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. If you are using Windows please install the latest version of Rtools from http://cran.r-project.org/bin/windows/Rtools/. And for OSX users make sure you have the latest version of Xcode https://developer.apple.com/xcode/.

To install all the required packages execute the following code at the R console.

```
install.packages("devtools", quiet = TRUE)
install.packages("dplyr", quiet = TRUE)
install.packages("ggplot2", quiet = TRUE)
install.packages("scales", quiet = TRUE)

library(devtools)
install_github("poissonconsulting/tulip@v0.0.11")
install_github("poissonconsulting/datalist@v0.4")
install_github("poissonconsulting/juggler@v0.1.3")
install_github("poissonconsulting/jaggernaut@v2.2.2")
```

If everything was successful, when you run the following code

```
library(jaggernaut)
model <- jags_model("model {
 bLambda ~ dlnorm(0,10^-2)
 for (i in 1:length(x)) {
   x[i]~dpois(bLambda)
 }
}")

summary(jags_analysis (model, data = data.frame(x = rpois(100,1))))
```

you should see something like

```
Analysis converged (rhat:1)

Model1:

Dimensions:
samples  chains
   1500       3

Convergence:
rhat
   1

Estimates:
         estimate     lower    upper       sd error significance
bLambda 0.8771658 0.7151765 1.062476 0.088936    20            0
```

1.3 R Script Headers

During the course you should begin R scripts with

```
library(dplyr)
library(ggplot2)
library(scales)
library(jaggernaut)
```

to load the required packages in the search path, and

```
graphics.off()
rm(list = ls())
```

to clean up the workspace and close any graphics windows.

2 Bayesian and 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., $p(\theta|y)$.

2.1 Coin Flips

Consider the case where $n = 10$ flips of a coin produce $y = 3$ tails. We can model this using a binomial distribution

$$y \sim dbin(\theta, n)$$

where θ is the probability of throwing a head.

2.1.1 Maximum Likelihood

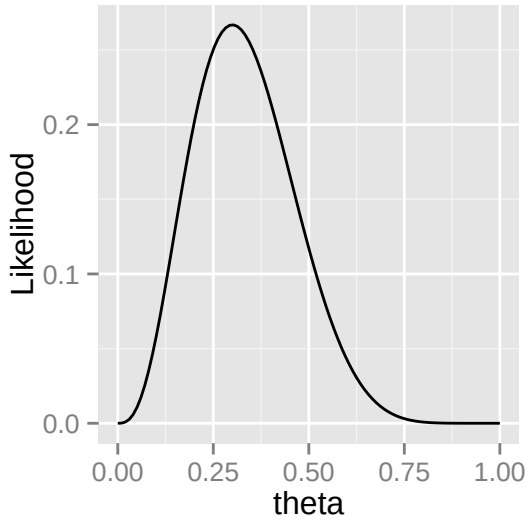
The likelihood for the binomial model is given by the following equation

$$p(y|\theta) = \binom{n}{y}\theta^y(1-\theta)^{n-y}$$

.

The likelihood values for different values of θ are therefore as follows

```
likelihood <- function(theta, n = 10, y = 3) {
    choose(n, y) * theta^y * (1 - theta)^(n - y)
}
theta <- seq(from = 0, to = 1, length.out = 100)

qplot(theta, likelihood(theta), geom = "line", ylab = "Likelihood", xlab = "theta")
```



The frequentist point estimate ($\hat{\theta}$) is the value of θ with the maximum likelihood (ML) value, which in this case is 0.3.

A 95% confidence interval (CI) can then be calculated using the asymptotic normal approximation

$$\hat{\theta} \pm 1.96\frac{1}{\sqrt{I(\hat{\theta})}}$$

where $I(\hat{\theta})$ is the expected second derivative of the log-likelihood at the estimate. This calculation is based on the assumption that the sample size is of sufficient size that the likelihood is normally distributed.

In the current case,

$$I(\hat{\theta}) = \frac{n}{\hat{\theta}(1-\hat{\theta})}$$

which gives a point estimate of 0.3 and lower and upper 95% confidence intervals of 0.02 and 0.58 respectively.

2.1.2 Posterior Probability

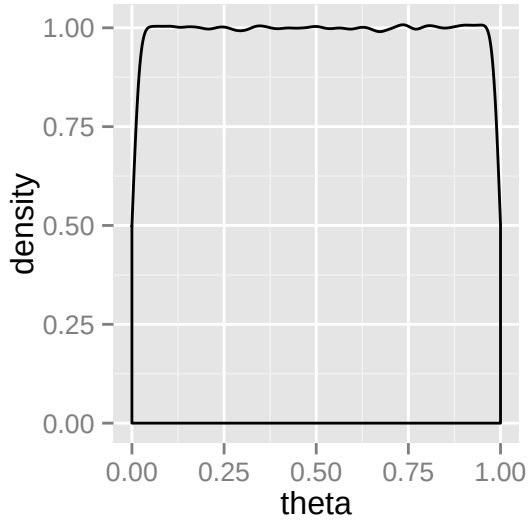
The posterior probability on the other hand is given by Bayes rule which states that

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

where $p(\theta)$ is the prior probability.

Bayesians consider the necessity to define prior probabilities 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 little influence on the posteriors. For example a uniform distribution with a lower limit of 0 and an upper limit of 1 ($dunif(0, 1)$) is commonly used for probabilities.

```
qplot(runif(10^6, 0, 1), geom = "density", xlab = "theta")
```



As it is generally not possible to calculate the posterior probability, the posterior probability distribution is sampled using Markov Chain Monte Carlo (MCMC) algorithms such as Gibbs Sampling.

2.1.2.1 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 for $\theta_1^{(0)}$ and $\theta_2^{(0)}$

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

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

Typically this is performed for two or more independent chains.

2.2 JAGS and BUGS

Programming an efficient MCMC algorithm 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 MCMC 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.

