MA 677 Final Report

Shicong Wang

5/11/2022

Part I: In All Likelihood problems

4.25

In this problem, we need to use the distribution of the order statistics.

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

integrand <- function(x,r,n,a=0, b=1) {
  x * (1 - F(x, a, b))^(r-1) * F(x, a, b)^(n-r) * f(x, a, b)
}

## expectation
E <- function(r,n, a=0, b=1) {
  (1/beta(r,n-r+1)) * integrate(integrand,-Inf,Inf, r, n, a, b)$value
}

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

E(2.5,5)
```

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

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

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

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

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

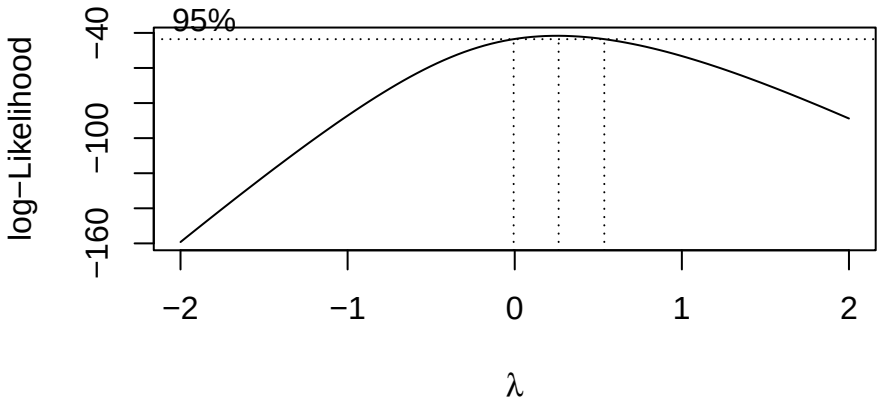
```
medianprrox(5,10)
```

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

We can respectively obtain expectations and medians, and find that expectations are approximately equal to the medians.

4.39

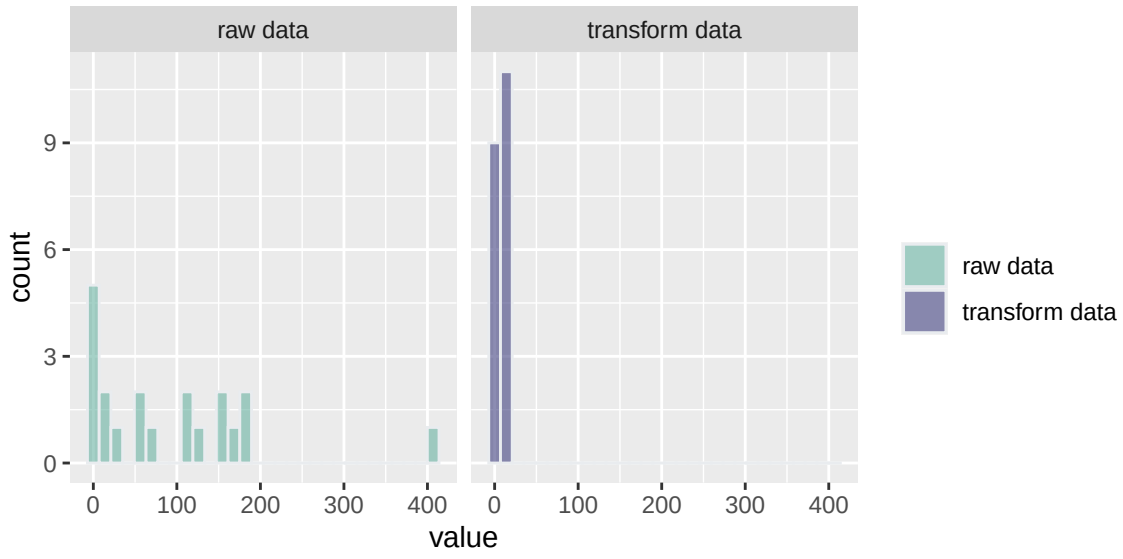
```
data<-c(0.4,1.0,1.9,3.0,5.5,8.1,12.1,25.6,50.0,56.0,70.0,115.0,115.0,119.5,154.5,157.0,175.0,179.0,180.0

# fit linear model
model <- lm(data~1)

#find optimal lambda for Box-Cox transformation
bc <- boxcox(data~1)
```




```
lambda <- bc$x[which.max(bc$y)]

#obtain new data using the Box-Cox transformation
transform_data <- (data ^ lambda - 1) / lambda
```

We can make comparison between the raw data and the transform data by the histogram plot.

```
# make comparison
combine_data <- data.frame(
   type = c( rep("raw data", 20), rep("transform data", 20) ),
   value = c(data, transform_data))

combine_data %>%
    ggplot(aes(x=value, fill=type)) +
    geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity',bins=30) +
    scale_fill_manual(values=c("#69b3a2", "#404080")) +
    #theme_ipsum() +
    labs(fill="")+
   facet_grid(~type)
```



##4.27

```
# obtain the data
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)

```
library(psych)
describe(Jan)
```

```
##    vars  n mean   sd median trimmed  mad min  max range skew kurtosis   se
## X1    1 28 0.72 0.77   0.43    0.62 0.48 0.1 3.17  3.07 1.47     1.63 0.14
```

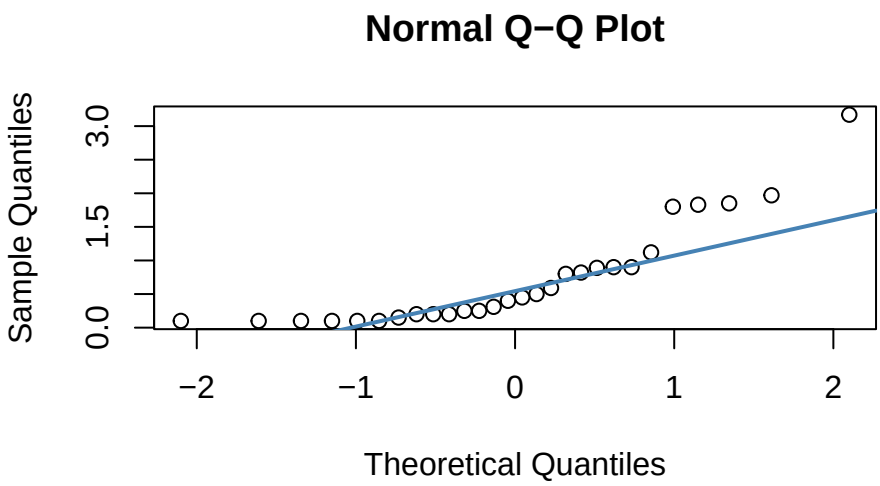
```
describe(Jul)
```

```
##    vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 64 0.39 0.48    0.2    0.29 0.15 0.1 2.8   2.7 2.64     8.41 0.06
```

