Logistic Regression and Maximum Likelihood

Nicholas Kortessis

2025-04-22

Contents

Part 1: Logistic Regression

Introduction to Logistic Regression

Logistic regression is a statistical method used to model the probability of a binary outcome. Unlike linear regression which predicts continuous values, logistic regression predicts the probability that an observation belongs to one of two categories.

Remember that a binary outcome can be modeled with a Bernoulli distribution (or equivalently a Binomial distribution with $n = 1$ trial). For example, if we are interested in modeling the prevalence of a disease, we could use Y_i as the random variable indicating whether someone has a disease. Written mathematically, we have

$$Y_i \sim \text{Bernoulli}(p_i)$$

where p_i is the probability that Y_i is a "success". This model only has successes and failures. In the example of Y_i reflecting the disease state of an individual, any given individual either has the disease or not.

Rather than modeling the average value of Y_i across a bunch of individuals, logistic regression tries to find a way to model the probability of a success, p_i, as a function of known predictors.

Key characteristics of logistic regression:

- Used for binary classification problems (though it can be extended to multi-class)
- Outputs probabilities between 0 and 1 (generic "failure" and "success")
- Uses the logistic function to transform linear predictions to p.
- Coefficients are most easily interpreted through odds ratios
- Rather than using least squares estimation, logistic regression uses maximum likelihood estimation for fitting (more on that later).

The inverse logit (or logistic) function

To be able to model probabilities, logistic regression makes use of the so-called "inverse logit" function (sometimes called the logistic function. The inverse logit function is a kind of sigmoid function used to model the probability of success. It looks like this

$$P(Y_i = \text{success}) = p_i = \frac{e^{(\beta_0+\beta_1 X_i)}}{1+e^{(\beta_0+\beta_1 X_i)}}$$

where: p_i is the probability of a success for a binary variable, β_0 is the intercept, β_1 is the coefficient describing the effect of X_i, which is the predictor variable for individual i. The real value of inverse logit function is that it gives us a way to model probabilities, which take values between 0 and 1, on a scale that can take any real value.

Before going to far, it's helpful to visualize the logistic function. The general form for the inverse logit is

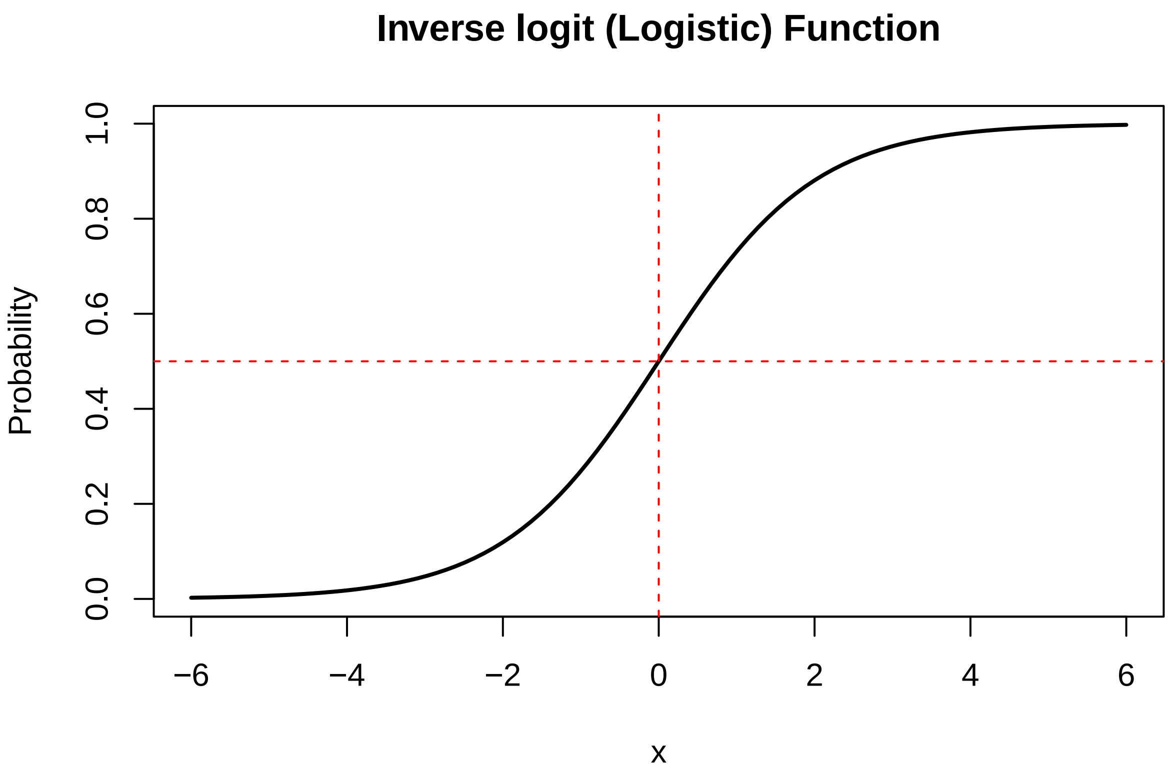
$$f(x) = \frac{e^x}{1+e^x}$$

or equivalently,

$$f(x) = \frac{1}{1+e^{-x}}.$$

These both are identical. Let's explore this function.

```r
# Create value for plotting the logistic function
x <- seq(from = -6, to = 6, length = 100)
y <- 1 / (1 + exp(-x))

# Plot logistic function
plot(x, y, type = "l", lwd = 2,
     xlab = 'x',
     ylab = "Probability",
     main = "Inverse logit (Logistic) Function")
abline(h = 0.5, lty = 2, col = "red")
abline(v = 0, lty = 2, col = "red")
```



Checkpoint 1: What is the value of the inverse logit function for x = 0, x = -1, x = 1, x = 2, x = -4? This function is most helpful for converting continuous values to values between 0 and 1. For the purposes of logistic regression, we create a linear model and then convert it to probabilities using the inverse logit function.

Log Odds and Odds Ratios

The logistic regression model can be rewritten in terms of the log odds using the logit transformation:

$$\ln\left(\frac{P(Y_i = \text{success})}{1 - P(Y_i = \text{success})}\right) = \ln\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + \beta_1 X_i$$

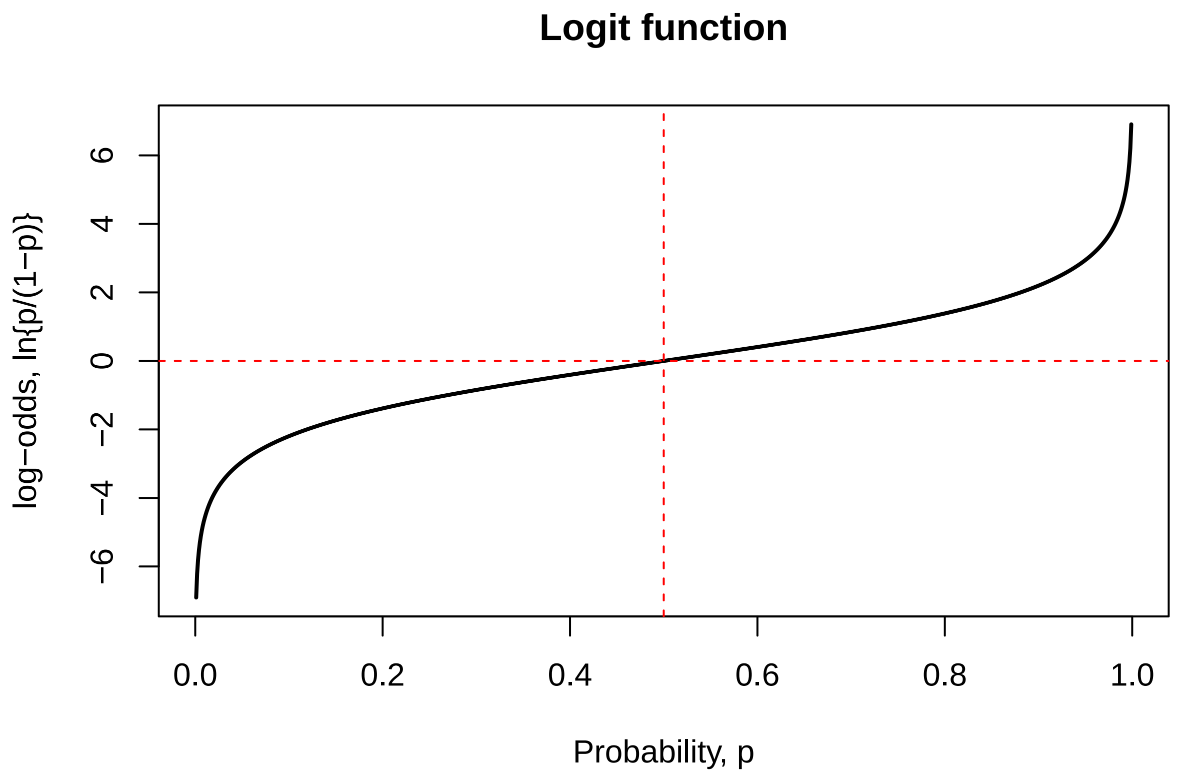
- The **odds** are defined as $\frac{p_i}{1-p_i}$
- The **log odds** are the natural logarithm of the odds
- The **odds ratio** for a unit increase in predictor X_i is e^{β_1}

The logit function is most helpful for taking probabilities and putting them on a scale that is more amenable to linear regression analysis.

