oct2023

Olof Swedberg

2024-10-18

1 Probabilistic Graphical Models

- U: Court orders the execution in the actual and hypothetical worlds.
- C: Captain orders fire in the actual world.
- A: Rifleman A shoots in the actual world.
- B: Rifleman B shoots in the actual world.
- D: Prisoner dies in the actual world.
- Ch: Captain orders fire in the hypothetical world.
- Ah: Rifleman A shoots in the hypothetical world.
- Bh: Rifleman B shoots in the hypothetical world.
- Dh: Prisoner dies in the hypothetical world.

```
library(bnlearn)
library(gRain)
```

```
## Loading required package: gRbase
```

```
##
## Attaching package: 'gRbase'
```

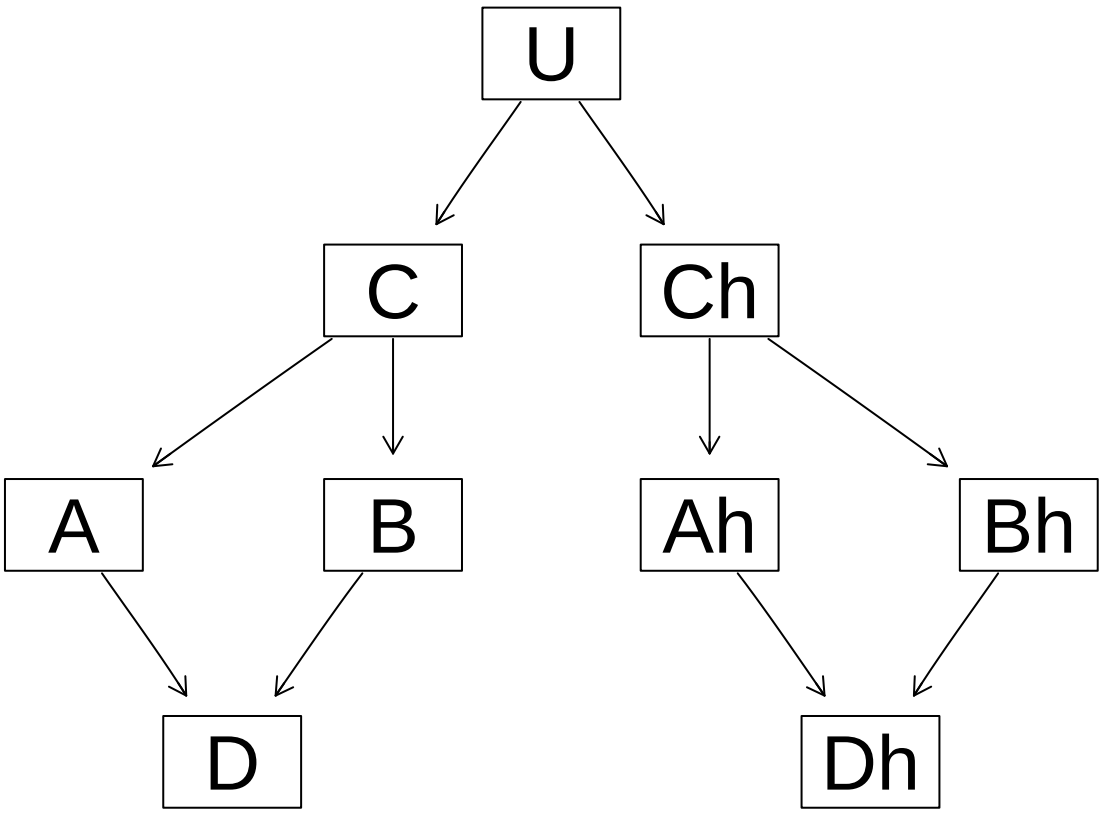
```
## The following objects are masked from 'package:bnlearn':
##
##     ancestors, children, nodes, parents
```

```
#Naive assumption: independence between all features
arc.set = matrix(c("U", "C",
                   "C", "A",
                   "C", "B",
                   "A", "D",
                   "B", "D",
                   "U", "Ch",
                   "Ch", "Ah",
                   "Ch", "Bh",
                   "Ah", "Dh",
                   "Bh", "Dh"),
                 ncol = 2, byrow = TRUE,
                 dimnames = list(NULL, c("from", "to")))

net = empty.graph(c("U", "C", "A", "B", "D", "Ch", "Ah", "Bh", "Dh"))
arcs(net) = arc.set

graphviz.plot(net)
```

```
## Loading required namespace: Rgraphviz
```




```
# U: Prob that Court orders the execution in the actual  and hypothetical worlds
cptU <- c(.5,.5)
dim(cptU) <- c(2)
dimnames(cptU) <- list(c("0", "1"))
print(cptU)
```

```
##   0   1
## 0.5 0.5
```

```
############ REAL WORLD #############
# C: Prob that Captain orders fire in the actual world given U
cptC <- matrix(c(.9,.1,.1,.9), nrow=2, ncol=2)
dim(cptC) <- c(2,2)
dimnames(cptC) <- list("C" = c("0", "1"), "U" =  c("0", "1"))
print(cptC)
```

```
##    U
## C     0   1
##   0 0.9 0.1
##   1 0.1 0.9
```

```
# A: Prob Rifleman A shoots in the actual world given C
cptA = matrix(c(1,0,0.2,0.8), ncol=2, nrow = 2)
dim(cptA) = c(2,2)
dimnames(cptA) = list("A" = c("0","1"), "C" = c("0","1"))
print(cptA)
```

```
##    C
## A   0   1
##   0 1 0.2
```

```
##   1 0 0.8
```

```
# B: Prob Rifleman B shoots in the actual world given C
cptB = matrix(c(1,0,0.2,0.8), ncol=2, nrow = 2)
dim(cptB) = c(2,2)
dimnames(cptB) = list("B" = c("0","1"), "C" = c("0","1"))
print(cptB)
```

