Lecture 6: Introduction to hierarchical modeling

As before, we need to load some packages and set some options prior to running any models:

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

Our first hierarchical model

In our examples so far, we have been using bernoulli and binomial likelihoods with beta priors.

- These beta priors have been parameterized in terms of modes (ω) and concentrations (κ).

$$
\begin{aligned}
y &\sim \text{Binomial}(N, \theta) \\
\theta &\sim \text{Beta}(a, b) \\
a &= \omega(\kappa - 2) + 1 \\
b &= (1 - \omega)(\kappa - 2) + 1
\end{aligned}
$$

Under this parameterization, each globe toss's probability of turning up W is described by the parameter θ, which depends on a parameter ω describing the typical proportion of water covering the globe.

- This means that when ω is 0.7, θ will be near 0.7.
- κ tells us how close θ is to ω, with larger values of κ generating $\theta's$ closer to ω
  - κ is prior certainty of dependence of θ on ω

Until now both ω and κ were considered as fixed by prior knowledge (i.e., data). Now we consider ω as a parameter to estimate with it's own prior distribution (we will assum κ is still fixed for now) such as

$$
p(\omega) = \text{Beta}(\omega | a_\omega, b_\omega)
$$

where a_ω & b_ω are constants and we believe ω is typically near the mode

$$\frac{a_\omega - 1}{a_\omega + b_\omega - 2}.$$

Although we parameterized the beta prior in terms of the mode and concentration, both ω and κ were specified as data.

Considering how Bayes's rule applies to this situation,

$$p(\theta, \omega|y) \propto p(y|\theta, \omega)p(\theta, \omega) \tag{1}$$

The likelihood function—$y \sim \text{Binomial}(N, \theta)$—does not involve ω. Therefore, we can rewrite $p(y|\theta, \omega)$ as $p(y|\theta)$. By the definition of conditional probabilities,

$$p(\theta|\omega) = \frac{p(\theta, \omega)}{p(\omega)} \tag{2}$$

we can refactor the joint prior: $p(\theta, \omega) = p(\theta|\omega)p(\omega)$.

Thus the model overall can be rewritten as a hierarchical chain of dependencies

$$\begin{aligned} p(\theta, \omega|y) &\propto p(y|\theta, \omega)p(\theta, \omega) \\ &\propto p(y|\theta)p(\theta|\omega)p(\omega) \end{aligned} \tag{3}$$

- could also be visualized as directed acyclic graph (DAG)

Setting up these models is not very hard. It only requires a few tweaks from the models we have already made.

To show what is going on, I am going to make two stan models, `simpleHier1` and `simpleHierNL1`, that are identical, only `simpleHierNL1` will have the likelihood commented out so that we can see the influence of the prior.

Also, for now we will concentrate on one observation from last time where we had 6 globe tosses and 5 waters.

```
data {
  int<lower=0> nObs;        // Total number of observations
  int<lower=0> N;
  int<lower=0> obs;   // obs as scalar
  real<lower=2> kappa;      // concentration
  real<lower=0> a_omega;    // priors on Omega
  real<lower=0> b_omega;
}
```

```
parameters {
  real<lower=0, upper=1> omega;     // overall prior mode
  real<lower=0, upper=1> theta;     // prob. of water
}

model {
  omega ~ beta(a_omega,b_omega);
  { //  a & b are local parameters and not saved.
    real a;
    real b;
    a = omega * (kappa - 2) +1;
    b = (1 - omega) * (kappa - 2) + 1;
    theta ~ beta(a,b);
  }
  obs ~ binomial(N, theta);
}
```

Low certainty in ω, high κ

We have three parameters to specify now: a_ω, b_ω, and κ. Initially, we will specify weak priors for a_ω & b_ω and strong priors for κ:

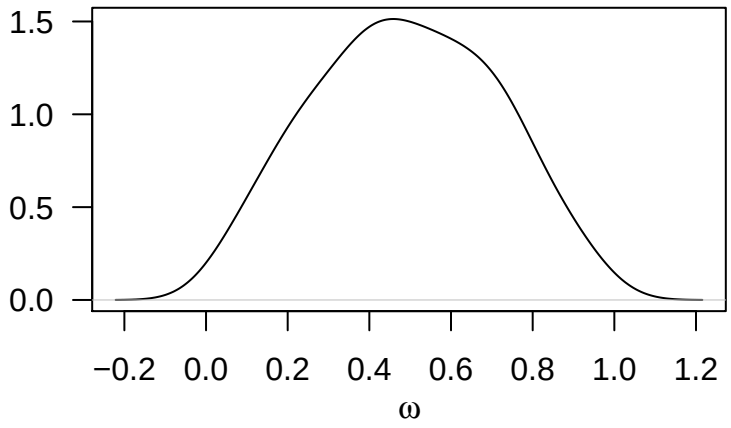
```
a_omega <- 2
b_omega <- 2
kappa <- 100

N <- 6
obs <- 5
nObs <- length(N)
datEx1 <- list(a_omega=a_omega, b_omega=b_omega, kappa=kappa,
               nObs=nObs, N=N, obs=obs)
```

First we will run the model without the likelihood to see the influence of the prior.

```
modExNL1 <- stan(file="simpleHierNL1.stan", data=datEx1,
                 iter=2000, chains=4, seed=3, verbose = FALSE)
parNL <- as.matrix(modExNL1, pars=c("theta","omega"))
```

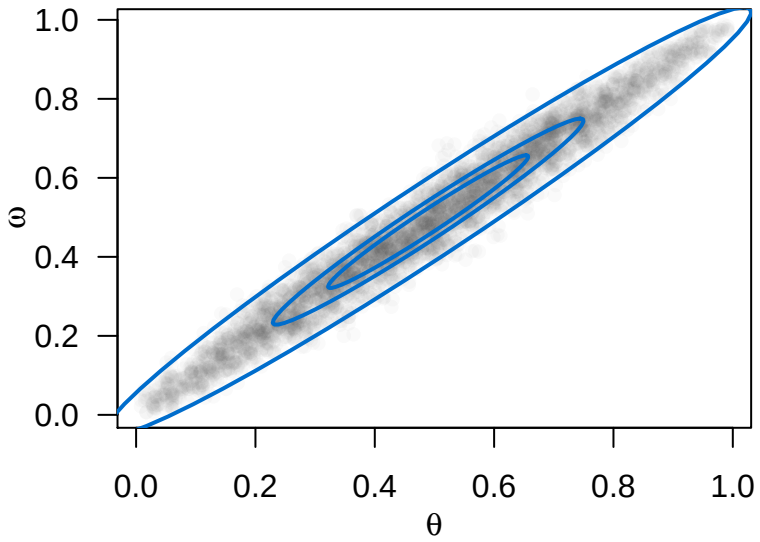
If we look at the marginal posterior probability of ω, we see—not surprisingly given a Beta(2,2) prior—that there is little prior certainty regarding the value of ω.

```
plot(density(parNL[,"omega"], adj=2), main="", xlab="", ylab="",las=1)
mtext(text = expression(paste(omega)), side=1, line = 2)
```



