Computational Bayes

Get this lecture at:

https://dl.dropboxusercontent.com/u/10495241/AstroR-Bayes-Lec2.pdf

Bayesian Astronomy with R

Aaron Robotham, ICRAR, UWA

Lecture 2

- Get R at http://cran.rstudio.com/

- Download Laplace's Demon from (not on CRAN):
http://www.bayesian-inference.com/LaplacesDemon_13.07.16.tar.gz

- Type 'R' on the terminal to begin.

```
> install.packages('magicaxis')
> Install.packages('ellipse')
> install.packages('mvtnorm')
> install.packages('PATH/?/?/LaplacesDemon_13.07.16.tar.gz',
type='source')
> library('magicaxis')
> library('ellipse')
> library('mvtnorm')
> library('LaplacesDemon')
```

- Done that? Good, now you've got everything you need to recreate the data and plots in this lecture.

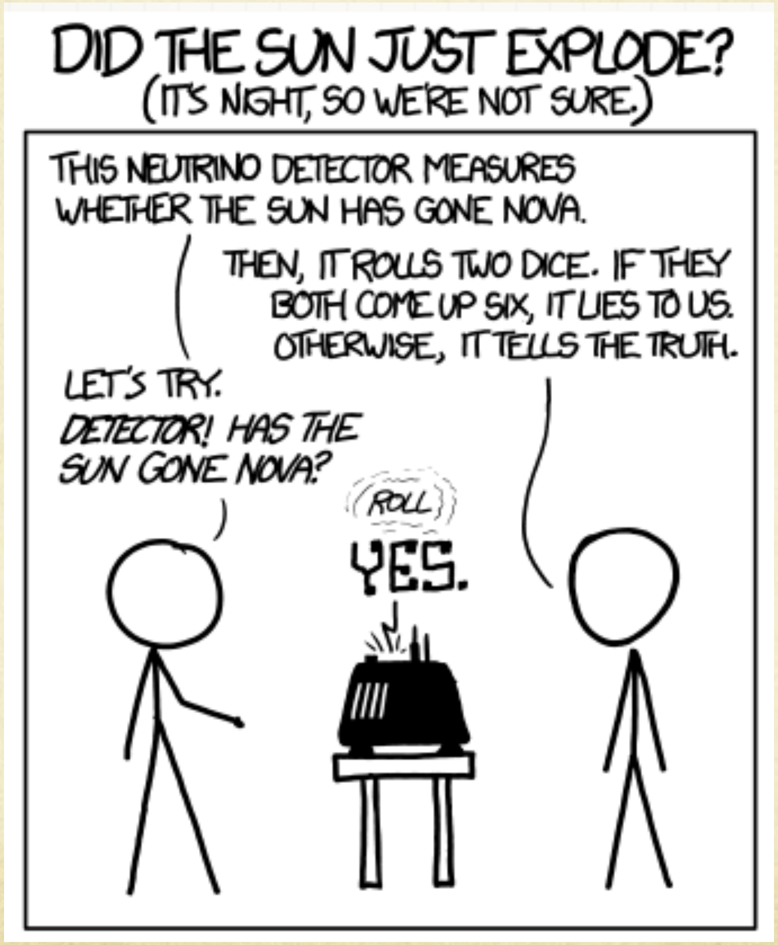
Bayesian Astronomy with R

Aaron Robotham, ICRAR, UWA

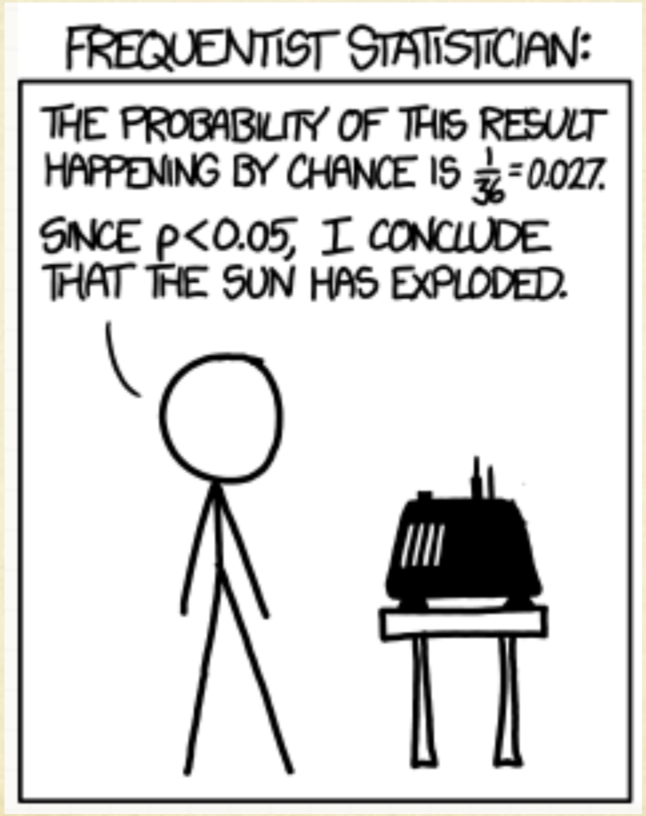
Summary of Lecture 1

- Bayes states some fairly obvious (intuitive) ideas about probability theory. Simply: $P(A|B)=P(B|A)P(A)/P(B)$
- Most important:
  - $P(\Theta,y) \propto P(y,\Theta) \times P(\Theta)$ (for probabilities)
  - $p(\Theta,y) \propto p(y,\Theta) \otimes p(\Theta)$ (for probability densities)
- It is applicable in almost every situation you can think of, but as we show, it is not always trivial to calculate Bayes Factor by hand.
- In the next lecture we will see how computers are used to attack Bayesian problems.

A Semi-fair joke (from XKCD)


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY. DETECTOR! HAS THE SUN GONE NOVA?
ROLL
YES.

A Semi-fair joke (from XKCD)


FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS $\frac{1}{36}$=0.027. SINCE $p<0.05$, I CONCLUDE THAT THE SUN HAS EXPLODED.

