Introductory Bayesian Course

Dr. Joseph L. Thorley

October 20th, 2014

Contents

1 Background

The purpose of these course notes is to introduce participants to Bayesian analysis with R, RStudio and JAGS. It is assumed that participants are familiar with R and RStudio as covered in the Introductory R Course notes at http://www.poissonconsulting.ca/course/2014/09/12/an-introduction-to-r-course.html.

1.1 Licence

1.2 Installation

If you haven't already done so, download the the most recent version of the R base distribution binary for your platform from http://cran.r-project.org/ and install using the default options. Next download and install RStudio from http://www.rstudio.com/products/rstudio/download/ using the default options. Then, download JAGS from http://sourceforge.net/projects/mcmc-jags/files/JAGS/ and install with the default options.

To make sure you have all the required packages execute the following code at the command line

```
install.packages("devtools", quiet = TRUE)
library(devtools)

install.packages("dplyr", quiet = TRUE)
library(dplyr)

install.packages("scales", quiet = TRUE)
```

```
library(scales)

install.packages("ggplot2", quiet = TRUE)
library(ggplot2)

install.packages("rjags", quiet = TRUE)
install_github("poissonconsulting/tulip@v0.0.11")
install_github("poissonconsulting/datalist@v0.4")
install_github("poissonconsulting/juggler@v0.1.3")

install_github("poissonconsulting/jaggernaut@v1.8.5")
library(jaggernaut)
```

1.3 Bayesian vs. Frequentist Statistical Analysis

Statistical analysis uses probability models to provide bounded estimates of parameter values (θ) from the data (y).

There are two primary approaches to statistical analysis: Bayesian and frequentist. As far as a frequentist is concerned the best estimates of θ are those values that maximise the *likelihood* which is the probability of the data given the estimates, i.e., $p(y|\theta)$. A Bayesian on the other hand chooses the values with the highest *posterior* probability - that is to say the probability of the estimates given the data , i.e., $p(\theta|y)$.

Exercise 1 *Which criterion makes the most sense to you? And why?*

There are several major consequences of this choice.

One major consequence is that Bayesians need to provide a *prior* probability distribution for θ because (as **Bayes** theorem demonstrates) $P(\theta|y) \propto P(y|\theta)P(\theta)$. Bayesians consider this an advantage because prior information can be incorporated into an analysis while frequentists consider it subjective. In most cases Bayesians use *low-information* priors which have negligible effect on the posteriors.

A second major consequence is that frequentist 95% *confidence intervals* (CIs) are expected to include the actual value of θ 95% of the time. Bayesian 95% *credible intervals* (CRIs), on the other hand, have a 95% probability of including the actual value of θ. The difference is subtle but important. For example, the reliability of frequentist confidence intervals depend on *sufficient* sample size - Bayesian credible intervals do not.

A third major consequence is that although likelihood functions can be derived for many models (and therefore used to quickly find the maximum likelihood (ML) estimates), it is typically not possible to to calculate the posterior probabilities. As a result the posterior probabilities have to be sampled using Markov Chain Monte Carlo (MCMC) algorithms such as Gibbs Sampling.

1.4 Gibbs Sampling

Consider the case where the parameters $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$ then Gibbs Sampling proceed as follows

Step 1 Choose starting *initial* values $\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_k^{(0)}$

Step 2 Sample $\theta_1^{(1)}$ from $p(\theta_1|\theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, y)$

Sample $\theta_2^{(1)}$ from $p(\theta_2|\theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, y)$

...

Sample $\theta_k^{(1)}$ from $p(\theta_k|\theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{k-1}^{(1)}, y)$

Step 3 *Iterate* step 2 thousands (or millions) of times to obtain a sample from $p(\theta|y)$.

1.5 JAGS and BUGS

Programming an efficient Gibbs Sampler for a particular model is outside the scope of most research projects. Fortunately, JAGS (which stands for Just Another Gibbs Sampler) can take a dataset and a model specified in the simple but flexible BUGS language (which stands for Bayesian Analysis Using Gibbs Sampling) and perform iterative sampling for us.

In order to do this we will use the `jaggernaut` package to talk to the standalone JAGS program via the `rjags` package.