If we plot the joint posterior of ω and θ, however, we see that, by specifying $\kappa = 100$, θ is highly dependent on ω.

```
col <- "#50505008" # specify nice grey color with transparancy
plot(parNL, pch=16, col=col, las=1)
dataEllipse(parNL,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(omega)), side=2, line = 2.2)
mtext(text = expression(paste(theta)), side=1, line = 2)
```

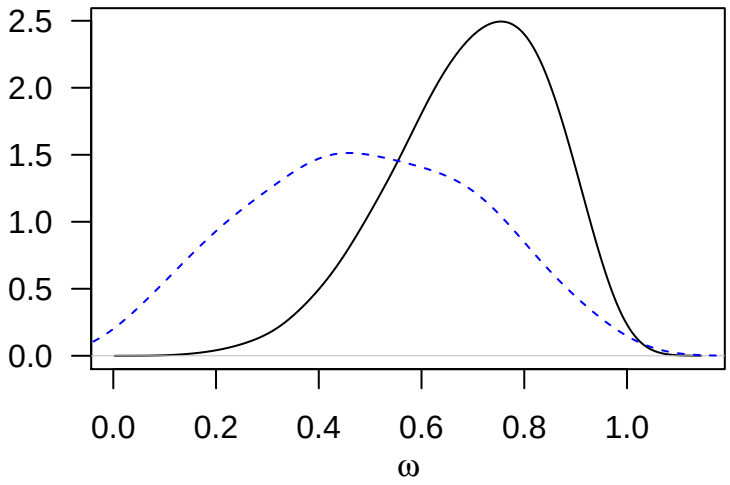


- The ellipses specify the 0.25, 0.5, and 0.95 density contours.

If we run the model with the likelihood included, things look different.

```
modEx1 <- stan(file="simpleHier1.stan", data=datEx1, iter=2000,
               chains=4, seed=3, verbose = FALSE)
parL <- as.matrix(modEx1, pars=c("theta","omega"))
```

```
plot(density(parL[,"omega"], adj=2), main="", xlab="", ylab="",las=1)
lines(density(parNL[,"omega"], adj=2),col="blue",lty=2)
mtext(text = expression(paste(omega)), side=1, line = 2)
```

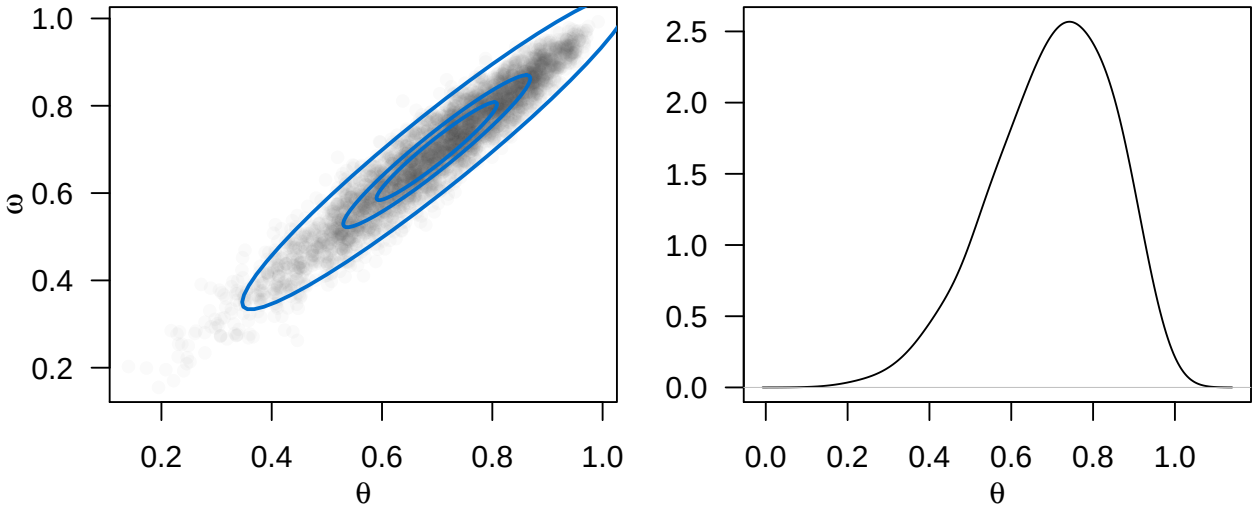


Note that the marginal distribution of $p(\omega|y)$ (black) is now much different than the marginal distribution of the prior $p(\omega)$.

- The low certainty in the value of ω means the data has had a noticable impact on beliefs as to $\omega's$ value.

The high certainty in the dependence of θ on ω means that the joint posterior $p(\theta, \omega)$ has not changed very much in shape. θ has just been shifted along with ω in light of the data.

```
par(mar=c(3,3,0.1,0.5))
par(mfrow=c(1,2))
plot(parL, pch=16, col=col, las=1)
dataEllipse(parL,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(omega)), side=2, line = 2.2)
mtext(text = expression(paste(theta)), side=1, line = 2)

plot(density(parL[,"theta"], adj=2), main="", xlab="", ylab="",las=1)
mtext(text = expression(paste(theta)), side=1, line = 2)
```



High certainty in ω, low κ

In contrast, consider what happens with high certainty in ω & low certainty on the dependence of θ on ω:

