STAT545 HW 6

Yi Yang

11/22/2018

1 Problem 1: Generating gamma random variables via rejection sampling

We will first generate an exponential random variable.

1. Let U be a Uniform(0,1) random variable. How will you transform this to generate an exponential with rate λ? Use the inverse-cdf method.
2. Let Y be a random variable on $[0, 1)$, with density $p(Y = y) \propto 1/y^a$, where $a \in (0, 1)$. This is the truncated Pareto distribution. What is its normalization constant? Describe the inverse-cdf method to generate Y by transforming U.
3. Write down the density of a Gamma random variable with shape parameter α and rate parameter β. Call this $p_{Gamma}(x|\alpha, \beta)$. Note that calculating its cdf is not easy.
4. Note that the sum of two exponentials with rate 1 is distributed as Gamma(2,1). How would you use previous results to sample from a Gamma distribution with parameters (alpha_int,1), where alpha_int is an integer.

Next, we will generate a Gamma random variable, with alpha < 1

5. For $\alpha < 1$, $p_{Gamma}(x|\alpha, 1) \leq C_1 x^{\alpha-1}$ for $x < 1$, and $p_{Gamma}(x) \leq C_2 \exp(-x)$ for $x \geq 1$. What are C_1 and C_2?
6. Use this information to write down a probability density $q(x|\alpha)$ and a scalar M such that $p_{Gamma}(x|\alpha, 1) \leq M \cdot q(x|\alpha)$.
7. Now provide pseudocode for a rejection sampler to smaple from $p_{Gamma}(x|\alpha, 1)$. Write down the acceptance probability.
8. How would you use this to generate a Gamma with arbitrary shape parameter, and rate equal to one. How about arbitrary shape and rate parameters?

Solution:

1. The CDF of an exponential distribution is given by

$$F(x) = 1 - e^{-\lambda x}$$

If $P = F(X)$ is uniformly disributed, the inverse of the $F(p)$ satisfies an exponential distribution. That is,

$$F^{-1}(p) = -\frac{\log(1-p)}{\lambda} \sim \text{Expon}(\lambda)$$

where p is an random variable uniformly distributed.

2. The normalization constant for the truncated Pareto distribution is given by

$$Z_Y = \int_0^1 \frac{1}{y^a} dy = \frac{1}{1-a}$$

The PDF of Pareto distribution is given by

$$p(Y = y) = \frac{1-a}{y^a}$$

The CDF of Pareto distribution is given by

$$F(Y \leq y) = \int_0^y \frac{1-a}{t^a} dt = y^{1-a}$$

The Pareto distribution can be sampled from a uniform distribution by inverting the CDF of the Pareto distribution.

$$F^{-1}(U) = U^{a-1} \sim \text{Pareto}(a)$$

3. The density function of a Gamma distribution is given by

$$p_{Gamma}(x|\alpha, \beta) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} \beta^{\alpha} \exp(-\beta x)$$

with $x > 0$ and $\alpha, \beta > 0$.

4. From the stated results, it is implied that if $X_i \overset{\text{iid}}{\sim} \text{Expon}(\lambda)$, then $Y = X_1 + X_2 + \cdots + X_k \sim \text{Gamma}(k, \lambda)$. Independently draw alpha_int samples from exponential distribution Expon(1), add them up will give a Gamma distribution with parameter (alpha_int,1).

5. The C_1 and C_2 are given by

$$C_1 = \max_{\substack{0<x<1 \\ 0<\alpha<1}} \{\frac{1}{\Gamma(\alpha)} \exp(-x)\} = \frac{1}{\Gamma(\alpha)}$$

$$C_2 = \max_{\substack{x \geq 1 \\ 0<\alpha<1}} \{\frac{1}{\Gamma(\alpha)} x^{\alpha-1}\} = \frac{1}{\Gamma(\alpha)}$$

6. The probability density can be given as

$$q(x|\alpha) = \frac{x^{\alpha-1}}{\frac{1}{\alpha} + e^{-1}} \quad \text{for } 0 < x < 1$$

$$q(x|\alpha) = \frac{\exp(-x)}{\frac{1}{\alpha} + e^{-1}} \quad \text{for } x \geq 1$$

The scalar coefficient is given by $M = \frac{1}{\Gamma(\alpha)} \left(\frac{1}{\alpha} + e^{-1}\right)$.

7. The pseudocode for rejection sampling is given by

```
Sample xs ~ q(x|alpha)
Sample u ~ Unif(0,1)
if (u > pGamma(xs|alpha,1)/(M*q(xs|alpha))){
  xs is rejected
} else{
  xs is accepted
}
```

The acceptance probability is given by

$$p(\text{accept}) = \int \frac{p_{Gamma}(x|\alpha, 1)}{M q(x|\alpha)} q(x|\alpha) dx = \frac{1}{M} \int p_{Gamma}(x|\alpha, 1) dx = \frac{1}{M}$$

8. In the previous pseudocode, we see that $q(x|\alpha)$ is a combination of the trancated Pareto distribution and the exponential distribution. Therefore, we can apply the pseudocode in 7 to sample a Gamma distribution with arbitray shape parameter and the unity rate parameter.

For a Gamma distribution with arbitray shape parameter and rate paramter, let us consider $q(x|\alpha, \beta) = p_{Gamma}(k, \beta - 1)$, where $k = \text{Floor}(\alpha)$. The ratio of the Gamma density to the new proposal probability is

$$\frac{p_{Gamma}(x|\alpha, \beta)}{q(x|\alpha, \beta)} = \frac{\Gamma(k)\beta^{\alpha}}{\Gamma(\alpha)(\beta-1)^k} x^{\alpha-k} \exp(-x)$$

This ratio obtains its maximum when $x = \alpha - k$. Therefore, seting $M = \frac{p_{Gamma}(\alpha-k|\alpha,\beta)}{p_{Gamma}(\alpha-k|k,\beta-1)}$ and proceed the rejection sampling will give samples from a Gamma distribution with arbitray shape and rate parameter.