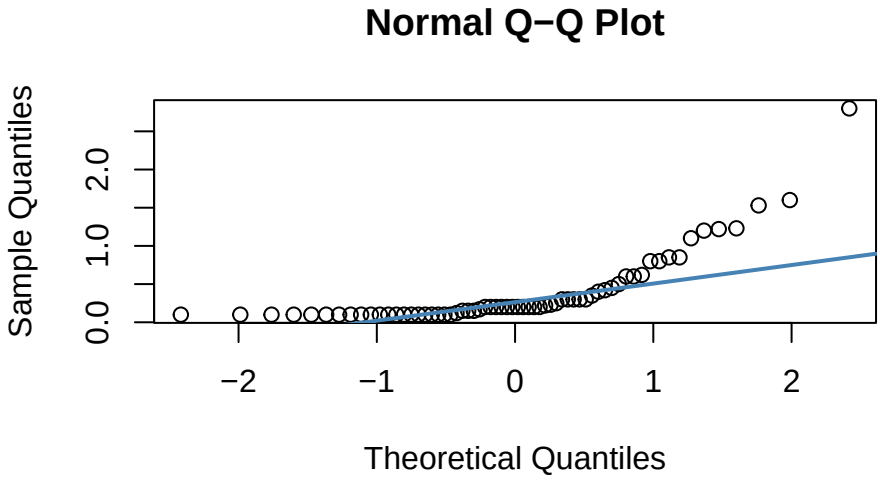
Based on the summary statistics from two data sets, we can conclude that data set Jan contains higher mean, median, max and range values, whereas data set Jul contains more variables and higher skew value.

(b)

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



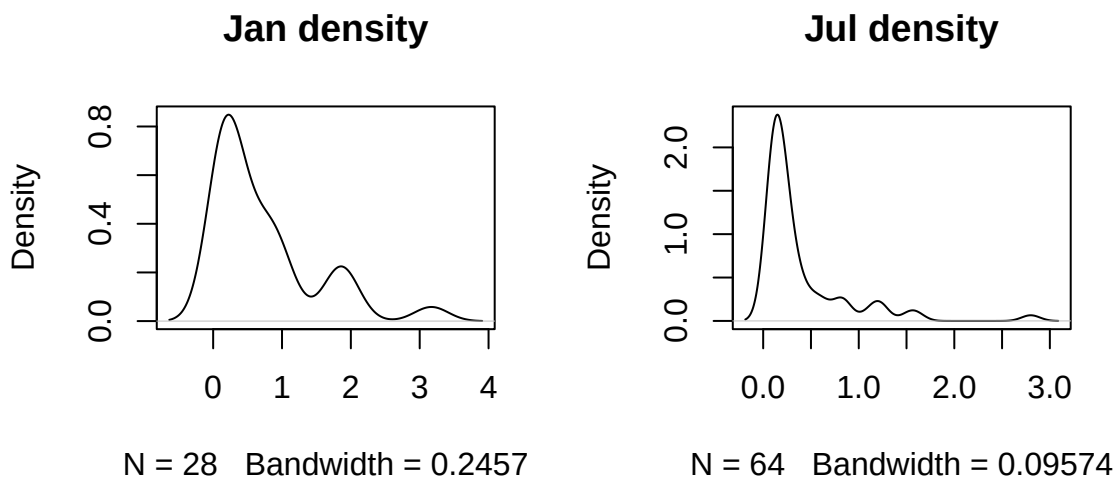

```
qqnorm(Jul, pch = 1)
qqline(Jul, col = "steelblue", lwd = 2)
```



The qqplots have light tails, as the result, we think the normal distribution is unreasonable for this problem.

We generate density plots to prove the conclusion. The distributions are closer to the gamma distribution rather than normal distribution.

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

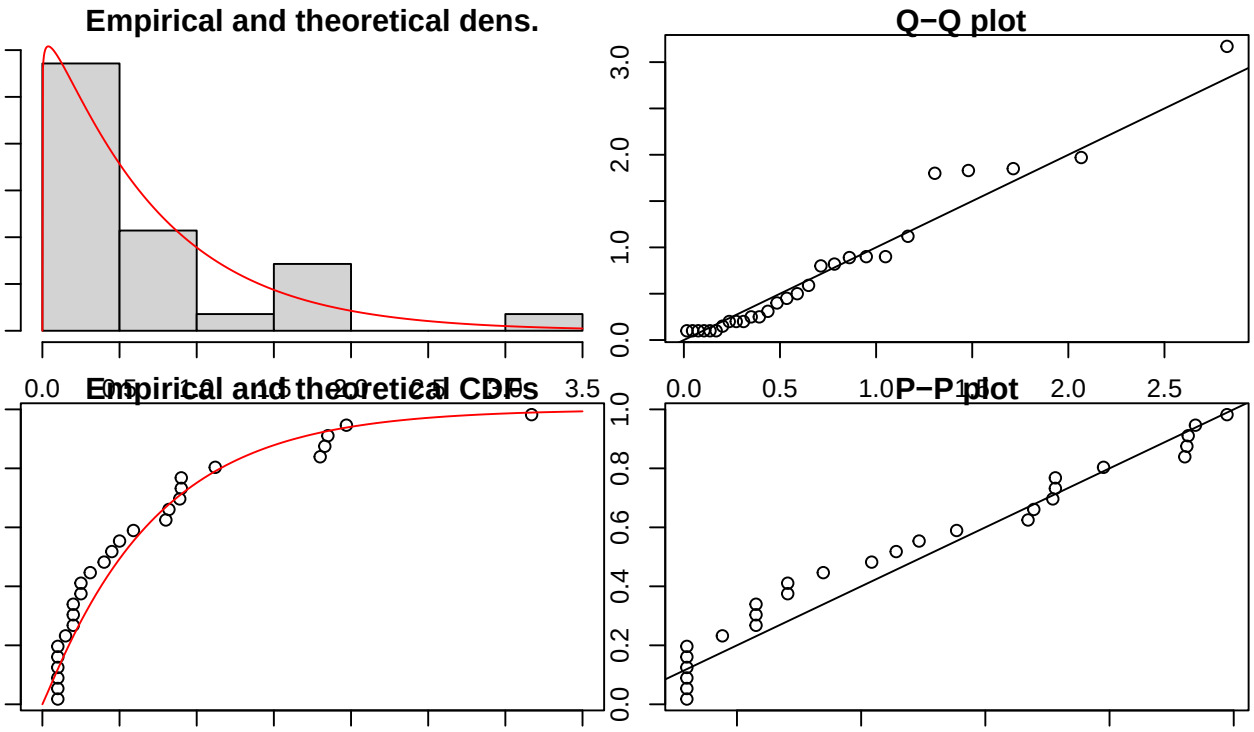


(c)

```
# fit a gamma model
library(fitdistrplus)
Jan.gamma <- fitdist(Jan, distr = "gamma", method = "mle")
summary(Jan.gamma)
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters :
##       estimate Std. Error
## shape 1.056222  0.2497495
## rate  1.467650  0.4396202
## Loglikelihood:  -18.7616   AIC:  41.5232   BIC:  44.18761
## Correlation matrix:
##           shape      rate
## shape 1.0000000 0.7893943
## rate  0.7893943 1.0000000
```

```
par(mar=c(1,1,1,1))
plot(Jan.gamma)
```



