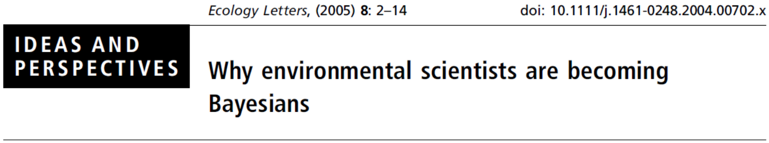
An introduction to Bayesian modelling with R, JAGS and STAN

Francisco Rodriguez-Sanchez (@frod_san)

November 2016

Bayesian modelling is growing fast



Ecology Letters, (2005) **8**: 2–14

doi: 10.1111/j.1461-0248.2004.00702.x

IDEAS AND PERSPECTIVES

Why environmental scientists are becoming Bayesians

- Powerful
- Flexible
- Knowledge synthesis
- Uncertainty

Even if you don't become Bayesian today,
this workshop will help you to understand
regression and 'mixed-effects' models better.

The purpose of models is not to fit data,

but to sharpen the questions

Samuel Karlin

Practical workshop: we'll see main concepts, but do read the literature!

General

- Data analysis using regression and multilevel/hierarchical models
- Statistical Rethinking: A Bayesian Course with Examples in R and Stan
- Bayesian data analysis
- The BUGS book
- Doing Bayesian data analysis: a tutorial with R, JAGS, and STAN
- Bayesian linear mixed models using Stan: a tutorial
- Bayesian basics
- Philosophy and the practice of Bayesian statistics

Practical workshop: we'll see main concepts, but do read the literature!

Ecology-oriented

- The ecological detective: confronting models with data
- Bayesian methods for ecology
- Models for ecological data
- Introduction to WinBUGS for ecologists
- Applied hierarchical modeling in ecology
- Bayesian Models: A Statistical Primer for Ecologists
- and more. . .

Bayesian modelling software

- WinBUGS/OpenBUGS
- JAGS
- STAN
- Filzbach
- Nimble
- Many R packages: MCMCpack, MCMCglmm, LaplacesDemon, r-inla, etc (see Bayesian task view)

We’ll focus on JAGS and STAN

- Fast, powerful, and most popular
- Similar to BUGS
- Easy to start
- Open-ended modelling: deal with complex models too
- But look for specific implementations of your analysis (e.g. hSDM, rstanarm)
- Once concepts understood, switching software not difficult

JAGS has to be installed independently

http://mcmc-jags.sourceforge.net/

Use latest version (4.2.0)

Calling JAGS from R

- rjags
- R2jags
- runjags
- jagsUI
- dclone
- rube

Calling STAN from R

- rstan
- rstanarm
- rethinking
- brms

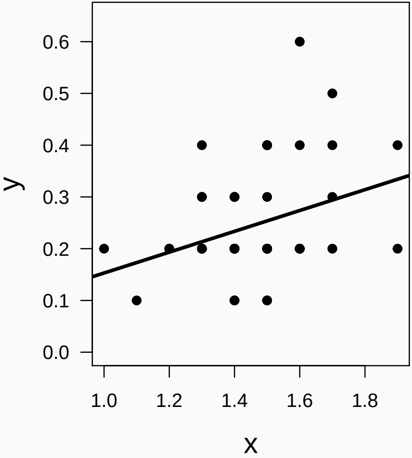
We will also need these R packages:

```r
library(arm)
library(R2jags)
library(ggmcmc)
library(shinystan)
library(rube)
library(lme4)
library(rstan)
library(rstanarm)
```

and their dependencies

The very basics: linear regression

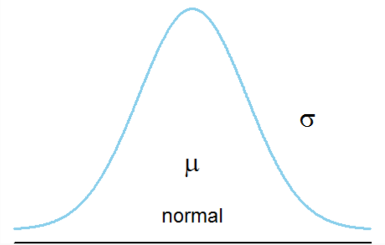
The very basics: linear regression



$$y_i = a + bx_i + \epsilon_i$$

How many parameters?

The very basics: linear regression



$$y_i = a + bx_i + \epsilon_i$$
$$\epsilon_i \sim N(0, \sigma^2)$$

Or also

$$y_i \sim N(\mu_i, \sigma^2)$$
$$\mu_i = a + bx_i$$

Our dataset: tree heights and DBH

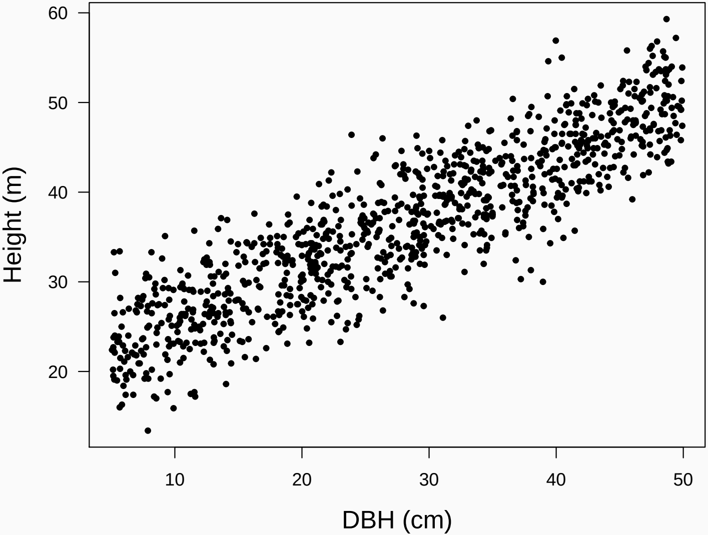
http://tinyurl.com/treesdata

- One species
- 10 plots
- 1000 trees
- Number of trees per plot ranging from 4 to 392

```r
trees <- read.csv("trees.csv")
summary(trees[, 1:3])
```

```
      plot            dbh             height     
 Min.   : 1.0   Min.   : 5.06   Min.   :13.40  
 1st Qu.: 1.0   1st Qu.:17.69   1st Qu.:29.68  
 Median : 2.0   Median :28.62   Median :36.55  
 Mean   : 2.7   Mean   :27.88   Mean   :36.51  
 3rd Qu.: 4.0   3rd Qu.:38.97   3rd Qu.:43.33  
 Max.   :10.0   Max.   :49.92   Max.   :59.30  
```

What's the relationship between DBH and height?



First step: linear regression (lm)

```r
simple.lm <- lm(height ~ dbh, data = trees)
arm::display(simple.lm)  # summary of key model elements
```

```
lm(formula = height ~ dbh, data = trees)
            coef.est coef.se
(Intercept) 19.34     0.31
dbh          0.62     0.01
---
n = 1000, k = 2
residual sd = 4.09, R-Squared = 0.79
```

Interpretation?

Always centre continuous variables

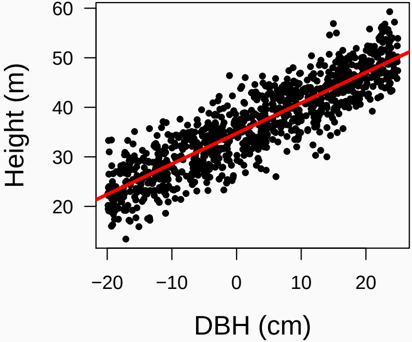
```r
summary(trees$dbh)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.06   17.69   28.62   27.88   38.96   49.92
```

```r
trees$dbh.c <- trees$dbh - 25
```

So, all parameters will be referred to a 25 cm DBH tree.

Linear regression with centred DBH



```
lm(formula = height ~ dbh.c, data = trees)
            coef.est coef.se
(Intercept) 34.73     0.13
dbh.c        0.62     0.01
---
n = 1000, k = 2
residual sd = 4.09, R-Squared = 0.79
```

Let's make it Bayesian

Things we'll need

- A function describing the model (including **priors**)
- Data
- Choose parameters to save
- Define initial values for MCMC chains
- Decide number of iterations (and burnin)

Remember our model structure

$$y_i \sim N(\mu_i, \sigma^2)$$
$$\mu_i = \alpha + \beta x_i$$

In this case:

$$Height_i \sim N(\mu_i, \sigma^2)$$
$$\mu_i = \alpha + \beta DBH_i$$

α: expected height when DBH $= 0$

β: how much height increases with every unit increase of DBH

JAGS uses precision instead of variance

$$\tau = \frac{1}{\sigma^2}$$

So, residual variance $\sigma^2 = 100$ expressed as $\tau = 0.01$.

