chapter 06

Chris Parrish

June 20, 2016

Contents

chapter 06

reference: McElreath, Statistical Rethinking, chap 6, pp.165-207

Outline of **deviance** as a measure of model performance:

- 1. joint probability is the right thing to measure (likelihood is joint probability)
- 2. need an information theoretic measure of distance from perfect
  how much is uncertainty reduced by learning an outcome?
  = how much we have learned
  = how much **information** we derive from learning an outcome
  information is reduction in uncertainty derived from learning an outcome
  divergence = additional uncertainty induced by using probabilities from one distribution to describe another distribution
- 3. **deviance** is an approximation of relative distance from perfect accuracy
  average log probability of a model, E log(q_i) estimates relative distance from q to target p
  deviance
  = relative value of E log(q_i)
  = d(q) = -2 sum log(q_i)
- 4. only **deviance out-of-sample** is of interest

```
library(rethinking)
library(ggplot2)
```

entropy

entropy = uncertainty in a probability distribution
= average log probability

```
## R code 6.9
p <- c( 0.3 , 0.7 )
-sum( p*log(p) )
```

```
## [1] 0.6108643
```

uncertainty increases as options multiply

```
information.entropy <- function(p){
  return(-sum( p*log(p) ))
}
```

```
p <- c( 0.15, 0.15, 0.7 )
information.entropy(p)
```

```
## [1] 0.8188085
```

difference in entropy

```
q <- c(0.7, 0.8, 0.15, 0.70)
information.entropy(q)
```

```
## [1] 0.9624278
```

```
information.entropy(p) - information.entropy(q)
```

```
## [1] -0.1436193
```

KL divergence

Note: this presumes as many cases in p as in q

```
dKL <- function(p, q){
  KLdivergence <- sum( p*(log(p) - log(q)) )
  return(KLdivergence)
}
```

```
dKL(p, q)
```

```
## Warning in log(p) - log(q): longer object length is not a multiple of
## shorter object length
```

```
## Warning in p * (log(p) - log(q)): longer object length is not a multiple of
## shorter object length
```

```
## [1] 0.3650816
```

```
p <- c(0.3, 0.7)
q <- c(0.25, 0.75)
dKL(p, q)
```

```
## [1] 0.006401457
```

```
p <- c(0.3, 0.7)
q <- c(0.01, 0.99)
dKL(p, q)
```

```
## [1] 0.777722
```

```
p <- c(0.3, 0.7)
q <- c(0.99, 0.01)
dKL(p, q)
```

```
## [1] 2.61577
```

```
dKL(p, p)
```

```
## [1] 0
```

deviance

deviance is computed by `logLik` in R

```
d <- function(q){
  dev <- -2 * sum(log(q))
}
```