Here is what it looks like.

```r
# Create probability values
p <- seq(from = 0.001, to = 0.999, length = 1000)
# Calculate log-odds
log.odds <- log(p/(1-p))

# Plot relationship
plot(p, log.odds, lwd = 2, typ = 'l',
     xlab = 'Probability, p',
     ylab = 'log-odds, ln{p/(1-p)}',
     main = 'Logit function')
```

```r
abline(v = 0.5, col = 'red', lty = 2)
abline(h = 0, col = 'red', lty = 2)
```



Checkpoint 2: What is the log-odds of a probability of 0.1, 0.9,0.25, 0.75, 0.5, 0.01, and 0.99?

Implementing Logistic Regression in R

Let's demonstrate logistic regression with an example. We've got data on tumor biopsies. Tumors that are biopsied are either labelled as malignant (M) or benign (B) and so can be modeled with logistic regression.

```r
# Load the data; remember to set your working directory
biopsy.df <- read.csv(file = "~/Library/CloudStorage/GoogleDrive-kortessn@wfu.edu/My Drive/Import/Wake 

# Look at the structure of the data
str(biopsy.df)
```

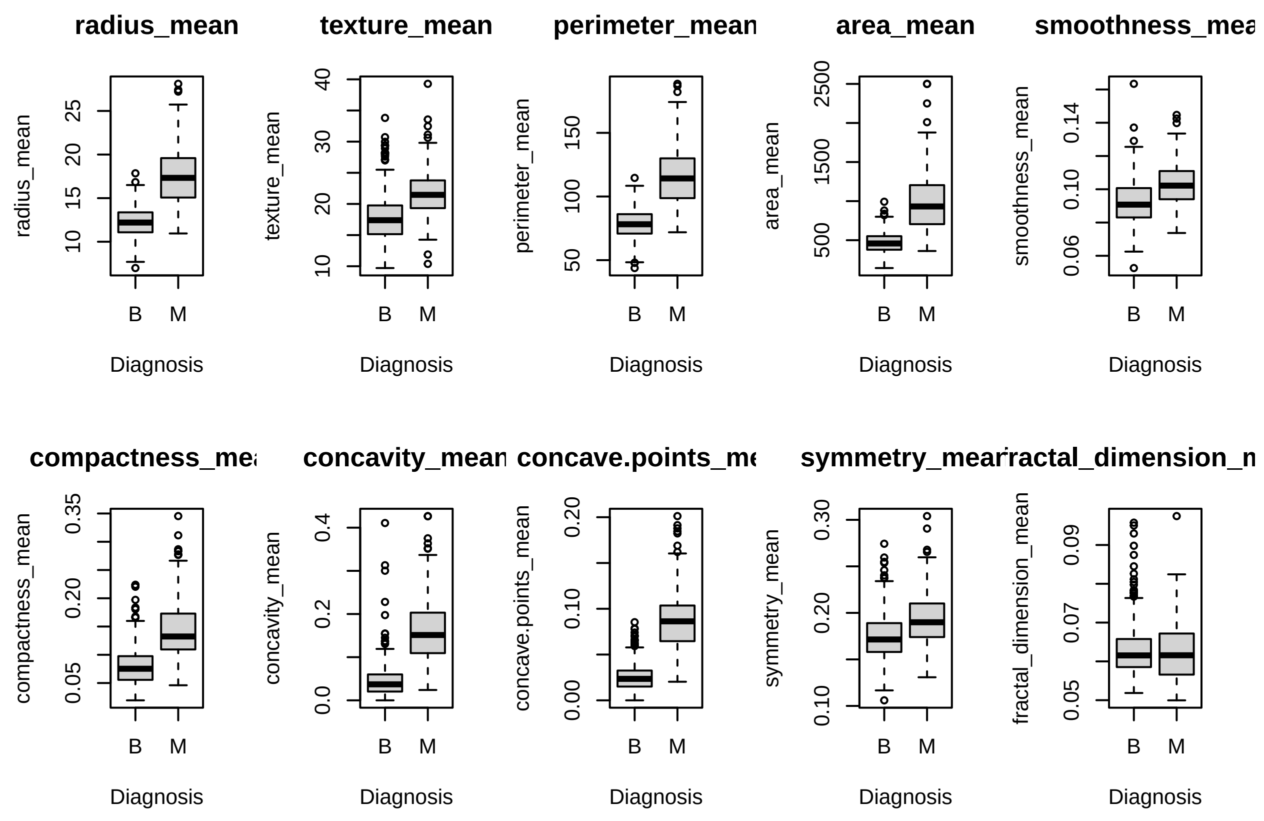
```
## 'data.frame':    569 obs. of  32 variables:
##  $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis              : chr  "M" "M" "M" "M" ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

There is a lot that is measured about the cells in these biopsies.

Data Exploration

Before modeling, let's explore the relationship between predictors and outcome. I don't have enough biological knowledge to know the factors associated with tumor malignancy. Since we have this data at hand, we can just look to see what it says. Let's look at the first 10 factors in the data set, which are all related to the mean character of cells in the biopsy.

```r
# Box plots for comparing distributions of predictors by biopsy diagnosis
par(mfrow = c(2, 5))
for(i in 3:12) {
  boxplot(biopsy.df[,i] ~ biopsy.df$diagnosis,
          main = names(biopsy.df)[i],
          xlab = "Diagnosis", ylab = names(biopsy.df)[i])
}
```



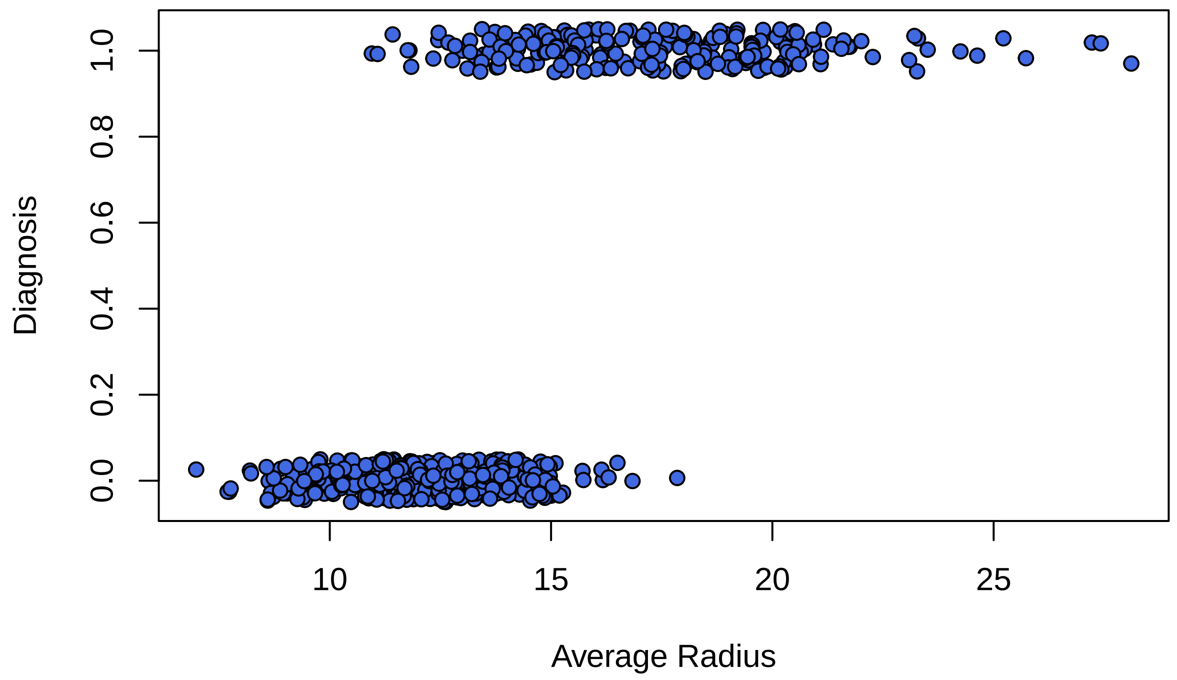
```
# Reset plotting parameters
par(mfrow = c(1, 1))
```

Building a Logistic Regression Model

Let's make a model using the mean radius of the cells in the tumor biopsy. The question here is how much knowing the radius of cells can be predictive of whether a tumor is ultimately labelled as malignant or benign.

Let's plot the data first.