```
##    C
## B   0   1
##   0 1 0.2
##   1 0 0.8
```

```
# D: Prob Prisoner dies in the actual world given A (1 = dies) and B
# Question here: how do we know the probability p(D=1|A=0, B=0) = 0.1?
cptD = matrix(c(0.9,0.1,0,1,0,1,0,1), nrow = 2, ncol = 4)
dim(cptD) = c(2,2,2)
dimnames(cptD) = list("D" = c("0", "1"), "A" = c("0","1"), "B" = c("0",1))
print(cptD)
```

```
## , , B = 0
##
##    A
## D     0 1
##   0 0.9 0
##   1 0.1 1
##
## , , B = 1
##
##    A
## D   0 1
##   0 0 0
##   1 1 1
```

```
############ HYPOTHETICAL WORLD #############
# Ch: Prob that Captain orders fire in the actual world given U
cptCh <- matrix(c(.9,.1,.1,.9), nrow=2, ncol=2)
dim(cptCh) <- c(2,2)
dimnames(cptCh) <- list("Ch" = c("0", "1"), "U" =  c("0", "1"))
print(cptCh)
```

```
##    U
## Ch    0   1
##   0 0.9 0.1
##   1 0.1 0.9
```

```
# Ah: Prob Rifleman Ah shoots in the actual world given Ch
cptAh = matrix(c(1,0,0.2,0.8), ncol=2, nrow = 2)
dim(cptAh) = c(2,2)
dimnames(cptAh) = list("Ah" = c("0","1"), "Ch" = c("0","1"))
print(cptAh)
```

```
##    Ch
## Ah  0   1
##   0 1 0.2
##   1 0 0.8
```

```
# Bh: Prob Rifleman Bh shoots in the actual world given Ch
cptBh = matrix(c(1,0,0.2,0.8), ncol=2, nrow = 2)
dim(cptBh) = c(2,2)
dimnames(cptBh) = list("Bh" = c("0","1"), "Ch" = c("0","1"))
print(cptBh)
```

```
##    Ch
## Bh  0   1
##   0 1 0.2
##   1 0 0.8
```

```
# Dh: Prob Prisoner dies in the actual world given Ah (1 = dies) and Bh
# Question here: how do we know the probability p(Dh=1|Ah=0, Bh=0) = 0.1?
cptDh = matrix(c(0.9,0.1,0,1,0,1,0,1), nrow = 2, ncol = 4)
dim(cptDh) = c(2,2,2)
dimnames(cptDh) = list("Dh" = c("0", "1"), "Ah" = c("0","1"), "Bh" = c("0",1))
print(cptDh)
```

```
## , , Bh = 0
##
##    Ah
## Dh    0 1
##   0 0.9 0
##   1 0.1 1
##
## , , Bh = 1
##
##    Ah
## Dh  0 1
##   0 0 0
##   1 1 1
```

```
# Fit the network with the Conditional Probability Tables
netfit <- custom.fit(net,list(U=cptU, C=cptC, A=cptA,
                              B=cptB, D=cptD, Ch=cptCh, Ah=cptAh, Bh=cptBh, Dh=cptDh))
# Compile to grain to be able to set evidence
netcom <- compile(as.grain(netfit))

# Calclate the probability of the Prisoner being dead given hypothetical Gurad A did not shoot
# given that he is dead in the real world
evidence = querygrain(setEvidence(netcom,nodes=c("D","Ah"),states=c("1","0")))
evidence$Dh
```

```
## Dh
##         0         1
## 0.6209572 0.3790428
```

