Lecture 9: Modeling continuous data or why we like the normal distribution

Zachary Marion

2/26/2018

```
library(rstan)
library(shinystan)
library(car)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
source("../utilityFunctions.R")
```

* *This lecture is based on chapter 4 of Statistical Rethinking by Richard McElreath.*

Thus far we have played with simple models and discrete, binomially distributed data. Now we will switch to modeling continuous data using the normal distribution and dive into linear regression.

Why go Gaussian?

Imagine 100 of us go and hang out on the 50 yd line in the Stadium on campus. We all begin flipping coins, and each time it comes up heads we take a step forward; each time we get a tails, we take a step back. Each of us do this for 20 flips and then stop.

Can we predict the proportion of us hanging out at the end on the 50 yd line? what about the south 40 yd line?

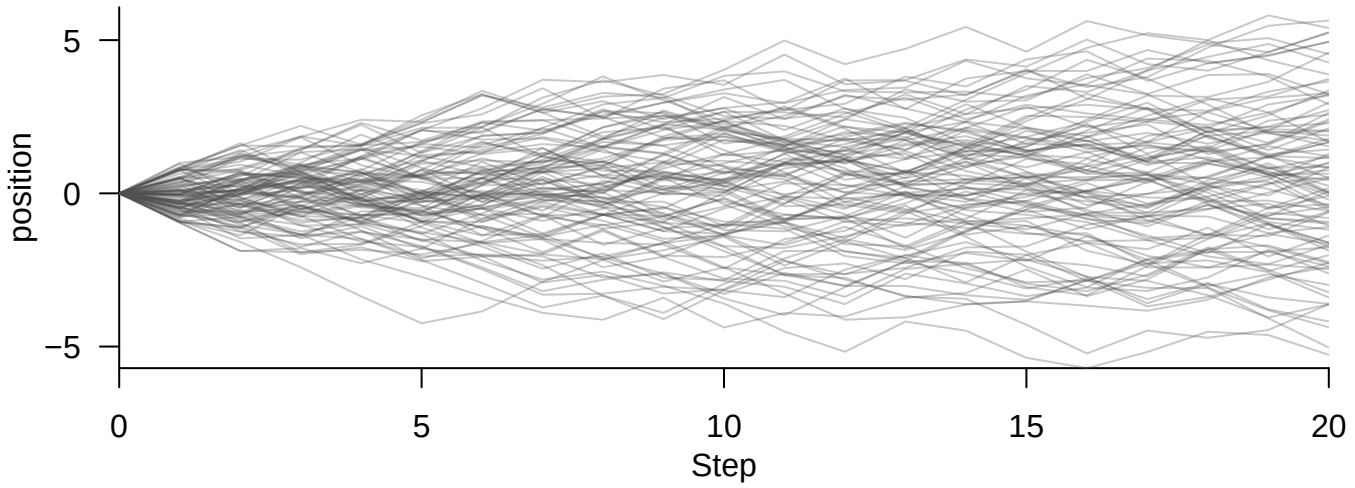
We can simulate this without having to sneak on the field. For each person, we generate a list of steps and then add them up. Because everyone has different gaits, we will use a uniform distribution to generate step sizes.

```
pos <- replicate(20, runif(100,-1, 1)) # simulate positions
cumPos <- t(apply(pos,1,cumsum)) # calculate cumulative position at each step
cumPos <- cbind(rep(0,100), cumPos) # add initial step
```

