Maximum Likelihood Estimation

Estimation

Goal

Estimate parameter values for a given model.

Compute values for the model that optimize some criterion

LS Estimation

Criterion: Sum of squared residuals (measure of the model error)

Optimization: Minimization

Estimation Methods

★ Least squares

★ Maximum likelihood

★ Method of Moments

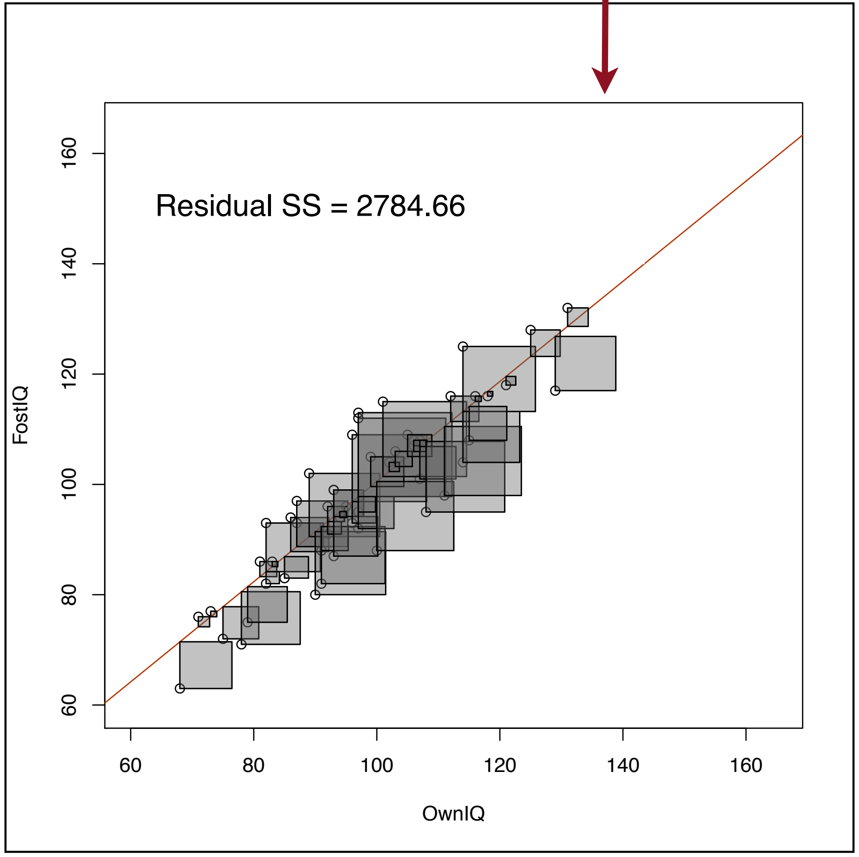
★ Bayesian

★ Many more

LS Estimation

Commonly used in estimating linear regression coefficients

Minimizes the sum of squared residuals



Maximum Likelihood

Key Question

Given the data and model, what are the most likely value of the parameters?

Maximum likelihood (ML) provides framework for answering this question

and

provides results which have attractive properties favorable for inference

Criterion: likelihood function (or log-likelihood function or deviance function)

Optimization: Minimization

$$L\left(x_1, x_2, x_3, \ldots, x_n; \theta\right)$$

Likelihood function represents the **joint probability** or **likelihood** of observing the data that have been collected

Advantages of ML

★ ML yields a **global fit index** for the model on top of the parameter estimates

- ✓ Can be used to compare models

★ Estimators produced under ML have **desirable large-sample (asymptotic) properties**

- ✓ Consistent, asymptotically normally distributed
- ✓ With small samples this may not hold

★ ML estimates are **always** approximate

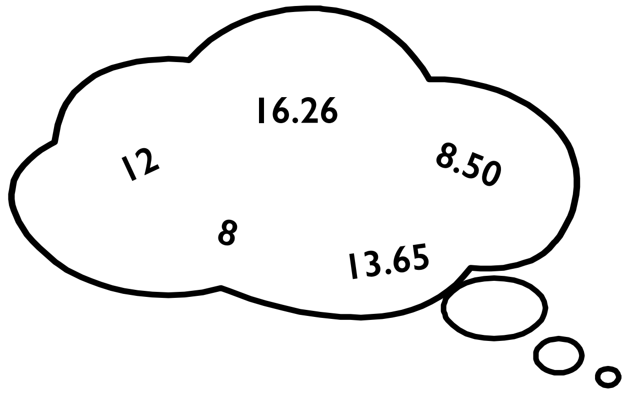
★ Approximations imply researchers **should not** get too hung up on rigid rules of practice

- ✓ Use of 2(SE) vs 1.96(SE)
- ✓ Inflexible cutoffs (0.05) are not warranted

★ Notes only consider **regular problems**

- ✓ ML solutions are possible
- ✓ Assumption of sample data coming from hypothetical, infinitely sized population (repeated sampling scenario)

Example 1: Estimate the Mean



Given the **data** and the **model**, what is the most likely value of the mean?