2 Hidden Markov Models

```
set.seed(12345)
library(HMM)

# Sector X: Steps -- implement this with a counter
# Sector 1: 2
```

```
# Sector 2: 3
# Sector 3: 2
# Sector 4: 1
# Sector 5: 2

states=c("Z=1.C=2","Z=1.C=1",
         "Z=2.C=3","Z=2.C=2","Z=2.C=1",
         "Z=3.C=2","Z=3.C=1",
         "Z=4.C=1",
         "Z=5.C=2","Z=5.C=1")

emissionSymbols = 1:5
transitionProb = matrix(0,nrow = length(states), ncol = length(states))

transitionProb = matrix(c(0, 1, 0, 0, 0, 0, 0, 0, 0, 0, # Z=1.C=2
                          0,.5,.5, 0, 0, 0, 0, 0, 0, 0, # Z=1.C=1 Can stay or move forward
                          0, 0, 0, 1, 0, 0, 0, 0, 0, 0, # Z=2.C=3
                          0, 0, 0, 0, 1, 0, 0, 0, 0, 0, # Z=2.C=2
                          0, 0, 0, 0,.5,.5, 0, 0, 0, 0, # Z=2.C=1 Can stay or move forward
                          0, 0, 0, 0, 0, 0, 1, 0, 0, 0, # Z=3.C=2
                          0, 0, 0, 0, 0, 0,.5,.5, 0, 0, # Z=3.C=1 Can stay or move forward
                          0, 0, 0, 0, 0, 0, 0,.5,.5, 0, # Z=4.C=1 Can stay or move forward
                          0, 0, 0, 0, 0, 0, 0, 0, 0, 1, # Z=5.C=2
                          .5, 0, 0, 0, 0, 0, 0, 0, 0,.5), # Z=5.C=1 Can stay or move forward
                        nrow = length(states), ncol = length(states), byrow = TRUE)
b = 1/3
emissionProb = matrix(c(b, b, 0, 0, b, # Z=1.C=2
                        b, b, 0, 0, b, # Z=1.C=1 Can stay or move forward
                        0, b, b, b, 0, # Z=2.C=3
                        0, b, b, b, 0, # Z=2.C=2
                        0, b, b, b, 0, # Z=2.C=1 Can stay or move forward
                        0, 0, b, b, b, # Z=3.C=2
                        0, 0, b, b, b, # Z=3.C=1 Can stay or move forward
                        b, 0, 0, b, b, # Z=4.C=1 Can stay or move forward
                        b, 0, 0, b, b, # Z=5.C=2
