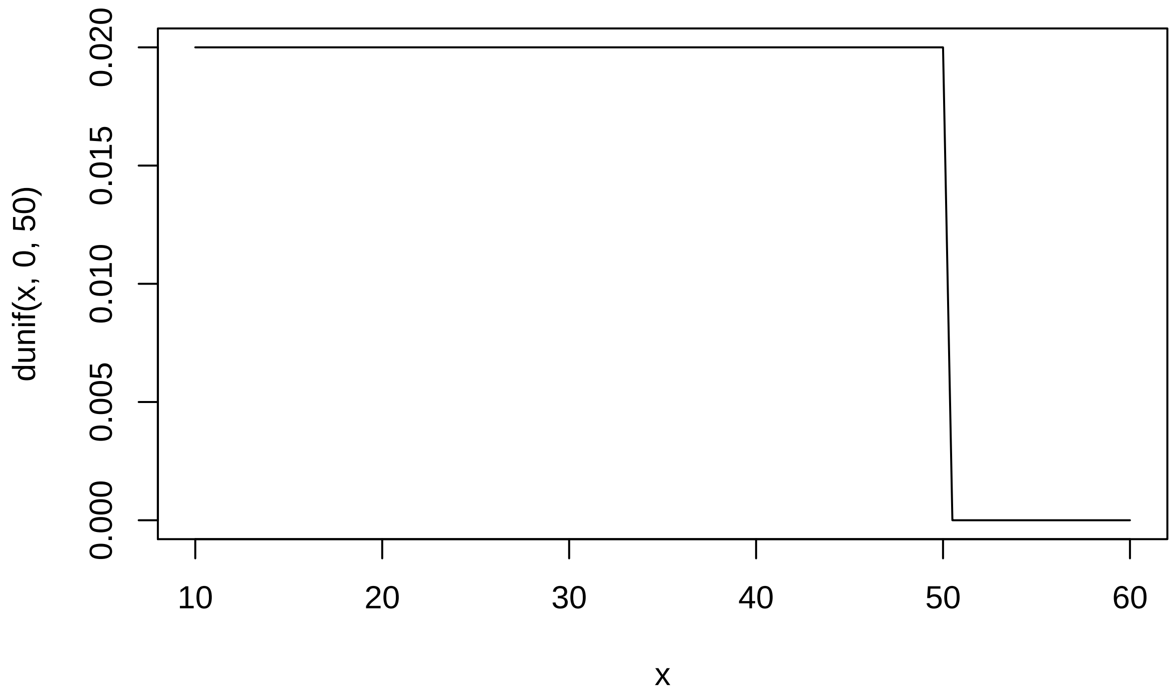
Height mean:

