MedianandGeoMeantest

Naveed Merchant

October 23, 2018

We're curious to see if our proof has the wrong idea. We examine two functions that are completely different but have the same mean and variance.

We draw from a normal and a normal mixture which have the same variance and same mean, but clearly are not the same function.

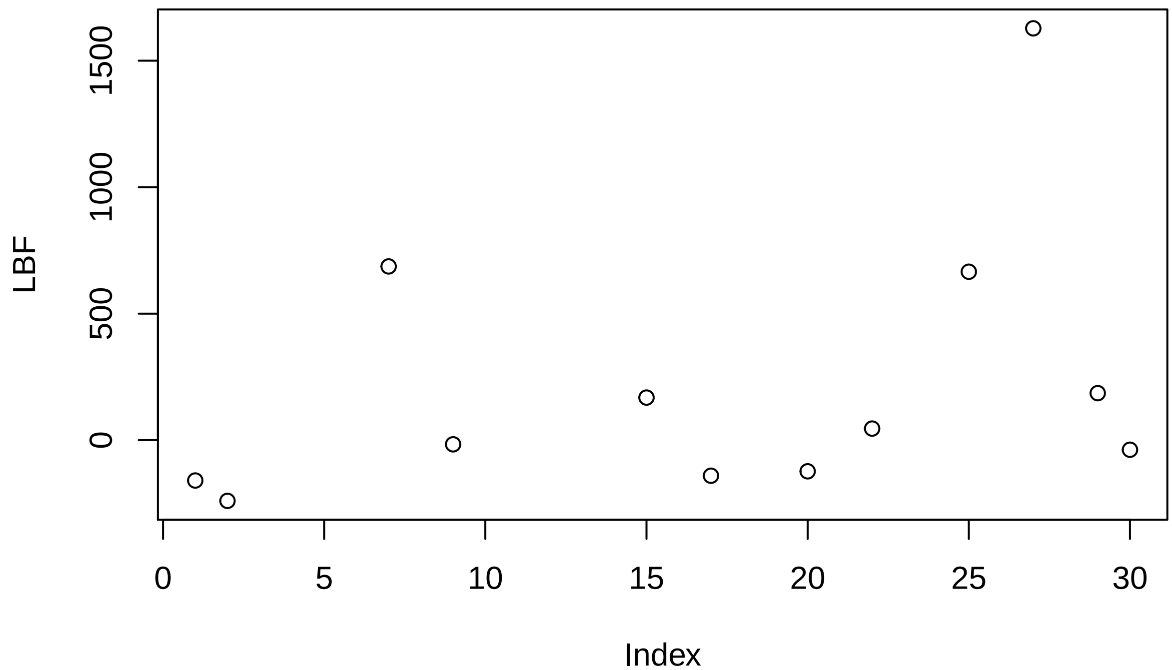
```
gm_mean = function(x, na.rm=TRUE){
  exp(sum(log(x[x > 0]), na.rm=na.rm) / length(x))
}
set.seed(10000)
#install.packages("rmutil")
library(matrixStats)
library(rmutil)
```

```
##
## Attaching package: 'rmutil'

## The following object is masked from 'package:stats':
##
##     nobs

## The following objects are masked from 'package:base':
##
##     as.data.frame, units
```

```
source("MarginalLikIntfunctions.R")
set.seed(10000)
dlength <- 600
LBF <- c()
LBF2 <- c()
dataset1 <- rcauchy(dlength)
dataset2 <- rcauchy(dlength)
for(i in 1:30)
{
  dataset1 <- sample(dataset1)
  dataset2 <- sample(dataset2)
  XT1 <- dataset1[1:(dlength*.3)]
  XV1 <- dataset1[-(1:dlength*.3)]
  XT2 <- dataset2[1:(dlength*.3)]
  XV2 <- dataset2[-(1:dlength*.3)]
  ExpectedKernML <- logmarg.kernMCimport(XT1,XV1,iter = 1, importsize = 100) + logmarg.kernMCimport(XT2
  ExpectedKernML2 <- logmarg.kernMCimport(c(XT1,XT2),c(XV1,XV2),iter = 1,importsize = 100)
  LBF[i] <- ExpectedKernML - ExpectedKernML2
}
plot(LBF)
```




```
LBF
```

```
##  [1] -159.63528 -240.09071        NaN       -Inf        Inf        NaN
##  [7]  686.69067       -Inf  -16.59037       -Inf       -Inf        NaN
## [13]        Inf        Inf  168.23056        NaN -140.69837        NaN
## [19]       -Inf -123.29432        Inf   45.67443        NaN       -Inf
## [25]  665.68649        Inf 1627.94933       -Inf  185.72644  -38.02012
```

```
median(sort(LBF))
```

```
## [1] -27.30525
```

The log BF is interesting at least…

Geometric mean has problems.

