Testing and Model Selection with Likelihoods

Womack

2/13/2022

1 Likelihood Ratio Test

In the second problem in Homework 2, you worked through the asymptotic distribution for the test statistic of the likelihood ration test in two specific examples. Here, we are going to state some general results and look at a few examples.

The test we are going to discuss is a point null hypothesis versus an alternative. That is, we will have a null hypothesis $H_0 : \theta = \theta_0$ and an alternative hypothesis $H_A : \theta \neq \theta_0$. This is easiest when θ is a scalar, but works more generally for vector parameters.

The parameter θ will have some sort of support set Θ in which it can take values. For what we say to be true, we need θ_0 to be in the interior of Θ. Otherwise, the asymptotics break down.

In the sitution of a point null hypothesis for a scalar parameter, the test statistic is defined as

$$\lambda = 2\ell\left(\widehat{\theta}; y_1, \ldots, y_n\right) - 2\ell\left(\theta_0; y_1, \ldots, y_n\right). \tag{1}$$

In the situation where we are testing a point null for θ and there are other parameters, we will call these extra parameters ϕ. The MLE for ψ when the null is true will be denoted by $\widehat{\phi}_0$ and when the alternative is true, it will be $\widehat{\phi}_A$. The test statistic is

$$\lambda = 2\ell\left(\widehat{\theta}, \widehat{\phi}_A; y_1, \ldots, y_n\right) - 2\ell\left(\theta_0, \widehat{\phi}_0; y_1, \ldots, y_n\right). \tag{2}$$

1.1 Motivating Example: Bernoulli Data

Going back to the Bernoulli data examples from the `likelihoods.pdf` notes. For the `stage_old` and `stage` data, we will test the point null of $\theta = 0.5$ and for the `dud` data, we will test the point null of $\theta = 0.25$. We will also test whether the probability for `old` and `young` is the same in the stage data. This is a point null of $\theta_{old} - \theta_{young} = 0$ and is will be our first example of a model with multiple parameters where we are testing whether a submodel represented by a point null hypothesis can be rejected.