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

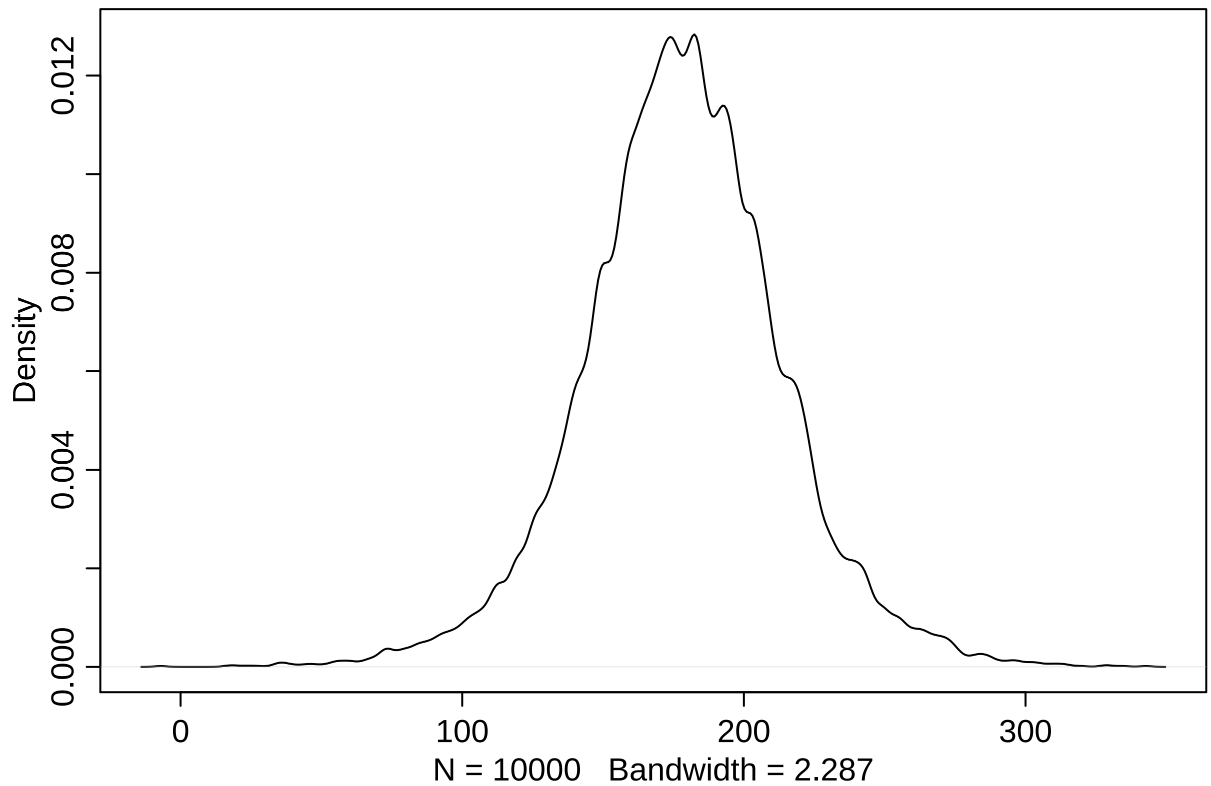


Height standard deviation:

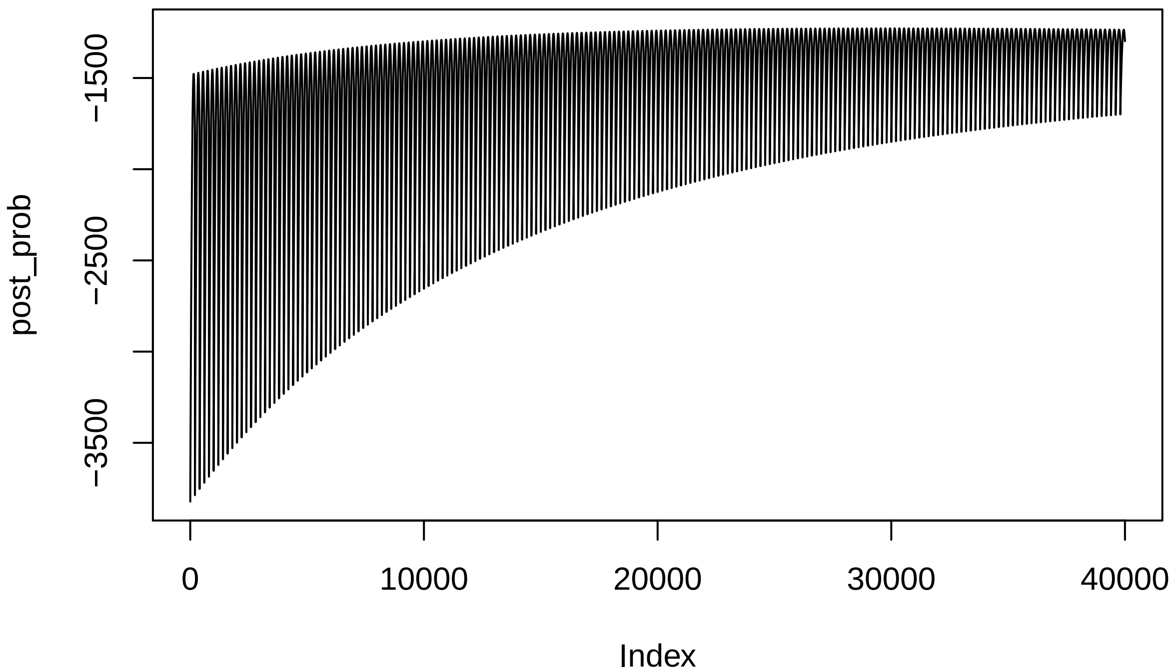
```r
# 4.12
curve(dunif(x, 0, 50), from=10, to=60)
```




```r
# 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)
```