Data: The data are given

Model: Specified by the researcher

e.g., the marginal mean model is an LM with distributional assumption on the errors

The Model

$$y_i = \beta_0 + \epsilon_i$$

The errors are normally distributed

The Data

In general are $y_1, y_2, y_3, \ldots, y_n$

Distributional assumption is normal

The probability density of y_i is $f(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left[-\frac{(y_i - \mu)^2}{2\sigma^2}\right]$

With the further usual assumption that $\mu_\varepsilon = 0$ this simplifies to

$$f(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left[-\frac{y_i^2}{2\sigma^2}\right]$$

Probability model associates different probability densities with different y_i's

Suppose $\mu = 0$, $\sigma^2 = 1$ and $y_i = -1$ then the probability density is given by $f(y_i) = f(-1)$

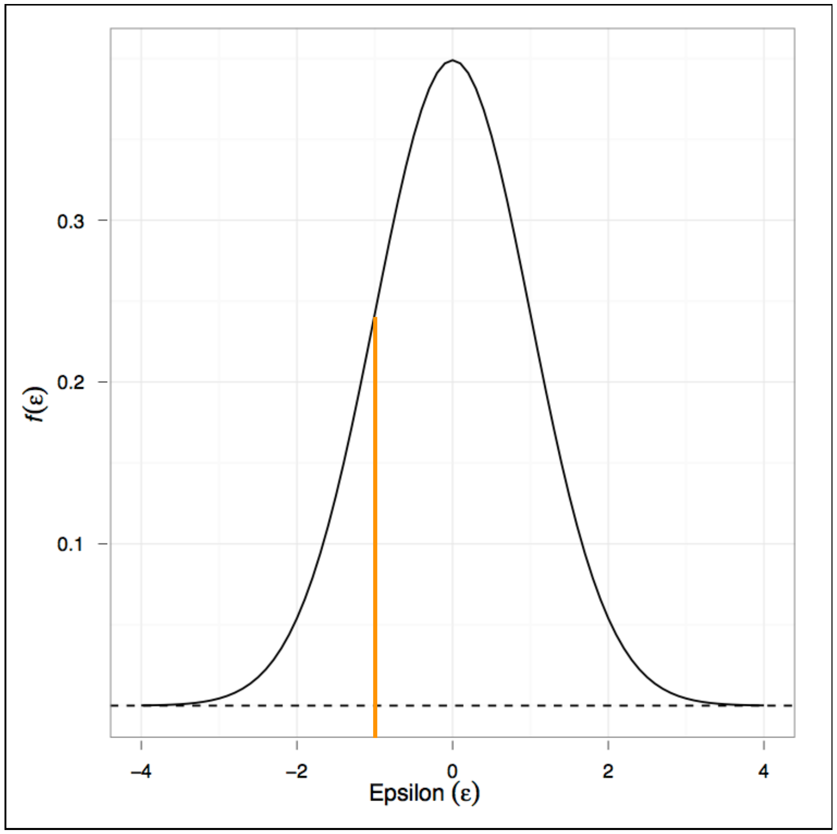
$$f(-1) = \frac{1}{\sqrt{2 \cdot \pi \cdot 1}} \times \exp\left[-\frac{(-1)^2}{2 \cdot 1}\right]$$
$$= 0.2419707$$

The `dnorm()` function computes probability densities from a normal distribution in R.

```
> dnorm(-1, mean = 0, sd = sqrt(1) )

[1] 0.2419707
```

Note it takes the SD rather than the variance of the distribution



Joint Probability Density

The **joint probability density** is the probability of observing y_1 *and* y_2 *and* y_3 *and* … *and* y_n

Assumption of **independence** allows us to multiply each probability density function to obtain **joint density**

$$p(y_1, y_2, \ldots, y_n) = f(y_1) \times f(y_2) \times \ldots \times f(y_N)$$

$$p(y_1, y_2, \ldots, y_n) = \boxed{\left[\frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left(-\frac{y_1^2}{2\sigma^2}\right)\right]} \times \boxed{\left[\frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left(-\frac{y_2^2}{2\sigma^2}\right)\right]} \times \ldots \times \boxed{\left[\frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left(-\frac{y_n^2}{2\sigma^2}\right)\right]}$$

$$p(y_1, y_2, \ldots, y_n) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \times \exp\left[-\frac{\sum y_i^2}{2\sigma^2}\right]$$

joint density function

In working with statistical models, typically the distributional assumption is on the errors. This whole process is carried out using the errors (ε_i) rather than the *y*'s

$$p(\epsilon_1, \epsilon_2, \ldots, \epsilon_n) = \left(\frac{1}{\sqrt{2\pi\sigma_\epsilon^2}}\right)^N \times \exp\left[-\frac{\sum \epsilon_i^2}{2\sigma_\epsilon^2}\right]$$

joint density of the errors

Likelihood

Then we have the joint probability density of the errors...

$$p(\epsilon_1, \epsilon_2, \ldots, \epsilon_n) = \left(\frac{1}{\sqrt{2\pi\sigma_\epsilon^2}}\right)^N \times \exp\left[-\frac{\sum \epsilon_i^2}{2\sigma_\epsilon^2}\right]$$

...and a model

$$y_i = \beta_0 + \epsilon_i$$

We can re-express εᵢ as $y_i - \beta_0$ and substitute into the function for the joint density

$$p(\epsilon_1, \epsilon_2, \ldots, \epsilon_n) = \left(\frac{1}{\sqrt{2\pi\sigma_\epsilon^2}}\right)^N \times \exp\left[-\frac{\sum (y_i - \beta_0)^2}{2\sigma_\epsilon^2}\right]$$

Once we express the probability density as a function of parameters it is called the **likelihood function**.

$$\mathcal{L}(\boldsymbol{\beta}; \mathbf{y}) = \left(\frac{1}{\sqrt{2\pi\sigma_\epsilon^2}}\right)^N \times \exp\left[-\frac{\sum (y_i - \beta_0)^2}{2\sigma_\epsilon^2}\right]$$

Likelihood function

Likelihood of the vector of betas given the data

Log-Likelihood

It is often easier to work mathematically with the log of the likelihood function (**log-likelihood**).

$$\log \mathcal{L}(\boldsymbol{\beta};\mathbf{y}) = \log\left(\left(\frac{1}{\sqrt{2\pi\sigma_\epsilon^2}}\right)^N \times \exp\left[-\frac{\sum(y_i-\beta_0)^2}{2\sigma_\epsilon^2}\right]\right)$$