1.6 Biased Coin

Now consider the example of a biased coin which has a 75% chance of giving a tail. We can generate a dummy data set of 10 values using the R function `rbinom()` which provides pseudo-random samples from a binomial distribution.

```
set.seed(563)  # specifies start point for pseudo-random number generation
y <- rbinom(n = 10, size = 1, prob = 0.75)  # generates vector of 1000 samples
y
```

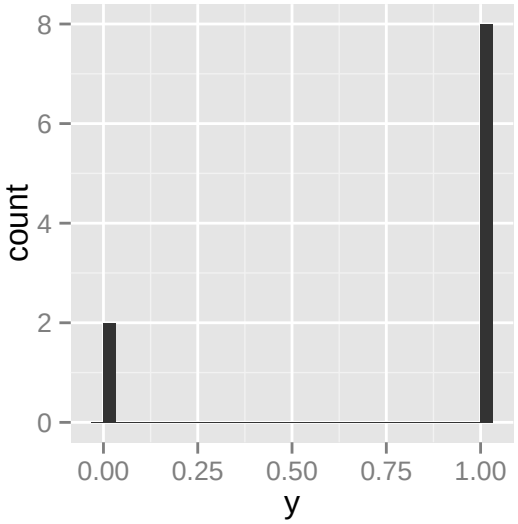
```
##  [1] 1 1 1 0 1 0 1 1 1 1
```

```
mean(y)  # calculates mean of vector
```

```
## [1] 0.8
```

```
qplot(y, geom = "histogram")  # plots histogram of vector
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



In order to estimate the underlying probability of throwing a head we specify the following model in the BUGS language and save it as an object of class `jags_model`.

```
mcoin1 <- jags_model("model { # start of BUGS model code
  theta ~ dunif(0, 1) # uniform prior distribution between 1 and 0 for theta
  for(i in 1:length(y)) { # for loop over the values 1, 2, ..., length of y
    y[i] ~ dbin(theta, 1) # assumed statistical distribution for observed throws
  } # end of for loop
} # end of BUGS model code
")
```

Exercise 2 *Plot theta's prior distribution as a histogram. You will likely find the* ***runif*** *and* ***qplot*** *functions useful.*

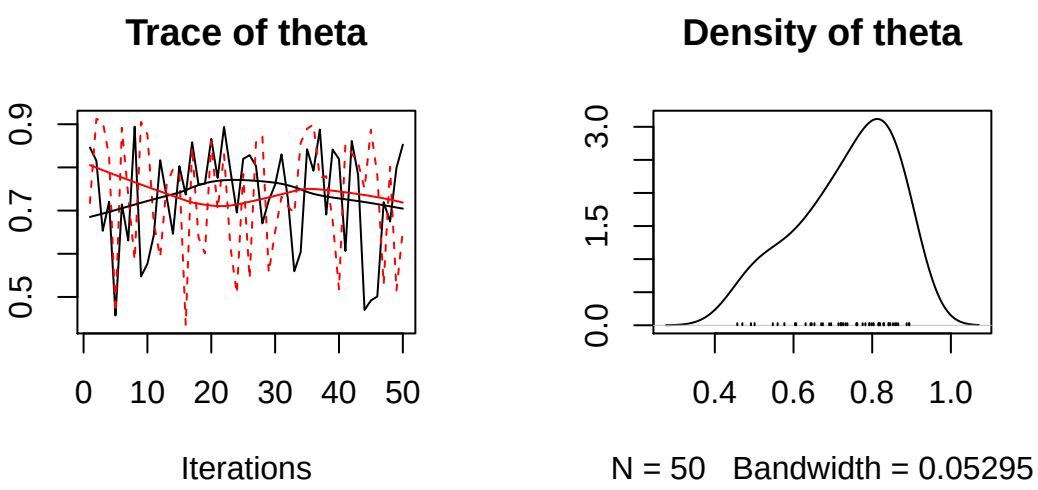
Then we call JAGS using `jaggernaut` in `debug` mode to generate 100 samples from θ's posterior probability distribution.

With these settings `jaggernaut`

- generates two chains of 50 iterations in length

```
acoin1 <- jags_analysis(mcoin1, data = data.frame(y = y), mode = "debug")
```

```
plot(acoin1)
```




```
coef(acoin1)
```

```
##        estimate     lower     upper      sd error significance
## theta 0.7326673 0.4666409 0.9023939 0.12546    30            0
```

The model output indicates that the point estimate is 0.73, the 95% credible interval is 0.47 to 0.9, the standard deviation is 0.13, the percent relative error is 30 and the significance is 0.

If previous studies indicated that the coin was biased towards tails then we could adjust the prior distribution as follows.