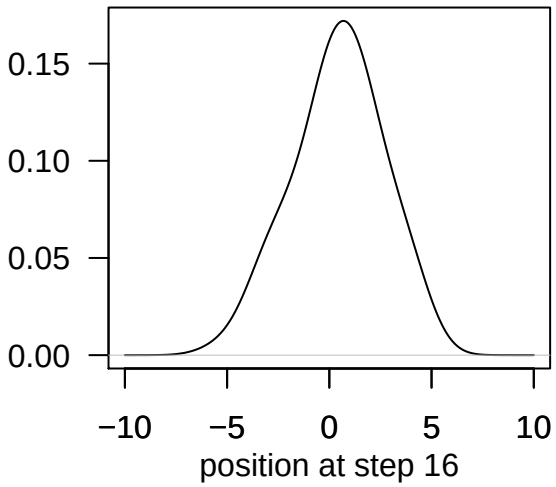
If we plot this out we see that even though we are simulating random walks from a uniform distribution, the familiar Gaussian shape emerges very quickly from the randomness.

```
par(mar=c(3,3,0.1,0.5))
plot(1:100, cumPos[,21], xlim=c(0,20), type="n", las=1, axes=FALSE, xaxs="i")

axis(1, at=seq(0,20, by=5))
axis(2, at=seq(-10,10, by=5), las=1)
mtext(text = "Step", side=1, line = 2)
mtext(text = "position", side=2, line = 2.1)
```

```
for(i in 1:nrow(cumPos)) {
  lines(0:20, cumPos[i,], col="#50505050")
}
```




```
par(mar=c(3,3,0.1,0.5))
plot(density(cumPos[,16],adj=1.5, from=-10, to=10), main="", las=1, xlab="",ylab="")
axis(1, at=seq(-10,10, length=5), las=1)
mtext(text = "position at step 16", side=1, line = 2)
```



Any process that adds together random values from the same distribution (e.g., uniform) converges to a normal given a large enough sample.

- Each sample can be thought of as a deviation from an average value.
- When those deviations are added together, those fluctuations cancel each other out. Large positive fluctuations cancel large negative ones. The more terms in the sum, the more chances for each deviation to be canceled by another, or by a series of smaller deviations in the opposite direction.
- Eventually the most likely way to realize the sums will be one in which every fluctuation is canceled and sum to 0, relative to the mean.

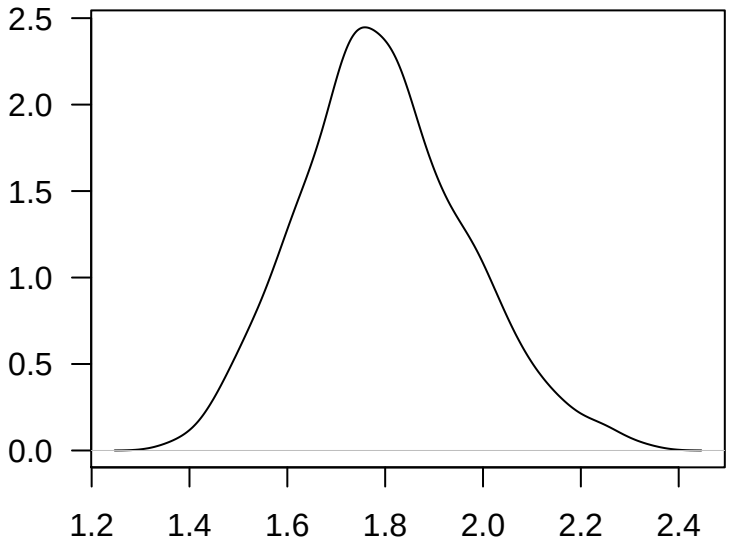
The same thing happens when small effects are multiplied. For example, suppose the growth rate of

an organism is affected by 10 loci, each with small interacting (i.e., multipicative affects). We can sample an individual by:

```
prod(1 + runif(10, 0,0.1))
```

where each of 12 loci has an effect from 1 (no multiplicative effect) to 1.1 (10% increase). If we sample 1,000 indidviduals,

```
par(mar=c(3,3,0.1,0.5))
growth <- replicate(1000, prod(1+runif(12,0,0.1)))
plot(density(growth), main="",las=1)
```