Specify the model as an R function

$$Height_i \sim N(\mu_i, \sigma^2)$$

$$\mu_i = \alpha + \beta DBH_i$$

```
bayes.lm <- function(){

  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau=precision (inverse var)
    mu[i] <- alpha + beta*dbhc[i]    # expected height ~ dbhc
  }

}
```

We need priors for every parameter!

```
bayes.lm <- function(){

  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau = precision (inverse var)
    mu[i] <- alpha + beta*dbhc[i]    # expected height ~ dbhc
  }

  # PRIORS (vague or weakly informative)
  alpha ~ dunif(1, 100)     # prior avg height of 25-cm-DBH tree
  beta ~ dunif(0, 10)       # how much do we expect height to scale with DBH?
  tau <- 1/(sigma*sigma)    # tau = 1/sigma^2
  sigma ~ dunif(0, 50)      # residual standard deviation
}
```

A note on priors

Avoid ‘non-informative’ priors (see this and this)

Use *weakly informative* (e.g. bounded Uniform, Normal with reasonable parameters, Cauchy. . .)

or *strongly informative* priors based on previous knowledge and common sense.

Some tips for setting priors:

- https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations
- http://www.mrc-bsu.cam.ac.uk/wp-content/uploads/bugsbook_chapter5.pdf

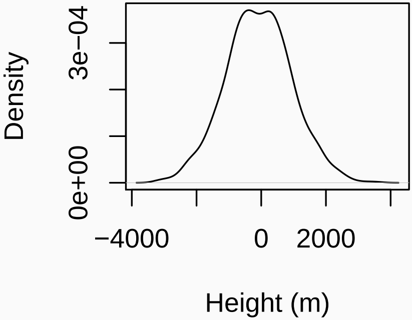
Usually good idea to try different priors and evaluate posterior sensitivity

Or run model without likelihood (priors only).

Example: estimating people height across countries

Unreasonable prior

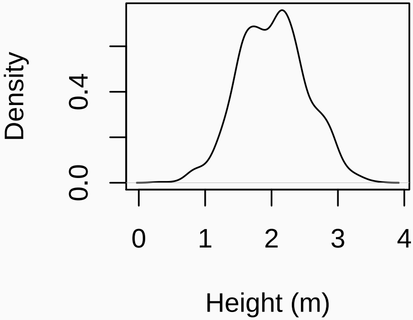
```
plot(density(rnorm(1000, 0, 1000)),
     main="", xlab="Height (m)")
```



(*from STAN manual*)

Reasonable prior

```
plot(density(rnorm(1000, 2, 0.5)),
       main="", xlab="Height (m)")
```



We already have our model definition

```
bayes.lm <- function(){

  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau = precision (inverse of vari
    mu[i] <- alpha + beta*dbhc[i]    # centred diameter
  }

  # PRIORS (vague or weakly informative)
  alpha ~ dunif(1, 100)     # prior for average height of a 25-cm-DBH t
  beta ~ dunif(0, 10)       # how much do we expect height to scale wit
  tau <- 1/(sigma*sigma)    # tau = 1/sigma^2
  sigma ~ dunif(0, 50)      # residual standard deviation
}
```

Next step: create list with data

Data = known values

```
data <- list(height = trees$height,  # response
             dbhc = trees$dbh.c)     # predictor
```

Next step: choose parameters to save

```r
params <- c("alpha", "beta", "sigma")
```

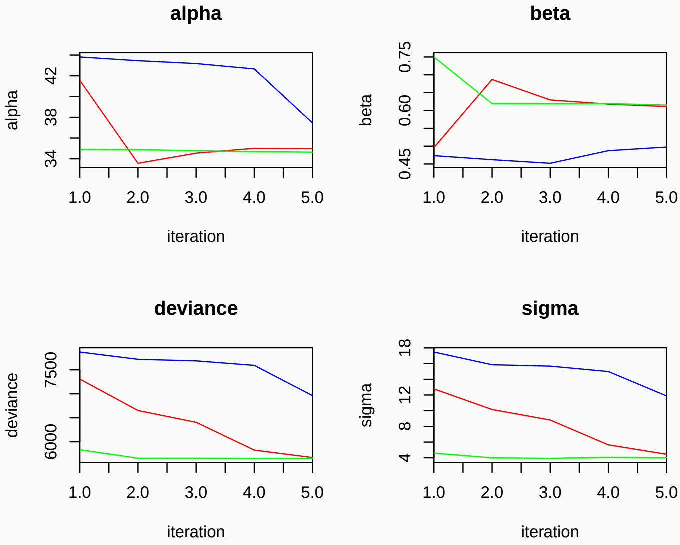
Now call JAGS to run the model

```r
library(R2jags)
```

```r
m1 <- jags(data,
           model.file = bayes.lm,
           parameters.to.save = params,
           n.chains = 3,
           inits = NULL,    # JAGS will create inits for each chain
           n.iter = 10,     # number of iterations
           n.burnin = 5)    # iterations to discard (before convergence)
```

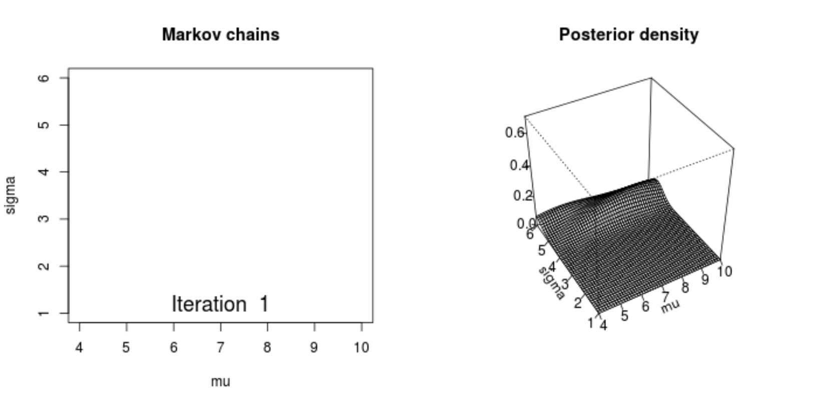
Traceplots: viewing MCMC in action

```
traceplot(m1, ask = FALSE, mfrow = c(2, 2))
```



Obviously we haven't achieved convergence yet

Viewing MCMC in action (II)



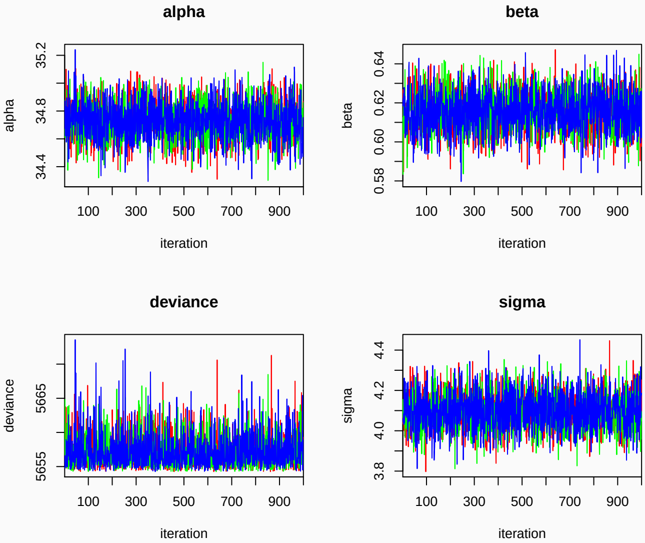
Source: `http://mbjoseph.github.io/2013/09/08/metropolis.html`

Let’s run JAGS for longer

```
m1 <- jags(data,
           model.file = bayes.lm,
           parameters.to.save = params,
           n.chains = 3,
           inits = NULL,
           n.iter = 10000,     # 10000 MCMC iterations
           n.burnin = 5000)    # discard first half (5000 iterations)
```

Use traceplots to asess convergence

```
traceplot(m1, ask = FALSE, mfrow = c(2, 2))
```



