Code ▾

Project

Arshia Singh

April 1, 2020

Hide

```
library(reshape2)
library(ggplot2)
library(Rfast)
library(tm)
```

Hide

```
reference_text=readLines("warandpeace.txt")
reference_text=toupper(reference_text)

transition_matrix=matrix(0,27,27)
rownames(transition_matrix)=colnames(transition_matrix)=c(toupper(letters),"")
last_letter=""
for (line in 1:length(reference_text)) {
  if (line %% 1000 ==0) {cat("Line",line,"\n")}
  for (position in 1:nchar(reference_text[line])) {
    current_letter=substring(reference_text[line],position,position)
    if (current_letter %in% toupper(letters)) {
      transition_matrix[rownames(transition_matrix)==last_letter,
                colnames(transition_matrix)==current_letter]=
        transition_matrix[rownames(transition_matrix)==last_letter,
                  colnames(transition_matrix)==current_letter]+1
      last_letter=current_letter
    } else {
      if (last_letter!="") {
        transition_matrix[rownames(transition_matrix)==last_letter,27]=
          transition_matrix[rownames(transition_matrix)==last_letter,27]+1
        last_letter=""
      }
    }
  }
  current_letter=""
  if (last_letter!="") {
    transition_matrix[rownames(transition_matrix)==last_letter,27]=
      transition_matrix[rownames(transition_matrix)==last_letter,27]+1
  }
  last_letter=""
}

transition_prob_matrix=sweep(transition_matrix+1,1,rowSums(transition_matrix+1),FUN
="/")
rm(reference_text)

ggplot(melt(transition_prob_matrix),aes(Var2,Var1))+geom_tile(aes(fill=value))+
  scale_fill_gradient(low="white",high="blue",limits=c(0,1))+
  labs(x="Prob of Second Letter",y="Conditioned on First Letter",fill="Prob")+
  scale_y_discrete(limits = rev(levels(melt(transition_prob_matrix)$Var1)))+
  coord_equal()
```

Hide

```
decode <- function(map,coded) {
  coded=toupper(coded)
  decoded=coded
  for (i in 1:nchar(coded)) {
    if (substring(coded,i,i) %in% toupper(letters)) {
      substring(decoded,i,i)=toupper(letters[map==substring(coded,i,i)])
    }
  }
  decoded
}


log_prob <- function(map,decoded) {
  logprob=0

  last_letter=""
  for (i in 1:nchar(decoded)) {
    current_letter=substring(decoded,i,i)
    if (current_letter %in% toupper(letters)) {
      logprob=logprob+log(transition_prob_matrix[rownames(transition_matrix)==last_let
ter,
                                         colnames(transition_matrix)==current_letter])
      last_letter=current_letter
    } else {
      if (last_letter!="") {
        logprob=logprob+log(transition_prob_matrix[rownames(transition_matrix)==last_l
etter,27])
        last_letter=""
      }
    }
  }

  if (last_letter!="") {
    logprob=logprob+log(transition_prob_matrix[rownames(transition_matrix)==last_lette
r,27])
    last_letter=""
  }
  logprob
}
```

Hide

```
correctTxt="The marshes were just a long black horizontal line then, as I stopped to look after him; and the river was just another horizontal line, not nearly so broad nor yet so black; and the sky was just a row of long angry red lines and dense black lines intermixed. On the edge of the river I could faintly make out the only two black things in all the prospect that seemed to be standing upright; one of these was the beacon by which the sailors steered,like an unhooped cask upon a pole,an ugly thing when you were near it; the other, a gibbet, with some chains hanging to it which had once held a pirate. The man was limping on towards this latter, as if he were the pirate come to life, and come down, and going back to hook himself up again. It gave me a terrible turn when I thought so; and as I saw the cattle lifting their heads to gaze after him, I wondered whether they thought so too. I looked all round for the horrible young man, and could see no signs of him. But now I was frightened again, and ran home without stopping."
correctTxt=toupper(removePunctuation(correctTxt))
coded=decode(sample(toupper(letters)),correctTxt)

print(correctTxt)
print(coded)
```