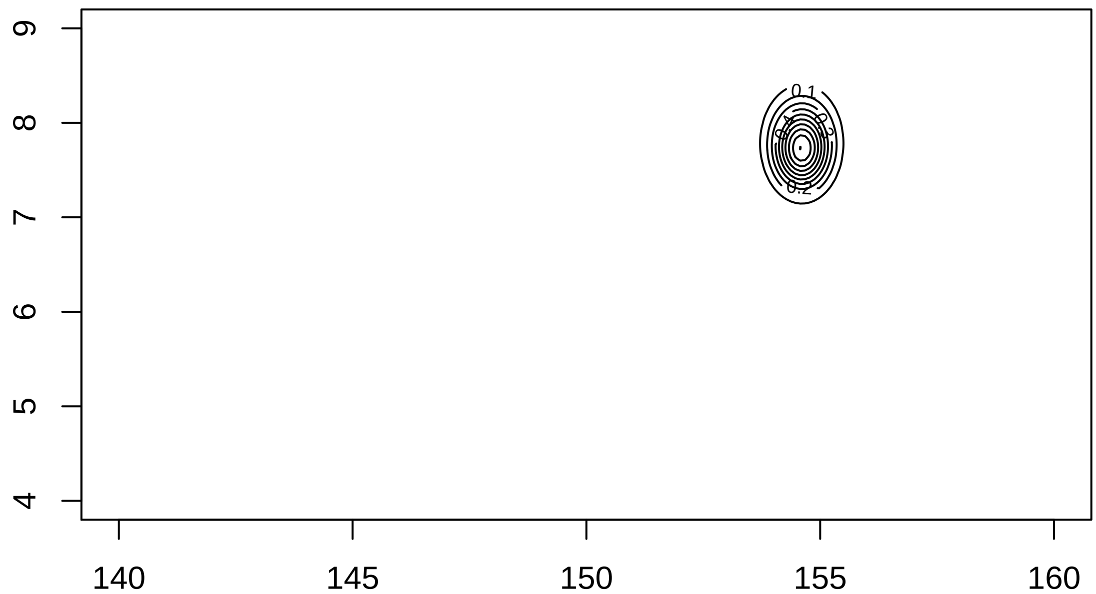



```r
# 4.14
mu_list <- seq(from=140, to=160, length.out=200)
sigma_list <- seq(from=4, to=9, length.out=200)
post <- expand.grid(mu=mu_list, sigma=sigma_list)
post_ll <- sapply(1:nrow(post), function(i) sum(dnorm(
  d2$height,
  mean=post$mu[i],
  sd=post$sigma[i],
  log=T
)))
post_prob <- post_ll +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

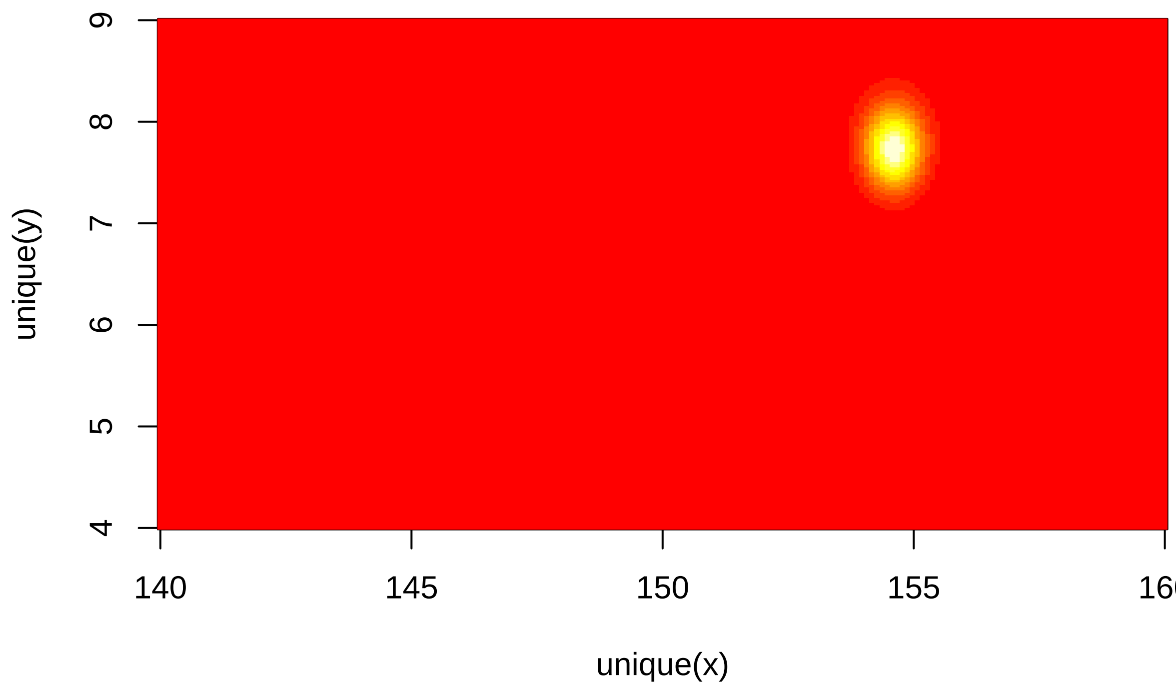