2 Problem 2: Exploring the space of permutations with Metropolis-Hastings

The file message.txt on the course webpage gives a paragraph of English encoded by a permutation code, where every symbol is mapped to a (usually) different one. The encryption key σ might thus be:

$$a \to v; b \to l; c \to n; d \to \cdot; \ldots$$

and a message like "beware of dog." mgght read as "lgavfgp wp. c". To make life simple, assume there are only 30 unique symbols, Symbol = ('a','b',...,'z',':',',','' and ':'). Thus the encription is a bijective function σ: Symbol $\to$ Symbol. Your job is to recover the original message (or equivalently find σ).

1. How many possible functions σ are there?

Clearly, brute force search is impossible. instead, we will fit and use a model of the English language.

Download a large corpus of English (I used 'War and Peace' from Project Gutenberg.) Use your favourite text-editor to cover all text to lowercase. You might also want to delete some of the clutter at the beginning and the end of the file.

2. Read the file into R using the readChar() command. This will return a long string, split it up into a list of characters using the strsplit() command. The table() command will then give you the frequency of each unique character. This text contains more characters than we are interested in (we only care about Symbol). Extract the appropriate symbols from the table, and calculate and plot a histogram of the frequency of each character in Symbol. What is the entropy of this distribution? (This is a very crude estimate of the entropy of the English language, if you are interested, see http://www.math.ucsd.edu/~crypto/java/ENTROPY/) for how Shannon got a better (model-free) estimate using human subjects.)

We will actually model English text as a first-order Markov process, characterized by a 30 X 30 transition matrix T. We will write this as $P_{Eng}(\cdot|T)$ (assume the distribution of the first symbol is uniform).

3. Given some data D, what are its sufficient statistics? These don't have to be minimal. What is the joint probability? What is the maximum likelihood estimate of T given D?

4. Now estimate T for the English text you download. The simplest way to do this is by understanding what table() with two arguments does and then running table(txt[1:length_txt-1], txt[2:length_txt]). Plot the estimated T using a heatmap. For which symbols does the distribution of the following symbol have the largest and smallest entropy?

We can now write a Bayesian model to estimate the permutation σ. We place a uniform prior over σ:

$$p(\sigma) \propto 1$$

Given a message X encoded with σ, we expect $\sigma^{-1}(X)$ to be English text. We can evaluate the probability it is English using our Markov model of English:

$$p(X|\sigma) = P_{Eng}(\sigma^{-1}(X)|T)$$

5. Write a function for this likelihood. It should take a transition matrix T, a message X (or the sufficient statistics of X) and a permutation perm, and calculate the log-probability. (I implemented the permutation as a named-list, so that e.g. perm['a'] gives 'v'). We are interested in the posterior $p(\sigma|X) \propto p(X|\sigma)p(\sigma)$.

6. Explain briefly why directly sampling from this is not easy.

Instead, we will use a simple Metropolis-Hastings algorithm. Start with an arbitray permutation (I used the identity map). Randomly pick two symbols and propose a new permutation that swaps these values of the pervious permutation.

7. Write down the proposal distribution and the acceptance probability?

8. Finally implement this MH algorithm and run it on message.txt. I recovered the (almost perfect) answer after about 3000 iterations. Plot the evolution of the likelihood over your MCMC run. Also print the first 20 characters of the decoded message every 100 iterations. It might help to start by using your own encoded message where you know the true answer. Specify how many iterations your ran it for and what your burn-in period was.
9. Under the posterior distirbution, which of the oberved symbols had the highest and lowest uncertainty about their true value? Which ones did it seem to get wrong?
10. Consider an i.i.d. model of English text, where we assume the next symbol is drawn from the solution to part 2? Does this work? All you have to do is change the function from part 5. Again, plot the decoded output every 100 iterations.
11. Comment on the usefulness and feasibility of using higher-order Markov models for decoding (where the next symbol depends e.g. on the previous two or five symbols).

Solution:

1. There are 30! possibilities of σ.
2.

```
fileName <- 'war-and-peace.txt'
warofpeace <- readChar(fileName, file.info(fileName)$size)
wopList <- strsplit(warofpeace,'')
wopList <- wopList[[1]]
table(wopList)
```

```
## wopList
##     \n     \r            -      ,      ;      :      !      ?      .
##  65007  65007 514908   6308  39891   1145   1014   3923   3137  30805
##      '      "      (      )      [      ]      @      *      /      #
##   7529  17970    670    670      1      1      2    300     29      1
##      %      =      $      0      1      2      3      4      5      6
##      1      2      2    179    392    147     61     23     55     57
##      7      8      9      a      à      ä      b      c      d      e
##     40    193     35 205806      4      1  34658  61623 118290 315233
##      é      ê      f      g      h      i      j      k      l      m
##      1     11  54896  51326 167404 174281   2574  20431  96527  61646
##      n      o      p      q      r      s      t      u      v      w
## 184173 192879  45533   2330 148428 162891 226406  65434  27086  59207
##      x      y      z
##   4384  46265   2388
```