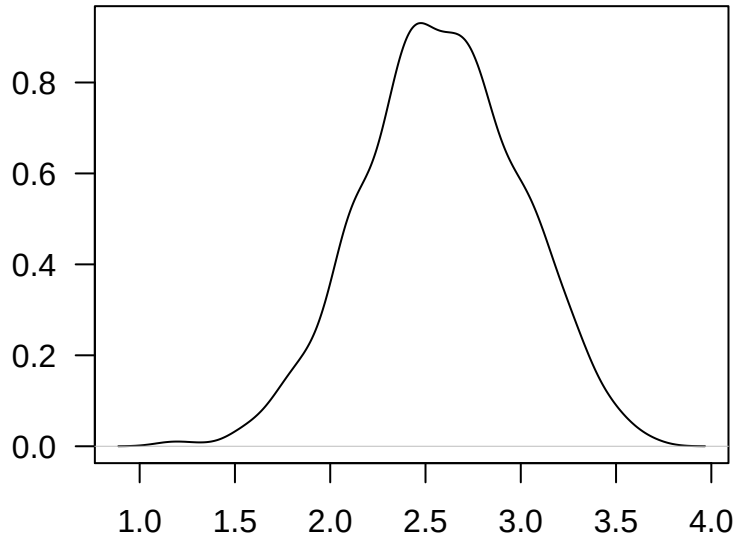


we approximate a bell curve.

This is because the effect at each locus is small. Multiplying small numbers is approximately the same as addition. The smaller the effect, the better the additive approximation will be.

Large deviates that are multiplied together do not produce Gaussian distributions on their original scale. But they do produce Gaussian distributions on the log scale.

```
par(mar=c(3,3,0.1,0.5))
logBig <- replicate(1000, log(prod(1+runif(12,0,0.5))))
plot(density(logBig), main="",las=1)
```



This is because adding logs is equivalent to multiplying the original numbers. And since measurement scales are arbitrary, there is nothing wrong with doing this transformation.

Given the phenomena described above, there are 2 good reasons for using a normal distribution for likelihoods and/or priors:

1. The normal distribution describes widespread patterns such as measurement errors, growth variation, etc.
    - Because the fluctuations of many different processes add together to resemble a normal distribution, we cannot easily identify the underlying process without additional information
    - this doesn't make the normal less useful for modeling.
2. The normal is generally the distribution with maximum entropy (to be elaborated on later).
    - Most natural expression of our ignorance. If all we can do is say there is finite variance, the Gaussian is the shape that realizes this ignorance in the largest number of ways without any additional assumptions.

Modeling the dependent variable with a Gaussian distribution

Data story

Dictyota menstrualis is a brown seaweed that produces > 250 dichyol terpenes as chemical deterrents against marine herbivores and biofouling microbes.

- When herbivore pressure is high, having higher terpene concentrations should lead to higher lifetime biomass. At least that's my story!

I have saved this data as a `.csv` file called `algae.csv`. Also, below is a function, `seaweedSim` with the code to simulate your own data if you want to play around with it.

```r
# You give it the total number of desired observations, the intercept,
# the slope, and the standard deviation and it makes a dataframe for you.
# Play with it, changing parameters as you wish to see how the model
# differs. Also, in the script, terpenes is the x variable. I didn't
# include arguments to change that, but it would be easy by changing
# the mean=50 & sd=3 to whatever you want.

seaweedSim <- function(nObs, alpha, beta, sigma) {
  terpenes <- round(rnorm(nObs, mean=50, sd=3), digits=2)
  error <- rnorm(nObs, mean=0, sd=sigma)
  biomass <- alpha + beta * terpenes + error
  out <- data.frame(terpenes, biomass)
  out <- out[order(terpenes),]
  return(out)
}

set.seed(20)
algae <- seaweedSim(nObs=50, alpha=0, beta=3, sigma=12)
#write.csv(algae, file = "algae.csv", row.names = FALSE)
```

```r
algae <- read.csv("algae.csv")
head(algae)
```

```
  terpenes  biomass
1    41.33 139.9417
2    42.58 111.4138
3    44.42 125.6067
4    45.37 112.4937
5    45.44 150.6760
6    45.58 141.4026
```

```r
summary(algae)
```

```
    terpenes        biomass
 Min.   :41.33   Min.   :111.4
 1st Qu.:47.69   1st Qu.:140.9
 Median :49.45   Median :149.9
 Mean   :49.52   Mean   :150.6
 3rd Qu.:51.70   3rd Qu.:161.6
 Max.   :55.36   Max.   :187.4
```

We will begin with a single measurement variable, `biomass`, to model as a normal distribution. There are two parameters describing the distribution's shape:

1. μ: the mean describing the central location
2. σ: the standard deviation describing the spread

Bayes and MCMC will allow us to explore a number of the most plausible distributions, each with their own μ and σ and rank them by their posterior plausability.