deviance

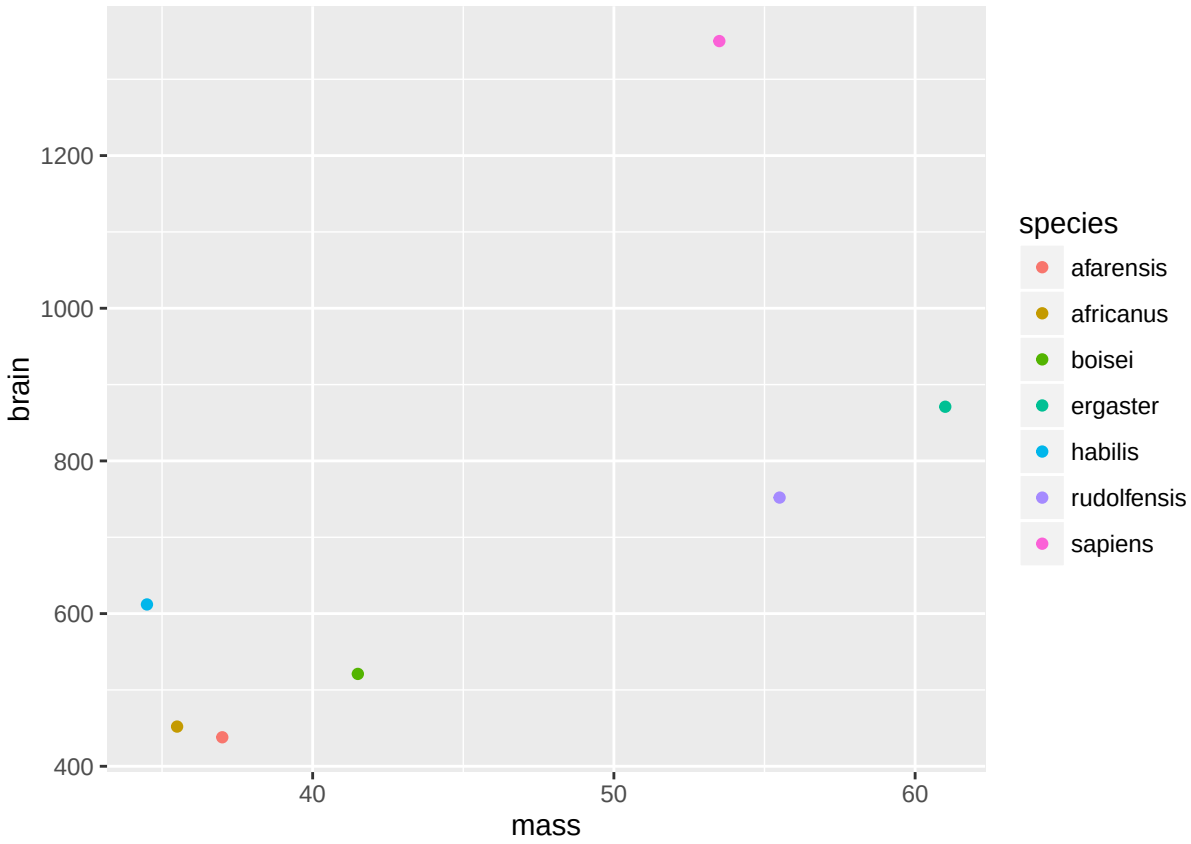
data

```
## R code 6.1
sppnames <- c( "afarensis","africanus","habilis","boisei",
    "rudolfensis","ergaster","sapiens")
brainvolcc <- c( 438 , 452 , 612, 521, 752, 871, 1350 )
masskg <- c( 37.0 , 35.5 , 34.5 , 41.5 , 55.5 , 61.0 , 53.5 )
d <- data.frame( species=sppnames , brain=brainvolcc , mass=masskg )
str(d)
```

```
## 'data.frame':    7 obs. of  3 variables:
##  $ species: Factor w/ 7 levels "afarensis","africanus",..: 1 2 5 3 6 4 7
##  $ brain  : num  438 452 612 521 752 871 1350
##  $ mass   : num  37 35.5 34.5 41.5 55.5 61 53.5
```

exploratory data analysis

```
ggplot(d, aes(x = mass, y = brain, color = species)) +
  geom_point(shape = 19)
```



logLik

```
## R code 6.10
# fit model with lm
m6.1 <- lm( brain ~ mass , d )

# compute deviance by cheating
(-2) * logLik(m6.1)
```

```
## 'log Lik.' 94.92499 (df=3)
```

