MA677 Final Project

Yuli Jin

2022/4/25

4.25

```r
# reference
# https://stackoverflow.com/questions/24211595/order-statistics-in-r?msclkid=fd6683dac56711ecbfcea9bd8a


f <- function(x, a=0, b=1) dunif(x, a,b) #pdf function
F <- function(x, a=0, b=1) punif(x, a,b, lower.tail=FALSE) #cdf function

#distribution of the order statistics
integrand <- function(x,r,n) {
  x * (1 - F(x))^(r-1) * F(x)^(n-r) * f(x)
}

#get expectation
E <- function(r,n) {
  (1/beta(r,n-r+1)) * integrate(integrand,-Inf,Inf, r, n)$value
}

# approx function
medianprrox<-function(k,n){
  m<-(k-1/3)/(n+1/3)
  return(m)
}


E(2.5,5)
```

```
## [1] 0.4166667
```

```r
medianprrox(2.5,5)
```

```
## [1] 0.40625
```

```r
E(5,10)
```

```
## [1] 0.4545455
```

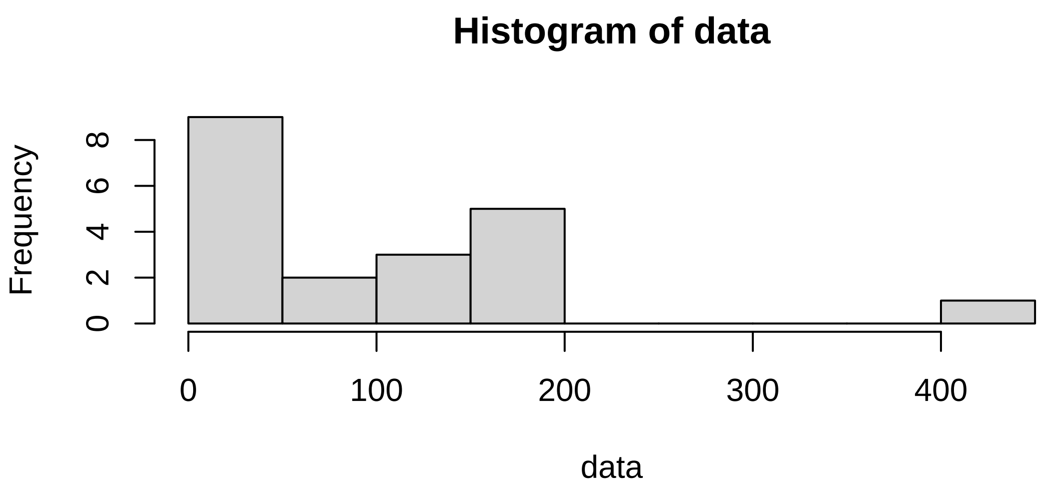
```r
medianprrox(5,10)
```

```
## [1] 0.4516129
```

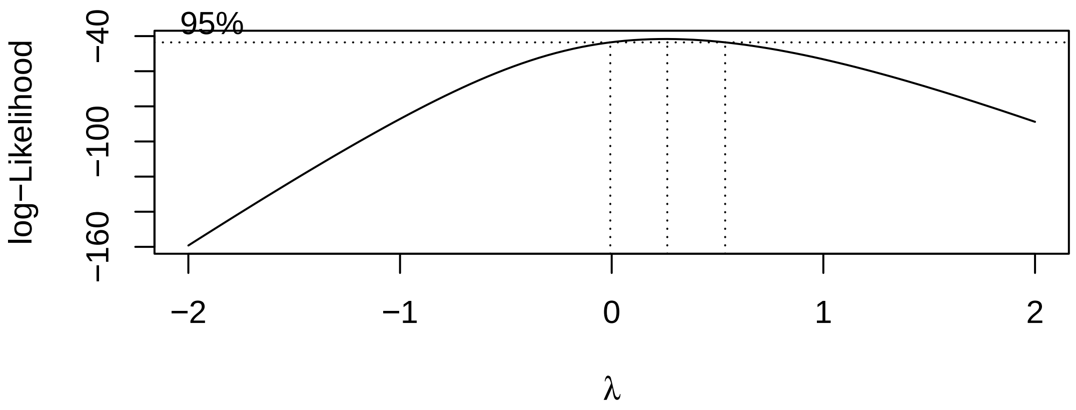
The result shows that they are similar.

4.39

```
data<-c(0.4,1.0,1.9,3.0,5.5,8.1,12.1,25.6,50.0,56.0,70.0,115.0,115.0,119.5,154.5,157.0,175.0,179.0,180.
hist(data)
```



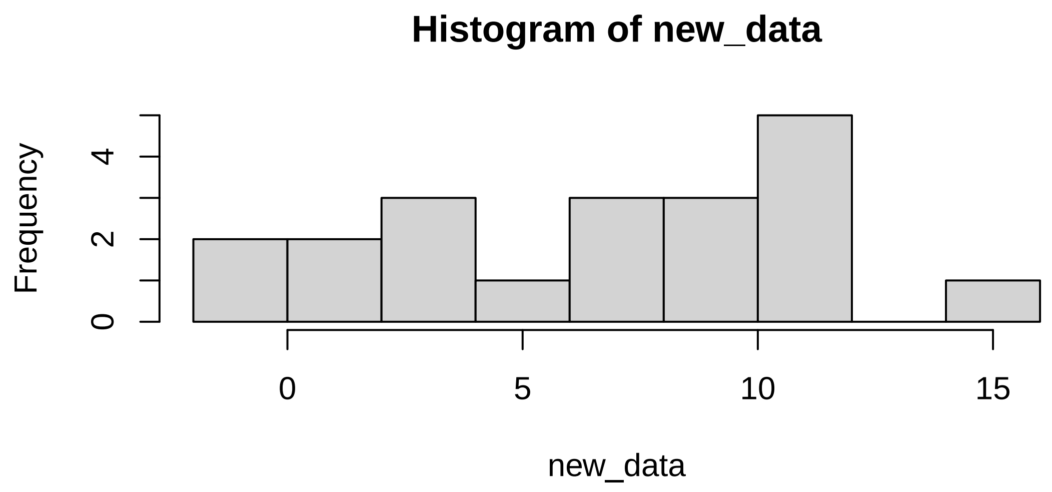
```
# Conduct boxcox transformation
b <- boxcox(lm(data ~ 1))
```



```
# Exact lambda
lambda <- b$x[which.max(b$y)]
lambda #lambda=0.2626263
```

```
## [1] 0.2626263
```

```
new_data <- (data ^ lambda - 1) / lambda
hist(new_data)
```



4.27

```
Jan<-c(0.15,0.25,0.10,0.20,1.85,1.97,0.80,0.20,0.10,0.50,0.82,0.40,1.80,0.20,1.12,1.83,
       0.45,3.17,0.89,0.31,0.59,0.10,0.10,0.90,0.10,0.25,0.10,0.90)
Jul<-c(0.30,0.22,0.10,0.12,0.20,0.10,0.10,0.10,0.10,0.10,0.10,0.17,0.20,2.80,0.85,0.10,
       0.10,1.23,0.45,0.30,0.20,1.20,0.10,0.15,0.10,0.20,0.10,0.20,0.35,0.62,0.20,1.22,
       0.30,0.80,0.15,1.53,0.10,0.20,0.30,0.40,0.23,0.20,0.10,0.10,0.60,0.20,0.50,0.15,
      0.60,0.30,0.80,1.10,
      0.2,0.1,0.1,0.1,0.42,0.85,1.6,0.1,0.25,0.1,0.2,0.1)
```