Choosing the natural log this is re-expressed as

$$\ell(\boldsymbol{\beta};\mathbf{y}) = -\frac{N}{2}\times\ln\left(2\pi\sigma_\epsilon^2\right) - \frac{1}{2\sigma_\epsilon^2}\times\sum(y_i-\beta_0)^2$$

Log-likelihood function

Do the math to convince yourself it reduces to this

- ★ Assume that the parameters of the error variance is known, and that the **only unknown** parameter to be estimated is the intercept

$$\sigma_\varepsilon^2 = 15$$

- ★ Given the model and the data, object is to choose "best" value for the intercept
- ★ "Best" value here means that after substituting values in to the deviance function equation, the smallest deviance possible has been obtained (i.e., deviance is minimized)
- ★ In ML theory, this is the estimate (value) of β_0 that is **maximally likely** given the data and the model– hence maximum likelihood

Everything in the log-likelihood function is known, except for β$_0$

$$\ell(\boldsymbol{\beta};\mathbf{y}) = -\underbrace{\frac{N}{2}}_{N = 5} \times \ln\left(2\pi\underbrace{\sigma_\epsilon^2}_{\sigma^2_\varepsilon = 15}\right) - \frac{1}{2\sigma_\epsilon^2} \times \sum(\underbrace{y_i}_{y_i = \{12, 16.26, 8.50, 8, 13.65\}} - \beta_0)^2$$

Substitute in the values for σ_ε^2, Y$_i$, and *N*

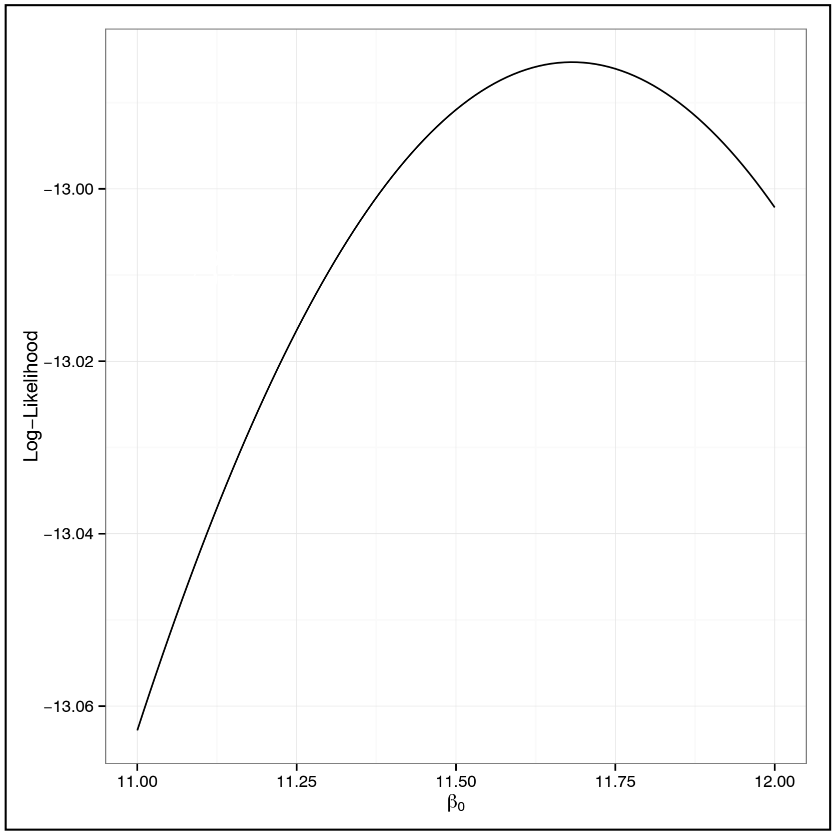
$$\ell(\boldsymbol{\beta};\mathbf{y}) = -\frac{5}{2} \times \ln(2 \times \pi \times 15) - \frac{1}{2 \times 15} \times \left[(12 - \beta_0)^2 + (16.26 - \beta_0)^2 + (8.50 - \beta_0)^2 + (8 - \beta_0)^2 + (13.65 - \beta_0)^2\right]$$

Given we want the largest (maximum) log-likelihood what value should we choose for β$_0$?

- ★ Pick several candidate values for β$_0$
- ★ Substitute them into the equation
- ★ Solve for deviance
- ★ Choose the value for β$_0$ that produces the smallest deviance

For example, consider candidate values for β_0 between 11 and 12

β_0	Log-likelihood
11	−13.06282
11.01	−13.06056
11.02	−13.05834
11.03	−13.05615
⋮	⋮



```
## Create a function to compute the log-likelihood
> LL = function(b0) {
     -5/2 * log(2 * pi * 15) - 1/(2 * 15) * ( (12 - b0)^2 + (8 - b0)^2 + (16.26 - b0)^2 +
     (13.65 - b0)^2 + (8.5 - b0)^2 )
     }

## Try function
> LL(11)

[1] -13.06282
```

Carry Out a Grid Search in R

```
> library(dplyr)
> library(ggplot2)

## Generate candidate values for b0
> candidates = data.frame(
    b0 = seq(from = 11, to = 12, by = 0.01)
    )

## Generate the deviance values and store the results
> gridSearch = candidates %>%
  rowwise() %>%
  mutate(ll = LL(b0))


## Examine search grid
> head(gridSearch) +

Source: local data frame [6 x 2]

     b0        ll
  (dbl)     (dbl)
1 11.00 -13.06282
2 11.01 -13.06056
3 11.02 -13.05834
4 11.03 -13.05615
5 11.04 -13.05399
6 11.05 -13.05187
```