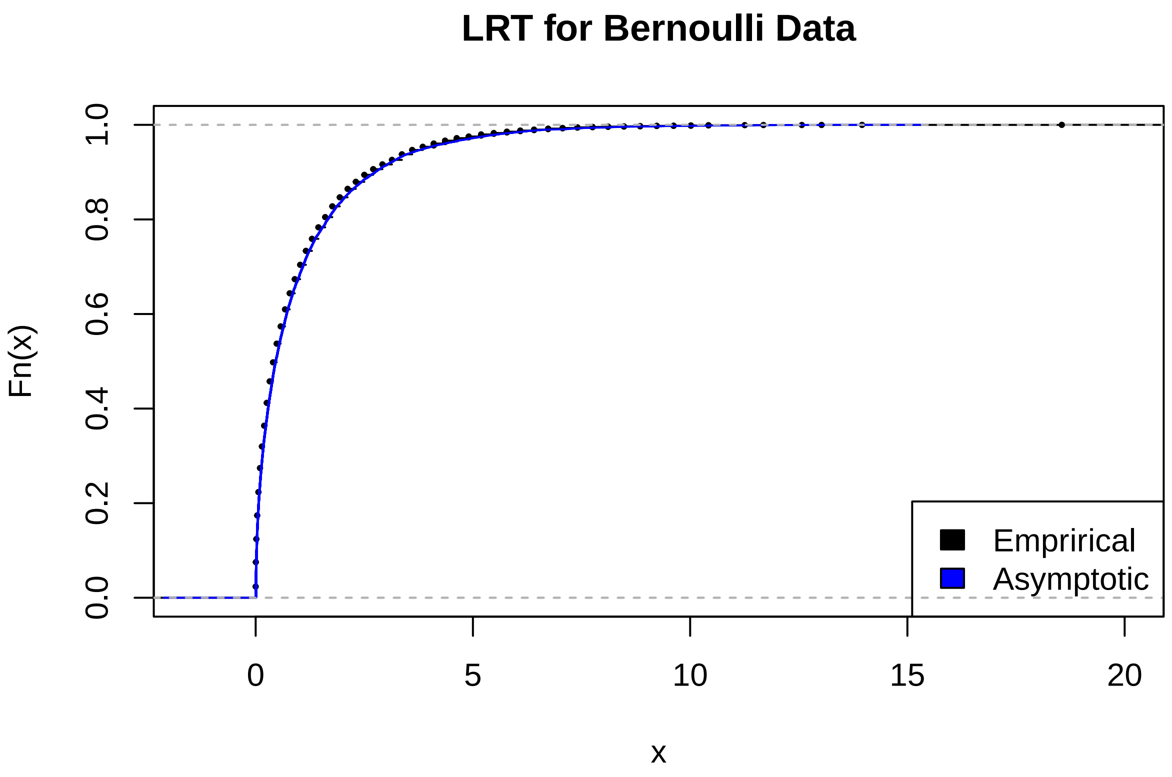
1.1.1 Bernoulli Point Null Hypothesis

Suppose we have iid Bernoulli(θ) data $y_1, \ldots, y_n$. We already know that the MLE is $\widehat{\theta} = \overline{y}$. We need to compute the test statistic λ.

$$
\begin{aligned}
\lambda &= 2\ell\left(\widehat{\theta}; y_1, \ldots, y_n\right) - 2\ell\left(\theta_0; y_1, \ldots, y_n\right) \\
&= 2\sum y_i \log\left(\widehat{\theta}\right) + 2\left(n - \sum y_i\right)\log\left(1-\widehat{\theta}\right) - 2\sum y_i \log\left(\theta_0\right) + 2\left(n - \sum y_i\right)\log\left(1-\theta_0\right) \\
&= 2n\widehat{\theta}\log\left(\widehat{\theta}\right) + 2n\left(1-\widehat{\theta}\right)\log\left(1-\widehat{\theta}\right) - 2n\widehat{\theta}\log\left(\theta_0\right) + 2n\left(1-\widehat{\theta}\right)\log\left(1-\theta_0\right) \\
&= 2n\left[\widehat{\theta}\log\left(\frac{\widehat{\theta}}{\theta_0}\right) + \left(1-\widehat{\theta}\right)\log\left(\frac{1-\widehat{\theta}}{1-\theta_0}\right)\right]
\end{aligned} \tag{3}
$$

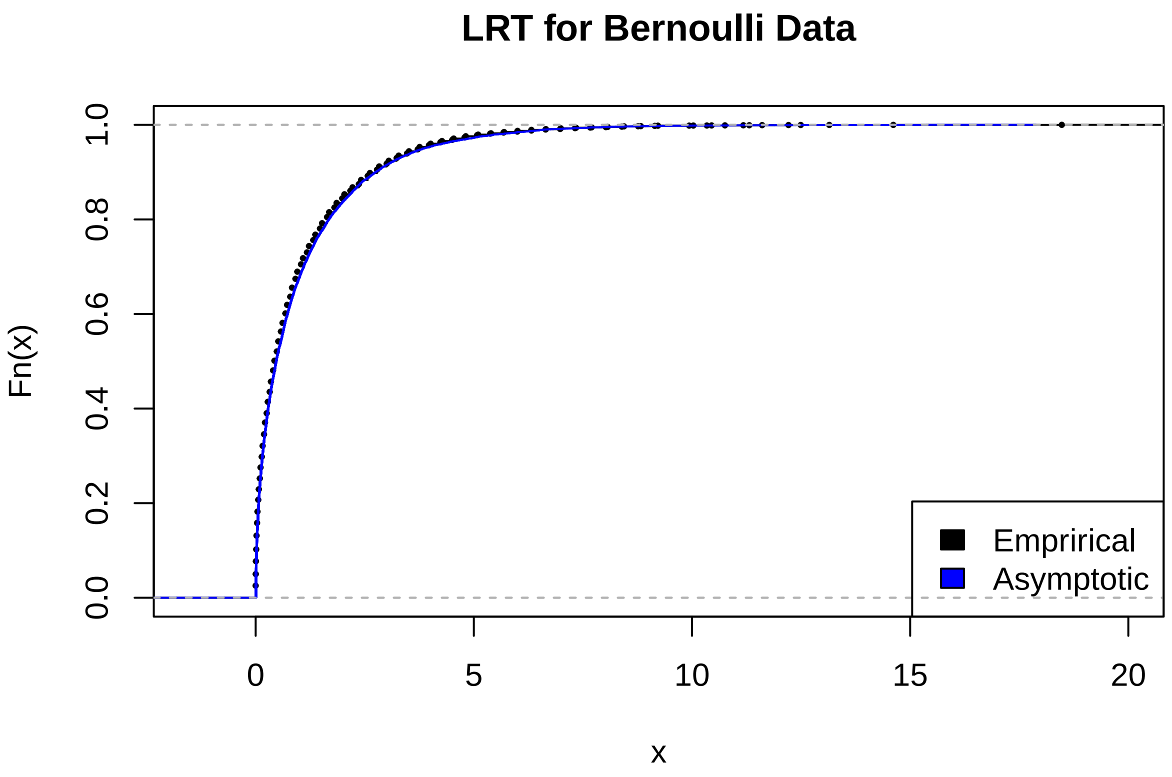
At face value, when the null is true, I have absolutely no freaking idea what the distribution of this statistic should be. For now, let's take n to be some large number, like 1000 and $\theta_0 = 0.5$ and just simulate the heck out of it and see it if comports with what I will tell you it is, χ^2_1.

```
n = 1000
N_sims = 10000
theta_0 = 0.5
lambda = rep(NA,N_sims)
for(i in 1:N_sims){
  y = rbinom(n,1,theta_0)
  theta_hat = mean(y)
  lambda[i] = 2*n*theta_hat*log(theta_hat/theta_0)+
    2*n*(1-theta_hat)*log((1-theta_hat)/(1-theta_0))
}
plot(ecdf(lambda),cex=0.3,main="LRT for Bernoulli Data")
yy = rchisq(10000,1)
lines(ecdf(yy),col="blue",cex=0)
legend("bottomright",c("Emprirical","Asymptotic"),fill=c("black","blue"))
```