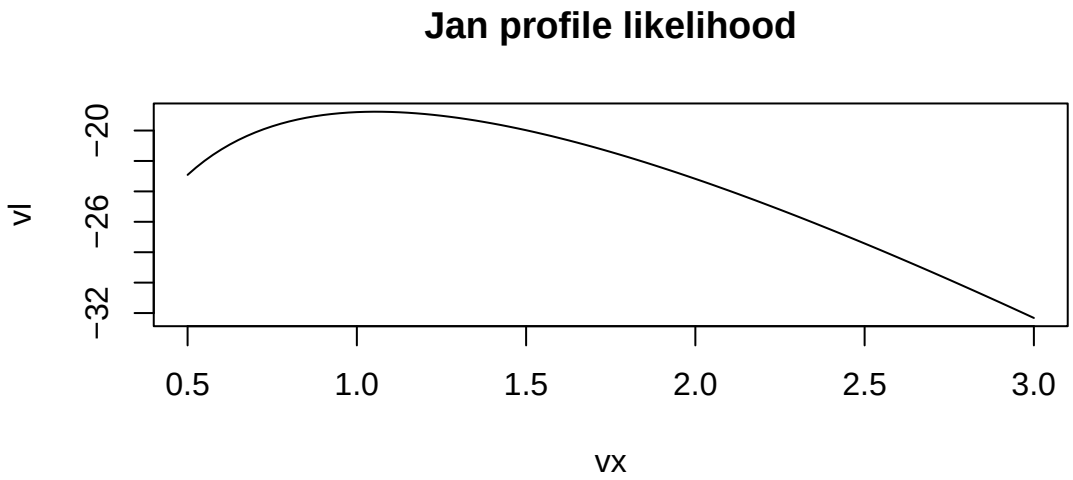

```
# maximum likelihood estimator
Jan.gamma$estimate[1]+c(-1,1)*1.96*Jan.gamma$sd[1]
```

```
## [1] 0.5667132 1.5457314
```

```
# use numerical optimization routine to get the maximum of the log-likelihood function
log_lik=function(theta){
   a=theta[1]
   b=theta[2]
   logL=sum(log(dgamma(Jan,a,b)))
   return(-logL)
 }

optim(c(1,1),log_lik)
```

```
## $par
## [1] 1.056378 1.467814
##
## $value
## [1] 18.7616
##
## $counts
## function gradient
##       55       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```r
# profile likelihood.
prof_log_lik=function(a){
   b=(optim(1,function(z) -sum(log(dgamma(Jan,a,z)))))$par
   return(-sum(log(dgamma(Jan,a,b))))
 }

vx=seq(.5,3,length=101)
vl=-Vectorize(prof_log_lik)(vx)
plot(vx,vl,type="l",main = "Jan profile likelihood")
```




```r
optim(1,prof_log_lik)
```

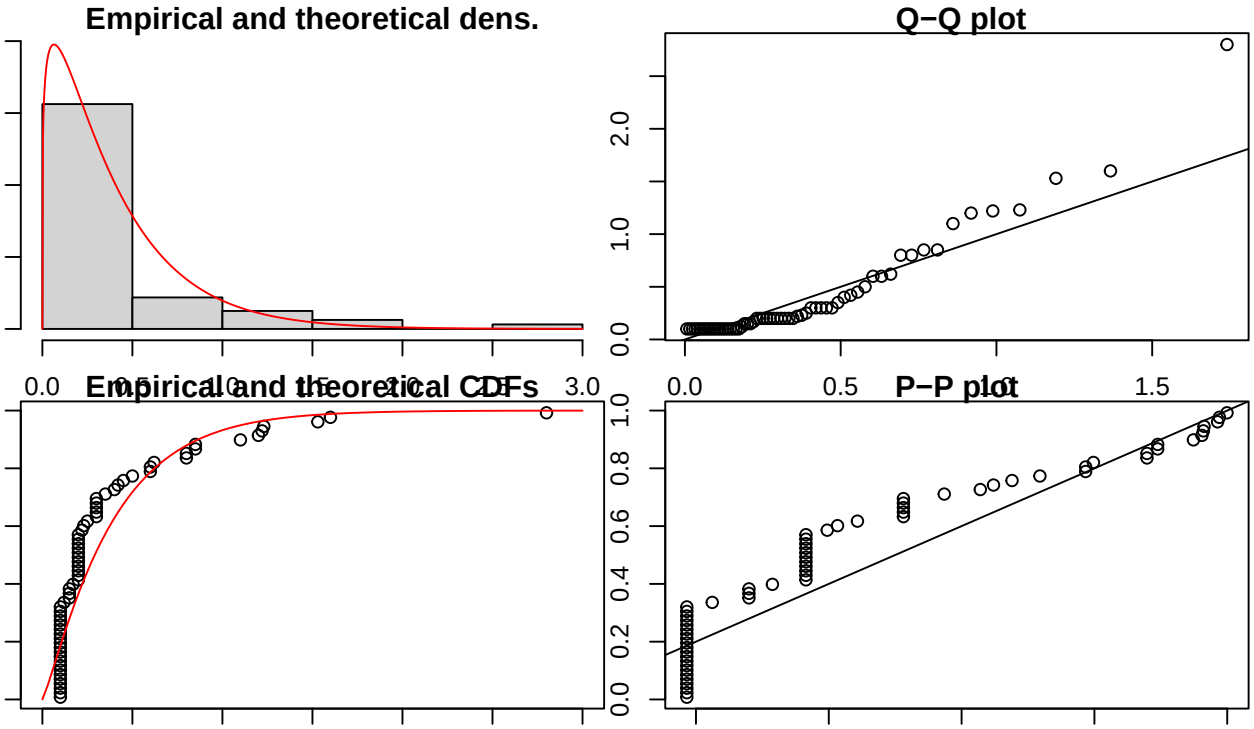
```
## $par
## [1] 1.05625
##
## $value
## [1] 18.7616
##
## $counts
## function gradient
##       20       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```r
Jul.gamma <- fitdist(Jul, distr = "gamma", method = "mle")
summary(Jul.gamma)
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters :
##       estimate Std. Error
## shape 1.196419  0.1891196
## rate  3.043403  0.5936302
## Loglikelihood:  -3.634886   AIC:  11.26977   BIC:  15.58754
## Correlation matrix:
##           shape      rate
## shape 1.0000000 0.8103948
```

```
## rate  0.8103948 1.0000000
```

```
par(mar=c(1,1,1,1))
plot(Jul.gamma)
```




```
# maximum likelihood estimator
Jul.gamma$estimate[1]+c(-1,1)*1.96*Jul.gamma$sd[1]
```