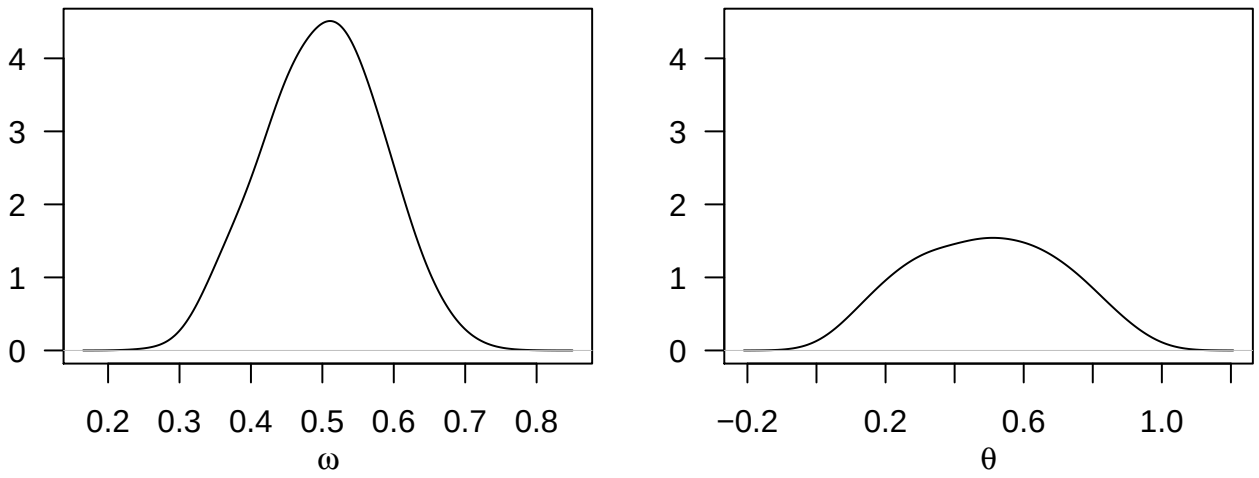
```
a_omega <- 20
b_omega <- 20
kappa <- 5

N <- 6
obs <- 5
datEx2 <- list(a_omega=a_omega, b_omega=b_omega, kappa=kappa,
               N=N, obs=obs)
```

```
modExNL2 <- stan(file="simpleHierNL1.stan", data=datEx2,
  iter=2000, chains=4, seed=3, verbose = FALSE)
parNL2 <- as.matrix(modExNL2, pars=c("theta","omega"))
```

Without the data, the marginal probability of ω is tight with a sharp peak at 0.5.

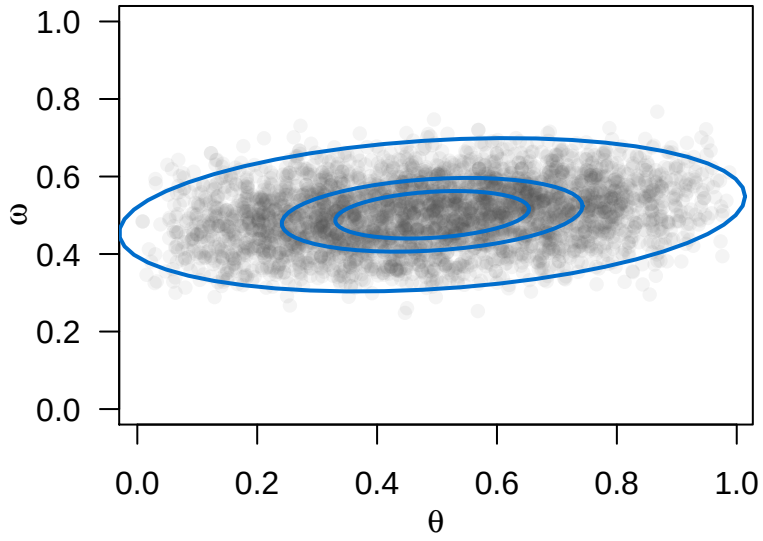
The marginal distribution of θ is fairly broad though because of the low certainty in the dependence of θ on ω.

```
plot(density(parNL2[,"omega"], adj=2), main="", xlab="", ylab="",
     las=1,ylim=c(0,4.5))
mtext(text = expression(paste(omega)), side=1, line = 2)

par(mar=c(3,3,0.1,0.5))
plot(density(parNL2[,"theta"], adj=2), main="", xlab="", ylab="",
     las=1, ylim=c(0,4.5))
mtext(text = expression(paste(theta)), side=1, line = 2)
```



- This is apparent from the joint prior $p(\theta, \omega)$.

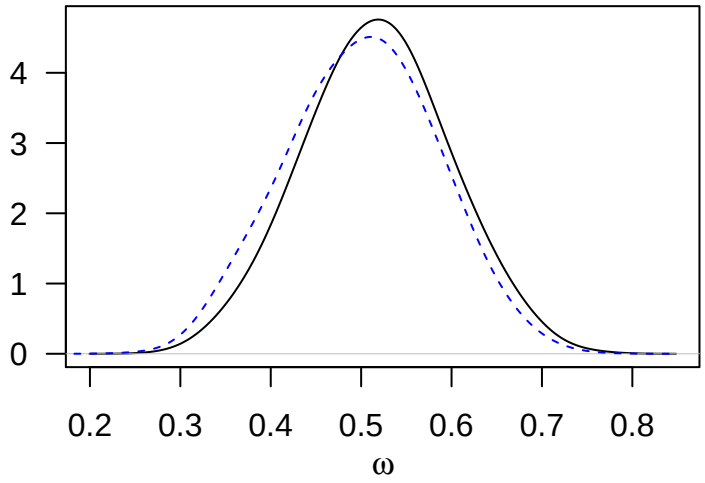
```
plot(parNL2, pch=16, col=col, las=1, ylim=c(0,1))
dataEllipse(parNL2,level=c(0.25,0.5,0.95), add=TRUE, labels=FALSE,
  plot.points=FALSE, center.pch=FALSE, col=c(col,"#006DCC"))
mtext(text = expression(paste(omega)), side=2, line = 2.2)
mtext(text = expression(paste(theta)), side=1, line = 2)
```



Now consider the marginal posterior distribution on ω $p(\omega|y)$ by including likelihood function.

```
modEx2 <- stan(file="simpleHier1.stan", data=datEx2,
  iter=2000, chains=4, seed=3, verbose = FALSE)
parL2 <- as.matrix(modEx2, pars=c("theta","omega"))
```

```
plot(density(parL2[,"omega"], adj=2), main="", xlab="", ylab="",las=1)
lines(density(parNL2[,"omega"], adj=2),col="blue",lty=2)
mtext(text = expression(paste(omega)), side=1, line = 2)
```