                        b, b, 0, 0, b), # Z=5.C=1 Can stay or move forward
                        nrow = length(states), ncol = length(emissionSymbols), byrow = TRUE)

# Assuming we start at with full countdown
startProb = c(.2, 0,.2, 0, 0,.2, 0, .2,.2,0)

hmm = initHMM(States = states, Symbols = emissionSymbols,
              startProbs = startProb, transProbs = transitionProb,
              emissionProbs = emissionProb)
nIter = 100
simulation = simHMM(hmm, nIter)
print(simulation)
```

```
## $states
##   [1] "Z=5.C=2" "Z=5.C=1" "Z=5.C=1" "Z=5.C=1" "Z=1.C=2" "Z=1.C=1" "Z=2.C=3"
##   [8] "Z=2.C=2" "Z=2.C=1" "Z=2.C=1" "Z=3.C=2" "Z=3.C=1" "Z=4.C=1" "Z=4.C=1"
##  [15] "Z=4.C=1" "Z=4.C=1" "Z=4.C=1" "Z=4.C=1" "Z=4.C=1" "Z=5.C=2" "Z=5.C=1"
##  [22] "Z=1.C=2" "Z=1.C=1" "Z=1.C=1" "Z=1.C=1" "Z=2.C=3" "Z=2.C=2" "Z=2.C=1"
##  [29] "Z=3.C=2" "Z=3.C=1" "Z=4.C=1" "Z=4.C=1" "Z=4.C=1" "Z=5.C=2" "Z=5.C=1"
```

```
##  [36] "Z=1.C=2" "Z=1.C=1" "Z=1.C=1" "Z=1.C=1" "Z=2.C=3" "Z=2.C=2" "Z=2.C=1"
##  [43] "Z=2.C=1" "Z=2.C=1" "Z=3.C=2" "Z=3.C=1" "Z=3.C=1" "Z=3.C=1" "Z=3.C=1"
##  [50] "Z=4.C=1" "Z=5.C=2" "Z=5.C=1" "Z=1.C=2" "Z=1.C=1" "Z=1.C=1" "Z=2.C=3"
##  [57] "Z=2.C=2" "Z=2.C=1" "Z=3.C=2" "Z=3.C=1" "Z=4.C=1" "Z=4.C=1" "Z=5.C=2"
##  [64] "Z=5.C=1" "Z=5.C=1" "Z=1.C=2" "Z=1.C=1" "Z=2.C=3" "Z=2.C=2" "Z=2.C=1"
##  [71] "Z=2.C=1" "Z=2.C=1" "Z=3.C=2" "Z=3.C=1" "Z=3.C=1" "Z=4.C=1" "Z=5.C=2"
##  [78] "Z=5.C=1" "Z=5.C=1" "Z=1.C=2" "Z=1.C=1" "Z=1.C=1" "Z=2.C=3" "Z=2.C=2"
##  [85] "Z=2.C=1" "Z=3.C=2" "Z=3.C=1" "Z=4.C=1" "Z=5.C=2" "Z=5.C=1" "Z=5.C=1"
##  [92] "Z=5.C=1" "Z=1.C=2" "Z=1.C=1" "Z=1.C=1" "Z=1.C=1" "Z=1.C=1" "Z=1.C=1"
##  [99] "Z=2.C=3" "Z=2.C=2"
##
## $observation
##   [1] 1 5 2 5 5 2 3 4 2 2 3 3 4 5 4 4 4 5 1 1 5 5 1 2 2 4 4 2 5 4 1 4 4 5 2 2 2
##  [38] 1 1 4 3 4 2 3 5 4 4 3 4 5 4 5 2 5 1 4 4 3 4 3 4 5 4 2 5 5 2 2 4 3 4 2 4 5
##  [75] 3 4 4 2 2 2 1 5 2 2 4 4 3 4 1 1 2 2 1 1 2 1 1 2 2 3
```