computing deviance

```
## R code 6.11
# standardize the mass before fitting
d$mass.s <- (d$mass-mean(d$mass))/sd(d$mass)
str(d)
```

```
## 'data.frame':    7 obs. of  4 variables:
##  $ species: Factor w/ 7 levels "afarensis","africanus",..: 1 2 5 3 6 4 7
##  $ brain  : num  438 452 612 521 752 871 1350
##  $ mass   : num  37 35.5 34.5 41.5 55.5 61 53.5
##  $ mass.s : num  -0.779 -0.917 -1.009 -0.367 0.917 ...
```

```
m6.8 <- map(
    alist(
        brain ~ dnorm( mu , sigma ) ,
        mu <- a + b*mass.s
    ) ,
    data=d ,
    start=list(a=mean(d$brain),b=0,sigma=sd(d$brain)) ,
    method="Nelder-Mead" )
```

```
# extract MAP estimates
theta <- coef(m6.8)

# compute deviance
dev <- (-2)*sum( dnorm(
            d$brain ,
            mean=theta[1]+theta[2]*d$mass.s ,
            sd=theta[3] ,
            log=TRUE ) )
dev
```

```
## [1] 94.92499
```

out-of-sample deviance

```
## R code 6.12
N <- 20
kseq <- 1:5
dev <- sapply( kseq , function(k) {
        print(k);
        r <- replicate( 1e2 , sim.train.test( N=N, k=k ) );
        c( mean(r[1,]) , mean(r[2,]) , sd(r[1,]) , sd(r[2,]) )
    } )
```

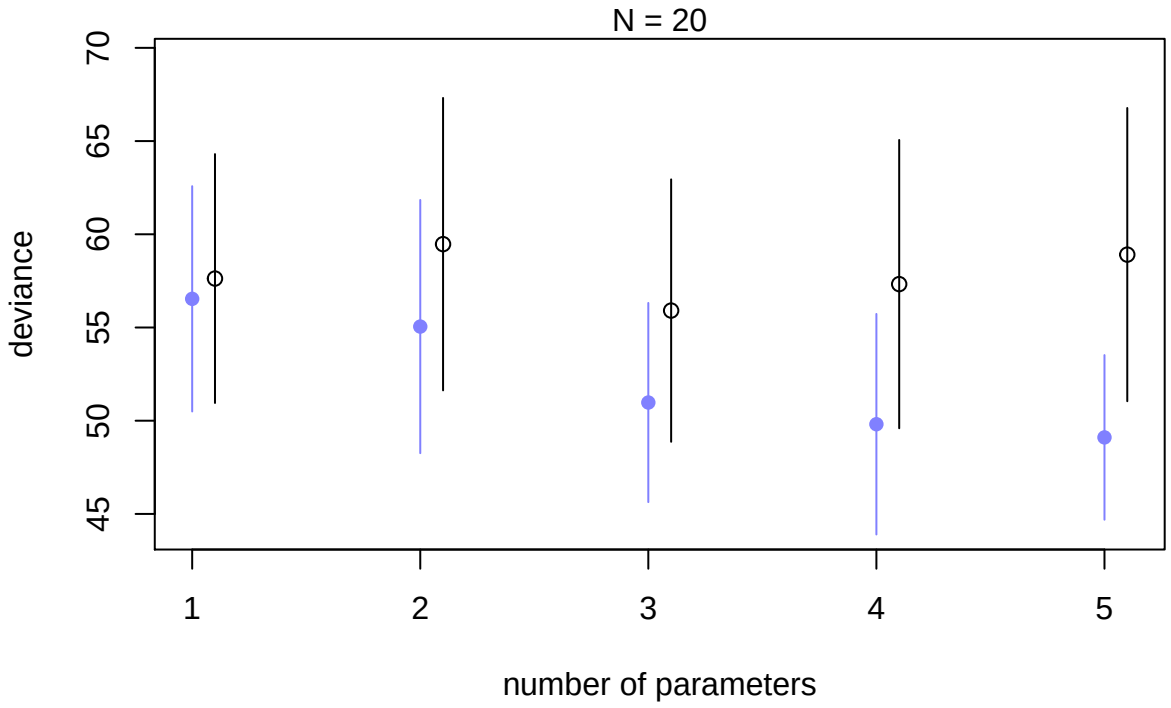
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