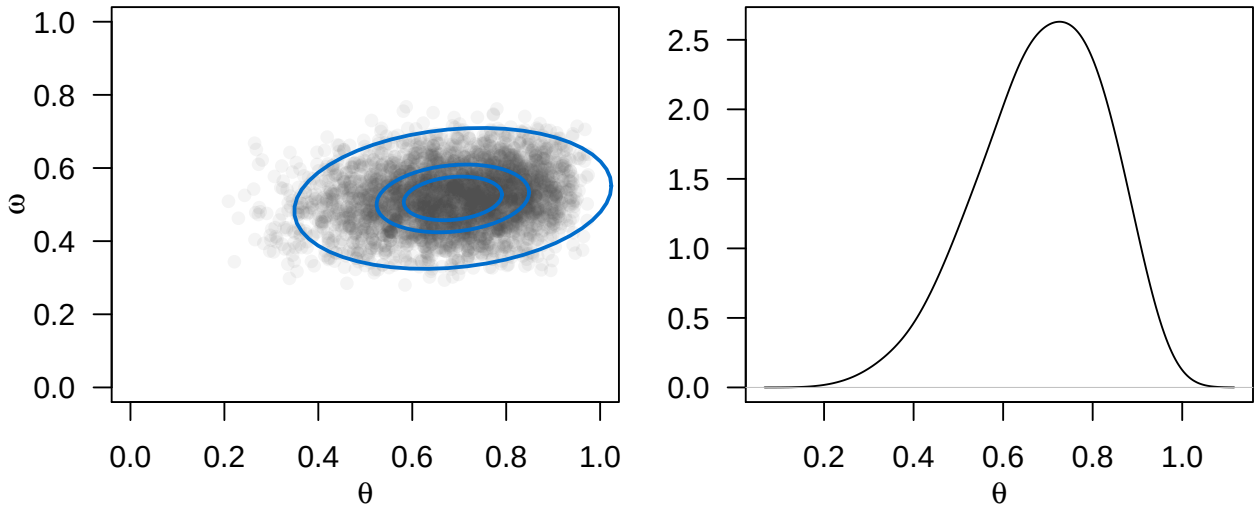


now the marginal distribution of $p(\omega|y)$ (black) is almost identical to the marginal distribution of the prior $p(\omega)$ (blue.

- The high certainty in the value of ω means the data has not affected our beliefs as to $\omega's$ value.

The dependence of θ on ω is quite different now. The low κ means that the posterior of θ is dominated by the data rather than it's dependence on ω.

```
par(mar=c(3,3,0.1,0.5))
par(mfrow=c(1,2))

plot(parL2, pch=16, col=col, las=1, xlim=c(0,1), ylim=c(0,1))
dataEllipse(parL2,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(omega)), side=2, line = 2.2)
mtext(text = expression(paste(theta)), side=1, line = 2)

plot(density(parL2[,"theta"], adj=2), main="", xlab="", ylab="",las=1)
mtext(text = expression(paste(theta)), side=1, line = 2)
```



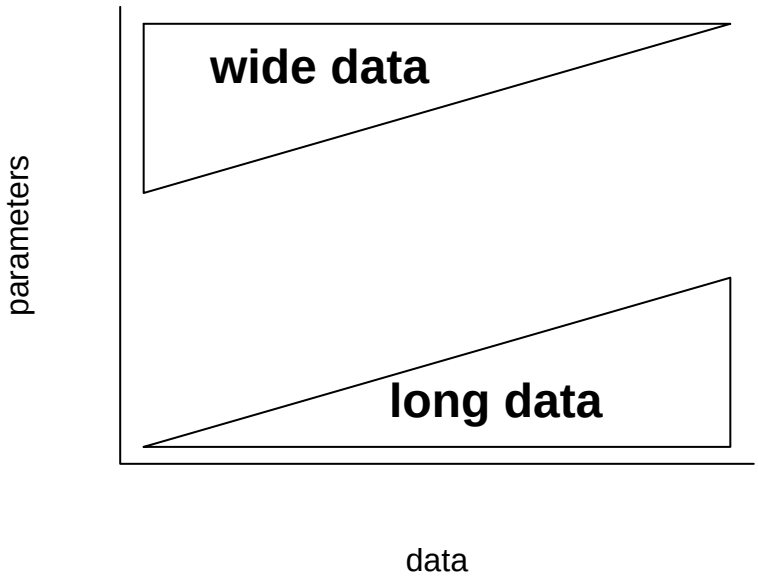
These examples are simple but show the basic concept that hierarchical modeling is Bayesian inference on joint parameter space.

- Shows importance of looking at both the marginal and joint distributions

Practical example of hierarchical models

The previous examples were designed to ease the class into thinking about joint and marginal parameter space, but they don't have much practical utility. So what is the big deal about hierarchical models?

Generally we can place both data and model complexity into a continuum from sparse to dense.



Working with long data isn't a problem, but more often we run into situations where we have more parameters than we do data. Fitting such models can be difficult.

- Want to balance bias (overfitting) and variance (underfitting)

Hierarchical models allow us to model not just individuals but populations through *partial pooling*.

- do this by relaxing *exchangability* (i.e., i.i.d) in the strict sense.
- We can permute any two individuals in the population, and the population still looks the same
  - Not strictly iid but still pretty strong within population

Partial pooling shares information among populations, groups, or clusters with several benefits:

1. **Improved estimates for repeat sampling:** When more than one observation arises from same individual, location, or time, traditional single-level models either over- or underfit.
2. **Improved estimates for sampling imbalance:** When some individuals, locations, or times are sampled more than others, multilevel models implicitly deal with differing uncertainty across groups.
   - Prevents oversampled groups from dominating inference
3. **Estimating variation:** If our questions include variation among individuals or groups, multilevel models are the bee's knees because they explicitly model such variation.
4. **Avoid averaging, retain variation:** Often in traditional analyses groups are collapsed by pre-averaging to make variables. This is naughty because it discards variation, and because there are multiple ways to average.

There are some costs to hierarchical modeling:

1. Hierachical models can be tricky to understand, because estimation occurs at multiple data levels.
2. We have to make more assumptions because now we need to define the distributions from which our population-level parameters arise.

- these distributions should be as biologically intuitive as possible, but...

3. Hierarchical models can be computationally difficult, especially when information or the number of groups is sparse.
   - Scale parameters are especially prone to sampling problems and priors often need to be more informative to keep the MCMC from wandering into regions of high improbability.

As a motivating example, suppose we are interested in the infection prevalance of Ranavirus in Eastern newts (*Notophthalmus viridescens*)—the best animal ever!!!—in eastern Tennessee.

- Newts are censused across $N = 60$ ponds and scored as to whether they are infected or not.

- In some ponds, nary a newt is noticed because the density is low or the pond is hard to sample. Other ponds might be beside themselves with newts (the best ponds).
- Depending on the question, we might be interested in estimating the average infection rate among ponds.
  - We might also be interested in good estimates of within-pond infection rate if conservation efforts need to target the most infected ponds first.

To look at how hierarchical models work, I have simulated data (`newtDat.csv`) for this scenario so that we can check our estimates against the "true" values.