Median doesn't.

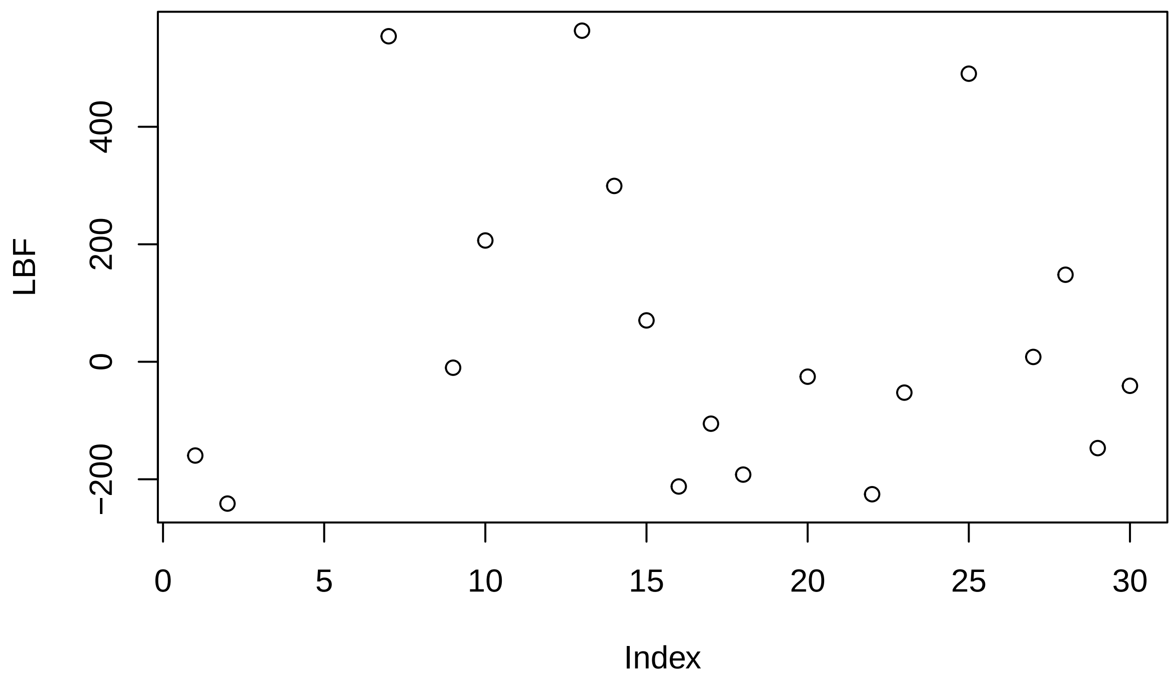
We didn't change sample size for this, all we did was shuffle training and validation.

Also using R's integrate seems to fail and just suggests that the integral is diverging. Importance sampling doesn't do anything with this, so it'll work fine, but its slow.

```
set.seed(10000)
#install.packages("rmutil")
library(rmutil)
source("MarginalLikIntfunctions.R")
set.seed(10000)
dlength <- 600
LBF <- c()
```

```r
LBF2 <- c()
dataset1 <- rcauchy(dlength)
dataset2 <- rcauchy(dlength)
trainprop <- seq(from = .3, to = .8, length = 30)
for(i in 1:30)
{
  dataset1 <- sample(dataset1)
  dataset2 <- sample(dataset2)
  XT1 <- dataset1[1:(dlength*trainprop[i])]
  XV1 <- dataset1[-(1:dlength*trainprop[i])]
  XT2 <- dataset2[1:(dlength*trainprop[i])]
  XV2 <- dataset2[-(1:dlength*trainprop[i])]
  ExpectedKernML <- logmarg.kernMCimport(XT1,XV1,iter = 1, importsize = 100) + logmarg.kernMCimport(XT2
  ExpectedKernML2 <- logmarg.kernMCimport(c(XT1,XT2),c(XV1,XV2),iter = 1,importsize = 100)
  LBF[i] <- ExpectedKernML - ExpectedKernML2
}

plot(LBF)
```




```r
LBF
```

```
##  [1] -159.635283 -241.402547         NaN        -Inf         Inf
##  [6]         NaN  554.108342        -Inf  -10.148233  206.458301
## [11]        -Inf         NaN  563.599046  299.326822   70.387064
## [16] -212.309634 -105.413988 -192.082173        -Inf  -25.365914
## [21]         Inf -225.500411  -52.542851        -Inf  490.376438
```

```
## [26]          Inf    8.219317  148.179788 -146.961187  -40.923881
```

```r
median(sort(LBF))
```

```
## [1] -40.92388
```

This is still with a pretty pathological distribution (none of its moments are defined!).