```
# Convert the diagnosis column to 0 (benign) and 1 (malignant)
diagnosis01 <- ifelse(biopsy.df$diagnosis == "M", 1, 0)

plot(biopsy.df$radius_mean,
     jitter(diagnosis01, amount = 0.05), # jitter points a bit
     pch = 21, bg = 'royalblue',
     xlab = 'Average Radius',
     ylab = 'Diagnosis')
```



```r
# Fit logistic regression model
model <- glm(factor(diagnosis) ~ radius_mean,
             data = biopsy.df,
             family = binomial()) # Specify family as binomial to run logistic regression

# Summarize the model
summary(model)
```

```
##
## Call:
## glm(formula = factor(diagnosis) ~ radius_mean, family = binomial(),
##     data = biopsy.df)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -15.24587    1.32463  -11.51   <2e-16 ***
## radius_mean   1.03359    0.09311   11.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 751.44  on 568  degrees of freedom
## Residual deviance: 330.01  on 567  degrees of freedom
## AIC: 334.01
##
## Number of Fisher Scoring iterations: 6
```

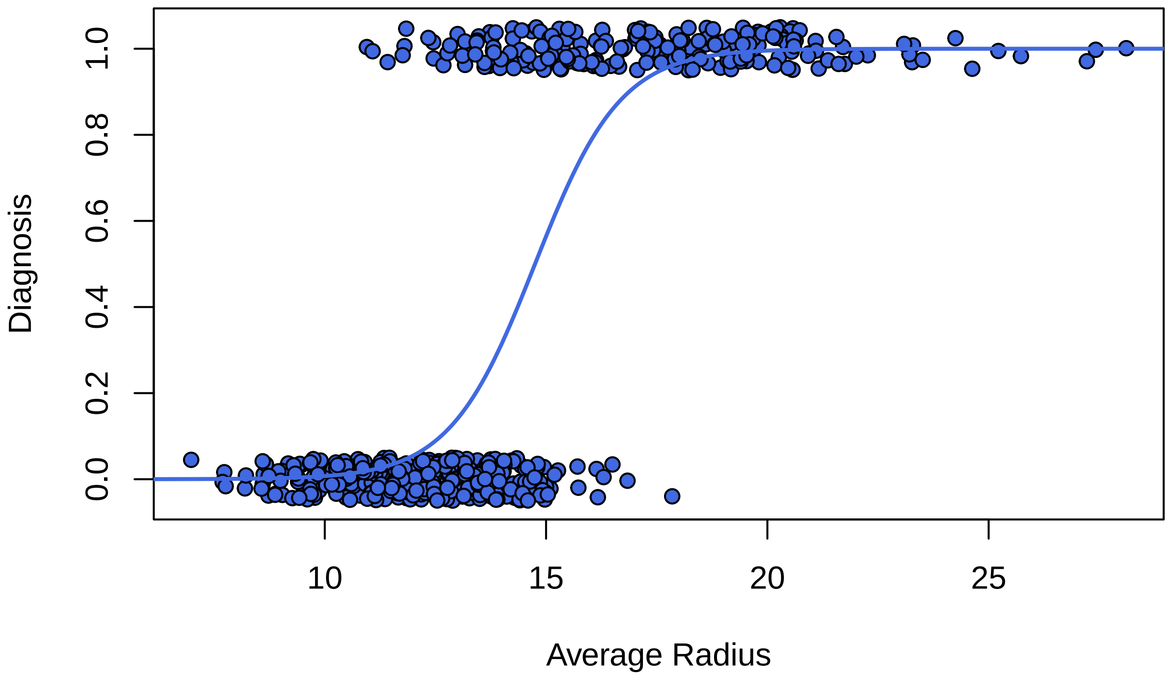
Plotting the best fit line

Let's add the best fit line. The best fit line on the log-odds scale is

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \hat{\beta}_0 + \hat{\beta}_1 X_i = -15.246 + 1.034 \times X_i$$

Let's make this line for a bunch of possible radii.

```r
# Make radii
radii <- seq(from = 0, to = 30, length = 1000)

# Get model coefficients
beta0 <- coef(model)[1]
beta1 <- coef(model)[2]

# Make log-odds line
log.odds.p <- beta0 + beta1*radii

# Convert to probability using inverse logit function
p <- exp(log.odds.p)/(1+exp(log.odds.p))
# Alternative: p <- 1/(1+exp(-log.odds.p))

# Recreate figure above
plot(biopsy.df$radius_mean,
     jitter(diagnosis01, amount = 0.05), # jitter points a bit
     pch = 21, bg = 'royalblue',
     xlab = 'Average Radius',
     ylab = 'Diagnosis')
# Add best fit line
lines(radii, p, lwd = 2, col = 'royalblue')
```



Interpreting Model Coefficients

The coefficients in logistic regression represent the change in log odds for a one-unit increase in the predictor variable. To convert to odds ratios:

```r
# Calculate odds ratios and 95% CI
odds_ratios <- exp(cbind(OR = coef(model), confint(model)))
```

```
## Waiting for profiling to be done...
```

```r
odds_ratios
```

```
##                       OR        2.5 %       97.5 %
## (Intercept) 2.392228e-07 1.472139e-08 2.689337e-06
## radius_mean 2.811136e+00 2.371618e+00 3.419651e+00
```

So the odds-ratio for the intercept is not that informative. But the odds-ratio for the radius of cells is. It tells us how much the odds of a malignant diagnosis change (in proportional terms) for every unit increase in the radius of the cells.

Remember, the odds-ratio is

$$\text{OR} = \frac{\text{odds malignant}(X+1)}{\text{odds maignant}(X)} = \frac{\frac{p(X+1)}{1-p(X+1)}}{\frac{p(X)}{1-p(X)}} = e^{\hat{\beta}_1}.$$

Here, the odds ratio is 2.81. This means that the odds of a malignant diagnosis increase by 181% for every unit increase in the mean radius of the cells in the biopsy. We come to this number because ratios, such as 2.81 can be written as *proportional changes* by first ocnverting them to percentages (2.81 = 281%) and then comparing that to the original by subtracting 100% (281% - 100% = 181% increase).

Another way to say this is that the odds nearly triple because 2.81 is close to 3.

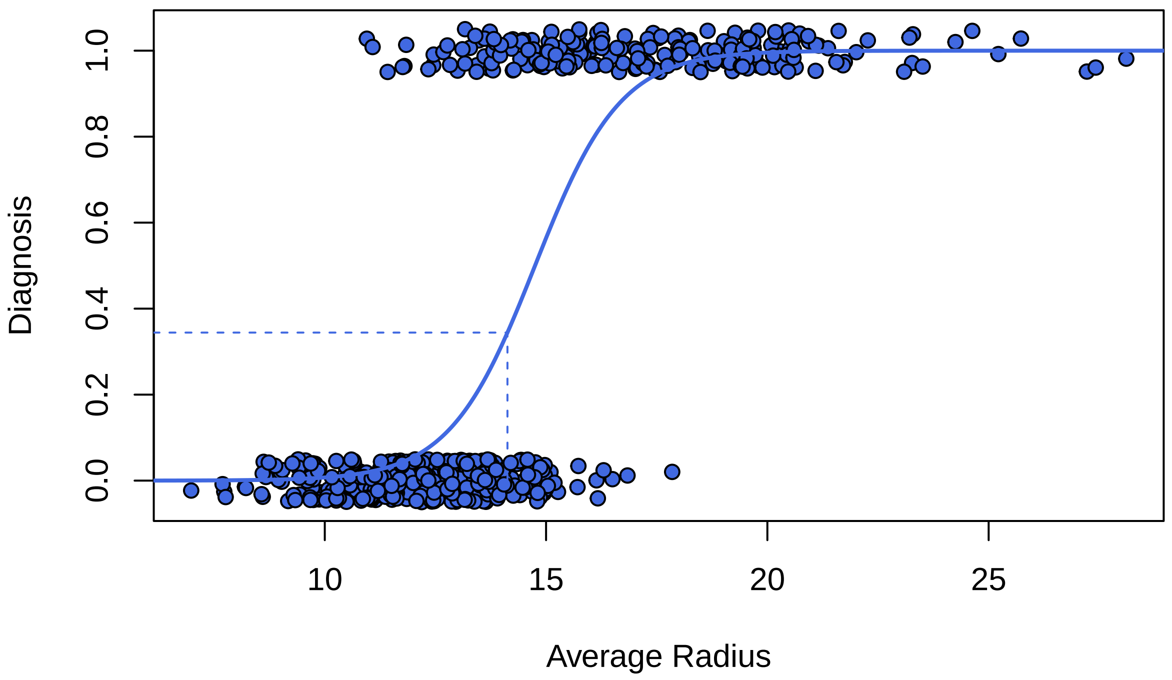
There are a couple of other things we can interpret from the model.

- What is the probability of a malignant diagnosis under average cell radius in the sample?
  - The equation for the log-odds is $\text{logit}(p_{\bar{X}}) = \beta_0 + \beta_1 \bar{X}$.