```
mcoin2 <- jags_model("model {
  theta ~ dunif(0.5, 1) # new prior distribution
  for(i in 1:length(y)) {
    y[i] ~ dbin(theta, 1)
  }
} ")
```

Exercise 3 *How does the new prior affect the posterior distribution?*

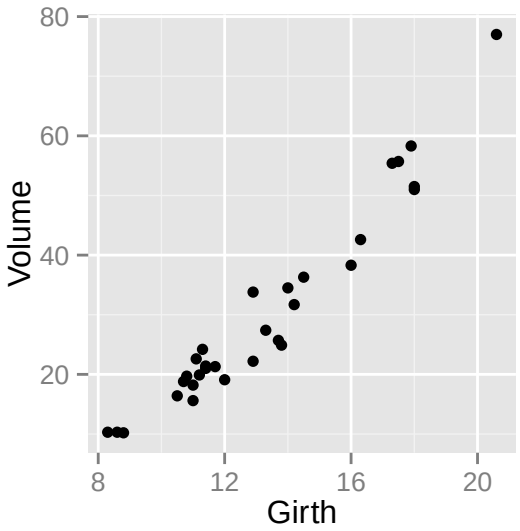
1.6.1 Recap

- The BUGS model code defines the posterior distributions and relationships between parameters.
- Based on the data and model definition, JAGS uses MCMC algorithms to iteratively sample from the posterior distributions of the monitored parameters.
- The posterior distributions can be summarised in terms of a point estimate and 95% credible intervals.

1.7 Black Cherry Trees

The `trees` data set in the dataset package provides information on the girth and volume of 31 black cherry trees.

```
qplot(x = Girth, y = Volume, data = trees)
```



Algebraically, the linear regression of `Volume` against `Girth` can be defined as follows

$$Volume_i = \alpha + \beta * Girth_i + \epsilon_i$$

where α is the intercept and β is the slope and the error terms (ϵ_i) are drawn from a normal distribution with an standard deviation of σ.

The model can be defined as follows in the BUGS language where `<-` indicates a *deterministic* as opposed to *stochastic* node (which is indicated by `~`).

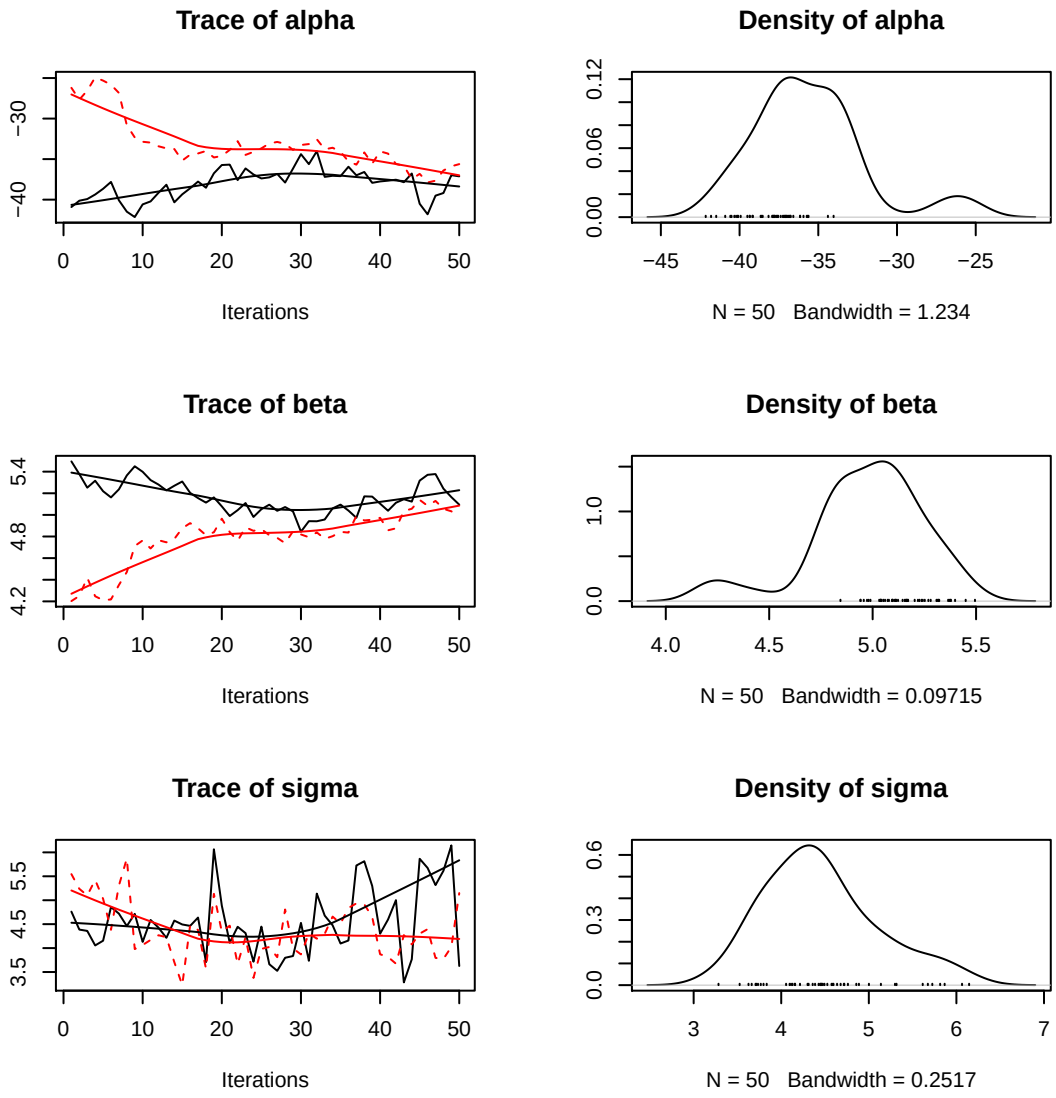
```
mtrees1 <- jags_model("model {
  alpha ~ dnorm(0, 50^-2) # normal prior on intercept
  beta ~ dnorm(0, 50^-2) # normal prior on slope
  sigma ~ dunif(0, 50) # positive uniform prior on residual variation

