Introduction to Time Series Analyis

Dr Rafael de Andrade Moral
Associate Professor of Statistics, Maynooth University

rafael.deandrademoral@mu.ie
https://rafamoral.github.io

GAMs

Generalized Additive Models (GAMs)

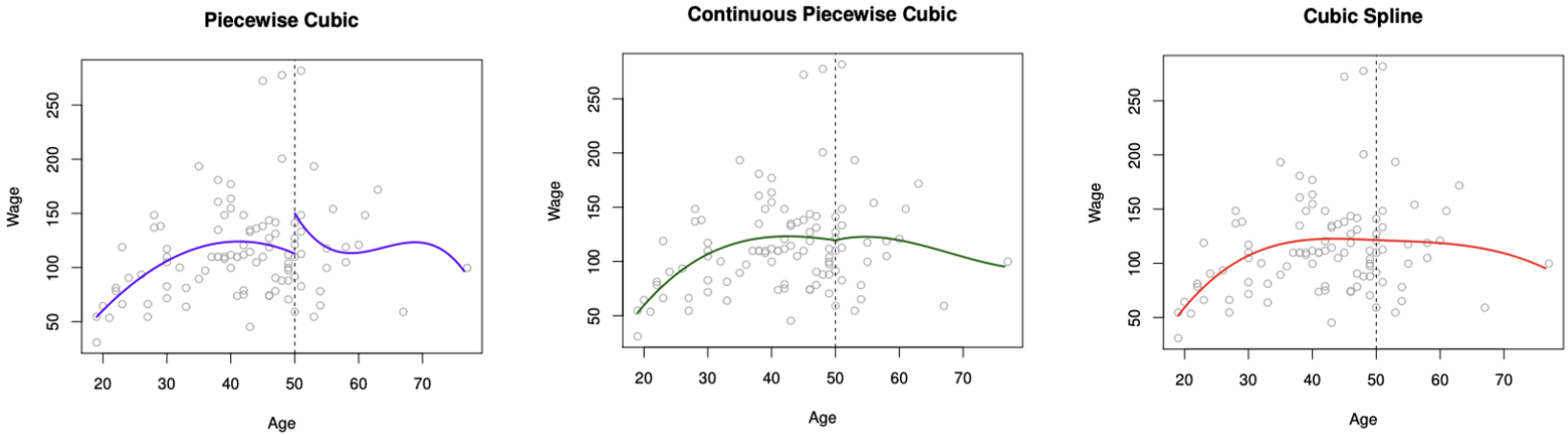
- GAMs are generalized linear models that include additive smooth terms in the linear predictor, e.g.

$$\begin{aligned} Y_i &\sim \mathcal{D}(\mu_i, \phi) \\ g(\mu_i) &= \beta_0 + f_1(x_1) + f_2(x_2) + f_3(x_3, x_4) \end{aligned}$$

- $f_j(\cdot)$ are *smooth* functions, and can take up many different forms
- typically in GAMs we use different types of *splines*

Generalized Additive Models (GAMs)

- E.g. cubic regression splines



- the `mgcv` package in R allows for the specification of many different types of *basis functions*
- Example: salary data

GAMs in time series analysis

- Splines can be used to smooth a time series
- Autocorrelation can be included in many different ways
- Example 1: non-seasonal time series of the number of campylobacterosis cases (reported every 28 days) in the North of Québec in Canada from January 1990 to the end of October 2000
- Example 2: seasonal time series of the monthly number of bulls slaughtered in Queensland from July 1976 to December 2018

Bayesian Modelling

Who was Bayes?

An essay towards solving a problem on the doctrine of chances (1763)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes Theorem in English

Bayes' theorem can be written in words as:

posterior is proportional to likelihood times prior

. . . or . . .

posterior $\propto$ likelihood $\times$ prior

Each of the three terms *posterior*, *likelihood*, and *prior* are *probability distributions* (pdfs).