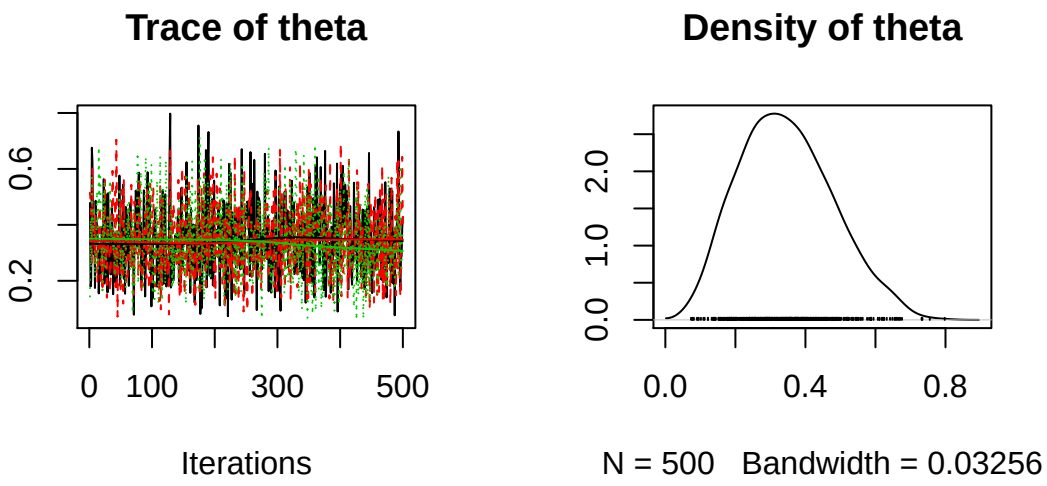
First we need to specify the underlying probability model in the BUGS language and save it as an object of class `jags_model`.

```
model1 <- jags_model("model {
  theta ~ dunif(0, 1)
  y ~ dbin(theta, n)
}")
```

then we call JAGS using `jaggernaut` in the default report mode to generate a total of $\geq 10^3$ samples using three chains from θ's posterior probability distribution.

```
data <- data.frame(n = 10, y = 3)
analysis1 <- jags_analysis(model1, data = data)
```

```
plot(analysis1)
```




```
coef(analysis1)
```

```
##        estimate     lower     upper     sd error significance
## theta 0.3455298 0.1216057 0.6272665 0.1326    73            0
```

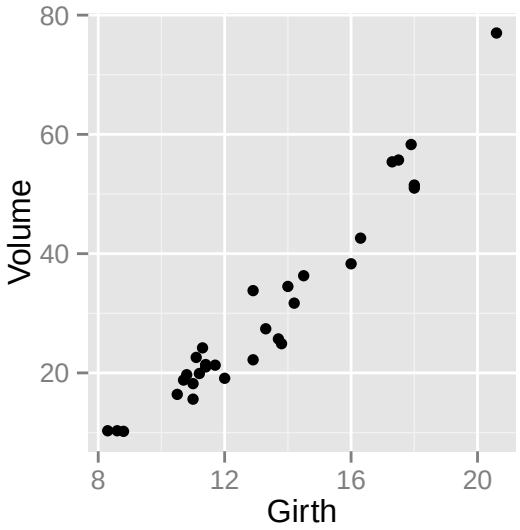
The model output indicates that the point estimate (in this case the mean of the samples) is 0.35 and the 95% credible interval (in this case the 2.25th and 97.75th percentiles) is 0.12 to 0.63. The model output also indicates that the posterior probability distribution has a standard deviation (sd) of 0.13. The significance and error values are discussed later.

Exercise 1 *Previous studies indicate that the coin was definitely biased towards tails. Modify the prior distribution accordingly and rerun the above model. How does the posterior distribution change?*

3 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 independently 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 `~`).

```
model1 <- jags_model("model {
  alpha ~ dnorm(0, 50^-2)
  beta ~ dnorm(0, 10^-2)
  sigma ~ dunif(0, 10)

  for(i in 1:length(Volume)) {
    eMu[i] <- alpha + beta * Girth[i]
    Volume[i] ~ dnorm(eMu[i], sigma^-2)
  }
}")
```

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 $\tau = 1/\sigma^2$.

3.1 Parallel Chains

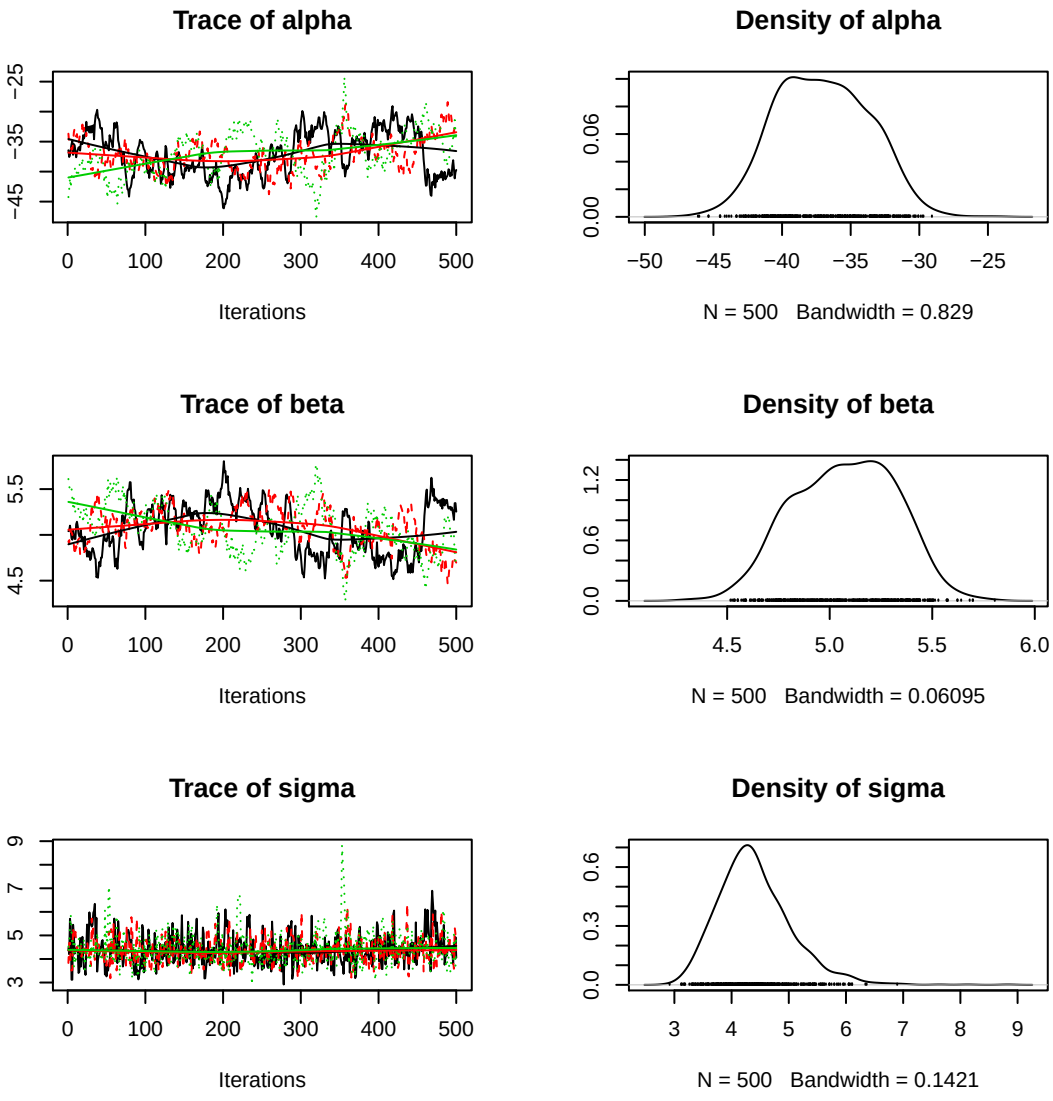
To reduce the analysis time, MCMC chains can be run on parallel processes. In `jaggernaut` this achieved using the `registerDoParallel` function and by setting the parallel option to `TRUE`. This only needs to be done once at the start of a session. I add it to my R script header.

```
if (getDoParWorkers() == 1) {
    registerDoParallel(3)
    opts_jagr(parallel = TRUE)
}
```