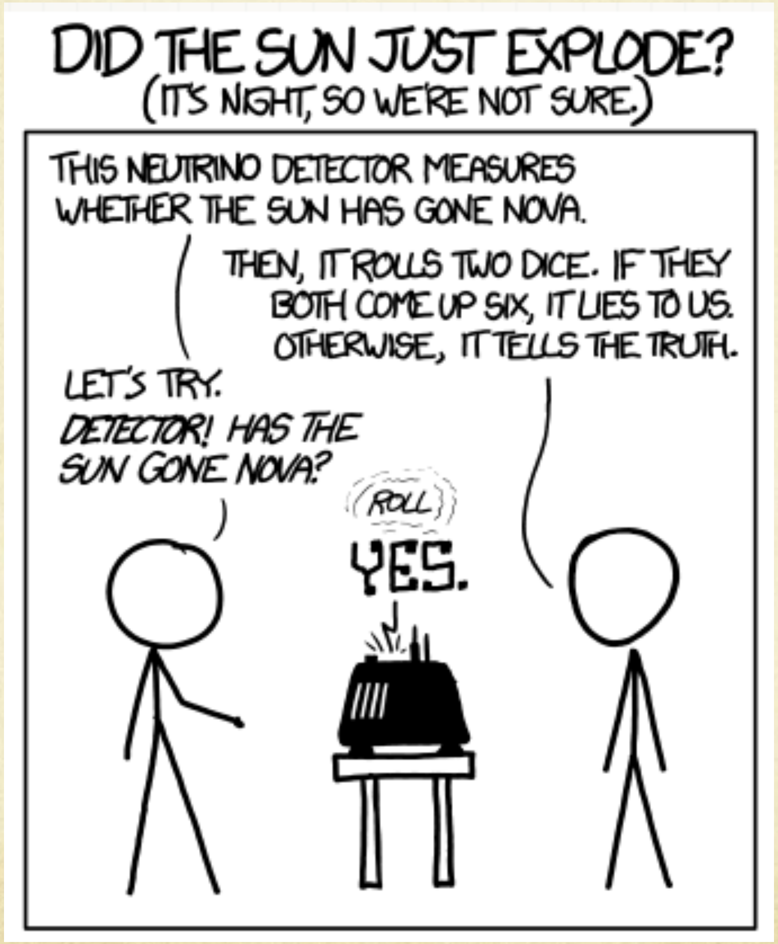
A Semi-fair joke (from XKCD)

BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.

A Semi-fair joke (explained)

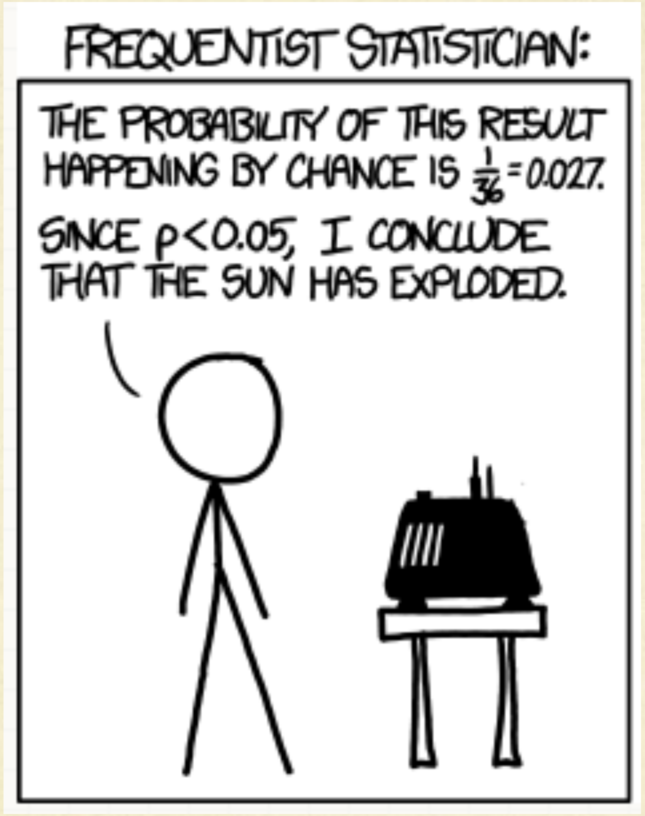
So far so good.

Whilst this doesn't sound like an ideal experimental setup, there's nothing 'wrong' with it from a statistical point-of-view.


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY.
DETECTOR! HAS THE SUN GONE NOVA?
ROLL
YES.

A Semi-fair joke (explained)

To apply frequentists statistical tests the experiment needs to be one realisation of infinite possible tests (which it is).


FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS 1/36 = 0.027. SINCE p<0.05, I CONCLUDE THAT THE SUN HAS EXPLODED.

The 'wrong' bit is that he shouldn't conclude the Sun has exploded. Rather, he rejects the hypothesis that the data is compatable with the model of the Sun not having gone nova. He could, of course, have chosen a different threshold to reject the hypothesis.

The frequentist's null hypothesis is that the Sun hasn't gone nova. If the Sun hasn't gone nova there is only a 0.027 chance of observing this experimental outcome (i.e. the data). This null hypothesis is rejected with a 0.05 significance criterion (which is a fairly popular threshold, for some reason).

A Semi-fair joke (explained)

BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.

The Bayesian naturally applies a very strong prior that the Sun hasn't gone nova (even though it is possible, eventually).

What would they need to reassess their prior and form a posterior where the Sun has gone nova? They would probably need to observe spontaneous and coordinated intense light emission from a region of the sky where the Sun should be- i.e., they would need to see it explode with their own eyes.