```r
# First calculate line fit at mean
log.odds.p.ave.radius <- coef(model)[1] + coef(model)[2]*mean(biopsy.df$radius_mean)

# Now convert from log-odds to probability
p.ave.radius <- exp(log.odds.p.ave.radius)/(1+exp(log.odds.p.ave.radius))
# Print result
print(p.ave.radius)
```

```
## (Intercept)
##   0.3443294
```

```r
# Plot on the figure
# Recreate figure above
plot(biopsy.df$radius_mean,
     jitter(diagnosis01, amount = 0.05), # jitter points a bit
     pch = 21, bg = 'royalblue',
     xlab = 'Average Radius',
     ylab = 'Diagnosis')
# Add best fit line
lines(radii, p, lwd = 2, col = 'royalblue')
# Add line corresponding to mean radius and probability of malignant biopsy
lines(c(rep(mean(biopsy.df$radius_mean),2), 0), # Xbar, Xbar, 0
      c(0,rep(p.ave.radius,2)),
      lty = 2,
      col = 'royalblue')
```



- We could also ask what the mean cell radius is that corresponds to even odds ($p = 0.5$) for tumor malignancy.
  - When $p = 0.5$, that means the odds are $p/(1-p) = 1$, and the log-odds are $\ln\{1\} = 0$. Hence, $0 = \beta_0 + \beta_1 X_{p=0.5}$. We can just solve for this equation as $X_{p=0.5} = -\beta_0/\beta_1$.

For our model, $X_{p=0.5} \approx 14.75$.

```
(X.even.odds <- -coef(model)[1]/coef(model)[2])
```

```
## (Intercept)
##    14.75042
```

Making Predictions

Making prediction is a bit easier in this case than in linear regression because we are directly modeling probabilities. Once we have the probabilities, we can just use them in a random number generator of the binomial distribution as we did before. For example, consider that we have an individual tumor with average radius size of $X = 20$.

The prediction from our logistic model is

$$\ln\left(\frac{p_i}{1-p_i}\right) = \hat{\beta}_0 + \hat{\beta}_1 \times 20 = -15.246 + 1.034 \times 20 = 5.426.$$

This corresponds to a probability of 99.56%. Here are both calculations in R.

```
# Calculate log-odds fit
log.odds.p.20 <- coef(model)[1] + coef(model)[2]*20
# Convert log-odds fit to probability
p.20 <- exp(log.odds.p.20)/(1+exp(log.odds.p.20))
print(p.20)
```

```
## (Intercept)
##   0.9956182
```

Now we can run this in a random number generator for the Bernoulli distribution. R doesn't have a Bernoulli name because a Binomial with $n = 1$ is a Bernoulli. We'll use that.

```r
set.seed(15)
(predict.20 <- rbinom(n = 1, size = 1, prob = p.20))
```

```
## [1] 1
```

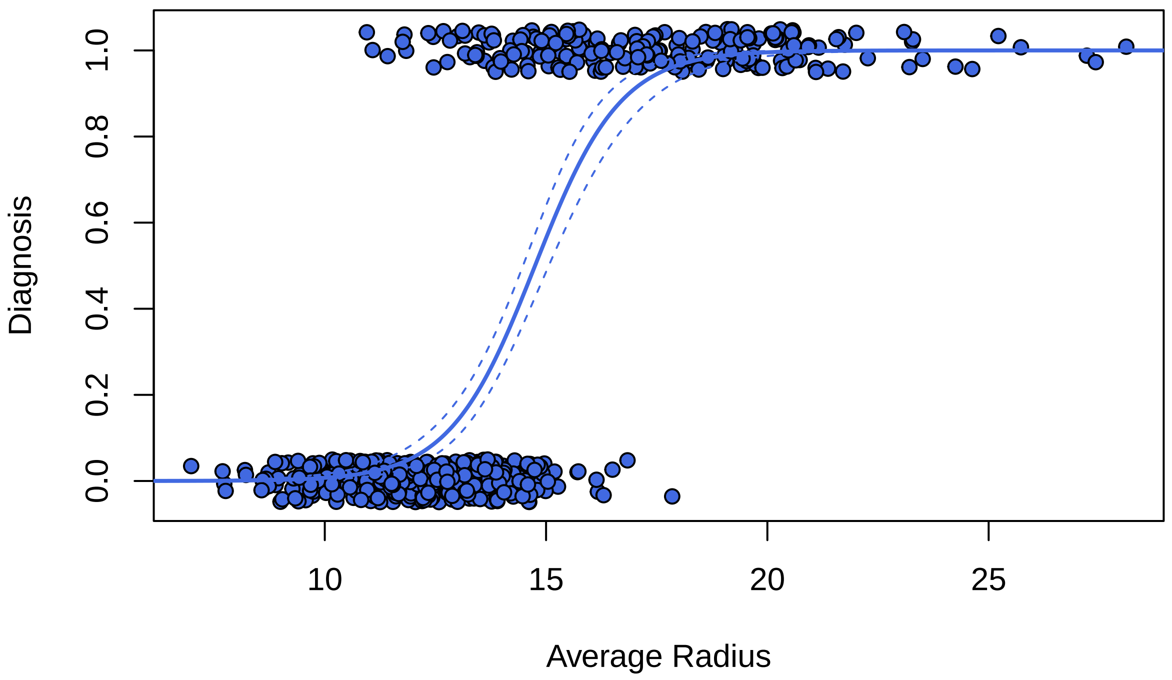
Confidence Intervals

There are confidence intervals associated with our predictions. Let's put those on as well. To do that, we will need the package "ciTools". If you don't have it, install it now.

```r
best.fit.line <- predict.glm(model,
                        newdata = data.frame(radius_mean = radii),
                        family = binomial(),
                        type = 'response')
# Get confidence intervals
# load library
library(ciTools)
```

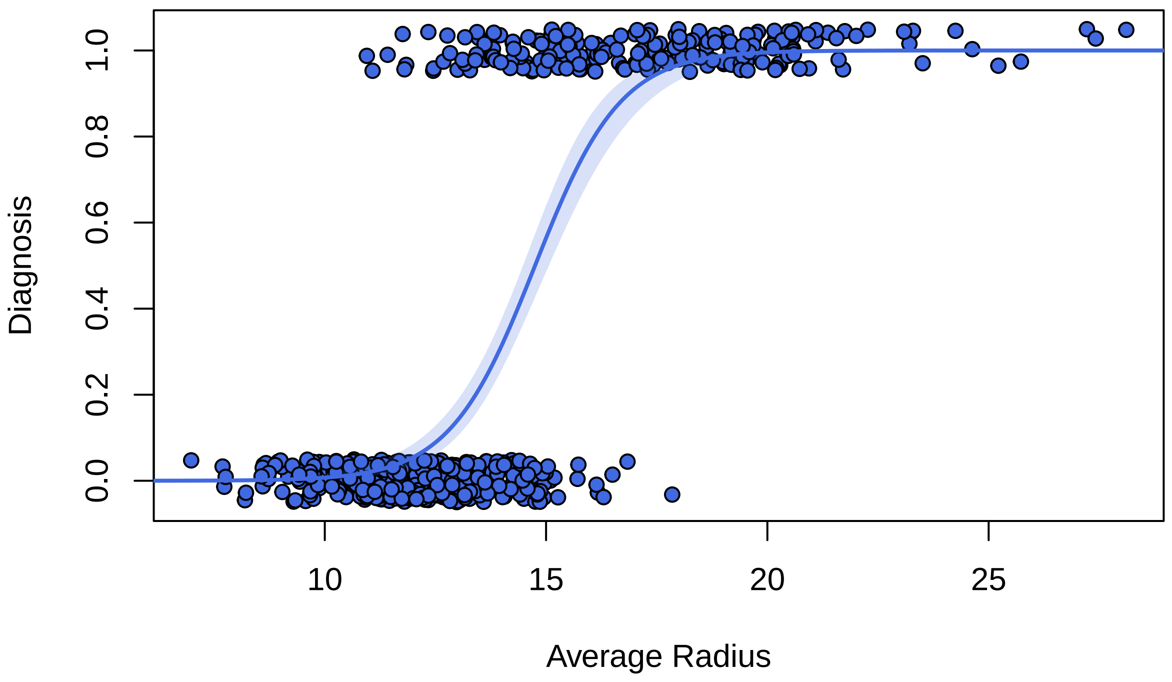
```
## ciTools version 0.6.1 (C) Institute for Defense Analyses
```

