Hasher-Lind Non-Normal

Ryan McClarren

November 3, 2016

The function that we will use as our test of Hasofer-Lind is

$$Z = -\beta_m(\log p + 1 - f)$$

. Let’s say that

```
\beta_\mathrm{m} \sim \mathrm{N}(.004,.0004),\\
p &\sim \mathrm{B}(36.37,21.3602),\\
f &\sim \mathrm{B}(5.06,.322979).
```

The failure surface for this function (i.e., where $Z = 0$) occurs at

$$x_2 = x_1 + 0.1\cos(4x_1)$$

.

The HL procedure begins at the mean point.

```
alphap = 36.37
betap = 21.3602
alphaf = 5.06
betaf = 0.322979

muB = qnorm(p=0.5, mean = 0.004, sd = 0.0004)
mup = qbeta(p=0.5, shape1 = alphap, shape2 = betap)
muf = qbeta(p=0.5, shape1 = alphaf, , shape2 = betaf)
```

Next we determine the sigmas, normalize the x_i and find a search direction.

```r
meanBeta = function(alpha, beta) alpha/(beta + alpha)

iteration1 = data.frame(betam = muB, p = meanBeta(alphap, betap),
                        f = meanBeta(alphaf, betaf))
sigmabeta = 0.0004
sigmap = (iteration1$p - mup)/qnorm(pbeta(iteration1$p, shape1 = alphap, shape2 = betap))
sigmaf = (iteration1$f - muf)/qnorm(pbeta(iteration1$f, shape1 = alphaf, shape2 = betaf))

iteration1$betamprime = (iteration1$betam-muB)/sigmabeta
iteration1$pprime = (iteration1$p-mup)/sigmap
iteration1$fprime = (iteration1$f-muf)/sigmaf

dg = c(dZdbeta(iteration1), dZdp(iteration1), dZdf(iteration1))
dgprime = dg*c(sigmabeta, sigmap, sigmaf)
absgsquared = sum(dgprime*dgprime)
xprime = 1/absgsquared*(sum(dgprime*c(iteration1$betamprime,iteration1$pprime, iteration1$fprime))

                        - Z(iteration1))*dgprime
iteration2 = data.frame(betam = xprime[1]*sigmabeta+muB, p  = xprime[2]*sigmap+mup,
                        f = xprime[3]*sigmaf+muf)

beta = sqrt(sum(xprime^2))
absg = abs(Z(iteration2))

print(paste("Iteration:",1,"beta = ",beta,"|Z| = ",absg))
```

```
## [1] "Iteration: 1 beta =  3.29127904099938 |Z| =  0.000242939922615366"
```

```r
iteration2$lab = "1"
allits = iteration2
```

After one iteration $\beta = 3.291279$, and $|Z| = 2.4293992 \times 10^{-4}$.