Carry Out a Grid Search in R

```
## Plot log-likelihood vs. b0
> ggplot(data = gridSearch, aes(x = b0, y = ll)) +
    geom_line() +
    theme_bw() +
    xlab(expression(beta[0])) +
    ylab("Log-Likelihood")



## Get the row with the largest log-likelihood
> gridSearch %>% slice(which.max(ll))

Source: local data frame [1 x 2]

     b0       ll
  (dbl)    (dbl)
1 11.68 -12.9853
```

Alternative: Use Calculus

- ★ The derivative of the log-likelihood function can be analytically computed and set equal to zero. Solving for β0 will give the value that maximizes the deviance
- ★ Avoids exhaustive search
- ★ In more complex models (e.g., GLMs) estimation is not so straightforward.
  - ✓ Exhaustive search methods are required (numerical analysis)
- ★ Once the log-likelihood function becomes more complex, the multidimensional form of the tangent line most be discovered

$$\frac{\partial}{\partial \beta_0} \quad -\frac{N}{2} \times \ln\left(2\pi\sigma_\epsilon^2\right) - \frac{1}{2\sigma_\epsilon^2} \times \sum (y_i - \beta_0)^2$$

Compute the partial derivative with respect to β0

Do you remember the product rule? What about the chain rule?

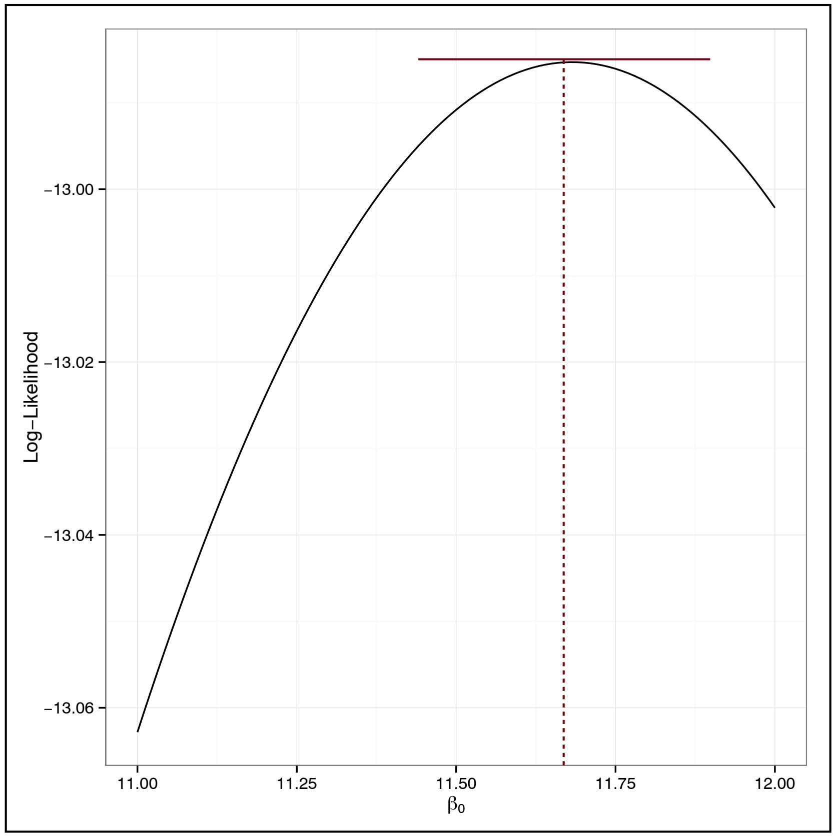
$$0 = -2\left[(y_1 - \beta_0) - (y_2 - \beta_0) - \ldots - (y_n - \beta_0)\right]$$

Set this equal to 0 and solve the equation for β0 (find the roots of the function)

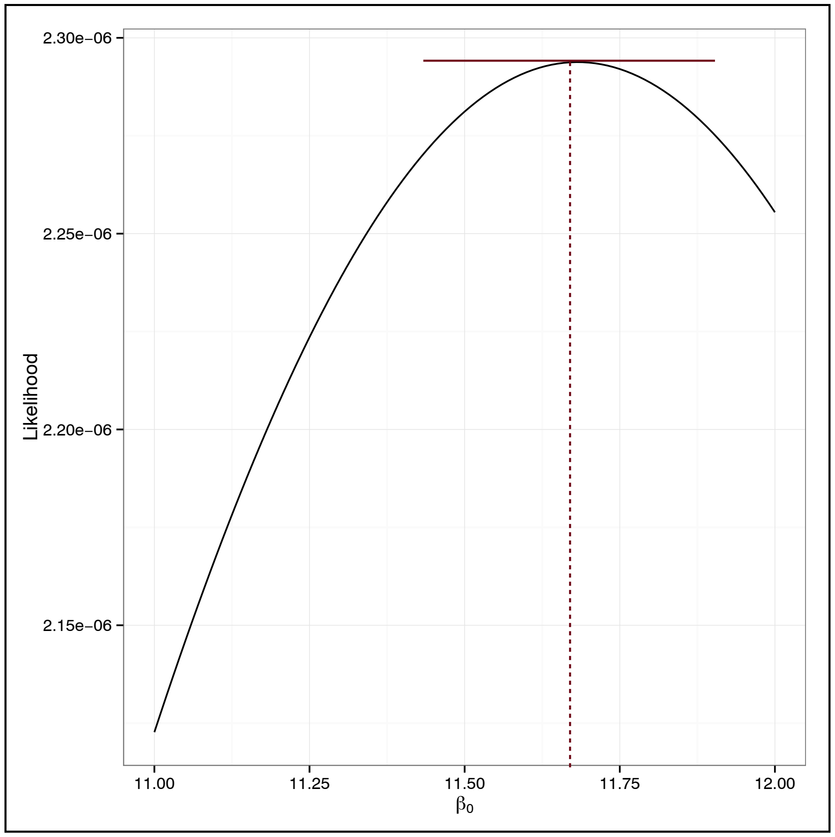
$$\beta_0 = \frac{\sum y_i}{n}$$

Sample mean is the estimate that maximizes the log-likelihood...in other words, the sample mean is the **maximum likelihood** estimate.

