

ECS256 - Homework III

Olga Prilepova, Christopher Patton, Alexander Rumbaugh,
John Chen, Thomas Provan

February 17, 2014

Problem 1.a

First, we'll derive π_i . The definition of the tree searching markov model leads to the following set of balance equations for the long-run state probabilities:

$$\pi_i = \pi_{i-1}q_{i-1} = \pi_0 \prod_{j=0}^{i-1} q_j \quad \text{for } i \geq 1, \text{ and}$$

$$\pi_0 = \sum_{i=1}^{\infty} \pi_i(1 - q_i) \quad \text{for } i = 0.$$

This definition for π_0 is a bit unwieldy. We can also think of this quantity as one over the expected recurrence time, as in eq. (10.63) in the book:

$$\begin{aligned} \pi_0 &= \frac{1}{E(T_{0,0})} \\ E(T_{0,0}) &= 1 + \sum_{k \neq 0} p_{0,k} E(T_{k,0}) \\ &= 1 + p_{0,1} E(T_{1,0}) \\ &= 1 + p_{0,1} (1 + \sum_{k \neq 0} p_{1,k} E(T_{k,0})) \\ &= 1 + p_{0,1} (1 + p_{1,2} E(T_{2,0})) \\ &= 1 + p_{0,1} (1 + p_{1,2} (1 + \sum_{k \neq 0} p_{2,k} E(T_{k,0}))) \\ &= 1 + p_{0,1} (1 + p_{1,2} (1 + p_{2,3} E(T_{3,0}))) \end{aligned}$$

and so on. This unravels into a familiar closed form:

$$\begin{aligned} E(T_{0,0}) &= 1 + q_0(1 + q_1(1 + q_2(1 + \dots) \dots)) \\ &= 1 + q_0 + q_0q_1 + q_0q_1q_2 + \dots \\ &= 1 + \sum_{i=1}^{\infty} \left[\prod_{j=0}^{i-1} q_j \right] \end{aligned}$$

If the model is positive recurrent, then there exists some value R such that

$$R = \sum_{i=1}^{\infty} \left[\prod_{j=0}^{i-1} q_j \right] < \infty.$$

Thus,

$$\pi_i = \frac{\prod_{j=0}^{i-1} q_j}{1 + R} \quad \text{for } i \geq 0.$$

Next, $E(T_{i,0})$ follows a similar pattern.

$$\begin{aligned} E(T_{i,0}) &= 1 + \sum_{k \neq 0} p_{i,k} E(T_{k,0}) \\ &= 1 + p_{i,i+1} E(T_{j+1,0}) \\ &= 1 + q_i + q_i q_{i+1} + q_i q_{i+1} q_{i+2} + \dots \\ &= 1 + \sum_{j=i}^{\infty} \left[\prod_{k=i}^j q_k \right]. \end{aligned}$$

Problem 1.b

If $q_i = 0.5$ for all i , then R is a geometric series that indeed converges.

$$\pi_2 = \frac{0.5 \cdot 0.5}{1 + \sum_{i=1}^{\infty} 0.5^{i-1}} = \frac{0.25}{1 + 2} \approx 0.083.$$

$$E(T_{2,0}) = 1 + \sum_{j=2}^{\infty} 0.5^{j-2} = 1 + \sum_{j=1}^{\infty} 0.5^{j-1} = 1 + 2 = 3.$$

Problem 1.c

The rate of backtracking, in terms of the stationary probabilities π_i , is simply