  for(i in 1:length(Volume)) {
    eVolume[i] <- alpha + beta * Girth[i] # expected volume
    Volume[i] ~ dnorm(eVolume[i], sigma^-2) # observed volume drawn from normal
  }
}")
```

The standard deviations of the normal distributions are raised to the power of `-2` because (for historical reasons) Bayesians quantify variation in terms of the *precision* (τ) as opposed to the variance (σ^2) or standard deviation (σ) where $\tau = 1/\sigma^2$.

The resultant trace plots and coefficients are as follows

```
atrees1 <- jags_analysis(mtrees1, data = datasets::trees, mode = "debug")
```

```
plot(atrees1)
```




```
coef(atrees1)
```

```
##         estimate      lower      upper      sd error significance
## alpha -35.640570 -41.212317 -26.002739 3.56340    21            0
## beta    4.974180   4.231665   5.388757 0.27483    12            0
## sigma   4.454759   3.447690   5.868135 0.64585    27            0
```

Exercise 4 *What do the trace plots suggest to you?*

The $\hat{R}$ metric uses the within-chain and between-chain variances to quantify the extent to which the chains have converged on the same distribution. Although a value of `1.0` indicates complete convergence, for most purposes an $\hat{R} < 1.1$ is considered sufficient.

The function `rhat` can return the $\hat{R}$ value for each monitored parameter.

```
rhat(atrees1, parm = "fixed", combine = FALSE)
```

```
##       rhat
## alpha 2.45
## beta  2.41
## sigma 1.03
```

Lack of convergence suggests that the MCMC samples may not be representative of the posterior distributions. To produce more representative samples, the user can attempt to reparameterise the model to increase chain mixing and/or increase the number of iterations.

1.7.1 Chain Mixing

Poor chain mixing, which manifests as high levels of autocorrelation, is often caused by cross correlations between parameters. Examination of the previous trace plots indicates that `alpha` and `beta` are cross correlated. The following code sets the `select` terms of `mtrees1` model so that it specifies the required variables in the input data with the `+` suffix on `Girth` indicating that it is to be centered (`x - mean(x)`) before being passed to JAGS.

```
select(mtrees1) <- c("Volume", "Girth+")
```

Exercise 5 *Does centering* `Girth` *improve chain mixing?*

Note if you ever want to examine the actual data being passed to JAGS set the modify_data term of your `jags_model` object to be a simple function that prints and returns its one argument

```
modify_data(mtrees1) <- function (data) { print(data); data }
```

1.7.2 Iterations

Convergence can also be achieved by increasing the number of iterations. This is easy to achieve using jaggernaut, just switch to `report` mode and set `niters = 10^4`

```
opts_jagr(mode = "report")
atrees3 <- jags_analysis(mtrees1, data = datasets::trees, niters = 10^4)
```

```
## $Volume
##  [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4
## [14] 21.3 19.1 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4
## [27] 55.7 58.3 51.5 51.0 77.0
##
## $Girth
##  [1] -4.9483871 -4.6483871 -4.4483871 -2.7483871 -2.5483871 -2.4483871
##  [7] -2.2483871 -2.2483871 -2.1483871 -2.0483871 -1.9483871 -1.8483871
## [13] -1.8483871 -1.5483871 -1.2483871 -0.3483871 -0.3483871  0.0516129
```