The calculus is essentially computing the β_0 estimate that has a tangent line to the log-likelihood function with a slope of zero.



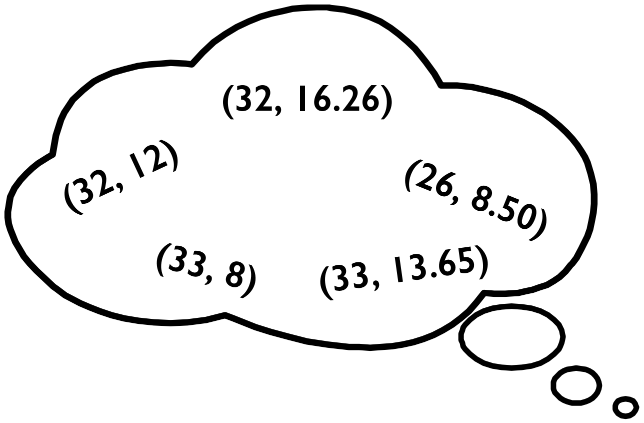
Log-Likelihood function



Likelihood function

Since the logarithm is a monotonic function, maximizing the log-likelihood function is equivalent to maximizing the likelihood function

Example 2: Regression (Estimate Intercept, Slope, and Error Variance)



Given the data and the model, what is the most likely value of the unknown intercept parameter, slope parameter and error variance parameter?

Data: The data are given

Model: Regression model

$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$

Embed the model

$$\ell(\beta_0, \beta_1, \sigma^2_\epsilon; \mathbf{y}) = -\frac{5}{2} \times \ln(2 \times \pi \times \sigma^2_\epsilon) - \frac{1}{2 \times \sigma^2_\epsilon} \times \sum (y_i - \beta_0 - \beta_1(x_i))^2$$

$$\ell(\beta_0, \beta_1, \sigma^2_\epsilon; \mathbf{y}) = -\frac{5}{2} \times \ln(2 \times \pi \times \sigma^2_\epsilon) - \frac{1}{2 \times \sigma^2_\epsilon} \times [(12 - \beta_0 - \beta_1(32))^2 + (8 - \beta_0 - \beta_1(33))^2 + \ldots + (8.5 - \beta_0 - \beta_1(26))^2]$$

Try alternative values for the three different parameters. Choose the set that gives the smallest deviance.

Candidate values for β_0, β_1, and σ^2 are considered simultaneously

β_0	β_1	σ^2	log-lik
–4.00	0	15	d_1
–3.99	0	15	d_2
–3.98	0	15	d_3
–3.97	0	15	d_4
⋮	⋮	15	d_5
–3.92	0.5	15	d_6
⋮	⋮	⋮	⋮

With three unknown parameters, we need to minimize a four-dimensional function.

Note that the search grid is **bigger by many magnitudes.**

Consider the following candidate values:

- β_0 = [–4, –3] by 0.01 (100 candidate values)
- β_1 = [0, 2] by 0.01 (200 candidate values)
- σ^2 = [10, 15] by 0.01 (500 candidate values)

The search grid consists of 100 × 200 × 500 = 10,000,000 possible sets of candidate values!

Computational time for me was about 14 minutes!

And that is under a very specified range of values for each parameter and not much precision (hundredths).

Imagine if you needed the parameters estimated to 3 or 4 decimal places and had no clue what the values were?

Log-Likelihood Function

```
## Create a function to compute the deviance
> LL2 = function(b0, b1, sigma2) {
     -5/2 * log(2 * pi * sigma2) - 1/(2 * sigma2) * (
       (12    - b0 - b1 * 32)^2 +
       (8     - b0 - b1 * 33)^2 +
       (16.26 - b0 - b1 * 32)^2 +
       (13.65 - b0 - b1 * 33)^2 +
       (8.5   - b0 - b1 * 26)^2
       )
     }


## Try function
> LL2(b0 = 0, b1 = 0, sigma2 = 15)

[1] -35.73015
```

Carry Out a Grid Search in R

```
## Generate candidate values for b0
> candidates = expand.grid(
    b0     = seq(from = -4, to = -3, by = 0.01)
    b0     = seq(from =  0, to =  1, by = 0.01)
    sigma2 = seq(from =  0, to = 20, by = 0.01)
    )

## Generate the log-likelihood values and store the results
> gridSearch = candidates %>%
  rowwise() %>%
  mutate(ll = dev(b0, b1, sigma2))


## Find the row with the smallest deviance
> gridSearch %>% slice(which.max(ll))

     b0     b1 sigma2         ll
  (dbl) (dbl)  (dbl)      (dbl)
1 -3.92   0.5   7.99 -12.28932
```

```
## Examine search grid
## Note: Getting NAs does not necessarily mean your function failed…

> head(gridSearch)

     b0    b1 sigma2       ll
  (dbl) (dbl)  (dbl)    (dbl)
1 -4.00     0      0       NA
2 -3.99     0      0       NA
3 -3.98     0      0       NA
4 -3.97     0      0       NA
5 -3.96     0      0       NA
6 -3.95     0      0       NA
```

Using calculus, we would minimize the function, by finding the roots for the three partial derivatives for the function setting each equal to zero, and solving for each parameter. In this case, using calculus saves us a lot of time.

