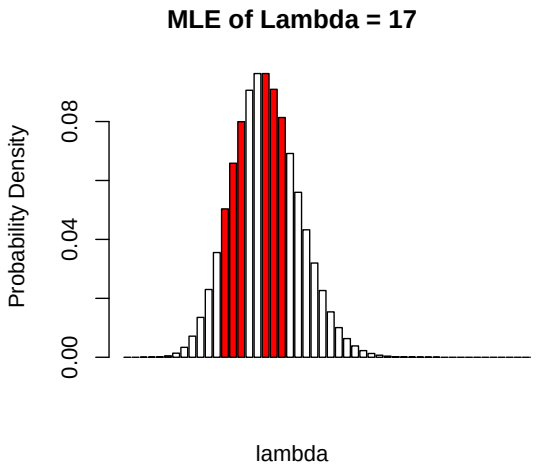
Fitting & Evaluating Likelihood Models

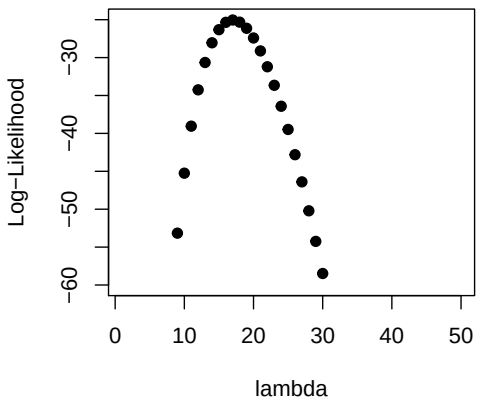
We Have the Maximum Likelihood Estimate of Lambda



p(a and b) = p(a)p(b)

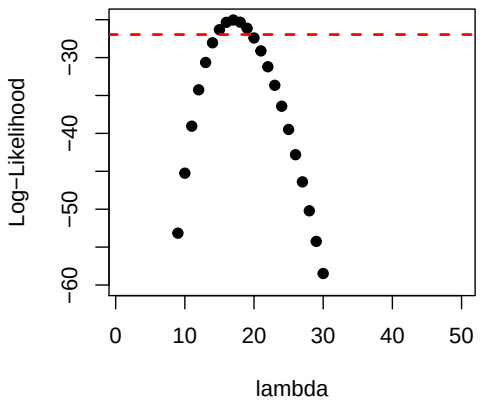
$$p(D|\theta) = \prod_{i=1}^{n} p(d_i|\theta)$$

What is the Variation Around our Estimate



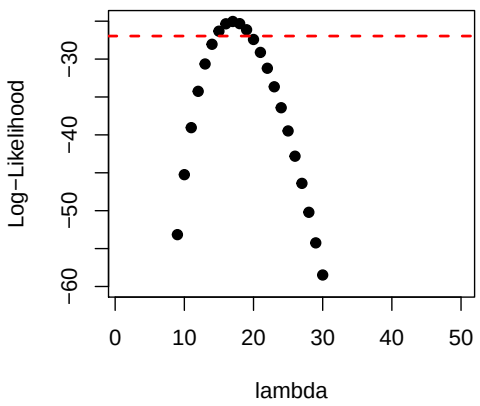
1. Log-Likelihood appxomiately χ^2 distirbuted
2. 95% CI holds all values within half of the .05 tail of $\chi^2_{df=1}$
3. (≈ 1.92)

Profile Likelihood CIs



1. Log-Likelihood appxomiately χ^2 distirbuted
2. 95% CI holds all values within half of the .05 tail of $\chi^2_{df=1}$
3. (≈ 1.92)

Profile Likelihood CIs




```
lConf <- max(ll) - 1.92
lConf

# [1] -26.96


#Get which values of lambda  that have
confVals <- lambdaVals[which(ll >= lCor

confVals[c(1, length(confVals))]

# [1] 15 19
```

How do we Compare Alternate Hypotheses?

$$G = 2ln(\frac{L_A}{L_0})$$

where L0 is from the more constrained hypothesis.
G is χ^2 distributed with DF = Difference in Parameters

$$G = 2(LogL_A - LogL_0)$$

How do we Compare Alternate Hypotheses?

```
#compare our estimated fit to the hypothesis that lambda = 12
G12 <- 2*(ll[17] - ll[12])
pchisq(G12, 1, lower.tail=F)

# [1] 1.768e-05



#compare our estimated fit to the hypothesis that lambda = 15
G15 <- 2*(ll[17] - ll[15])
pchisq(G15, 1, lower.tail=F)

# [1] 0.1099
```

Exercise: Likelihood and Bees!