```
## [19]  0.4516129  0.5516129  0.7516129  0.9516129  1.2516129  2.7516129
## [25]  3.0516129  4.0516129  4.2516129  4.6516129  4.7516129  4.7516129
## [31]  7.3516129
##
## Analysis converged (Rhat:1)
```

With these settings `jaggernaut`

- undergoes an adaptive phase of xx iterations to maximize sampling efficiency (chain mixing)
- generates three chains of 10^4 iterations in length.
- discards the first half of each chain as burnin
- thins each chain so that the total number of MCMC samples for each monitored parameter is as close as possible to but not less than 10^3
- tests for convergence under the criterion that $\hat{R} < 1.1$
- if convergence has not been achieved it discards the current samples as burnin and doubles the number of iterations before thinning again
- it repeats this up to three times or until convergence is achieved

For more information on the possible modes and options type `?opts_jagr`.

Exercise 6 *How does switching to report mode with 10^3 iterations affect the trace plots with and without centering Girth?*

1.7.3 Predictions and Residuals

Many researchers estimate fitted values, predictions and residuals by monitoring additional nodes in their model code. The disadvantages of this approach are that:

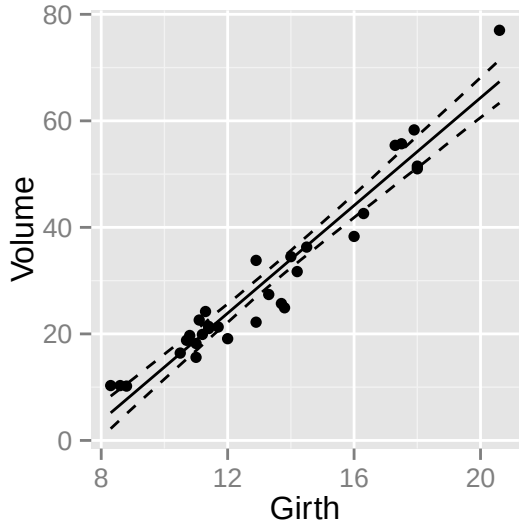
- the model code becomes more complicated.
- the MCMC sampling takes longer.
- adding deried parameters requires a model rerun.
- the table of parameter estimates becomes unwiedly.

`jaggernaut` overcomes these problems by allowing derived parameters to be defined in a separate chunk of BUGS code as demonstrated below.

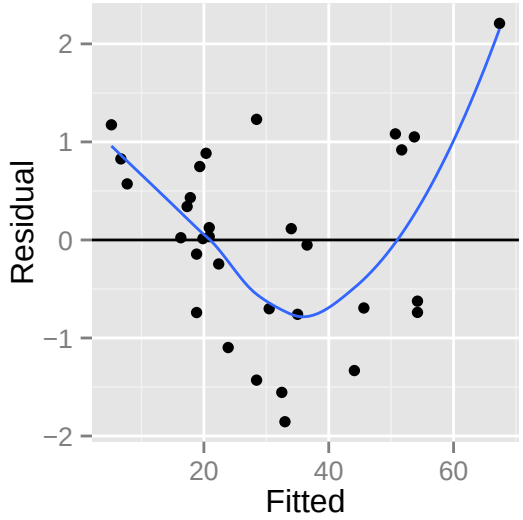
```
derived_code(mtrees1) <- "data {
  for(i in 1:length(Volume)) {
    prediction[i] <- alpha + beta * Girth[i]
  }
  residual <- (Volume - prediction) / sigma
}"

atrees4 <- jags_analysis(mtrees1, data = datasets::trees, niters = 10^4)

prediction <- predict(atrees4, newdata = "Girth")
```

```
gp <- ggplot(data = prediction, aes(x = Girth, y = estimate))
gp <- gp + geom_point(data = dataset(atrees3), aes(y = Volume))
gp <- gp + geom_line()
gp <- gp + geom_line(aes(y = lower), linetype = "dashed")
```

```
gp <- gp + geom_line(aes(y = upper), linetype = "dashed")
gp <- gp + scale_y_continuous(name = "Volume")

print(gp)
```




```
residuals <- residuals(atrees4)
fitted <- fitted(atrees4)

qplot(fitted$estimate, residuals$estimate, xlab = "Fitted", ylab = "Residual") +
    geom_hline(yintercept = 0) + geom_smooth(se = FALSE)
```



As discussed in the R course notes the relationship between Volume and Girth is expected to be allometric because the cross-sectional area at an given point scales to the square of the girth (circumference).

Exercise 7 *Fit an allometric relationship by log transforming Volume and Girth within the model code (in this case do not center Girth). Does the residual plot suggest an improvement in the model?*

Exercise 8 *Does adding log transformed Height improve the model?*

The basic ANCOVA (analysis of covariance) model includes one categorical and one continous predictor variable without interactions. It can be expressed algebraically as follows

$$y_{ij} = \alpha_i + \beta * x_j + \epsilon_{ij}$$

where α_i is the intercept for the i$^{\text{th}}$ group mean and β is the slope and the error terms (ϵ_{ij}) are independently drawn from a normal distribution with standard deviation σ.

The following code fits the basic ANCOVA model to the ToothGrowth data and plots the model's predictions and residuals.

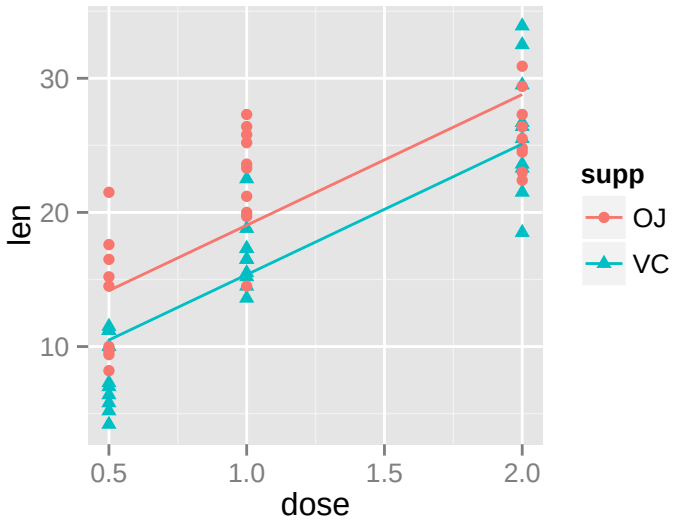
```
mtg1 <- jags_model("model {
  for(i in 1:nsupp) {
    alpha[i] ~ dnorm(0, 40^-2)
  }
  beta ~ dnorm(0, 20^-2)
  sigma ~ dunif(0, 20)

