Practical 3 - Gaussian processes for time series

Doug McNeall & Andrew Parnell

12 May 2016

Introduction

Welcome to Practical 3, on using JAGS to fit Gaussian process (GP) models for time series analysis. In this practical we'll:

- Simulate some data from a Gaussian process, and fit an approriate model using JAGS
- Fit a GP model to some interesting simulated data, and see how our assumptions about that data change our predictions.

You should follow and run the commands shown in the grey boxes below. 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 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`.

Gaussian processes for time series analysis

First, clear up the environment and load some useful libraries.

```
rm(list=ls())
library(R2jags)
library(MASS) # Useful for mvrnorm function
```

Simulate from a Gaussian process (GP)

We'll simualte some data from a GP. Here is the maths:

y is the vector of observations of y_t, a response variable at time t. We can model y as drawn from a **multivariate Normal** distribution:

$$y \sim MVN(\mu, \Sigma)$$

$$\mu_t = \alpha$$

Σ is a covariance matrix where

$$\Sigma_{ij} = \tau^2 e^{-\rho(t_i - t_j)^2}$$

if $i \neq j$ and

$$\Sigma_{ij} = \tau^2 + \sigma^2$$

if $i = j$ (i.e. on the diagonal).

We can translate that notation into R code:

```
# y(t): Response variable at time t, defined on continuous time
# y: vector of all observations
# alpha: Overall mean parameter
# sigma: residual standard deviation parameter (sometimes known in the GP world as the nugget)
# rho: decay parameter for the GP autocorrelation function
# tau: GP standard deviation parameter
```

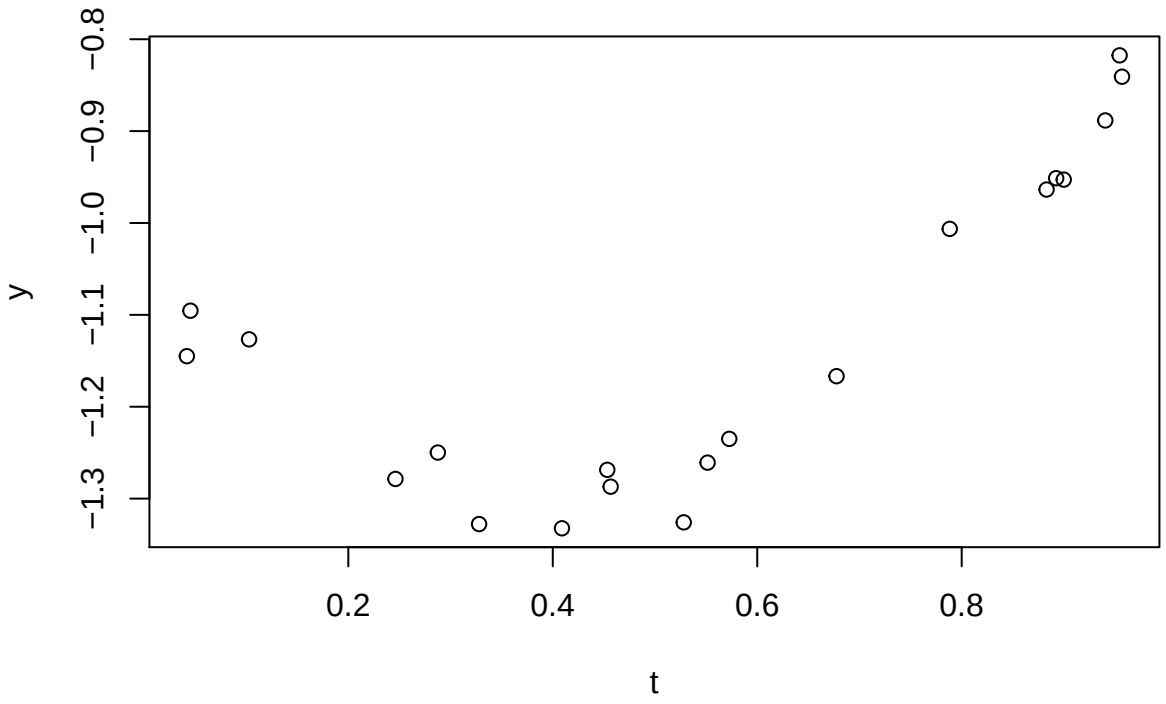
and

```
# Likelihood:
# y ~ MVN(Mu, Sigma)
# where MVN is the multivariate normal distribution and
# Mu[t] = alpha
# Sigma is a covariance matrix with:
# Sigma_{ij} = tau^2 * exp( -rho * (t_i - t_j)^2 ) if i != j
# Sigma_{ij} = tau^2 + sigma^2 if i=j
# The part exp( -rho * (t_i - t_j)^2 ) is known as the autocorrelation function

# Prior
# alpha ~ N(0,100)
# sigma ~ U(0,10)
# tau ~ U(0,10)
# rho ~ U(0.1, 5) # Need something reasonably informative here
```