Lecture 1 Problems

A new killer and rubbish clinic
1001 German tanks

A new killer and rubbish clinic

- Alas, in Sydney they only have access to Sydney Bayes Health. The nominal test is the same as at Perth Bayes Health, so the test is 99.9% accurate at correctly determining you are positive, and 99% accurate at correctly determining you are negative. However, their practices are much sloppier than Perth Bayes Health- 5% of samples are mixed up, and 2% of the time the computer will return a random (50/50 positive/negative) result. Alas, you once again test positive. Whilst annoyed at the sloppiness of the testing procedure (no doubt you'll move to Perth some time soon), you still want to know how woried you should be. What's the answer?

- Hint: This is a much more complicated question, and is an example of a case where calculating P(TestPos) is difficult and computationally expensive. Think of a way to simplify the problem.

- If you crack this then you're thinking like a Bayesian.

1001 German tanks

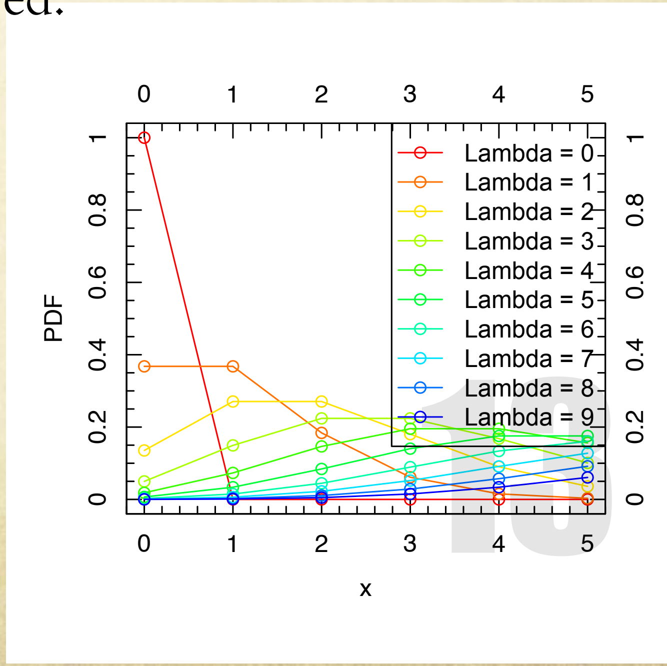
- This problem is based on real scenario that occurred during WWII.
- German tanks had sequentially numbered gear boxes, so in principal if 5 tanks were randomly captured with the numbers 20, 40, 60, 80 and 100 you might reasonably be able to estimate how many tanks the German army has in total.
- With this situation in mind, imagine if one (and only one) tank was captured, and it had a gear box ID #1001. What would a statistician working for the British army estimate as the mean median and mode of the total number of tanks?
- Assume no priors on the tank distribution.
- This is an example of a situation where an instinctive frequentist response gives you a *very* different answer to a Bayesian answer.

Attacking Bayes with computers

- Say we have a non detection in a histogram bin (this could be interpreted as zero counts in a luminosity function bin). What is the most likely value of λ for the underlying Possion distribution that generated this?

- We can intuit, even without knowledge of Bayes, that the true value of λ might well not be 0, since it could be 0, 1 or even 10 with some chance of zero events being observed.

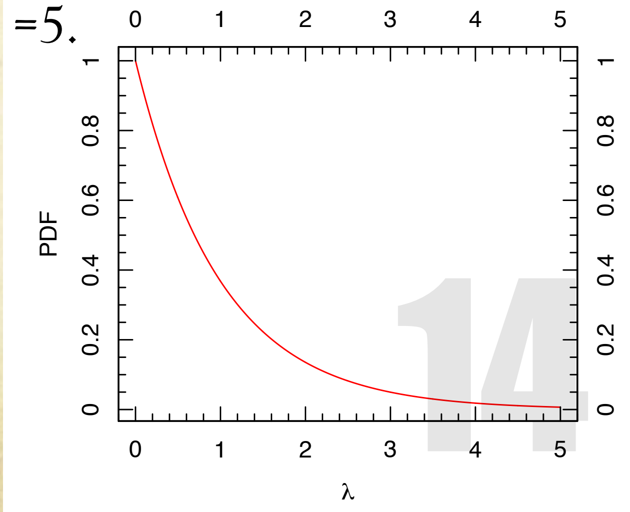
```
> plot.new();plot.window(c(0,5),c(0,1))
> magaxis(xlab='x',ylab='PDF')
> for(i in 1:10){
lines(0:5,dpois(0:5,i-1),col=rainbow(10,end=2/3)[i])
points(0:5,dpois(0:5,i-1),col=rainbow(10,end=2/3)[i])}
> legend('topright',legend=paste('Lambda =',
0:9),col=rainbow(10,end=2/3),lty=1,pch=1)
```



Let's go Bayes on this thing

- The "prior" p(Θ) is the probability of the model based on prior knowledge of how the input parameters might be distributed (in this case just λ).
- In this case we say we have no prior knowledge, so the prior for λ is a uniform distribution from 0 to infinity (more on this later).
- We now calculate the p(y=0,Θ) for *all* possible λ. In practice it will get very small very quickly, so we stop at λ=5.

```
> xval=seq(0,5,len=1000)
> magplot(xval,dpois(0,xval), type='l', col='red',
xlab=expression(lambda), ylab='PDF')
> mean(xval*dpois(0,xval))
[1] 0.191739
```



Let's go Bayes on this thing

- The mode is easy:

```
> xval[which.max(dpois(0,xval))]
[1] 0
```

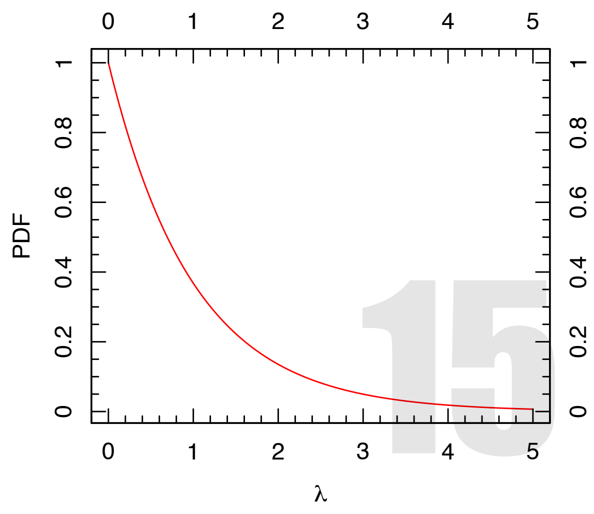
- The mean is easy:

```
> temp=dpois(0,xval)
> sum(temp)/1e3
[1] 0.1989575
```