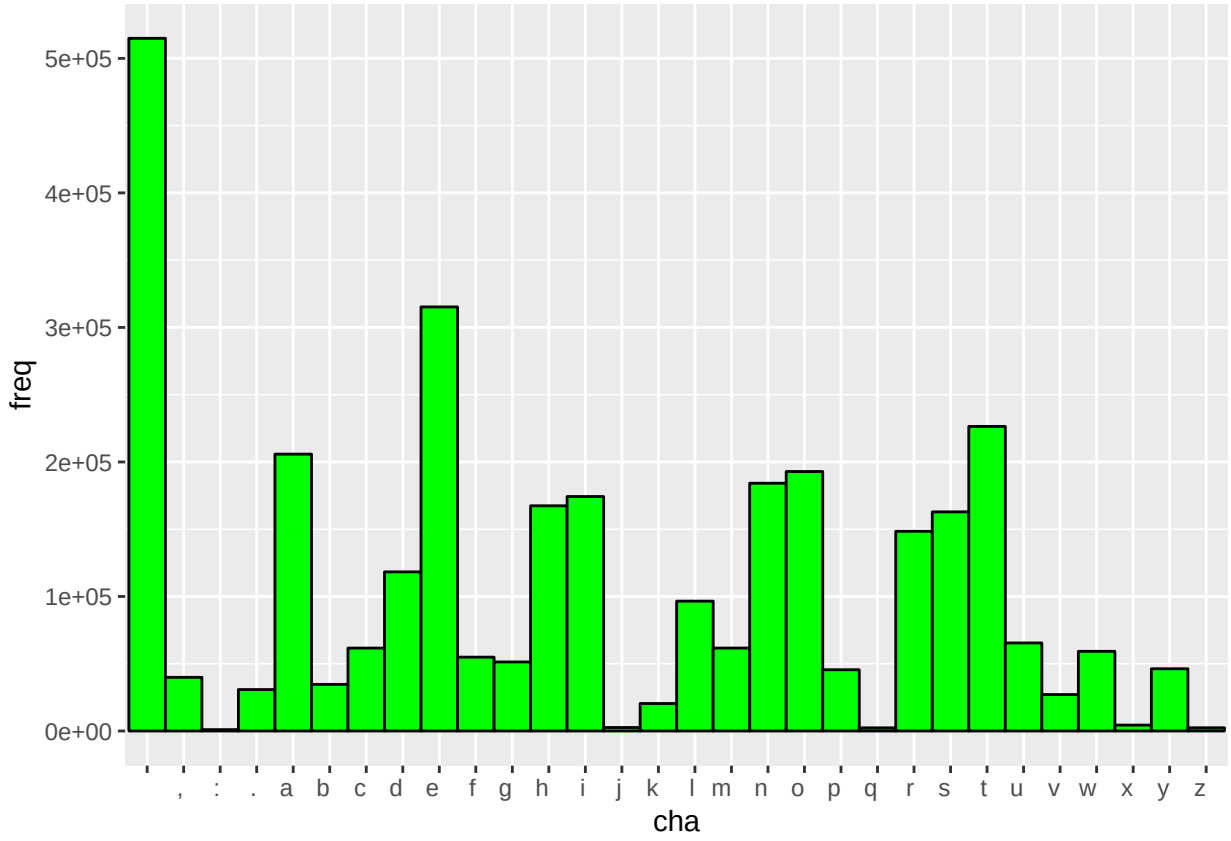
```
wopList_rev <- sub('[^a-z\\.\\,\\: ]','',wopList)
wopTable <- table(wopList_rev)
#wopTable <- wopTable[, !names(wopTable) %in% c('')]
#wopTable
names(wopTable)
```

```
##  [1] ""  " " "," ":" "." "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
## [18] "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```
wopDF <- data.frame(cha=names(wopTable),freq=as.vector(wopTable))
```

```
#wopDF
wop.df <- wopDF[-c(1),]
row.names(wop.df) <- seq(1,30)
library(ggplot2)
```

```
ggplot(data=wop.df, aes(cha, freq)) + geom_bar(stat = 'identity', width = 1, color = 'black', fill = 'g
  theme(axis.text.x=element_text(angle = 0, hjust=1))
```




```
wop.df_freq <- wop.df$freq/sum(wop.df$freq)
entropy <- - sum(wop.df_freq * log(wop.df_freq))
entropy
```

```
## [1] 2.915402
```

The entropy of a discrete distribution is given by

$$\mathbb{H}(X) = -\sum_{k=1}^{K} p(X = k) \log p(X = k)$$

Hence, the entropy for this distribution is 2.915402.

3. The sufficient statistic is a function $\phi(X)$ whose value contains all the information needed to compute any estimate of the parameter. According to the factorization theorem, the joint probability is given by $p(X) = h(X)g(\theta, \phi(X))$. From the joint likelihood probability, it can be seen the maximum likelihood estimate of θ depends only on $\phi(X)$, which satifies $\frac{\partial g(\theta,\phi(X))}{\partial \theta} = 0$.

4. The estimate of T and the heatmap of T are given as follows,

```
Trans <- table(wopList_rev[1:length(wopList_rev)-1],wopList_rev[2:length(wopList_rev)])
Trans <- Trans[2:31,2:31]
row.sums <- rowSums(Trans)
for (i in seq(1,30)){
  Trans[i,] <- Trans[i,]/row.sums[i]
}
```

```
Trans <- as.matrix(Trans)
Trans
```