- If you are interested in the simulation, here is the code:

```
omega <- 0.3 # population mode (avg. infection rate across sites)
kappa <- 20  # Concentration (how variable sites are)
nObs <- 60   # total number of sites

a <- omega*(kappa-2) +1       # Beta shape parameter
b <- (1-omega)*(kappa-2) +1   # Beta shape parameter

set.seed(5) # for repeatability
# Total n. of newts sampled at a site. Sites have 5, 10, 25, or 40 newts
N <- as.integer(rep(c(5, 10, 25, 40), each=15))
trueInfect <- rbeta(nObs, a, b)       # true infection prevalance at a site
infect <- rbinom(nObs, N, trueInfect) # simulated number of infected newts
# Site indicator (not really necessary here but included for generality)
site <- 1:nObs

newtDat <- data.frame(site=site, N=N, trueInfect=trueInfect,
                      infect=infect, propInf=infect/N)
```

```
newtDat <- read.csv("newtDat.csv")
head(newtDat)
```

```
  site N trueInfect infect propInf
1    1 5  0.2266825      1     0.2
2    2 5  0.5166810      1     0.2
3    3 5  0.1843030      1     0.2
4    4 5  0.3283834      2     0.4
5    5 5  0.5750651      3     0.6
6    6 5  0.2519853      2     0.4
```

We have repeated measures and heterogeneity. There is a lot of variation in the data across sites (because I simulated it thus) due to unmeasured factors affecting ranavirus prevalance. Thus site is a grouping variable with multiple observations (newts) within each site.

- Ignoring sites by estimating one infection prevalance across all ponds means we ignore important variation and may underfit (*complete pooling model*).

- Estimating an infection probability for each site separately (*no pooling model*) may introduce bias by overfitting noise and ignoring that sites may not be spatially independent of each other—newts migrate.
- A *partial pooling model* compromises with an adaptive prior that learns from all site-level infection estimates.

Complete pooling

We are going to create all three models. We will start with the simplest, a complete pooling model that estimates one θ across all sites.

This model, `completePooling.stan`, is specified as:

$$\begin{aligned} infected &\sim \text{Binomial}(N, \theta) \\ \theta &\sim \text{Beta}(\alpha, \beta). \end{aligned} \tag{4}$$

where θ is the infection probability for all sites and α and β are constants. This should be simple to code in stan by now.

[Draw out as diagram]

```
data {
  int<lower=0> nObs;          // No. obs.
  int<lower=0> N[nObs];       // No. sampled newts
  int<lower=0> obs[nObs];     // No. infected newts
  real<lower=0> alpha;        // priors on theta
  real<lower=0> beta;         // priors on theta
}

parameters {
  real<lower=0, upper=1> theta; // grand infect prob.
}

model {
  theta ~ beta(alpha, beta);  // prior for thetas
  obs ~ binomial(N, theta);
}
```

No pooling

The second model we will build is a no-pooling model that estimates separate $\theta's$ for each site without cross-talk among sites. This model will be called `noPooling.stan` where

$$
\begin{aligned}
infected &\sim \text{Binomial}(N, \theta_i) \\
\theta_i &\sim \text{Beta}(\alpha, \beta)
\end{aligned} \tag{5}
$$

- Each site $theta_i$ will be given a weak $\text{Beta}(2, 2)$ prior.

[Draw out as diagram]

```
  int<lower=0> nObs;          // No. obs.
  int<lower=0> nSites;        // No. groupings (redundant here)
  int<lower=0> site[nSites];  // Indicator values for groupings
  int<lower=0> N[nObs];       // No. sampled newts
  int<lower=0> obs[nObs];     // No. infected newts
  real<lower=0> alpha;        // priors on thetas
  real<lower=0> beta;         // priors on thetas
}

parameters {
  vector<lower=0, upper=1>[nSites] theta; // w/in-site infect prob.
}

model {
  theta ~ beta(alpha, beta);      // prior for thetas

  // lik. (loop not necessary here but used to show indexing)
  for(n in 1:nObs)
  obs[n] ~ binomial(N[n], theta[site[n]]);
}
```

Partial pooling

Finally, we will build a partial pooling model, `partialPooling.stan`, as specified below:

$$
\begin{aligned}
infected &\sim \text{Binomial}(N, \theta_i) \\
\theta_i &\sim \text{Beta}(a, b) \\
a &= \omega(\kappa - 2) + 1 \\
b &= (1 - \omega)(\kappa - 2) + 1 \\
\omega &\sim \text{Beta}(\alpha_\omega, \beta_\omega) \\
\kappa &\sim \text{Normal}^+(2, \sigma_\kappa)
\end{aligned} \tag{6}
$$

where θ_i is the infection probability at the ith site and $\alpha_\omega, \beta_\omega$, and σ_κ are constants.

- Here ω will be given a weak Beta$(2, 2)$ prior
- Because κ must be positive and > 2, it has a folded normal distribution and will be given a regularizing prior of $\sigma = 2.5$.

[Draw out as diagram]

We will code the model as follows:

```
data {
  int<lower=0> nObs;          // No. obs.
  int<lower=0> nSites;        // No. groupings (redundant here)
  int<lower=0> site[nSites];  // Indicator values for groupings
  int<lower=0> N[nObs];       // No. sampled newts
  int<lower=0> obs[nObs];     // No. infected newts
  real<lower=0> alpha;        // priors on Omega
  real<lower=0> beta;         // priors on Omega
  real<lower=0> sigma;        // prior on kappa scale
}

parameters {
  real<lower=0, upper=1> omega;     // avg. amg-site infect prob.
  real<lower=2> kappa;              // similarity of sites
  vector<lower=0, upper=1>[nSites] theta; // w/in-site infect prob.
}

transformed parameters {
  real<lower=0> a;
  real<lower=0> b;
    a = omega * (kappa - 2) +1;
    b = (1 - omega) * (kappa - 2) + 1;
}

model {
  omega ~ beta(alpha,beta);               // prior on omega
  kappa ~ normal(2, sigma);               // prior on kappa
  theta ~ beta(a,b);                      // prior for thetas

  // lik. (loop not necessary here but used to show indexing)
  for(n in 1:nObs)
  obs[n] ~ binomial(N[n], theta[site[n]]);
}
```