- The median is a bit trickier:

```
> temp=dpois(0,xval)
> tempsum=cumsum(temp)
> xval[max(which(tempsum<=sum(temp)/2))]
[1] 0.6806807
```

- Mode=0, Mean=0.20, Median=0.68



How to interpret the posterior

- There are a few ways to think about the meaning of such a posterior PDF.
- The mode really is the "most likely" value, but that's not always the most meaningful…
- Why's that? What would you do if I said you had to choose 1 or 2:6 on a die, where I would give you $1,000 if I roll a 1, and $1 otherwise? I hope you would pick 1, even though it is "less likely".
- If we observed 0 aid packages were required by a country in 1 year, the expectation of 0.19 per year would be a more useful guide to planning ahead for 100 years worth of aid, i.e. we should buy 19 packages for that period.
- For this reason the mean, or expectation, is often seen as the most useful measure. But it depends on the question being asked.

Building Probability Chains

- Earlier we considered the Bayesian approach to determing disease tests, we'll now look at a similar but more complicated problem.

- You're a band on Spotify and you'd like more of your target demographic to listen to your music. There are two routes, those annoying voice ads that interupt the music, or a more passive campaign that uses banner ads within the Spotify player.

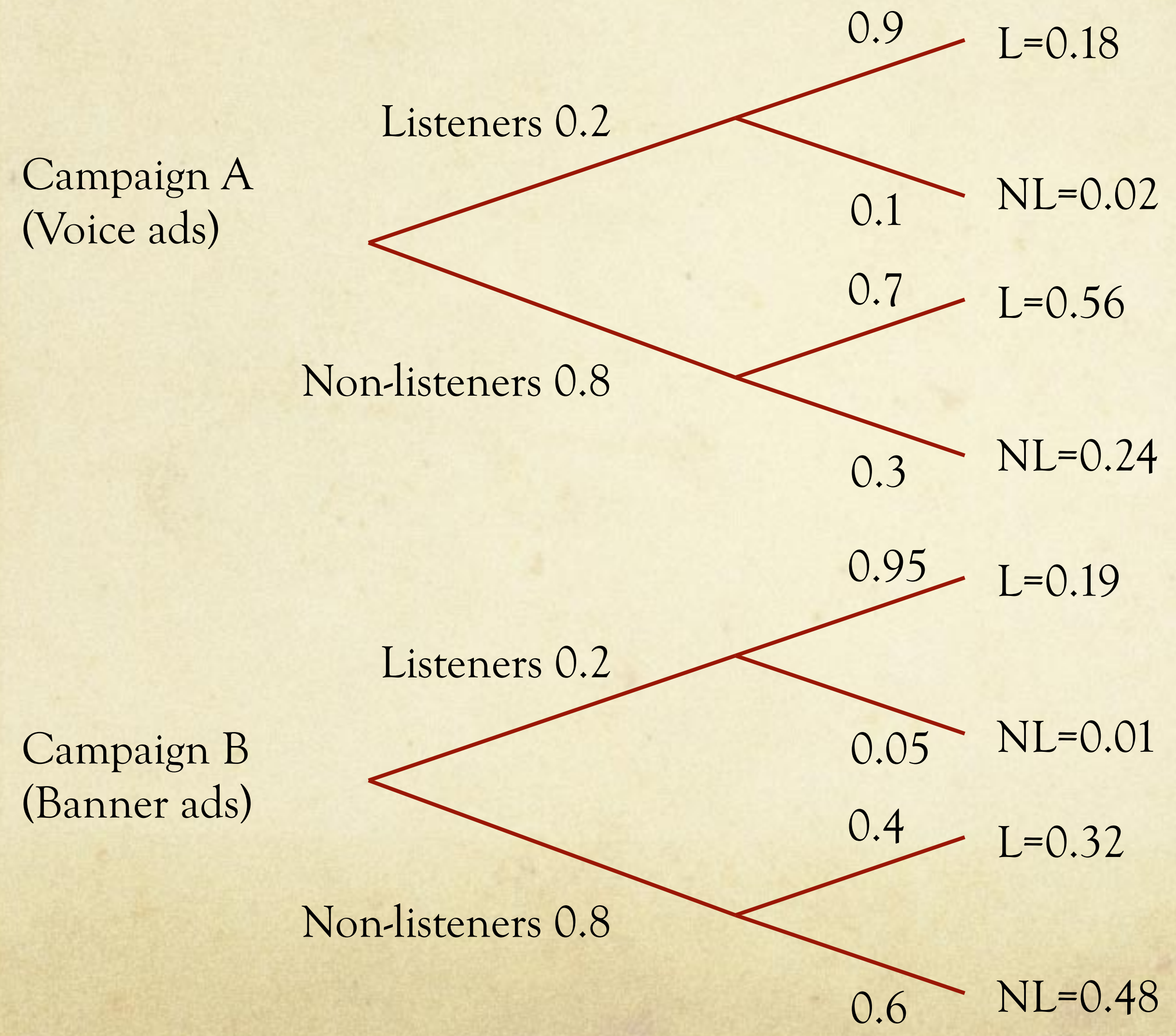
- Initially, 20% of your target demographic listens.

- Campaign A (voice ads) will convert 70% of non-listeners to listeners, but it will also turn off 10% of current listeners per week (because it's annoying!)

- Campaign B (banner ads) will convert 40% of people of non-listeners to listeners, but will only turn off 5% of current listeners per week.