plot out-of-sample deviance

```
## R code 6.13
#        r <- mcreplicate( 1e4 , sim.train.test( N=N, k=k ) , mc.cores=4 )

## R code 6.14
plot( 1:5 , dev[1,] , ylim=c( min(dev[1:2,])-5 , max(dev[1:2,])+10 ) ,
    xlim=c(1,5.1) , xlab="number of parameters" , ylab="deviance" ,
    pch=16 , col=rangi2 )
mtext( concat( "N = ",N ) )
points( (1:5)+0.1 , dev[2,] )
for ( i in kseq ) {
    pts_in <- dev[1,i] + c(-1,+1)*dev[3,i]
    pts_out <- dev[2,i] + c(-1,+1)*dev[4,i]
    lines( c(i,i) , pts_in , col=rangi2 )
```

```
    lines( c(i,i)+0.1 , pts_out )
}
```



regularization

information criteria

Start value (1)

```
## R code 6.15
data(cars)
m <- map(
    alist(
        dist ~ dnorm(mu,sigma),
        mu <- a + b*speed,
        a ~ dnorm(0,100),
        b ~ dnorm(0,10),
        sigma ~ dunif(0,30)
    ) , data=cars )
post <- extract.samples(m,n=1000)

## R code 6.16
n_samples <- 1000
ll <- sapply( 1:n_samples ,
    function(s) {
        mu <- post$a[s] + post$b[s]*cars$speed
        dnorm( cars$dist , mu , post$sigma[s] , log=TRUE )
    } )

## R code 6.17
```

```
n_cases <- nrow(cars)
lppd <- sapply( 1:n_cases , function(i) log_sum_exp(ll[i,]) - log(n_samples) )

## R code 6.18
pWAIC <- sapply( 1:n_cases , function(i) var(ll[i,]) )

## R code 6.19
-2*( sum(lppd) - sum(pWAIC) )
```

```
## [1] 421.2314
```

```
## R code 6.20
waic_vec <- -2*( lppd - pWAIC )
sqrt( n_cases*var(waic_vec) )
```

```
## [1] 14.65709
```

using information criteria

model comparison

```
## R code 6.21
data(milk)
d <- milk[ complete.cases(milk) , ]
d$neocortex <- d$neocortex.perc / 100
dim(d)
```

```
## [1] 17  9
```

```
## R code 6.22
a.start <- mean(d$kcal.per.g)
sigma.start <- log(sd(d$kcal.per.g))
m6.11 <- map(
    alist(
        kcal.per.g ~ dnorm( a , exp(log.sigma) )
    ) ,
    data=d , start=list(a=a.start,log.sigma=sigma.start) )
m6.12 <- map(
    alist(
        kcal.per.g ~ dnorm( mu , exp(log.sigma) ) ,
        mu <- a + bn*neocortex
    ) ,
    data=d , start=list(a=a.start,bn=0,log.sigma=sigma.start) )
m6.13 <- map(
    alist(
        kcal.per.g ~ dnorm( mu , exp(log.sigma) ) ,
        mu <- a + bm*log(mass)
    ) ,
    data=d , start=list(a=a.start,bm=0,log.sigma=sigma.start) )
m6.14 <- map(
    alist(
        kcal.per.g ~ dnorm( mu , exp(log.sigma) ) ,
        mu <- a + bn*neocortex + bm*log(mass)
```

```
    ) ,
    data=d , start=list(a=a.start,bn=0,bm=0,log.sigma=sigma.start) )
```

```
## R code 6.23
WAIC( m6.14 )
```

```
## Constructing posterior predictions
```