```r
# Ask for confidence intervals
line.ci <- add_ci(df = data.frame(radius_mean = radii),
                  fit = model,
                  alpha = 0.05)

# Plot the data
plot(biopsy.df$radius_mean,
     jitter(diagnosis01, amount = 0.05), # jitter points a bit
     pch = 21, bg = 'royalblue',
     xlab = 'Average Radius',
     ylab = 'Diagnosis')
# Add best fit line
lines(radii, best.fit.line,
      col = 'royalblue', lwd = 2)
# Add confidence intervals
lines(radii, # Radii values we fit
      line.ci$LCB0.025, # Lower confidence bound
      col = 'royalblue', lty = 2)
lines(radii, # Radii values we fit
      line.ci$UCB0.975, # Upper confidence bound
      col = 'royalblue', lty = 2)
```



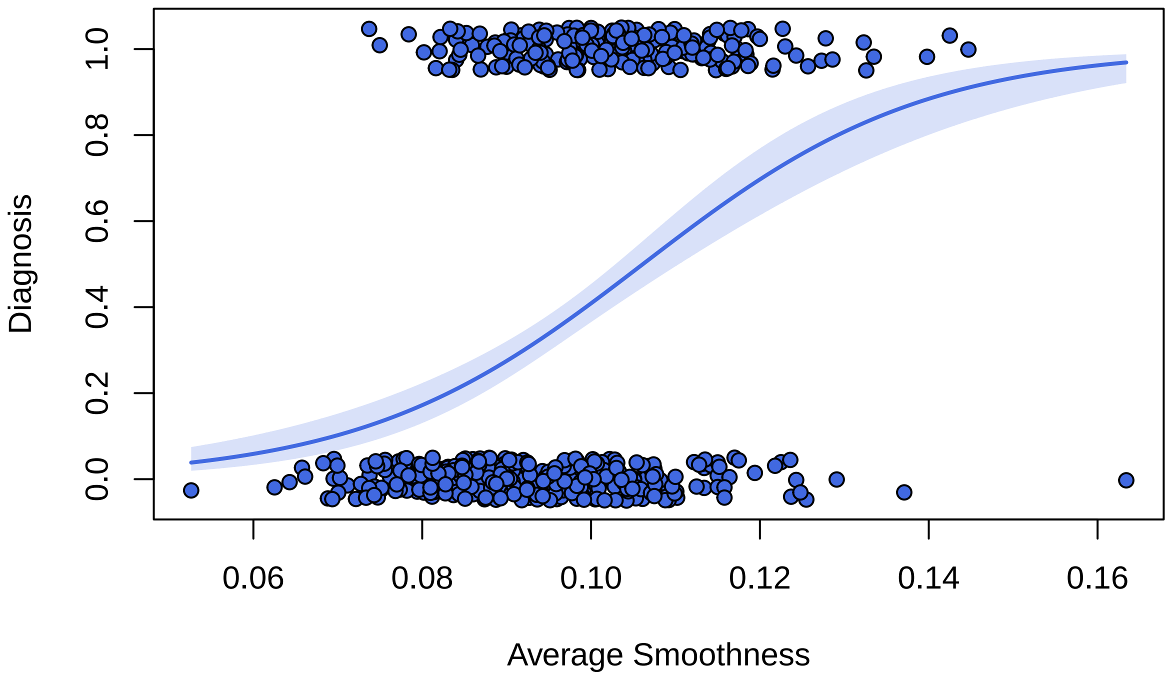
I'm actually a big fan of making the confidence interval look like a shaded area. I think it looks better. You can do this with the polygon function.

```r
# Plot the data
plot(biopsy.df$radius_mean,
     jitter(diagnosis01, amount = 0.05), # jitter points a bit
     pch = 21, bg = 'royalblue',
     xlab = 'Average Radius',
     ylab = 'Diagnosis')

# Add confidence intervals as a polygon
polygon(c(radii, rev(radii)),
        c(line.ci$LCB0.025, rev(line.ci$UCB0.975)),
        col = adjustcolor('royalblue', alpha.f = 0.2),
        border = NA)