  for(i in 1:length(len)) {
    eLen[i] <- alpha[supp[i]] + beta * dose[i]
    len[i] ~ dnorm(eLen[i], sigma^-2)
  }
}",
derived_code  = " data{
  for(i in 1:length(len)) {
    prediction[i] <- alpha[supp[i]] + beta * dose[i]
  }
  residual <- (len - prediction) / sigma
}",
select = c("len", "dose+", "supp"))
```

```
atg1 <- jags_analysis(mtg1, data = datasets::ToothGrowth)
```

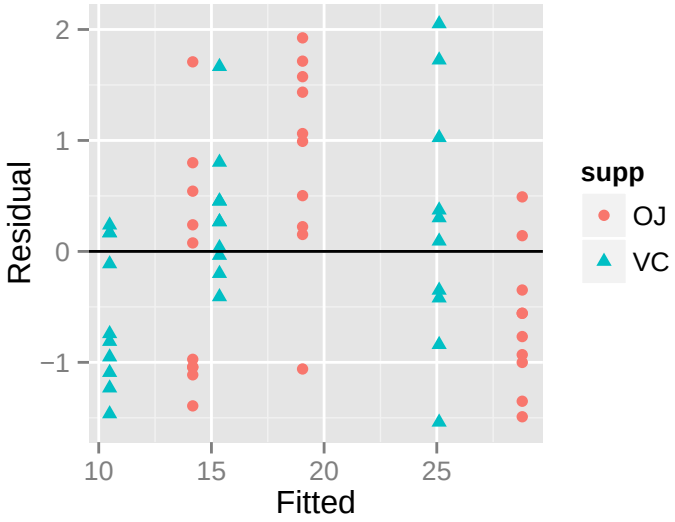
```
coef(atg1)
```

```
##           estimate     lower     upper      sd error significance
## alpha[1] 20.668716 19.092096 22.174057 0.78805     7            0
## alpha[2] 16.981615 15.462679 18.491222 0.78034     9            0
## beta      9.746653  8.025048 11.511744 0.91472    18            0
## sigma     4.328619  3.622054  5.238726 0.42046    19            0
```

```
prediction <- predict(atg1, newdata = c("supp", "dose"))

gp <- ggplot(data = prediction, aes(x = dose, y = estimate, color = supp,
    shape = supp))
gp <- gp + geom_point(data = dataset(atg1), aes(y = len))
gp <- gp + geom_line()
gp <- gp + scale_y_continuous(name = "len")

print(gp)
```




```
residuals <- residuals(atg1)
residuals$fitted <- fitted(atg1)$estimate

qplot(fitted, estimate, color = supp, shape = supp, data = residuals, xlab = "Fitted",
    ylab = "Residual") + geom_hline(yintercept = 0)
```



Exercise 9 *What does the plot of the residual versus fitted values suggest about the model fit?*

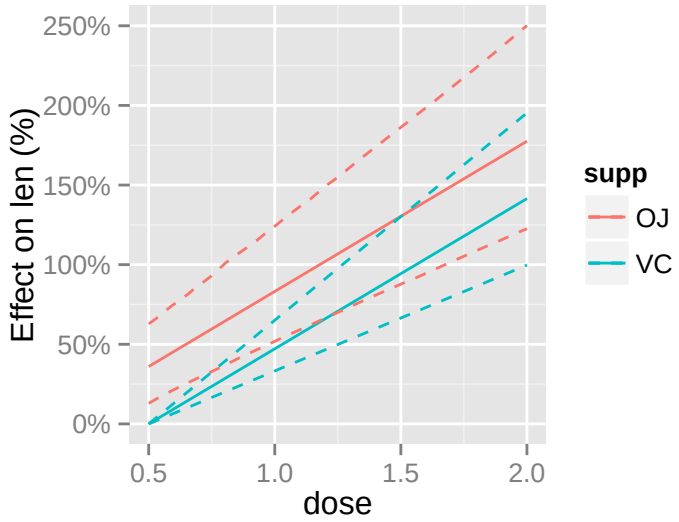
Exercise 10 *Is the effect of OJ significantly different from that of VC?*

Exercise 11 *Modify the new ANCOVA model to fit 1) a linear regression for dose, 2) an ANOVA for supp and 3) an ANCOVA with an interaction between dose and supp. Which model do you prefer?*

1.7.4 Effects Size

Often the results of an analysis are easier to understand when they are presented in terms of the percent change in the response. The following code predicts and plots the percent change in `len` relative to 0.5 mg of VC for the ToothGrowth data set .