In a Bayesian model, every item of interest is either data (which we will write as x) or parameters (which we will write as θ). Often the parameters are divided up into those of interest, and other *nuisance parameters*

Bayes theorem in maths

Bayes' equation is usually written mathematically as:

$$p(\theta|x) \propto p(x|\theta) \times p(\theta)$$

or, more fully:

$$p(\theta|x) = \frac{p(x|\theta) \times p(\theta)}{p(x)}$$

- The *posterior* is the probability of the parameters given the data
- The *likelihood* is the probability of observing the data given the parameters (unknowns)
- The *prior* represents external knowledge about the parameters

What's different from what we were doing before?

- We still have a likelihood and parameters to estimate
- We now also have some extra constraints (defined by us) called the *prior distribution*
- There is a clever Bayesian algorithm to create the resulting parameter estimates and their uncertainties
- This full probability distribution of the outputs is the posterior distribution
- The full posterior probability distribution is provided to us as a set of samples

Choosing a prior

- The key to choosing a prior distribution is to choose values which you believe represent the reasonable range that the parameter can take, or come from a related study in the literature
- A prior which is a strong constraint on the parameters is called an *informative prior*
- Some people argue that informative priors are bad, others that they are absolutely necessary in every model
- Sometimes an informative prior can be the difference between being able to fit the model or not
- Most people forget that choosing a likelihood probability distribution is exactly the same task as choosing a prior

Practical differences between frequentist statistics and Bayes

- In frequentist statistics you tend to get a single best estimate of a parameter and a standard error, often assumed normally distributed, and a p-value
- In Bayesian statistics you get a large set of samples of the parameter values which match the data best. You get to choose what you do with these
- In frequentist statistics if the p-value is less than 0.05 you win. If not you cry and try a different model
- In Bayesian statistics you try to quantify the size of an effect from the posterior distribution, or find a particular posterior probability, e.g. P(slope $>$ 0 given the data).

Stan and JAGS

- We will be using two different software tools to calculate posterior distributions. These represent the state of the art for user-friendly, research quality Bayesian statistics.
- Both have their own programming language which you can write in R and then fit the models to get the posterior distribution
- All we have to do in the programming language is specify the likelihood and the priors, and give it the data. The software does the rest

Steps for running JAGS and Stan

1. Write some Stan or JAGS code which contains the likelihood and the prior(s)
2. Get your data into a list so that it matches the data names used in the Stan/JAGS code
3. Run your model through Stan/JAGS
4. Get the posterior output
5. Check convergence of the posterior probability distribution
6. Create the output that you want (forecasts, etc)

Stan vs JAGS

- Stan positives: very flexible, uses sensible distribution names, everything is declared, lots of documentation support, written by people at the top of the field
- Stan negatives: cannot have discrete parameters, some odd declaration choices, slower to run code, code tends to be longer
- JAGS positives: very quick for simple models, no declarations required, a bit older than Stan so more queries answered online
- JAGS negatives: harder to get complex models running, not as fancy an algorithm as Stan, crazy way of specifying normal distributions
- Example[1]: sea level data
- Example[2]: sheep numbers in Asia

What are JAGS and Stan doing in the background?

- JAGS and Stan run a stochastic algorithm called Markov chain Monte Carlo to create the samples from the posterior distribution
- This involves:
  1. Guessing at *initial values* of the parameters. Scoring these against the likelihood and the prior to see how well they match the data
  2. Then iterating:
     1. Guessing *new parameter values* which may or may not be similar to the previous values
     2. Seeing whether the new values match the data and the prior by calculating *new scores*
     3. If the scores for the new parameters are higher, keep them. If they are lower, keep them with some probability depending on how close the scores are, otherwise discard them and keep the old values
- What you end up with is a set of parameter values for however many iterations you chose.

How many iterations?

- Ideally you want a set of posterior parameter samples that are independent across iterations and is of sufficient size that you can get decent estimates of uncertainty
- There are three key parts of the algorithm that affect how good the posterior samples are:
  1. The starting values you chose. If you chose bad starting values, you might need to discard the first few thousand iterations. This is known as the *burn-in* period
  2. The way you choose your new parameter values. If they are too close to the previous values the MCMC might move too slowly so you might need to *thin* the samples out by taking e.g. every 5th or 10th iteration
  3. The total number of iterations you choose. Ideally you would take millions but this will make the run time slower

JAGS and Stan have good default choices for these but for complex models you often need to intervene

How many chains?