To define our model for biomass as normally distributed with mean μ and standard deviation σ, we need to define a prior $\Pr(\mu, \sigma)$—the *joint prior probability* for the parameters.

For many purposes, priors are specified independently for each parameter (as we have done previously). Thus we assume $\Pr(\mu, \sigma) = \Pr(\mu)\ \Pr(\sigma)$.

The basic model is as follows:

$$\begin{aligned} BM_i &\sim \text{Normal}(\mu, \sigma) \\ \mu &\sim \text{Normal}(150, 30) \\ \sigma &\sim \text{Cauchy}^{+}(0, 10) \end{aligned} \tag{1}$$

I have set the priors as follows:

The prior for μ is weakly informative and centered on the mean of `biomass` with 95% probability the average is between 150 ± 60.

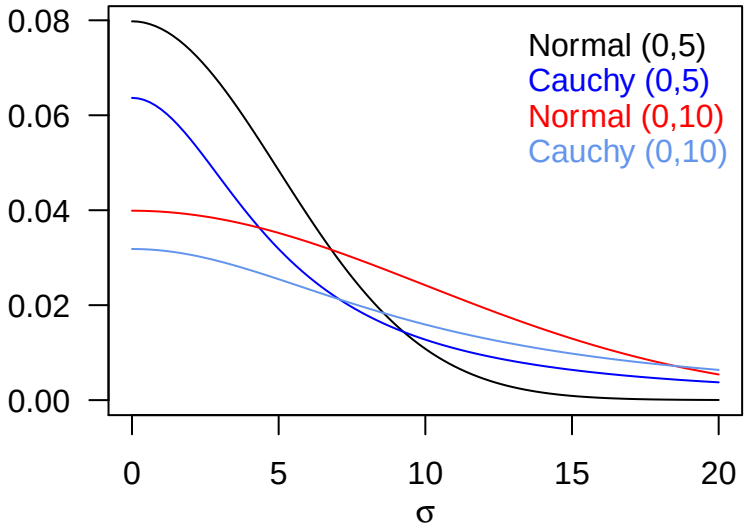
- Later we will play with more restrictive *regularizing priors*

The prior for σ must be positive. A uniform distibution distribution for this usually doesn't sample well from $\text{U}(0, \infty)$, and setting upper bounds on uniforms can cause issues with the MCMC sampler getting stuck against boundaries.

Instead we will use a *half-Cauchy* distribution. This is equivalent to a folded t-distribution with $df = 1$.

- It centers most of the probability mass around zero and therefore credible values, but has fat tails for extremes. You can play with the Cauchy with the `dcauchy` function.

```
# normal(0,5) curve
curve(dnorm(x, 0, 5), from=0, to=20, las=1)
# half-Cauchy(0,5) curve
curve(dcauchy(x, 0, 5), add=TRUE, col="blue")
# normal(0,10) curve
curve(dnorm(x, 0, 10), add=TRUE, col="red")
# half-Cauchy(0,10) curve
curve(dcauchy(x, 0, 10), add=TRUE, col="cornflowerblue")

mtext(text = expression(bold(sigma)), side=1, line = 2)
text(13.5, 0.075, "Normal (0,5)", font=1,cex=1, col="black", adj=c(0, 0.5))
text(13.5, 0.0675, "Cauchy (0,5)", font=1,cex=1, col="blue", adj=c(0, 0.5))
text(13.5, 0.06, "Normal (0,10)", font=1,cex=1, col="red", adj=c(0, 0.5))
text(13.5, 0.0525, "Cauchy (0,10)", font=1,cex=1, col="cornflowerblue",
  adj=c(0, 0.5))
```



We set up our model (`09.modMean.stan`) similarly to how we set up the simple binomial models previously, except that our data are now part of a vector.

```
data {
  int<lower=0> nObs;           // No. obs.
  vector<lower=0>[nObs] BM;    // biomass observations
  real<lower=0> muMean;        // mean of prior mu
  real<lower=0> muSD;          // SD of prior mu
  real<lower=0> sigmaSD;       // scale for sigma
}

parameters {
  real mu;
  real<lower=0> sigma;
}

model {
  mu ~ normal(muMean, muSD);
  sigma ~ cauchy(0, sigmaSD);

  BM ~ normal(mu, sigma);
}
```

