Practical 1 - an introduction to R and JAGS

Doug McNeall

10th May 2016

Introduction

Welcome to Practical 1, an introduction to using R and JAGS. In this practical we'll

- Learn some basic functions of R
- Learn some time series basics in R
- Learn how to fit some simple statistical models in JAGS and R

You should follow and run the commands shown in the grey boxes. At various points you will see a horizontal line in the text which indicates a question you should try to answer, like this:

Exercise X
What words does the following command print to the console?

```
print("Hello World")
```

If you are a confident R user, you might want to skip some questions: that's fine, we'd you to get as much as possible out of the course, rather than completing every last thing.

If you get stuck, please get our attention and we will try to help! There aren't prepared answers to all of these questions so keep you own record as you go. At the end of the practical are harder questions which you can attempt if you get through all of the material. If you find any mistakes in the document please let us know.

You can run the code from these practicals by loading up the `.Rmd` file in the same directory in Rstudio. This is an R markdown document containing all the text. Feel free to add in your own answers, or edit the text to give yourself extra notes. You can also run the code directly by highlighting the relevant code and clicking `Run`.

Getting started with R

R is an open-source **object oriented** language for doing statistics. All the data you load in, the internal functions and any functions you write are **objects** with **attributes**. We'll get a feel for what this means as we go along.

If you need help with a function, you can type `?functionname` into the command line, for example `?print`

QuickR is a really nice reference website for R, and StackOverflow has loads of code for more difficult queries.

Loading some data

Exercise 1

1. Use the `read.csv()` function [a variant of `read.table()`] to load up the data in file `'https://raw.githubusercontent.`
2. Use the `str()` and `head()` functions to interrogate the resulting R object. What class of object is it?

Indexing in R

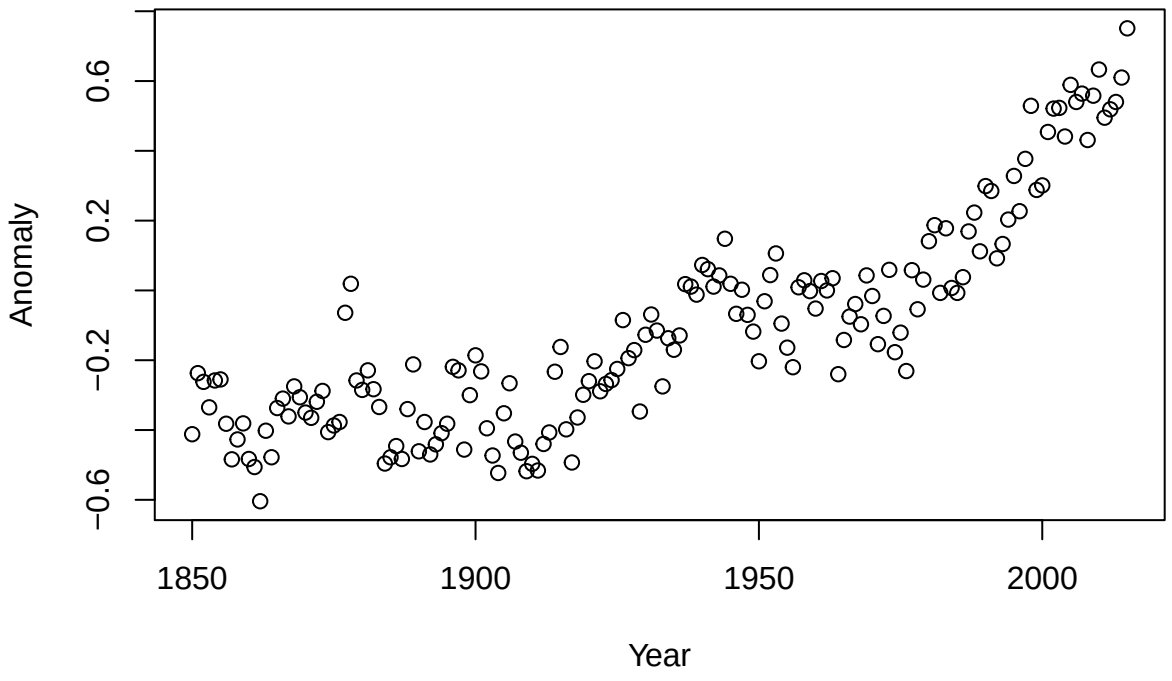
You can use a number of ways to index into a data frame (and other objects) in R. For example:

```
dat = hadcrut$Anomaly
dat = hadcrut[,2]
dat = hadcrut[,'Anomaly']
```

should all give the same result.

