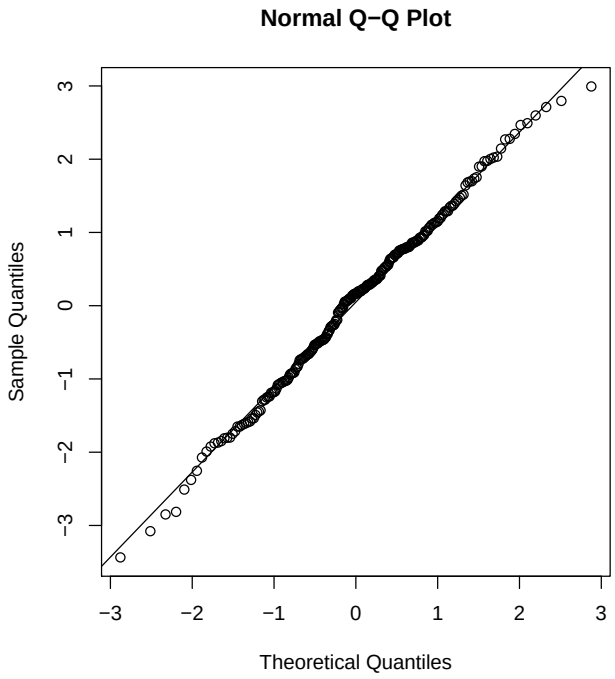
Statistics II: Estimating simple parametric models

Richard Gill*

March 6, 2009

First of all we look at a normal plot of a sample from the t distribution with 9 degrees of freedom (and before that, we set the seed to my birthday, so that the results will be perfectly reproducible)

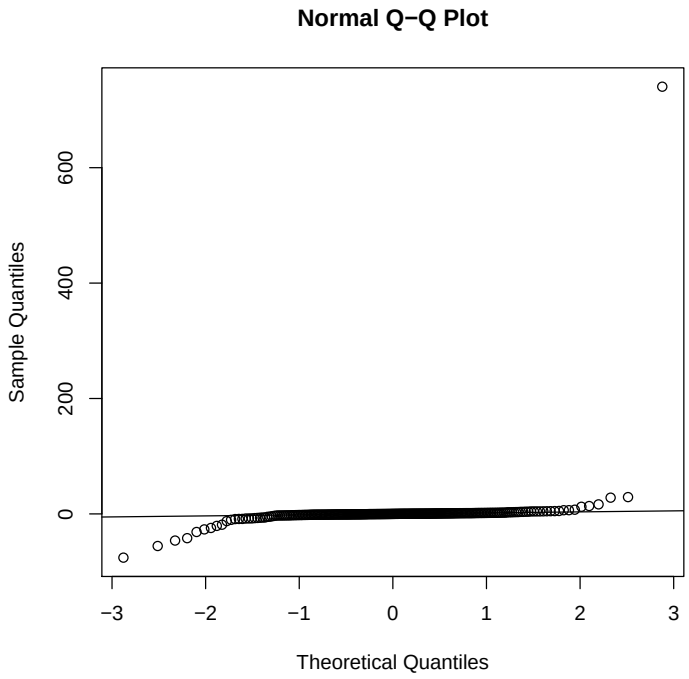
```
> set.seed(11091951)
> xt <- rt(250, df = 9)
> xc <- rcauchy(250)
> x <- xt
> par(pty = "s")
> qqnorm(x)
> qqline(x)
```