Lets set up the data and look at the simplest model first:

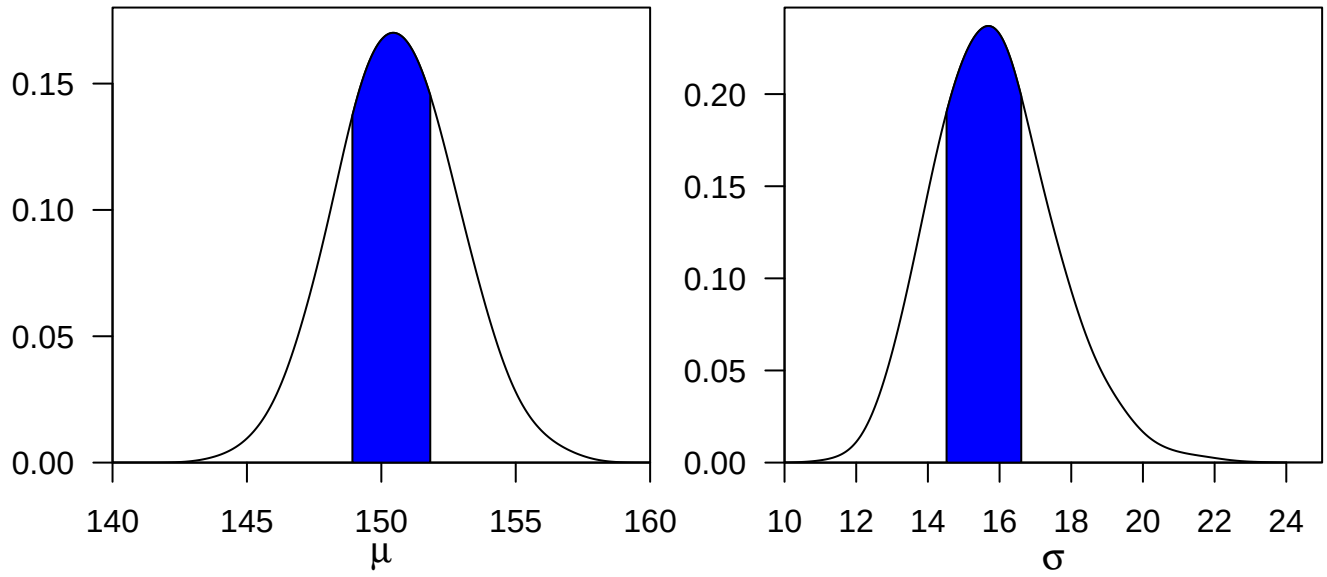
```
dat <- list(nObs=dim(algae)[1], BM=algae$biomass, muMean=150,
            muSD=30, sigmaSD=10)

intMod <- stan(file="09.modMean.stan", data=dat, iter=2000, chains=4, seed=3)

parMod <- as.data.frame(intMod, pars=c("mu", "sigma"))
```

```
print(intMod, pars=c("mu", "sigma"), digits.summary=2)

Inference for Stan model: 09.
```

```
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
mu    150.56    0.04 2.21 146.33 149.08 150.51 152.03 154.98  2663    1
sigma  15.86    0.03 1.66  13.00  14.69  15.76  16.87  19.42  2428    1

Samples were drawn using NUTS(diag_e) at Sat Feb 24 13:18:45 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

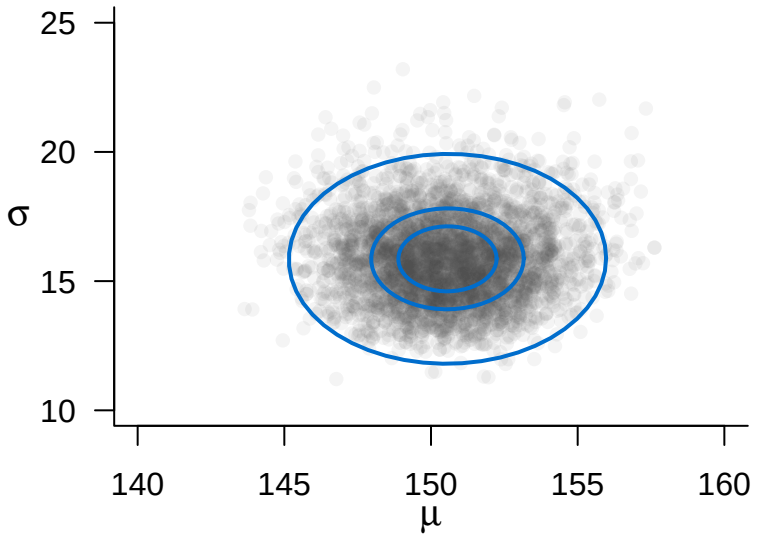
We can plot the marginal densities and 95% HDI's of μ & σ:

