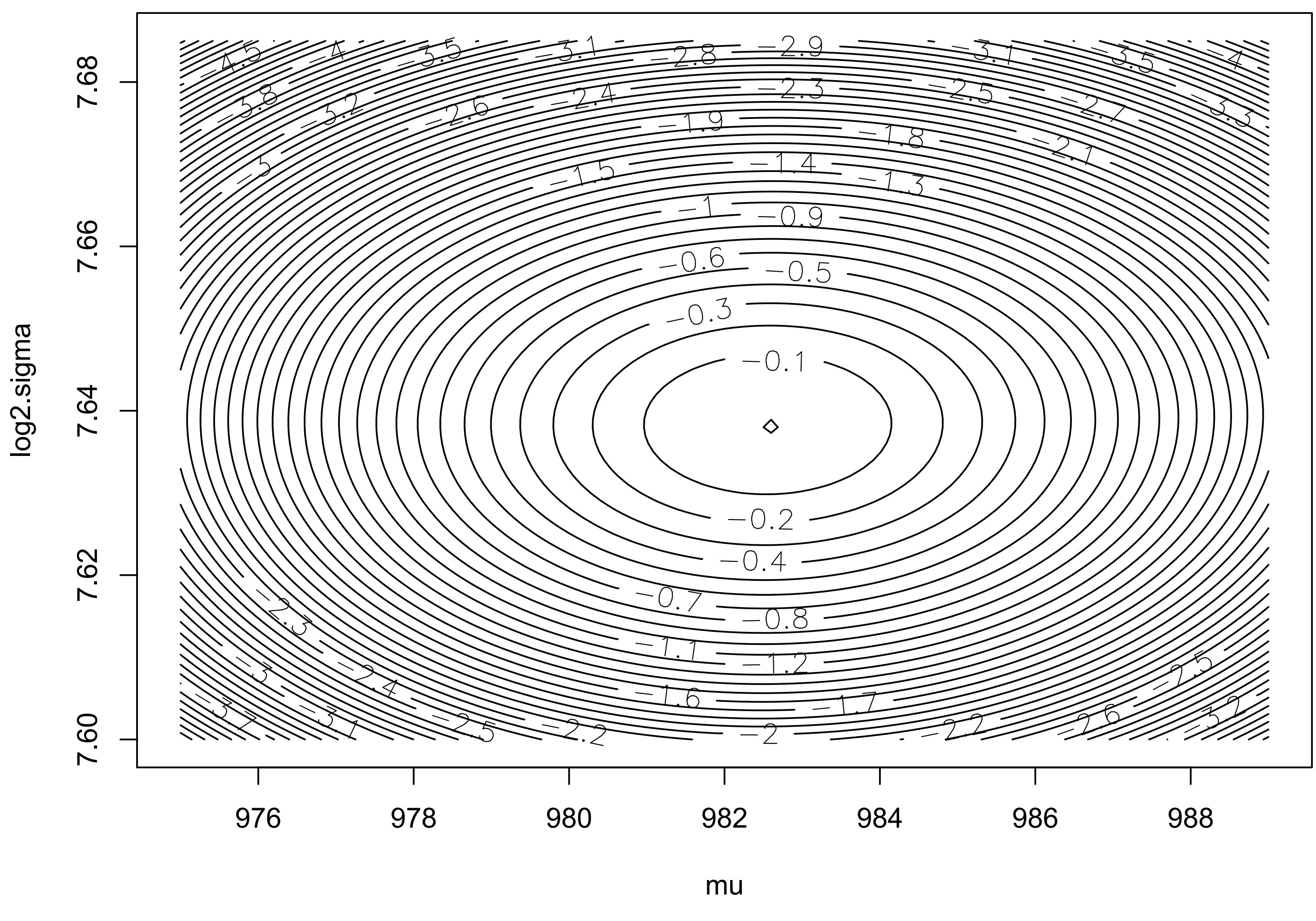
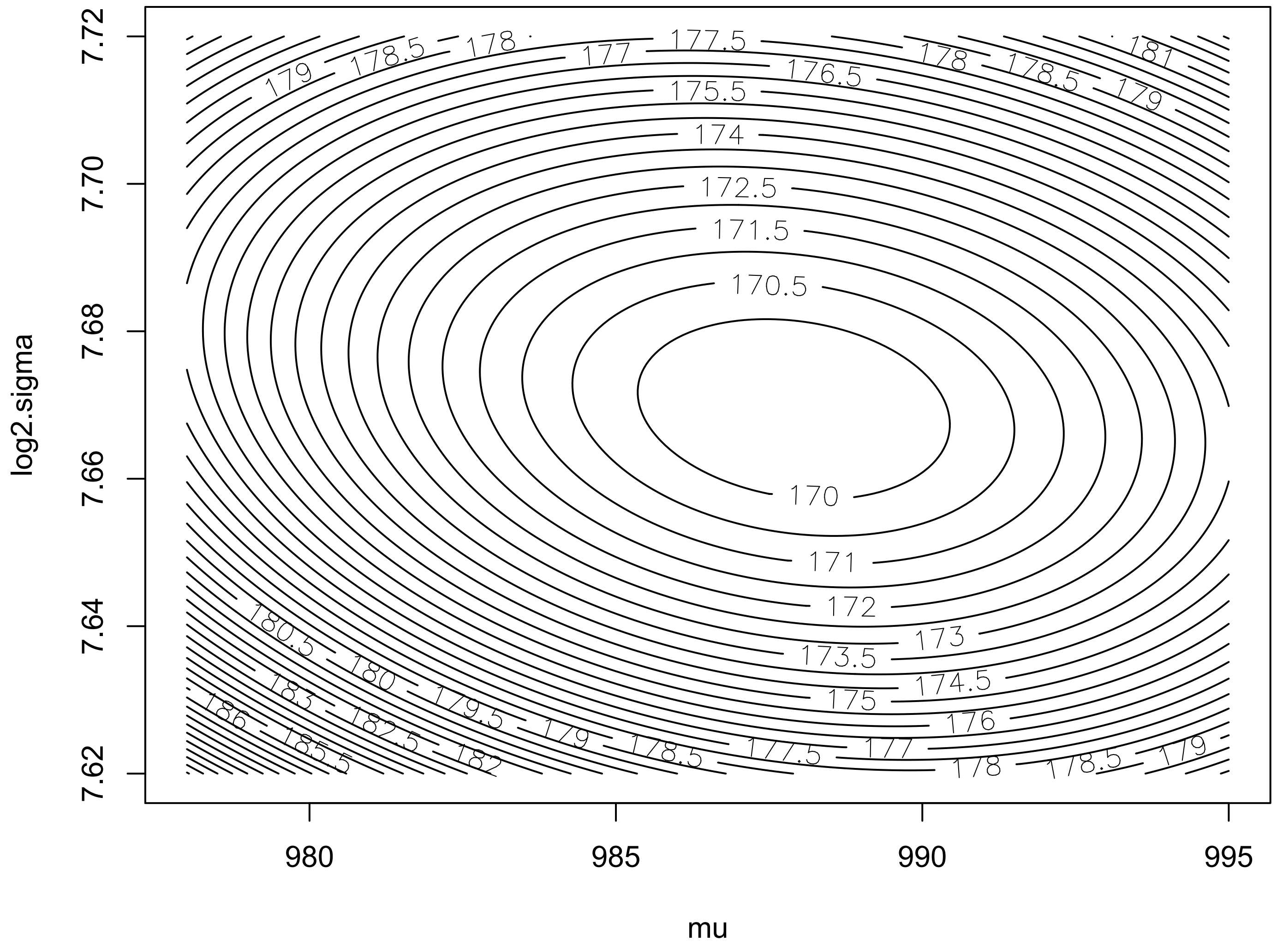
```
> f=scan()
1: 22
2: 43
3: 164
4: 284
5: 598
6: 645
7: 683
8: 396
9: 132
10: 120
11: 120
12:
Read 11 items
> #
>
> f;
 [1]  22  43 164 284 598 645 683 396 132 120 120
> n=sum(f) ; n = sum(f) ; n;   # check
[1] 3207
>
> lower = c(0,seq(500,1400,100)) ; lower
 [1]    0  500  600  700  800  900 1000 1100 1200 1300 1400
> upper = c(seq(500,1400,100),3000) ; upper
 [1]  500  600  700  800  900 1000 1100 1200 1300 1400 3000
>
>
> midpoints = c(475, seq(550,1350,100), 1500)
>
> ave.speed = sum(f*midpoints) / n;  ave.speed
[1] 983.5672
>
> d.f. = n-1
>
> var.speed = sum(f*(midpoints-ave.speed)^2)/d.f. ; var.speed
[1] 40667.57
>
> var.speed  = var.speed  - 100^2/12 ; var.speed ;
[1] 39834.24
>
> sd = sqrt(var.speed) ; sd
[1] 199.5852
>
> se.m = sd/sqrt(n); round(se.m,1)
[1] 3.5
>
> # see notes..