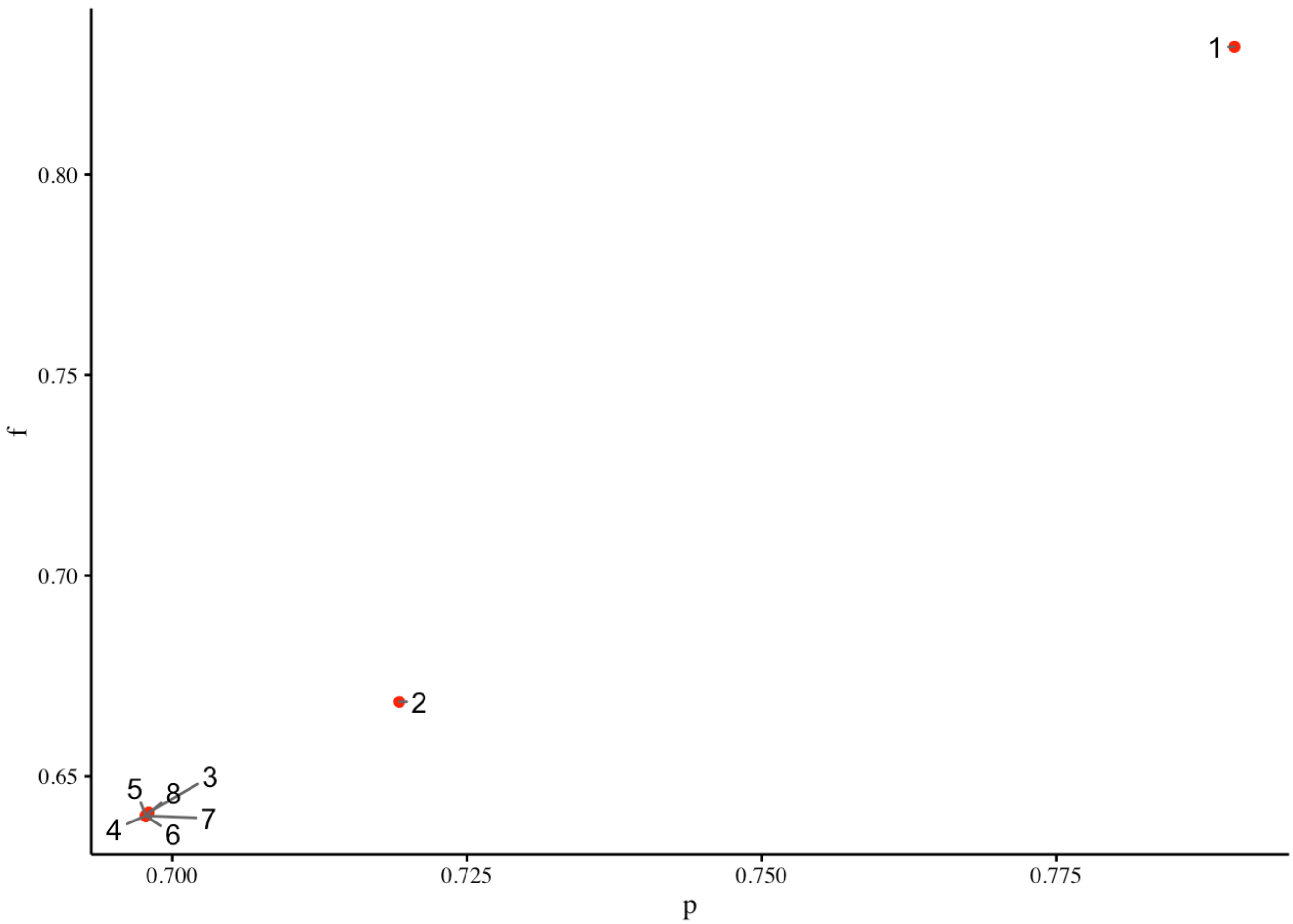
Now let’s turn this into an procedure that runs until either $\Delta\beta$ or $|Z|$ is less than 10^{-6}.

```
delta = 1e-6
converged = 0
it = 2
beta_old = beta
old_iteration = iteration2
while (!(converged)){
sigmabeta = 0.0004
sigmap = (old_iteration$p - mup)/qnorm(pbeta(old_iteration$p, shape1 = 36.37, shape2 = 21.3602))
sigmaf = (old_iteration$f - muf)/qnorm(pbeta(old_iteration$f, shape1 = 5.06, shape2 = .322979))

old_iteration$betamprime = (old_iteration$betam-muB)/sigmabeta
old_iteration$pprime = (old_iteration$p-mup)/sigmap
old_iteration$fprime = (old_iteration$f-muf)/sigmaf

dg = c(dZdbeta(old_iteration), dZdp(old_iteration), dZdf(old_iteration))
dgprime = dg*c(sigmabeta, sigmap, sigmaf)
absgsquared = sum(dgprime*dgprime)
xprime = 1/absgsquared*(sum(dgprime*c(old_iteration$betamprime,old_iteration$pprime, old_iterati
on$fprime))
                        - Z(old_iteration))*dgprime
new_iteration = data.frame(betam = xprime[1]*sigmabeta+muB, p  = xprime[2]*sigmap+mup,
                        f = xprime[3]*sigmaf+muf)

beta = sqrt(sum(xprime^2))
absg = abs(Z(new_iteration))

print(paste("Iteration:",it,"beta = ",beta,"|Z| = ",absg))

it = it + 1

if ( (abs(beta_old-beta) < delta) & (absg < delta))
  converged = 1
if (it > 10)
  converged = 1
beta_old = beta
old_iteration = new_iteration
new_iteration$lab = it-1
allits = rbind(allits,new_iteration)
}
```