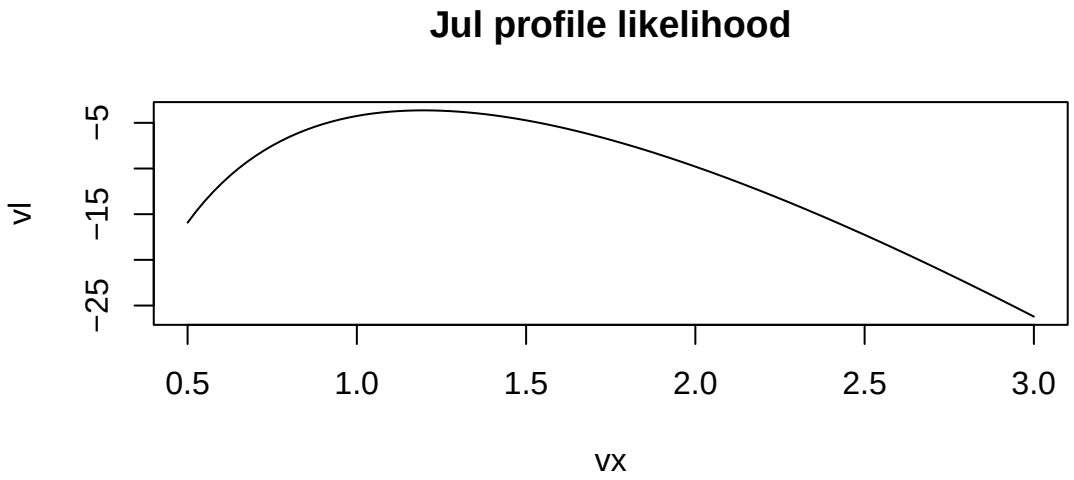
```
## [1] 0.8257444 1.5670931
```

```
# use numerical optimization routine to get the maximum of the log-likelihood function
log_lik=function(theta){
   a=theta[1]
   b=theta[2]
   logL=sum(log(dgamma(Jul,a,b)))
   return(-logL)
 }

optim(c(1,1),log_lik)
```

```
## $par
## [1] 1.196268 3.042774
##
## $value
## [1] 3.634887
##
## $counts
## function gradient
##       71       NA
##
## $convergence
## [1] 0
```

```
##
## $message
## NULL
```

```
# profile likelihood.
prof_log_lik=function(a){
   b=(optim(1,function(z) -sum(log(dgamma(Jul,a,z)))))$par
   return(-sum(log(dgamma(Jul,a,b))))
 }

vx=seq(.5,3,length=101)
vl=-Vectorize(prof_log_lik)(vx)
plot(vx,vl,type="l",main = "Jul profile likelihood")
```




```
optim(1,prof_log_lik)
```

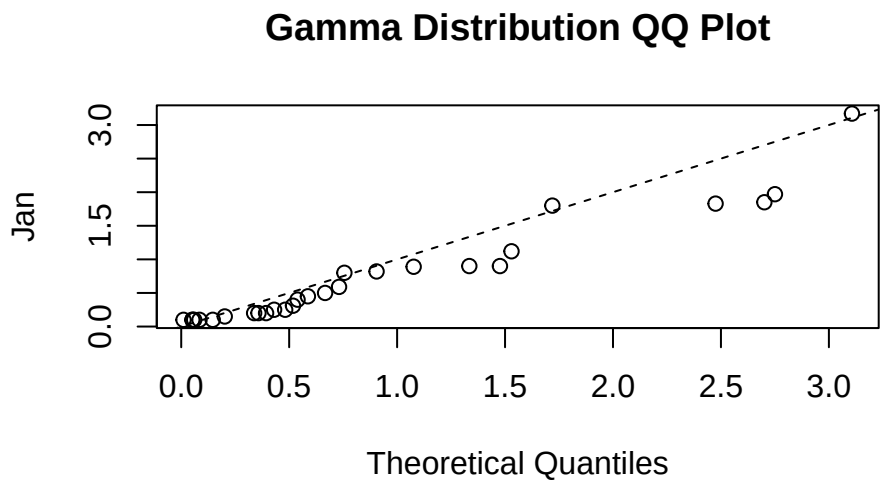
```
## $par
## [1] 1.196289
##
## $value
## [1] 3.634887
##
## $counts
## function gradient
##       22       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Compare the parameters, Jul data set has higher maximum likelihood estimator, and it fits better.

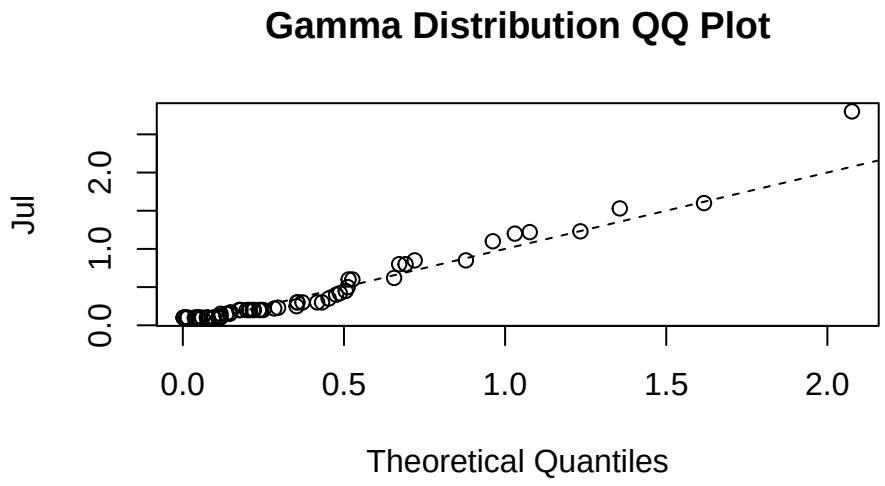
(d)