The two cdfs coincide well, except for the fact that the empirical cdf for λ is a pure step function because $\widehat{\theta} = \overline{y}$ can only take a finite number of values. We can do this same thing for a different value of θ_0 to see that it does not depend on θ_0.

```r
n = 1000
N_sims = 10000
theta_0 = runif(1)
lambda = rep(NA,N_sims)
for(i in 1:N_sims){
  y = rbinom(n,1,theta_0)
  theta_hat = mean(y)
  lambda[i] = 2*n*theta_hat*log(theta_hat/theta_0)+
    2*n*(1-theta_hat)*log((1-theta_hat)/(1-theta_0))
}
plot(ecdf(lambda),cex=0.3,main="LRT for Bernoulli Data")
yy = rchisq(10000,1)
lines(ecdf(yy),col="blue",cex=0)
legend("bottomright",c("Emprirical","Asymptotic"),fill=c("black","blue"))
```



This tells us two things. First, the asymptotic distribution does not depend on θ_0. Second, because we know the asymptotic distribution for any θ_0, the test statistic λ is a pivot (we know its distribution).

Let's go ahead and compute these test statistics for our three datasets,

```
library(boot)
data(nodal)
stage_old = nodal$stage[nodal$aged==1]
stage = nodal$stage
data(aids)
dud = aids$dud

theta_hat = mean(stage_old)
n = length(stage_old)
theta_0 = 0.5
lambda_stage_old = 2*n*theta_hat*log(theta_hat/theta_0)+
  2*n*(1-theta_hat)*log((1-theta_hat)/(1-theta_0))

theta_hat = mean(stage)
n = length(stage)
theta_0 = 0.5
lambda_stage = 2*n*theta_hat*log(theta_hat/theta_0)+
  2*n*(1-theta_hat)*log((1-theta_hat)/(1-theta_0))

theta_hat = mean(dud)
n = length(dud)
theta_0 = 0.25
```

```
lambda_dud = 2*n*theta_hat*log(theta_hat/theta_0)+
  2*n*(1-theta_hat)*log((1-theta_hat)/(1-theta_0))

test_stats = c(stage_old=lambda_stage_old,stage=lambda_stage,dud=lambda_dud)
print(test_stats)
```

```
##   stage_old       stage         dud
##  0.16686011  0.01886904 14.06715260
```

We know what distribution these statistics are from (asymptotically at least), a χ^2_1. How do we make this into a test. Well, first we have to define a significance level α. Let's set it at $\alpha = 0.01$. Second, we need to determine what rejection means in terms of the statistic. The alternative hypothesis is a less restricted version of the null hypothesis, which means that the alternative has to fit the data better than the null. Even when the null is true, the alternative has to fit better. When that fit is extremely better (when λ is large), then the amount that the alternative fits better (quantified by λ) is less likely to be from random chance. This means, the rejection region for the test is large values of λ. Third, we need to compare to this cutoff or compute the p-value and compare to the significance level. The p-value is the going to be the probability under the null of getting a statistic larger than the observed statistic.

Let's compute the cutoff using the asymptotic χ^2_1 distribution and also compute the p-values using this asymptotic distribution.

```
alpha = 0.01
cutoff = qchisq(1-alpha,1) # 1-alpha is below cutoff
print(test_stats>cutoff)
```

```
## stage_old     stage       dud
##     FALSE     FALSE      TRUE
```

```
p_values = pchisq(test_stats,1,lower.tail = FALSE) # get prob above
print(p_values)
```

```
##    stage_old        stage          dud
## 0.6829175415 0.8907425797 0.0001763977
```

For these data, we reject the null for `dud` but not for `stage_old` and `stage`.