- Load the Bee Lifespan Data
- Model Bee Lifespans as a Gamma Distribution with shape = 1 (1 bee per death)
- What is the ML estimate of a Bee's Lifespan?
- What is the 95% CI?
- Is the scale different from 10?

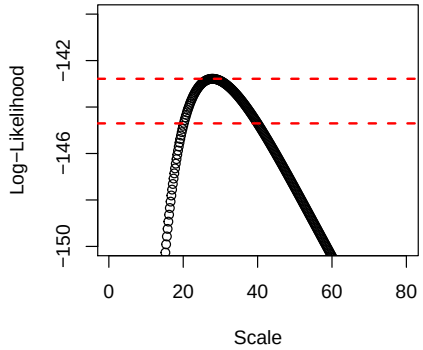
Exercise: Likelihood and Bees!

```
bees <- read.csv("./data/20q18BeeLifespans.csv")

#find the mle
scaleVals <- seq(0.2, 80, 0.2)

beeD <- function(x) sum(dgamma(bees$hours, shape=1,
                               scale=x, log=TRUE))
mll <- sapply(scaleVals, beeD)
```

Exercise: Likelihood and Bees!



abline(h=x) for horizontal lines, use v for vertical lines

Exercise: Likelihood and Bees!

```
#a function to get a CI given values and their log-likelihood
mllCI <- function(values, logl) {
  ci <- values[which(logl > max(logl) - 1.92)]
  ci[c(1, length(ci))]
}

mllCI(scaleVals, mll)

# [1] 20.2 39.8
```

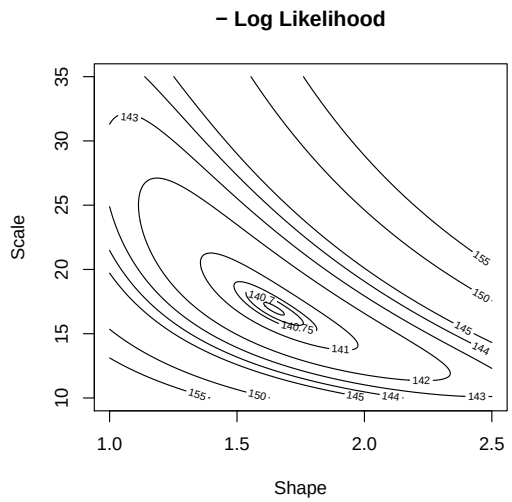
Exercise: Likelihood and Bees!

```
G <- 2 * max(mll - beeD(10))
pchisq(G, df=1, lower.tail=F)

# [1] 1.386e-12
```

What if you have multiple parmeters?

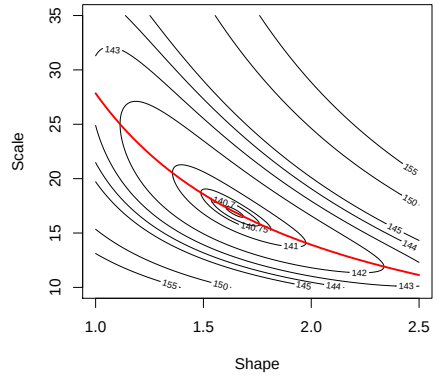
What if we Estimated Shape and Scale?



New Issues with Multiple Parameters

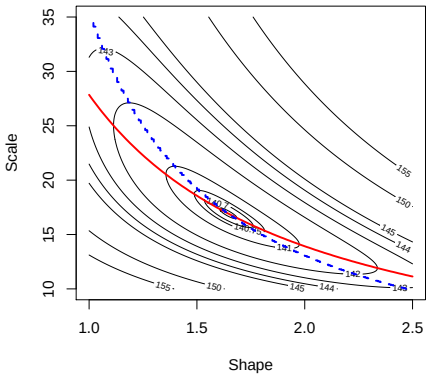
1. What Log-Likelihood Values Are Used for 95% CI?
2. Brute-Force Becomes Slow
3. Algorithmic Solutions Necessary
4. Specification Unwieldy