Prove this expression:

$p(z^{T-1}|x^{0:T}) = \Sigma_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T})p(z^T|x^{0:T})}{\Sigma_{z^{T-1}} p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})}$

Step 1: Marginalization over z^T
$p(z^{T-1}|x^{0:T}) = \sum_{z^T} p(z^{T-1}, z^T|x^{0:T})$

This step comes from the law of total probability, where you marginalize over the hidden state z^T at time T. Essentially, you are summing over all possible values of z^T to get the marginal distribution of z^{T-1} conditioned on the observations.

Step 2: Conditional Probability Decomposition
$= \sum_{z^T} p(z^{T-1}, z^T|x^{0:T}) = \sum_{z^T} p(z^{T-1}|z^T, x^{0:T})p(z^T|x^{0:T})$

This is just applying the product rule of conditional probability. You are expressing the joint probability $\sum_{z^T} p(z^{T-1}, z^T|x^{0:T})$ as the product of the conditional probability $\sum_{z^T} p(z^{T-1}|z^T, x^{0:T})$ and the marginal probability $p(z^T|x^{0:T})$.

Step 3: Exploit conditional independence
Next, simplify $\sum_{z^T} p(z^{T-1}|z^T, x^{0:T})$ Due to the Markov property (which assumes that the current state only depends on the previous state and is conditionally independent of earlier states given the current state), we can simplify it to the expression: $p(z^{T-1}|z^T, x^{0:T-1})$. The whole expression:

$= \sum_{z^T} p(z^{T-1}|z^T, x^{0:T})p(z^T|x^{0:T}) = \sum_{z^T} p(z^{T-1}|z^T, x^{0:T-1})p(z^T|x^{0:T})$

Step 4: Apply Bayes rule
Apply Bayes rule to the expression $p(z^{T-1}|z^T, x^{0:T-1})$ to decompose the term into known conditional probabilites.

$= \sum_{z^T} p(z^{T-1}|z^T, x^{0:T-1})p(z^T|x^{0:T}) = \sum_{z^T} \frac{p(z^T|z^{T-1}, x^{0:T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{p(z^T|x^{0:T-1})}$

Step 5: Simplification Using Conditional Independence
Simplify $p(z^T|z^{T-1}, x^{0:T-1})$ to $p(z^T|z^{T-1})$ Since the Markov property tells us that only z^T only depends on z^{T-1}, not the full observation history:

$= \sum_{z^T} \frac{p(z^T|z^{T-1}, x^{0:T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{p(z^T|x^{0:T-1})} = \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{p(z^T|x^{0:T-1})}$

Step 6: Denominator Expansion

In this step, expand $p(z^T|x^{0:T-1})$ using the law of probability

$= \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{p(z^T|x^{0:T-1})} = \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{\sum_{z^{T-1}} p(z^{T-1},z^T|x^{0:T-1})}$

Step 7: Expand even more

$= \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{\sum_{z^{T-1}} p(z^{T-1},z^T|x^{0:T-1})} = \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{\sum_{z^{T-1}} p(z^T|z^{T-1},x^{0:T-1})p(z^{T-1}|x^{0:T-1})}$

Step 8: Use the Markov Property

$= \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{\sum_{z^{T-1}} p(z^T|z^{T-1},x^{0:T-1})p(z^{T-1}|x^{0:T-1})} = \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{\sum_{z^{T-1}} p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})}$

Step 9: Conclusion

$p(z^{T-1}|x^{0:T}) = \sum_{z^T} \frac{p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})p(z^T|x^{0:T})}{\sum_{z^{T-1}} p(z^T|z^{T-1})p(z^{T-1}|x^{0:T-1})}$

3 Reinforcement Learning

```
R = array(0,dim=10)
R[10] = 1
V = array(0, dim = 10) # state values
pi = array(0, dim = 10) # policies
theta = 0.1
gamma = 0.95

repeat{
  delta = 0
  for (s in 1:9) {
    v = V[s]
    V[s] = max(gamma*V[s], R[s+1] + gamma*V[s+1]) # Why add gamma in the first term like Jose?
    delta = max(delta, abs(v-V[s]))
  }
if(delta<theta) break
}

for(s in 1:9) {
  pi[s] <- which.max(c(gamma*V[s],R[s+1]+gamma*V[s+1]))
  # Why add gamma in the first term like Jose?
}

V
```

```
##  [1] 0.6634204 0.6983373 0.7350919 0.7737809 0.8145062 0.8573750 0.9025000
##  [8] 0.9500000 1.0000000 0.0000000
```

```
pi
```

```
##  [1] 2 2 2 2 2 2 2 2 2 0
```