```
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)
```



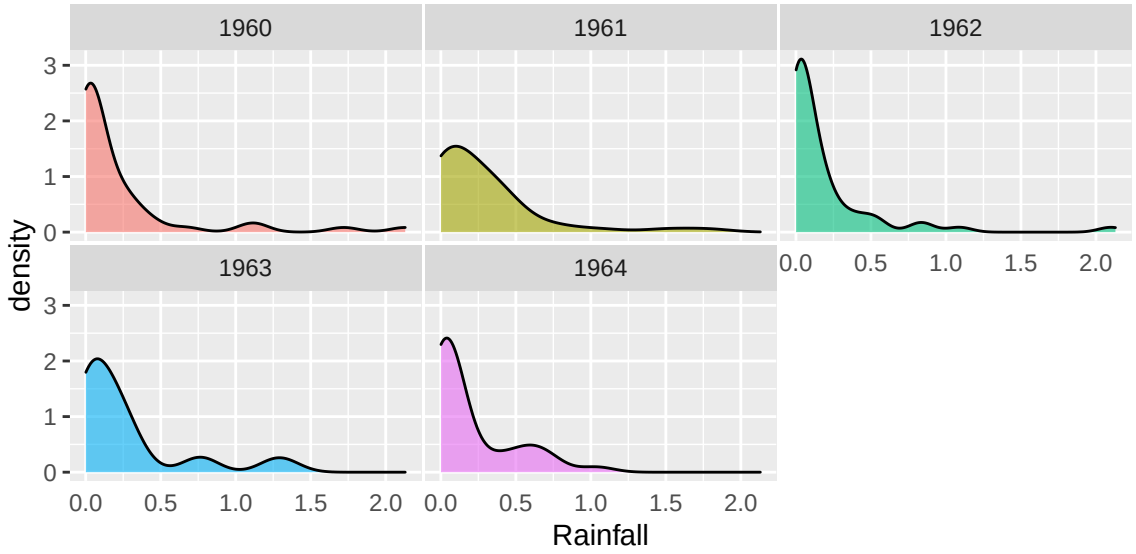
It seems that Jul data set fits better in gamma distribution.

Part II: Illinois rain

The distribution of rainfall

```
# read data
rain<- read.xlsx('Illinois_rain_1960-1964(2).xlsx')
```

```
rain_new<- melt(rain)
colnames(rain_new)<- c("Years", "Rainfall")

# generate density plot
rain_new %>% na.omit() %>%
ggplot(aes(x=Rainfall, group=Years, fill=Years)) +
    geom_density(adjust=1.5, alpha=.6) +
    #theme_ipsum()+
    facet_wrap(~Years) +
    theme(
      legend.position="none",
      panel.spacing = unit(0.1, "lines"),
      axis.ticks.x=element_blank()
    )
```



Given on the plots above, we draw conclusion that the distribution of the rainfalls in each year is close to gamma distribution. Consequently, we fit into a gamma regression model and estimate the parameters of the distribution using MLE.

The 95% confidence interval is shown below:

Table 1: MLE fit of Rain

	Median	2.5%	97.5%
shape	0.4450287	0.3860092	0.5129881
rate	1.9764427	1.5858839	2.5226602

And the plots show the model fit is satisfactory.

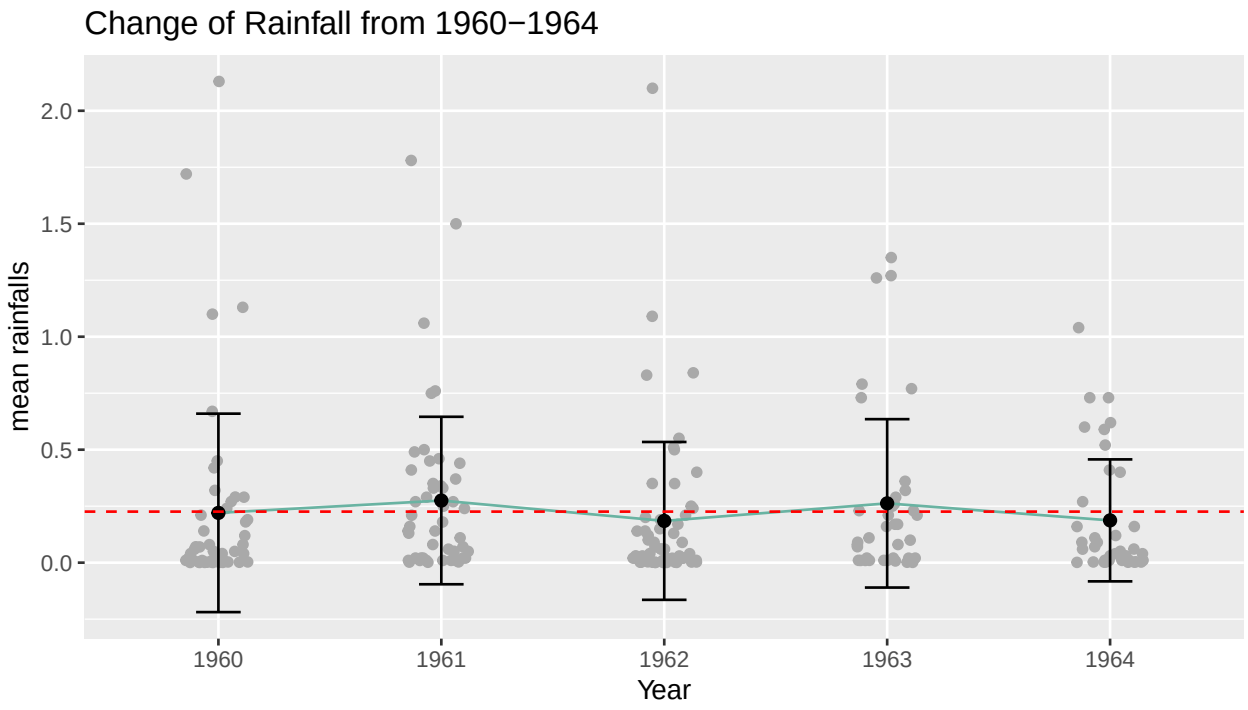
Identify wet years and dry years

```
num_storm<- count(na.omit(rain_new), "Years")


dat2<- rain %>% summarise(Years = c(1960, 1961, 1962, 1963, 1964),
```

```
          sd = apply(X=rain, MARGIN=2, FUN=sd, na.rm=TRUE),
          Rainfall = apply(X=rain, MARGIN=2, FUN=mean, na.rm=TRUE),
          Storm_num = num_storm[,2])

dat2$type<- ifelse(dat2$Rainfall > mean(dat2$Rainfall), "wet",
                   ifelse(dat2$Rainfall > mean(dat2$Rainfall) - 0.01, "normal", "dry"))

dat2
```

```
##   Years        sd  Rainfall Storm_num   type
## 1  1960 0.4391915 0.2202917        48 normal
## 2  1961 0.3706052 0.2749375        48    wet
## 3  1962 0.3493277 0.1847500        56    dry
## 4  1963 0.3726197 0.2624324        37    wet
## 5  1964 0.2699068 0.1871053        38    dry
```