The resultant trace plots and coefficients for the `trees` analysis are as follows.

```
data(trees)
analysis1 <- jags_analysis(model1, data = trees)
```

```
plot(analysis1)
```




```
coef(analysis1)
```

```
##         estimate      lower      upper      sd error significance
## alpha -37.049511 -43.082476 -30.853140 3.37630    17            0
## beta    5.072903   4.600128   5.504592 0.24827     9            0
## sigma   4.404543   3.370720   5.911371 0.64671    29            0
```

Exercise 2 *What do you notice about the trace plots? The output of* `auto_corr(analysis1)` *and* `cross_cor(analysis1)` *might give you some clues.*

3.2 Convergence

The $\hat{R}$ convergence 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 full convergence, $\hat{R} \leq 1.05$ is typically considered sufficient for most papers and ≤ 1.10 for most reports.

Exercise 3 *What is the R-hat value for each of the parameters in the current model? Use the* `convergence` *function with* `combine = FALSE`*.*

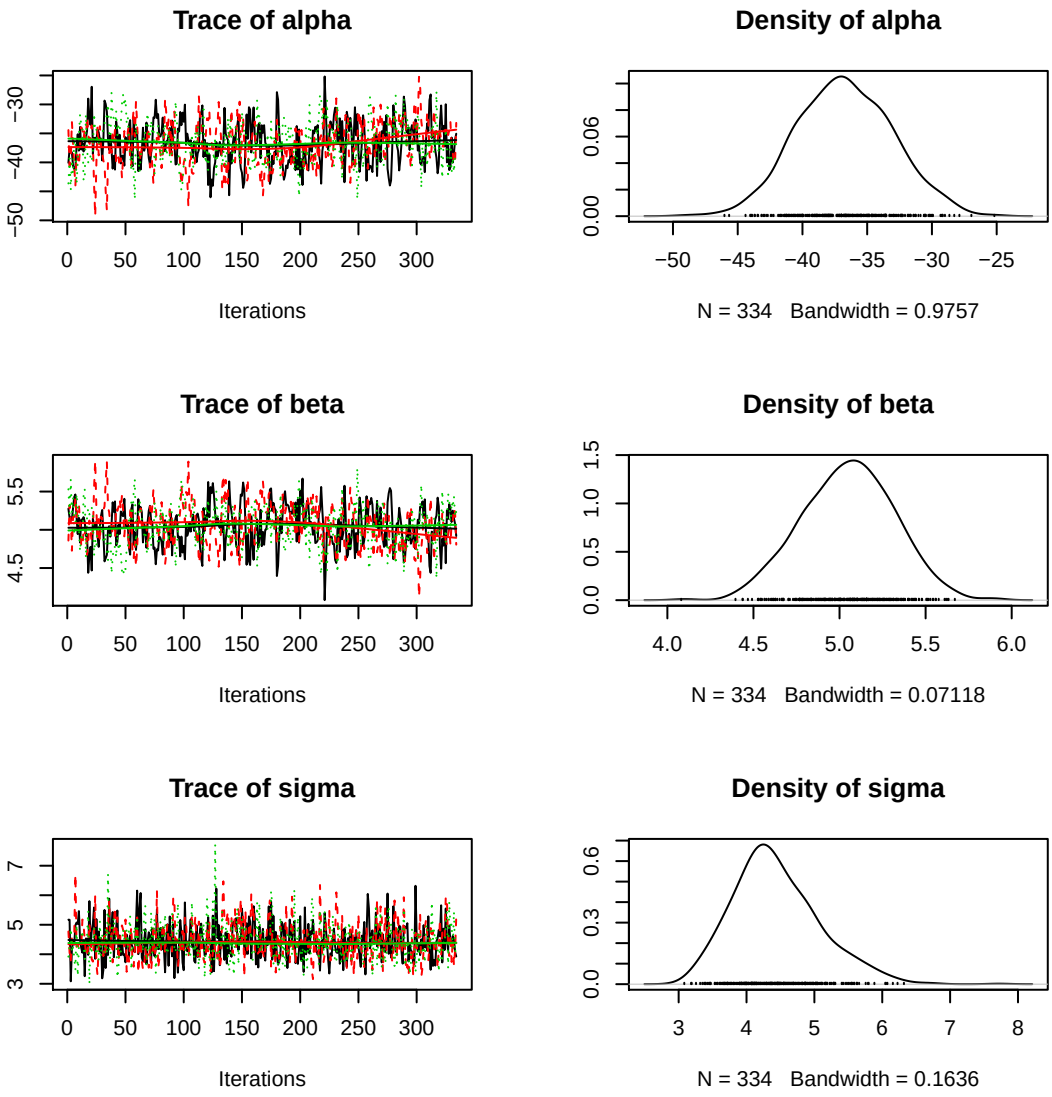
Lack of convergence suggests that the MCMC samples may not be representative of the posterior distributions.