Results: parameter estimates

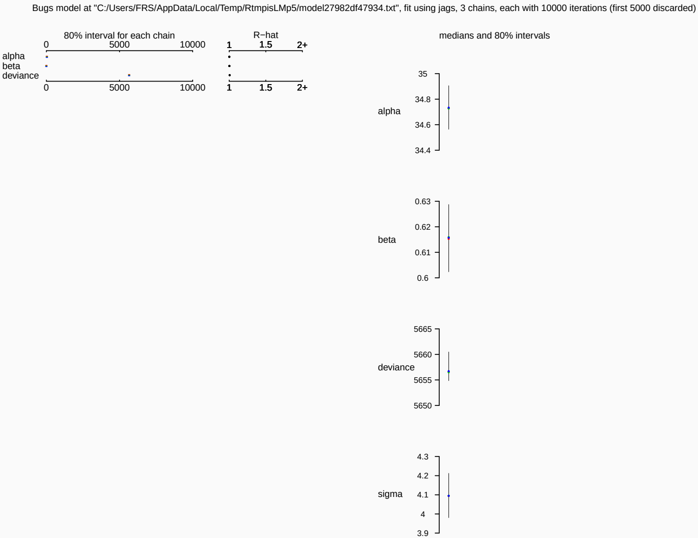
```
Inference for Bugs model at "C:/Users/FRS/AppData/Local/Temp/RtmpisLMp5
 3 chains, each with 10000 iterations (first 5000 discarded), n.thin =
 n.sims = 3000 iterations saved
          mu.vect sd.vect     2.5%    97.5%  Rhat n.eff
alpha      34.732   0.134   34.469   34.992 1.001  3000
beta        0.616   0.010    0.596    0.635 1.002  1500
sigma       4.095   0.090    3.924    4.276 1.001  2000
deviance 5657.236   2.422 5654.464 5663.367 1.006   770

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=

DIC info (using the rule, pD = var(deviance)/2)
pD = 2.9 and DIC = 5660.2
DIC is an estimate of expected predictive error (lower deviance is bett
```

Compare with `simple.lm`

```
lm(formula = height ~ dbh.c, data = trees)
            coef.est coef.se
(Intercept) 34.73     0.13
dbh.c        0.62     0.01
---
n = 1000, k = 2
residual sd = 4.09, R-Squared = 0.79
```

Results pretty similar, because of vague priors

A plot of the whole model



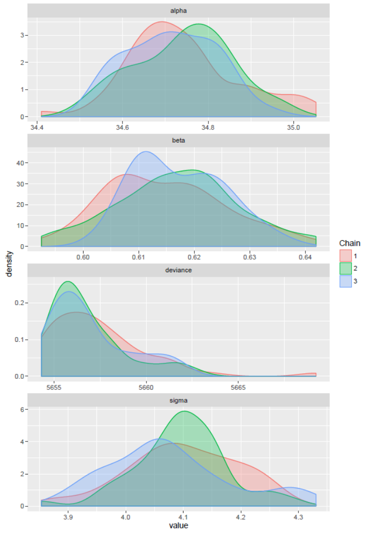
Model checking

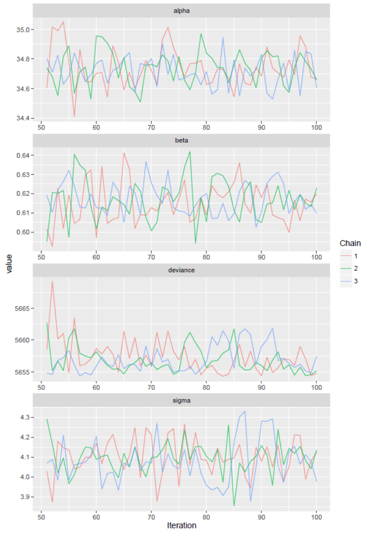
Using ggmcmc to produce diagnostic plots

```r
suppressPackageStartupMessages(library(ggmcmc))
m1.mcmc <- as.mcmc(m1)   # Get list of MCMC values
m1.tidy = ggs(m1.mcmc)   # Produce tidy data frame
ggmcmc(m1.tidy)
```

```
Plotting histograms

Plotting density plots

Plotting traceplots

Plotting running means

Plotting comparison of partial and full chain

Plotting autocorrelation plots

Plotting crosscorrelation plot

Plotting Potential Scale Reduction Factors

Plotting Geweke Diagnostic

Plotting caterpillar plot
```

ggmcmc output sample





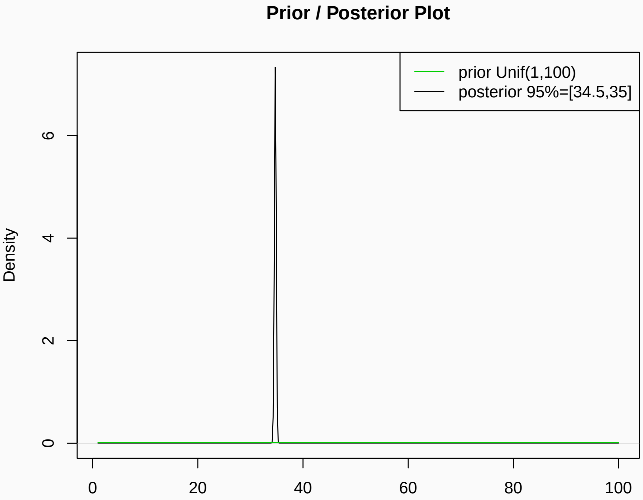
shinystan: assess model interactively

```
library(shinystan)
launch_shinystan(as.shinystan(m1.mcmc))
```

Comparing prior and posterior

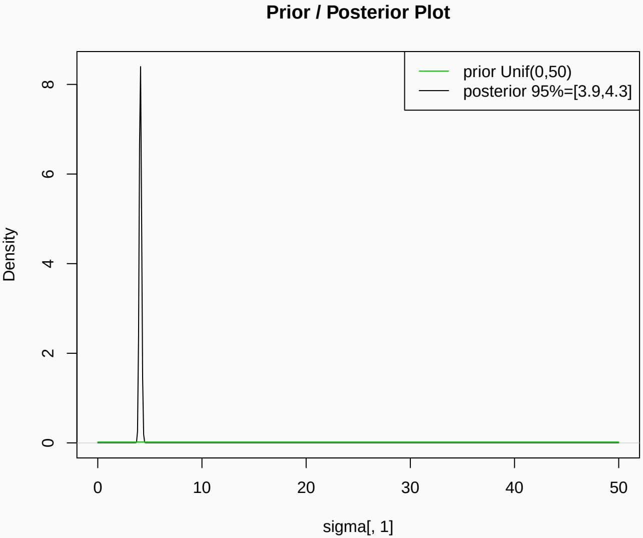
Comparing prior and posterior

```
library(rube)
priPost(post = m1$BUGSoutput$sims.list$alpha[, 1],
        dist = "Uniform", pripar = c(1, 100))
```

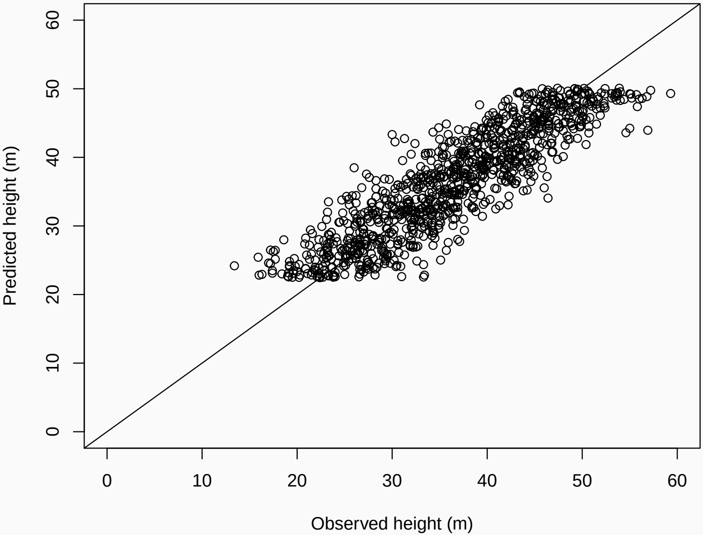


Comparing prior and posterior (sigma)

```
priPost(post = m1$BUGSoutput$sims.list$sigma[, 1],
        dist = "Uniform", pripar = c(0, 50))
```



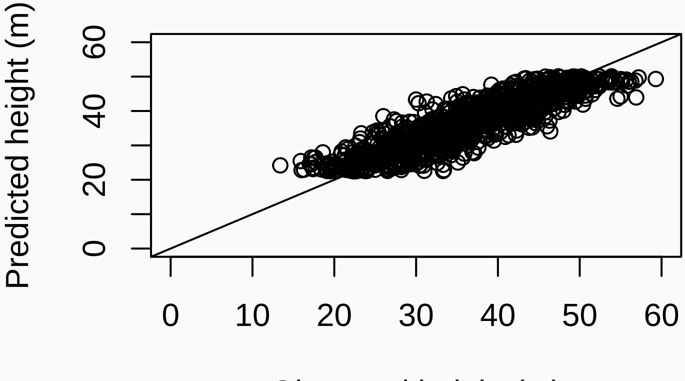
Observed vs Predicted values



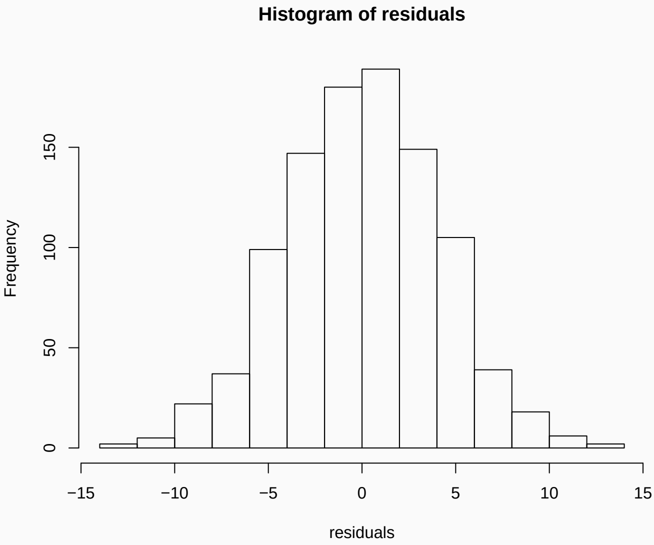
How would you do this?