Here is some R code to simulate data from a Gaussian process. Run and plot the simulated data.

```
T = 20 # default is 20 can take to e.g T = 100 but fitting gets really slow ...
alpha = 0 # default is 0
sigma = 0.03 # default is 0.01
tau = 1 # default is 1
rho = 1# default is 1
set.seed(123) # ensure reproducablility
t = sort(runif(T))
Sigma = sigma^2 * diag(T) + tau^2 * exp( - rho * outer(t,t,'-')^2 )
y = mvrnorm(1,rep(alpha,T), Sigma)
plot(t,y)
```



Exercise 1

1. Change the paramaters `rho, sigma,` and `tau` and see how the output changes. You might want to set the seed (uncomment the line) if you'd like to have the same output each time.

Fit a GP model to the data using JAGS

Load the GP model code for JAGS into R

```
# Jags code to fit the model to the simulated data
model_code = '
model
{
  # Likelihood
  y ~ dmnorm(Mu, Sigma.inv)
  Sigma.inv <- inverse(Sigma)

  # Set up mean and covariance matrix
  for(i in 1:T) {
    Mu[i] <- alpha
    Sigma[i,i] <- pow(sigma, 2) + pow(tau, 2)

    for(j in (i+1):T) {
      Sigma[i,j] <- pow(tau, 2) * exp( - rho * pow(t[i] - t[j], 2) )
      Sigma[j,i] <- Sigma[i,j]
```

```
    }
  }

  alpha ~ dnorm(0, 0.01) # default dnorm(0, 0.01)
  sigma ~ dunif(0, 10) # default dunif(0,10)
  tau ~ dunif(0, 10) # default dunif(0, 10)
  rho ~ dunif(0.1, 5) # default dunif(0.1, 5)

}
'
```

Set up the data as a list object, and choose the parameters that we'd like to watch.

```
# Set up the data
model_data = list(T = T, y = y, t = t)

# Choose the parameters to watch
model_parameters =  c("alpha", "sigma", "tau", "rho")

# Run the model - This can be slow with lots of data
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
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 4
##    Total graph size: 1411
##
## Initializing model
```

Exercise 2

1. Print the `model_run` and check for convergence.
2. Plot histograms of samples from `alpha, sigma, tau` and `rho` to see how well they are estimated.

Make predictions using the GP model

Now we'll create some predictions of new values at new times Now create some predictions of new values at new times t^new These are based on the formula:

```
# y^new | y ~ N( Mu_new + Sigma_new^T solve(Sigma, y - Mu), Sigma_* - Sigma_new^T solve(Sigma, Sigma_ne
# where:
# Mu^new[t] = alpha
# Sigma_new[i,j] = tau^2 * exp( -rho * (t^new_i - t_j)^2 )
# Sigma_*[i,j] = tau^2 * exp( -rho * (t^new_i - t^new_j)^2 ) if i != j
```

```
T_new = 20
t_new = seq(0,1,length=T_new)
Mu = rep(mean(alpha), T)
Mu_new = rep(mean(alpha), T_new)
Sigma_new = mean(tau)^2 * exp( -mean(rho) * outer(t, t_new, '-')^2 )
Sigma_star = mean(sigma)^2 * diag(T_new) + mean(tau)^2 * exp( - mean(rho) * outer(t_new,t_new,'-')^2 )
Sigma = mean(sigma)^2 * diag(T) + mean(tau)^2 * exp( - mean(rho) * outer(t,t,'-')^2 )

# Use fancy equation to get predictions
pred_mean = Mu_new + t(Sigma_new)%*%solve(Sigma, y - Mu)
pred_var = Sigma_star - t(Sigma_new)%*%solve(Sigma, Sigma_new)
```

Exercise 3

1. Plot the mean prediction, and 95% CI
2. Extend `t` out beyond the limits of where there are observation (i.e. t>1). What happens to the prediction mean and the uncertainty?

Summarise the uncertainty in the mean function

Exercise 4

1. Show the uncertainty in the mean function. Sample a number of possible values of each parameter from their joint posterior distribution. Plot them as lines on the graph.