3.3 Iterations

Convergence can often be improved by simply increasing the number of iterations.

```
analysis1 <- jags_analysis(model1, data = trees, niters = 10^4)
```

```
plot(analysis1)
```



3.4 Modes

In the default `opts_jagr()` `report` mode

- an adaptive phase of 100 iterations is undertaken to maximize sampling efficiency (chain mixing)
- three chains of `niters` iterations in length are generated.
- the first half of each chain is discarded as burn in.
- each chain os thinned so that the total number of MCMC samples for each monitored parameter is $\geq 10^3$.
- convergence is tested under the criterion that $\hat{R} < 1.1$
- if convergence has not been achieved the current samples are discarded as burn in and the number of iterations doubled before thinning again.
- this is repeated up to three times or until the convergence threshold of 1.1 is achieved.

To see the current mode and option settings type `opts_jagr()` and for more information type `?opts_jagr`.

Exercise 4 *Call* `jags_analysis` *with* `niters = 10^2` *and* `mode = 'paper'` *How many updates are required for convergence to be achieved?*

3.5 Chain Mixing

Cross-correlations between parameters, which cause poor chain mixing (i.e., high auto-correlation and poor convergence), can sometimes be eliminated or reduced by reparameterising the model. In the current model, `Girth` can be centered, i.e., `Girth - mean(Girth)`, in the BUGS code or by setting the `select_data` argument to be `select_data(model1) <- c("Volume", "Girth+")`.

Exercise 5 *What is the effect of centering* `Girth` *on the trace plots?*

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(model1) <- function (data) { print(data); data }
```

3.6 Derived Parameters

Many researchers estimate fitted values and residuals, generate predictions and perform posterior predictive checks by monitoring additional nodes in their model code.

The disadvantages of this approach are that:

- the model code becomes cluttered.
- the MCMC sampling takes longer.
- adding derived parameters requires a model rerun.
- the table of parameter estimates becomes unwieldy.

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

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

    simulated[i] ~ dnorm(prediction[i], sigma^-2)

    D_observed[i] <- log(dnorm(Volume[i], prediction[i], sigma^-2))
    D_simulated[i] <- log(dnorm(simulated[i], prediction[i], sigma^-2))
  }
  residual <- (Volume - prediction) / sigma
  discrepancy <- sum(D_observed) - sum(D_simulated)
}"
```

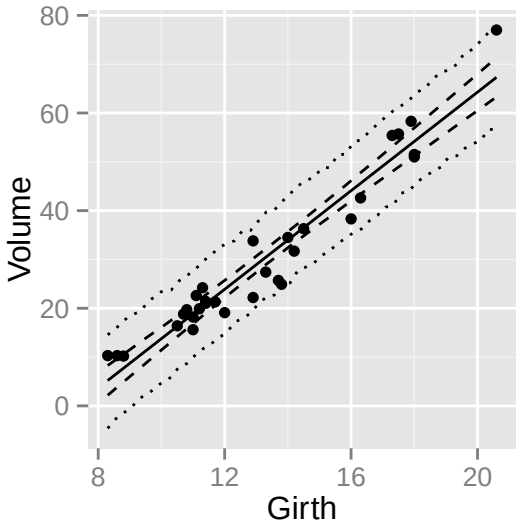
3.6.1 Predictions

To better understand a model and/or inform management decisions it is generally useful to plot a model's predictions.

In the following example, the predict function is used to estimate the `Volume` with 95% CRIs and 95% Prediction Intervals (PRIs) across the range of the observed values of `Girth`.

```
prediction <- predict(analysis1, newdata = "Girth", derived_code = derived_code)
simulated <- predict(analysis1, parm = "simulated", newdata = "Girth", derived_code = derived_code)
```

```
gp <- ggplot(data = prediction, aes(x = Girth, y = estimate))
gp <- gp + geom_point(data = dataset(analysis1), 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 + geom_line(data = simulated, aes(y = lower), linetype = "dotted")
gp <- gp + geom_line(data = simulated, aes(y = upper), linetype = "dotted")
gp <- gp + scale_y_continuous(name = "Volume")

print(gp)
```

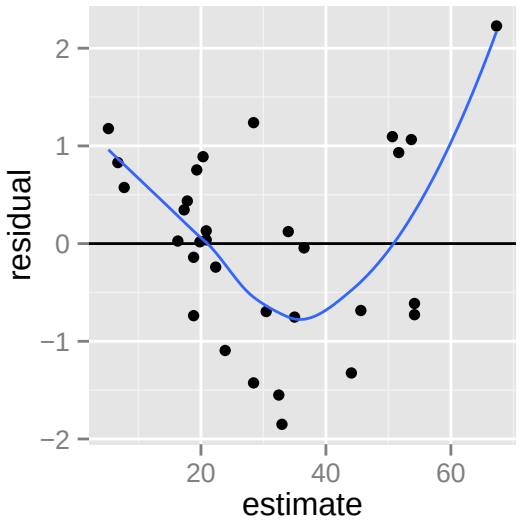


Exercise 6 *The* `newdata` *argument in the* `predict` *function can also take a* `data.frame`*. What is the 95% Prediction Interval for the* `Volume` *of a tree of* `Girth` *8?*

3.6.2 Residuals

The model assumes that the error terms are independently drawn from a normal distribution. It is possible to assess the adequacy of this assumption by plotting the residual variation.

```
fitted <- fitted(analysis1, derived_code = derived_code)
fitted$residual <- residuals(analysis1, derived_code = derived_code)$estimate

qplot(estimate, residual, data = fitted) + geom_hline(yintercept = 0) +
    geom_smooth(se = FALSE)
```



Exercise 7 *What does the current residual plot suggest to you about model adequacy?*

3.6.3 Posterior Predictive Checks

A complementary approach to assessing model adequacy is to simulate data given the model parameters and compare it to the observed data. Any systematic differences indicate potential failings of the model.

```
predictive_check(analysis1, derived_code = derived_code)
```

```
##              estimate     lower    upper     sd error significance
## discrepancy 0.2253773 -10.84623 11.26278 5.5865  4905       0.9641
```

Exercise 8 *What does the posterior predictive check suggest to you about model adequacy?*

3.7 Allometry

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).