```
prediction <- predict(atg1, newdata = c("supp", "dose"), base = data.frame(supp = "VC",
    dose = 0.5))

gp <- ggplot(data = prediction, aes(x = dose, y = estimate, color = supp,
    shape = supp))
gp <- gp + geom_line()
gp <- gp + geom_line(aes(y = lower), linetype = "dashed")
gp <- gp + geom_line(aes(y = upper), linetype = "dashed")
gp <- gp + scale_y_continuous(name = "Effect on len (%)", labels = percent)

print(gp)
```

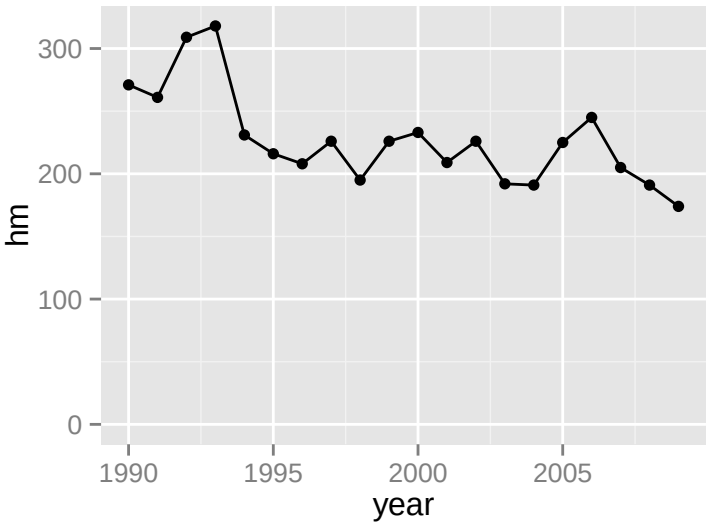


Exercise 12 *Plot the percent change in* `len` *for your preferred ToothGrowth model relative to 1 mg of OJ*

1.7.5 House Martins and Hierarchical Models

Consider the annual surveys of house martins in a small Swiss village from 1990 to 2009 from Kery and Schaub (xxxx).

```
data(hm)
qplot(year, hm, data = hm) + geom_line() + expand_limits(y = 0)
```



Algebraically a state-space population growth model to estimate the underlying population abundance N_t and growth rate r_t at time t can be written

$$log(N_{t+1}) = log(N_t) + r_t$$

$$r_t \sim N(\bar{r}, \sigma_r)$$

$$log(Count_t) = log(N_t) + \epsilon_t$$

$$\epsilon_t \sim N(0, \sigma_\epsilon)$$

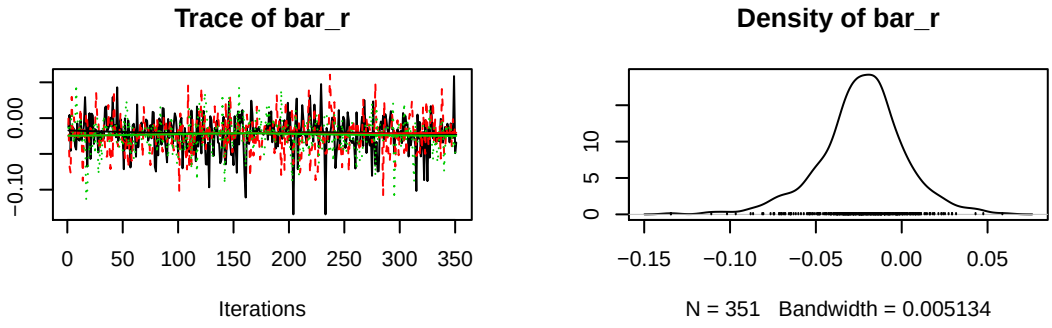
In the BUGS language the model can be written as follows.

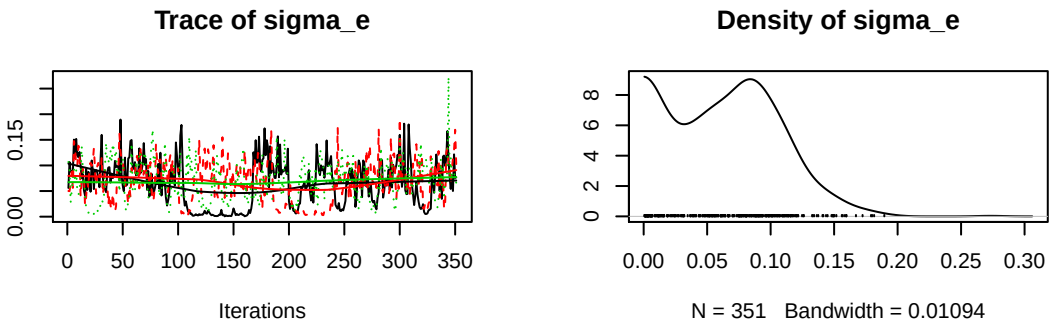
```
mhm1 <- jags_model("model {