CAVEAT AND WARNING: For discrete random variables, the test statistic actually only takes a discrete set of values. This means that the asymptotic distribution has to be wrong (it is always wrong, but usually everything is fine). Where this is the most pernicious is when the sample size is small (and so the number of values the test statistic can take is small). Let's go ahead and actually compute all of the possible values for the test statistic and get exact p-values for these tests. Luckily there is a function `binom.test` we can use to do this.

```
binom.test(sum(stage_old),length(stage_old),0.5,conf.level = 1-alpha)
```

```
##
##  Exact binomial test
##
## data:  sum(stage_old) and length(stage_old)
## number of successes = 13, number of trials = 24, p-value = 0.8388
## alternative hypothesis: true probability of success is not equal to 0.5
## 99 percent confidence interval:
##  0.2738342 0.7930313
## sample estimates:
## probability of success
##              0.5416667
```

```
binom.test(sum(stage),length(stage),0.5,conf.level = 1-alpha)
```

```
##
##  Exact binomial test
##
## data:  sum(stage) and length(stage)
## number of successes = 27, number of trials = 53, p-value = 1
## alternative hypothesis: true probability of success is not equal to 0.5
## 99 percent confidence interval:
##  0.3292519 0.6878827
## sample estimates:
## probability of success
##               0.509434
```

```
binom.test(sum(dud),length(dud),0.25,conf.level = 1-alpha)
```

```
##
##  Exact binomial test
##
## data:  sum(dud) and length(dud)
## number of successes = 105, number of trials = 570, p-value = 0.000229
## alternative hypothesis: true probability of success is not equal to 0.25
## 99 percent confidence interval:
##  0.1443090 0.2295105
## sample estimates:
## probability of success
##              0.1842105
```

While the actual p-values have changed, our inference remains the same. In general, we are probably better off using these exact tests for the binomial proportion. Now, look back at the test statistic λ. It only depends on the data through $\bar{y}$ and so the distribution of this statistic determines the distribution of λ. This is how `binom.test` does its computations, through the distribution for $\bar{y}$.

1.1.2 Testing equality of Bernoulli probabilities

Here, we have two datasets $y_1, \ldots, y_n$ and $x_1, \ldots, x_m$ each of which follow a Bernoulli distribution. Let's call the probabilities of success θ_Y and θ_X. We want to test the null $H_0 : \theta_Y = \theta_X$ against the alternative $H_A : \theta_Y \neq \theta_X$. In the alternative, we are treating the two datasets as being totally separate and get an MLE for each. In the null, we are pooling the data together and getting an MLE that is the from analyzing this pooled dataset.

To compute the likelihood ratio statistic, we need to compute the alternative likelihood. This is the two likelihoods when we fit both datasets seperately added together. We also need to compute the null likelihoods.

```
stage_young = nodal$stage[nodal$aged==0]
mean_young = mean(stage_young)
loglik_young = length(stage_young)*(
  mean_young*log(mean_young)+(1-mean_young)*log(1-mean_young)
)
mean_old = mean(stage_old)
loglik_old = length(stage_old)*(
  mean_old*log(mean_old)+(1-mean_old)*log(1-mean_old)
)
mean_pooled = mean(stage)
loglik_pooled = length(stage)*(
  mean_pooled*log(mean_pooled)+(1-mean_pooled)*log(1-mean_pooled)
)
lambda = 2*(loglik_young+loglik_old-loglik_pooled)
print(lambda)
```

```
## [1] 0.1824807
```

```
print(pchisq(lambda,1,lower.tail = FALSE))
```

```
## [1] 0.66925
```

1.1.3 Parametric Bootstrap for comparing Bernoulli proportions

This might be fun, right? I mean, we have the bootstrap at our disposal.