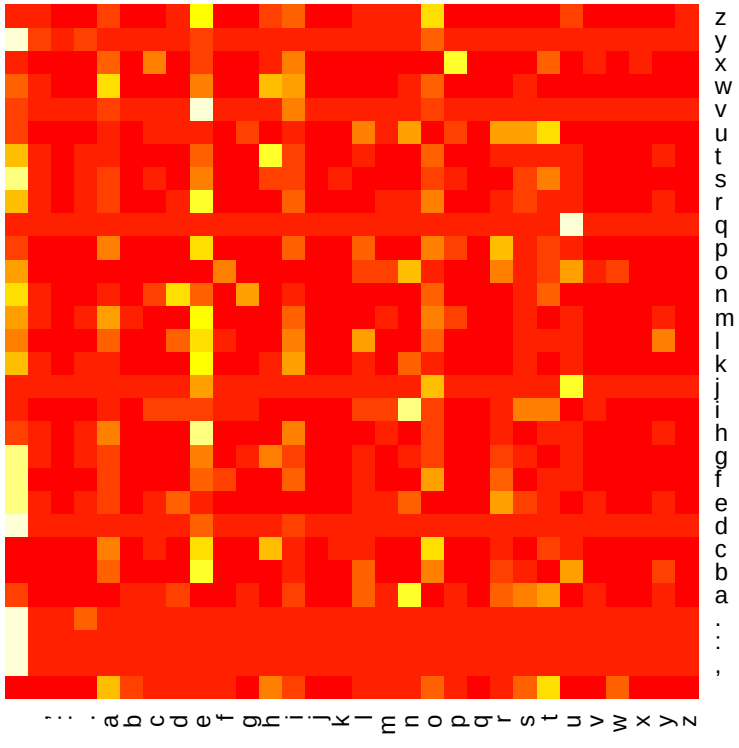
```
##
##                             ,            :            .            a
##     3.291472e-04 0.000000e+00 0.000000e+00 0.000000e+00 1.248467e-01
##   , 9.999422e-01 0.000000e+00 2.889004e-05 2.889004e-05 0.000000e+00
##   : 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   . 7.902559e-01 4.709823e-04 0.000000e+00 2.054529e-01 2.093254e-04
##   a 6.027072e-02 4.719524e-03 2.940513e-05 2.548445e-03 3.430599e-05
##   b 3.349310e-03 1.010568e-03 0.000000e+00 5.774672e-04 9.259687e-02
##   c 7.638302e-03 6.175648e-04 1.625171e-05 4.875512e-04 1.092927e-01
##   d 5.628628e-01 3.731171e-02 1.655924e-03 2.512375e-02 2.194544e-02
##   e 2.992882e-01 2.008926e-02 5.192381e-04 1.440721e-02 4.382961e-02
##   f 3.194330e-01 9.344213e-03 3.572234e-04 7.952922e-03 7.249756e-02
##   g 3.616191e-01 2.969496e-02 1.173234e-03 2.128004e-02 6.859374e-02
##   h 7.409393e-02 7.656996e-03 8.434141e-05 3.301364e-03 1.670803e-01
##   i 2.492327e-02 1.300495e-03 5.779979e-06 7.513973e-04 1.682552e-02
##   j 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 3.735409e-02
##   k 1.880000e-01 2.775000e-02 5.000000e-04 1.935000e-02 2.865000e-02
##   l 1.033700e-01 1.461257e-02 2.206068e-04 6.345071e-03 8.074208e-02
##   m 1.393522e-01 2.370935e-02 7.424640e-04 2.709169e-02 1.555545e-01
##   n 1.809764e-01 1.685321e-02 3.967434e-04 1.122057e-02 3.082305e-02
##   o 1.235992e-01 4.569780e-03 7.345226e-05 1.956978e-03 7.235047e-03
##   p 5.004196e-02 8.413939e-03 2.870898e-04 5.984718e-03 1.046111e-01
##   q 4.291845e-04 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   r 1.653364e-01 1.948384e-02 4.211602e-04 1.447134e-02 5.771966e-02
##   s 3.041701e-01 3.868409e-02 8.975575e-04 2.185966e-02 5.351607e-02
##   t 2.185623e-01 1.461397e-02 2.715510e-04 1.119695e-02 4.023933e-02
##   u 4.578964e-02 4.765577e-03 7.711289e-05 2.961135e-03 2.040407e-02
##   v 5.745559e-02 2.147483e-02 2.272469e-04 1.060486e-02 6.999205e-02
##   w 1.043213e-01 1.669552e-02 2.571179e-04 9.170538e-03 2.084712e-01
##   x 4.205056e-02 8.740846e-03 7.087172e-04 4.488542e-03 7.961257e-02
##   y 5.446197e-01 7.219189e-02 2.336232e-03 4.987047e-02 3.083364e-02
##   z 2.957330e-02 1.309675e-02 4.224757e-04 5.914660e-03 5.787917e-02
##
##                 b            c            d            e            f
##     4.399404e-02 3.551655e-02 2.992300e-02 2.038165e-02 3.798319e-02
##   , 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   : 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   . 5.233136e-05 1.046627e-04 5.233136e-05 1.151290e-03 5.233136e-04
##   a 1.695206e-02 3.438930e-02 5.546788e-02 8.282446e-04 8.208933e-03
##   b 6.612000e-03 8.662008e-05 4.619738e-04 3.289253e-01 0.000000e+00
##   c 0.000000e+00 1.704804e-02 1.462654e-04 2.134987e-01 0.000000e+00
##   d 1.068338e-04 8.902817e-05 1.160927e-02 1.224939e-01 7.567394e-04
##   e 8.347245e-04 1.785785e-02 9.408726e-02 2.657645e-02 1.040119e-02
##   f 5.076333e-04 0.000000e+00 0.000000e+00 8.686170e-02 5.328270e-02
##   g 2.022817e-05 0.000000e+00 8.698115e-04 1.190023e-01 0.000000e+00
##   h 3.735120e-04 3.735120e-04 3.614632e-04 4.533170e-01 4.096583e-04
##   i 7.750952e-03 4.933212e-02 5.001416e-02 4.885239e-02 2.164602e-02
##   j 0.000000e+00 0.000000e+00 0.000000e+00 2.194553e-01 0.000000e+00
##   k 3.500000e-04 2.350000e-03 1.500000e-04 2.830000e-01 6.500000e-04
##   l 7.458610e-04 7.353559e-04 7.261114e-02 1.797210e-01 2.419321e-02
##   m 1.790163e-02 5.939712e-04 0.000000e+00 2.503424e-01 2.194394e-03
```

```
##   n 8.437781e-04 5.086697e-02 1.931805e-01 8.288583e-02 5.096196e-03
##   o 4.700944e-03 7.219307e-03 1.601784e-02 2.639035e-03 8.876180e-02
##   p 2.429221e-04 2.804646e-03 4.416766e-05 1.871384e-01 2.208383e-03
##   q 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   r 1.470609e-03 1.015617e-02 3.005427e-02 2.462406e-01 3.093111e-03
##   s 2.705404e-03 1.417632e-02 4.010363e-04 1.117618e-01 1.909697e-03
##   t 6.788774e-05 2.869389e-03 1.357755e-05 8.663381e-02 8.327563e-04
##   u 1.404997e-02 3.170882e-02 2.180753e-02 2.941086e-02 5.120296e-03
##   v 0.000000e+00 0.000000e+00 0.000000e+00 5.523993e-01 0.000000e+00
##   w 1.028472e-04 2.742591e-04 6.290817e-03 1.329642e-01 7.542124e-04
##   x 0.000000e+00 1.419797e-01 0.000000e+00 7.417907e-02 0.000000e+00
##   y 3.423390e-03 3.862879e-03 3.007032e-04 5.977054e-02 2.266839e-03
##   z 0.000000e+00 0.000000e+00 1.140684e-02 3.096747e-01 0.000000e+00
##
##               g            h            i            j            k
##     1.720773e-02 8.970044e-02 5.370780e-02 2.856527e-03 7.045317e-03
##   , 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   : 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   . 