```r
post_prob <- exp(post_prob - max(post_prob))
```

```r
# 4.15
contour_xyz(post$mu, post$sigma, post_prob)
```



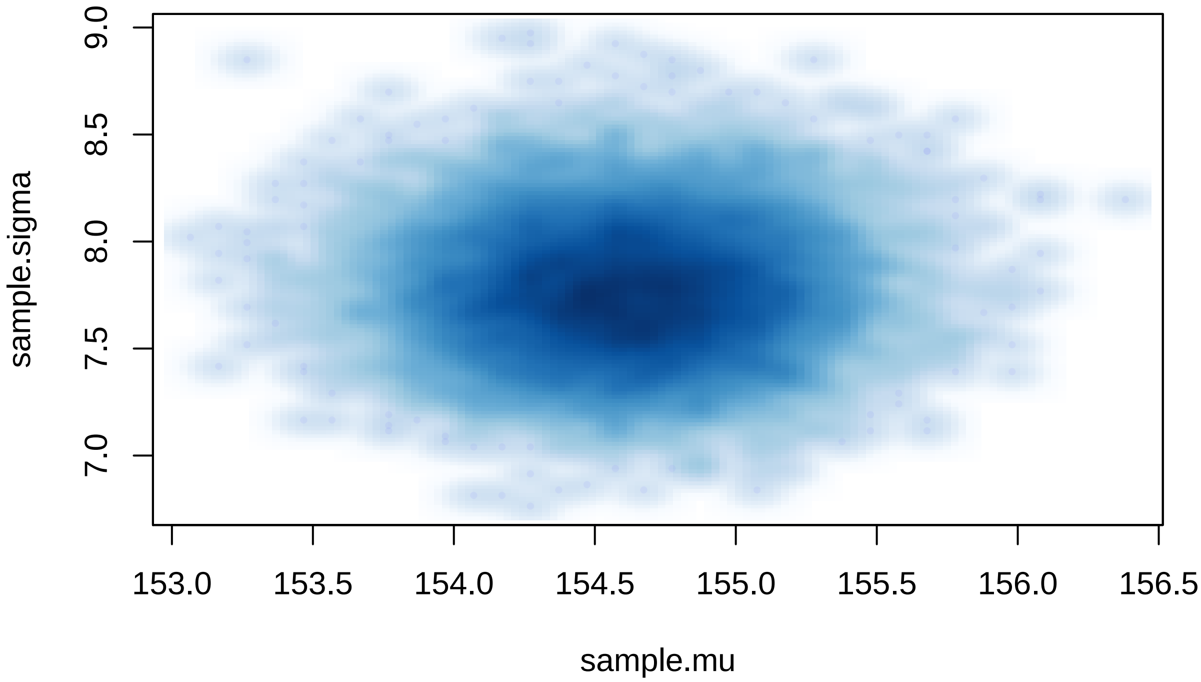