Observed vs Predicted values

```r
alpha.avg <- mean(m1$BUGSoutput$sims.list$alpha)
beta.avg <- mean(m1$BUGSoutput$sims.list$beta)
mu <- alpha.avg + beta.avg*trees$dbh.c
plot(trees$height, mu,
     xlim = c(0, 60), ylim = c(0, 60),
     xlab = "Observed height (m)", ylab = "Predicted height (m)")
abline(a = 0, b = 1)
```



Residuals


Histogram of residuals
Frequency
0
50
100
150
−15
−10
−5
0
5
10
15
residuals

posterior predictive distribution

Now using Normal vague priors

Model with Normal priors

```
bayes.lm.N <- function(){

  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau = precision (inverse of vari
    mu[i] <- alpha + beta*dbhc[i]    # centred diameter
  }

  # PRIORS
  alpha ~ dnorm(0, 0.01)      # prior for intercept
  beta ~ dnorm(0, 0.01)       # prior for beta (slope)
  tau <- 1/(sigma*sigma)      # tau = 1/sigma^2
  sigma ~ dunif(0, 50)        # residual standard deviation
}
```

Calling JAGS

```r
m1b <- jags(data,
           model.file = bayes.lm.N,
           parameters.to.save = params,
           n.chains = 3,
           inits = NULL,
           n.iter = 4000,
           n.burnin = 2000)
```

Results with Normal priors on intercept and slope

```
Inference for Bugs model at "C:/Users/FRS/AppData/Local/Temp/RtmpisLMp5
 3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
 n.sims = 3000 iterations saved
         mu.vect sd.vect     2.5%    97.5%  Rhat n.eff
alpha     34.727   0.130   34.466   34.978 1.001  3000
beta       0.616   0.010    0.597    0.635 1.001  3000
sigma      4.095   0.091    3.926    4.272 1.001  2000
deviance 5657.132   2.368 5654.429 5663.500 1.001  3000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=

DIC info (using the rule, pD = var(deviance)/2)
pD = 2.8 and DIC = 5659.9
DIC is an estimate of expected predictive error (lower deviance is bett
```

Very similar

Let’s try strongly informative prior on alpha

```
bayes.lm.N2 <- function(){

  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau = precision (inverse of vari
    mu[i] <- alpha + beta*dbhc[i]    # centred diameter
  }

  # PRIORS
  alpha ~ dnorm(10, 10)      # prior for intercept
  beta ~ dnorm(0, 0.01)      # prior for beta (slope)
  tau <- 1/(sigma*sigma)     # tau = 1/sigma^2
  sigma ~ dunif(0, 50)       # residual standard deviation
}
```

Calling JAGS

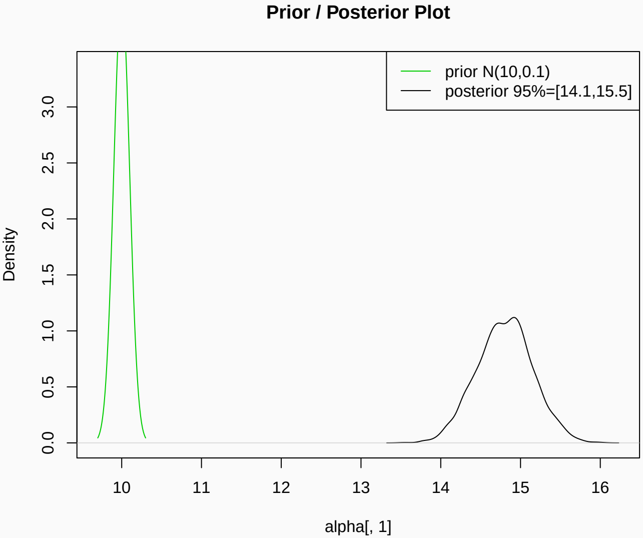
```r
m1c <- jags(data,
           model.file = bayes.lm.N2,
           parameters.to.save = params,
           n.chains = 3,
           inits = NULL,
           n.iter = 4000,
           n.burnin = 2000)
```

Posteriors: between prior and likelihood

```
Inference for Bugs model at "C:/Users/FRS/AppData/Local/Temp/RtmpisLMp5
 3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
 n.sims = 3000 iterations saved
         mu.vect sd.vect     2.5%    97.5%  Rhat n.eff
alpha     14.801   0.352   14.104   15.495 1.001  2800
beta       0.950   0.048    0.856    1.042 1.001  3000
sigma     19.900   0.553   18.831   20.998 1.001  3000
deviance 8817.718  33.837 8750.531 8884.142 1.001  2900

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=

DIC info (using the rule, pD = var(deviance)/2)
pD = 572.5 and DIC = 9390.2
DIC is an estimate of expected predictive error (lower deviance is bett
```

Prior vs Posterior

```
priPost(post = m1c$BUGSoutput$sims.list$alpha[,1],
        dist = "Normal", pripar = c(10, 0.1))
```



Bayesian inference

Bayes theorem: prior, likelihood, posterior

$$P(Hypothesis|Data) = \frac{P(Data|Hypothesis)P(Hypothesis)}{P(Data)}$$

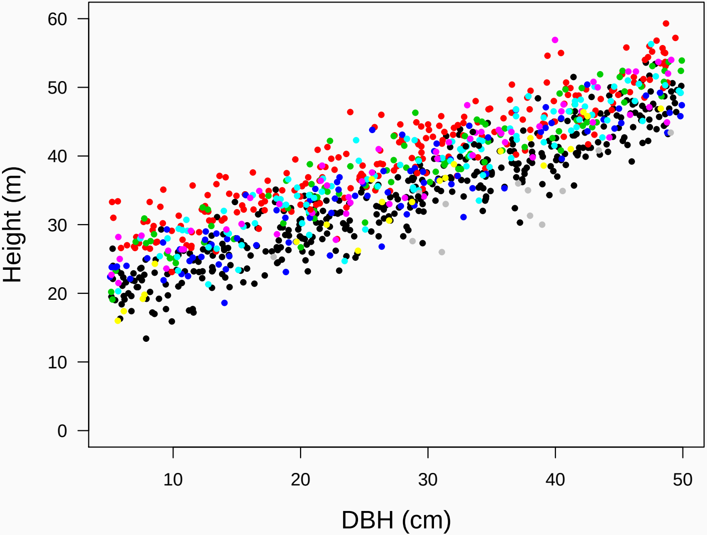
$$P(Hypothesis|Data) \propto P(Data|Hypothesis) \times P(Hypothesis)$$

$$Posterior \propto Likelihood \times Prior$$

Hence, prior influence decreases with size of data set.