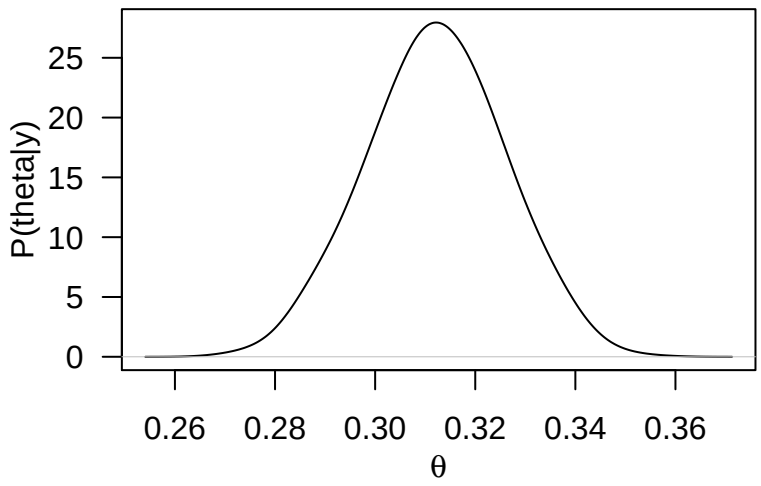
Lets set up the data and look at the simplest model first:

```
# set up the data
nObs    <- nrow(newtDat)
```

```
nSites  <- max(newtDat$site)
site    <- newtDat$site
N       <- newtDat$N
obs     <- newtDat$infect
alpha   <- 2                  # weak beta
beta    <- 2                  # priors
sigma   <- 2.5                # regularizing scale prior

dat <- list(nObs=nObs, nSites=nSites, site=site, N=N, obs=obs,
            alpha=alpha, beta=beta, sigma=sigma)

modCP <- stan(file="completePooling.stan", data=dat, iter=2000,
              chains=4, seed=3, verbose = FALSE)

thetaCP <- as.matrix(modCP, pars="theta")
```

If we plot out the posterior distribution of θ, we see that the median infection rate across sites is 0.31.

```
plot(density(thetaCP, adj=2), main="", xlab="", ylab="",las=1)
mtext(text = expression(paste(theta)), side=1, line = 2)
mtext(text = "P(theta|y)", side=2, line = 2.1)
```



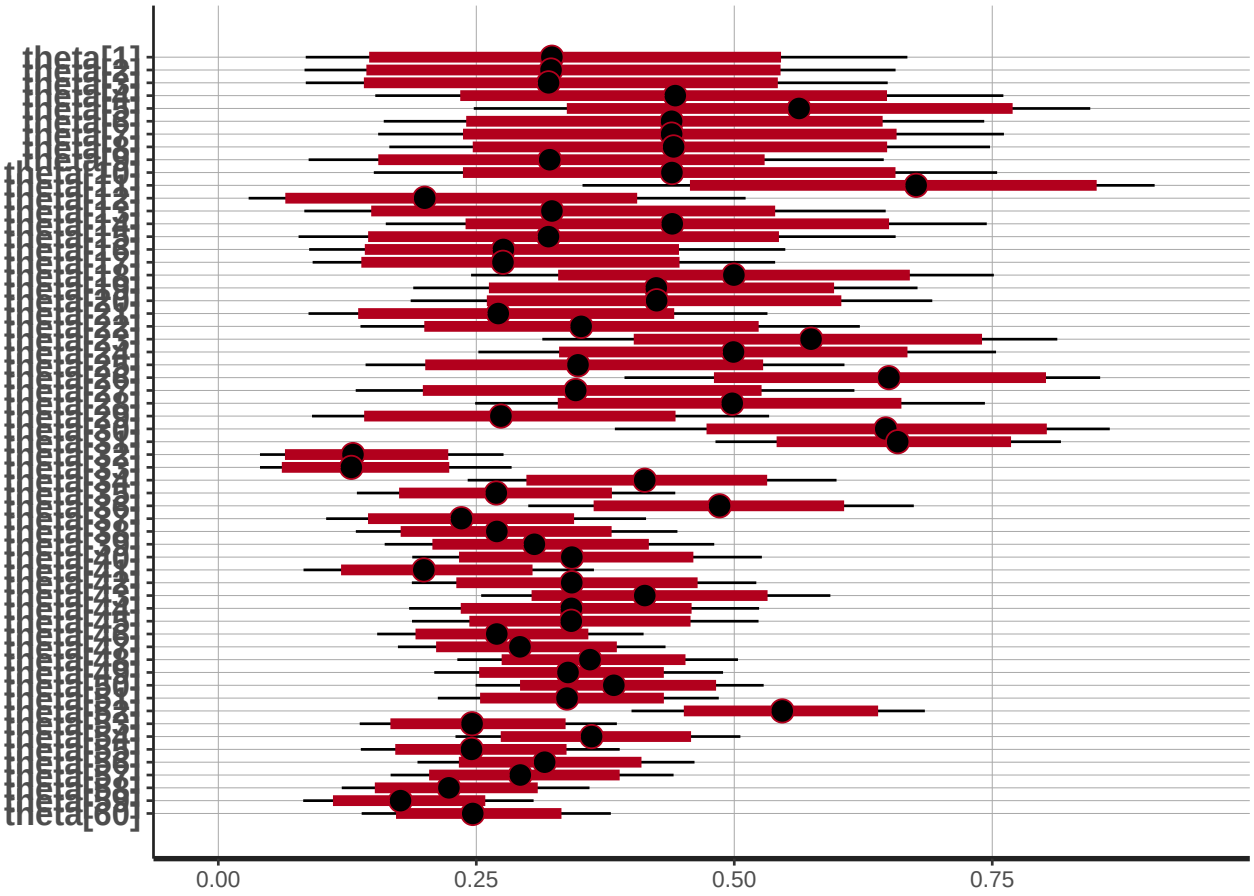
- However, this ignores among-site heterogeneity.

Next let's look at the no-pooling model:

```
modNP <- stan(file="noPooling.stan", data=dat, iter=2000,
              chains=4, seed=3, verbose = FALSE)

thetaNP <- as.matrix(modNP, pars="theta")
medianNP <- apply(thetaNP, 2, median)
```

A plot of the $\theta's$ reveals there is substantial variation among sites, with more uncertainty concentrated in sites with fewer observations (top to bottom):

```
plot(modNP, pars="theta")
```