It is convenient to use a computer to estimate the parameters and determine values of the log-likelihood.

```
## Create data
> x = c(32, 33, 32, 33, 26)
> y = c(12, 8, 16.26, 13.65, 8.5)


## Fit linear model
> lm.a = lm(y ~ x)


## Compute log-likelihood
> logLik(lm.a)

'log Lik.' -12.28932 (df=3)
```

Deviance

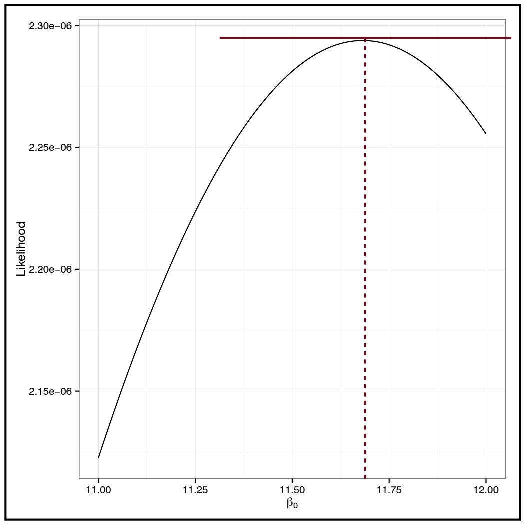
Some statisticians prefer to work with the **deviance**.

$$-2 \times \ell(\boldsymbol{\beta};\mathbf{y}) = -2 \times \left(-\frac{N}{2} \times \ln\left(2\pi\sigma_\epsilon^2\right) - \frac{1}{2\sigma_\epsilon^2} \times \sum (y_i - \beta_0)^2\right)$$

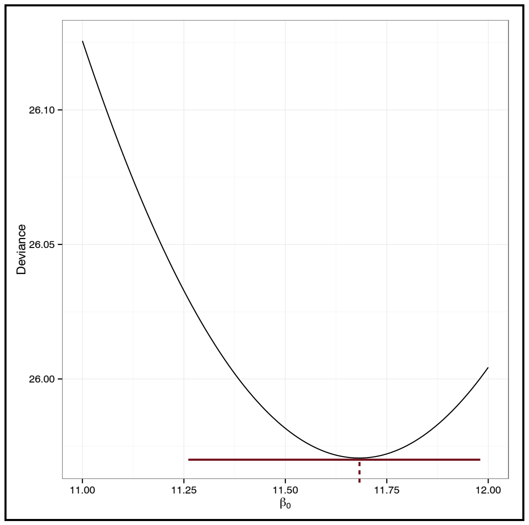
The deviance function is the log-likelihood multiplied by –2

$$\mathcal{D}(\boldsymbol{\beta};\mathbf{y}) = N \times \ln\left(2\pi\sigma_\epsilon^2\right) + \frac{1}{\sigma_\epsilon^2} \times \sum (y_i - \beta_0)^2$$

Deviance function



Log-Likelihood function



Deviance function

Maximizing the log-likelihood function is equivalent to **minimizing the deviance** function

```
## Compute deviance
> -2 * logLik(lm.a)[1]

24.57864
```

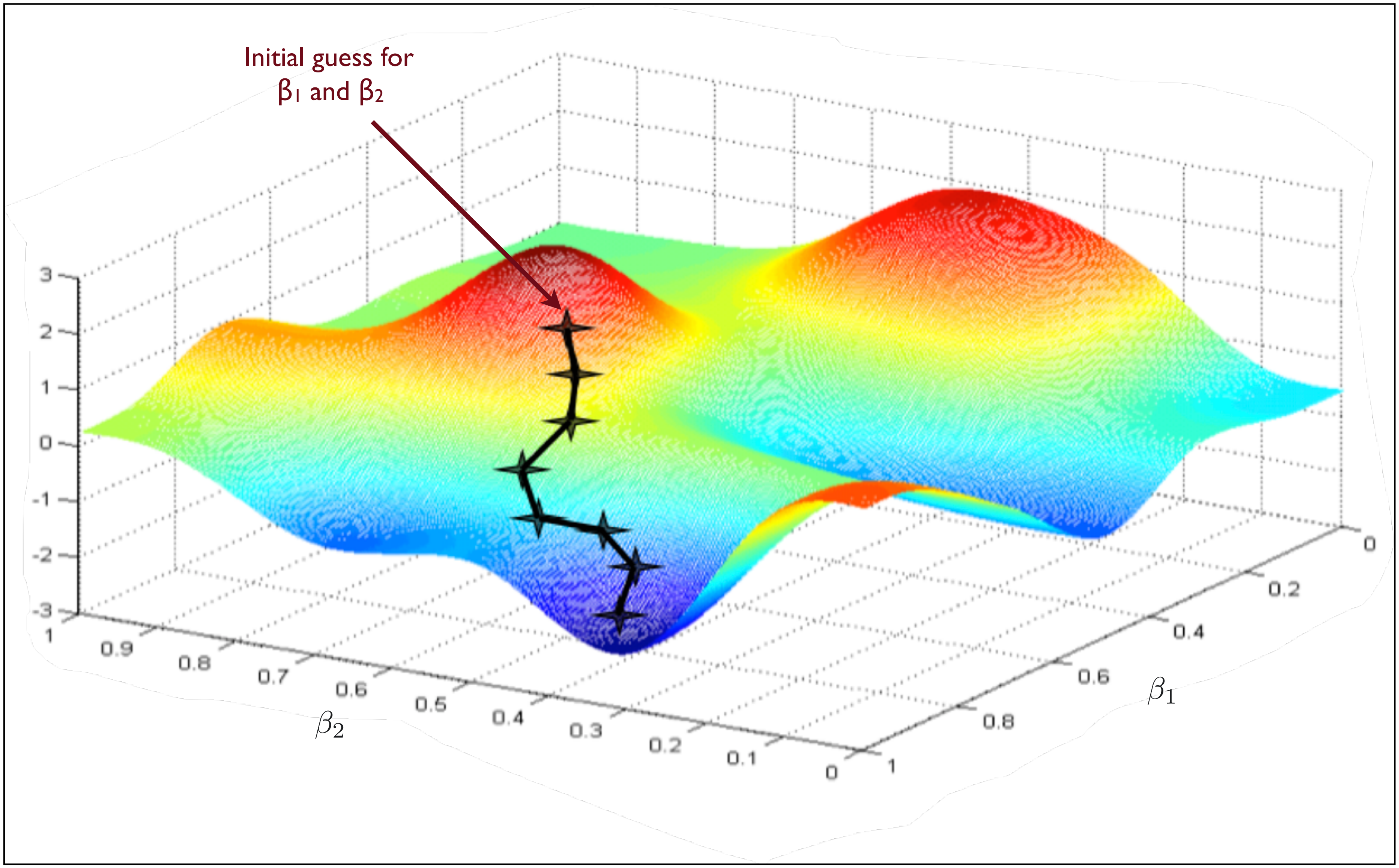
If we computed the deviance we would look for the MINIMUM value.