2.616568e-04 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   a 1.724121e-02 1.538869e-03 4.297560e-02 8.233438e-04 1.260500e-02
##   b 0.000000e+00 2.887336e-05 2.947970e-02 5.052838e-03 0.000000e+00
##   c 0.000000e+00 1.886173e-01 4.183189e-02 0.000000e+00 4.262823e-02
##   d 3.872725e-03 4.451408e-04 6.952210e-02 2.101065e-03 6.142944e-04
##   e 7.085300e-03 2.346430e-03 1.139695e-02 2.793370e-04 9.004509e-04
##   f 0.000000e+00 5.640370e-05 8.783936e-02 0.000000e+00 1.316086e-04
##   g 9.001537e-03 1.204790e-01 5.055021e-02 0.000000e+00 0.000000e+00
##   h 6.024387e-06 3.614632e-05 1.565618e-01 0.000000e+00 5.723167e-04
##   i 2.481923e-02 2.889990e-05 2.658790e-03 0.000000e+00 7.554433e-03
##   j 0.000000e+00 0.000000e+00 4.280156e-03 0.000000e+00 0.000000e+00
##   k 6.000000e-04 4.085000e-02 1.700000e-01 0.000000e+00 0.000000e+00
##   l 9.034373e-04 4.202034e-05 9.991386e-02 0.000000e+00 1.088327e-02
##   m 1.649920e-05 0.000000e+00 8.355195e-02 0.000000e+00 9.899520e-05
##   n 1.396425e-01 7.990746e-04 3.461725e-02 6.705521e-04 8.337198e-03
##   o 4.050367e-03 1.888772e-03 1.176810e-02 6.348374e-04 1.789612e-02
##   p 1.987545e-04 9.164790e-03 9.279625e-02 0.000000e+00 3.975089e-04
##   q 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   r 9.638355e-03 1.456800e-03 9.215123e-02 1.380853e-05 6.365733e-03
##   s 2.800889e-04 6.901008e-02 5.951252e-02 1.909697e-05 1.371162e-02
##   t 4.073264e-05 3.338040e-01 6.337999e-02 0.000000e+00 4.525849e-05
##   u 5.058606e-02 7.248612e-04 2.698951e-02 1.542258e-05 3.871067e-03
##   v 1.136235e-04 0.000000e+00 1.841457e-01 0.000000e+00 3.029959e-04
##   w 0.000000e+00 2.044087e-01 1.717719e-01 0.000000e+00 1.079895e-03
##   x 0.000000e+00 1.393811e-02 1.211906e-01 0.000000e+00 0.000000e+00
##   y 2.081791e-04 2.775722e-04 2.579108e-02 0.000000e+00 8.789785e-04
##   z 0.000000e+00 6.506126e-02 1.035065e-01 0.000000e+00 1.689903e-03
##
##               l            m            n            o            p
##     2.350267e-02 3.458200e-02 2.599479e-02 6.197175e-02 3.188613e-02
##   , 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   : 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   . 5.233136e-05 0.000000e+00 1.569941e-04 7.849704e-04 5.233136e-05
##   a 6.974898e-02 2.267626e-02 2.240524e-01 2.058359e-04 2.273017e-02
##   b 1.049547e-01 1.992262e-03 2.598603e-04 1.183808e-01 0.000000e+00
##   c 3.256842e-02 3.250341e-05 0.000000e+00 2.085744e-01 0.000000e+00
```

```
##   d 1.102169e-02 2.795484e-03 2.581817e-03 4.698907e-02 3.561127e-05
##   e 3.406596e-02 2.347745e-02 8.730430e-02 5.320547e-03 1.126221e-02
##   f 2.273069e-02 3.760247e-05 9.400617e-05 1.521208e-01 0.000000e+00
##   g 2.993770e-02 3.641071e-04 1.873129e-02 6.290962e-02 0.000000e+00
##   h 9.036580e-04 2.271194e-03 6.988289e-04 8.267868e-02 2.409755e-05
##   i 4.784667e-02 5.685766e-02 2.799071e-01 4.298571e-02 4.820503e-03
##   j 0.000000e+00 0.000000e+00 0.000000e+00 2.785992e-01 0.000000e+00
##   k 1.695000e-02 1.450000e-03 1.035500e-01 2.990000e-02 5.000000e-05
##   l 1.387407e-01 4.117993e-03 1.281620e-03 8.770695e-02 3.991932e-03
##   m 2.672870e-03 2.280189e-02 3.745318e-03 1.108746e-01 5.659226e-02
##   n 1.175701e-02 7.711350e-04 1.089088e-02 7.085501e-02 5.308538e-04
##   o 4.070304e-02 5.806401e-02 1.454722e-01 3.775446e-02 1.601784e-02
##   p 9.321585e-02 5.962634e-04 1.766706e-04 1.051853e-01 5.797005e-02
##   q 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   r 8.692470e-03 2.020878e-02 1.767492e-02 1.019760e-01 3.755920e-03
##   s 7.645153e-03 1.073886e-02 2.794523e-03 5.729727e-02 2.061836e-02
##   t 1.356850e-02 1.240083e-03 8.056012e-04 1.119967e-01 1.900857e-04
##   u 9.691548e-02 2.108267e-02 1.354102e-01 2.282542e-03 4.282850e-02
##   v 8.143014e-03 0.000000e+00 2.014923e-02 5.673598e-02 0.000000e+00
##   w 3.411097e-03 3.428238e-05 3.808773e-02 7.878092e-02 1.714119e-05
##   x 4.724781e-04 0.000000e+00 0.000000e+00 1.181195e-03 3.201039e-01
##   y 4.672465e-03 3.862879e-03 1.087158e-03 1.383003e-01 1.295337e-03
##   z 1.816646e-02 2.661597e-02 7.182087e-03 2.674271e-01 0.000000e+00
##
##                q           r           s           t           u
##     2.478400e-03 2.651398e-02 7.219294e-02 1.570149e-01 9.929273e-03
##   , 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   : 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   . 