Experiment with the GP

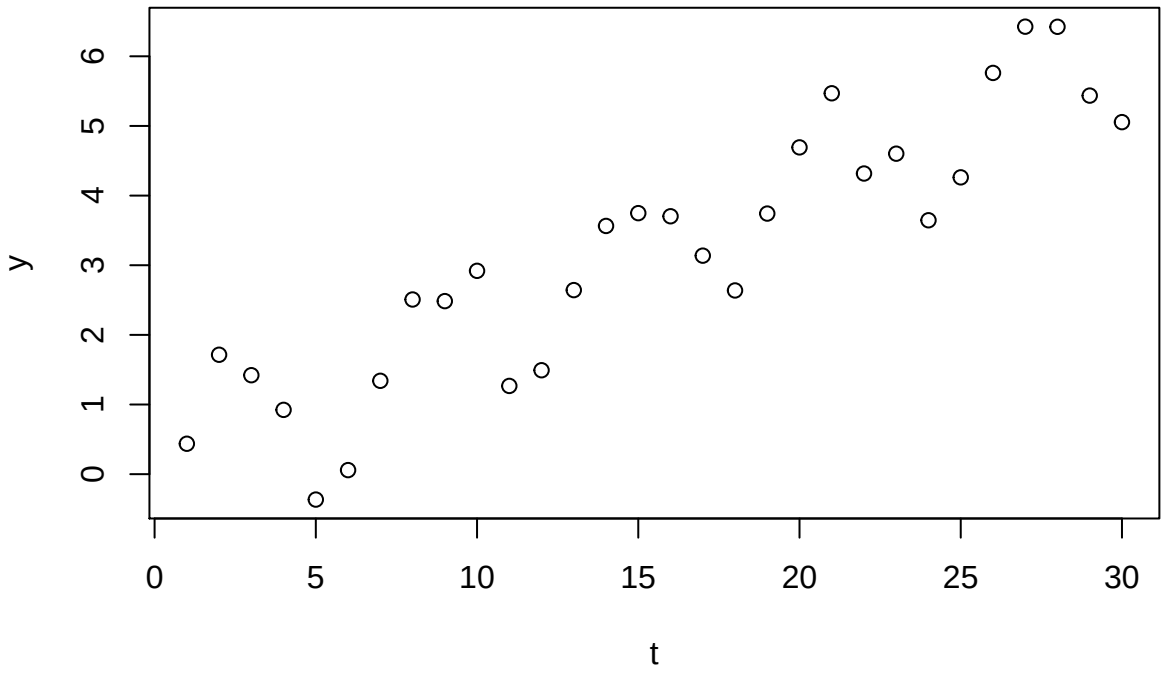
Exercise 6

1. Set the number of points `T` to be very low - say 4 or 5. Increase sigma to get a less regular and smooth timeseries. Put a very low prior on `sigma` (the nugget), and put a prior indicating a large value of `tau`, which controls how fast the GP 'forgets'. What happens to the uncertainty bounds and the mean function?

ADVANCED exercises

You can create a periodic time series with a trend, like this:

```
T = 30
t = 1:T
y <- rnorm(T,t/5 + sin(seq(from = 0, to = T, length.out=T)), 0.3)
plot(t,y)
```



Use our standard JAGS model to fit a GP

```
# Jags code to fit the model to the simulated data
model_code = '
model
{
  # Likelihood
  y ~ dmnorm(Mu, Sigma.inv)
  Sigma.inv <- inverse(Sigma)

  # Set up mean and covariance matrix
  for(i in 1:T) {
  Mu[i] <- alpha
  Sigma[i,i] <- pow(sigma, 2) + pow(tau, 2)

  for(j in (i+1):T) {
  Sigma[i,j] <- pow(tau, 2) * exp( - rho * pow(t[i] - t[j], 2) )
  Sigma[j,i] <- Sigma[i,j]
  }
  }

  alpha ~ dnorm(0, 0.01)
  sigma ~ dunif(0, 10) # default dunif(0,10)
```

```
  tau ~ dunif(0, 1)
  rho ~ dunif(0.1, 5)

}
  '

# Set up the data
model_data = list(T = T, y = y, t = t)

# Choose the parameters to watch
model_parameters =  c("alpha", "sigma", "tau", "rho")

# Run the model - can be slow
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
```