```r
# 4.16
image_xyz(post$mu, post$sigma, post_prob)
```



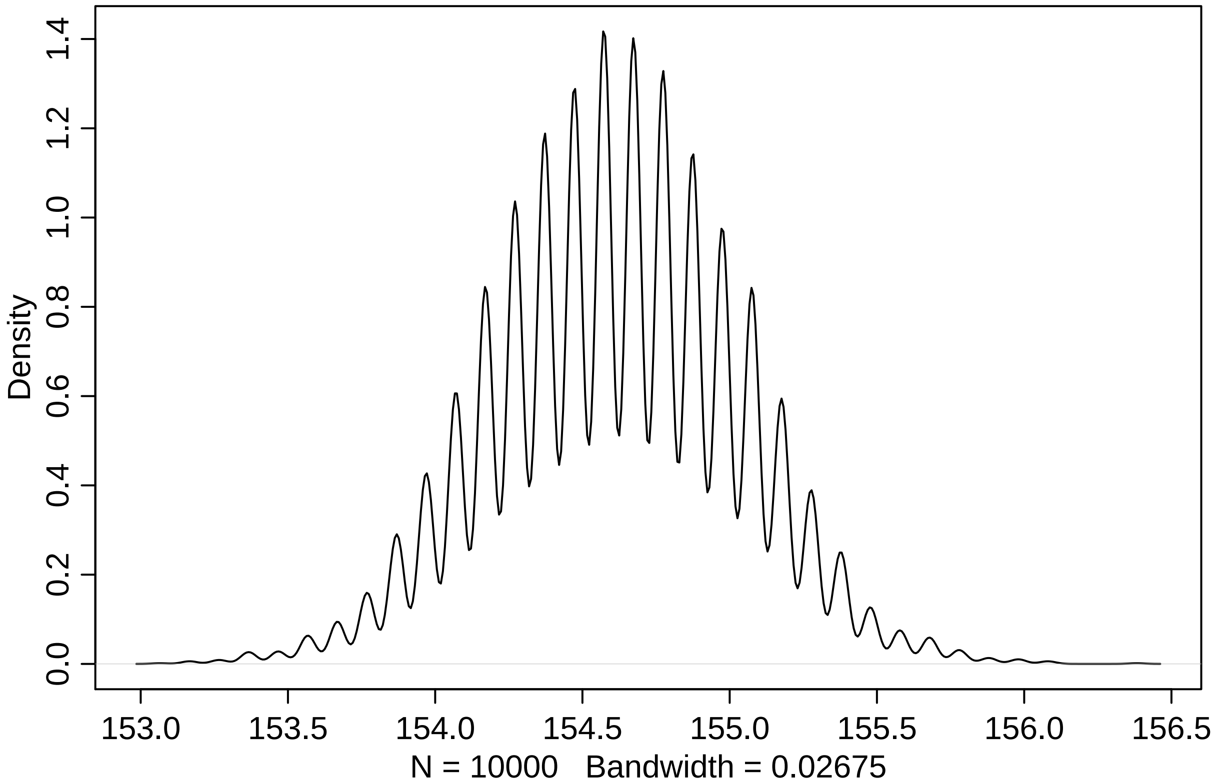
4.3.4 Sampling from the posterior