Varying-intercept models

Accounting for plot effects



Do it yourself using lm

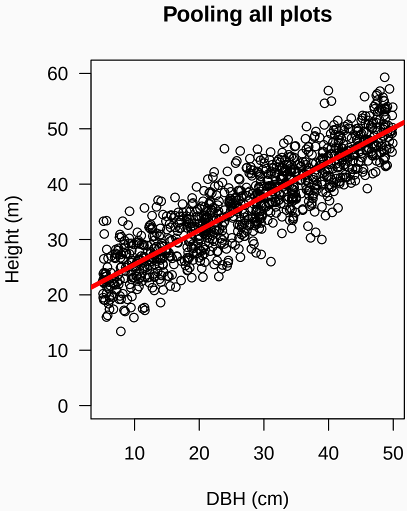
lm results

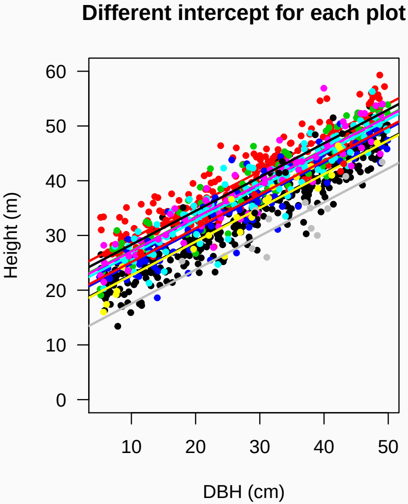
```r
lm.plot <- lm(height ~ factor(plot) + dbh.c, data = trees)
```

```
lm(formula = height ~ factor(plot) + dbh.c, data = trees)
               coef.est coef.se
(Intercept)    32.13     0.16
factor(plot)2   6.50     0.26
factor(plot)3   4.36     0.35
factor(plot)4   1.93     0.36
factor(plot)5   3.64     0.34
factor(plot)6   4.20     0.42
factor(plot)7  -0.18     0.67
factor(plot)8  -5.31     0.89
factor(plot)9   5.44     1.09
factor(plot)10  2.26     1.37
dbh.c           0.62     0.01
---
n = 1000, k = 11
residual sd = 3.04, R-Squared = 0.88
```

Interpretation?

Single vs varying intercept





Let's make it Bayesian

Things we'll need

- A function describing the model (including **priors**)
- Data
- Choose parameters to save
- Define initial values for MCMC chains
- Decide number of iterations (and burnin)

Bayesian varying-intercept model with no pooling

```
varint.nopool <- function(){
  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau = precision (inverse of vari
    mu[i] <- alpha[plot[i]] + beta*dbhc[i]     # centred diameter
  }
  # PRIORS
  #alpha ~ dnorm(0, .001)      # previous model
  for (j in 1:10){
    alpha[j] ~ dnorm(0, .001)  # Plot effects drawn from Normal distrib
                               # with large **fixed** variance
  }
  beta ~ dnorm(0, .001)
  tau <- 1/(sigma*sigma)       # tau = 1/sigma^2
  sigma ~ dunif(0, 50)
}
```

This fits same model as `lm.plot`

Call JAGS

```r
data <- list(height = trees$height,
             dbhc = trees$dbh.c,
             plot = trees$plot)
m2 <- jags(data,
           model.file = varint.nopool,
           parameters.to.save = params,
           n.chains = 3,
           inits = NULL,
           n.iter = 4000,
           n.burnin = 2000)
```

Running JAGS in parallel (multiple cores)

```r
m2 <- jags.parallel(data,
           model.file = varint.nopool,
           parameters.to.save = params,
           n.chains = 3,
           inits = NULL,
           n.iter = 4000,
           n.burnin = 2000)
```

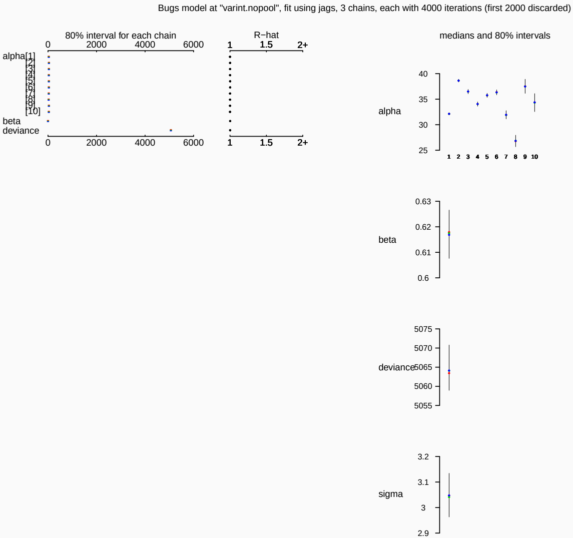
Results of Bayesian varying intercept model without pooling

```
Inference for Bugs model at "varint.nopool", fit using jags,
 3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
 n.sims = 3000 iterations saved
          mu.vect sd.vect     2.5%    97.5%  Rhat n.eff
alpha[1]    32.126   0.154   31.821   32.427 1.001  3000
alpha[2]    38.631   0.202   38.228   39.033 1.002  3000
alpha[3]    36.487   0.328   35.830   37.109 1.002  1700
alpha[4]    34.050   0.322   33.397   34.663 1.001  2200
alpha[5]    35.754   0.309   35.149   36.346 1.002  1400
alpha[6]    36.339   0.389   35.550   37.094 1.002  1100
alpha[7]    31.934   0.632   30.746   33.168 1.001  3000
alpha[8]    26.810   0.870   25.174   28.493 1.001  3000
alpha[9]    37.519   1.089   35.460   39.709 1.001  3000
alpha[10]   34.341   1.367   31.708   36.983 1.001  3000
beta         0.617   0.008    0.603    0.631 1.003   950
sigma        3.047   0.067    2.919    3.183 1.003   680
deviance  5064.489   4.777 5056.981 5075.523 1.002  1000

For each parameter, n.eff is a crude measure of effective sample size,
```

Same results as `lm.plot`

```
lm(formula = height ~ factor(plot) + dbh.c, data = trees)
               coef.est coef.se
(Intercept)    32.13     0.16
factor(plot)2   6.50     0.26
factor(plot)3   4.36     0.35
factor(plot)4   1.93     0.36
factor(plot)5   3.64     0.34
factor(plot)6   4.20     0.42
factor(plot)7  -0.18     0.67
factor(plot)8  -5.31     0.89
factor(plot)9   5.44     1.09
factor(plot)10  2.26     1.37
dbh.c           0.62     0.01
---
n = 1000, k = 11
residual sd = 3.04, R-Squared = 0.88
```

Plot whole model



The varying-intercept model seems better

Based on Deviance Information Criterion (DIC)

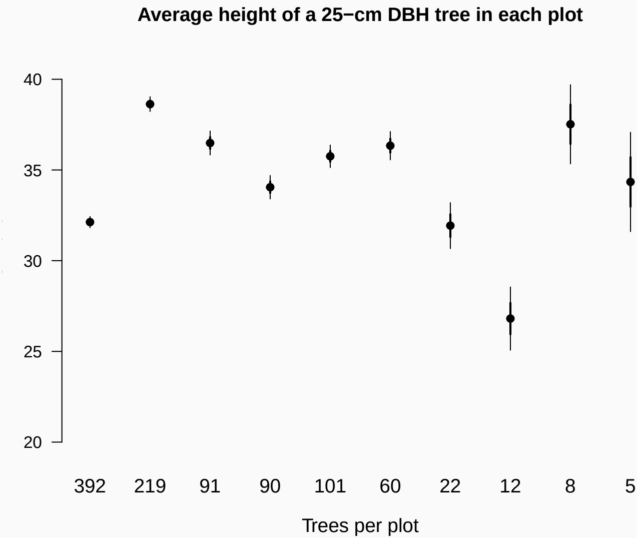
DIC(m1) = 5660

DIC(m2) = 5076

But note DIC (as AIC and BIC) have been criticised

WAIC and cross-validation are preferred alternatives.

Estimation of plot effects improves with sample size



Varying-intercepts with pooling (shrinkage)

Multilevel model with varying intercepts

$$\begin{aligned} y_i &= a_j + bx_i + \varepsilon_i \\ a_j &\sim N(0, \tau^2) \\ \varepsilon_i &\sim N(0, \sigma^2) \end{aligned}$$

In our example:

$$\begin{aligned} Height_i &= plot_j + bDBH_i + \varepsilon_i \\ plot_j &\sim N(0, \tau^2) \\ \varepsilon_i &\sim N(0, \sigma^2) \end{aligned}$$

Random vs fixed effects

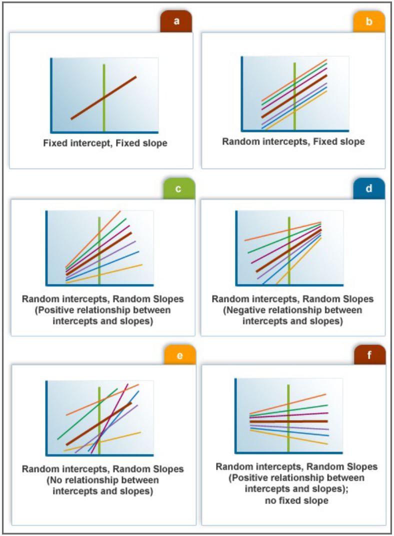
Many definitions, so much confusion

What is a random effect, really?