Things are behaving as expected again suprisingly? The median LBF is essentially 0...

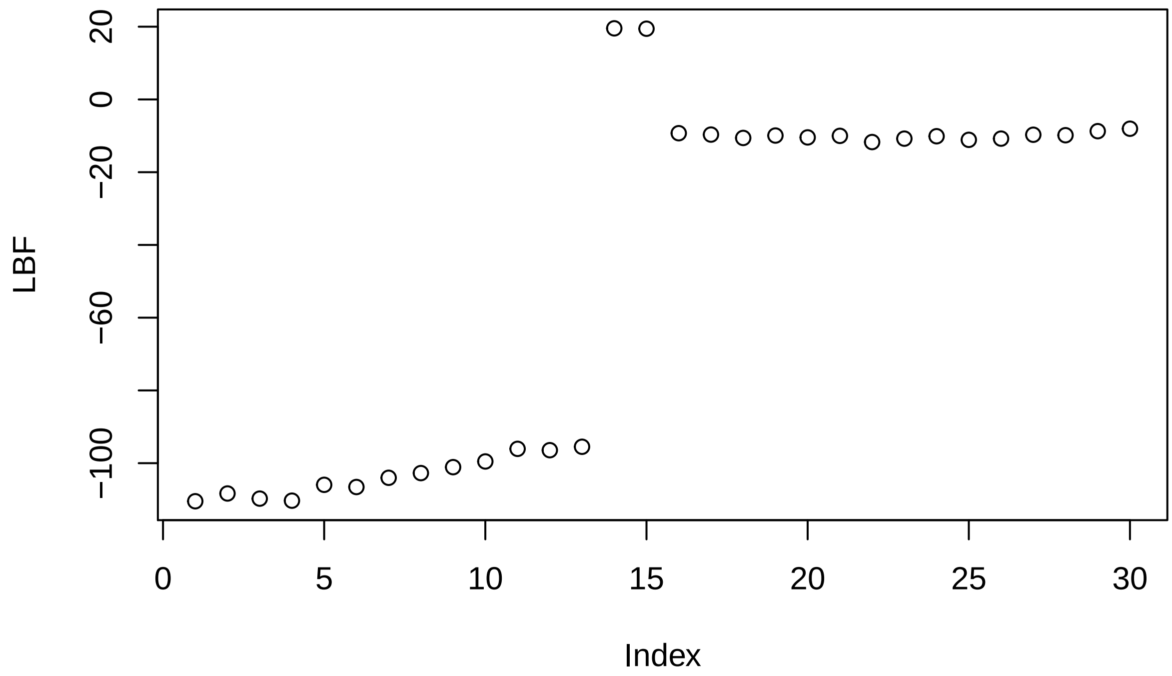
Again computing geometric mean is essentially impossible.

I'm not sure where the NaN is coming from?

What about the pareto?

```r
#Finite mean but infinite variance

set.seed(10000)
dlength <- 300
dataset1 <- rpareto(dlength, m=2.5, s=2.5)
dataset2 <- rpareto(dlength, m=2.5, s=2.5)
LBF <- c()
trainprop <- seq(from = .3, to = .8, length = 30)
for(i in 1:30)
{
  XT1 <- dataset1[1:(dlength*trainprop[i])]
  XV1 <- dataset1[-(1:(dlength*trainprop[i]))]
  XT2 <- dataset2[1:((dlength*trainprop[i]))]
  XV2 <- dataset2[-(1:(dlength*trainprop[i]))]
  ExpectedKernML <- logmarg.kern(XT1,XV1)[[2]] + logmarg.kern(XT2,XV2)[[2]]
  ExpectedKernML2 <- logmarg.kern(c(XT1,XT2),c(XV1,XV2))[[2]]
  LBF[i] <- ExpectedKernML - ExpectedKernML2
}
plot(LBF)
```




```
mean(LBF)
```

```
## [1] -48.61682
```

```
median(LBF)
```

```
## [1] -10.94106
```

Taking the mean of the logBF is the same as examining the log of the geometric mean of the BF...