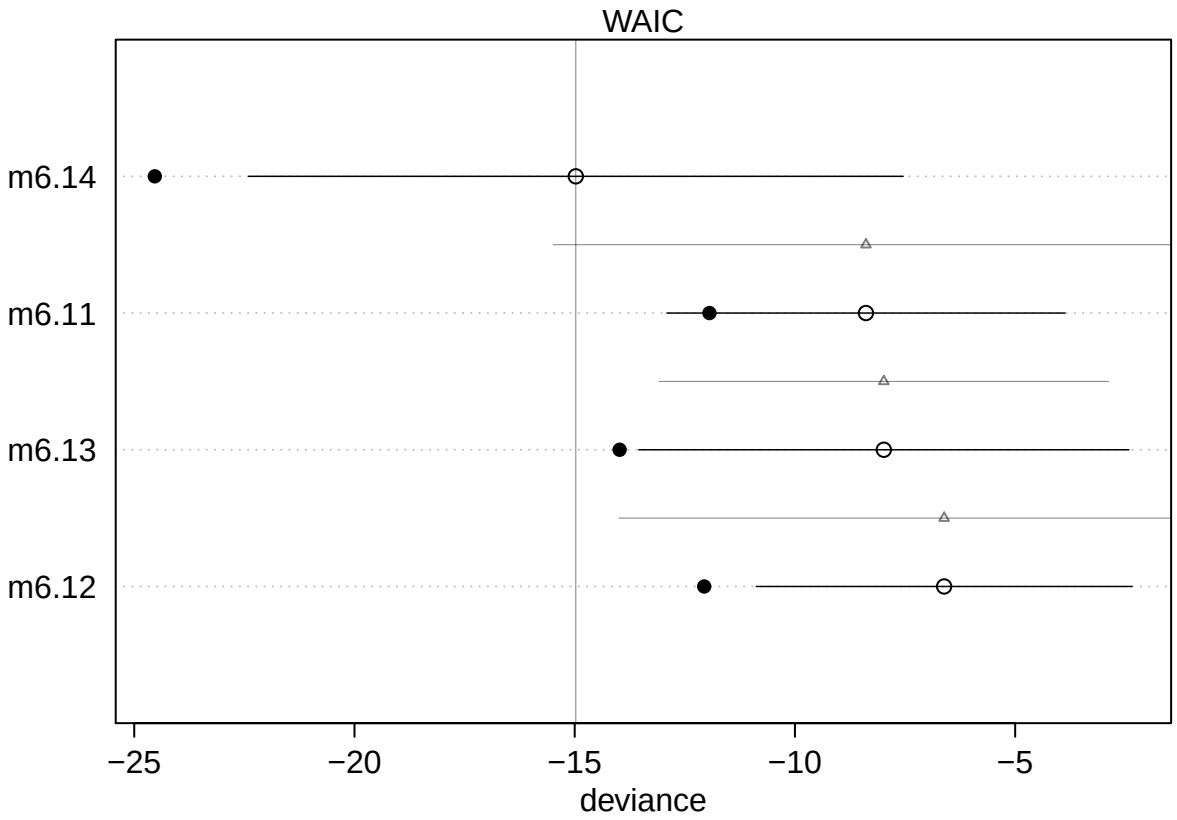
```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
## [1] -14.92737
## attr(,"lppd")
## [1] 12.33825
## attr(,"pWAIC")
## [1] 4.874571
## attr(,"se")
## [1] 7.643988
```

```
## R code 6.24
( milk.models <- compare( m6.11 , m6.12 , m6.13 , m6.14 ) )
```

```
##        WAIC pWAIC dWAIC weight   SE  dSE
## m6.14 -15.0   4.8   0.0   0.92 7.43   NA
## m6.11  -8.4   1.8   6.6   0.03 4.52 7.10
## m6.13  -8.0   3.0   7.0   0.03 5.56 5.11
## m6.12  -6.6   2.7   8.4   0.01 4.27 7.38
```

```
## R code 6.25
plot( milk.models , SE=TRUE , dSE=TRUE )
```