1. Varies by group
2. Variation estimated with probability model

Random effects are estimated with partial pooling, while fixed effects are not (infinite variance).

Random vs fixed effects


a
Fixed intercept, Fixed slope
b
Random intercepts, Fixed slope
c
Random intercepts, Random Slopes (Positive relationship between intercepts and slopes)
d
Random intercepts, Random Slopes (Negative relationship between intercepts and slopes)
e
Random intercepts, Random Slopes (No relationship between intercepts and slopes)
f
Random intercepts, Random Slopes (Positive relationship between intercepts and slopes); no fixed slope

http://www.esourceresearch.org/

Fitting mixed models with lmer

```
mixed <- lmer(height ~ dbh.c + (1|plot), data = trees)
```

```
lmer(formula = height ~ dbh.c + (1 | plot), data = trees)
            coef.est coef.se
(Intercept) 34.43     1.08  
dbh.c        0.62     0.01  

Error terms:
 Groups   Name        Std.Dev.
 plot     (Intercept) 3.35    
 Residual             3.04    
---
number of obs: 1000, groups: plot, 10
AIC = 5116.3, DIC = 5096.3
deviance = 5102.3
```

lmer coefficients

```
coef(mixed)
```

```
$plot
   (Intercept)     dbh.c
1     32.13118 0.6169271
2     38.61479 0.6169271
3     36.46547 0.6169271
4     34.06404 0.6169271
5     35.75313 0.6169271
6     36.30517 0.6169271
7     32.04003 0.6169271
8     27.30620 0.6169271
9     37.27097 0.6169271
10    34.39546 0.6169271

attr(,"class")
[1] "coef.mer"
```

Bayesian varying-intercept model with pooling across plots

```
varint.pool <- function(){
    # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau = precision (inverse of vari
    mu[i] <- alpha[plot[i]] + beta*dbhc[i]     # centred diameter
  }
    # PRIORS
  for (j in 1:10){
    alpha[j] ~ dnorm(grandmu, tauplot)  # Now we are estimating the plo
  }
  grandmu ~ dnorm(0, .001)     # Overall mean height across all plots
  tauplot <- 1/(sigmaplot*sigmaplot)
  sigmaplot ~ dunif(0, 20)     # between-plot variance
  beta ~ dnorm(0, .001)
  tau <- 1/(sigma*sigma)
  sigma ~ dunif(0, 50)         # residual variance
}
```

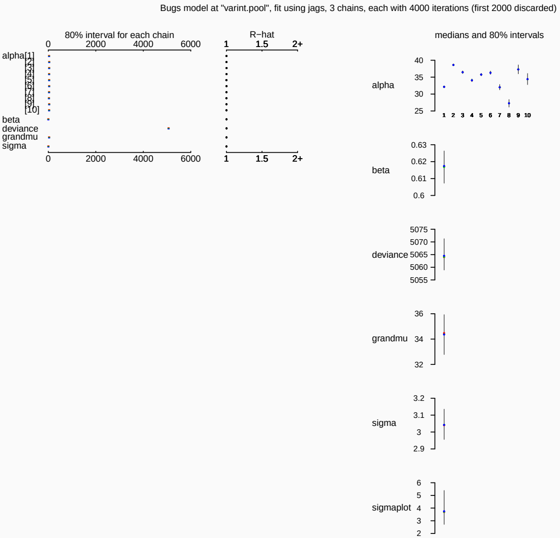
Call JAGS

```r
data <- list(height = trees$height,
             dbhc = trees$dbh.c,
             plot = trees$plot)
m3 <- jags.parallel(data,
           model.file = varint.pool,
           parameters.to.save = c("alpha", "beta", "sigma", "grandmu",
           n.chains = 3,
           inits = NULL,
           n.iter = 4000,
           n.burnin = 2000)
```

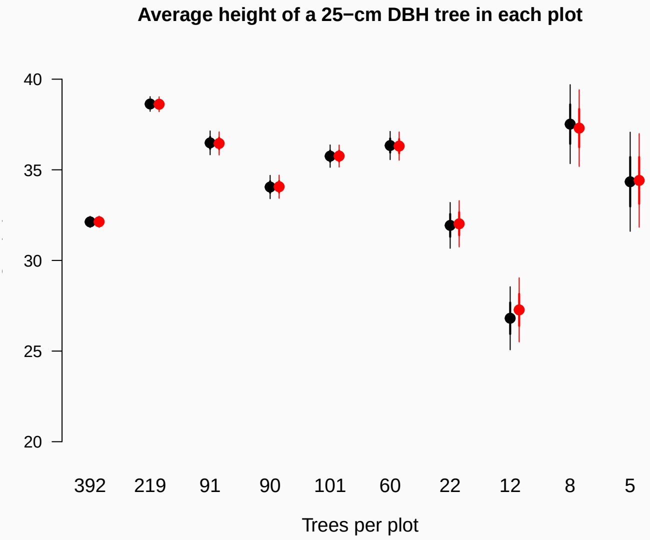
Results of multilevel model with pooling

```
Inference for Bugs model at "varint.pool", fit using jags,
 3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
 n.sims = 3000 iterations saved
           mu.vect sd.vect    2.5%    97.5%  Rhat n.eff
alpha[1]    32.133   0.156  31.827   32.440 1.001  3000
alpha[2]    38.616   0.205  38.211   39.010 1.001  2800
alpha[3]    36.455   0.317  35.839   37.090 1.002  1400
alpha[4]    34.067   0.318  33.436   34.689 1.002  1700
alpha[5]    35.760   0.303  35.174   36.352 1.001  3000
alpha[6]    36.308   0.389  35.544   37.093 1.002  1800
alpha[7]    32.022   0.636  30.818   33.296 1.001  3000
alpha[8]    27.273   0.884  25.561   28.951 1.001  3000
alpha[9]    37.299   1.056  35.230   39.347 1.001  2700
alpha[10]   34.414   1.290  31.822   36.969 1.001  3000
beta         0.617   0.007   0.602    0.631 1.002  1900
grandmu     34.372   1.280  31.696   36.863 1.001  2500
sigma        3.044   0.070   2.913    3.184 1.001  3000
sigmaplot    3.927   1.162   2.353    6.881 1.001  3000
deviance  5064.690   4.881 5056.896 5075.604 1.002  1300
```

A plot of the whole model

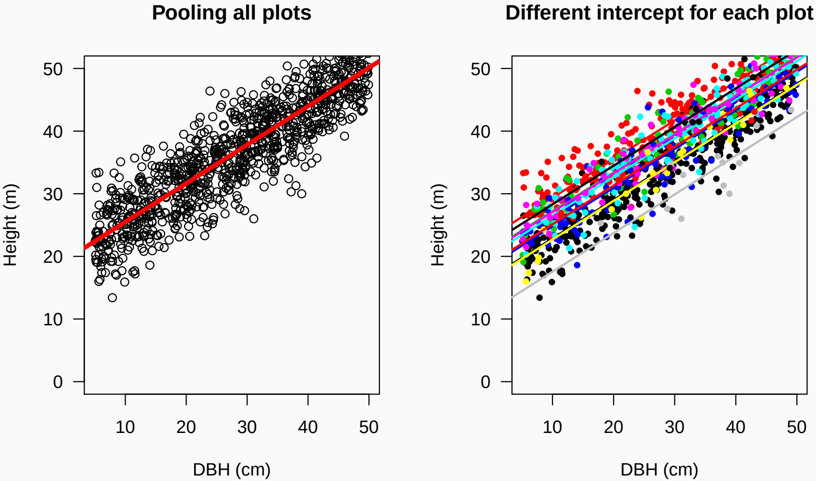


Comparing plot coefficients



A gradient from complete to no pooling

The multilevel model (with pooling) is somewhere between the complete-pooling (single intercept) and the no-pooling (one intercept for each plot, without shrinkage) models.