*http://www.math.leidenuniv.nl/∼gill/teaching/statistics

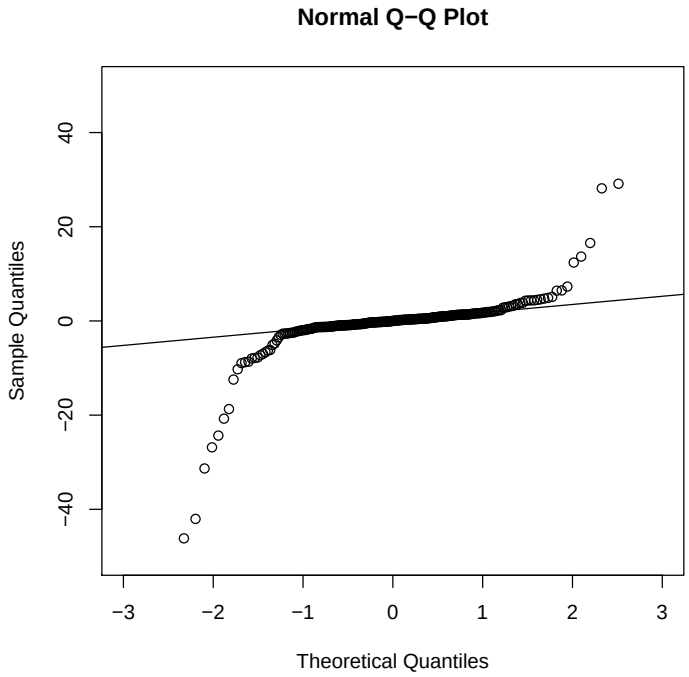
And now the same for a sample from the Cauchy

```
> x <- xc
> qqnorm(x)
> qqline(x)
```



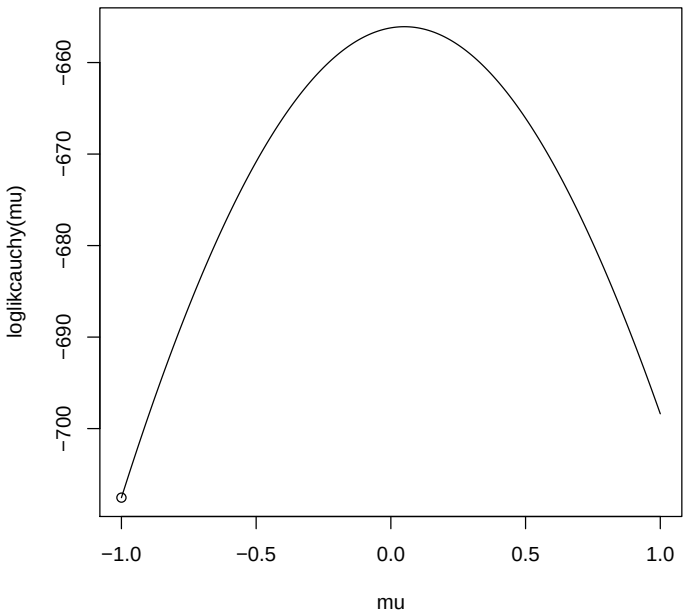
That was a strange graph! Let's zoom in a bit:

```
> qqnorm(x, xlim = c(-3, 3), ylim = c(-50, 50))
> qqline(x)
```



Now we'll fit the Cauchy location model to our data by maximum likelihood

```
> library(MASS)
> fitdistr(x, "cauchy", start = list(location = 0, scale = 1))

    location      scale
  0.05083830   1.12719758
 (0.10492878) (0.09716141)

> logLik(fitdistr(x, "cauchy", start = list(location = 0, scale = 1)))

'log Lik.' -655.1167 (df=2)

> logLik(fitdistr(x, "cauchy", start = list(location = 0), scale = 1))

'log Lik.' -656.0805 (df=1)

> logdcauchy <- function(mu, x) {
+     dcauchy(x, scale = 1, location = mu, log = TRUE)
+ }
> loglikcauchy <- function(mu) {
+     apply(outer(mu, x, logdcauchy), 1, sum)
+ }
> loglikcauchy(0)

[1] -656.2065

> optimize(loglikcauchy, lower = -1, upper = 1)

$minimum
[1] -0.999959

$objective
[1] -707.542

> result <- optimize(loglikcauchy, lower = -1, upper = 1)
> mu <- seq(from = -1, to = 1, by = 0.01)
> plot(mu, loglikcauchy(mu), type = "l")
> optimize(loglikcauchy, lower = -1, upper = 1)

$minimum
[1] -0.999959

$objective
[1] -707.542

> points(result[[1]], result[[2]])
```