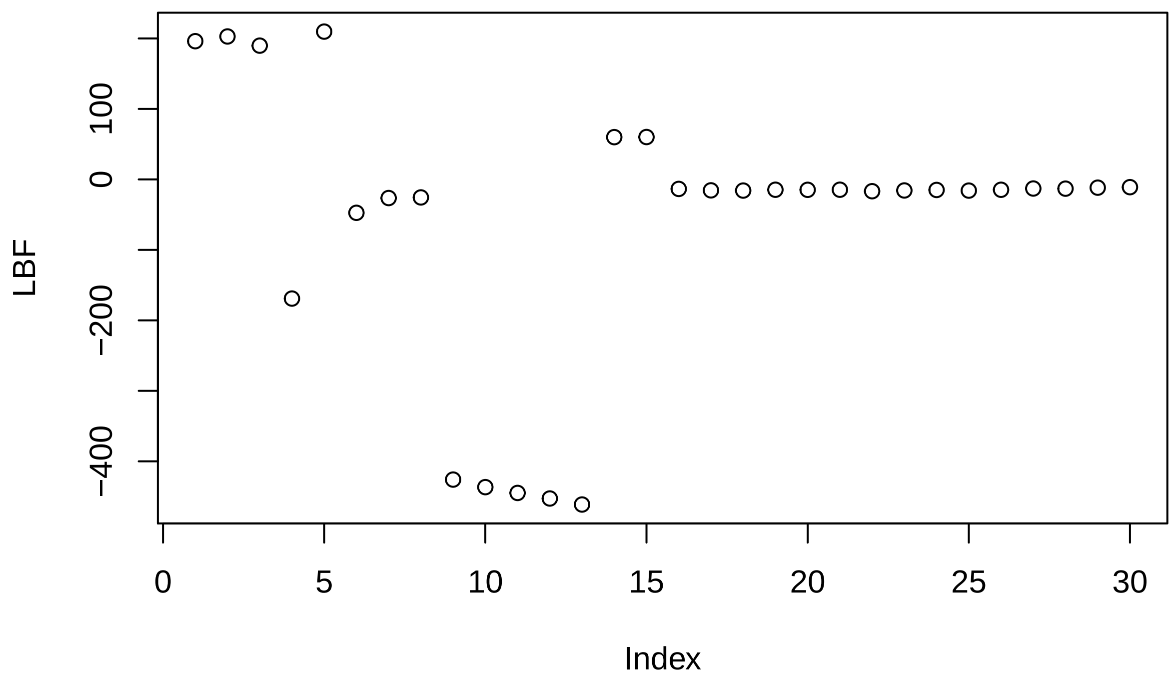
What's suprising is that the logBF seems to be dependent on how many samples are being chosen for training for this problem?

We let more moments be 0 and see what happens. We used the integrate function this time as we didn't have computational issues...

```
#Finite mean and variance, but infinite higher moments.

set.seed(10000)
dlength <- 300
dataset1 <- rpareto(dlength, m=1.5, s=1.5)
dataset2 <- rpareto(dlength, m=1.5, s=1.5)
LBF <- c()
trainprop <- seq(from = .3, to = .8, length = 30)
for(i in 1:30)
{
  XT1 <- dataset1[1:(dlength*trainprop[i])]
  XV1 <- dataset1[-(1:(dlength*trainprop[i]))]
  XT2 <- dataset2[1:((dlength*trainprop[i]))]
```

```
  XV2 <- dataset2[-(1:(dlength*trainprop[i]))]
  ExpectedKernML <- logmarg.kern(XT1,XV1)[[2]] + logmarg.kern(XT2,XV2)[[2]]
  ExpectedKernML2 <- logmarg.kern(c(XT1,XT2),c(XV1,XV2))[[2]]
  LBF[i] <- ExpectedKernML - ExpectedKernML2
}
plot(LBF)
```




```
LBF
```

```
##  [1]   196.12228   202.81022   189.77193 -169.08943   209.67786   -47.41013
##  [7]   -26.43591   -25.57979 -425.93403 -436.63184 -444.87965 -452.74082
## [13] -461.28962    59.97783    60.11329  -13.51910  -15.54137  -15.84968
## [19]   -14.66632   -14.79153  -14.64685  -16.76680  -15.70944  -14.96856
## [25]   -15.88894   -14.71320  -12.94612  -13.14477  -11.72664  -10.99363
```

```
log(gm_mean(exp(LBF)))
```

```
## [1] -59.57969
```

```
mean(LBF)
```