```r
# 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]
```

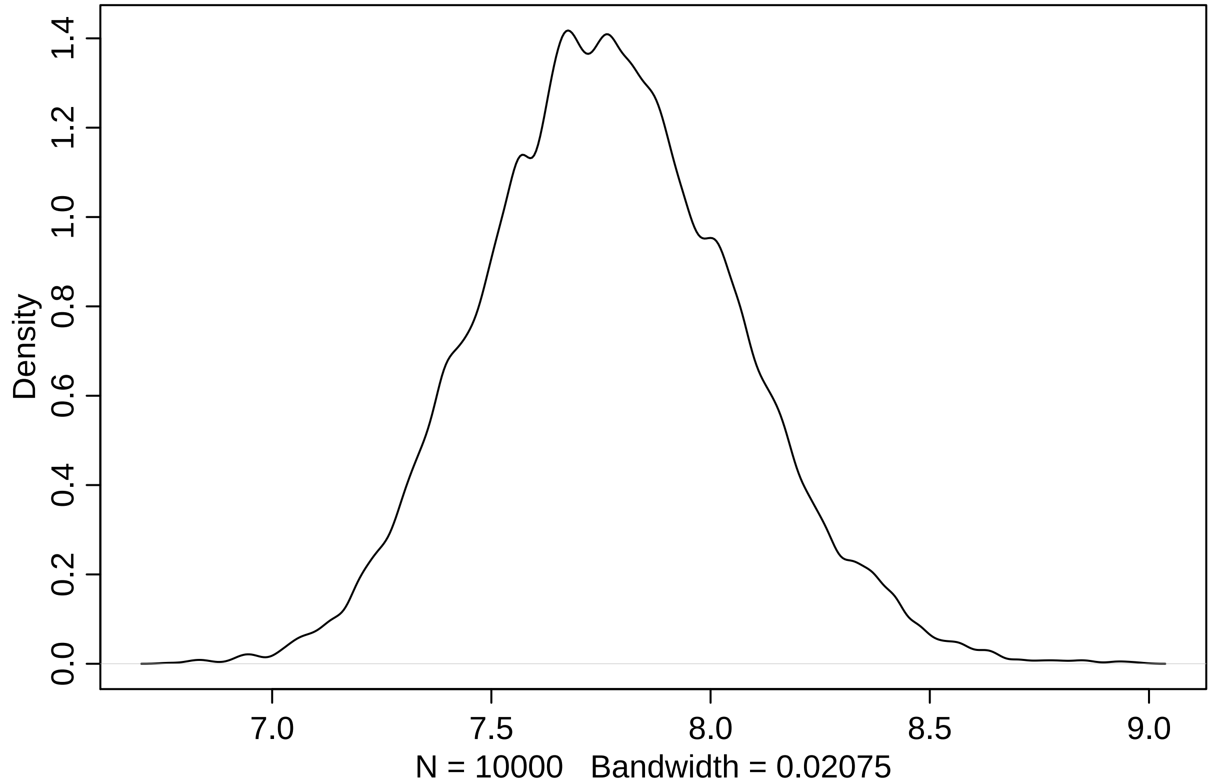
```r
# 4.18
smoothScatter(sample.mu, sample.sigma, cex=0.5, pch=16, col=col.alpha(rangi2, 0.1))
```




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




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




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

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

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