>
> # sd^2 ~ (1/d.f.) * chi.sq.d.f. * sigma^2 ;
>
> # var(sd^2)  = (1/d.f.)^2 * 2*d.f. * sigma^4  ;
> # var(sd)  = var( [sd^2]^(1/2) ) *   [ (d sqrt x / d x )  ]^2;
> # (d sqrt / d x ) = (1/2) x ^(-1/2) ;  (d sqrt / d x )^2 = (1/4) x^(-1)
> # var(sd)  =  (1/d.f.)^2 * 2*d.f. * sigma^4    *  (1/4) * (sigma^2)^(-1)
> #          =  (1/[2d.f.]) *  sigma^2
> # SE(sd)  =  sigma/sqrt(2.d.f.)    ; CV(sd) = 100 / sqrt( 2 d.f.)
>
> se.sd  =  sd / sqrt(2*d.f.) ; se.sd
[1] 2.492479
>
> # limits on sigma
>
> sqrt( var.speed * (1/d.f.) * qchisq(c(0.025,0.975),d.f.)  )
[1] 194.6994 204.4693
>
>
> ###### ML estimates of mu, sigma  # working on log2(sigma) scale..
>
> log.lik = function(mu,log2.sigma)
```

```
+   sum(f*log( pnorm(upper,mu,2^log2.sigma) - pnorm(lower,mu,2^log2.sigma)   ))
>
> log.lik(1000,7)
[1] -7476.13
>
> # this wont work in outer function (since lower upper and f are themselves vectors),
> # so 'force' the issue ...
>
> log.lik.vector = Vectorize( log.lik, c("mu", "log2.sigma") )
>
> mu = seq(900,1100,1) ; log2.sigma= seq(6,9,0.1)
>
> mu = seq(975,989,0.1) ; log2.sigma= seq(7.60,7.685,0.001)
>
> log.lik.values = outer(mu, log2.sigma, log.lik.vector)
>
> max.log.lik.values  = max(log.lik.values)
>
> i.max = apply(log.lik.values==max.log.lik.values,1,sum)
> j.max = apply(log.lik.values==max.log.lik.values,2,sum)
>
> mle = c(mu[i.max],round(2^log2.sigma[j.max],1)) ; mle
[1] 975 194
>
> rel.log.lik.values = log.lik.values  - max.log.lik.values
>
> contour( mu, log2.sigma, rel.log.lik.values, nlevels=40,
+  xlab="mu", ylab="log2.sigma", labcex=1.1)
> chi.sq = function(mu,log2.sigma) {
+       fitted.f = n * (pnorm(upper,mu,2^log2.sigma) - pnorm(lower,mu,2^log2.sigma))
+       return( sum((f-fitted.f)^2/fitted.f) )
+       }
>
> chi.sq(1000,7)
[1] 9840.334
>
> chi.sq.vector = Vectorize( chi.sq, c("mu", "log2.sigma") )
>
> mu = seq(900,1100,1) ; log2.sigma= seq(6,9,0.1)
>
> mu = seq(978,995,0.1) ; log2.sigma= seq(7.62,7.72,0.0005)
>
> chi.sq.values = outer(mu, log2.sigma, chi.sq.vector)
>
> min.chi.sq.values  = min(chi.sq.values)
>
> i.max = apply(chi.sq.values==min.chi.sq.values,1,sum)
> j.max = apply(chi.sq.values==min.chi.sq.values,2,sum)
>
> min.ch.sq.e = c(mu[i.max],round(2^log2.sigma[j.max],1)) ; min.ch.sq.e
[1] 978.0 196.7
>
> contour( mu, log2.sigma, chi.sq.values, nlevels=40,
+  xlab="mu", ylab="log2.sigma", labcex=1.0)
>
```



log2.sigma
mu
7.60
7.62
7.64
7.66
7.68
976
978
980
982
984
986
988
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
−1
−1.1
−1.2
−1.3
−1.4
−1.5
−1.6
−1.7
−1.8
−1.9
−2
−2.1
−2.2
−2.3
−2.4
−2.5
−2.6
−2.7
−2.8
−2.9
−3
−3.1
−3.2
−3.3
−3.5
−3.7
−3.8
−4
−4.5
−5



log2.sigma
mu
7.62
7.64
7.66
7.68
7.70
7.72
980
985
990
995
170
170.5
171
171.5
172
172.5
173
173.5
174
174.5
175
175.5
176
176.5
177
177.5
178
178.5
179
179.5
180
180.5
181
182
182.5
183
185.5
186