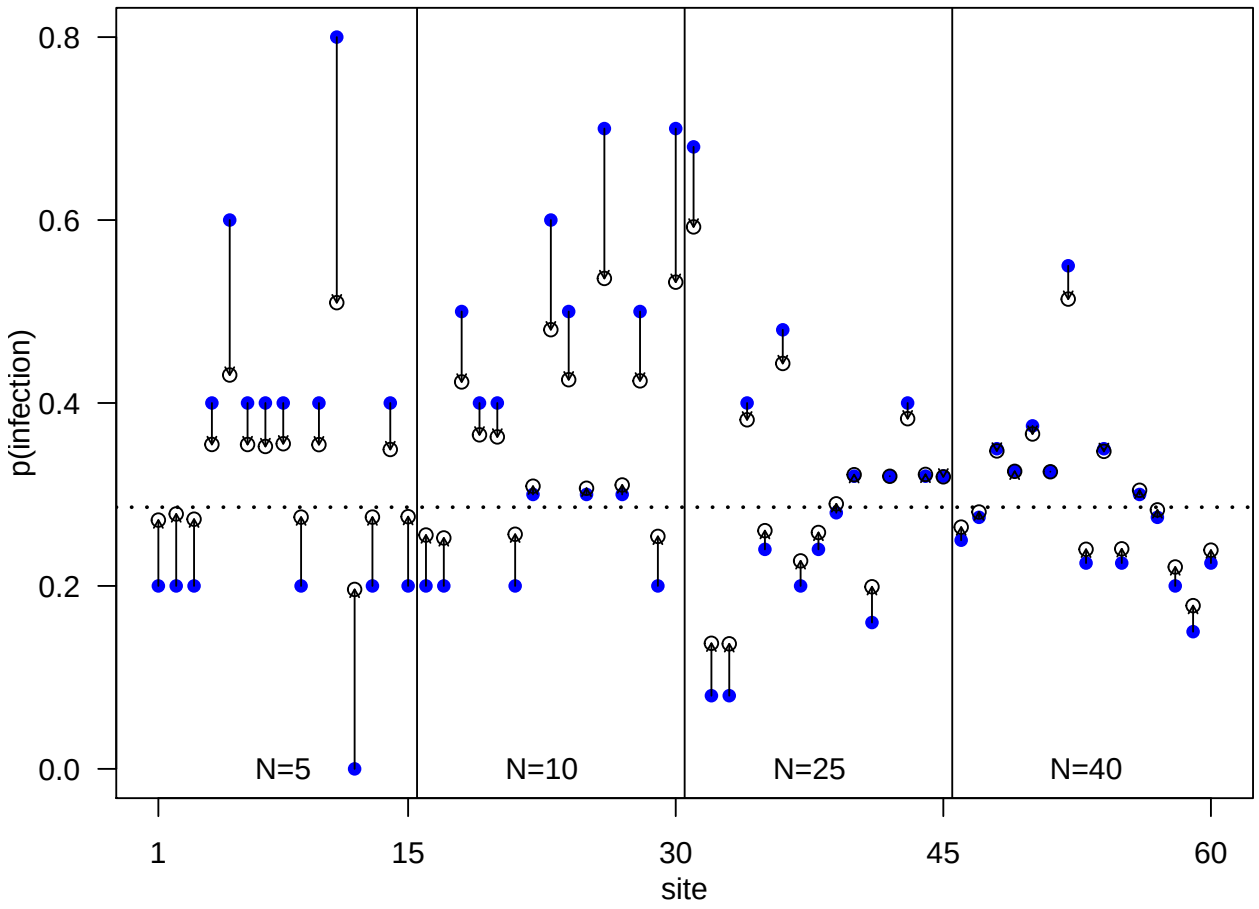
Finally we can run our partial pooling model `partialPooling.stan` and look at it's effects.

```
modPP <- stan(file="partialPooling.stan", data=dat, iter=2000,
              chains=4, seed=3, verbose = FALSE)

thetaPP <- as.matrix(modPP, pars="theta")
omega   <- as.matrix(modPP, pars="omega")
kappa   <- as.matrix(modPP, pars="kappa")

medianPP <- apply(thetaPP, 2, median)
```

To show the impact of partial pooling, we will first plot the empirical proportion of newts infected and then overlay the partial-pooling medians.

```
# Display no-pooling estimates of infection for each pond
plot(newtDat$propInf, col="blue", pch=16, xaxt="n", las=1, xlab="", ylab="")
axis(1, at=c(1, 15, 30, 45, 60))
```

```
mtext(text = "site", side=1, line = 2, cex=1)
mtext("p(infection)", side=2, line=2.2, cex=1)

# Draw vertical dividers between site sample sizes
abline(v=c(15.5, 30.5, 45.5))
text(8, 0, "N=5")
text(22.5, 0, "N=10")
text(37.5, 0, "N=25")
text(53, 0, "N=40")

points(medianPP)
abline(h=median(omega), lty=3, lwd=2)

for(i in 1:length(medianNP)) {
  arrows(x0=i, x1=i, y0=newtDat$propInf[i], y1=medianPP[i], length = 0.05)
}
```



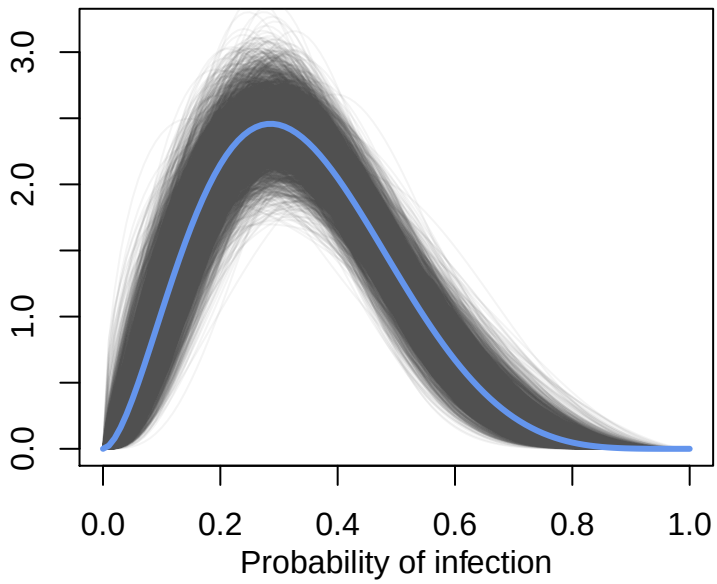
Here the filled blue points represent the raw infection proportions and the black circles indicate the posterior θ medians. The horizontal line indicates the median of ω, the estimated median infection rate across east Tennessee.

Notice that the estimated $\theta's$ are closer to the dashed line than the raw estimates. This is called *shrinkage*.

- Because of information sharing, the estimates will be pulled towards the overall mean.
- Estimates from sites with less information (smaller $N's$) are shrunk more. Also, the greater the distance from the overall mean, the more shrinkage wil happen.