Expressed as an allometric relationship the model becomes

$$Volume_i = \alpha * Girth_i^{\beta} * \epsilon_i$$

which can be reparameterised as a linear regression by log transforming `Volume` and `Girth`.

$$log(Volume_i) = \alpha + \beta * log(Girth_i) + \epsilon_i$$

Variables can be log transformed in the model code or in the `select_data` argument, i.e., `select_data(model1) <- c("log(Volume)", "log(Girth)")`.

Exercise 9 *Fit the linear regression of the allometric model to the* `trees` *data set. Is the model fit improved?*

Exercise 10 *Is there any support for adding* `log(Height)` *to the model?*

3.8 Significance Values

The significance value in the jaggernaut table of coefficients is twice the probability that the posterior distribution spans zero. As such it represents the Bayesian equivalent of a frequentist two-sided p-value (Greenland and Poole 2013). By definition, parameters that represent standard deviations will have a significance value of 0 (as they must be greater than zero).

3.9 Error Values

The error value in the table of coefficients is the *percent relative error*, which is half the credible interval as a percent of the point estimate, i.e.,

$$error = (upper - lower) * 0.5/estimate * 100$$

.

Standard deviations with a uniform prior distribution that is not updated by the data have error values of 0.95. As a general rule, I question the informativeness of parameters representing standard deviations which have an error value ≥ 0.8.

4 Tooth Growth

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.

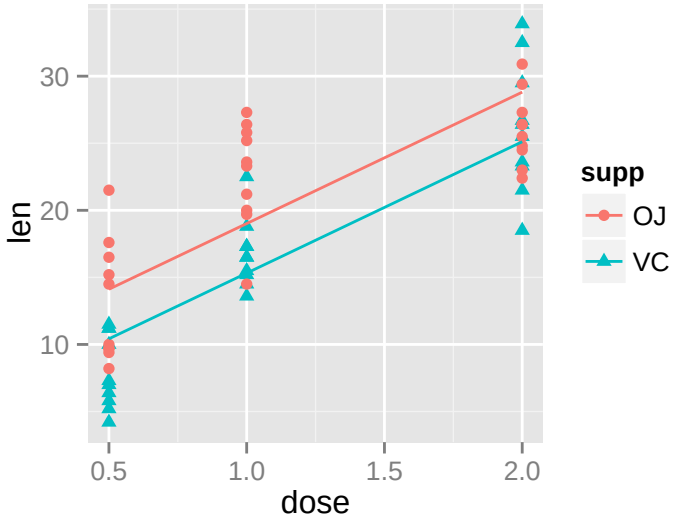
```
model1 <- jags_model("model {
  for(i in 1:nsupp) {
    alpha[i] ~ dnorm(0, 40^-2)
  }
  beta ~ dnorm(0, 20^-2)
  sigma ~ dunif(0, 20)

  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
}")
```

```
data(ToothGrowth)
analysis1 <- jags_analysis(model1, data = ToothGrowth)
```

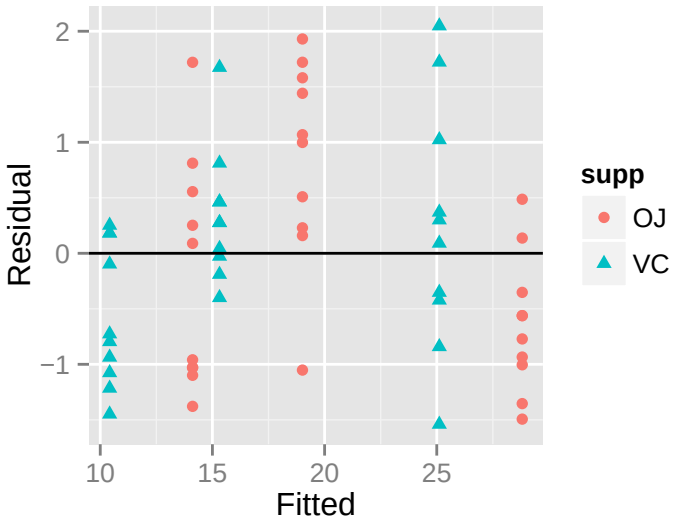
```
coef(analysis1)
```

```
##          estimate    lower     upper      sd error significance
## alpha[1] 9.217635 6.597608 11.925747 1.36400    29            0
## alpha[2] 5.520765 2.680145  8.217476 1.39560    50            0
## beta     9.795862 7.949522 11.775864 0.96174    20            0
## sigma    4.328632 3.627076  5.187427 0.40366    18            0
```

```
prediction <- predict(analysis1, newdata = c("supp", "dose"))

gp <- ggplot(data = prediction, aes(x = dose, y = estimate, color = supp,
    shape = supp))
gp <- gp + geom_point(data = dataset(analysis1), aes(y = len))
gp <- gp + geom_line()
gp <- gp + scale_y_continuous(name = "len")

print(gp)
```




```
residuals <- residuals(analysis1)
residuals$fitted <- fitted(analysis1)$estimate

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



Exercise 11 *Is the effect of `OJ` significantly different from that of `VC`?*

Exercise 12 *What does the residual plot suggest about the model fit?*

4.1 Multiple Models

The linear regression on `dose` is given by

```
model2 <- jags_model("model {
  alpha ~ dnorm(0, 40^-2)
  beta ~ dnorm(0, 20^-2)
  sigma ~ dunif(0, 20)