  bar_r ~ dnorm(1, 10^-2)
  sigma_r ~ dunif(0, 1)
  sigma_e ~ dunif(0, 1)

  N[1] ~ dlnorm(5.6, 10^-2)
  for (i in 1:length(year)) {
    log(N[i+1]) <- log(N[i]) + r[i]
    r[i] ~ dnorm(bar_r, sigma_r^-2)
    hm[i] ~ dlnorm(log(N[i]), sigma_e^-2)
  }
}",
 derived_code = "data{
  for (i in 1:length(year)) {
    log(prediction[i]) <- log(N[year[i]])
  }
}",
random_effects = list(r = "year", N = "year"),
)
```

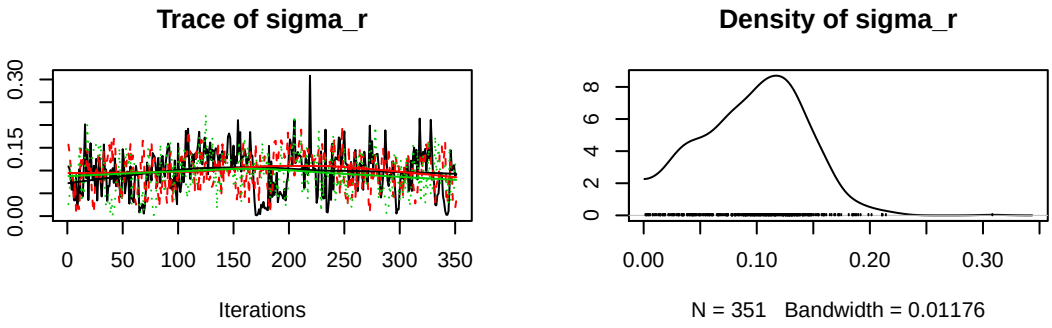
The specification of `r` and `N` as random effect means that by default they are excluded from the trace plots and table of coefficients.

```
hm$year <- factor(hm$year)
ahm1 <- jags_analysis(mhm1, data = hm, niters = 10^4)
```

```
plot(ahm1)
```



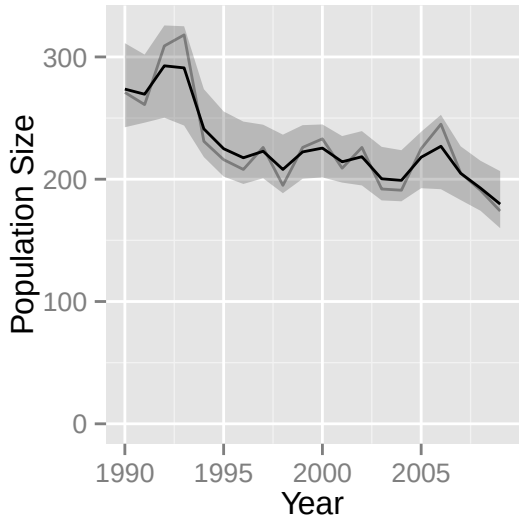





```
coef(ahm1)
```

```
##            estimate        lower      upper       sd error
## bar_r    -0.02298996 -0.076718286 0.02478937 0.024927   221
## sigma_e   0.06779232  0.002905339 0.15014148 0.041527   109
## sigma_r   0.09761384  0.011623019 0.17682532 0.044645    85
##          significance
## bar_r          0.2792
## sigma_e        0.0000
## sigma_r        0.0000
```

```
prediction <- predict(ahm1, newdata = "year")

gp <- ggplot(data = prediction, aes(x = as.integer(as.character(year)),
    y = estimate))
gp <- gp + geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 1/4)
gp <- gp + geom_line(data = dataset(ahm1), aes(y = hm), alpha = 1/3)
gp <- gp + geom_line()
gp <- gp + scale_x_continuous(name = "Year")
gp <- gp + scale_y_continuous(name = "Population Size")
gp <- gp + expand_limits(y = 0)

print(gp)
```



Exercise 13 *What is the probability that the population in 2015 will be less than that in 2009? Note you can produce the projections by simply appending six years of missing counts to the dataset*