```
##     |0.89    0.89|
## 7.291457 8.195980
```

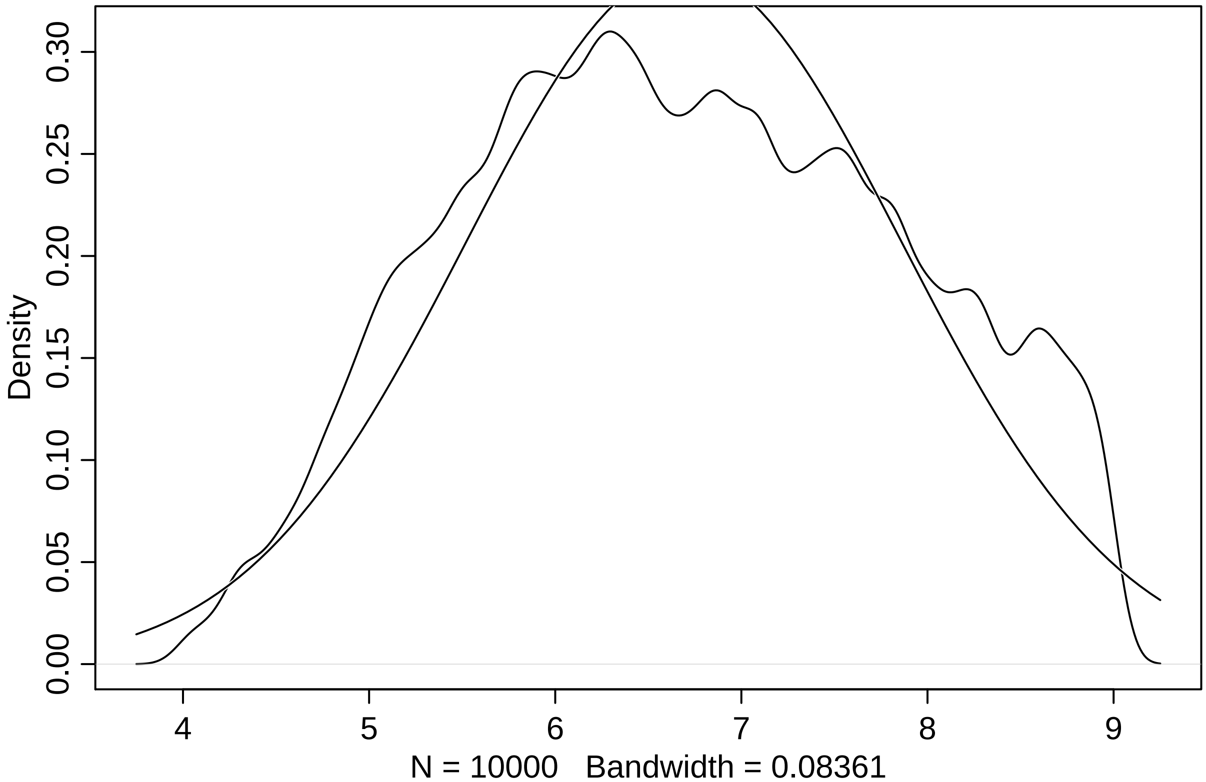
Smaller Sample

To illustrate the posterior is not always Guassian in shape.

```r
# 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]
```

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

```r
# 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.

```r
# 4.25
model.list <- alist(
  height ~ dnorm(mu, sigma),
  mu ~ dnorm(178, 20),
  sigma ~ dunif(0, 50)
)
```

```r
# 4.26
model.solved <- map(model.list, data=d2)
```

```r
# 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.

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

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

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

```
##    |0.89    0.89|
## 7.291457 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.

```r
#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 concerntrated around 178, the machine had to change σ to accomodate the data.

4.3.6. Sampling from a *map* fit.

Variance-covariance matrix:

```r
# 4.30
vcov(model.solved)
```

```
##                 mu        sigma
## mu    0.1697394408 0.0002180701
## sigma 0.0002180701 0.0849056094
```

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