- Beyond increasing the number of iterations, thinning, and removing a burn-in period, JAGS and Stan automatically run *multiple chains*
- This means that they start the algorithm from 3 or 4 different sets of starting values and see if each *chain* converges to the same posterior distribution
- If the MCMC algorithm has converged then each chain should have the same mean and variance.
- Both JAGS and Stan report the `Rhat` value, which is close to 1 when all the chains match
- It's about the simplest and quickest way to check convergence. If you get `Rhat` values above 1.1, run your MCMC for more iterations

Summary

- Bayesian methods just add on a set of extra constraints on to the likelihood called prior distributions
- We now know how to run some simple time series models in JAGS and Stan
- We know that the fitting algorithm (MCMC) produces best parameter estimates and their uncertainties
- We have to do a little bit more work to get the predictions out of JAGS or Stan
- The big advantage of using these methods is the extra flexibility we get from being able to write our own models

ARCH and GARCH models

General principles of models for changing variance

- So far we have looked at models where the mean changes but the variance is constant:

$$y_t \sim N(\mu_t, \sigma^2)$$

- In this module we look at methods where instead:

$$y_t \sim N(\alpha, \sigma_t^2)$$

- These are:
  - Autoregressive Conditional Heteroskedasticity (ARCH)
  - Generalised Autoregressive Conditional Heteroskedasticity (GARCH)
  - Stochastic Volatility Models (SVM)
- They follow the same principles as ARIMA, but work on the standard deviations or variances instead of the mean

Extension 1: ARCH

- An ARCH(1) Model has the form:

$$\sigma_t^2 = \gamma_1 + \gamma_2 \epsilon_{t-1}^2$$

  where ϵ_t is the residual, just like an MA model
- Note that $\epsilon_t = y_t - \alpha$ so the above can be re-written as:

$$\sigma_t^2 = \gamma_1 + \gamma_2 (y_{t-1} - \alpha)^2$$

- The variance at time t thus depends on the previous value of the forecast error (more like an MA model than AR)
- The residual needs to be squared to keep the variance positive.
- The parameters γ_1 and γ_2 also need to be positive, and usually $\gamma_2 \sim U(0, 1)$
- Example: forest fires in Canada

From ARCH to GARCH

- The Generalised ARCH model works by simply adding the previous value of the variance, as well as the previous value of the observation
- The GARCH(1,1) model thus has:

$$\sigma_t^2 = \gamma_1 + \gamma_2(y_{t-1} - \alpha)^2 + \gamma_3\sigma_{t-1}^2$$

- There are, as always, complicated restrictions on the parameters, though like the stationarity conditions in ARIMA models we can relax this assumption and see if the data support it
- It's conceptually easy to extend to general GARCH(p,q) models which add in extra previous lags

Stochastic Volatility Modelling

- Both ARCH and GARCH propose a deterministic relationship for the current variance parameter
- By contrast a Stochastic Volatility Model (SVM) models the variance as its own *stochastic process*
- SVMs, ARCH and GARCH are all closely linked if you go into the bowels of the theory
- The general model structure is often written as:

$$y_t \sim N(\alpha, \exp(h_t))$$

$$h_t \sim N(\mu + \phi h_{t-1}, \sigma^2)$$

- You can think of an SVM being like a GLM but with a log link on the variance parameter

Summary

- We know that ARCH extends the ARIMA idea into the variance using the previous values of the series
- We know that GARCH extends ARCH with previous values of the variance too
- We know that SVMs give the variance its own stochastic process
- We can combine these new models with all the techniques we have previously learnt

Continuous time models

Discrete time vs continuous time

- Almost all of the models we have studied so far assume that time is discrete, e.g. $t = 1, 2, 3, \ldots$, or year $= 1850, 1851, \ldots$
- Many real world time series do not work on discrete times scales. Instead the time value t might be any integer or non-integer value
- Continuous time series might occur because:
  1. The data are recorded in irregular time
  2. The data contain many missing values. We could use the `NA` trick but if there are too many this becomes impractical
- Really all time series are recorded in continuous time, we just sometimes approximate them onto a grid. There can be lots of subtle issues when data are aggregated incorrectly