$$\sum_{i=1}^{\infty} \pi_i (1 - q_i).$$

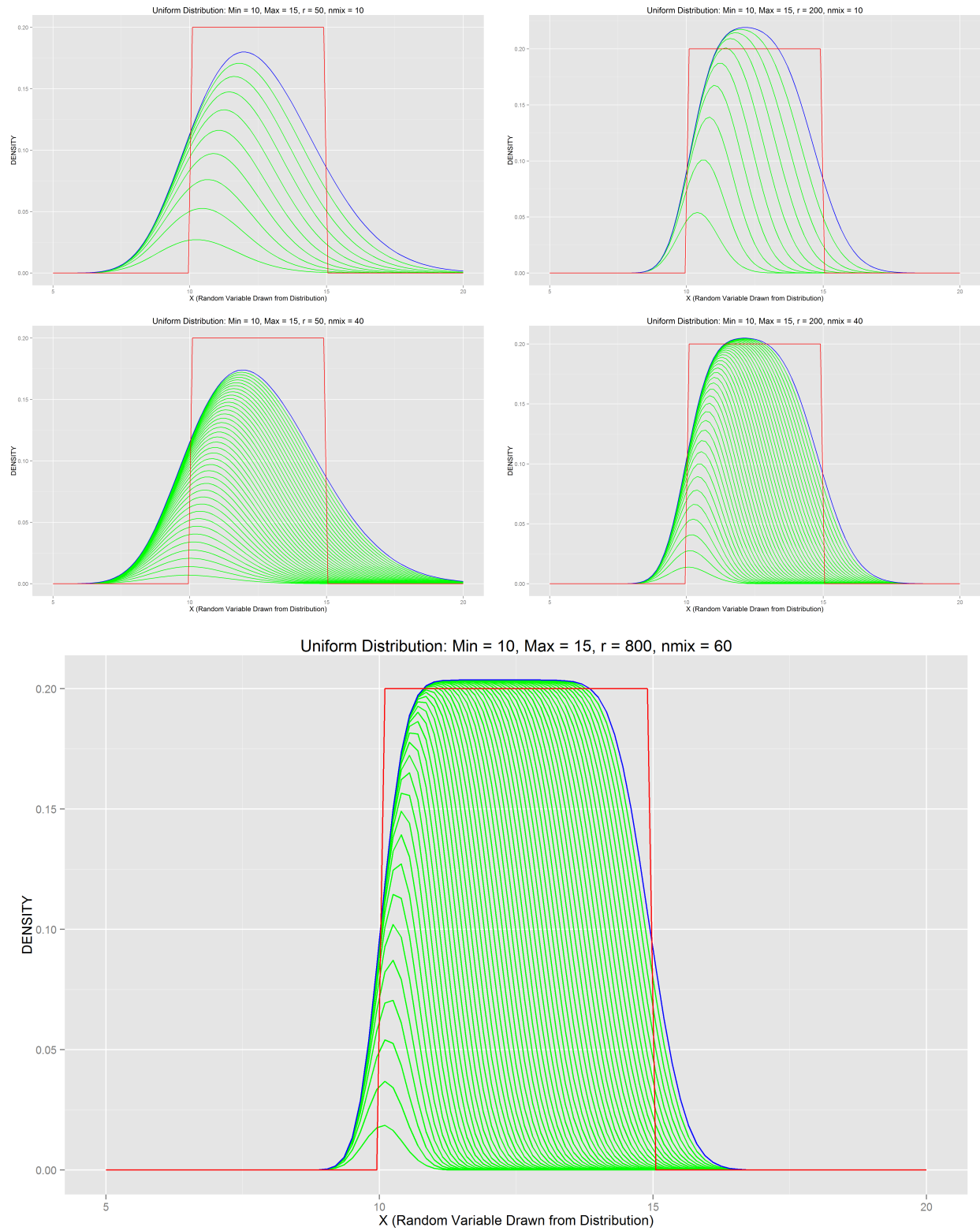
Problem 2.b

Using the lambdas generated by `erlangmix()`, we are able to generate a set of `nmix` erlang distributions with parameters given as: `Shape = R Rate = lambda`