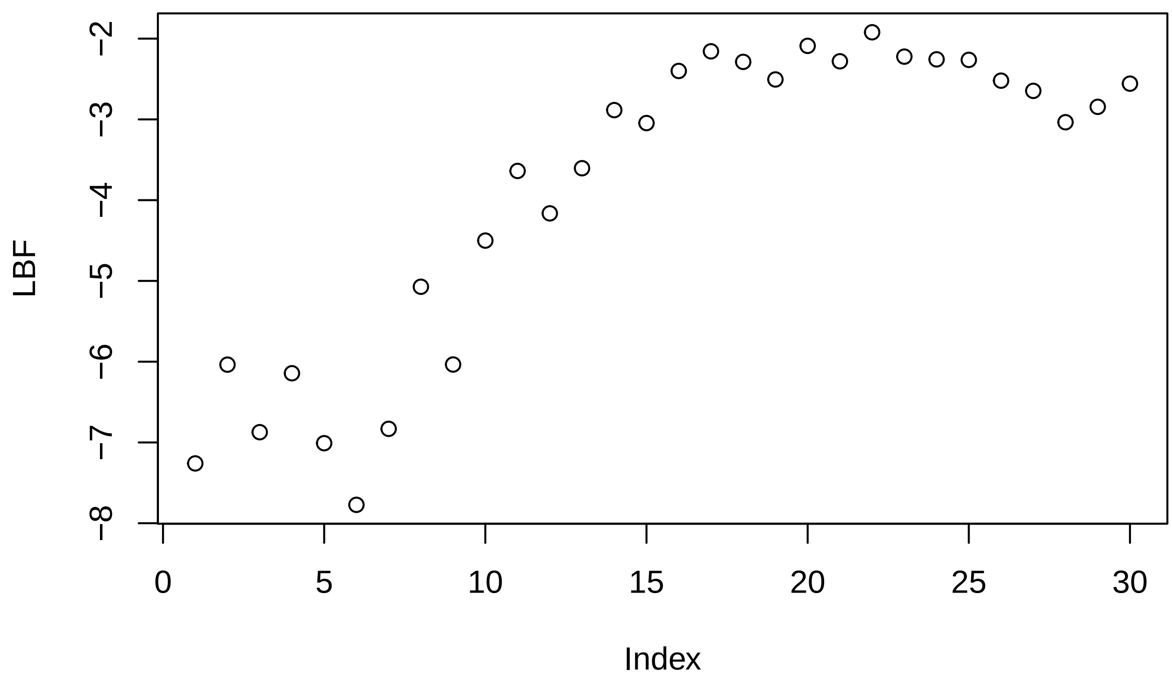
```
## [1] -59.57969
```

```
median(LBF)
```

```
## [1] -14.88004
```

This favors the null hypothesis for most choices of training size, but not all. Lets check the normal distribution.

```r
set.seed(10000)
dlength <- 300
dataset1 <- rnorm(dlength)
dataset2 <- rnorm(dlength)
LBF <- c()
trainprop <- seq(from = .3, to = .8, length = 30)
for(i in 1:30)
{
  XT1 <- dataset1[1:(dlength*trainprop[i])]
  XV1 <- dataset1[-(1:(dlength*trainprop[i]))]
  XT2 <- dataset2[1:((dlength*trainprop[i]))]
  XV2 <- dataset2[-(1:(dlength*trainprop[i]))]
  ExpectedKernML <- logmarg.kern(XT1,XV1)[[2]] + logmarg.kern(XT2,XV2)[[2]]
  ExpectedKernML2 <- logmarg.kern(c(XT1,XT2),c(XV1,XV2))[[2]]
  LBF[i] <- ExpectedKernML - ExpectedKernML2
}
plot(LBF)
```




```r
LBF
```

```
##  [1] -7.259560 -6.036764 -6.872874 -6.143237 -7.009277 -7.772346 -6.831376
##  [8] -5.072312 -6.035072 -4.500693 -3.638267 -4.163574 -3.604568 -2.885059
## [15] -3.045327 -2.400169 -2.157126 -2.287655 -2.505402 -2.089583 -2.280554
## [22] -1.921290 -2.222880 -2.256415 -2.263121 -2.520348 -2.646635 -3.035001
## [29] -2.844375 -2.557447
```

```r
log(gm_mean(exp(LBF)))
```

```
## [1] -3.895277
```

```r
mean(LBF)
```

```
## [1] -3.895277
```

```r
median(LBF)
```

```
## [1] -2.96003
```

I'm not sure if there's a trend I should be aware of?

If things are normal things are fine.