- Which is the best campaign for one week? How about forever?

Building Probability Chains



L after 1 week=0.74
NL after 1 week=0.26

L after 1 week=0.51
NL after 1 week=0.49

Building Probability Chains

- We can simplify this to a couple of matrix operations:
  - Campaign A $=\begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \otimes \begin{bmatrix} 0.9 & 0.1 \\ 0.7 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.74 & 0.26 \end{bmatrix}$
  - Campaign B $=\begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \otimes \begin{bmatrix} 0.95 & 0.05 \\ 0.4 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.51 & 0.49 \end{bmatrix}$

```
> matrix(c(0.2,0.8),ncol=2) %*% matrix(c(0.9,0.1,0.7,0.3),byrow=T,ncol=2)
     [,1] [,2]
[1,] 0.74 0.26
> matrix(c(0.2,0.8),ncol=2) %*% matrix(c(0.95,0.05,0.4,0.6),byrow=T,ncol=2)
     [,1] [,2]
[1,] 0.51 0.49
```

- To generalise to infinity we can therefore do:

```
> domatmult=function(vec=matrix(c(0.2,0.8), ncol=2), mat, loop=100){
  temp=vec
  for(i in 1:loop){temp=temp %*% mat}
  return(temp)}
> domatmult(mat=matrix(c(0.9,0.1,0.7,0.3),byrow=T,ncol=2))
      [,1]  [,2]
[1,] 0.875 0.125
> domatmult(mat=matrix(c(0.95,0.05,0.4,0.6),byrow=T,ncol=2))
          [,1]      [,2]
[1,] 0.8888889 0.1111111 #More effective to not have annoying adverts over time!
```

Markov Chain

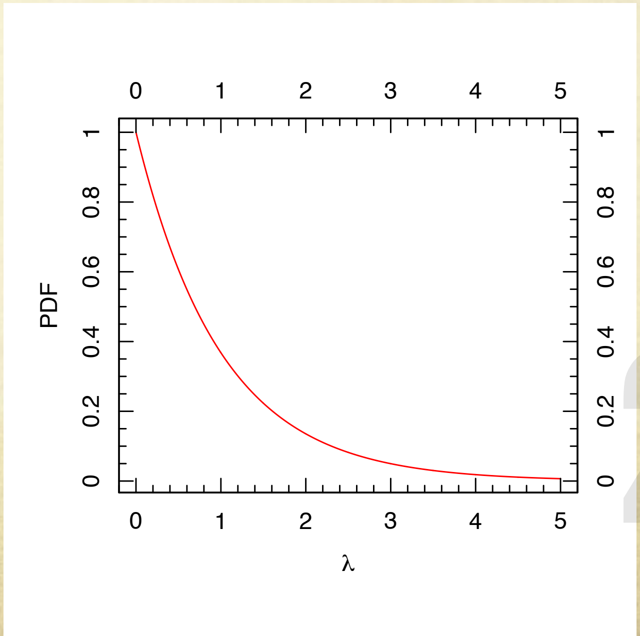
- In the simplest sense we have just produced a 'Markov Chain'.
- This is a mathematical system that undergoes transitions from one state (listeners=20%, non-listeners=80%) to another (listeners=74%, non-listeners=26%).
- Markov Chains are a means to calculate the posterior of complex systems numerically, where analytic results might be hard or impossible.
- The key requirement to create a Markov chain is that each step can only depend on the preceding step, not the sequence of events that got us there (as we will see, there are ways around this requirement). This kind of 'memorylessness' is known as the 'Markov Property'.
- Famous examples are Brownian motion and the 'Drunkard's Walk'. Google's page-rank scheme uses a Markov Chain analysis.

Remember this problem?

- Say we have a non detection in a histogram bin (this could be interpreted as zero counts in a luminosity function bin). What is the most likely value of λ for the underlying Possion distribution that generated this?

- We can intuit, even without knowledge of Bayes, that the true value of λ might well not be 0, since it could be 0, 1 or even 10 with some chance of zero events being observed.

- Mode=0, Mean=0.20, Median=0.68



The Data and the Model

- To generically solve a Bayesian problem we need three things: Data, a likelihood Model, and a scheme that can produce the posterior distribution (we don't necessarily care how).

```
> Data=list(data=0,mon.names='',parm.names='lambda',N=1)
> Model=function(parm,Data){
parm=interval(parm,0,100) #This ensures it can't go negative or above 100
val.prior=log(dunif(parm,0,100))
LL=log(dpois(Data$data,parm))
LP=LL+val.prior
Modelout=list(LP=LP,Dev=2*LL,Monitor=1,yhat=1,parm=parm)}
```

- Don't worry about the slightly odd form of the Data object, this is to make our lives simpler later, we could have said "Data=0" and changed "Data$data" to just "Data".

- Incidentally, 2*LL = $-\chi^2$ when the best fit variables are covariant (do not need to be independent) Gaussians. This is because covariant Gaussians can always be rotated and distorted back to "independent" looking distributions.

Brute force Bayes

- It is usually not easy to calculate the posterior directly, so we need a scheme that can integrate the posterior likelihood space in a manner that reflects the parameter distributions.

- If we choose a value x for λ, and compare that posterior likelihood to another value for λ (x+dx), the ratio of likelihoods (the BF) tells us the relative densities of the posterior PDF at those two locations.

- To walk through the posterior space we can do the following:

  - If the likelihood at λ (x+dx) is higher than λ (x) we update the posterior.

  - If the likelihood at λ (x+dx) is lower than λ (x) we accept it if a random number between 0 and the likelihood at λ (x) is less than the likelihood at λ (x+dx), so if it is 2/3 the likelihood we will update 2/3 of the time.

- If we keep a chain of all posterior values, the PDF of this chain will reflect the posterior likelihood distribution.

Metropolis Hastings MCMC

- We have just described the Metropolis Hastings (MH) Markov Chain Monte Carlo (MCMC) algorithm. An MCMC is a type of random stepping algorithm that obeys certain Markoc Chain conditions, in particular it has to be “memoryless”, i.e. the next step can only be affected by its current state. Metropolis Hastings is a particularly simple type of MCMC.