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

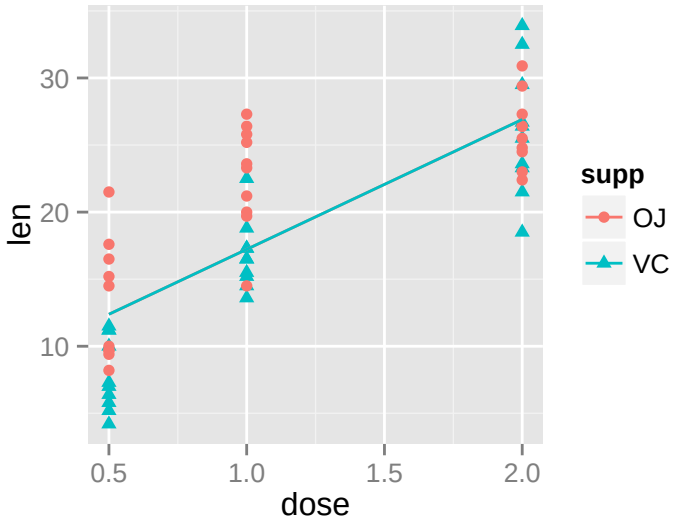
The `combine` function allows multiple `jags_model` objects which can have unique `mode_id`'s to be combined into a single object.

```
model_id(model1) <- "ANCOVA"
model_id(model2) <- "regression"

models <- combine(model1, model2)
analyses <- jags_analysis(models, data = datasets::ToothGrowth)
```

```
coef(analyses)
```

```
## $ANCOVA
##          estimate    lower     upper      sd error significance
## alpha[1] 9.200638 6.537561 11.837303 1.31830    29            0
## alpha[2] 5.457261 2.969853  8.031193 1.29460    46            0
## beta     9.831129 8.073856 11.536372 0.88563    18            0
## sigma    4.342960 3.633665  5.349963 0.43104    20            0
##
## $regression
##       estimate    lower     upper      sd error significance
## alpha 7.541517 5.290187  9.890651 1.18780    31            0
## beta  9.681888 7.951726 11.400260 0.89943    18            0
## sigma 4.696242 3.954851  5.584868 0.42560    17            0
```

```
prediction <- predict(analyses, newdata = c("supp", "dose"), model_id = "regression")

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

print(gp)
```

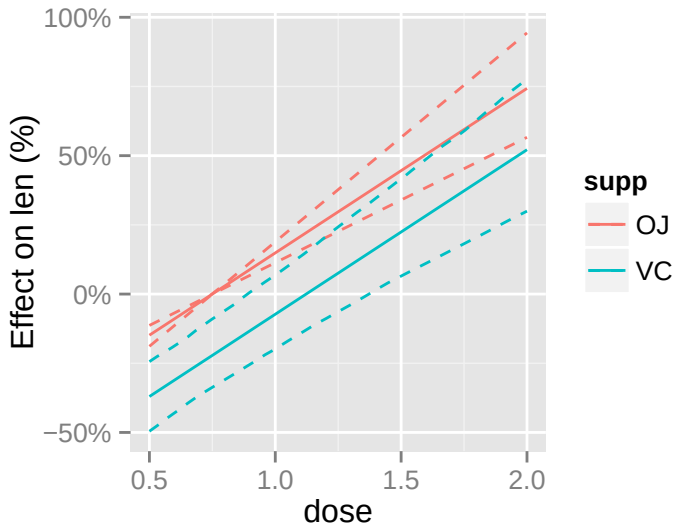


Exercise 13 *Fit 1) the ANCOVA, 2) the linear regression, 3) the ANOVA and 4) the ANCOVA with an interaction between dose and supp models and plot their predictions. Which model do you prefer?*

4.1.1 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 0.75 mg of Vitamin C.

```
prediction <- predict(analysis1, newdata = c("supp", "dose"), base = data.frame(supp = "OJ",
    dose = 0.75))

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 14 *Plot the percent change in* `len` *for all four models relative to 0.5 mg of Vitamin C*

5 Peregrine Falcon Population

Consider the peregrine falcon population data `?peregrine`. the following code regresses the number of reproductive `Pairs` on `Year`.

```
model1 <- jags_model("model {
  alpha ~ dnorm(0, 100^-2)
  beta ~ dnorm(0, 100^-2)
  sigma ~ dunif(0, 100)
  for(i in 1:length(Pairs)) {
    ePairs[i] <- alpha + beta * Year[i]
    Pairs[i] ~ dnorm(ePairs[i], sigma^-2)
  }
}",
derived_code = "data {
  for(i in 1:length(Pairs)) {
    prediction[i] <- alpha + beta * Year[i]
  }
}",
select_data = c("Pairs", "Year+"))
```

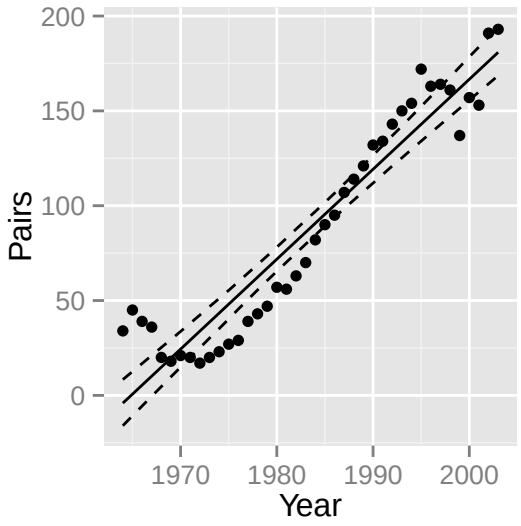
```
data(peregrine)

analysis1 <- jags_analysis(model1, data = peregrine)
```

```
coef(analysis1)
```

```
##        estimate     lower     upper      sd error significance
## alpha 90.768596 84.512934 96.969213 3.16510     7            0
## beta   4.742526  4.207775  5.286656 0.27885    11            0
## sigma 19.628955 15.539507 24.966134 2.44740    24            0
```