# Add best fit line
lines(radii, best.fit.line,
      col = 'royalblue', lwd = 2)
```



That looks slick.

Checkpoint 3: Your turn. Run a logistic regression on the diagnosis from a tumor biopsy based on the smoothness mean of the sample. Use this to answer the following questions. 1. Is smoothness related to the diagnosis? 2. Are biopsies more likely to lead to a malignant diagnosis when samples are more or less smooth? 3. What is the odds ratio of a unit increase in the smoothness of a sample on a malignant diagnosis? 4. What is the probability of a malignant diagnosis for a sample with the average smoothness? 5. What is the smoothness of the tumor sample that leads to equal odds of a malignant diagnosis? Be sure to include a figure showing your results. Here is what my figure looks like.



Part 2: Maximum Likelihood Estimation (MLE)

The concept of likelihood

Logistic regression is not fit using least squares estimation. There are residuals in the model, but the residuals are clunky and are difficult to work with. Instead, logistic regression is fit by a method called "Maximum Likelihood Estimation" is a statistical method used to estimate the parameters of a probability distribution by maximizing a likelihood function. This approach follows the concept of finding the parameter values that make the observed data most probable.

Reminder that we have been working with a number of probability models; the Bernoulli, Binomial, Normal, t, χ^2, and F distributions are all examples. Each of these specifies the probability of a particular outcome given some distributional parameters. We can write this as

$$Pr(X = \text{some outcome} \mid \text{some parameters})$$

For example, if we have the Binomial distribution, the probability of getting an outcome of k successes depends on how many trials we run and what the underlying probability of success is. That is

$$Pr(X = k|n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

We can find this exactly in R using the function `dbinom(k,n,p)`. This is a great way to create expectations for outcomes to observe in the world (X) given specific hypotheses about the underlying biological and sampling processes, p and n, respectively.

Likelihood flips this on its head and asks "For a given sampling design and outcome, what parameters most probably gave rise to the data?"

To answer this question, we write

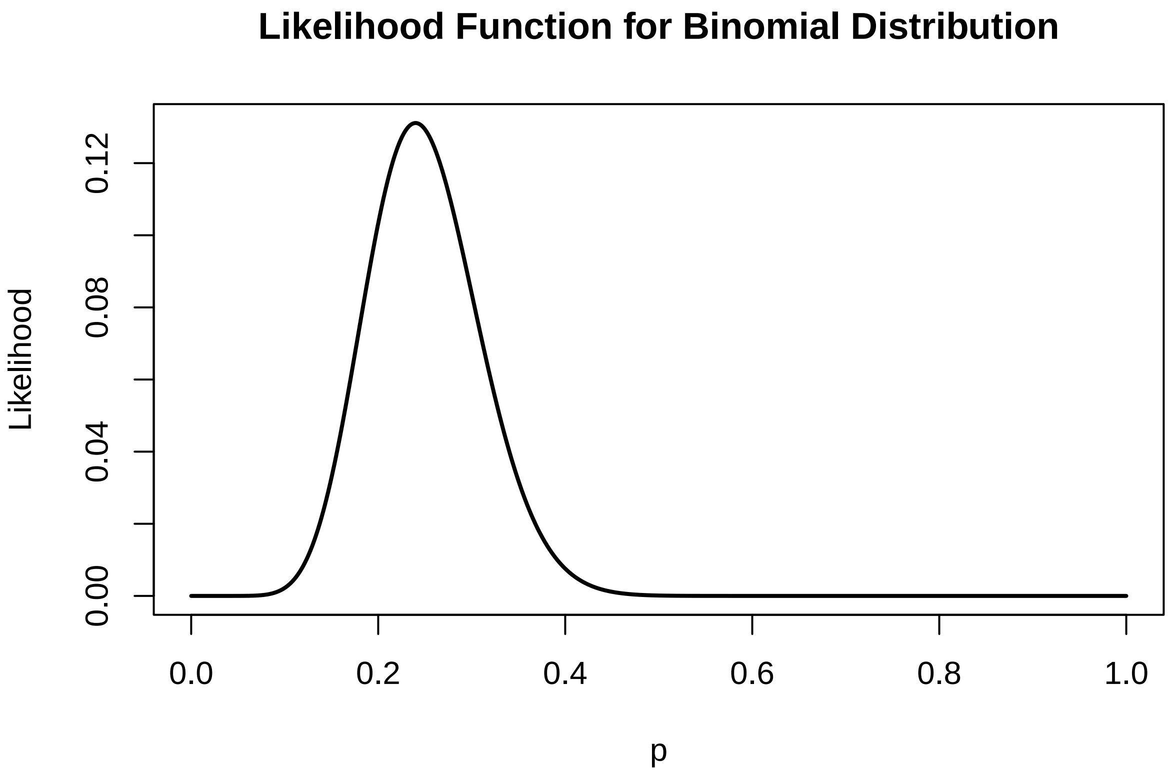
$$L(\text{some parameters } |X = \text{some outcome})$$

For Binomial data, the likelihood function is just the probability function, but where p is unknown. For example, if we sampled 50 individuals and found 12 successes, the likelihood of p is

$$L(p|n = 50, k = 12) = \binom{50}{12} p^{12} (1-p)^{50-12}$$

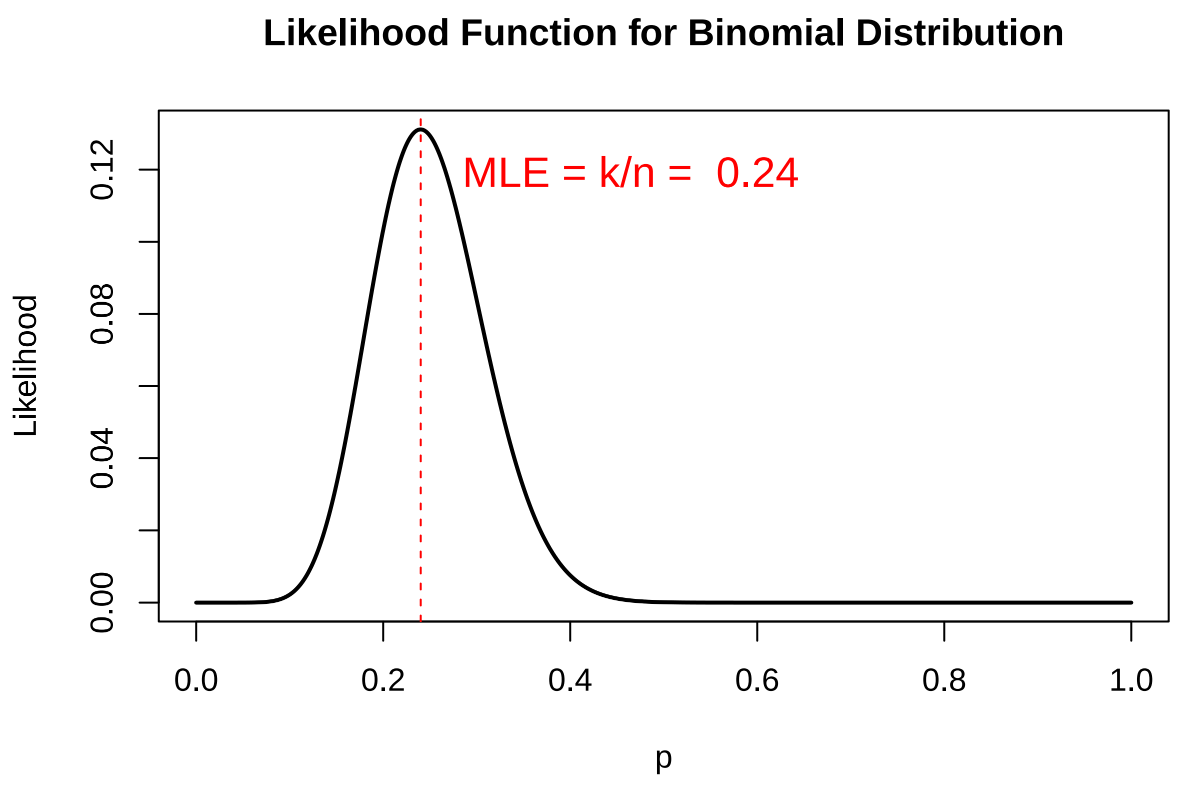
This is a function of p because p is the only unknown in the equation. We can see what this function looks like by picking a bunch of values for p and calculating the likelihood using the function `dbinom`.

```r
# Define the number of trials (n) and the number of successes (k)
n <- 50
k <- 12

# Define a sequence of probabilities (p) to evaluate the likelihood function
p <- seq(0, 1, length = 2000)

# Compute the likelihood function for a binomial distribution
likelihood <- dbinom(k, n, p)

# Plot the likelihood function
plot(p, likelihood, type = "l", lwd = 2,
     xlab = "p", ylab = "Likelihood",
     main = "Likelihood Function for Binomial Distribution")
```



Checkpoint 4: Make the likelihood function for 4 cases: i. $(n = 5, k = 1)$, ii. $(n = 500, k = 100)$, iii. $(n = 5, k = 4)$, and iv. $(n = 500, k = 400)$. How does the likelihood function change with n? What does this mean about the role of sample size in the likelihood function? Now that we have the likelihood function, we can answer the question of what value of p is the one that makes the data we observed **most likely**? This is simply the value of p for which the likelihood function is the largest. The value of p that has the highest likelihood is called the **Maximum Likelihood Estimator** or MLE for short. Let's find it and add it to the plot

```r
# Calculate the Maximum Likelihood Estimate (MLE) for p
p.MLE <- p[likelihood == max(likelihood)]

# Plot the likelihood function
plot(p, likelihood, type = "l", lwd = 2,
     xlab = "p", ylab = "Likelihood",
     main = "Likelihood Function for Binomial Distribution")

# Add a vertical line at the MLE
abline(v = p.MLE, col = "red", lty = 2)

# Annotate the plot with the MLE value
text(p.MLE * 1.1, max(likelihood) * 0.9,
     paste("MLE = k/n = ", round(p.MLE, 2)),
     pos = 4, col = "red", cex = 1.3)
```



This is the long way of finding the MLE. The quick way is to look in the most general case where we imagine we know k and n, but we don't know p and use calculus to find the highest point of the function. When you do that, you find that the MLE for p is

$$\hat{p}_{MLE} = \frac{k}{n},$$

which is just the fraction of successes in your sample.

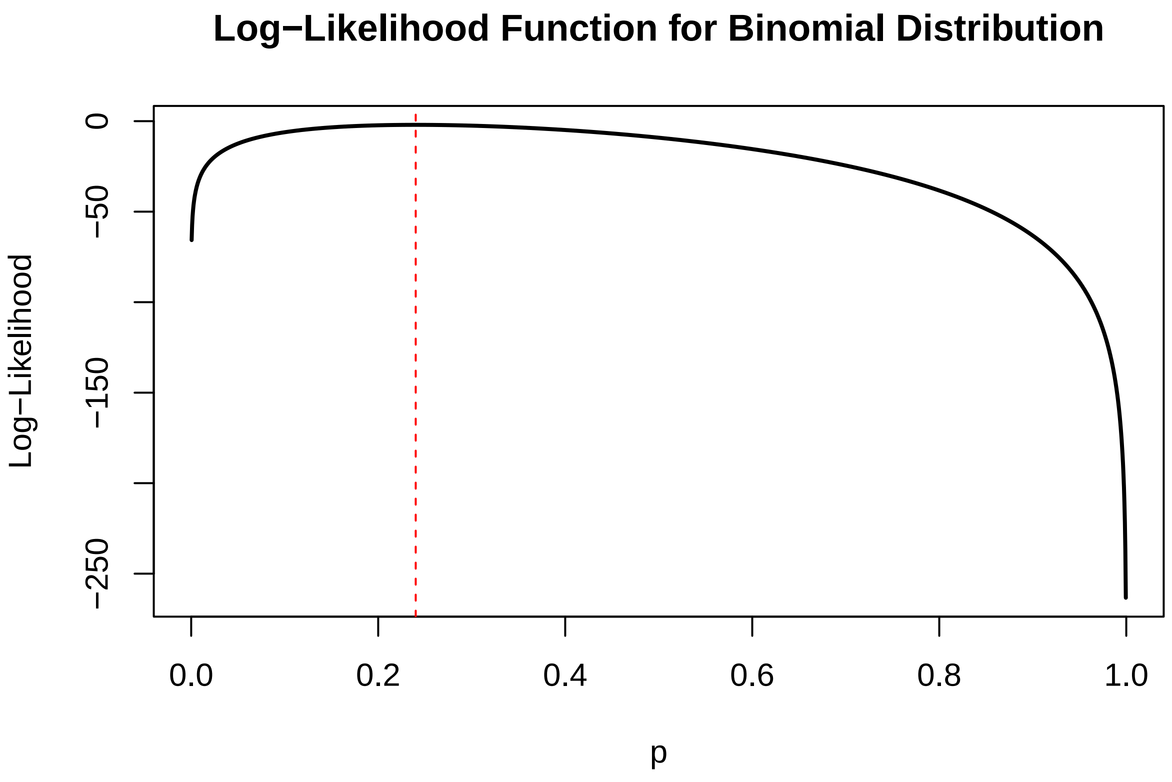
The likelihood function is the key to doing maximum likelihood estimation. It turns out that sometimes it easier to think about the **log-likelihood** rather than the likelihood function. For a very large class of probability models and distributions, doing things on the log scale is *much* easier. We often write log-likelihoods with a script l, like

$$\ell(p|n,k) = \ln L(p|n,k).$$

This often visually shows much more of the variation in the likelihood function as well. Luckily, maximizing the log-likelihood is the same as maximizing the likelihood function.

Here is the log-likelihood with the MLE