(a)

```
summary(Jan)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1000  0.1875  0.4250  0.7196  0.9000  3.1700
```

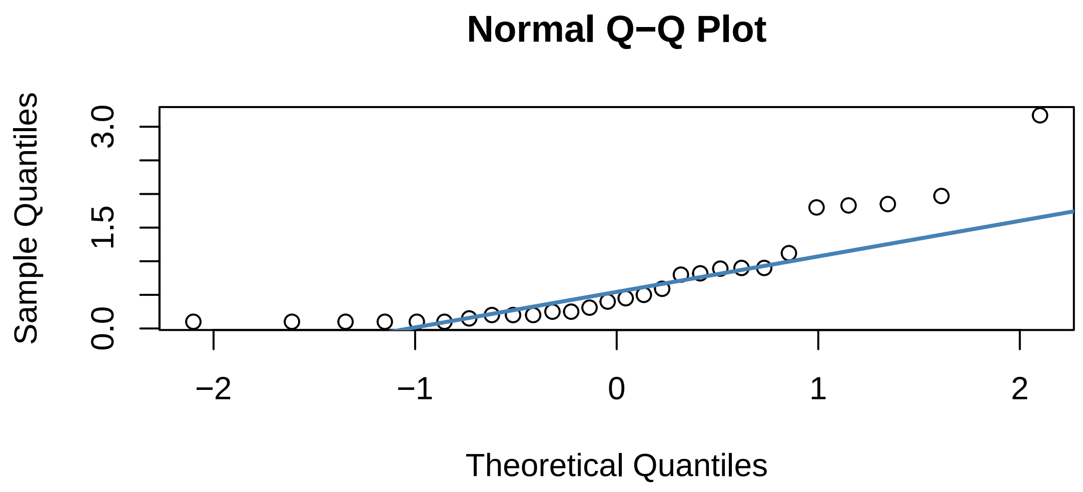
```
summary(Jul)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1000  0.1000  0.2000  0.3931  0.4275  2.8000
```

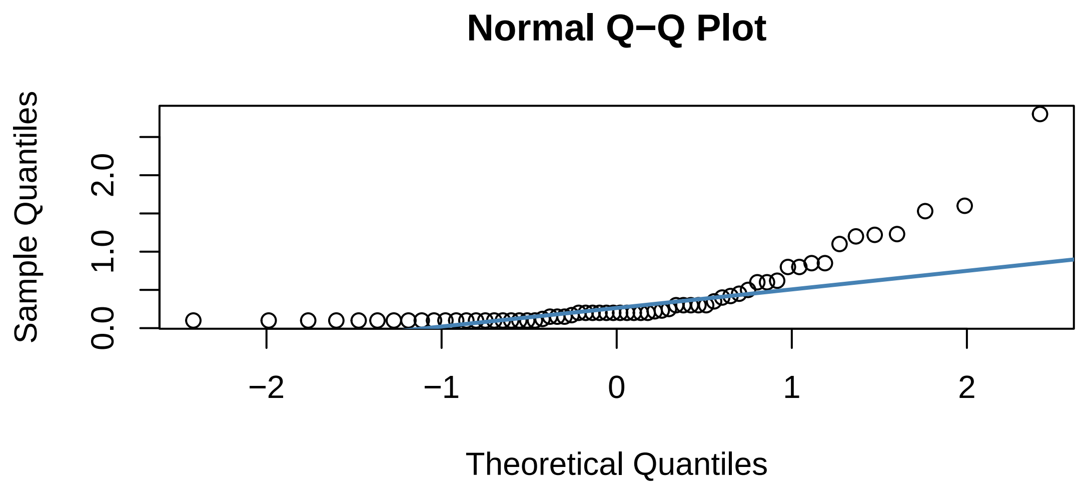
Jan's 1st, Median, Mean 3rd Max are higher than the one in Jul. Also, Jan's IQR is higher than the one in Jul.

(b)

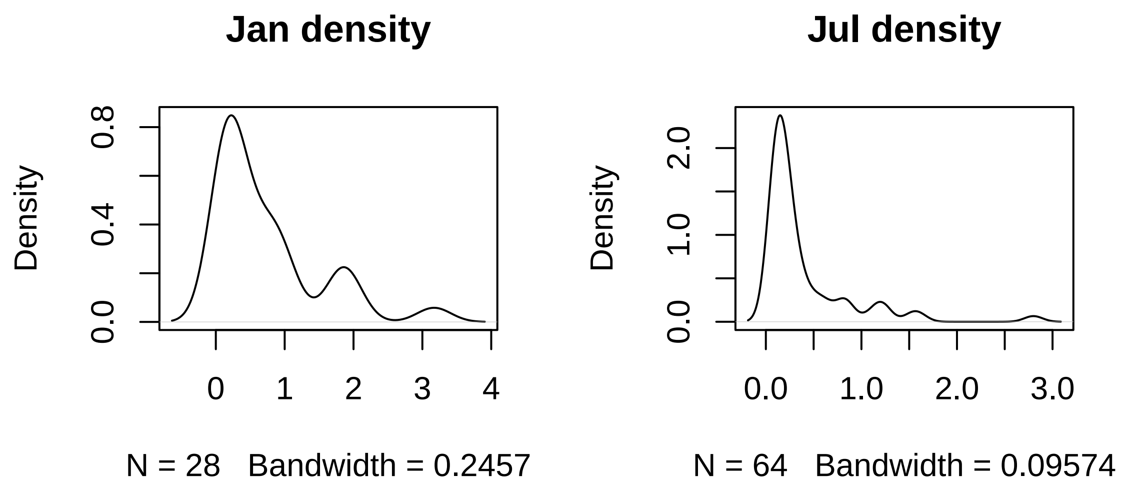
```
qqnorm(Jan, pch = 1)
qqline(Jan, col = "steelblue", lwd = 2)
```



```r
qqnorm(Jul, pch = 1)
qqline(Jul, col = "steelblue", lwd = 2)
```



```r
par(mfrow = c(1, 2))
plot(density(Jan),main='Jan density')
plot(density(Jul),main='Jul density')
```



The qqplots show that the sample doesn't follow normal distribution.
From the density plot, these data looks like gamma distribution. Therefore, gamma distribution can be considered to fit the model.

(c)

There are many ways to solve the problem. I listed three methods here. The first one is to use fitdist:

```
Jan.fit1=fitdist(Jan,'gamma','mle')
Jan.fit1
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##       estimate Std. Error
## shape 1.056222  0.2497495
## rate  1.467650  0.4396202
```

```
Jul.fit1=fitdist(Jul,'gamma','mle')
Jul.fit1
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##       estimate Std. Error
## shape 1.196419  0.1891196
## rate  3.043403  0.5936302
```

The second method is to nlm:

```
#https://stackoverflow.com/questions/59435824/nlm-with-multiple-variables-in-r
data<-Jan
neg_likelihood<-function(param){
  alpha<-param[1]
  beta<-param[2]
  p<-dgamma(data,shape=alpha,scale=1/beta)
  re<--1*sum(log(p))
  return(re)
}
#neg_likelihood(c(0.5,1))
```

```
p <- array(c(0.4, 0.4), dim = c(2, 1))
ans_jan <- nlm(f = neg_likelihood,p,hessian=T)
ans_jan$estimate
```

```
## [1] 1.056259 1.467754
```

```
data<-Jul
ans_jul <- nlm(f = neg_likelihood,p,hessian=T)
ans_jul$estimate
```

```
## [1] 1.196403 3.043315
```