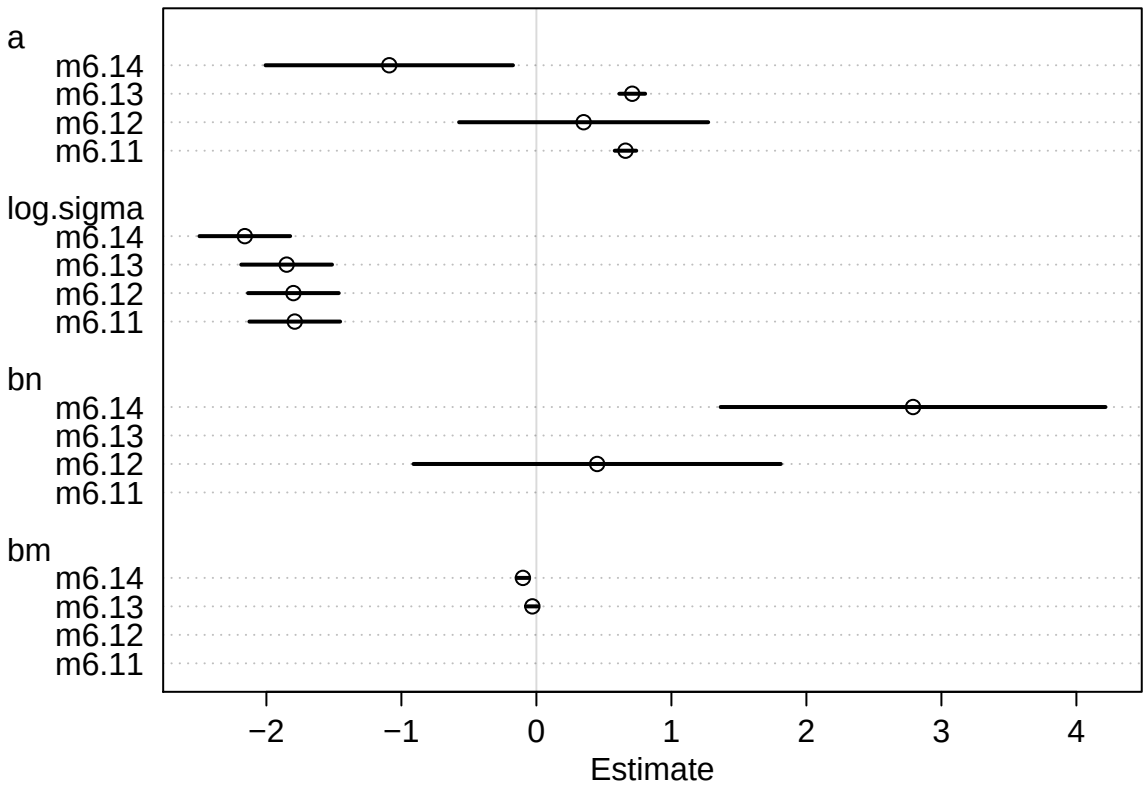

```
## R code 6.26
diff <- rnorm( 1e5 , 6.7 , 7.26 )
sum(diff<0)/1e5
```

```
## [1] 0.17774
```

```
## R code 6.27
coeftab(m6.11,m6.12,m6.13,m6.14)
```

```
##           m6.11   m6.12   m6.13   m6.14
## a            0.66    0.35    0.71   -1.09
## log.sigma   -1.79   -1.80   -1.85   -2.16
## bn             NA    0.45      NA    2.79
## bm             NA      NA   -0.03   -0.10
## nobs           17      17      17      17
```

```
## R code 6.28
plot( coeftab(m6.11,m6.12,m6.13,m6.14) )
```



model averaging

```
## R code 6.29
# compute counterfactual predictions
# neocortex from 0.5 to 0.8
nc.seq <- seq(from=0.5,to=0.8,length.out=30)
d.predict <- list(
    kcal.per.g = rep(0,30), # empty outcome
    neocortex = nc.seq,     # sequence of neocortex
    mass = rep(4.5,30)      # average mass
)
pred.m6.14 <- link( m6.14 , data=d.predict )
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
mu <- apply( pred.m6.14 , 2 , mean )
mu.PI <- apply( pred.m6.14 , 2 , PI )

# plot it all
```

```r
plot( kcal.per.g ~ neocortex , d , col=rangi2 )
lines( nc.seq , mu , lty=2 )
lines( nc.seq , mu.PI[1,] , lty=2 )
lines( nc.seq , mu.PI[2,] , lty=2 )

## R code 6.30
milk.ensemble <- ensemble( m6.11 , m6.12 , m6.13 , m6.14 , data=d.predict )
```

```
## Constructing posterior predictions
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
## Constructing posterior predictions
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
## Constructing posterior predictions
```