```
ggplot(data = rain_new,aes(x = as.factor(Years), y = Rainfall)) +
  geom_jitter(color = "darkgray",position = position_jitter(0.15))+
  geom_line(aes(group = 1),data= dat2, color="#69b3a2") +
  geom_errorbar(data= dat2,aes(ymin = Rainfall-sd, ymax =Rainfall+sd), width = 0.2)  +
  geom_point(data= dat2,size = 2) +
  geom_hline(yintercept=mean(dat2$Rainfall), linetype="dashed", color = "red")+
  labs(title="Change of Rainfall from 1960-1964", x="Year", y="mean rainfalls")
```



Comparing the mean rainfall of each year, we contrive that the wet years are 1961 and 1963, the dry years are 1964 and 1962, and the normal year is 1960. However, more storms are not correspond to more rainfall.For instance, 1962, the year with most storms, is a dry year. Further more, fewer storms happened in 1963 than 1960, nevertheless the former year is a wet year. .

Extent

The article by Floyd Huff discussed that the individual effects of mean rainfall, storm duration,and other storm factors were small and erratic in behavior when the foregoing analytical technique was used. As a result, we don't have enough confidence to claim that the storm has no relationship with rainfall due to the small data set. What we can extent in next step is collecting enough data to make a more solid conclusion.

Introduction to Empirical Bayes

insurance claims

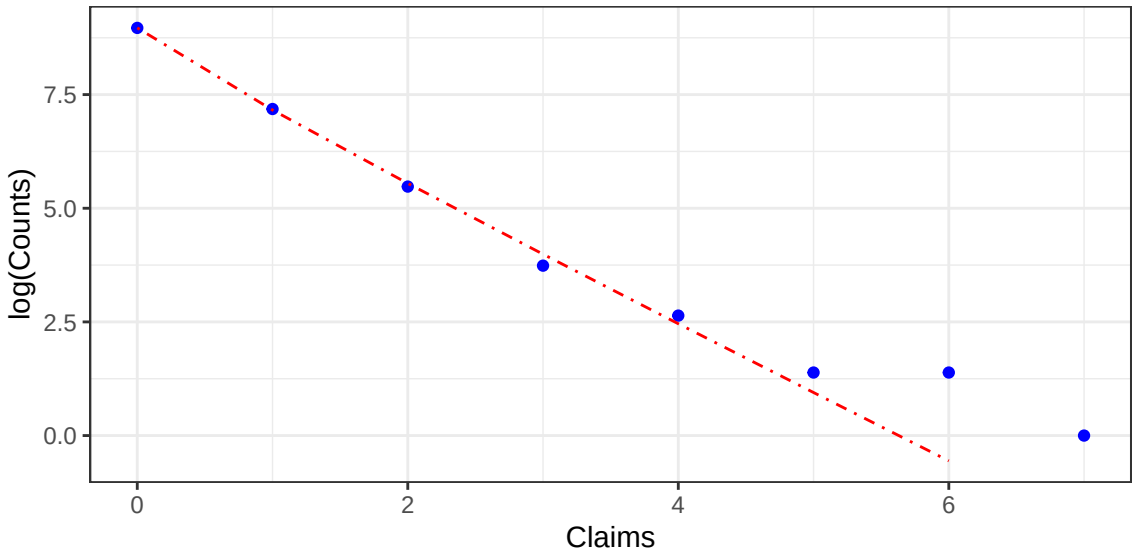
```
auto=data.frame(Claims=seq(0,7),
           Counts=c(7840,1317,239,42,14,4,4,1))
l=length(auto$Counts)
robbin1<-(auto$Counts[2:l]/auto$Counts[1:l-1])*(auto$Claims+1)[1:l-1]
robbin1
```

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

```
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
```

```
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(
```



species discovery

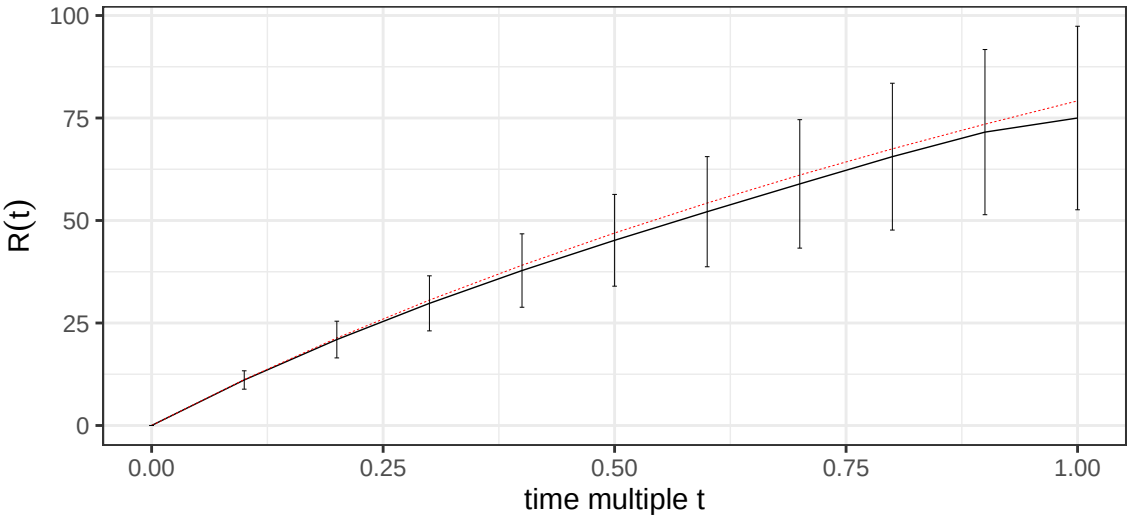
```
butterfly=data.frame(y=c(118,74,44,24,29,22,20,19,20,15,12,14,
               6,12,6,9,9,6,10,10,11,5,3,3),
           x=seq(1,24))