```
prediction <- predict(analysis1)
```

```
gp <- ggplot(data = prediction, aes(x = Year, y = estimate))
gp <- gp + geom_point(data = dataset(analysis1), aes(y = Pairs))
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 = "Pairs")
gp <- gp + expand_limits(y = 0)

print(gp)
```



Exercise 15 *Do you consider the model to be adequate?*

5.1 Log-link Function

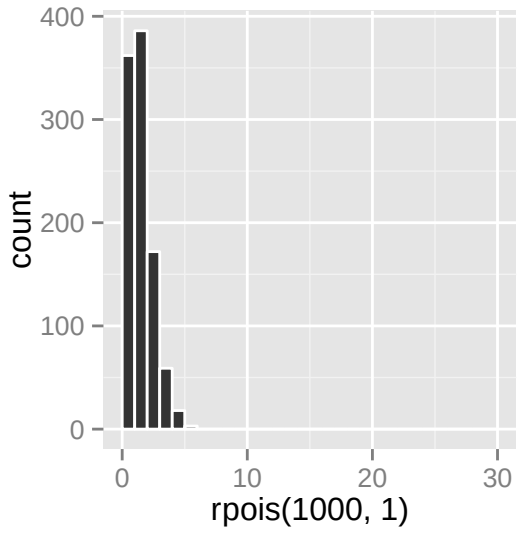
When a response variable cannot take negative values, the use of a log-link function ensures that the expected value must be positive.

Exercise 16 *Replace* `ePairs[i] <-` *with* `log(ePairs[i]) <- ...`. *How does the log-link function alter the model?*

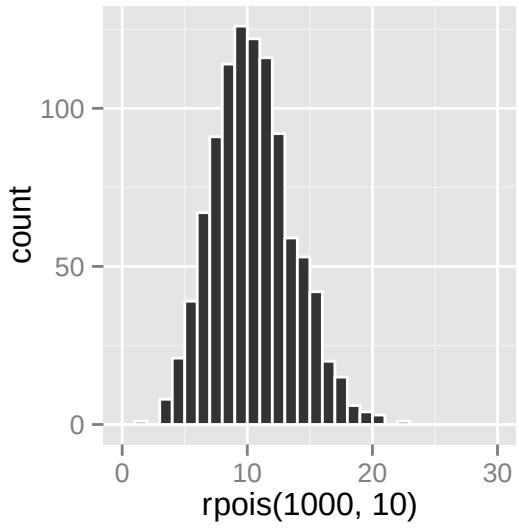
5.2 Poisson Distribution

The Poisson distribution models counts about a positive mean expected value. It can only assume discrete non-negative values and has a variance equal to the mean.

```
qplot(rpois(1000, 1), geom = "histogram", binwidth = 1, color = I("white"),
    xlim = c(0, 30))
```




```
qplot(rpois(1000, 10), geom = "histogram", binwidth = 1, color = I("white"),
    xlim = c(0, 30))
```



Exercise 17 *Next, replace* `Pairs[i] ~ dnorm(ePairs[i], sigma^-2)` *with* `Pairs[i] ~ dpois(ePairs[i])`. *How does the assumption of Poisson distributed counts alter the model?*

5.3 Polynomial Regression

Curvature in a predictor variable can be modelled by allowing the influence of a variable to vary with the variable. For example, the following model code fits a second-order polynomial on `Year`.

```
model_code <- "model {
  alpha ~ dnorm(0, 100^-2)
  beta ~ dnorm(0, 100^-2)
  beta2 ~ dnorm(0, 100^-2)
  for(i in 1:length(Pairs)) {
    log(ePairs[i]) <- alpha + beta * Year[i]  + beta2 * Year[i]^2
    Pairs[i] ~ dpois(ePairs[i])
  }
}"
```

Exercise 18 *Does it improve the model?*

Exercise 19 *Fit a third-order polynomial (`beta3 * Year[i]^3`). How does it improve the model?*

Exercise 20 *Use the third-order polynomial model to predict the number of breeding pairs in 2006. How confident are you in your answer?*

5.4 State-Space Population Growth Model

An alternative to a polynomial would be to fit a state-space population growth model which explicity estimates the step changes in the underlying population

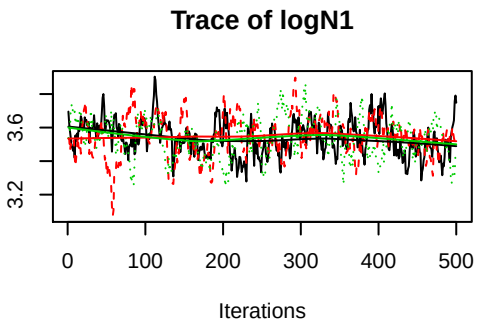
$$log(N_{t+1}) = log(N_t) + r_t$$
$$r_t \sim dnorm(\bar{r}, \sigma_r)$$

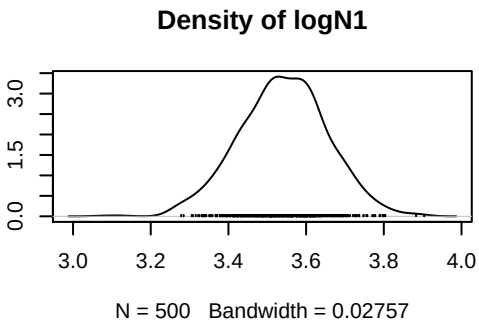
```
model6 <- jags_model("model {
  mean_r ~ dnorm(0, 1^-2)
  sd_r ~ dunif(0, 1)

  logN1 ~ dnorm(0, 10^-2)
  logN[1] <- logN1
  for(i in 2:nYear) {
    logN[i] <- logN[i-1] + r[i-1]
  }