Here is the std

```
#https://stats.stackexchange.com/questions/81542/standard-error-of-mle#:~:text=How%20are%20you%20obtain
sqrt(diag(solve(ans_jan$hessian))) #use hessian matrix to get std
```

```
## [1] 0.2498280 0.4397828
```

```
sqrt(diag(solve(ans_jul$hessian)))
```

```
## [1] 0.1891739 0.5938104
```

For MLE, do some stransformation loglikelihood into MLE:

```
exp(Jan.fit1$loglik)
```

```
## [1] 7.11117e-09
```

```
exp(Jul.fit1$loglik)
```

```
## [1] 0.02638693
```

```
exp(-ans_jan$minimum)
```

```
## [1] 7.11117e-09
```

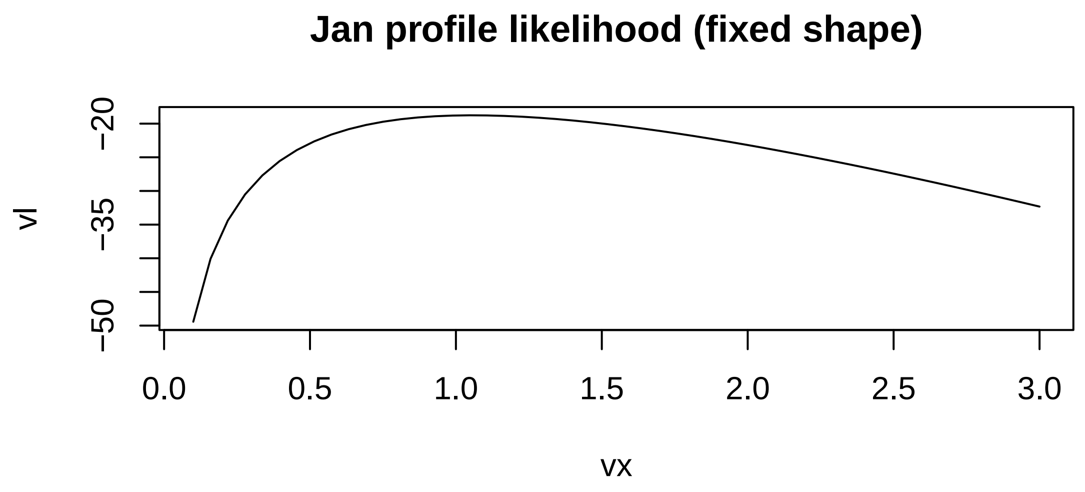
```
exp(-ans_jul$minimum)
```

```
## [1] 0.02638693
```

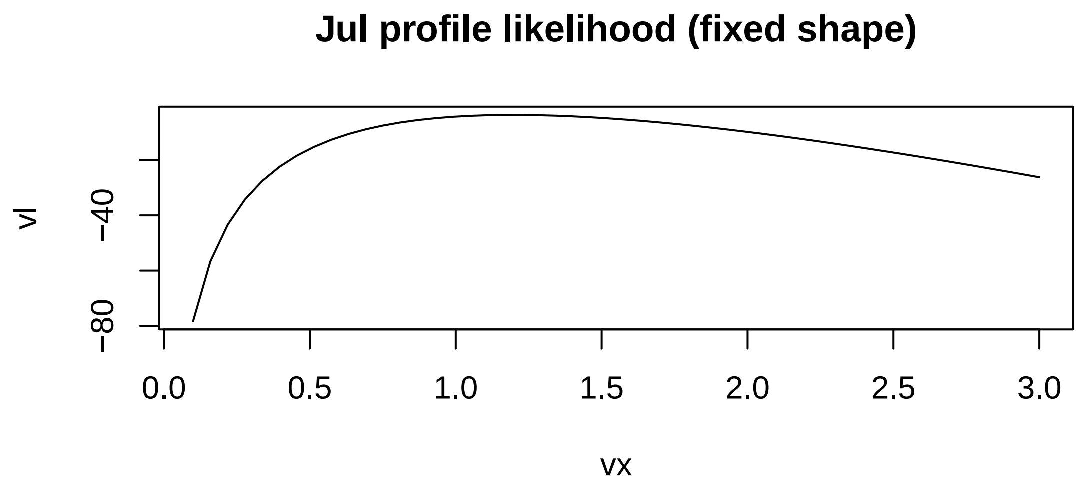
From MLE, Jul's MLE is higher than the one of Jan. Jul's model is better than Jan's.
Parameter comparison: Jan's alpha is lower than Jul's alpha. Jan's beta is lower than Jul's beta.

The third way is to use optim. I will use this method to conduct profile likelihood.

```
#https://www.r-bloggers.com/2015/11/profile-likelihood/
# optim is similar to nlm
# Jan
x=Jan
prof_log_lik=function(a){
   b=(optim(1,function(z) -sum(log(dgamma(x,a,z)))))$par
   return(-sum(log(dgamma(x,a,b))))
 }

vx=seq(.1,3,length=50)
vl=-Vectorize(prof_log_lik)(vx)
plot(vx,vl,type="l",main='Jan profile likelihood (fixed shape)')
```

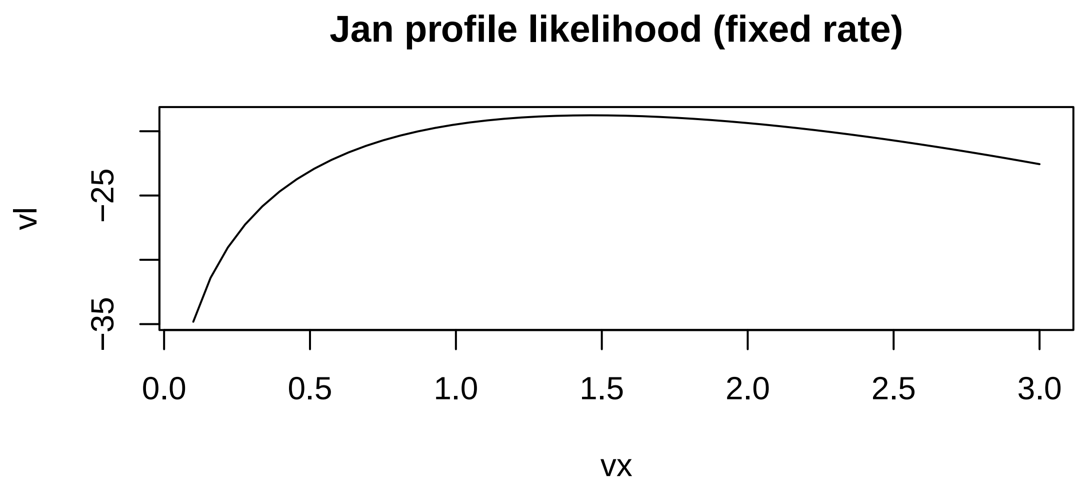


```r
x=Jul
vx=seq(.1,3,length=50)
vl=-Vectorize(prof_log_lik)(vx)
plot(vx,vl,type="l",main='Jul profile likelihood (fixed shape)')
```