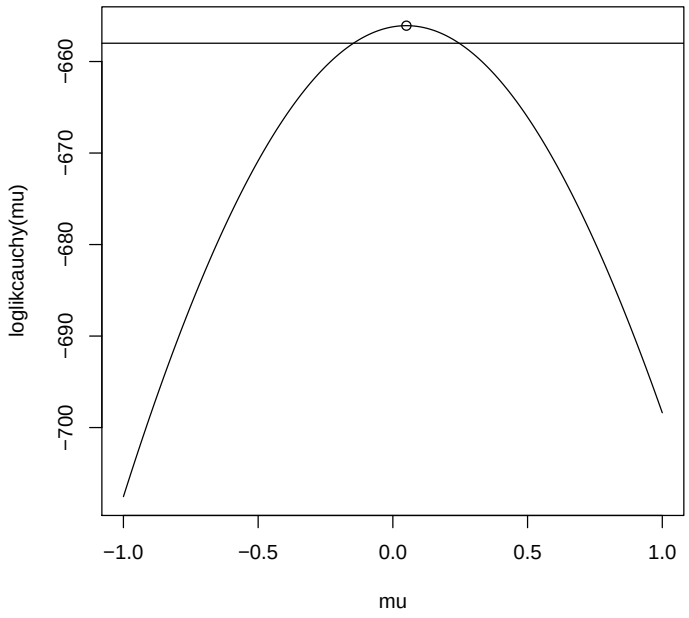
It turned out that in "optimize", we were minimizing, not maximizing!

The second call of logLik(fitdist) (with just one parameter) produced a load of errors. I'm not sure why. Here are the error messages, for completeness:

```
Warning messages:
1: In optim(x = c(-1.33917884613114, 4.48099859454494, 0.105654587255878,  :
  one-diml optimization by Nelder-Mead is unreliable: use optimize
2: In optim(x = c(-1.33917884613114, 4.48099859454494, 0.105654587255878,  :
  one-diml optimization by Nelder-Mead is unreliable: use optimize
3: In optim(x = c(-1.33917884613114, 4.48099859454494, 0.105654587255878,  :
  one-diml optimization by Nelder-Mead is unreliable: use optimize
```

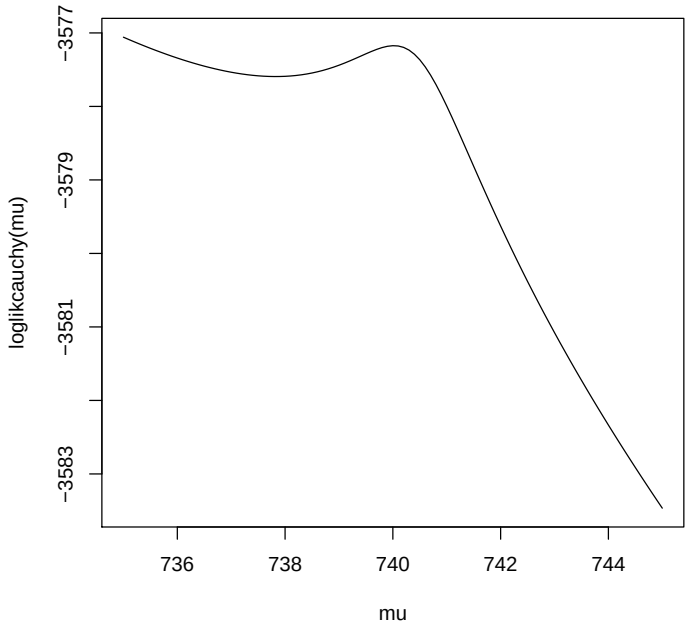
Now we'll fix the optimization problem. And show the cut-off of an approximate 95% confidence interval for the location parameter.

```
> negloglikcauchy <- function(mu) {
+     -apply(outer(mu, x, logdcauchy), 1, sum)
+ }
> optimize(negloglikcauchy, lower = -1, upper = 1)

$minimum
[1] 0.05021444

$objective
[1] 656.0805

> result <- optimize(negloglikcauchy, lower = -1, upper = 1)
> plot(mu, loglikcauchy(mu), type = "l")
> points(result[[1]], -result[[2]])
> abline(h = -result[[2]] - (qchisq(0.95, 1)/2))
```