```r
plot(p, log(likelihood), type = "l", lwd = 2,
     xlab = "p", ylab = "Log-Likelihood",
     main = "Log-Likelihood Function for Binomial Distribution")
# Add MLE
abline(v = p.MLE, col = "red", lty = 2)
```



Normal Distribution Example

You can do likelihood with any known probability model. One we work with a lot is the normal distribution. The normal distribution is a continuous probability distribution with parameters μ, the mean, and σ, the standard deviation. If we know μ and σ, then we can find the probability of different outcomes.

For example, if we have a set of randomly sampled data, $X = \{x_1 = 5, x_2 = 7, x_3 = 4.5\}$ that we think is normally distributed, the probability of getting these three if $\mu = 5$ and $\sigma = 1$ is

$$Pr(X|\mu = 5, \sigma = 1) = Pr(x_1 = 5|\mu = 5, \sigma = 1) \times Pr(x_2 = 7|\mu = 5, \sigma = 1) \times Pr(x_3 = 4.5|\mu = 5, \sigma = 1)$$

by the fact that the X_i are independent and so we can multiply their probabilities together. We just calculate each of these using `dnorm(x, mu, sigma)`.

```r
X <- c(5,7,4.5)
mu <- 5
sigma <- 1

# Find probabilities of each x value
dnorm(X, mu, sigma)
```

```
## [1] 0.39894228 0.05399097 0.35206533
```

```r
# Find the probability of the set
prod(dnorm(X, mu, sigma))
```

```
## [1] 0.007583233
```

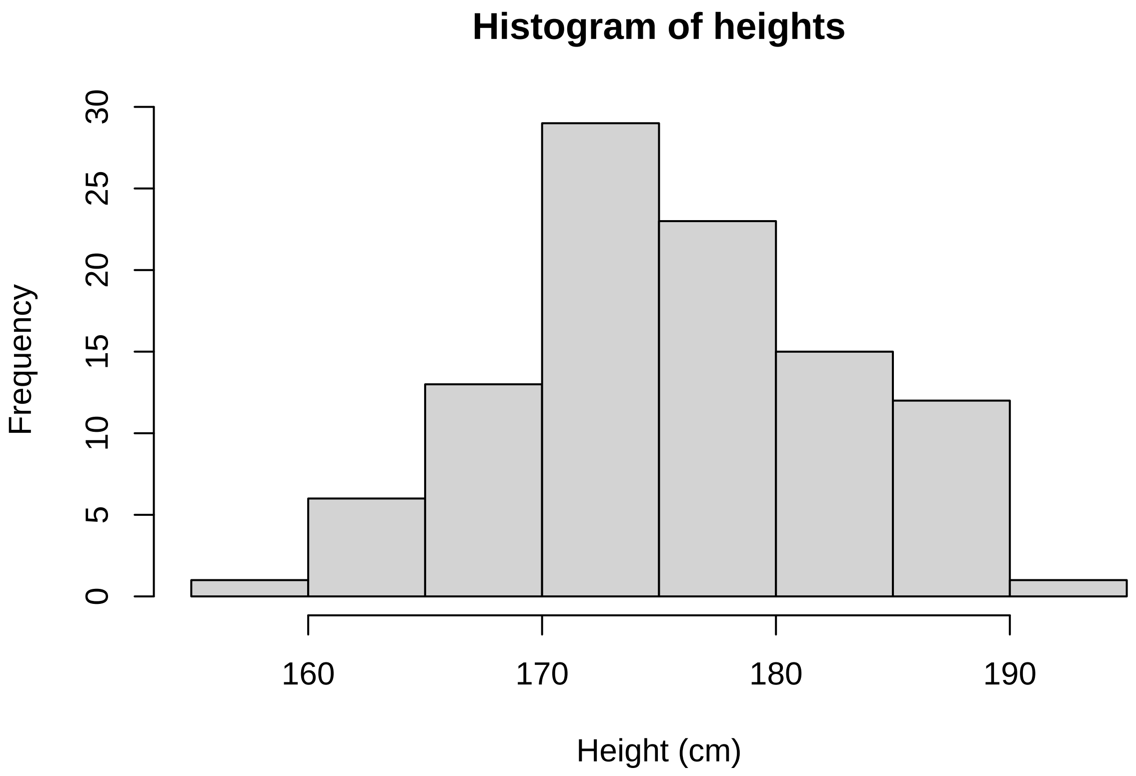
This is the likelihood of the three of them together, i.e., the likelihood that the data X came from the normal distribution with mean $\mu = 5$ and standard deviation $\sigma = 1$.

More generally, the likelihood function for a normal distribution is given by:

$$L(\mu, \sigma|X) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X_i-\mu)^2}{2\sigma^2}}$$

where X is the observed data, μ is the mean, and σ is the standard deviation. To see how this works in practice, let's create some sample data from a normal distribution and find the likelihood function for μ and σ given this data.

```r
# Example with human height
# Simulate height data (in cm)
set.seed(456)
true_mean <- 175     # True mean height in cm
true_sd <- 7         # True standard deviation in cm
n_subjects <- 100    # Number of subjects
heights <- rnorm(n_subjects, true_mean, true_sd)