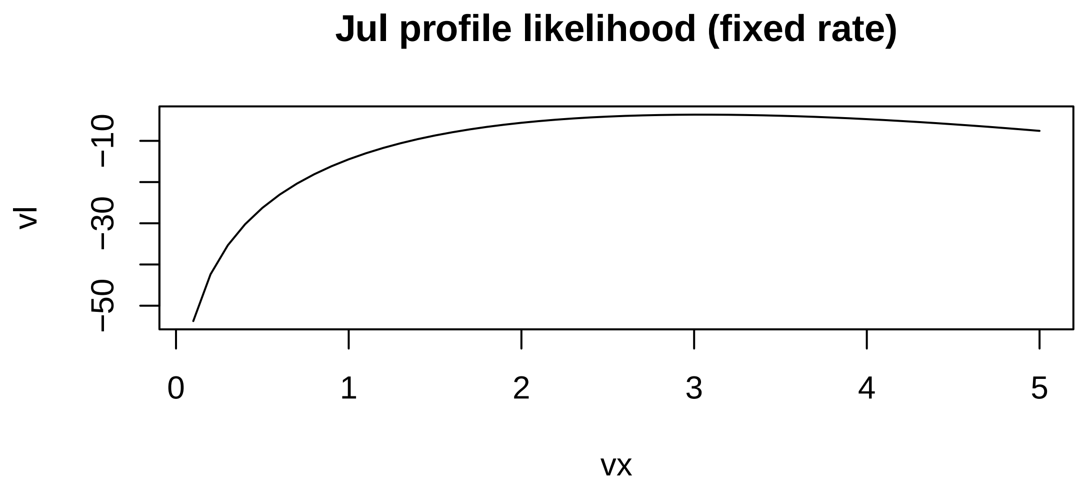


For fixed rate, we can use the same method to get the profile likelihood.

```r
x=Jan
prof_log_lik=function(z){
   a=(optim(1,function(a) -sum(log(dgamma(x,a,z)))))$par
   return(-sum(log(dgamma(x,a,z))))
 }

vx=seq(.1,3,length=50)
vl=-Vectorize(prof_log_lik)(vx)
plot(vx,vl,type="l",main='Jan profile likelihood (fixed rate)')
```



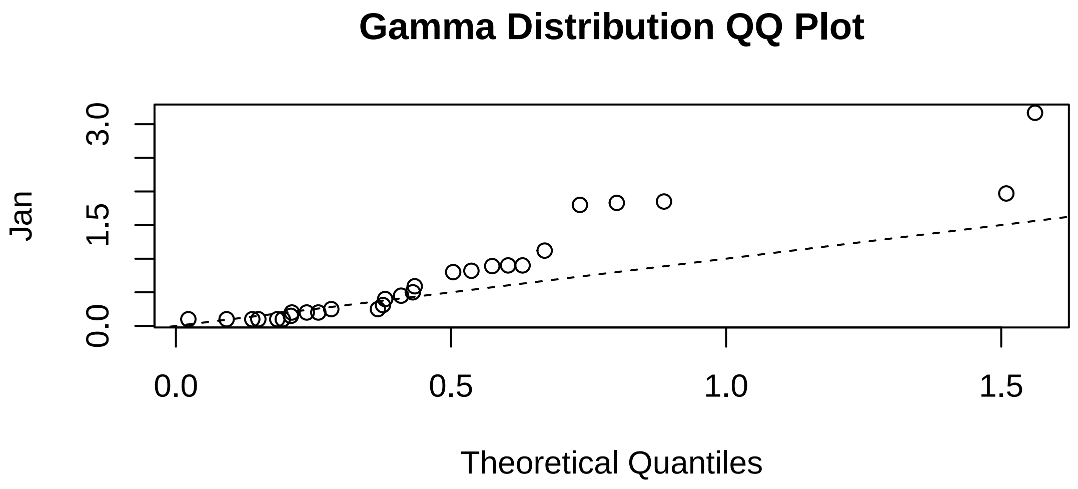
```
x=Jul

vx=seq(.1,5,length=50)
vl=-Vectorize(prof_log_lik)(vx)
plot(vx,vl,type="l",main='Jul profile likelihood (fixed rate)')
```



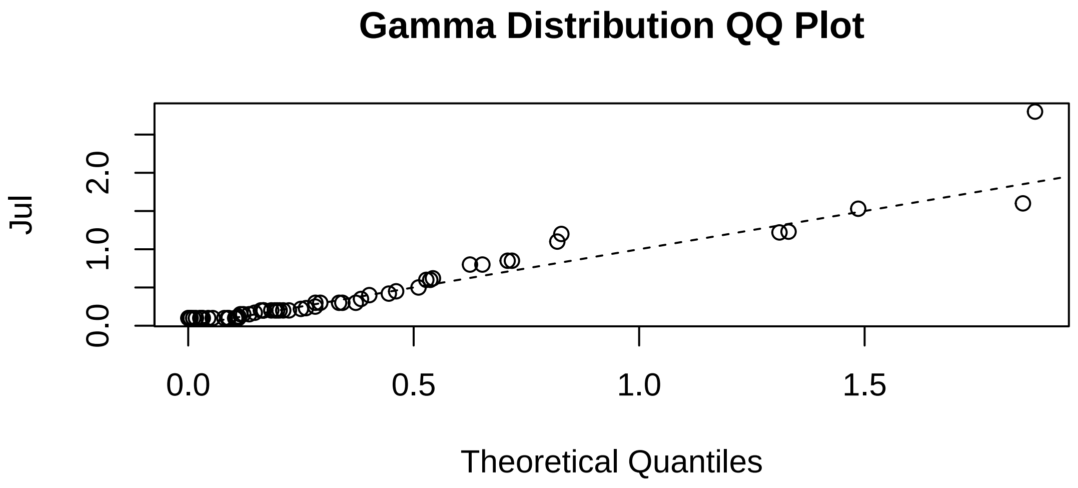
(d)

```
# library(qpToolkit)
# qqGamma(resid(Jan.fit))
# reference:qpToolkit
# https://github.com/qPharmetra/qpToolkit/blob/master/R/qqGamma.r
qqGamma <- function(x
                    , ylab = deparse(substitute(x))
                    , xlab = "Theoretical Quantiles"
                    , main = "Gamma Distribution QQ Plot",...)
```

```
{
    # Plot qq-plot for gamma distributed variable

    xx = x[!is.na(x)]
    aa = (mean(xx))^2 / var(xx)
    ss = var(xx) / mean(xx)
    test = rgamma(length(xx), shape = aa, scale = ss)

    qqplot(test, xx, xlab = xlab, ylab = ylab, main = main,...)
    abline(0,1, lty = 2)
}


qqGamma(Jan)
```



```
qqGamma(Jul)
```