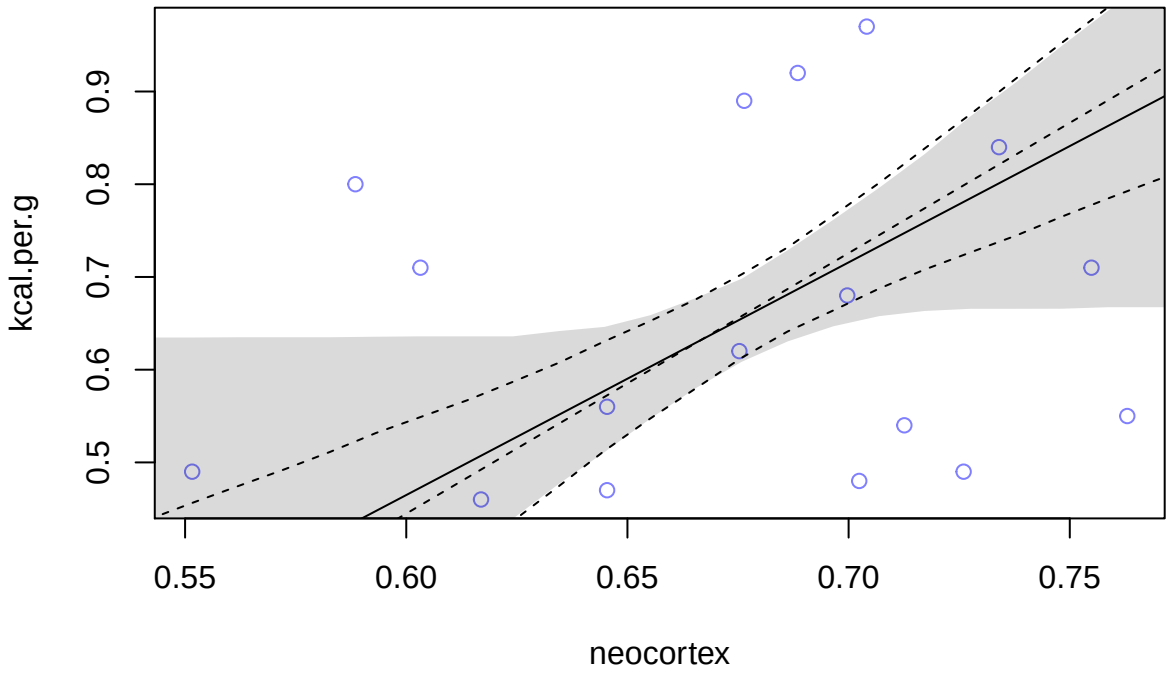
```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
## Constructing posterior predictions
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
```

```
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
mu <- apply( milk.ensemble$link , 2 , mean )
mu.PI <- apply( milk.ensemble$link , 2 , PI )
lines( nc.seq , mu )
shade( mu.PI , nc.seq )
```




```
## R code 6.31
library(rethinking)
data(Howell1)
d <- Howell1
d$age <- (d$age - mean(d$age))/sd(d$age)
set.seed( 1000 )
i <- sample(1:nrow(d),size=nrow(d)/2)
d1 <- d[ i , ]
d2 <- d[ -i , ]

## R code 6.32
sigma <- 1
sum( dnorm( d2$height , mu , sigma , log=TRUE ) )
```

```
## [1] -2688478
```