0.000000e+00 0.000000e+00 2.616568e-04 5.233136e-05 5.233136e-05
##   a 9.801711e-06 8.799486e-02 1.022613e-01 1.380130e-01 1.233545e-02
##   b 0.000000e+00 5.711151e-02 1.374372e-02 7.420454e-03 1.524225e-01
##   c 1.901450e-03 3.365728e-02 1.560164e-03 6.716830e-02 2.748164e-02
##   d 4.006268e-04 3.184538e-02 1.981767e-02 4.184324e-04 9.819807e-03
##   e 8.380109e-04 1.451961e-01 6.450712e-02 2.530464e-02 1.031904e-03
##   f 0.000000e+00 1.064150e-01 2.801384e-03 4.042265e-02 3.500790e-02
##   g 0.000000e+00 5.546565e-02 1.743669e-02 3.438790e-03 2.710575e-02
##   h 0.000000e+00 7.723264e-03 9.398043e-04 2.527230e-02 9.639019e-03
##   i 1.560594e-04 3.424638e-02 1.265295e-01 1.240210e-01 7.513973e-04
##   j 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 4.603113e-01
##   k 0.000000e+00 4.750000e-03 3.735000e-02 2.500000e-04 3.385000e-02
##   l 1.050508e-05 4.863854e-03 1.633541e-02 1.723884e-02 1.592571e-02
##   m 0.000000e+00 1.336435e-03 3.164547e-02 9.734528e-04 2.758666e-02
##   n 1.352280e-03 4.135072e-04 3.449991e-02 8.937901e-02 5.353241e-03
##   o 1.311647e-04 1.038877e-01 3.541973e-02 5.254984e-02 1.235519e-01
##   p 0.000000e+00 1.734685e-01 2.369595e-02 4.785566e-02 2.577183e-02
##   q 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 9.995708e-01
##   r 8.975545e-05 3.401041e-02 5.574504e-02 3.175962e-02 2.098897e-02
##   s 1.075796e-03 1.591414e-04 6.165138e-02 1.130795e-01 2.516980e-02
##   t 0.000000e+00 2.558010e-02 1.765986e-02 1.985038e-02 1.638810e-02
##   u 1.542258e-05 1.240130e-01 1.376774e-01 1.697872e-01 0.000000e+00
##   v 0.000000e+00 3.295080e-03 9.885240e-03 7.574897e-05 8.332386e-04
##   w 0.000000e+00 9.050549e-03 1.247879e-02 1.199883e-04 4.970946e-04
##   x 9.449563e-04 0.000000e+00 0.000000e+00 1.112686e-01 4.016064e-03
##   y 0.000000e+00 1.919874e-03 3.034789e-02 1.838916e-02 1.387861e-03
```

```
##   z 0.000000e+00 0.000000e+00 8.449514e-04 0.000000e+00 5.534432e-02
##
##               v            w            x            y            z
##     7.053153e-03 7.116436e-02 8.796850e-04 1.109500e-02 2.488196e-04
##   , 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   : 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   . 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 5.233136e-05
##   a 2.208816e-02 1.055154e-02 2.499436e-04 2.657244e-02 1.877028e-03
##   b 1.328175e-03 5.774672e-04 0.000000e+00 7.362707e-02 0.000000e+00
##   c 0.000000e+00 1.137619e-04 0.000000e+00 5.038029e-03 8.125853e-05
##   d 2.848901e-03 3.917239e-04 0.000000e+00 1.050532e-02 1.780563e-05
##   e 1.801230e-02 1.048007e-02 1.029932e-02 1.237956e-02 6.211139e-04
##   f 0.000000e+00 5.076333e-04 0.000000e+00 1.598105e-03 0.000000e+00
##   g 0.000000e+00 3.641071e-04 0.000000e+00 1.921677e-03 4.045635e-05
##   h 2.409755e-05 2.168779e-04 0.000000e+00 5.379777e-03 0.000000e+00
##   i 2.008543e-02 2.311992e-05 1.872713e-03 5.779979e-06 3.427528e-03
##   j 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   k 6.000000e-04 4.100000e-03 0.000000e+00 5.000000e-03 0.000000e+00
##   l 4.443651e-03 4.905874e-03 0.000000e+00 1.051349e-01 2.626271e-04
##   m 0.000000e+00 9.899520e-05 0.000000e+00 4.052203e-02 0.000000e+00
##   n 4.207715e-03 5.923211e-04 4.582106e-04 1.160614e-02 1.229346e-04
##   o 3.616474e-02 5.205142e-02 7.397692e-04 3.877230e-03 6.033578e-04
##   p 0.000000e+00 4.195928e-04 0.000000e+00 7.309748e-03 0.000000e+00
##   q 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   r 4.867507e-03 1.912482e-03 0.000000e+00 3.967881e-02 5.661498e-04
##   s 5.028868e-04 4.258624e-03 0.000000e+00 2.368024e-03 2.546262e-05
##   t 7.693944e-05 5.263563e-03 4.525849e-06 1.418401e-02 6.200414e-04
##   u 1.233806e-03 3.084516e-05 6.631709e-04 3.238742e-04 9.454041e-03
##   v 3.787448e-05 3.787448e-05 0.000000e+00 4.090444e-03 0.000000e+00
##   w 0.000000e+00 2.914003e-04 0.000000e+00 6.685065e-04 0.000000e+00
##   x 3.685330e-02 0.000000e+00 3.732577e-02 9.449563e-04 0.000000e+00
##   y 2.313101e-05 1.850481e-03 0.000000e+00 0.000000e+00 2.313101e-04
##   z 0.000000e+00 4.224757e-04 0.000000e+00 5.914660e-03 1.985636e-02
```

```
heatmap(Trans, Colv=NA, Rowv=NA)
```