Growing the hierarchy: adding plot-level predictors

Model with group-level predictors

We had:

$$\begin{aligned} y_i &= a_j + bx_i + \varepsilon_i \\ a_j &\sim N(0, \tau^2) \\ \varepsilon_i &\sim N(0, \sigma^2) \end{aligned}$$

Now

$$\begin{aligned} y_i &= a_j + bx_i + \varepsilon_i \\ a_j &\sim N(\mu_j, \tau^2) \\ \mu_j &= \gamma + \delta \cdot predictor_j \\ \varepsilon_i &\sim N(0, \sigma^2) \end{aligned}$$

Reading plot data

```r
plotdata <- read.csv("plotdata.csv")
temp.c <- plotdata$temp - 15
```

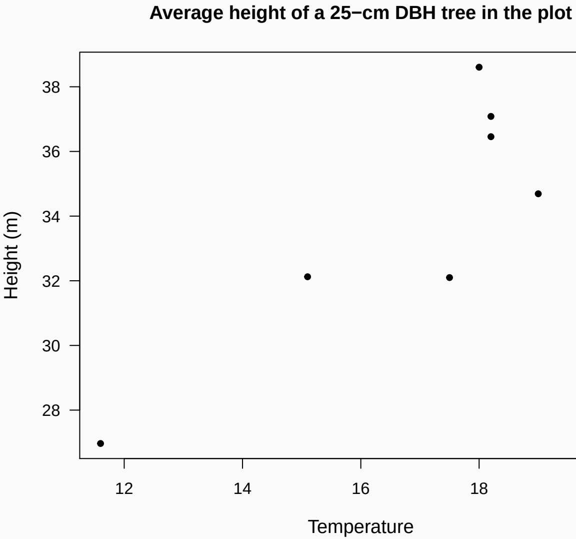
Model with group-level predictors

```
group.preds <- function(){
  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha[plot[i]] + beta*dbhc[i]
  }
  # PRIORS
  for (j in 1:10){
    alpha[j] ~ dnorm(grandmu + beta.temp*tempc[j], tauplot)
  }
  beta.temp ~ dnorm(0, .001)   # slope for temperature effects
  grandmu ~ dnorm(0, .001)
  tauplot <- 1/(sigmaplot*sigmaplot)
  sigmaplot ~ dunif(0, 20)
  beta ~ dnorm(0, .001)
  tau <- 1/(sigma*sigma)
  sigma ~ dunif(0, 50)
}
```

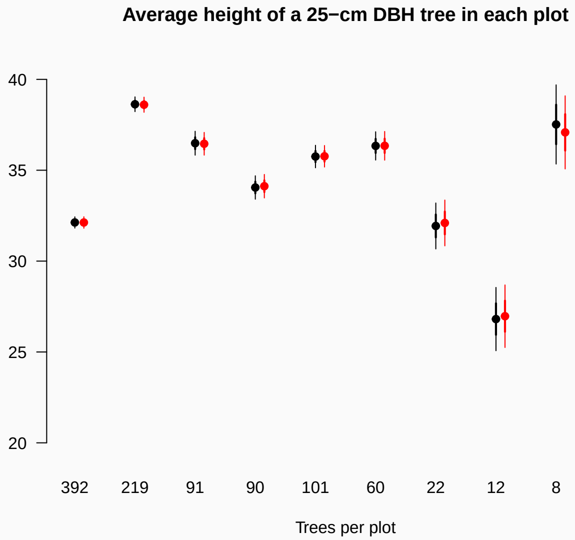
running JAGS...

```
data <- list(height = trees$height,
             dbhc = trees$dbh.c,
             plot = trees$plot,
             tempc = temp.c)
params = c("alpha","beta","sigma", "grandmu", "sigmaplot", "beta.temp")
m4 <- jags.parallel(data,
           model.file = group.preds,
           parameters.to.save = params,
           n.chains = 3,
           inits = NULL,
           n.iter = 4000,
           n.burnin = 2000)
```

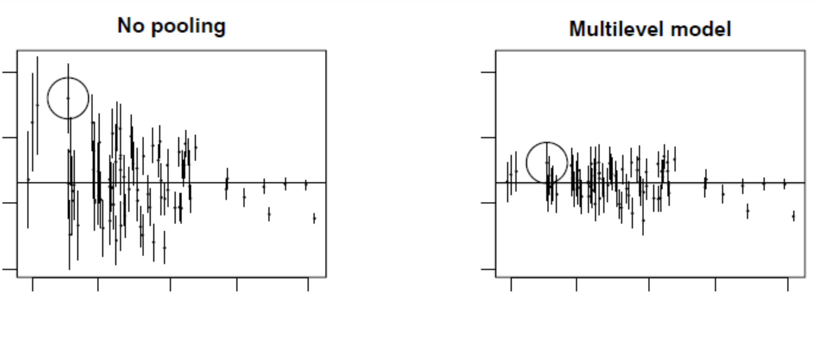
Average heights among plots related to temperature



Adding plot-level predictors (pooling) may improve parameter estimation



Adding plot-level predictors (pooling) may improve parameter estimation



From Gelman & Hill p. 253

Slopes can also vary. . .

- and coefficients be estimated with pooling
- but taking into account correlation between slopes and intercepts
- (see e.g. Gelman & Hill 2007, ch. 13)

Recapitulating. . .

So what's a multilevel/hierarchical model?

- Parameters/coefficients are given a probability model (with their own hyperparameters estimated from data).
- Intercepts and/or slopes may vary, and can be modelled (sometimes including their own predictors).

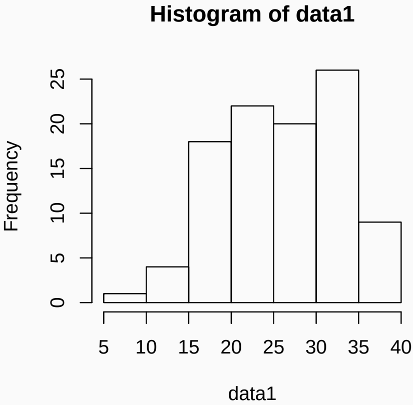
Advantages of hierarchical Bayes

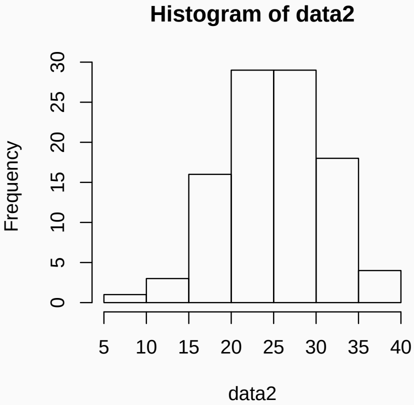
- Perfect for structured data (space-time)
- Predictors enter at the appropriate level
- Accommodate variation in treatment effects
- More efficient inference of regression parameters
- Using all the data to perform inferences for groups with small sample size
- Predictions fully accounting for uncertainty and variability
- Prior information

Datasets are stochastic realisations of a process

```
data1=rnorm(100, 2 + 1.6*x, 5)
```

```
data2=rnorm(100, 2 + 1.6*x, 5)
```

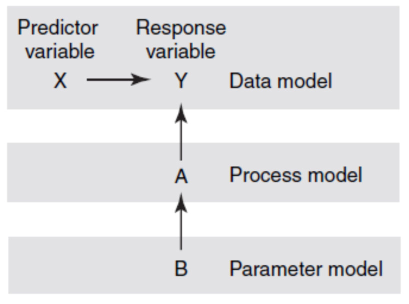




These two datasets are different, even though they arise from same process.

Data inform about process, but have ‘noise’ too.

Hierarchical Bayes: data, process, parameters



Clark et al. 2006, Clark 2007

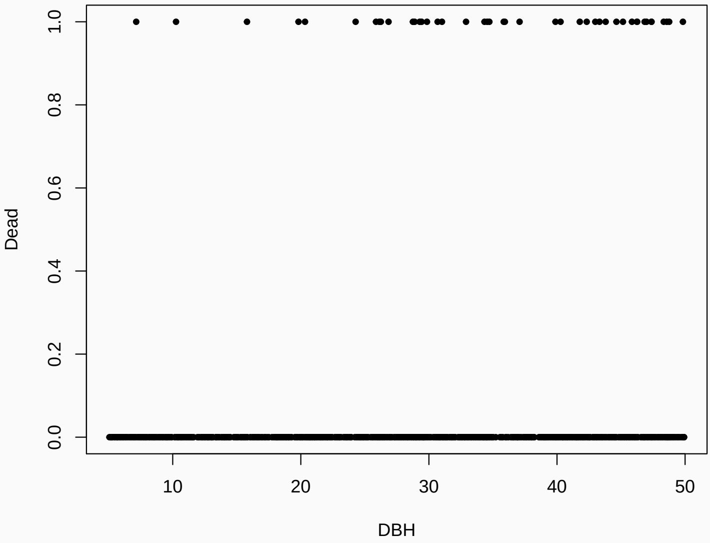
$$
\begin{aligned}
& f(\text{data,process,parameters}) \\
& \quad \propto f(\text{data}|\text{process, parameters}) \\
& \qquad \times f(\text{process}|\text{parameters}) \\
& \qquad \times f(\text{parameters}).
\end{aligned}
$$

Exercise

Does sex influence height?