Iterative Methods of Estimation

Exhaustive Search vs. Numerical Methods

- ★ The exhaustive searches carried out in the examples were convenient
  - ✓ Ranges were used where the ML estimates were known to reside
  - ✓ In practice, these are not known
  - ✓ Increments are usually finer than 0.01 in practice
- ★ Computing time and memory becomes a valuable resource
- ★ Numerical methods, based on calculus are usually employed
  - ✓ Newton-Raphson method is most common algorithm
  - ✓ Deviance function assumed to be smooth and continuous with only one minimum (regularity assumption)
- ★ Functions such as `glm()` and `lmer()` generally combine both methods
  - ✓ Numerical methods are used to get in the neighborhood of the minimum deviance
  - ✓ Once more limited search space has been defined an iterative method can hone in on the minimum deviance until it is "good enough"

Gradient Descent



Gradient Descent

Iterative optimization algorithm to minimize a function, $f(\boldsymbol{\beta})$

Begin with an initial guess for the values of the parameters

$$\beta_0^{(0)}, \beta_1^{(0)}, \ldots, \beta_k^{(0)}$$

Compute the derivative of the function at the parameter value(s) and multiply by some small constant amount (called the learning rate), α

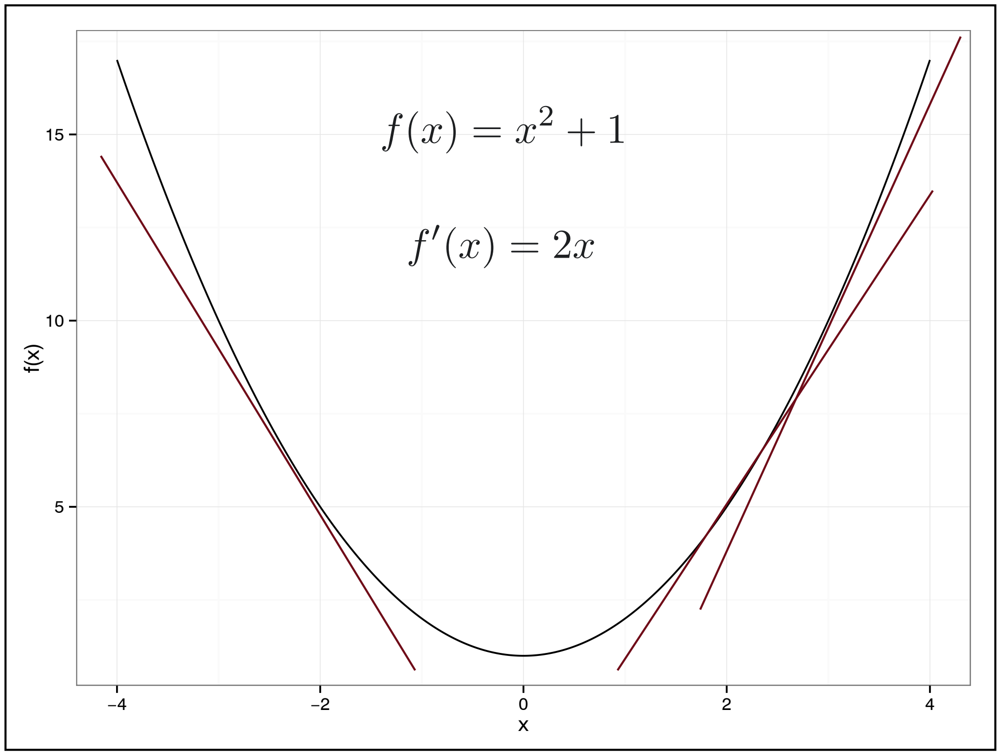
$$\alpha \left(\frac{\partial}{\partial \beta_j} f(\boldsymbol{\beta}) \right)$$

Update the values of the parameters by

$$\beta_j^{(1)} = \beta_j^{(0)} - \alpha \left(\frac{\partial}{\partial \beta_j} f(\boldsymbol{\beta}) \right)$$