Largest and smallest entropy? How to get?

5. The likelihood function can be calculated as follows,

```
# initialize a permutation function
wop <- wopList_rev[!wopList_rev %in% '']
wop.table.freq <- table(wop)
symbols <- names(wop.table.freq)
perm <- symbols
names(perm) <- symbols
# define the decode function
decode <- function(perm, encoded.msg){
  perm.inv <- names(perm)
  names(perm.inv) <- as.vector(perm)
  decoded.msg <- sapply(encoded.msg, function(x) perm.inv[x],USE.NAMES = FALSE)
  return(unname(decoded.msg))
}
#wop[1:11]
# helper function slog
slog <- function(x) return(ifelse(x==0,0,log(x)))
# calculate the log likelihood
cal_log_likelihood <- function(transition, decoded.msg){
  len.decoded.msg <- length(decoded.msg)
  probs <- mapply(function(x,y) return(slog(transition[x,y])), decoded.msg[1:len.decoded.msg-1], decode
  return(sum(probs))
}
#decoded.msg <- decode(perm,wop[1:11])
#decoded.msg
#cal_log_likelihood(Trans, decoded.msg)
```

6. Directly sampling from the posterior probability (or likelihood when prior is uniform) is not easy since the distribution is expressed in a matrix form and the the smapling variable is a functional mapping σ. There is no direct way to sample a mapping σ from a matrix-form PMF (probability mass function).

7. The proposal distirbution is given as a uniform distribution, which demands the sampling from it is to randomly swapping the mappling of two pairs of symbols in σ. The proposal distirbution is then given by

$$q(\sigma^*|\sigma^{(n)}) = \frac{1}{30 \times 29} = \frac{1}{870}$$

The stationary distribution of the Markov chain should be the posterior or the likelihood when prior is uniform. Hence, the acceptance probability is given as

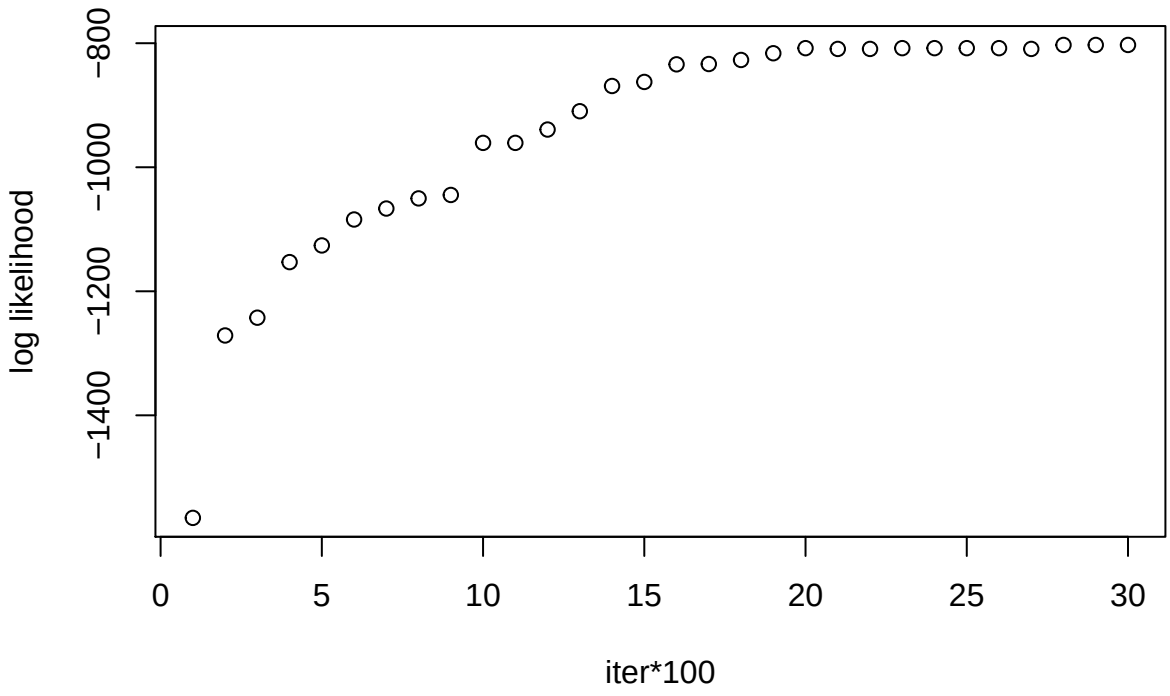
$$\alpha = \frac{P(X|\sigma^*)q(\sigma^{(n)}|\sigma^*)}{P(X|\sigma^{(n)})q(\sigma^*|\sigma^{(n)})} = \frac{P(X|\sigma^*)}{P(X|\sigma^{(n)})}$$