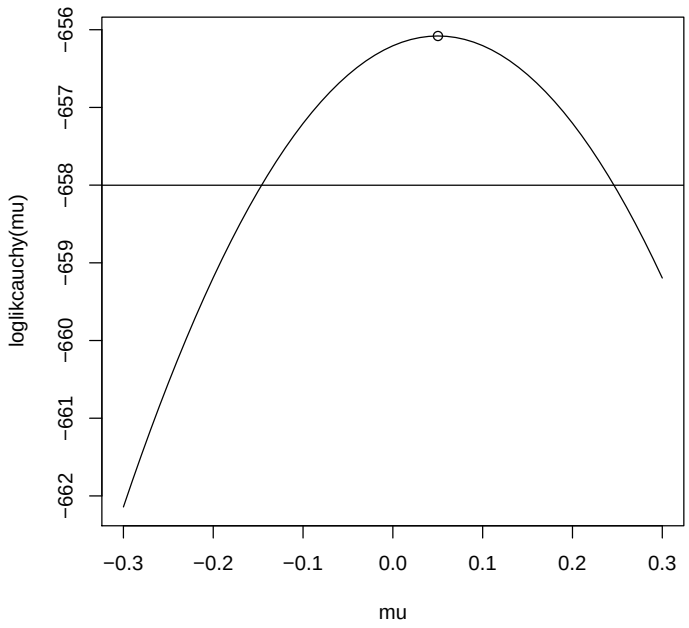
According to Reeds (1986), the number of inconsistent local maxima of the log likelihood for the Cauchy location model is asymptotically Poisson $(1/\pi)$ distributed. We certainly caught one with this sample:

```
> max(x)

[1] 740.4062

> mu <- seq(from = 735, to = 745, by = 0.1)
> plot(mu, loglikcauchy(mu), type = "l")
```



Now we go back to the real mle, zooming in on where the action is.

```
> mu <- seq(from = -0.3, to = 0.3, by = 0.01)
> optimize(negloglikcauchy, lower = -0.3, upper = 0.3)

$minimum
[1] 0.05021571

$objective
[1] 656.0805

> plot(mu, loglikcauchy(mu), type = "l")
> result <- optimize(negloglikcauchy, lower = -0.3, upper = 0.3)
> points(result[[1]], -result[[2]])
> abline(h = -result[[2]] - (qchisq(0.95, 1)/2))
```

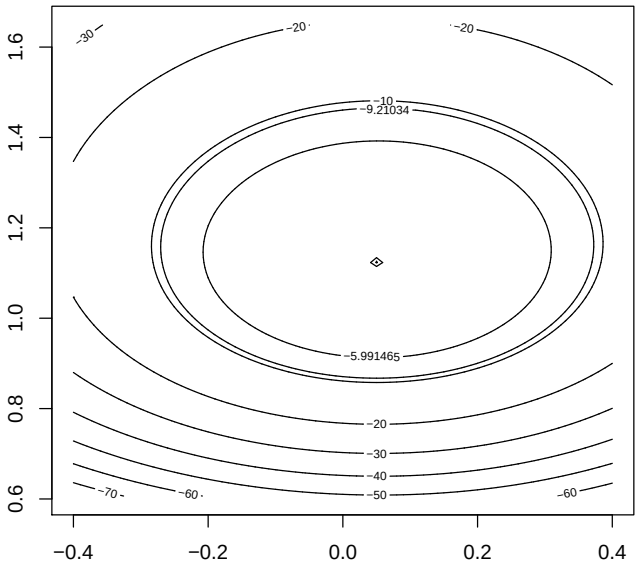


Now we'll do contour plots and wire frame plots of the log likelihood surface. I'll subtract off the maximum and multiply by two, so that the vertical scale becomes more meaningful.

```
> mu <- seq(from = -0.4, to = 0.4, by = 0.01)
> sigma <- exp(seq(from = -0.5, to = 0.5, length = 100))
> loglikcauchy <- function(mu, sigma) {
+     sum(dcauchy(x, location = mu, scale = sigma, log = TRUE))
+ }
> loglik <- function(mu, sigma) {
+     matrix(mapply(loglikcauchy, mu = as.vector(outer(mu, rep(1,
+         length(sigma)))), sigma = as.vector(outer(rep(1, length(mu)),
+         sigma))), length(mu), length(sigma), dimnames = list(mu = mu,
+         sigma = sigma))
+ }
> ll <- loglik(mu, sigma)
> contour(mu, sigma, 2 * (ll - max(ll)))
> contour(mu, sigma, 2 * (ll - max(ll)), levels = -c(0.001, qchisq(0.95,
+     2), qchisq(0.99, 2)), add = TRUE)
> library(MASS)
> fitdistr(x, "cauchy", start = list(location = 0, scale = 1))
```