```
plotInterval(parMod$mu, HDI=TRUE, credMass=0.95, xlims=c(140, 160),
             col="blue", yOffset=0.01)
mtext(expression(paste(bold(mu))), side=1, line=2, cex=1.2)

plotInterval(parMod$sigma, HDI=TRUE, credMass=0.95, xlims=c(10, 25),
            col="blue", yOffset=0.01)
mtext(expression(paste(bold(sigma))), side=1, line=2, cex=1.2)
```



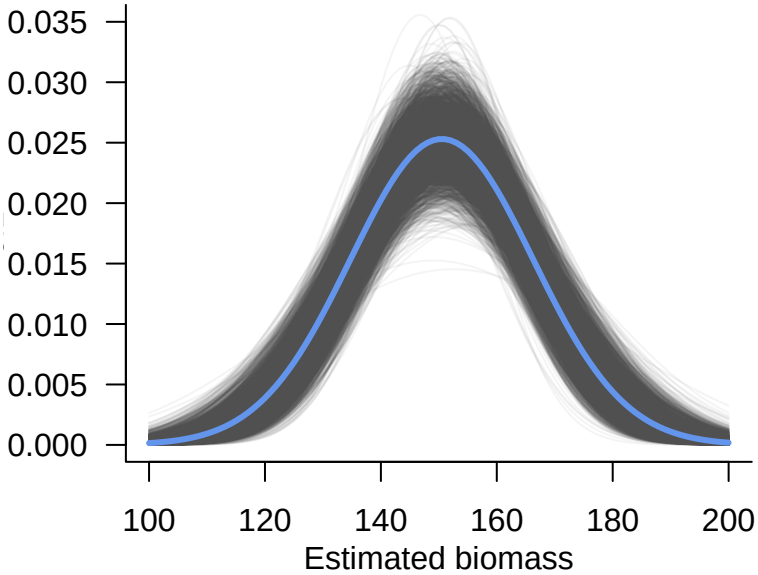
Or plot the joint posterior density $\Pr(\mu, \sigma)$:

```
col <- "#50505010"
plot(parMod, pch=16, col=col, las=1, ylim=c(10,25),
      xlim=c(140,160), bty="l")
dataEllipse(as.matrix(parMod),level=c(0.25,0.5,0.95), add=TRUE, labels=FALSE,
            plot.points=FALSE, center.pch=FALSE, col=c(col,"#006DCC"))
mtext(text = expression(paste(sigma)), side=2, line=2.2, cex=1.2, las=1)
mtext(text = expression(paste(mu)), side=1, line=2, cex=1.2)
```



As before, if we want to estimate the biomass for the *Dictyota* population, we need to consider both the posterior mean and standard deviations.

```
# plot empty plot
plot(0:1,0:1, type="n", xlim=c(100, 200), ylim=c(0,0.035), las=1, bty="l")
mtext(text = "Estimated biomass", side=1, line = 2, cex=1)

# Overlay posterior biomass densities
for (n in 1:nrow(parMod)) {
  curve(dnorm(x, parMod[n,1], parMod[n,2]), add=TRUE, col="#50505010")
}

# Overlay median posterior probability density
medBM <- apply(parMod,2,median)
curve(dnorm(x,medBM[1], medBM[2]), add=TRUE, col="cornflowerblue", lwd=3)
```



For this intercept only model, μ and σ are relatively uncorrelated.

```
cor(parMod)
```

```
               mu       sigma
mu    1.000000000 0.007143987
sigma 0.007143987 1.000000000
```

This can change though once we add a predictor.