Gaussian Processes

P1: Covariance

```
library(kernlab)

SquaredExpKernel <- function(sigmaF=1,ell=3) {
  rval <- function(x1, x2) {
    n1 <- length(x1)
    n2 <- length(x2)
    K <- matrix(NA,n1,n2)
    for (i in 1:n2) {
      K[,i] = sigmaF^2*exp(-0.5*( (x1-x2[i])/ell)^2 )
    }
    return(K)
    }
  class(rval) <- "kernel"
  return(rval)
  }

ExtendedSquaredExpKernel <- function(sigmaF,ell1, ell2, d){
  rval <- function(x1, x2) {
    n1 <- length(x1)
    n2 <- length(x2)
    K <- matrix(NA,n1,n2)
    for (i in 1:n1) {
      for (j in 1:n2) {
        diff = sqrt(crossprod(x1[i] - x2[j]))
        K[i, j] = sigmaF^2*exp(-2*sin((3.14*diff)/d)^2/ell1^2) * exp(-0.5*diff^2/ell2^2)
      }
    }
    return(K)
  }
  class(rval) <- "kernel"
  return(rval)
}


SE_kernel1 = SquaredExpKernel(20,100)
SE_kernel2 = ExtendedSquaredExpKernel(20,1,1000,365) # Should be 1000 here, not 100
print("SE Kernel")
```

```
## [1] "SE Kernel"
```

```
SE_kernel1(c(1,182,365),c(1,182,365))
```

```
##             [,1]      [,2]        [,3]
## [1,] 400.0000000  77.74347   0.5308183
## [2,]  77.7434690 400.00000  74.9644909
## [3,]   0.5308183  74.96449 400.0000000
```

```
print("SE Periodic Kernel")
```

```
## [1] "SE Periodic Kernel"
```

```
SE_kernel2(c(1,182,365),c(1,182,365))
```

```
##           [,1]      [,2]      [,3]
## [1,] 400.00000  53.27459 374.28168
## [2,]  53.27459 400.00000  53.23652
## [3,] 374.28168  53.23652 400.00000
```

Here we can see that the periodic kerlen has much lower covariance between data points half a year apart (1,2) which corresponds to (1, 182). Meanwhile, points 1 year appart (1,3) which corresponds to (1,365) has high correlation. This is because the periodic behaviour of the kernel, ensuring high correlation between the same day but different years.

P2: Covariance

```
posteriorGP = function(X, y, sigmaNoise, XStar, k, ...) {
  # Line 2
  n = length(X) # No of training points
  K = k(X,X,...) # Covariance for training points
  kStar = k(X,XStar,...) # Covariance for training and test points
  # Cholesky decomposition, Lower triangular matrix
  L = t(chol(K + (sigmaNoise**2) * diag(n)))
  alpha = solve(t(L), solve(L, y))
  # Line 4
  fStar = t(kStar)%*%alpha # posterior mean
  v = solve(L, kStar)
  # Line 6 :  Posterior variance
  V_fStar = k(XStar, XStar,...) - t(v)%*%v
  log_marg_likelihood = -(0.5)*(t(y)%*%alpha) - sum(log(diag(L))) - (n/2)*log(2*3.14)
  return(list(mean = fStar, variance = V_fStar, log_likelihood = log_marg_likelihood))
}

data = read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTullinge.cs
                header=TRUE, sep=";")
data$time = 1:nrow(data)
# Subsample every 5th observation
subsample_idx = seq(1, nrow(data), by = 5)
data_sub = data[subsample_idx, ]
temp = data_sub$temp
time = data_sub$time

polyFit = lm(temp ~ time + I(time^2))
sigmaNoise = var(polyFit$residuals)

posterior1 = posteriorGP(X=time, y=temp, sigmaNoise = sigmaNoise,
                         XStar = time, k = SquaredExpKernel(20,100))
posterior2 = posteriorGP(X=time, y=temp, sigmaNoise = sigmaNoise,
                          XStar = time, k = ExtendedSquaredExpKernel(20,1,1000,365))

posterior1$log_likelihood
```

```
##          [,1]
## [1,] -2254.87
```

```
posterior2$log_likelihood
```

```
##          [,1]
## [1,] -2251.63
```

```
# Not sure why I'm not getting the same results ase Jose.
# i have dubble checked every difference between my code
# and his, and cant find any differences in the results.
```

We cant create a validation set because the data is not iid. The temperature increases over time and we cant sample some random data to make predictions, as that would mean using the future to predict the past.