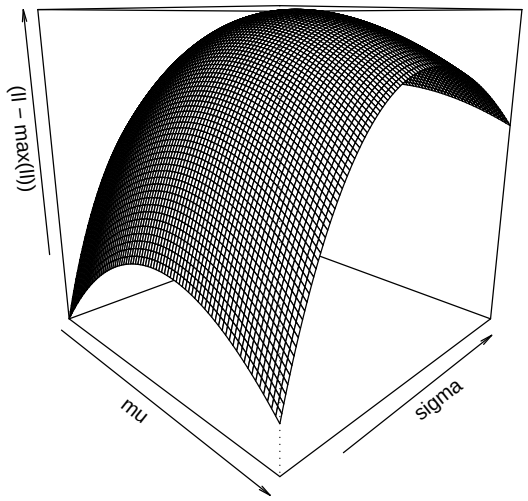
```
    location      scale
  0.05083830   1.12719758
 (0.10492878) (0.09716141)
```

```
> points(fitdistr(x, "cauchy", start = list(location = 0, scale = 1))[[1]])
```



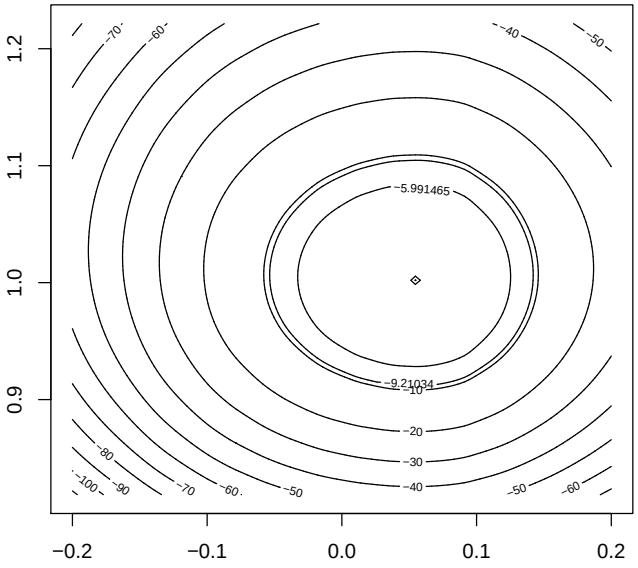
Here's a wireframe plot. It is even more fun to use the package rgl to get this, so that you can rotate the wireframe with your mouse. Unfortunately we can't do this in a pdf yet.

```
> persp(mu, sigma, (ll - max(ll)), theta = 45)
```



Now we'll do the same for the location-scale model based on the Laplace or double exponential distribution. The interesting thing about this example is that the log likelihood is not differentiable at one important point. The identity "expected score equals zero" does hold, but the identity "variance of score equals minus expected Hessian" does not. Still, the maximum likelihood estimator behaves perfectly well, and even the fitdist function from MASS, which computes first and second derivatives of log likelihood by numerical approximation (differences, and differences of differences) gives perfectly meaningful standard deviations as if the observed information matrix behaved as it does for smooth models. The "usual asymptotic theory" is valid as usual, as long as we compute the observed information by making a quadratic approximation to the log likelihood in a one over square root of n neighbourhood of the mle, rather than by differentiating twice at a particular point.

```
> set.seed(11091951)
> x <- rexp(1000) * (2 * (runif(1000) < 0.5) - 1)
> loglikdexp <- function(mu, sigma) {
+     sum(-((abs(x - mu))/sigma) - log(sigma))
+ }
> mu <- seq(from = -0.2, to = 0.2, length = 100)
> sigma <- exp(seq(from = -0.2, to = 0.2, length = 100))
> loglik <- function(mu, sigma) {
+     matrix(mapply(loglikdexp, mu = as.vector(outer(mu, rep(1,
+         length(sigma)))), sigma = as.vector(outer(rep(1, length(mu)),
+         sigma))), length(mu), length(sigma), dimnames = list(mu = mu,
+         sigma = sigma))
+ }
> ll <- loglik(mu, sigma)
> contour(mu, sigma, 2 * (ll - max(ll)))
> contour(mu, sigma, 2 * (ll - max(ll)), levels = -c(0.001, qchisq(0.95,
+     2), qchisq(0.99, 2)), add = TRUE)
> ddexp <- function(x, mu, sigma) {
+     (1/(2 * sigma)) * exp(-abs(x - mu)/sigma)
+ }
> fitdistr(x, ddexp, start = list(mu = 0, sigma = 1))

       mu         sigma
  0.05539934   1.00202043
 (0.02333902) (0.03168830)

> points(fitdistr(x, ddexp, start = list(mu = 0, sigma = 1))[[1]])
```