Since the proposal is symmetric. In the acceptance probabiliy, $\sigma^{(n)}$ is the state of σ at the n-th step of the Markov chain, σ^* is the new updated state.

8. The Matrapolis Hasting Algorithm is implemented as follows,

```
MH <- function(transition, perm, encoded.msg, max.iter=3000){
  set.seed(12)
  len.perm <- length(perm)

  old.perm <- perm
  old.decoded.msg <- decode(old.perm, encoded.msg)
  old.loglikelihood <- cal_log_likelihood(transition, old.decoded.msg)
  list.loglikelihood <- c()
  list.decoded.msg <- c()

  iter <- 0
  while (iter < max.iter) {
    # sampling according to the proposal distribution
    idx.swap <- sample(1:len.perm, 2)
    perm <- old.perm
    perm[idx.swap[1]] <- old.perm[idx.swap[2]]
    perm[idx.swap[2]] <- old.perm[idx.swap[1]]
    decoded.msg <- decode(perm, encoded.msg)
    loglikelihood <- cal_log_likelihood(transition, decoded.msg)

    # accept or reject the updated sigma
    if (runif(1) < exp(loglikelihood - old.loglikelihood)){
      old.perm <- perm
      old.decoded.msg <- decoded.msg
      old.loglikelihood <- loglikelihood
    }

    if ((iter + 1) %% 100 == 0){
      list.loglikelihood <- c(list.loglikelihood, old.loglikelihood)
      list.decoded.msg <- c(list.decoded.msg, paste(old.decoded.msg[1:20],collapse = ''))
    }
    iter <- iter + 1
  }
  return(list(list.loglikelihood = list.loglikelihood, list.decoded.msg = list.decoded.msg, perm = perm
}

filename.msg <- 'message.txt'
msg <- readChar(filename.msg, file.info(filename.msg)$size)
msg <- sub('\n','',msg)
```

```
msg <- strsplit(msg,'')[[1]]
rslt <- MH(Trans, perm, msg)
plot(1:length(rslt$list.loglikelihood), rslt$list.loglikelihood, xlab = "iter*100", ylab = "log likelih
```




```
msg[1:20]
```

```
##  [1] "y" "e" "," "o" ":" "t" "o" "r" "m" "h" "x" "w" "e" "o" "v" ":" "d"
## [18] "o" "," "d"
```

```
rslt$list.decoded.msg
```

```
##  [1] "sq,g ygrpx:mqgv cg,c" "wx,gongrpj:mxgvocg,c" "wx,gongmpj:ixgvocg,c"
##  [4] "yx,gengpmjzixgvecg,c" "yx,gengpmjzixgvecg,c" "yx,gengpljzrxgvecg,c"
##  [7] ".x,gengpljzrxgvecg,c" ".x,gengpljzrxgqecg,c" ".xcgengpkjzrxgqe,gc,"
## [10] ".xcgengpfjzrxgqe,gc," ".xcgengpfjzrxgqe,gc," ".xfgebgpcjzrxgqe,gf,"
## [13] ".xfgebgpcjzrxgqe,gf," ".xfgozgpcjbrxgqo,gf," ".xfgozgpcjbexgqo,gf,"
## [16] ".xfgocgpzjbexgqo,gf," ".xfgocgpzjbexgqo,gf," ".xfgocgpzjbexgqo,gf,"
## [19] ".xfgocgpzjb:xgqo,gf," ".xfgocgpzjb:xgqo,gf," ".xfgocgpzjb:xgqo,gf,"
## [22] ".xfgocgpzjb:xgqo,gf," ".xfgocgpzjb:xgqo,gf," ".xfgocgpzjb:xgqo,gf,"
## [25] ".xfgocgpzjb:xgqo,gf," ".xfgocgpzjb:xgqo,gf," ".xfgocgpzjb:xgqo,gf,"
## [28] ".xfgocgkzjb:xgqo,gf," ".xfgocgkzjb:xgqo,gf," ".xfgocgkzjb:xgqo,gf,"
```

```
rslt$perm
```

```
##       ,   :   .   a   b   c   d   e   f   g   h   i   j   k   l   m   n
## " " "d" "w" "y" "a" "x" "t" "." "b" "," "o" "f" "p" "h" "r" "z" "g" "s"
##   o   p   q   r   s   t   u   v   w   x   y   z
## ":" "q" "v" "l" "c" "u" "j" "i" "k" "e" "n" "m"
```