We Get the Likelihood Profile of One Coefficient by Iterating Over the Other



Shapes

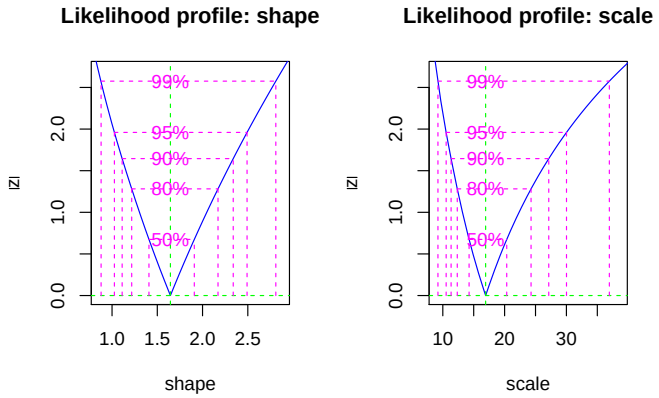
We Get the Likelihood Profile of One Coefficient by Iterating Over the Other



Shape, Scale

We Get the Likelihood Profile of One Coefficient by Iterating Over the Other

```
plot(profile(beeLL_Fit))
```



How do we Search Likelihood Space?

Optimizing to find a Minimum Value

- optim
- nlm
- nlminb
- mle2 (wrapper for all of the above)

Did You Say Minimum?

YES!

We optimize using -sum(LL Function)

Deviance = -2 * LL

How do we Search Likelihood Space?

There are many Algorithms

- Newtown-Raphson (algorithmicly implemented in `nlm` and `BFGS` method) uses derivatives
  - good for smooth surfaces & good start values
- Brent's Method - for single parameter fits
- Nelder-Mead Simplex (`optim`'s default)
  - good for rougher surfaces, but slower
- Simulated Annealing (`SANN`) uses Metropolis Algorithm search
  - global solution, but slow

Warning: If your algorithm fails to converge, you cannot evaluate your model or coefficients

Fitting Multiple Parameters with MLE2 from bbmle

```
#first write a function that you want to minimize
beeLL <- function(shape, scale) -sum(dgamma(hours,
shape=shape, scale=scale,
log=TRUE))

#now feed the function to an optimizer
beeLL_Fit <- mle2(beeLL, data=bees, start=list(shape=1, scale=4))
```

Fitting Multiple Parameters with MLE2 from bbmle

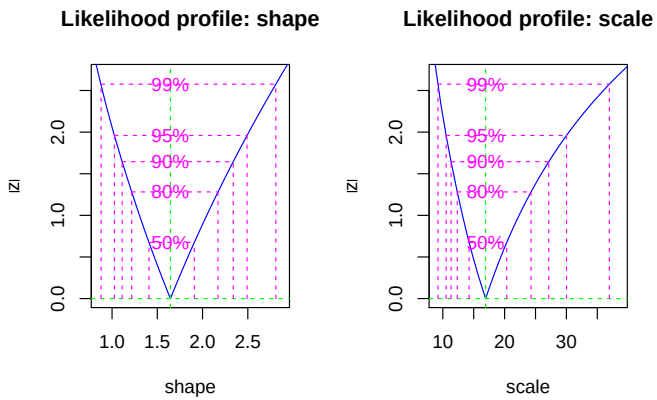
```
summary(beeLL_Fit)

# Maximum likelihood estimation
#
# Call:
# mle2(minuslogl = beeLL, start = list(shape = 1, scale = 4), data = be
#
# Coefficients:
#       Estimate Std. Error z value   Pr(z)
# shape    1.645      0.371    4.43 9.3e-06
# scale   16.931      4.458    3.80 0.00015
#
# -2 log L: 281.3
```

Coefficient tests based on Wald Confidence Intervals

Easy Profiling

```
plot(profile(beeLL_Fit))
```



Confidence Intervals

```
confint(beeLL_Fit)

#         2.5 % 97.5 %
# shape  1.027  2.492
# scale 10.569 30.020


# Wald CIs assume quadratic relationship at peak
confint(beeLL_Fit, method="quad")

#         2.5 % 97.5 %
# shape 0.9175  2.372
# scale 8.1935 25.668
```

Or...MLE2 has Many Probability Functions Builtin

```
mleBees <- mle2(hours ~ dgamma(shape=shape, scale=scale),
                data=bees, start=list(shape=1,scale=4))
```