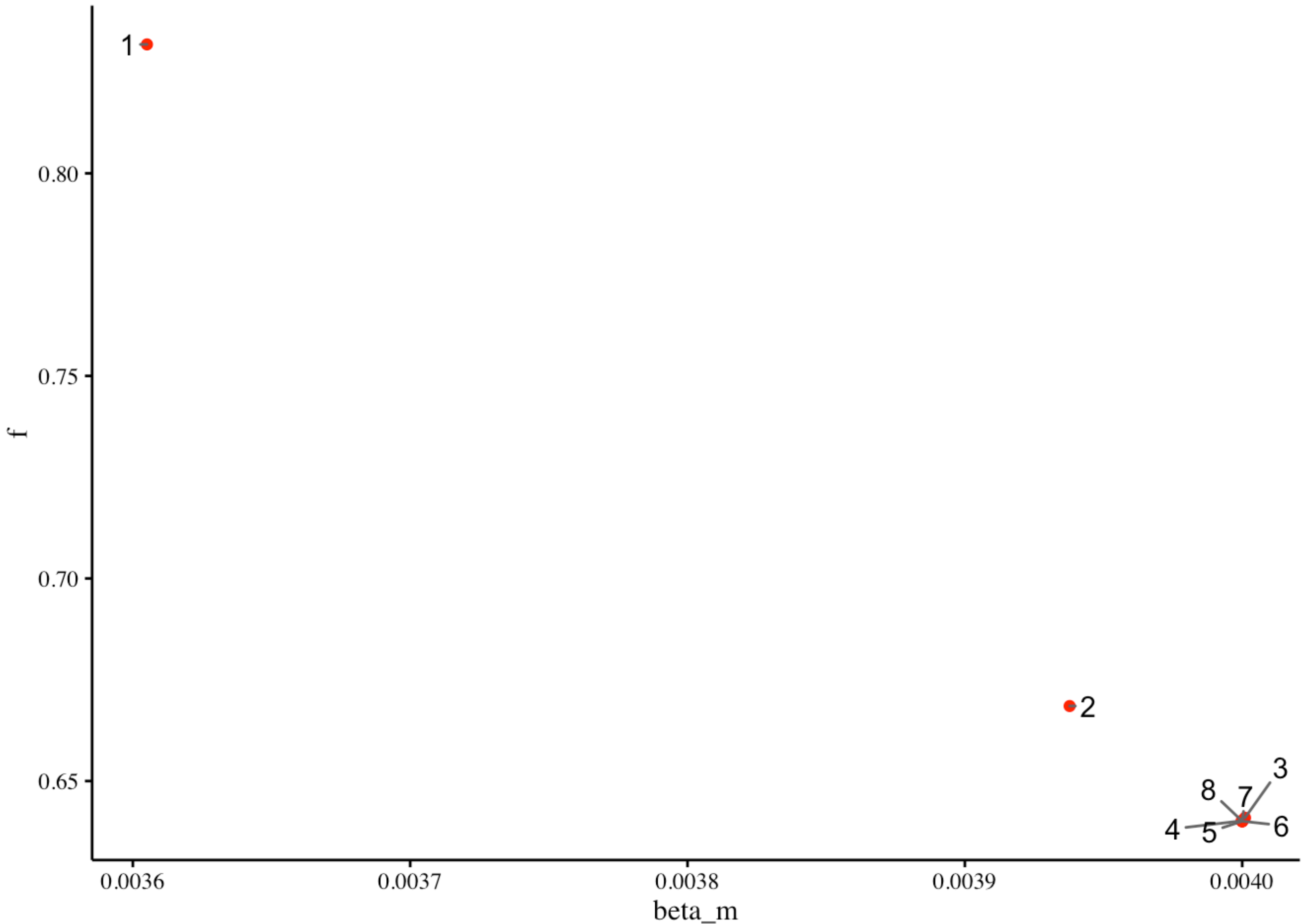
```
## [1] "Iteration: 2 beta =  2.97423430342776 |Z| =  7.62517751970462e-06"
## [1] "Iteration: 3 beta =  2.35802911045196 |Z| =  1.90718926935838e-06"
## [1] "Iteration: 4 beta =  2.29874007184125 |Z| =  7.48846899465322e-10"
## [1] "Iteration: 5 beta =  2.29727313582944 |Z| =  1.25680076830453e-11"
## [1] "Iteration: 6 beta =  2.29710928664281 |Z| =  5.79358782950245e-15"
## [1] "Iteration: 7 beta =  2.29710577663873 |Z| =  2.22044604924992e-18"
## [1] "Iteration: 8 beta =  2.29710570180354 |Z| =  4.44089209850063e-19"
```

The iteration procedure looks like this on a graph.

```
ggplot(allits, aes(x=p, y=f)) + geom_point(color="red") + geom_text_repel(data=allits, aes(label=l
b)) +theme_tufte() + theme(axis.line = element_line(color="red"),axis.line.x = element_line(size
 = .5, colour = "black"), axis.line.y=element_line(size = .5, colour = "black"))
```




```
ggplot(allits, aes(x=betam, y=f)) + geom_point(color="red") + geom_text_repel(data=allits, aes(l
abel=lab)) + scale_x_continuous(name="beta_m")+ theme_tufte() + theme(axis.line = element_line(c
olor="red"),axis.line.x = element_line(size = .5, colour = "black"), axis.line.y=element_line(si
ze = .5, colour = "black"))
```



The probability of failure is $1 - \Phi(\beta) = 0.0108064$. Let's check that with Monte Carlo.

```
N = 1e6
df <- data.frame(betam = rnorm(N, mean = 0.004, sd = 0.0004),
                 p = rbeta(N, shape1 = alphap, shape2 = betap),
                 f = rbeta(N, shape1 = alphaf, shape2 = betaf) )
df$Z = Z(df)
pfail = length(df$Z[df$Z<0])/N
```

The actual probability of failure is 0.011106. The Hasofer-Lind approximation is off by 2.6979084%.