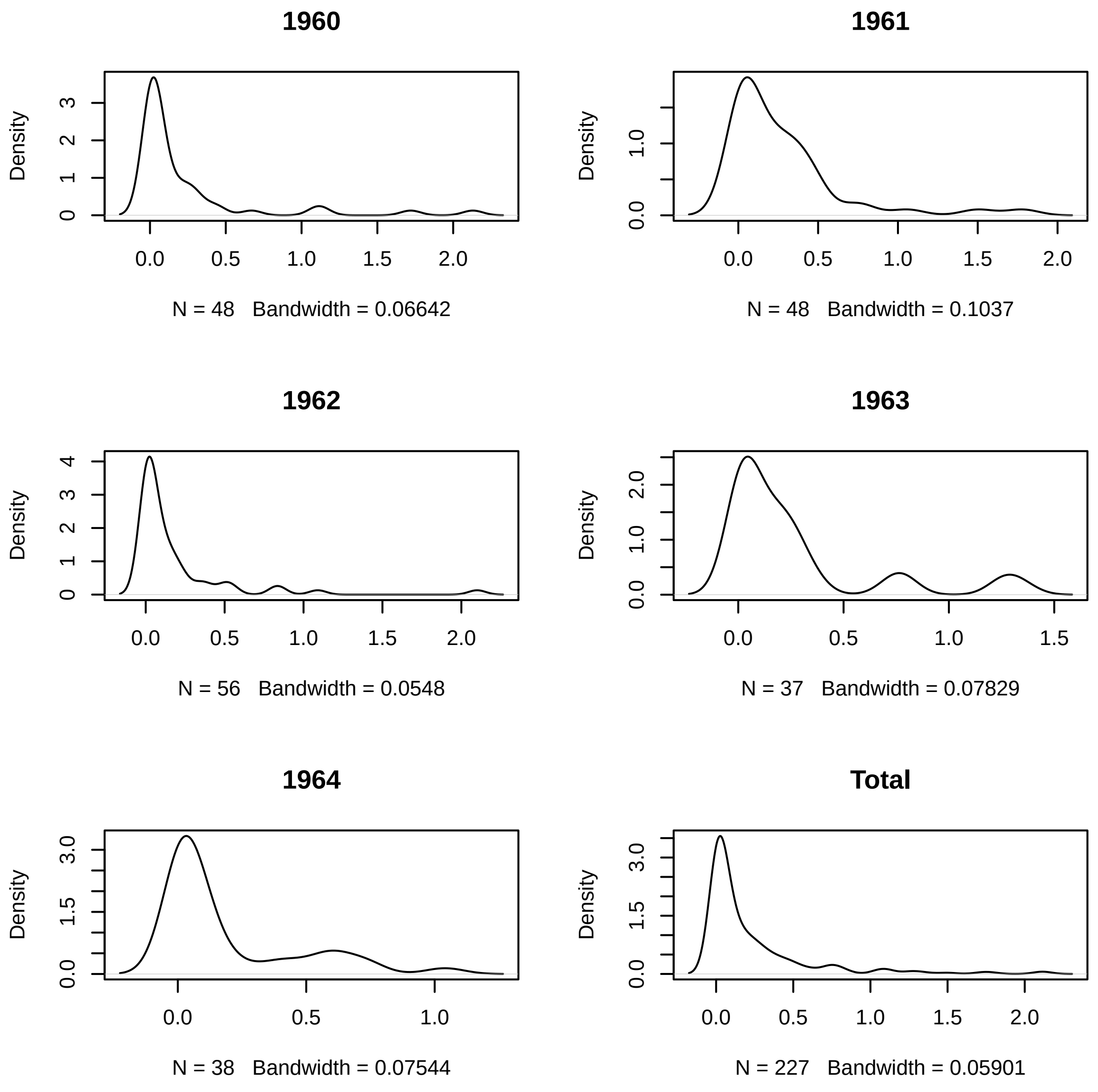
It seems that Jul is better.

Illinois rain

Q1

Use the data to identify the distribution of rainfall produced by the storms in southern Illinois. Estimate the parameters of the distribution using MLE. Prepare a discussion of your estimation, including how confident you are about your identification of the distribution and the accuracy of your parameter estimates.

```r
rain=read.xlsx('Illinois_rain_1960-1964.xlsx')
par(mfrow = c(3, 2))
density(rain$`1960` %>% na.omit()) %>% plot(main='1960')
density(rain$`1961` %>% na.omit()) %>% plot(main='1961')
density(rain$`1962` %>% na.omit()) %>% plot(main='1962')
density(rain$`1963` %>% na.omit()) %>% plot(main='1963')
density(rain$`1964` %>% na.omit()) %>% plot(main='1964')
density(unlist(rain) %>%  na.omit()) %>% plot(main='Total')
```



First I use the whole dataset to conduct fitdist. For method, MLE and MSE are selected to compare which method is better.

```r
fit1<-fitdist(unlist(rain) %>%  na.omit() %>% c(),'gamma',method='mle') #MLE estimation
fit2<-fitdist(unlist(rain) %>%  na.omit() %>% c(),'gamma',method='mse') #MSE estimation
```

```r
summary(bootdist(fit1)) #boot get confidence interval
summary(bootdist(fit2)) #boot get confidence interval
```

Table 1: MLE fit of Rain

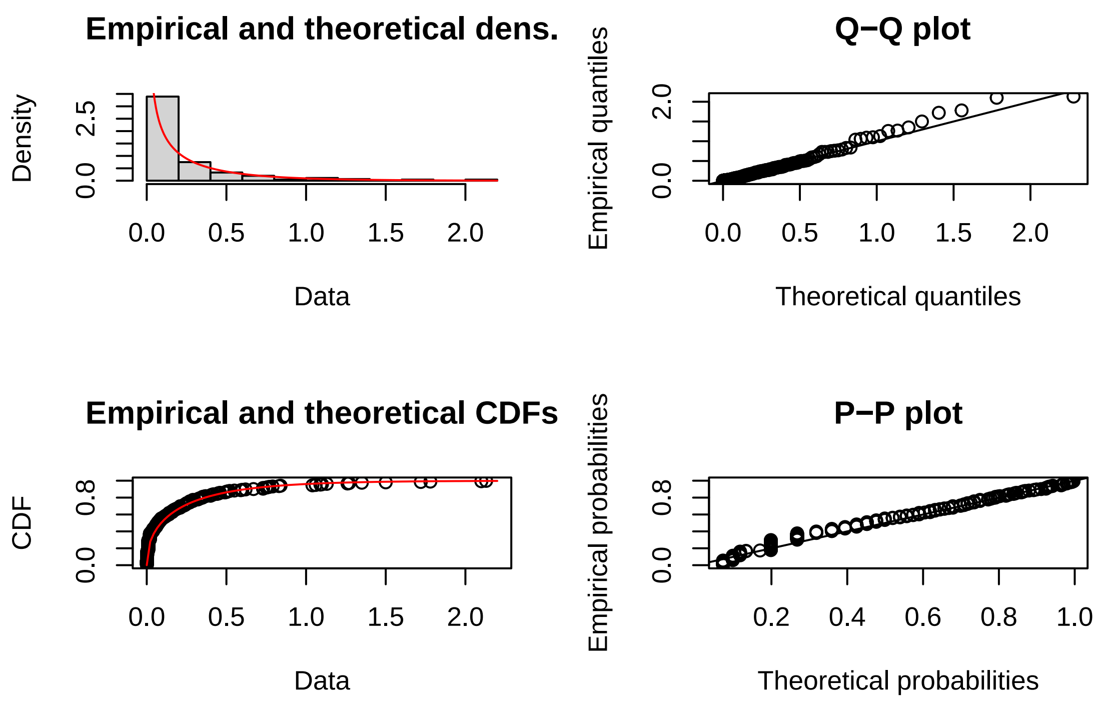
	Median	2.5%	97.5%
shape	0.4447568	0.3867703	0.521934
rate	1.9825172	1.5781062	2.567854

Table 2: MSE fit of Rain

	Median	2.5%	97.5%
shape	0.3917468	0.2726087	0.5300762
rate	1.7586151	1.1622105	2.5227109

The median and 95% confidence interval is shown in the above table. Compared to MSE, MSE's confidence interval is much narrower. Therefore, MLE fits the rain data better. The confidence interval indicates that the estimation is reliable.

```r
plot(fit1)
```



Q2

Using this distribution, identify wet years and dry years. Are the wet years wet because there were more storms, because individual storms produced more rain, or for both of these reasons?

```
rain_mean=fit1$estimate[1]/fit1$estimate[2] #get mean for whole dataset
re=apply(rain,2,mean,na.rm =TRUE) # get mean for each year

out<-c(re,rain_mean %>% as.numeric() %>% round(4))
names(out)[6]='mean'
#out

num_storm<-c(nrow(rain)-apply(is.na(rain),2,sum),'/')

knitr::kable(rbind(out,num_storm)) # show the result
```

	1960	1961	1962	1963	1964	mean
out	0.220291666666667	0.2749375	0.18475	0.262432432432432	0.187105263157895	0.2244
num_storm	48	48	56	37	38	/

Compared to mean, 1962, 1964 are dryer years, 1961 and 1963 are wetter years. 1960 is the normal year. We can also conclude that more storms don't necessarily result in wet year and more rain in individual storm don't necessarily result in wet year.
Therefore, both of these reasons have influence on amount of rainfall.