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
```

```
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_errorbar(mapping = aes(x = time, ymin = (est1 - sd),
ymax = (est1 + sd)),width=0.005, color="black", size = 0.1) +
labs(x = "time multiple t", y = expression(R(t)), caption = "Figure")+theme_bw()
```



Figure

Shakespeare's vocabulary

Here we are given the word counts for the entire Shakespeare canon in the data set `bardWordCount`. We assume the ith distinct word appeared $X_i \sim Poisson(\Theta_i)$ times in the canon.

We take the support set $\mathcal{T}$ for Θ to be equally spaced on the log-scale and the sample space for $\mathcal{X}$ to be $(1, 2, \ldots, 100)$.

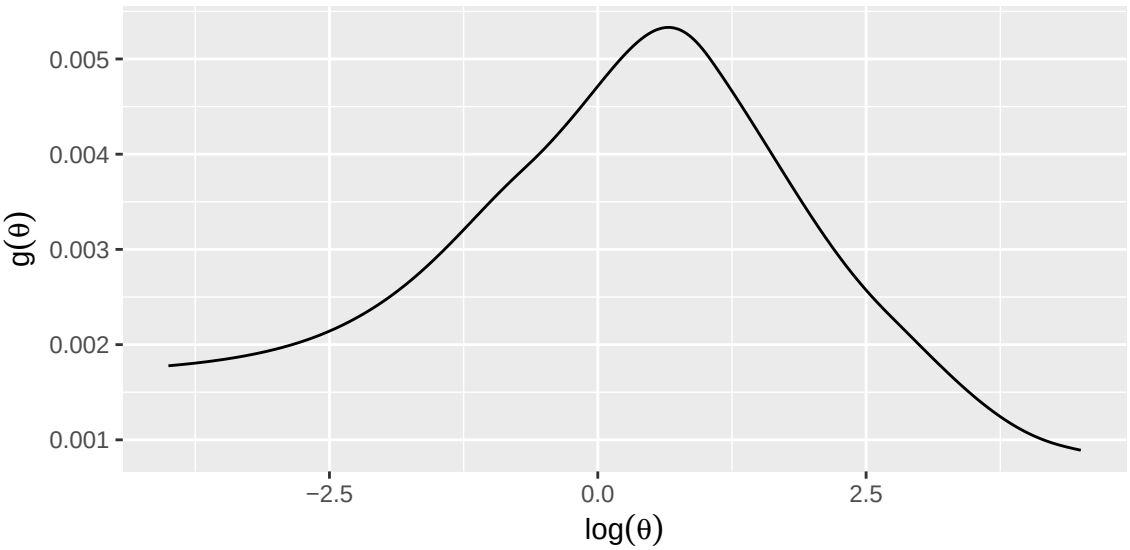
```
data(bardWordCount)
str(bardWordCount)
```

```
##  num [1:100] 14376 4343 2292 1463 1043 ...
```

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

The plot below shows the Empirical Bayes de-convoluation estimates for the Shakespeare word counts.

```
ggplot() +
    geom_line(mapping = aes(x = lambda, y = stats[, "g"])) +
    labs(x = expression(log(theta)), y = expression(g(theta)))
```



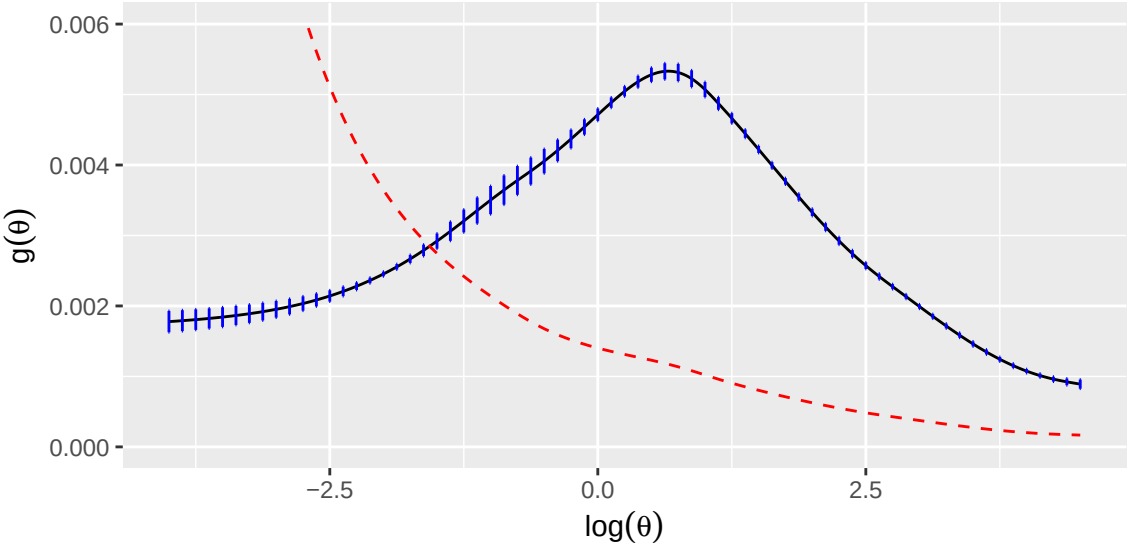
As noted in the paper citing this package, about 44 percent of the total mass of $\hat{g}$ lies below $\Theta = 1$, which is an underestimate. This can be corrected for by defining

$$\tilde{g} = c_1 \hat{g}/(1 - e^{-\theta_j}),$$

where c_1 is the constant that normalizes $\tilde{g}$.

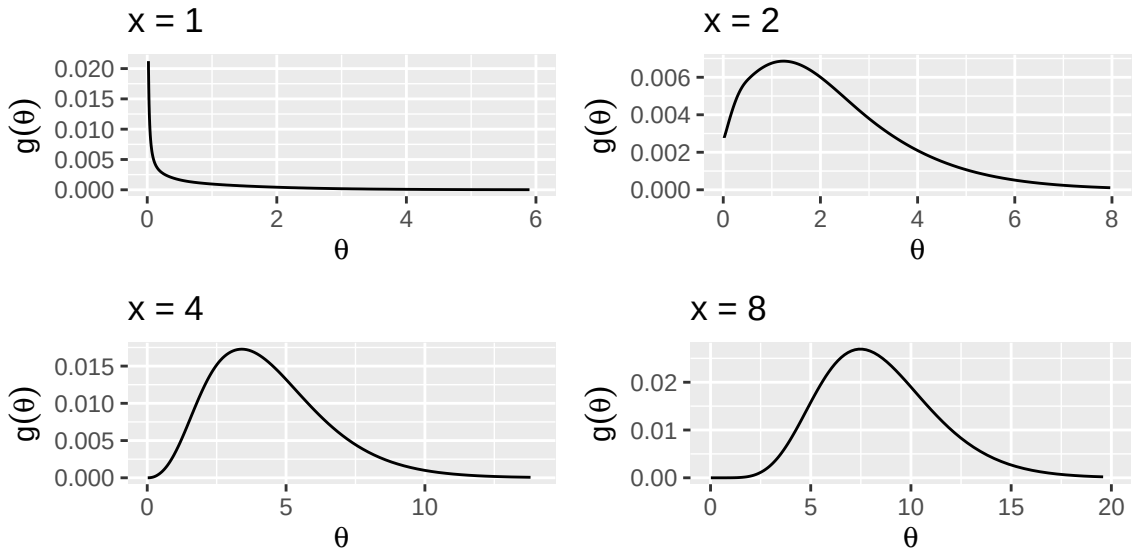
When there is truncation at zero, as is the case here, the `deconvolveR` package now returns an additional column in `stats[, "tg"]` which contains this correction for *thinning*. (The default invocation of `deconv` assumes zero truncation for the Poisson family, argument `ignoreZero = FALSE`).