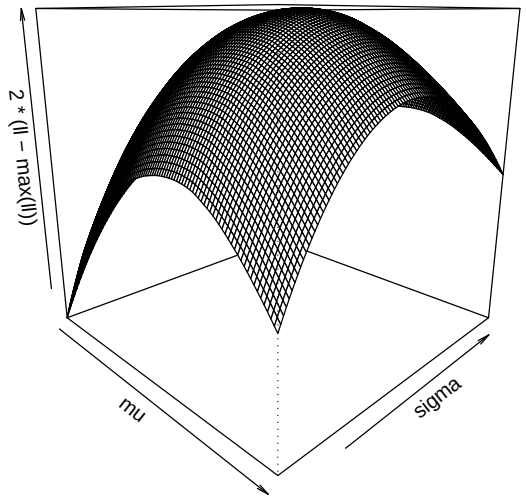


We got some error messages here but they seem harmless:

```
4: In log(dens(parm, ...)) : NaNs produced
5: In log(dens(parm, ...)) : NaNs produced
6: In log(dens(parm, ...)) : NaNs produced
7: In log(dens(parm, ...)) : NaNs produced
8: In log(dens(parm, ...)) : NaNs produced
9: In log(dens(parm, ...)) : NaNs produced
```

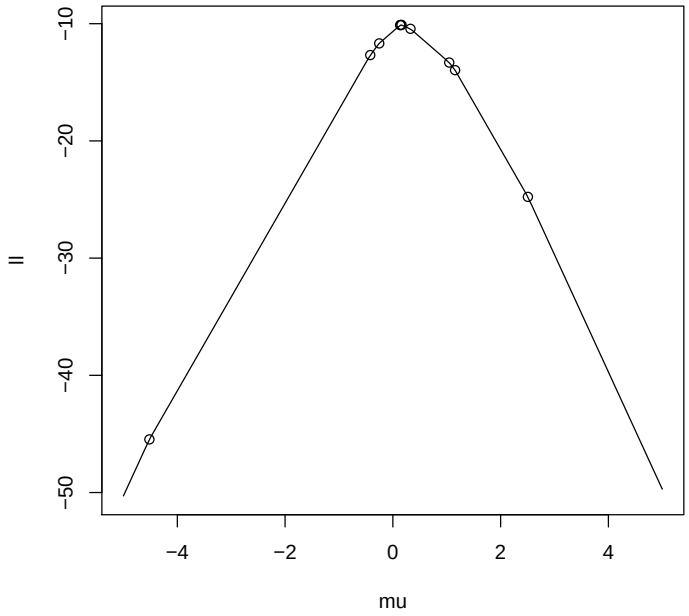
For completeness, here's the wireframe plot:

```
> persp(mu, sigma, 2 * (ll - max(ll)), theta = 45)
```



And finally, the log likelihood for 10 observations, just to convince you of its nonsmoothness

```
> mu <- seq(from = -5, to = 5, length = 100)
> set.seed(11091951)
> x <- rexp(10) * (2 * (runif(10) < 0.5) - 1)
> ll <- loglik(mu, 1)
> plot(mu, ll, type = "l")
> points(x, loglik(x, 1))
```



Exercise Write me an elegant short transparent R function for computing the log likelihood for user specified two-parameter models on a user supplied grid of points, to replace my clumsy code above. (A beer or bar of chocolate for the winner).