I also conduct fitdist on each individual year. The results are shown below:

	1960	1961	1962	1963	1964
mean	0.22032	0.27494	0.18475	0.26245	0.18713
num_storm	48	48	56	37	38

Q3

To what extent do you believe the results of your analysis are generalizable? What do you think the next steps would be after the analysis? An article by Floyd Huff, one of the authors of the 1967 report is included.

Five years' data isn't credential enough for verification. The next step is to collect more data to confirm the result. Huff's article mainly focuses on description statistics. He didn't build a reliable data to further analyze it.

Bayes Part

Auto Insurance Company

```
auto=data.frame(Claims=seq(0,7),
          Counts=c(7840,1317,239,42,14,4,4,1))
```

```
l=length(auto$Counts)
robbin1<-(auto$Counts[2:l]/auto$Counts[1:l-1])*(auto$Claims+1)[1:l-1]
robbin1
```

```
## [1] 0.1679847 0.3629461 0.5271967 1.3333333 1.4285714 6.0000000 1.7500000
```

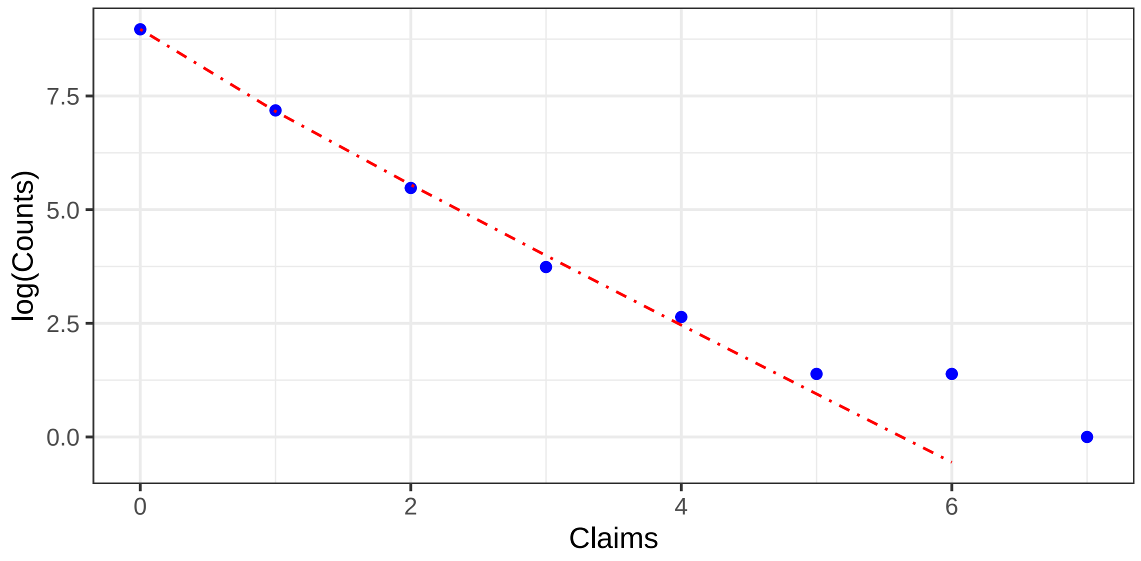
```
#idea from:
#https://github.com/jrfiedler/CASI_Python/blob/master/chapter06/ch06s01.ipynb



f<-function(x,mu,sigma){
  gamma = sigma / (1 + sigma)
  numer = gamma ^ (mu + x) * gamma(mu + x)
  denom = sigma ^ mu * gamma(mu) * factorial(x)
  return(numer/denom)
}

neg_like<-function(param){
  mu=param[1]
  sigma=param[2]
  tmp=-sum(auto$Counts*log(f(auto$Claims,mu=mu,sigma=sigma)))
  return(tmp)
}


p <- array(c(0.5, 1), dim = c(2, 1))
ans_auto <- nlm(f = neg_like,p,hessian=T)

mu=ans_auto$estimate[1]
sigma=ans_auto$estimate[2]

re=(seq(0,6)+1)*f(seq(0,6)+1,mu,sigma)/f(seq(0,6),mu,sigma)
re %>% round(3)
```

```
## [1] 0.164 0.398 0.632 0.866 1.100 1.334 1.568
```

```
rbind(robbin1,re %>% round(3))
```

```
##              [,1]      [,2]      [,3]     [,4]     [,5]  [,6]  [,7]
## robbin1 0.1679847 0.3629461 0.5271967 1.333333 1.428571 6.000 1.750
##         0.1640000 0.3980000 0.6320000 0.866000 1.100000 1.334 1.568
```

```
auto$pred=c(f(seq(0,6),mu,sigma)*9461,NA)
auto %>% ggplot() + geom_point(aes(x=Claims,y=log(Counts)),color='blue') +geom_line(aes(x=Claims,y=log(
```



Missing species

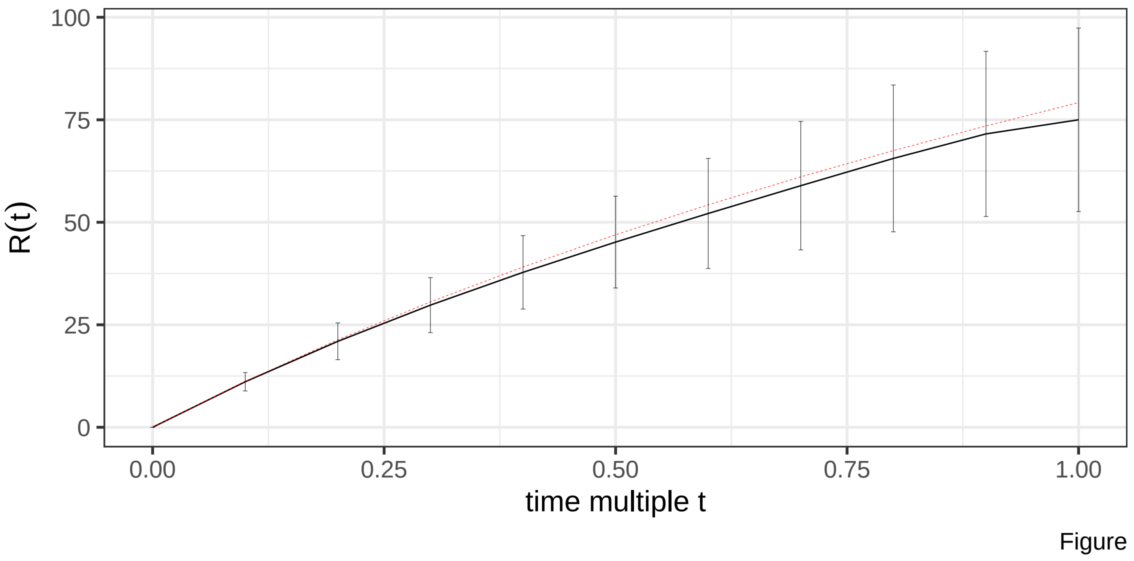
```
butterfly=data.frame(y=c(118,74,44,24,29,22,20,19,20,15,12,14,
               6,12,6,9,9,6,10,10,11,5,3,3),
           x=seq(1,24))

Fisher1<-function(t){
  re<-butterfly$y * t^(butterfly$x)* (-1)^(butterfly$x-1)
  sd<-(sum(butterfly$y * (t)^(2)))^(1/2)
  return(list('est'=sum(re),'sd'=sd))
}

F1<-sapply(seq(0,1,0.1),Fisher1)
F1
```

```
##     [,1] [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## est 0    11.10187 20.96169 29.79148 37.79271 45.17149 52.14693 58.92833
## sd  0    2.238303 4.476606 6.714909 8.953212 11.19151 13.42982 15.66812
##     [,9]     [,10]    [,11]
## est 65.57362 71.55992 75
## sd  17.90642 20.14473 22.38303
```

```
v <- 0.104
sigma <-  89.79
gamma <- sigma / (1 + sigma)
e1 <- 118
fisherFn <- function(t){
  re<-e1*((1 - (1+gamma*t)^(-v)) / (gamma * v))
  return(re)
}

EST2<-sapply(seq(0,1,0.1),fisherFn)
EST2
```

```
##  [1]  0.00000 11.19732 21.33347 30.58504 39.08842 46.95109 54.25922 61.08287
##  [9] 67.47981 73.49817 79.17850
```

```r
df<-data.frame(time=seq(0,1,0.1),est1=unlist(F1[1,]),sd=unlist(F1[2,]),est2=EST2)
df %>% ggplot() +
geom_line(mapping = aes(x = time, y = est1), size = 0.25) +
geom_line(mapping = aes(x = time, y = est2), color = "red", size = 0.1, linetype = "dashed") +
## geom_hline(yintercept = 0.0, color = "blue", linetype="dotted") +
## geom_vline(xintercept = 0.0, color = "blue", linetype="dotted") +
geom_errorbar(mapping = aes(x = time, ymin = (est1 - sd),
ymax = (est1 + sd)),
width=0.005, color="black", size = 0.001) +
labs(x = "time multiple t", y = expression(R(t)), caption = "Figure")+theme_bw()
```