Bayesian logistic regression

Relationship between tree size and mortality



Logistic regression model

```
logreg <- function(){

  # LIKELIHOOD
  for (i in 1:length(dead)){
    dead[i] ~ dbern(pdeath[i])
    logit(pdeath[i]) <- mu + beta*dbhc[i]
  }

  # PRIORS
  mu ~ dnorm(0, .001)
  beta ~ dnorm(0, .001)
}
```

Calling JAGS

```r
data <- list(dead = trees$dead,
             dbhc = trees$dbh.c)
m5 <- jags(data,
           model.file = logreg,
           parameters.to.save = c("mu", "beta"),
           n.chains = 3,
           inits = NULL,
           n.iter = 4000,
           n.burnin = 2000)
```

Results

```
Inference for Bugs model at "C:/Users/FRS/AppData/Local/Temp/RtmpisLMp5
 3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
 n.sims = 3000 iterations saved
         mu.vect sd.vect    2.5%   97.5%  Rhat n.eff
beta       0.056   0.014   0.028   0.086 1.003   880
mu        -3.484   0.250  -3.951  -3.043 1.002  1500
deviance 346.416  12.370 343.740 352.129 1.001  3000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=

DIC info (using the rule, pD = var(deviance)/2)
pD = 76.6 and DIC = 423.0
DIC is an estimate of expected predictive error (lower deviance is bett
```

Compare with glm

```r
logreg <- glm(dead ~ dbh.c, data = trees, family = binomial)
```

```
glm(formula = dead ~ dbh.c, family = binomial, data = trees)
            coef.est coef.se
(Intercept) -3.44     0.21
dbh.c        0.05     0.01
---
  n = 1000, k = 2
  residual deviance = 343.7, null deviance = 360.9 (difference = 17.2)
```

STAN

STAN models are defined similarly to JAGS
(but see STAN manual for differences)

Our Bayesian linear regression in JAGS

```
bayes.lm <- function(){

  # LIKELIHOOD
  for (i in 1:length(height)){
    height[i] ~ dnorm(mu[i], tau)    # tau = precision (inverse of vari
    mu[i] <- alpha + beta*dbhc[i]    # centred diameter
  }

  # PRIORS (vague or weakly informative)
  alpha ~ dunif(1, 100)     # prior for average height of a 25-cm-DBH t
  beta ~ dunif(0, 10)       # how much do we expect height to scale wit
  tau <- 1/(sigma*sigma)    # tau = 1/sigma^2
  sigma ~ dunif(0, 50)      # residual standard deviation
}
```

Bayesian linear regression in STAN

```
bayes.lm.stan <- "

  data {
    int<lower=0> N;
    vector[N] dbhc;
    vector[N] height;
  }

  parameters {
    real<lower=0, upper=100> alpha;
    real<lower=0, upper=10> beta;
    real<lower=0, upper=50> sigma;
  }

  model {
    height ~ normal(alpha + beta * dbhc, sigma);
  }
"
```

Fitting model in STAN

```r
library(rstan)
lm.stan <- stan(model_code = bayes.lm.stan,
                data = list(height = trees$height, dbhc = trees$dbh.c,
                            N = nrow(trees)),
                pars = c("alpha", "beta", "sigma"),
                chains = 3,
                iter = 2000,
                init = "random")
```

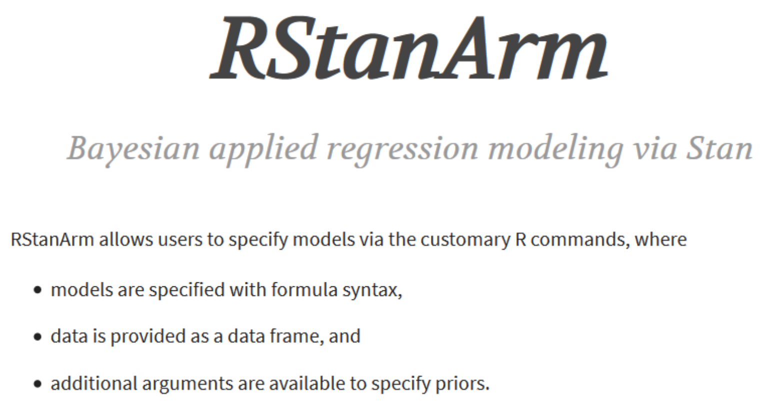
Similar results as JAGS

```
Inference for Stan model: 57b6d06b34f74072ecc985613f1ce936.
3 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=3000.

          mean se_mean   sd     2.5%      25%      50%      75%    97.5
alpha    34.73    0.00 0.13    34.48    34.64    34.73    34.81    34.9
beta      0.62    0.00 0.01     0.60     0.61     0.62     0.62     0.6
sigma     4.10    0.00 0.09     3.93     4.03     4.10     4.16     4.2
lp__  -1905.70    0.03 1.19 -1908.75 -1906.23 -1905.36 -1904.84 -1904.4
      n_eff Rhat
alpha  2671    1
beta   2462    1
sigma  2334    1
lp__   1526    1

Samples were drawn using NUTS(diag_e) at Tue Nov 15 17:29:09 2016.
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).
```

Lots of example models in STAN

https://github.com/stan-dev/example-models/wiki

To fit ‘standard’ Bayesian GLMs and GLMMs, use rstanarm



RStanArm

Bayesian applied regression modeling via Stan

RStanArm allows users to specify models via the customary R commands, where

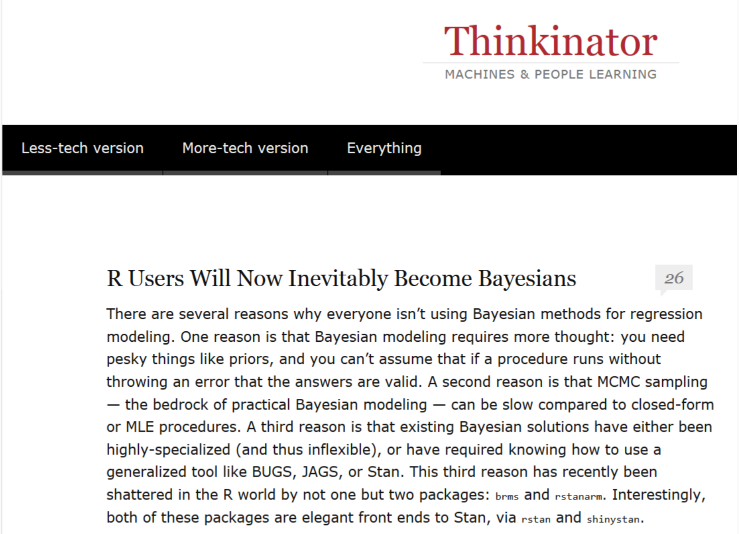
- models are specified with formula syntax,
- data is provided as a data frame, and
- additional arguments are available to specify priors.

Bayesian modelling in one line of code!

```r
library(rstanarm)
easybayes <- stan_glm(height ~ dbh.c, data = trees, family = gaussian,
                      prior_intercept = normal(30, 30),
                      prior = normal(0, 10))
```

and results make sense :)

```
stan_glm(formula = height ~ dbh.c, family = gaussian, data = trees,
    prior = normal(0, 10), prior_intercept = normal(30, 30))

Estimates:
            Median MAD_SD
(Intercept) 34.7    0.1
dbh.c        0.6    0.0
sigma        4.1    0.1

Sample avg. posterior predictive
distribution of y (X = xbar):
         Median MAD_SD
mean_PPD 36.5    0.2

Observations: 1000  Number of unconstrained parameters: 3
```



R Users Will Now Inevitably Become Bayesians

There are several reasons why everyone isn't using Bayesian methods for regression modeling. One reason is that Bayesian modeling requires more thought: you need pesky things like priors, and you can't assume that if a procedure runs without throwing an error that the answers are valid. A second reason is that MCMC sampling — the bedrock of practical Bayesian modeling — can be slow compared to closed-form or MLE procedures. A third reason is that existing Bayesian solutions have either been highly-specialized (and thus inflexible), or have required knowing how to use a generalized tool like BUGS, JAGS, or Stan. This third reason has recently been shattered in the R world by not one but two packages: `brms` and `rstanarm`. Interestingly, both of these packages are elegant front ends to Stan, via `rstan` and `shinystan`.

END

Slides and source code available at

https://github.com/Pakillo/Bayes-R-JAGS-intro