Some models for continuous time we have met

- Linear, log-linear and logistic regression work for data recorded in continuous time
- However it could be argued that they are not true time series models since they borrow predictive strength from both the future and the past

Brownian motion

- Perhaps the simplest of all continuous time series models is that of *Brownian Motion* (BM)
- As we are now working in continuous time we write the time series as $y(t)$ rather than y_t to allow for y to be a function of continuous t
- The likelihood for BM is:

$$y(t) - y(t-s) \sim N(0, s\sigma^2)$$

  where s is any positive value. Note that if s is 1 we have the standard random walk model
- You can also add in a *drift* parameter and re-write the model as:

$$y(t) - y(t-s) \sim N(\alpha s, s\sigma^2)$$

- Example: ice core data

The Ornstein Uhlenbeck process

- One extension of Brownian Motion is called the *Ornstein-Uhlenbeck* (OU) process
- It can also be thought of as the continuous time version of the AR(1) process
- The likelihood is:

$$y(t) - y(t-s) \sim N(\theta(\alpha - y(t-s))s, s\sigma^2)$$

- It looks very much like the BM model but with an extra parameter θ which controls the dependence of $y(t)$ on $y(t-s)$ according to how far away it is
- With a bit of algebra if you set $s = 1$ above you end up with the AR(1) model
- Like the AR(1) model, θ needs to be between -1 and 1 to be stationary, but in practice can go beyond that range

Multivariate time series

Introduction to multivariate models

- Often we have multiple different observations at each time and we want to model them together
- For example, we might have multiple different climate variables observed, or multiple chemical signatures. If they are correlated then by fitting them separately we will lose precision
- If the observations are observed at different times for each time series we can use the NA trick, or create a latent state space model on which all measurements are regular

The Vector AR model

- We can extend most of the models we have met to multivariate scenarios by applying the multivariate normal distribution instead of the univariate version
- Suppose now that y_t is a vector of length k containing all the observations at time t
- We can write:

$$y_t = A + \Phi y_{t-1} + \epsilon_t,\ \epsilon_t \sim MVN(0, \Sigma)$$

or equivalently

$$y_t \sim MVN(A + \Phi y_{t-1}, \Sigma)$$

- where MVN is the multivariate normal distribution
- Here the parameter vector A controls the overall mean level for each of the k series, Φ is a $k \times k$ matrix which controls the influence on the current value of the previous time points of *all* series
- Σ here is a $k \times k$ matrix that controls the variance of the process and the residual correlation between the multiple series

Time Series: the state of the art

Mixing up state space models, multivariate time series, Gaussian processes

- We can extend the simple state space model we met earlier to work for multivariate series
- We would have a state equation that relates our observations to a multivariate latent time series (possibly of a different dimension)
- We could change the time series model of the latent state to be an ARIMA model, an O-U process, a Gaussian process, or anything else you can think of!

Dynamic linear models

- So far in all our models we have forced the time series parameters to be constant over time
- In a *Dynamic Linear Model* we have a state space model with :

$$y_t = F_t x_t + \epsilon_t,\ \epsilon_t \sim MVN(0, \Sigma_t)$$

$$x_t = G_t x_{t-1} + \gamma_t,\ \gamma_t \sim N(0, \Psi_t)$$

- The key difference here is that the transformation matrices F_t and G_t can change over time, as can the variance matrices Σ_t and Ψ_t, possibly in an ARCH/GARCH type framework
- These are very hard models to fit in JAGS/Stan but simple versions can work

Latent factor time series models

- If we have very many series, a common approach to reduce the dimension is to use Factor Analysis or Principal components
- In a latent factor model we write:

$$y_t = Bf_t + \epsilon_t$$

  where now B is a $numseries \times numfactors$ factor loading matrix which transforms the high dimensional y_t into a lower dimensional f_t.
- f_t can then be run using a set of univariate time series, e.g. random walks
- The B matrix is often hard to estimate and might require some tight priors