Shaksapre

For this data, there is no specific requirement in the book. Therefore, I use haviland's code.

```r
data("bardWordCount", package = "deconvolveR")
lambda <- seq(-4, 4.5, .025)
tau <- exp(lambda)
result <- deconv(tau = tau, y = bardWordCount, n = 100, c0=2)
stats <- result$stats

d <- data.frame(lambda = lambda, g = stats[, "g"], tg = stats[, "tg"],
SE.g = stats[, "SE.g"])
indices <- seq(1, length(lambda), 5)
print(
ggplot(data = d) +
geom_line(mapping = aes(x = lambda, y = g)) +
geom_errorbar(data = d[indices, ],
mapping = aes(x = lambda, ymin = g - SE.g, ymax = g + SE.g),
width = .01, color = "green") +
labs(x = expression(log(theta)), y = expression(g(theta))) +
##ylim(-0.001, 0.006) +
xlim(-4, 4) +
geom_vline(xintercept = 0.0, linetype = "dotted", color = "blue") +
geom_hline(yintercept = 0.0, linetype = "dotted", color = "blue") +
geom_line(mapping = aes(x = lambda, y = tg),
linetype = "dashed", color = "red") +
annotate("text", x = c(-4, -3, -2, -1, 0, 1, 2, 3, 4),
y = rep(-0.0005, 9),
label = c("0.02", "0.05", "0.14", "0.37", "1.00", "2.72", "7.39", "20.09", "90.02"), size = 2) +
scale_y_continuous(breaks = c(-0.0005, 0.0, 0.002, 0.004, 0.006),
labels = c(expression(theta), "0.000", "0.002", "0.004", "0.006"),
limits = c(-0.0005, 0.006)) +
labs(caption="Figure 1")
)
```

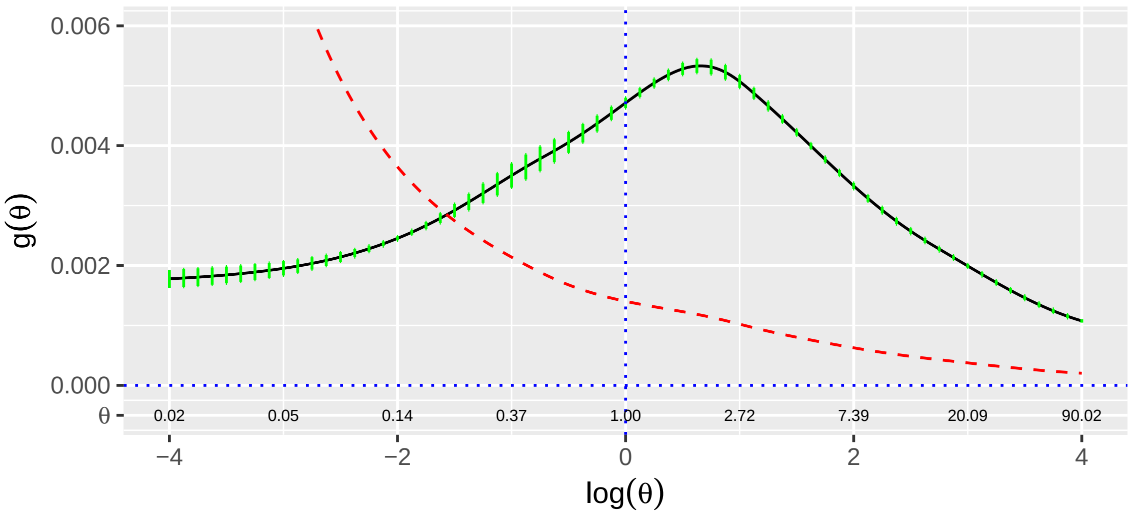
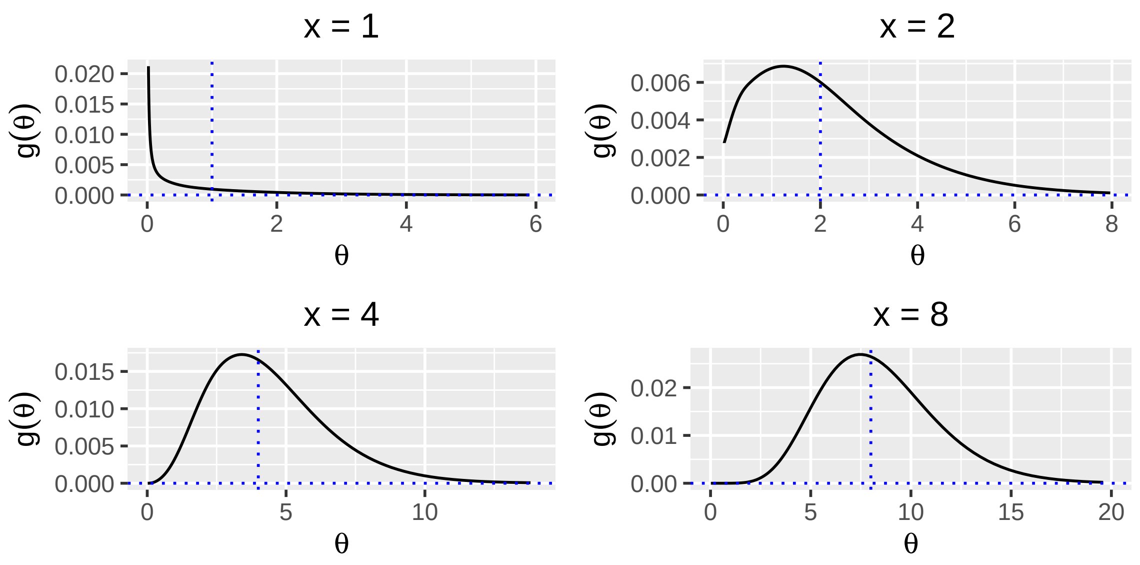