```r
# 4.31
diag(vcov(model.solved))
```

```
##         mu      sigma
## 0.16973944 0.08490561
```

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

```
##                mu       sigma
## mu    1.000000000 0.001816505
## sigma 0.001816505 1.000000000
```

Sampling from the posterior:

```r
# 4.34
coef(model.solved)
```

```
##         mu      sigma
## 154.607025   7.731329
```

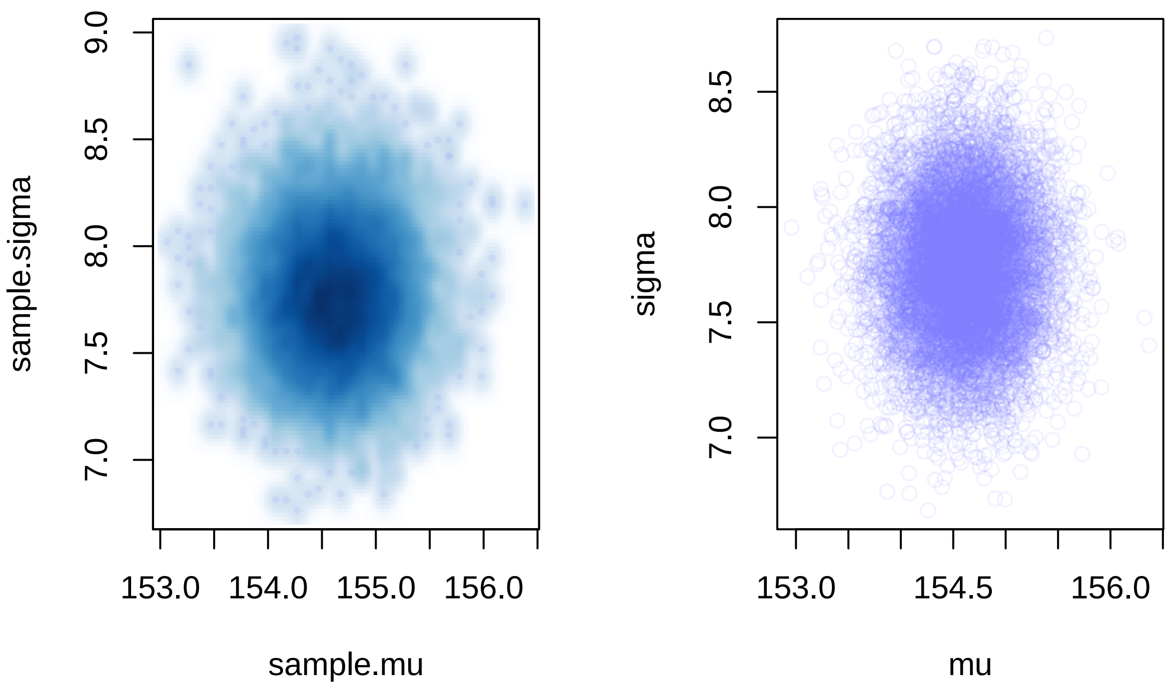
```r
library(MASS)
post <- mvrnorm(n=1e4, mu=coef(model.solved), Sigma=vcov(model.solved))
post = data.frame(post)
head(post)
```

```
##         mu    sigma
## 1 154.5434 7.159419
## 2 154.3728 7.570438
## 3 154.9818 7.540129
## 4 154.8959 7.492039
## 5 153.5992 7.654188
## 6 154.4936 7.357296
```

```r
# 4.33
precis(post)
```

```
##         Mean StdDev  |0.89  0.89|
## mu    154.61   0.41 153.95 155.25
## sigma   7.74   0.29   7.28   8.20
```

```r
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))
```



sample.sigma
9.0
8.5
8.0
7.5
7.0
153.0
154.0
155.0
156.0
sample.mu
sigma
8.5
8.0
7.5
7.0
153.0
154.5
156.0
mu