```
d <- data.frame(lambda = lambda, g = stats[, "g"], tg = stats[, "tg"], SE.g = stats[, "SE.g"])
indices <- seq(1, length(lambda), 5)
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 = "blue") +
    labs(x = expression(log(theta)), y = expression(g(theta))) +
    ylim(0, 0.006) +
    geom_line(mapping = aes(x = lambda, y = tg), linetype = "dashed", color = "red")
```



We can now plot the posterior estimates.

```
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])) +
        labs(x = expression(theta), y = expression(g(theta)),
             title = sprintf("x = %d", i))
})
plots <- Map(f = function(p, xlim) p + xlim(0, xlim), plots, list(6, 8, 14, 20))
plot_grid(plotlist = plots, ncol = 2)
```



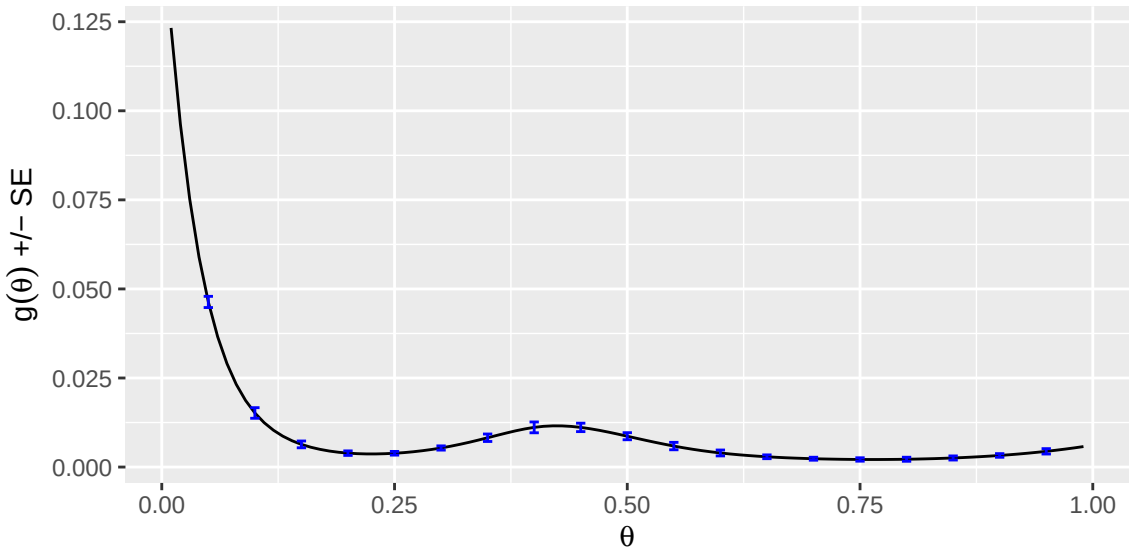
lymph node counts

The dataset `surg` contains data on intestinal surgery on 844 cancer patients. In the study, surgeons removed *satellite* nodes for later testing. The data consists of pairs (n_i, X_i) where n_i is the number of satellites removed and X_i is the number found to be malignant among them.

We assume a binomial model with $X_i \sim Binomial(n_i, \theta_i)$ with θ_i being the probability of any one satellite site being malignant for the ith patient.

We take $\mathcal{T} = (0.01, 0.02, \ldots, 0.09)$, so $m = 99$. We take Q to be the default 5-degree natural spline with columns standardized to mean 0 and sum of squares equal to 1. The penalization parameter is set to 1. The figure below shows the estimated prior density of $g(\theta)$.

```
tau <- seq(from = 0.01, to = 0.99, by = 0.01)
result <- deconv(tau = tau, X = surg, family = "Binomial", c0 = 1)
d <- data.frame(result$stats)
indices <- seq(5, 99, 5)
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 = "blue") +
    labs(x = expression(theta), y = expression(paste(g(theta), " +/- SE")))
```



The complete table of estimates and standard errors is also available.

```
knitr::kable(d[indices, ], row.names = FALSE)
```

theta	g	SE.g	G	SE.G	Bias.g
0.05	0.0463457	0.0015738	0.4000686	0.0187280	-0.0001386
0.10	0.0151816	0.0014905	0.5227891	0.0179368	0.0005557
0.15	0.0063608	0.0009530	0.5679756	0.0177951	0.0004028
0.20	0.0039081	0.0006332	0.5910638	0.0182757	0.0002736
0.25	0.0038724	0.0005167	0.6098155	0.0187490	0.0002012
0.30	0.0053573	0.0005917	0.6330463	0.0188231	0.0000993
0.35	0.0082256	0.0010482	0.6679794	0.0180358	-0.0001306
0.40	0.0111402	0.0015163	0.7184355	0.0170109	-0.0004107
0.45	0.0111343	0.0011809	0.7755223	0.0167801	-0.0003743
0.50	0.0086588	0.0009940	0.8242825	0.0155204	-0.0000980
0.55	0.0058962	0.0010407	0.8590792	0.0135817	0.0001197
0.60	0.0039564	0.0008251	0.8823476	0.0125338	0.0002037
0.65	0.0028839	0.0005341	0.8986379	0.0120984	0.0002143
0.70	0.0023398	0.0004140	0.9112647	0.0114461	0.0002039
0.75	0.0021301	0.0004763	0.9222208	0.0103935	0.0001911
0.80	0.0021935	0.0005645	0.9329514	0.0092023	0.0001796
0.85	0.0025512	0.0005742	0.9448671	0.0082261	0.0001632
0.90	0.0032568	0.0005083	0.9595831	0.0072551	0.0001253
0.95	0.0044141	0.0007505	0.9791329	0.0048537	0.0000355

The posterior distribution of θ_i given (n_i, X_i) is computed using Bayes rule as

$$\hat{g}(\theta|X_i = x_i, n_i) = \frac{g_{\hat{\alpha}}(\theta)\binom{n_i}{x_i}\theta^{x_i}(1-\theta)^{n_i-x_i}}{f_{\hat{\alpha}}(n_i, x_i)}$$

where the denominator is given by

$$f_\alpha(n_i, x_i) = \int_0^1 \binom{n_i}{x_i}\theta^{x_i}(1-\theta)^{n_i-x_i} g_\alpha(\theta)\, d\theta.$$

with the mle $\hat{\alpha}$ in place of α.

Since $g(\theta)$ is discrete, the integrals are mere sums as shown below.

```
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)
}
```

We plot a few posterior distributions.

```
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")
```