Repeat using the new estimated solution

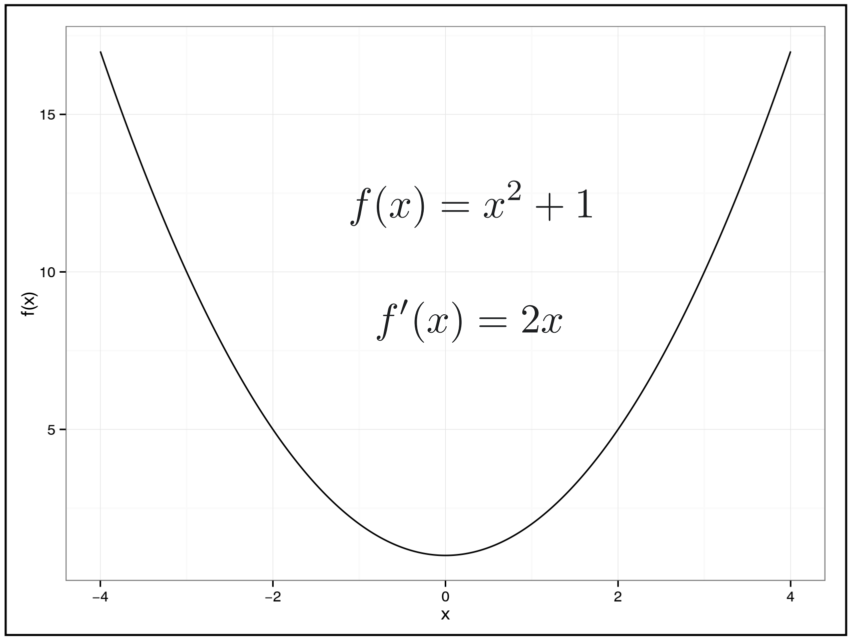
Stop when some criteria for convergence has been met



$f'(x = -2) = -4$ $f'(x = 2) = 4$ $f'(x = 3) = 6$

Key: Notice if derivative is negative the function slopes downward to the right; if derivative is positive the function slopes downward to the left; size of derivative indicates steepness of line

Gradient descent makes use of this to update the parameters



The minimum of the function is clearly at $x = 0$.

Start by making an initial guess of the parameter(s)…in this case *x*

$$x^{(0)} = 4$$

Pick a learning rate (something small)

$$\alpha = 0.1$$

The value of the learning rate determines how fast the search converges.

Update the parameter, *x*, according to

$$\beta_j^{(1)} = \beta_j^{(0)} - \alpha \left(\frac{\partial}{\partial \beta_j} f(\boldsymbol{\beta}) \right)$$

$$\frac{\partial}{\partial \beta_j} f(\boldsymbol{\beta}) = 2x = 2(4) = 8$$

Evaluate at $x^{(0)}$

$$\begin{aligned} x^{(1)} &= 4 - 0.1(8) \\ &= 3.2 \end{aligned}$$

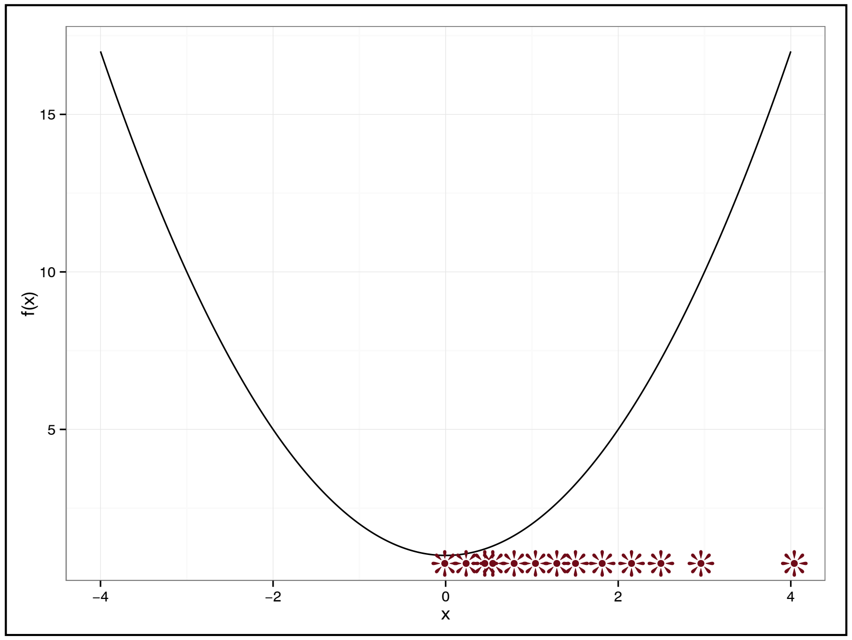
$x^{(1)}$ is closer to the minimum than $x^{(0)}$

Repeat

$$\begin{aligned} x^{(2)} &= 3.2 - 0.1(2 \cdot 3.2) \\ &= 2.56 \end{aligned}$$

$x^{(2)}$ is closer to the minimum than $x^{(1)}$

Repeat until the decrease is less than some value



Here is an example of some output from an `lmer()` fitted model.

```
  0:     2129.0419:
  1:     2067.3454:
  2:     2057.7418:
  3:     2050.2435:
  4:     2048.7592:
  5:     2048.6277:
  6:     2048.4524:
  7:     2048.3884:
  8:     2048.3731:
  9:     2048.3708:
 10:     2048.3705:
 11:     2048.3705:
 12:     2048.3705:
 13:     2048.3705:
 14:     2048.3705:
```

The first column indexes the iteration. The second gives the value of the deviance. Here the iterative estimation is done via penalized iteratively re-weighted least squares (PIRLS) rather than gradient descent.

Problems with the Estimation

★ Two potential problems for iterative procedures

- ✓ Possible for one of the $\boldsymbol{\beta}_k$ **to tend to infinity** (leads to non-convergence)
  - Generally means the model has been poorly specified (e.g., too many interactions, poor predictors, linear separation in a predictor, etc.)
  - Could also be due to data sparseness
  - Often the parameter can be fixed to a value to allow the other parameters to be estimated (the fixed parameter is then ignored in the output)
- ✓ Under particular conditions it is possible for the **estimates to iterate in such a way that they never converge** (but don't tend toward infinity)
  - The likelihood (or log-likelihood) should be increasing at each iteration
  - If not, it is a sign that the true root has converged on a local maximum or will not converge
  - Step-halving functions can be applied where half the distance to the prior and current estimates is evaluated. If the likelihood is lower, the step-halving function is re-applied. If the likelihood increases, sub-iterations continue until there is no change in the likelihood

Possible solution: Try different starting values

Possible solution: Fix some of the parameter values