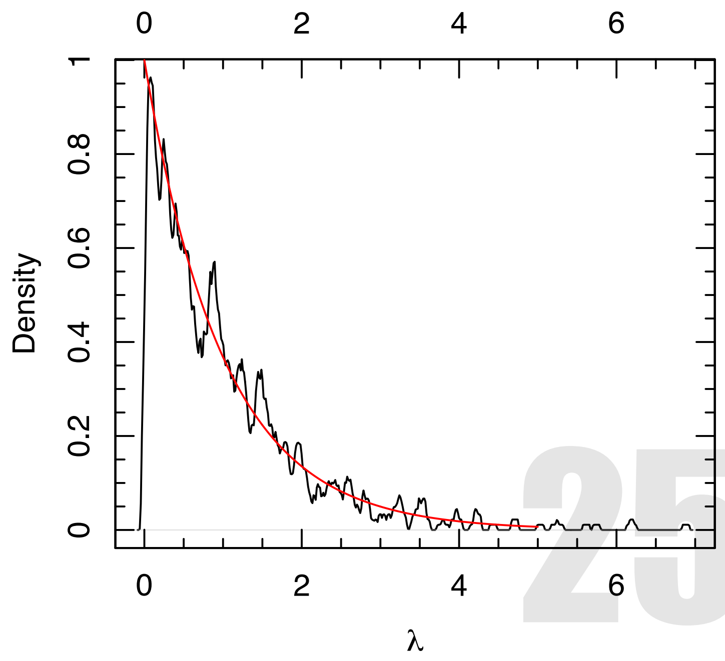
```
> MH=function(Model,Data,start=1,iterations=1e4,thin=100,step=1,burn=0.1){
posterior={} ;mod=Model(start,Data); parm=mod$parm; currentLP=mod$LP;keep={}
for(i in 1:iterations){
        trial=parm+rnorm(1,sd=step)
        mod=Model(trial,Data)
#Update trial in case the original value was negative (this is done within the Model function above)
        trial=mod$parm; newLP=mod$LP
#If new likelihood is higher keep it and update all parameters accordingly
        if(newLP>currentLP){
                parm=trial; currentLP=newLP; keep=c(keep,TRUE)
        }else{
#If new likelihood is lower we test to see if we want to randomly update.
                check=runif(1,0,exp(currentLP))<exp(newLP)
                if(check){parm=trial; currentLP=newLP; keep=c(keep,TRUE)}else{keep=c(keep,FALSE)}
        }
#reduce by the thinning amount
if(i %% thin ==0){posterior=c(posterior,parm)}
}
#Remove the burn in population
posterior=posterior[1:length(posterior)>burn*length(posterior)]
return=list(posterior=posterior,acrate=length(which(keep))/iterations)
}
```

Metropolis Hastings MCMC

- Most arguments are obvious. ‘thin’ is the fraction of posterior samples we want to keep. ‘step’ is the SD of the Gaussian we sample from to find dx. ‘burn’ is the fraction of early samples we ignore because they might not be close to the posterior distribution mode. We now run this new function with a start value of x=1.

```
> testMH=MH(Model, Data, start=1, iterations=1e5, thin=100, step=5, burn=0.1)
> magplot(density(testMH$posterior,kern='rect', bw=0.1/sqrt(12)), xlab='Lamba',
ylab='Density')
> lines(xval,dpois(0,xval),col='red')
```

- The MH ran in only a few seconds.
- The two distributions agree very closely.
- The MH median = 0.72.
- This simple example gives us confidence.
- We’ll now use a more sophisticated MCMC...

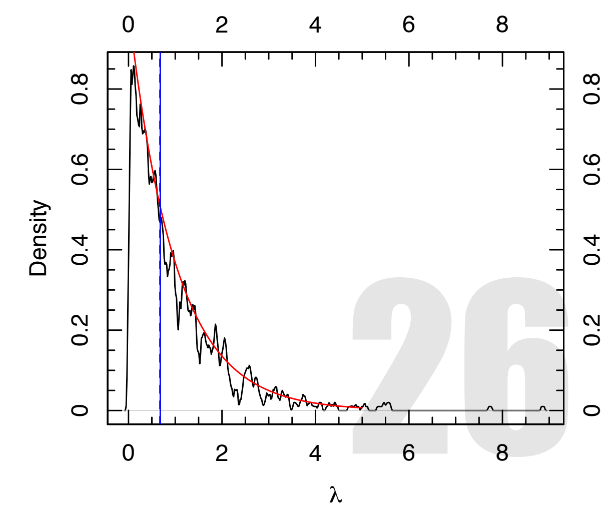


Laplace's Demon

- Laplace's Demon is named for the idea that a being (well, a demon) could exist that, given all the properties of the Universe, could intuit all of the future and all of the past.

- This idea fails for reasons of entropy and quantum physics, but it's name lives on in an R package "LaplacesDemon" that aims to intuit much about the Universe via a particularly complete and sophisticated set of Bayesian tools, including many MCMCs. So same as above:

```
> Fit1D=LaplacesDemon(Model,Data,Initial.Values=1,Algorithm='HARM')
> magplot(density(Fit1D$Posterior2,kern='rect',bw=0.1/
sqrt(12)),xlab=expression(lambda),ylab='Density')
> lines(xval,dpois(0,xval),col='red')
> abline(v=0.68,col='blue')
> abline(v=Fit1D$Summary1[1,'Median'],col='blue',lty=2)
```



- Uses a "Hit And Run Metropolis" here.

- Looks pretty similar to before, so we're feeling confident!