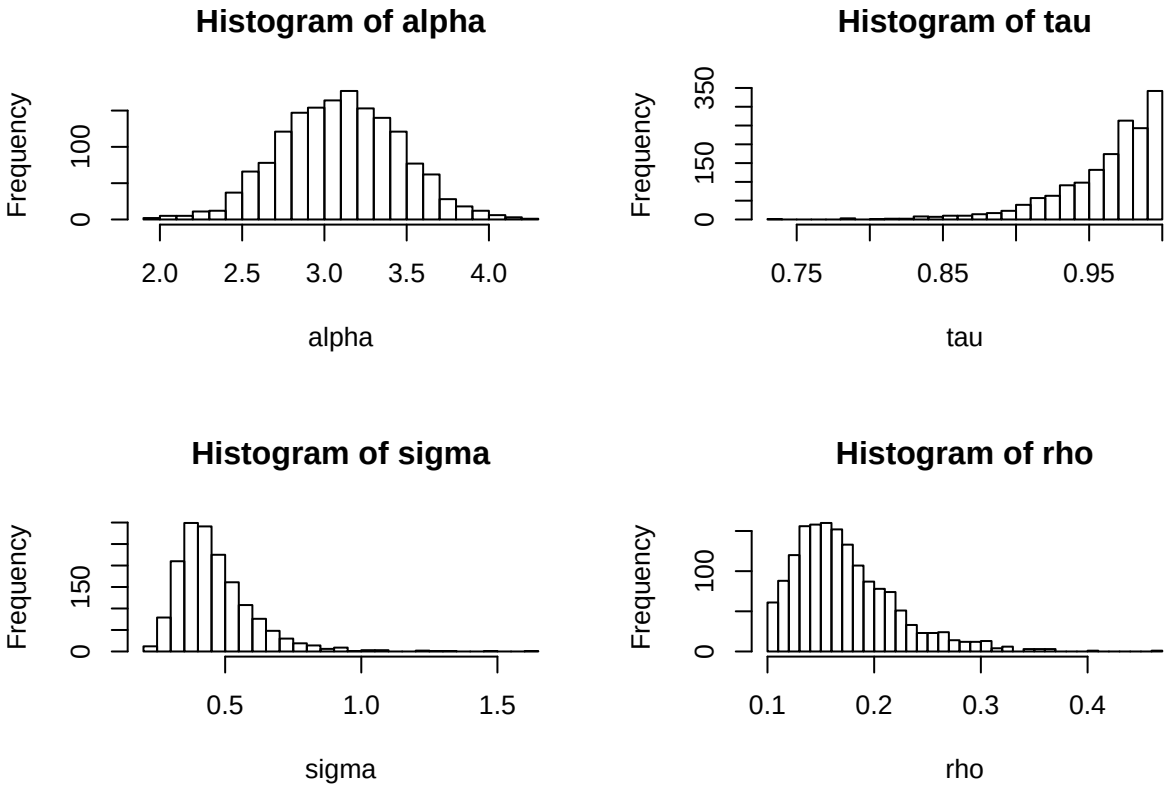
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 4
##    Total graph size: 1532
##
## Initializing model
```

```
# Simulated results -----------------------------------------------------

# Results and output of the simulated example, to include convergence checking, output plots, interpret
print(model_run)
```

```
## Inference for Bugs model at "6", fit using jags,
##  4 chains, each with 1000 iterations (first 200 discarded), n.thin = 2
##  n.sims = 1600 iterations saved
##          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## alpha      3.096   0.360  2.418  2.845  3.098  3.339  3.800 1.002   950
## rho        0.173   0.048  0.107  0.139  0.164  0.198  0.292 1.006   410
## sigma      0.462   0.142  0.277  0.367  0.430  0.522  0.806 1.004   620
## tau        0.962   0.035  0.868  0.945  0.972  0.988  0.999 1.000  1600
## deviance  89.406   3.326 85.174 87.023 88.641 91.016 97.614 1.002  1600
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 5.5 and DIC = 94.9
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```r
alpha = model_run$BUGSoutput$sims.list$alpha
tau = model_run$BUGSoutput$sims.list$tau
sigma = model_run$BUGSoutput$sims.list$sigma
rho = model_run$BUGSoutput$sims.list$rho
par(mfrow = c(2,2))
hist(alpha, breaks=30)
hist(tau, breaks=30)
hist(sigma, breaks=30)
hist(rho, breaks=30)
```




```r
par(mfrow=c(1,1))
```

Bonus Exercise 1

1. What happens when you extrapolate beyond the observation?

Bonus Exercise 2

1. Include a linear term and corresponding parameter to estimate in the JAGS model. How does this change the fit of the GP, and what happens in extrapolation?

2. Include a periodic term in the covariance function. Does this help in extrapolation?