A nice way to refer to the data in object while keeping code easy to understand is using the `with()` function. For example, you can plot the data combining the `plot()` and the `with()` functions:

```
with(hadcrut,plot(Year, Anomaly))
```



Exercise 2

1. Is there a way to make the structure of the timeseries clearer? [Hint: look for the "type" option].
2. You can convert the Anomaly data to a timeseries object using the function `ts()` if you like. To keep things simple, you'll need to use the `start`, `end` or possibly `frequency` attributes.

Autocorrelation in the time series

Is there a relationship between a value of the timeseries and the next time step? You should be able to see such a relationship in a **lagged scatter plot** (lag plot), plotting the value of the time series y_i against previous value y_{i-1}.

Exercise 3

1. Create a lagged scatter plot of the time series, to see if there is a hint of autocorrelation in the data set. You can create a scatter plot in R using `plot(x,y)`, where `x` and `y` are your variables.

R contains a great many useful statistical functions - examples for checking for autocorrelation are `acf()` and `pacf()`. We'll come to these in a later module.

BONUS excercises

1. Apply a 30 year moving window to the hadcrut data set. What do you see? Is this a real signal?
2. Apply a similar window to an AR1 process (with drift). Do you see something similar? Fit a lowess (local regression smoother) to the HadCRUT data.

An introduction to JAGS

In this part of the practical, we will:

1. Fit a linear model with JAGS to simulated and real data
2. Fit a logistic regression with JAGS to simulated data

Estimating the coefficients of the linear model

A similar linear model is written:

$y = \alpha + \beta x_i + \epsilon$

Where α is a constant, β is the regression coefficient and ϵ is gaussian noise with variance σ^2, and is independent across observations:

$$\epsilon \sim N(0, \sigma^2)$$

In R, we can write this as:

```
# Likelihood:
# y_t ~ N(alpha + beta * x[i], sigma^2)
# Prior
# alpha ~ N(0,100) - vague priors
# beta ~ N(0,100)
# sigma ~ U(0,10)
```

Simulate data from the linear model

Here is some R code that you can run to simulate data from the linear model.

```
T = 50
alpha = 2
beta = 3
#sigma = 1
sigma = 3
# Set the seed so this is repeatable
set.seed(123)
x = sort(runif(T, 0, 10)) # Sort as it makes the plotted lines neater
y = rnorm(T, mean = alpha + beta * x, sd = sigma)
```

Exercise 4

1. Change the parameters in the model and plot the results, to make sure you are happy with what each does.

Input a JAGS model

We'd like to fit a Bayesian model to the simulated data. Here is how we specify the JAGS model code in R:

```
model_code = '
model
{
  # Likelihood
  for (t in 1:T) {
    y[t] ~ dnorm(alpha + beta * x[t],tau)
  }

  # Andrews Priors
  #alpha ~ dnorm(0.0,0.01)
  #beta ~ dnorm(0.0,0.01)
  #tau <- 1/pow(sigma,2) # Turn precision into standard deviation
  #sigma ~ dunif(0.0,10.0)
  #Dougs priors
  alpha ~ dnorm(0.0,3)
  beta ~ dnorm(0.0,3)
  tau <- 1/pow(sigma,2) # Turn precision into standard deviation
  sigma ~ dunif(0.0,10.0)
}
'
```

We specify a likelihood and priors for each of the parameters. We specify a prior for sigma, but need to calculate precision ($1/\sigma^2$) for the likelihood.

Bonus exercise

1. What do the priors look like? Plot them up.

We set up the data as a list object so that we can get it into the `jags` function.

```
# Set up the data as a list object, and inspect it
model_data = list(T = T, y = y, x = x)
str(model_data)
```

```
## List of 3
##  $ T: num 50
##  $ y: num [1:50] -2.32 5.78 3.83 1.67 9.93 ...
##  $ x: num [1:50] 0.246 0.421 0.456 1.029 1.388 ...
```

We choose the model parameters that we'd like the function to watch and output results for.

```
model_parameters =  c("alpha","beta","sigma")
```

Then we run the jags model with some appropriate choices.

```r
model_run = jags(data = model_data,
                 parameters.to.save = model_parameters,
                 model.file=textConnection(model_code),
                 n.chains=4, # Number of different starting positions
                 n.iter=1000, # Number of iterations
                 n.burnin=200, # Number of iterations to remove at start
                 n.thin=2) # Amount of thinning
```

```
## module glm loaded

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 50
##    Unobserved stochastic nodes: 3
##    Total graph size: 214
##
## Initializing model
```

Simulated results

The jags function has created a sequence of samples from the posterior distribution of the parameters that we asked it to save.