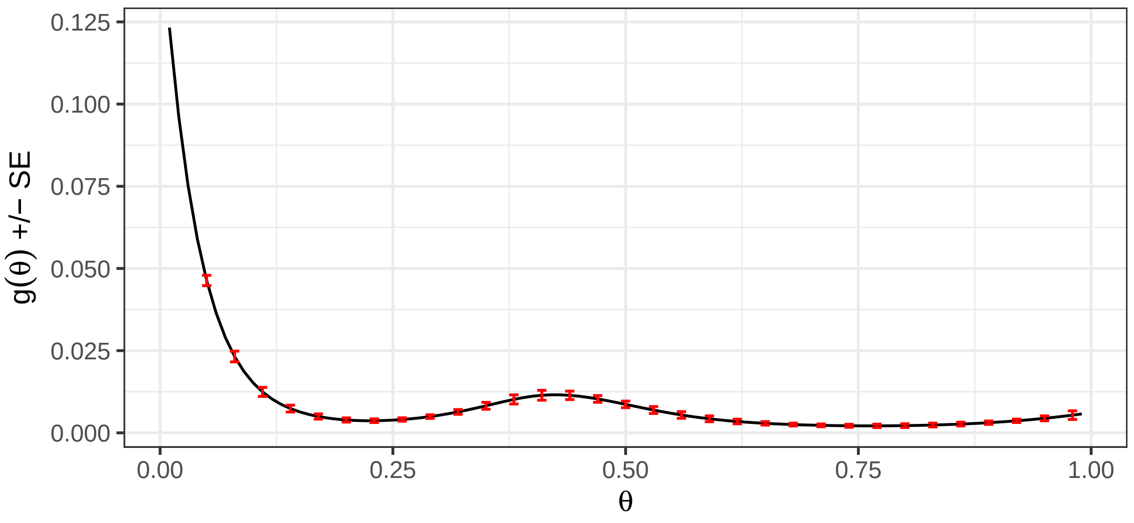
Figure 1

```
library("cowplot")
gPost <- sapply(seq_len(100), function(i) local({tg <- d$tg * result$P[i, ]; tg / sum(tg)}))
plots <- lapply(c(1, 2, 4, 8), function(i) {
ggplot() +
geom_line(mapping = aes(x = tau, y = gPost[, i])) +
geom_vline(xintercept = i, linetype = "dotted", color = "blue") +
geom_hline(yintercept = 0.0, linetype = "dotted", color = "blue") +
labs(x = expression(theta), y = expression(g(theta)),
title = sprintf("x = %d", i))
})
plots <- Map(f = function(p, xlim) p + xlim(0, xlim) + theme(plot.title=element_text(hjust=0.5)),
plots, list(6, 8, 14, 20))
print(plot_grid(plotlist = plots, ncol = 2))
```



Medical Example

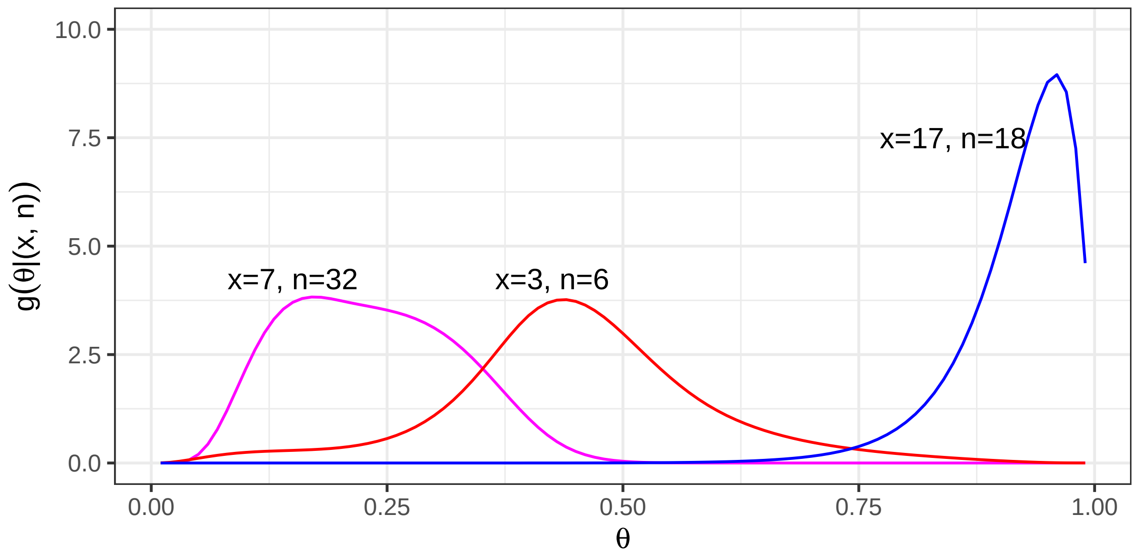
```
#alternative:data(surg)
nodes=read.table('nodes.txt',header=T)
p<-nodes$x/nodes$n
tau <- seq(from = 0.01, to = 0.99, by = 0.01)
result <- deconv(tau = tau, X = nodes, family = "Binomial")
d <- data.frame(result$stats)
indices <- seq(5, 99, 3)
errorX <- tau[indices]

ggplot() +
geom_line(data = d, mapping = aes(x = tau, y = g)) +
geom_errorbar(data = d[indices, ],
mapping = aes(x = theta, ymin = g - SE.g, ymax = g + SE.g),
width = .01, color = "red") +
labs(x = expression(theta), y = expression(paste(g(theta), " +/- SE")), caption = "Figure")+theme_bw()
```



Figure

```
# idea from:
# https://github.com/bnaras/deconvolveR/blob/master/vignettes/deconvolution.Rmd

theta <- result$stats[, 'theta']
gTheta <- result$stats[, 'g']
f_alpha <- function(n_k, x_k) {
    ## .01 is the delta_theta in the Riemann sum
    sum(dbinom(x = x_k, size = n_k, prob = theta) * gTheta) * .01
}
g_theta_hat <- function(n_k, x_k) {
    gTheta * dbinom(x = x_k, size = n_k, prob = theta) / f_alpha(n_k, x_k)
}
```

```
g1 <- g_theta_hat(x_k = 7, n_k = 32)
g2 <- g_theta_hat(x_k = 3, n_k = 6)
g3 <- g_theta_hat(x_k = 17, n_k = 18)

ggplot() +
    geom_line(mapping = aes(x = theta, y = g1), col = "magenta") +
    ylim(0, 10) +
    geom_line(mapping = aes(x = theta, y = g2), col = "red") +
    geom_line(mapping = aes(x = theta, y = g3), col = "blue") +
    labs(x = expression(theta), y = expression(g(paste(theta, "|(x, n)")))) +
    annotate("text", x = 0.15, y = 4.25, label = "x=7, n=32") +
    annotate("text", x = 0.425, y = 4.25, label = "x=3, n=6") +
    annotate("text", x = 0.85, y = 7.5, label = "x=17, n=18") + theme_bw()
```



10.0
7.5
5.0
2.5
0.0
g(θ|(x, n))
x=17, n=18
x=7, n=32
x=3, n=6
0.00
0.25
0.50
0.75
1.00
θ