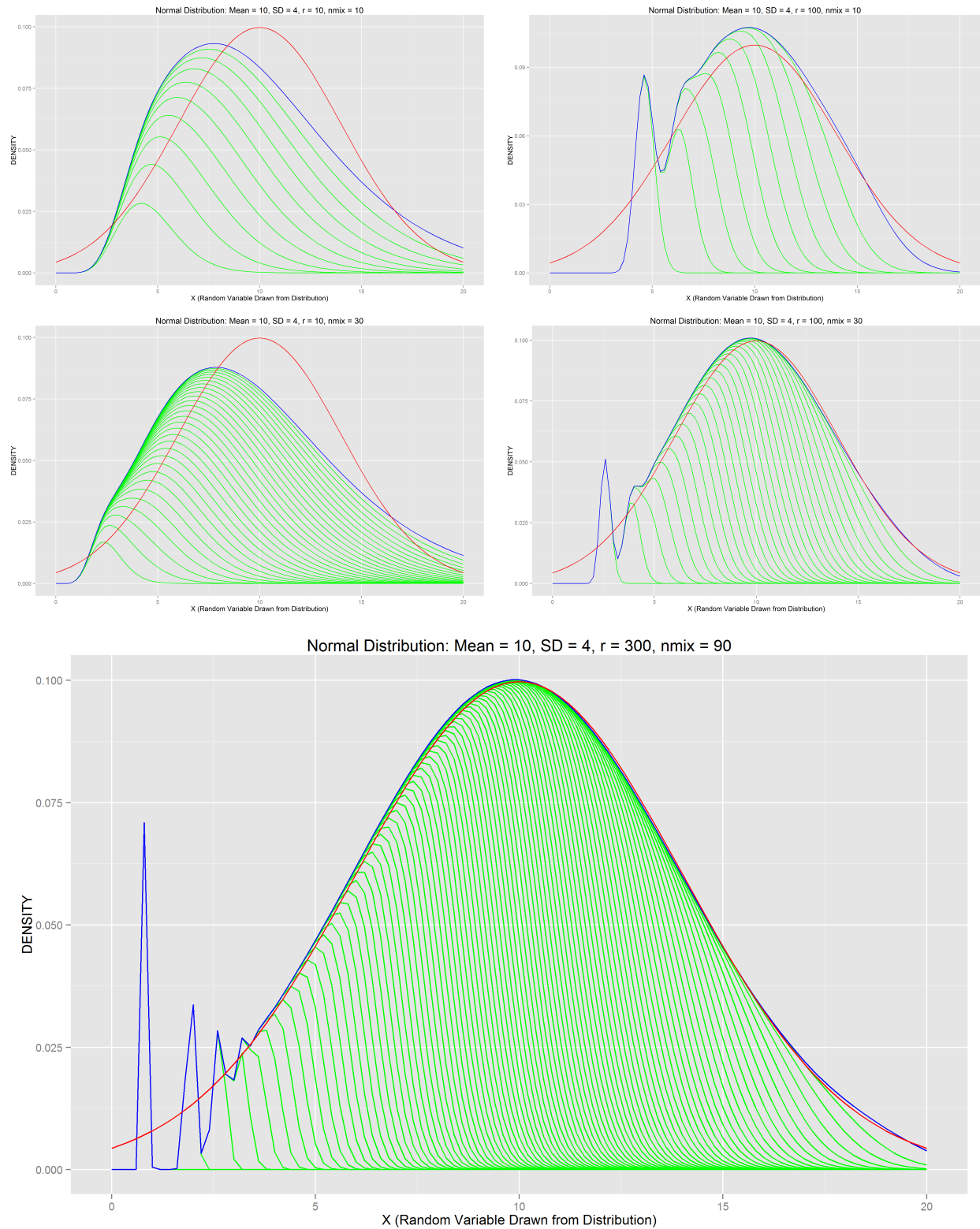
The combination of all `nmix` erlang distributions yields our method-of-stages approximation of the quantile function fed into `erlangmix()`.

Here, we explored the effect of different values of `r` and `nmix` on the approximation.

For a uniform distribution with minimum = 10, maximum = 15:



For a normal distribution with mean = 10, standard deviation = 4:



Problem 3.a

See HtoF.R.

```
1 htof = function(hftn,t,lower){
2   density_val = c()
3   for(val in t)
4     {
5       density_val = c(density_val, hftn(val) *
6         exp(-1*integrate(hftn,lower,val)$value))
7     }
8   return (density_val)
9 }
```

Problem 3.b

Given a hazard function, $h(t)$, the density function, $f(t)$, can be found as follows:

$$f(t) = h(t) \cdot e^{-\int_0^t h(s) ds}$$

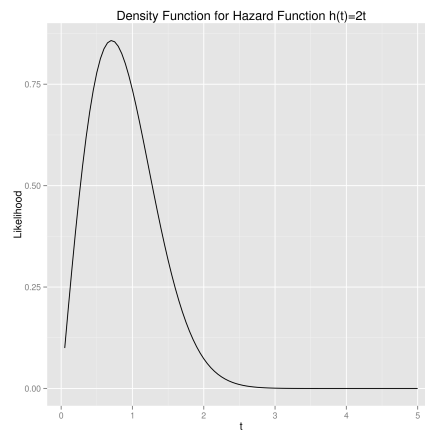
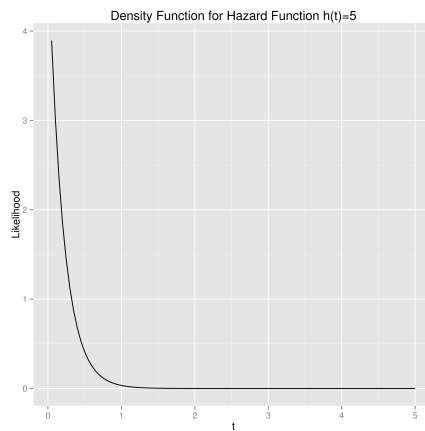
We looked at the following hazard functions to explore what their density would look like:

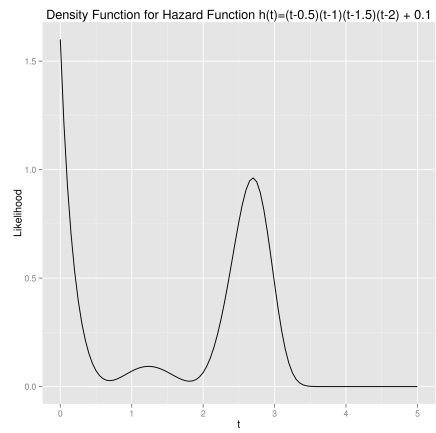
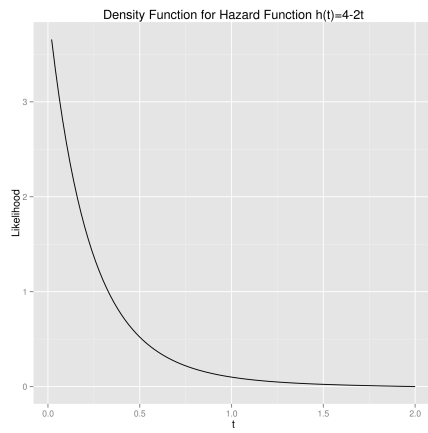
$$h(t) = 5$$

$$h(t) = 2t$$

$$h(t) = 4 - 2t$$

$$h(t) = (t - 0.5)(t - 1)(t - 1.5)(t - 2) + 0.1$$

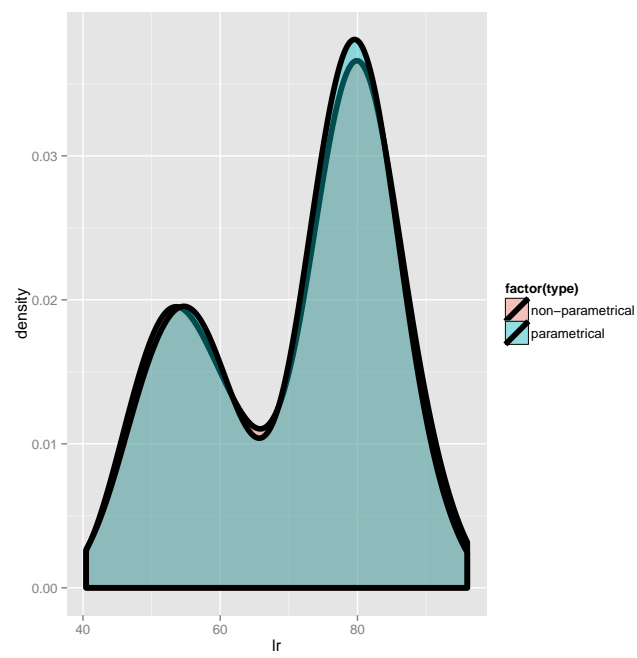




(Plots generated with 3.R).

Problem 4

4.a-b



Appendix

Problem 4

```
1 #install.packages("ggplot2")
2 #install.packages("mixtools")
3 library(mixtools)
4 library(ggplot2)
5
6 #4.a #####
7 p<-ggplot(data.frame(faithful))
8 p+geom_density(aes(x=faithful$waiting))
9
10 #4.b #####
11 simulateFromDist <- function(n,p1,m1,s1,m2,s2){
12     k1 <- p1*n #proportion of type 1
13     k2 <- n-k1+1 #proportion of type 2
14     x1 <- rnorm(k1, mean=m1, sd=s1)
15     x2 <- rnorm(k2, mean=m2, sd=s2)
16     c(x1,x2) #order of events doesn't matter for histogram, so simply concatenate
17 }
18
19 #####from mixtools simulation
20 mixout<-normalmixEM(faithful$waiting,lambda=0.5,mu=c(55,80),sigma=10,k=2)
21 str(mixout)
22 # $ lambda      : num [1:2] 0.361 0.639
23 # $ mu           : num [1:2] 54.6 80.1
24 # $ sigma        : num [1:2] 5.87 5.87
25
26
27 # Is it necessary to simulate this? Can we plot the function directly?
28 sim_waiting<-simulateFromDist(length(faithful$waiting),0.361,54.6,5.87,80.1,5.87)
29
30 data <- rbind( data.frame(type="non-parametrical", lr=faithful$waiting), data.frame(type="parametrical", lr=sim_waiting))
31 m <- ggplot(data, aes(x=lr))
32 m <- m + geom_density(aes(fill=factor(type)), size=2, alpha=.4)
33
34 #save m
35 pdf('plot4b.pdf')
36 m
37 dev.off()
38
39 #4.f #####
40
41 EL2 <- ( mixout$sigma[1]^2 * mixout$lambda[1] + mixout$sigma[2]^2 * mixout$lambda[2] ) + (mixout$mu[1] * mixout$lambda[1] + mixout$mu[2] * mixout$lambda[2])
42 EL <- mixout$mu[1] * mixout$lambda[1] + mixout$mu[2] * mixout$lambda[2]
```