Exercise 5

1. Use the `names()` function to examine the structure of the `model_run` object.
2. Use the `plot()`, print() and `trace_plot()` functions on your `model_run` object to check the output.
3. Are the true values inside the 95% CI (confidence interval)?
4. Look at the R-hat values - they need to be close to 1 if convergence has been achieved.
5. How close is the posterior mean regression line to the true value?

Bonus exercises

1. With the simulated data, experiment with changing the value of T when creating the data? What happens to the posterior distribution of the parameters as N gets bigger?
2. Try experimenting with the priors. Suppose that you *knew* that beta was negative and used the prior beta ~ dunif(-2,-1). What happens to the posterior mean lines?

Some real data - Sea level rise

Church and White (2011) estimated sea level rise from the late 19th to the early 21st century, splicing tide guage and satellite altimetry data together.

Load the data set using the following code:

```
sea_level = read.csv('https://raw.githubusercontent.com/andrewcparnell/tsme_course/master/data/church_a
```

Exercise 6

1. What is the rate of sea level rise for the whole data set? Ignore the errors for the moment and run a linear regression model.
2. Plot the residuals through time. What do you notice? What simple alteration to the linear regression could you make to better capture the sea level rise?

A logistic regression example

As a bonus, we'll look at a logistic regression example. You'll need the `boot` package for the logit transform.

```
library(boot) # Package contains the logit transform
```

Here is the R notation for the model that we'll fit in this example.

```
# y_t = binomial (often binary) response variable for observation t=1,...,N
# x_{1t} = first explanatory variable for observation t
# x_{2t} = second " " " " " " " "
# p_t = probability of y_t being 1 for observation t
# alpha = intercept term
# beta_1 = parameter value for explanatory variable 1
# beta_2 = parameter value for explanatory variable 2

# Likelihood
# y_t ~ Binomial(K,p_t), or Binomial(1,p_t) if binary
# logit(p_t) = alpha + beta_1 * x_1[t] + beta_2 * x_2[t]
# where logit(p_i) = log( p_t / (1 - p_t ))
# Note that p_t has to be between 0 and 1, but logit(p_t) has no limits

# Priors - all vague
# alpha ~ normal(0,100)
# beta_1 ~ normal(0,100)
# beta_2 ~ normal(0,100)
```

And here is some R code to simulate from the model.

```
T = 100
set.seed(123)
x_1 = sort(runif(T,0,10))
x_2 = sort(runif(T,0,10))
alpha = 1
beta_1 = 0.2
beta_2 = -0.5
logit_p = alpha + beta_1 * x_1 + beta_2 * x_2
p = inv.logit(logit_p)
y = rbinom(T,1,p)
```

Exercise 7

1. What is the effect of of `x_1` and `x_2` on `y` ?

Here is the Jags code to fit the model to the simulated data.

```
model_code = '
model
{
  # Likelihood
  for (t in 1:T) {
    y[t] ~ dbin(p[t], K)
    logit(p[t]) <- alpha + beta_1 * x_1[t] + beta_2 * x_2[t]
  }

  # Priors
  alpha ~ dnorm(0.0,0.01)
  beta_1 ~ dnorm(0.0,0.01)
  beta_2 ~ dnorm(0.0,0.01)
}
'
```

Exercise 8

1. Fit the logistic regression model to the data. You'll need to set up both of the input variables in the model data, and have a couple of beta parameters to estimate.
2. Create two plots holding one of the variables fixed whilst varying the other.

A logistic regression example with Real data

We'll use adapted Moth mortality data from Royla and Dorazio (Chapter 2).

```
T = 12
K = 20
y = c(1,4,9,13,18,20, 0,2,6,10,12,16)
sex = c(rep('male',6), rep('female',6))
dose = rep(0:5, 2)
sexcode = as.integer(sex == 'male')
```

Exercise 9

1. What are the effects of dose and sex?