Hide

```
mcmc_starttime=Sys.time()

map=sample(toupper(letters))
i=1
iters=300
mcmc_times=numeric(iters)
current_decode=decode(map,coded)
current_loglikelihood=log_prob(map,current_decode)
max_loglikelihood=current_loglikelihood
max_decode=current_decode
while (i<=iters) {
  proposal=sample(1:26,2)
  prop_map=map
  prop_map[proposal[1]]=map[proposal[2]]
  prop_map[proposal[2]]=map[proposal[1]]

  proposed_decode=decode(prop_map,coded)
  proposed_loglikelihood=log_prob(prop_map,proposed_decode)

  if (runif(1)<exp(proposed_loglikelihood-current_loglikelihood)) {
    map=prop_map
    current_decode=proposed_decode
    current_loglikelihood=proposed_loglikelihood

    if (current_loglikelihood>max_loglikelihood) {
      max_loglikelihood=current_loglikelihood
      max_decode=current_decode
    }

    cat(i,current_decode,"\n")
    mcmc_times[i]=difftime(Sys.time(), mcmc_starttime, units = "secs")
    i=i+1
  }
}
```

Hide

```
cat("Total time of MCMC run:", mcmc_times[79], "\n")
cat("Average step length:", mean(diff(mcmc_times[1:79])), "\n")
```

Hide

```
emcmc_starttime=Sys.time()

Ncount=3
Lcount=10
map=matrix(replicate(Ncount, sample(toupper(letters))), ncol=26, nrow=Ncount, byrow=TRUE) # initial n maps
i=1
iters=300
emcmc_times=numeric(iters)
current_decode=vector(mode='character',length=Ncount)
current_loglikelihood=numeric(Ncount)
pop_map=matrix(NA, ncol=26, nrow=Ncount*Lcount)
pop_decode=vector(mode='character',length=Ncount*Lcount)
pop_loglike=numeric(Ncount*Lcount)
L=1
for (N in 1:Ncount){
  current_decode[N]=decode(map[N,],coded)
  current_loglikelihood[N]=log_prob(map[N,],current_decode[N])
  for (v in 1:Lcount) {
    pop_map[L,]=map[N,]
    pop_decode[L]=current_decode[N]
    pop_loglike[L]=current_loglikelihood[N]
    L=L+1
  }
}

while (i<=iters) {
  proposal=matrix(replicate(Ncount*Lcount, sample(1:26,2)), nrow=Ncount*Lcount, byrow=TRUE) # propose swap
  L=1
  for (N in 1:Ncount) {
    for (v in 1:Lcount) {
      propnow <- proposal[L,]
      prop_map=map[N,]
      prop_map[propnow[1]]=map[N,propnow[2]]
      prop_map[propnow[2]]=map[N,propnow[1]]
      proposed_decode=decode(prop_map,coded)
      proposed_loglikelihood=log_prob(prop_map,proposed_decode)


  if (runif(1)<exp(proposed_loglikelihood-current_loglikelihood[N])) {
    pop_map[L,]=prop_map
    pop_decode[L]=proposed_decode
    pop_loglike[L]=proposed_loglikelihood
  }
      L=L+1
    }
  }
  survivors=which(pop_loglike %in% sort(pop_loglike,T)[1:N])
  map=pop_map[survivors,]
  current_decode=pop_decode[survivors]
  current_loglikelihood=pop_loglike[survivors]
  max_loglikelihood=current_loglikelihood[which.max(current_loglikelihood)]
  max_decode=current_decode[which.max(current_loglikelihood)]

    cat(i,max_decode,"\n")
    emcmc_times[i]=difftime(Sys.time(), emcmc_starttime, units = "secs")
    i=i+1
}
```

Hide

```
cat("Total time of eMCMC run:", emcmc_times[108], "\n")
cat("Average step length:", mean(diff(emcmc_times[1:108])), "\n")
```