We can also look at what the inferred infection proportions for East TN as a whole should look like:

```
# calculate a & b for each posterior iteration and turn them into a matrix
a <- omega * (kappa - 2) + 1
b <- (1-omega) * (kappa - 2) + 1
propTN <- cbind(a,b)

plot(0:1,0:1, type="n", ylim=c(0,3.2)) # plot empty plot
mtext(text = "Probability of infection", side=1, line = 2, cex=1)

# Overlay infection probability densities for each posterior draw
for (n in 1:nrow(propTN)) {
  curve(dbeta(x, propTN[n,1], propTN[n,2]), add=TRUE, col="#50505010")
}

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



Note that these are different from the densities of ω (can plot as line if you want).

- The density of ω only encompasses uncertainty in the estimation of the posterior mode.
- Estimation of the infection probability for all of TN requires uncertainty about both

the mode and the variation among sites, κ.
  - All of the uncertainty is then propagated together.

Bias vs. variance

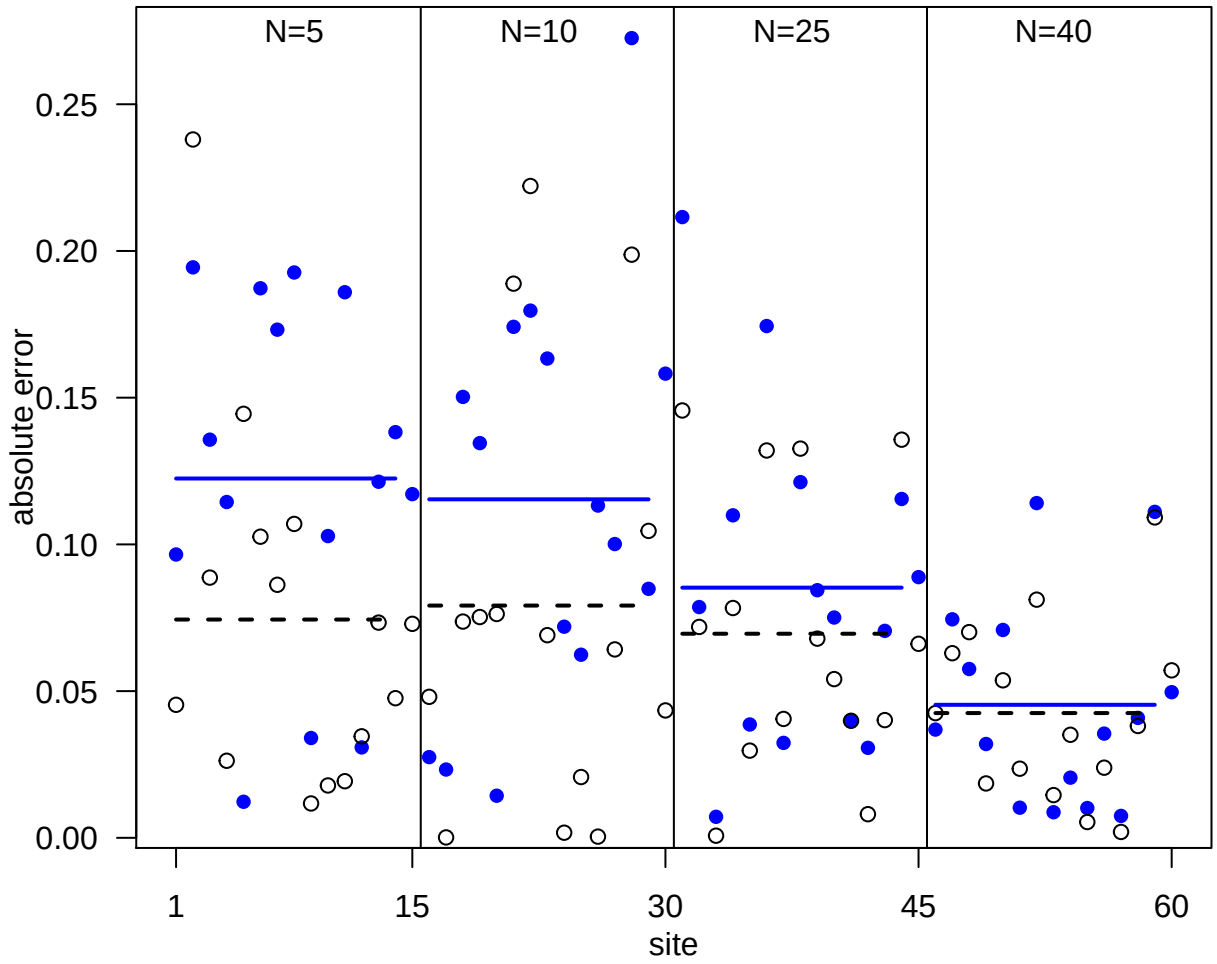
Because the complete pooling approach ignores among-group heterogeneity, it ignores information and tends to underfit.

Conversely, the no-pooling model can result in individual estimates that are imprecise because very little information goes into each estimate of θ—especially for smaller ponds.

Because I simulated the data, we have estimates of the "true" population infection probabilities 'trueInfect'. Therefore we can see how the partial pooling model compromises between over- and underfitting.

We can calculate an estimate of absolute error by taking the absolute value of the difference between each median estimate and the "true" value: $|\theta_{est} - \theta_{true}|$.

```
# calculate the error for no-pooling and partial pooling models
errorNP <- abs(medianNP - newtDat$trueInfect)
errorPP <- abs(medianPP - newtDat$trueInfect)

# calculate the average error for each sample size
avgErrNP <- aggregate(errorNP~N, FUN=mean)$errorNP
avgErrPP <- aggregate(errorPP~N, FUN=mean)$errorPP

# plot the no pooling error in blue
plot(errorNP, col="blue", pch=16, xaxt="n", las=1, xlab="", ylab="")
axis(1, at=c(1, 15, 30, 45, 60))
mtext(text = "site", side=1, line = 2, cex=1)
mtext("absolute error", side=2, line=2.5, cex=1)

# denote the different sample sizes
abline(v=c(15.5, 30.5, 45.5))
text(8, 0.275, "N=5")
text(22.5, 0.275, "N=10")
text(37.5, 0.275, "N=25")
text(53, 0.275, "N=40")

points(errorPP) # plot the partial pooling error estimates

# plot the average error
segments(x0=c(1, 16, 31, 46), x1=c(14, 29, 44, 59), y0=avgErrPP,
         y1=avgErrPP, lty=2, lwd=2)
segments(x0=c(1,16,31,46), x1=c(14, 29, 44, 59), y0=avgErrNP,
         y1=avgErrNP, lty=1, lwd=2, col="blue")
```



First, both no-pooling and partial-pooling estimates are better for sites with more samples More data means more information and better estimation.

Second, note that the dashed line is always lower than the blue lines. This means the partial pooling models always do a better job (lower error than the no pooling models).

Overall, information sharing means better estimation for sites with less information by borrowing from sites with more information. Sites with more information need less correction and so pooling doesn't affect them very much.

However, partial pooling models are not panaceas, and there may be times when no-pooling is better. We are assuming, after all, that groups belong to a large population which may or may not be the case.