# Plot a histogram of the data
hist(heights, xlab = 'Height (cm)')
```



```r
# Summary Statistics
summary(heights)
```

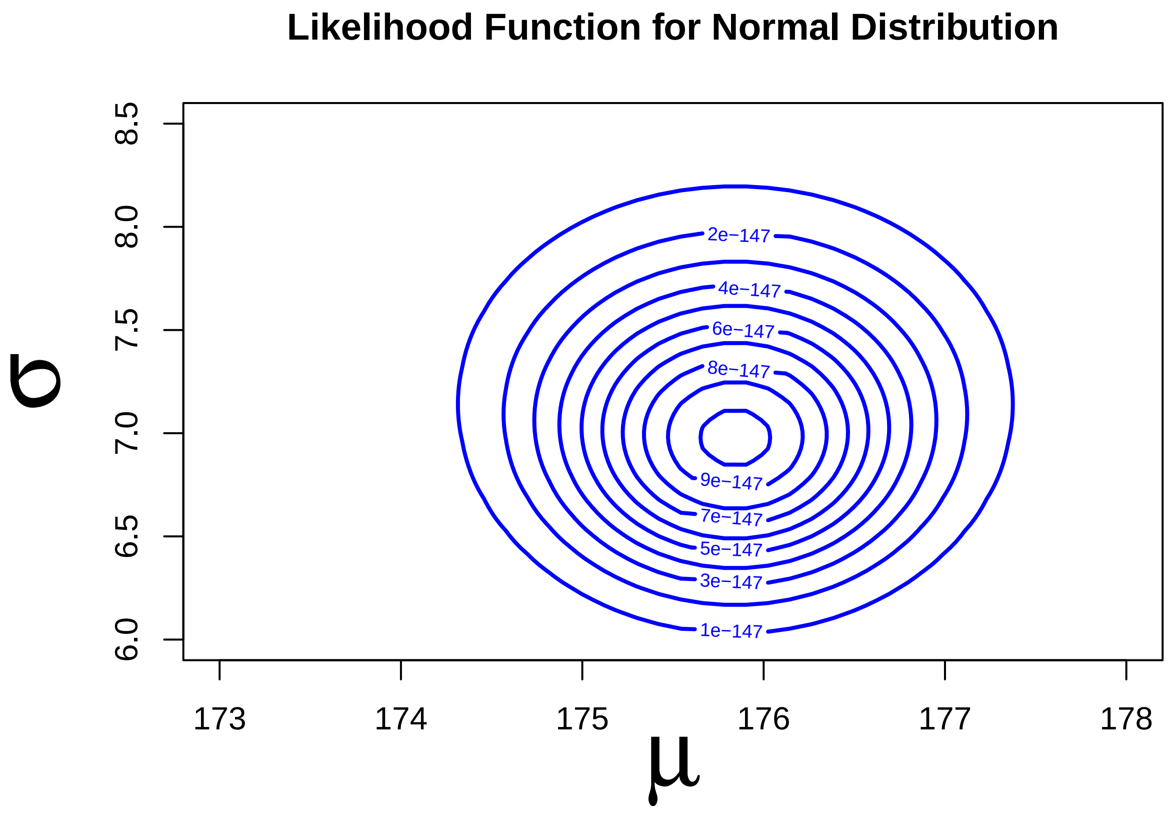
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   159.2   171.3   175.5   175.8   180.6   191.0
```

```r
# Define a range of values for mu and sigma to try out
mu <- seq(160, 190, length = 250)
sigma <- seq(4, 11, length = 250)

# Create a matrix to store the likelihood values
```

```r
likelihood <- matrix(0, nrow = length(mu), ncol = length(sigma))

# Compute the likelihood for each combination of mu and sigma
for (i in 1:length(mu)) {
  for (j in 1:length(sigma)) {
    likelihood[i, j] <- prod(dnorm(heights, mean = mu[i], sd = sigma[j]))
  }
}
```

Now we have the likelihood. This likelihood depends on both the values of μ and σ. A particular value of σ may be likely for some values of μ but not for others. Thus, we need a graph that shows both together. A way to think about this that μ and σ are latitude and longitudes and the likelihood is the elevation of any point on land. We want to find the point with the highest elevation. Here is a graph that mimics this thinking and displays the likelihood as a contour plot. Interpret this contour plot just like a topographical map you might look at for hiking.

```r
# Contour plot of the likelihood function
par(mar = c(4, 5, 4, 2))
contour(mu, sigma, likelihood,
        xlab = expression(mu), ylab = expression(sigma),
        main = "Likelihood Function for Normal Distribution",
        col = "blue", lwd = 2, cex.lab = 3,
        xlim = c(173,178), ylim = c(6,8.5))
```



Now that we have the map, we want to find the point on the map with the highest likelihood. We can ask R for this in the same way we asked R for the maximum likelihood estimate of p in the binomial example.

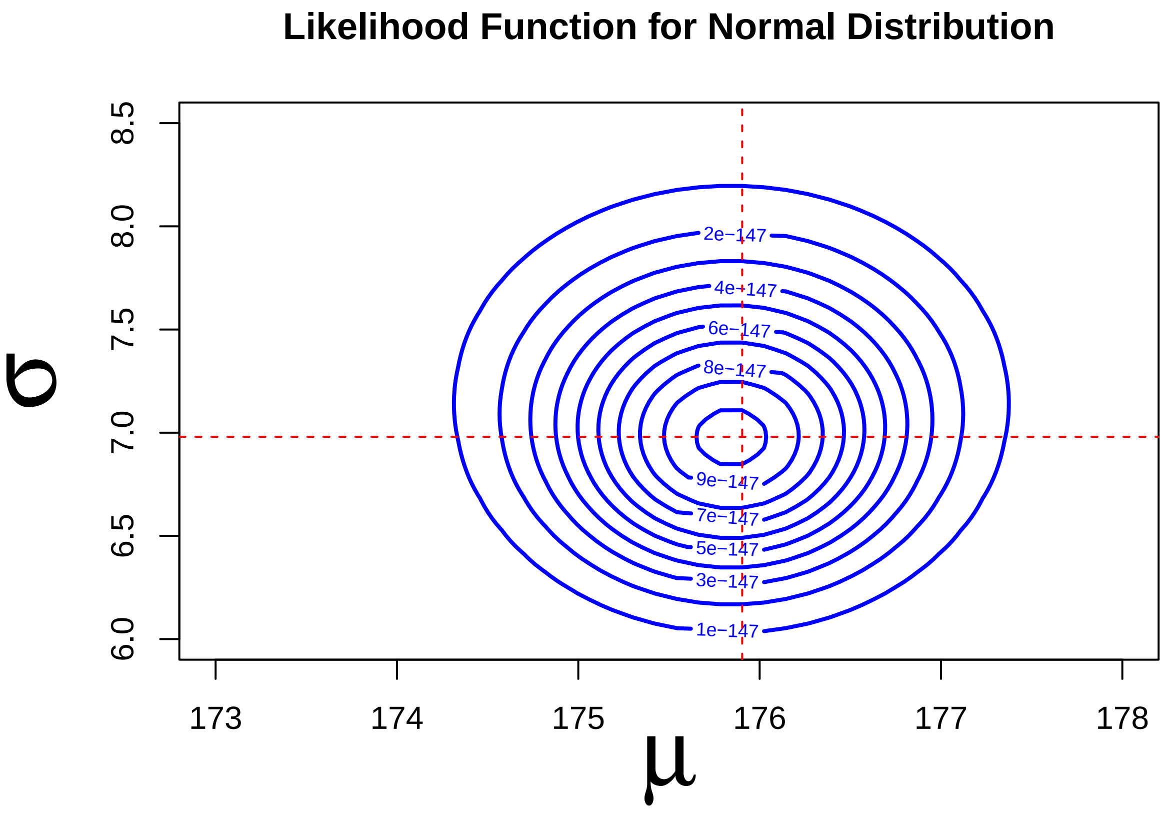
```r
# Find the Maximum Likelihood Estimates (MLEs) for mu and sigma
max_index <- which(likelihood == max(likelihood))
```

```r
row_index <- (max_index - 1) %% nrow(likelihood) + 1
col_index <- ceiling(max_index / nrow(likelihood))
mu.MLE <- mu[row_index]
sigma.MLE <- sigma[col_index]

print(c(mu.MLE, sigma.MLE))
```

```
## [1] 175.90361   6.97992
```

Now let's point out these maximum likelihood estimators on the plot.

```r
# Contour plot of the likelihood function
par(mar = c(4, 5, 4, 2))
contour(mu, sigma, likelihood,
        xlab = expression(mu), ylab = expression(sigma),
        main = "Likelihood Function for Normal Distribution",
        col = "blue", lwd = 2, cex.lab = 3,
        xlim = c(173, 178), ylim = c(6, 8.5))
# Annotate the contour plot with the MLE values
abline(v = mu.MLE, col = "red", lty = 2)
abline(h = sigma.MLE, col = "red", lty = 2)
```



You can see that they are right at the peak of the map, which is the parameter combination of μ and σ that gives the highest probability of generating the data.

Analytical Solution for Normal MLE For the normal distribution, the MLEs for μ and σ are

$$\hat{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}$$

and

$$\hat{\sigma}_{MLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2} = \sqrt{\frac{n-1}{n}}s.$$

You can see that the maximum likelihood estimate for the mean of a normal population is just the sample mean. The maximum likelihood estimate for the standard deviation of a normal population is just sum of squared deviations from the divided by n, rather than $n-1$ as is the case for the sample standard deviation.

```r
# Analytical MLE for normal parameters
mle_mean <- mean(heights)
mle_sd <- sqrt(mean((heights - mle_mean)^2))  # Using n divisor for MLE

cat("MLE estimate for mean:", round(mle_mean, 2), "cm\n")
```

```
## MLE estimate for mean: 175.84 cm
```

```r
cat("MLE estimate for standard deviation:", round(mle_sd, 2), "cm\n")
```

```
## MLE estimate for standard deviation: 6.98 cm
```

Another way to view this is how much one estimator compares to another by taking their ratio:

$$\frac{\hat{\sigma}_{MLE}}{s} = \sqrt{\frac{n-1}{n}}.$$

Checkpoint 5: Which value is larger, the sample standard deviation, s, or the MLE, $\hat{\sigma}_{MLE}$? When is their difference large? When is their difference small?