Now we shift to higher dimensions

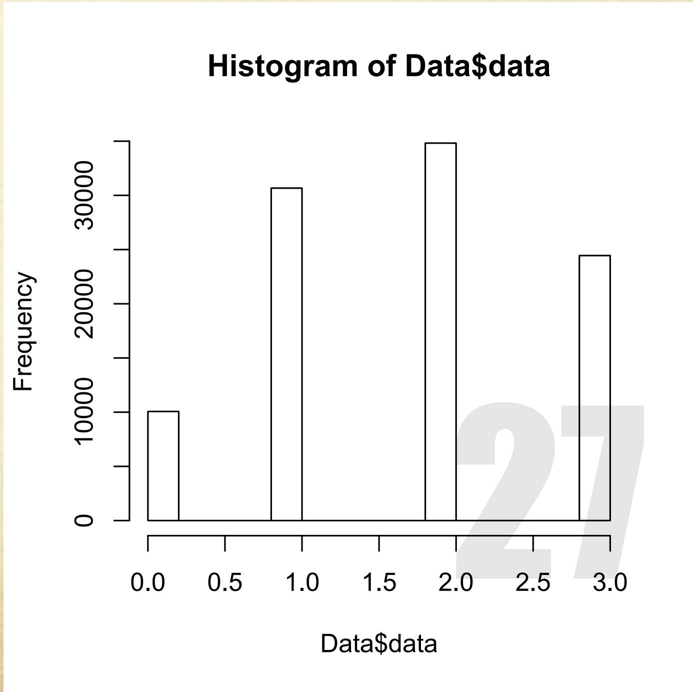
- Let's create a much more interesting problem than the earlier biased coin in bag.
- We simulate 1e5 draws from a bag containing 20% biased coins, where the biased coins have a 90% chance of coming up heads. The coin is tossed 3 times:

```
> draws=rbinom(1e5,1,prob=0.8) #alternative is draws=runif(1e5)>0.8
> Data=list(data=c(rbinom(length(which(draws==1)),3,0.5), rbinom(length(which(draws==0)),
3,0.9)), mon.names='', parm.names=c('biasfrac','coinbias'),N=1e5)
> hist(Data$data)
```

- All we know is that:
  - Some fraction (parm[1]) of the coins are biased.
  - The biased coins are biased by some amount parm[2].

```
> Model=function(parm,Data){
parm[1]=interval(parm[1],0,1);parm[2]=interval(parm[2],0,1)
val.prior=log(dunif(parm[1],0,1))+log(dunif(parm[2],0,1))
LL=sum(log(dbinom(Data$data,3,0.5)*(1-parm[1])+
dbinom(Data$data,3,parm[2])*(parm[1])))
LP=LL+val.prior
return=list(LP=LP,Dev=2*LL,Monitor=1,yhat=1,parm=parm)}
```

- This will be slow, think of a way to speed it up a lot!



Ascending to higher dimensions...

- Now we command the Demon:

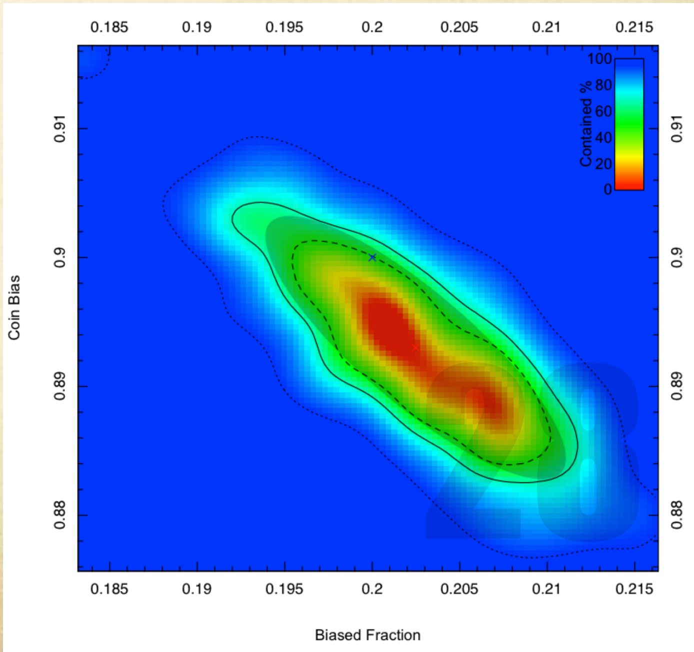
```
> FitCoin2D=LaplacesDemon(Model, Data, Initial.Values=c(0.5,0.5), Algorithm='HARM',
Iterations=1e5, Status=1e4)
> magcon(FitCoin2D$Posterior2[100:1000,1],FitCoin2D$Posterior2[100:1000,2],
conlevels=c(0.5,0.68,0.95), lty=c(2,1,3))
> title(xlab='Biased Fraction',ylab='Coin Bias')
> polygon(ellipse(cov.rob(FitCoin2D$Posterior2)$cov,centre=FitCoin2D$Summary2[,'Mean'],
level=c(pnorm(1)-pnorm(-1))), col=hsv(v=0,alpha=0.2),border=NA)
> points(0.2,0.9,col='blue',pch=4)
> points(FitCoin2D$Summary2[1,'Mean'],FitCoin2D$Summary2[2,'Mean'],
col='red',pch=4)
```

- The best MCMC fit is shown by the red cross and blue shows the 'true' value, so we're pretty close here.

- The solid line shows the 68% of the posterior.

- The shaded polygon shows the 68% implied by the measured covariance of the 2D posterior distribution.

- They agree pretty well- i.e. parameter errors are well represented by covariant Gaussians.