  for(i in 1:nYear) {
    r[i] ~ dnorm(mean_r, sd_r^-2)
    Pairs[i] ~ dpois(exp(logN[i]))
  }
}",
derived_code = "data {
  for(i in 1:length(Pairs)) {
    log(prediction[i]) <- logN[Year[i]]
  }
}",
modify_data = function (data) {
  stopifnot(!is.unsorted(data$Year))
  data
},
select_data = c("Pairs", "Year"),
random_effects = list(r = "Year", logN = "Year"))
```

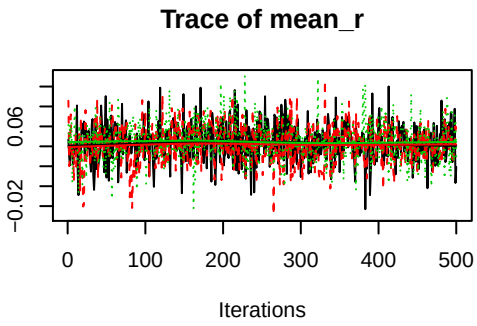
The specification of `r` and `logN` as random effect means that by default they are excluded from the trace plots and table of coefficients.

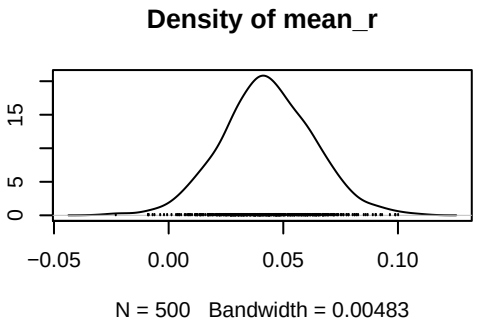
```
peregrine$Year <- factor(peregrine$Year)
analysis6 <- jags_analysis(model6, data = peregrine)
```

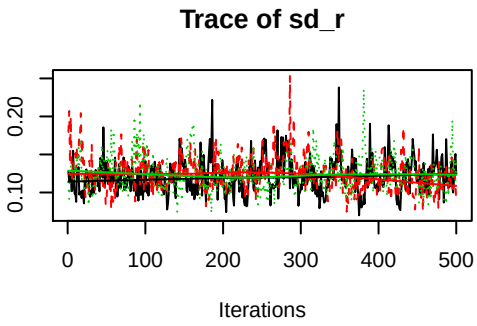
```
plot(analysis6)
```

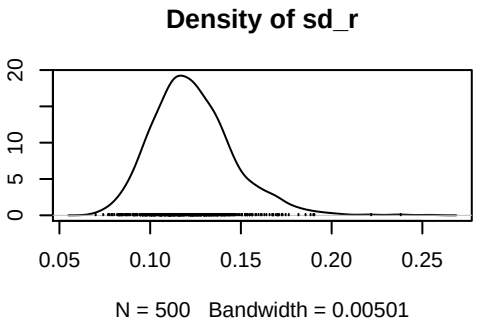










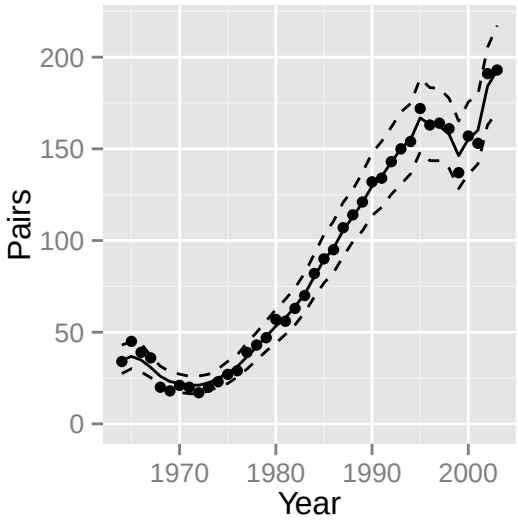



```
coef(analysis6)
```

```
##          estimate       lower      upper       sd error significance
## logN1  3.54107001 3.312540218 3.76074442 0.113720     6        0.000
## mean_r 0.04319152 0.004474976 0.08362089 0.020046    92        0.032
## sd_r   0.12359139 0.086573719 0.17384713 0.022592    35        0.000
```

```
prediction <- predict(analysis6)
```

```
gp <- ggplot(data = prediction, aes(x = as.integer(as.character(Year)),
    y = estimate))
gp <- gp + geom_point(data = dataset(analysis6), aes(y = Pairs))
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_x_continuous(name = "Year")
gp <- gp + scale_y_continuous(name = "Pairs")
gp <- gp + expand_limits(y = 0)

print(gp)
```



Exercise 21 *Plot the state-space population growth rate model predictions in terms of the percent change since 1970. What is the estimated percent change in the population in 2003 compared to 1970?*

Exercise 22 *What is the probability that the population in 2008 will be less than that in 2003? Note you can produce the projections by simply appending five years of missing counts to the dataset.*

5.5 Logistic Model

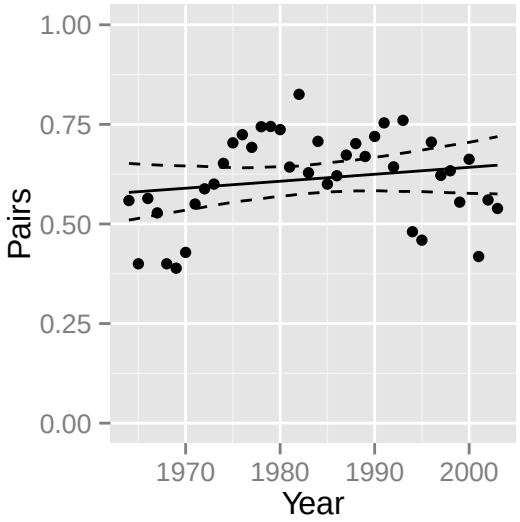
Now let us consider the number of `Pairs` that successfully reproduced (`R.pairs`).

```
data(peregrine)

peregrine$Proportion <- peregrine$R.Pairs/peregrine$Pairs

model7 <- jags_model("model {\n  alpha ~ dnorm(0, 1^-2)\n  beta ~ dnorm(0, 1^-2)\n  sigma ~ dunif(0, 1)\
    derived_code = "data {\n  for(i in 1:length(Proportion)) {\n    prediction[i] <- alpha + beta * Yea
    select_data = c("Proportion", "Year+"))
```

```
analysis7 <- jags_analysis(model7, data = peregrine)
```

```
prediction <- predict(analysis7)

gp <- ggplot(data = prediction, aes(x = Year, y = estimate))
```

```
gp <- gp + geom_point(data = dataset(analysis7), aes(y = Proportion))
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_x_continuous(name = "Year")
gp <- gp + scale_y_continuous(name = "Pairs")
gp <- gp + expand_limits(y = c(0, 1))

print(gp)
```



Exercise 23 *Fit a second-order polynomial regression to the proportion of successful pairs*

Exercise 24 *Add a logistic-link function, i.e., logit(eProportion[i]) <-*

Exercise 25 *Replace the normal distribution* `Proportion[i] ~ dnorm(eProportion[i], sigma^-2)` *with the binomial distribution* `R.Pairs ~ dbin(eProportion[i], Pairs[i])`

Exercise 26 *Replace the polynomial with a first-order difference auto-regressive term*

Exercise 27 *Add extra-binomial variation through a normally distributed random effect on* `logit(eProportion[i])`

References

Greenland, Sander, and Charles Poole. 2013. "Living with P Values: Resurrecting a Bayesian Perspective on Frequentist Statistics." *Epidemiology* 24 (1): 62–68. doi:10.1097/EDE.0b013e3182785741.