Or...MLE2 has Many Probability Functions Builtin

```
#You can represent many relationships with a function
#or just using mle2 - so, poisson regression
#you could write a function
flowerLL <- function(b, int){
  fittedFlowers = b * nitrogen + int

  -sum(dpois(Flowers, lambda = fittedFlowers, log=T))
}
mleFlowers <- mle2(flowersLL, data=flowers,
            start=list(b=2, int = 4))
```

Or...MLE2 has Many Probability Functions Builtin

```
#Or...
mleFlowers <- mle2(Flowers ~ dpois(lambda = b * nitrogen + Int),
                data=flowers, start=list(b=2, Int = 4))
```

We Can Flexibly Compare Models using LRT

```
mleBeesOrig <- mle2(hours ~ dgamma(shape=1, scale=scale),
                data=bees, start=list(scale=4))

anova(mleBees, mleBeesOrig)

# Likelihood Ratio Tests
# Model 1: mleBees, hours~dgamma(shape=shape,scale=scale)
# Model 2: mleBeesOrig, hours~dgamma(shape=1,scale=scale)
#   Tot Df Deviance Chisq Df Pr(>Chisq)
# 1      2      281
# 2      1      286  4.26  1      0.039
```

To fit Ho for a linear model fit, just drop the predictor variable alltogether.

Exercise: Wolf Inbreeding and Litter Size

- Load the wolf pup data
- Write a MLE regression for pups N(inbreeding)
- This model will have three parameters
- Evluate it's CIs and Wald Tests
- Compare it to your Ho
- Compare it to `lm` results

Regression as Function

```
wolves <- read.csv("./data/16e2InbreedingWolves.csv")

wolfFun <- function(intercept, inbreeding, wolves_sd){
  pupsHat <- intercept + inbreeding*wolves$inbreeding.coefficient
  -sum(dnorm(wolves$pups, mean = pupsHat, sd = wolves_sd, log=TRUE))
}

wolf_mle<-mle2(wolfFun,
               start=list(intercept = 0, inbreeding=0, wolves_sd=3))
```

Regression with Distribution

```
wolf_mle2<-mle2(pups ~ dnorm(mean = intercept +
                               inbreeding*inbreeding.coefficient,
                              sd = wolves_sd),
                data=wolves,
                start=list(intercept = 0, inbreeding=0, wolves_sd=3))
```

Confidence Intervals

```
confint(wolf_mle2)

#              2.5 % 97.5 %
# intercept    5.022  8.112
# inbreeding -17.679 -5.214
# wolves_sd    1.127  1.993
```

Null Hypothesis Test

```
wolf_mleNull<-mle2(pups ~ dnorm(intercept, wolves_sd),
                data=wolves, start=list(intercept = 0, wolves_sd=3))

anova(wolf_mle2, wolf_mleNull)

# Likelihood Ratio Tests
# Model 1: wolf_mle2, pups~dnorm(mean=intercept+
#           inbreeding*inbreeding.coefficient,sd=wolves_sd)
# Model 2: wolf_mleNull, pups~dnorm(intercept,wolves_sd)
#   Tot Df Deviance Chisq Df Pr(>Chisq)
# 1      3     86.2
# 2      2     97.3  11.1  1    0.00088
```

Comparison to LM

```
wolf_lm <- lm(pups ~ inbreeding.coefficient, data=wolves)

summary(wolf_lm)

#
# Call:
# lm(formula = pups ~ inbreeding.coefficient, data = wolves)
#
# Residuals:
#    Min     1Q Median     3Q    Max
# -2.133 -0.820 -0.434  0.668  3.608
#
# Coefficients:
#                        Estimate Std. Error t value Pr(>|t|)
# (Intercept)               6.567      0.791    8.31  3.1e-08
# inbreeding.coefficient  -11.447      3.189   -3.59   0.0016
#
# Residual standard error: 1.52 on 22 degrees of freedom
# Multiple R-squared: 0.369,  Adjusted R-squared: 0.341
```

Comparison to LM

```
summary(wolf_mle2)

# Maximum likelihood estimation
#
# Call:
# mle2(minuslogl = pups ~ dnorm(mean = intercept + inbreeding *
#     inbreeding.coefficient, sd = wolves_sd), start = list(intercept =
#     inbreeding = 0, wolves_sd = 3), data = wolves)
#
# Coefficients:
#            Estimate Std. Error z value   Pr(z)
# intercept     6.567      0.757    8.68 < 2e-16
# inbreeding  -11.447      3.053   -3.75 0.00018
# wolves_sd     1.459      0.211    6.93 4.3e-12
#
# -2 log L: 86.23
```