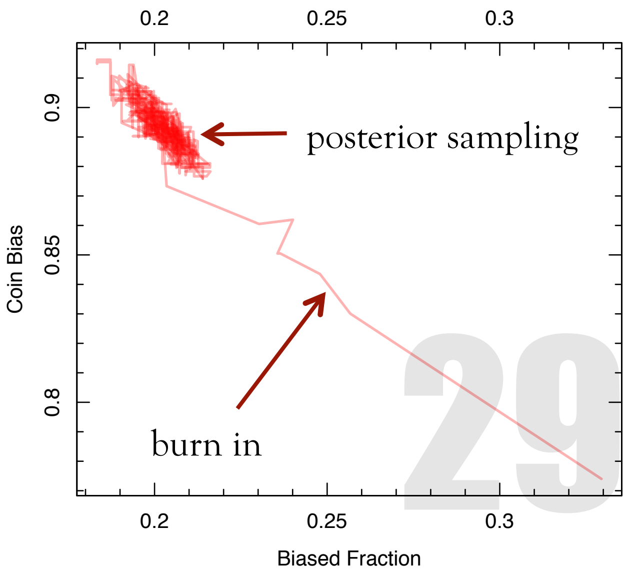


How did we get there?

- It's instructive to look at the MCMC posterior to get a feeling of how the chains make noisy progress towards the answer.

```
> magplot(FitCoin2D$Posterior1,type='l',col=hsv(alpha=0.3),xlab='Biased Fraction',ylab='Coin Bias')
```

- The chain shoots downhill very rapidly with only a few kinks on the journey.
- Once it has got near to the correct solution it jumps around, sampling the posterior parameter space in proportion to the likelihood- which is exactly the feature we want.
- The initial walk towards the solution is called the 'burn in'. In general people throw these links away when doing further analysis. A rule of thumb is to ignore the first 10% of the posterior MCMC links (this gets us into the true Markov Chain regime).



Lecture 2 Problems

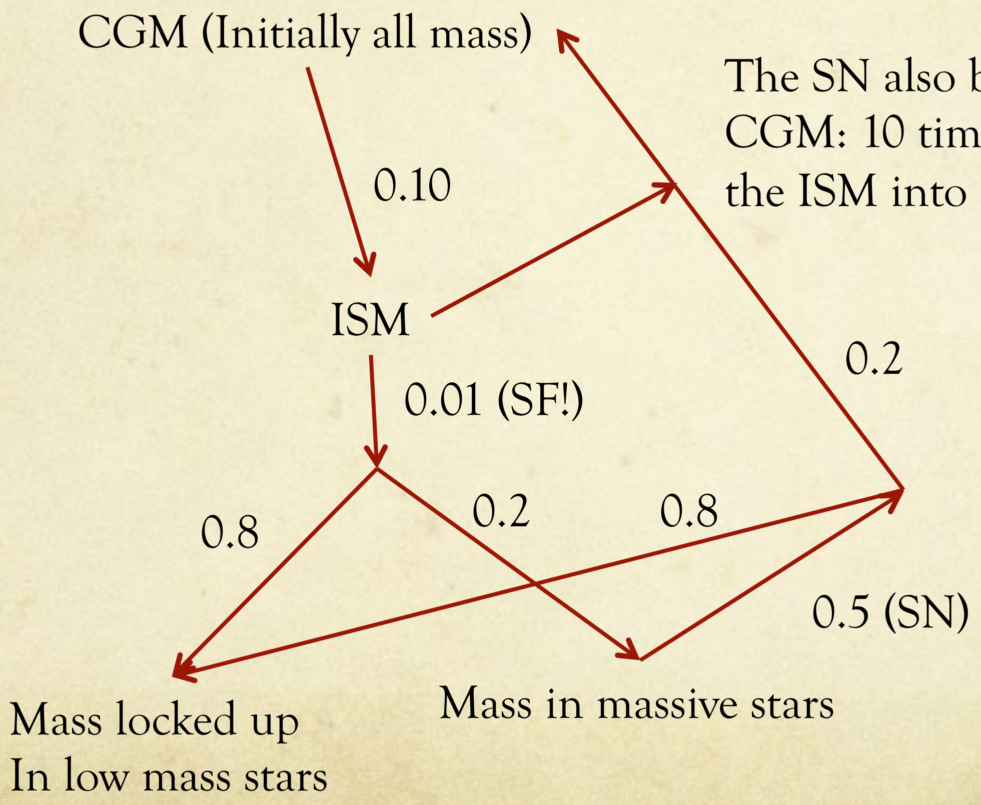
Higher dimensional drunks
Open box model of star formation
A really annoying bag of weird coins

Higher dimensional drunks

- You are an N dimensional drunk starting at the origin of an integer cartesian grid. What is the chance you will revisit the origin when N is 1, 2 or 3? Each step is of size 1 in a random direction on the cartesian grid.
- In the limiting case the chance of returning to the origin is the same as the chance of visiting any other place in cartesian space. i.e. it is the same thing as asking whether, given infinite random steps, you will manage to stumble back to your house.
- You can solve this problem analytically, but it is *much* easier to code up a computational routine that will generically tell you the answer in N dimensions.

Open box model of star formation

- After how many steps is 50% of mass in stars (low and high mass)?



The SN also blows out ISM gas into the CGM: 10 times the SN mass is lost from the ISM into the CGM

A really annoying bag of weird coins

- You are presented with the results of an experiment. Our experimenters sampled 10^4 coins from a bag with a fraction (FracBias) that are biased by CoinBias (the rest that are unbiased). To make our lives even harder they tossed each coin a number of times determined from a Poisson distribution with an unknown λ. All they recorded was the number of heads they recorded each time.
- The priors:
  - FracBias uniform 0 to 1, CoinBias uniform 0 to 1, λ uniform 0 to 10
- This data is available at:
  - https://dl.dropboxusercontent.com/u/10495241/BayesRLec2.dat
  - Put it somewhere then load the data with:

```
> data=read.table('PATH/?/?/BayesRLec2.dat')
```

- What are the values for FracBias, CoinBias and λ?

Summary of Lecture 2

- Some Bayesian problems can be solved by direct numerical (or even analytical) attack with a computer (e.g. the expectation for λ that produces a 0 count).
- Markov Chains are a generic method to iteratively explore posterior parameter space.
- Complex problems may require Markov Chain Monte Carlo techniques (e.g. Metropolis Hastings).
- For harder problems with complex posterior space an MCMC approach is flexible and powerful, and offers more post analysis options due the posterior chains.