Parametric with plug-in: Above, we used the asymptotic distribution to compute our p-value. Another way is to simulate data from the null hypothesis (using our best guess of θ under the null hypothesis). We can use this simulated data to compute a bunch of simulated λ values and compare our observed lambda to these simulated values. This is one version of the parametric bootstrap.

```
theta_null = mean(stage)
N_sims = 10000
lambda_sims = rep(NA,N_sims)
for(i in 1:N_sims){
  stage_sim = rbinom(length(stage),1,theta_null)
  stage_young_sim = stage_sim[nodal$aged==0]
  mean_young = mean(stage_young_sim)
  loglik_young = length(stage_young)*(
    mean_young*log(mean_young)+(1-mean_young)*log(1-mean_young)
  )
  stage_old_sim = stage_sim[nodal$aged==1]
  mean_old_sim = mean(stage_old_sim)
  loglik_old = length(stage_old)*(
    mean_old*log(mean_old)+(1-mean_old)*log(1-mean_old)
  )
  mean_pooled = mean(stage_sim)
  loglik_pooled = length(stage)*(
    mean_pooled*log(mean_pooled)+(1-mean_pooled)*log(1-mean_pooled)
  )
  lambda_sims[i] = 2*(loglik_young+loglik_old-loglik_pooled)
}
print(c(cutoff=quantile(lambda_sims,1-alpha),lambda=lambda))
```

```
## cutoff.99%     lambda
##  4.0109933  0.1824807
```

```
print(mean(lambda_sims<=lambda))
```

```
## [1] 0.5229
```

Parametric and non-parametric: We do not know what the actual probability of success is under the null (we have only estimated it). We can do another level of bootstrap on it. This is another flavor that combines parametric and non-parametric bootstrap. We make a replicate of the dataset, fit the null, and then simulate a new dataset assuming that the null is true. We use this new dataset to compute the simulated λ.

```
N_sims = 10000
lambda_sims = rep(NA,N_sims)
for(i in 1:N_sims){
  stage_rep = sample(stage,length(stage),replace=TRUE)
  theta_null = mean(stage_rep)
  stage_sim = rbinom(length(stage_rep),1,theta_null)
  stage_young_sim = stage_sim[nodal$aged==0]
  mean_young = mean(stage_young_sim)
  loglik_young = length(stage_young)*(
    mean_young*log(mean_young)+(1-mean_young)*log(1-mean_young)
  )
  stage_old_sim = stage_sim[nodal$aged==1]
  mean_old_sim = mean(stage_old_sim)
  loglik_old = length(stage_old)*(
    mean_old*log(mean_old)+(1-mean_old)*log(1-mean_old)
  )
  mean_pooled = mean(stage_sim)
  loglik_pooled = length(stage)*(
    mean_pooled*log(mean_pooled)+(1-mean_pooled)*log(1-mean_pooled)
  )
  lambda_sims[i] = 2*(loglik_young+loglik_old-loglik_pooled)
}
print(c(cutoff=quantile(lambda_sims,1-alpha),lambda=lambda))
```

```
## cutoff.99%     lambda
##  4.6715126  0.1824807
```

```
print(mean(lambda_sims<=lambda))
```

```
## [1] 0.586
```

Parametric and non-parametric respecting grouping: One last thing we might think is that it could matter to respect the grouping over age when replicating the dataset.

```
N_sims = 10000
lambda_sims = rep(NA,N_sims)
for(i in 1:N_sims){
  stage_old_rep = sample(stage_old,length(stage_old),replace=TRUE)
  stage_young_rep = sample(stage_young,length(stage_young),replace=TRUE)
  stage_rep_pooled = c(stage_old_rep,stage_young_rep)
  theta_null = mean(stage_rep_pooled)
  stage_young_sim = rbinom(length(stage_young_rep),1,theta_null)
  mean_young = mean(stage_young_sim)
  loglik_young = length(stage_young)*(
    mean_young*log(mean_young)+(1-mean_young)*log(1-mean_young)
  )
  stage_old_sim = rbinom(length(stage_old_rep),1,theta_null)
  mean_old_sim = mean(stage_old_sim)
  loglik_old = length(stage_old)*(
    mean_old*log(mean_old)+(1-mean_old)*log(1-mean_old)
  )
  stage_sim_pooled = c(stage_old_sim,stage_young_sim)
  mean_pooled = mean(stage_sim_pooled)
  loglik_pooled = length(stage)*(
    mean_pooled*log(mean_pooled)+(1-mean_pooled)*log(1-mean_pooled)
  )
  lambda_sims[i] = 2*(loglik_young+loglik_old-loglik_pooled)
}
print(c(cutoff=quantile(lambda_sims,1-alpha),lambda=lambda))
```

```
## cutoff.99%     lambda
##  4.4267027  0.1824807
```

```
print(mean(lambda_sims<=lambda))
```

```
## [1] 0.6056
```

We could have done all of this using the `boot` function in a variety of ways. We will come back to this later. It gets rather ugly and we have enough to worry about for now.

1.2 General Idea

The likelihood ratio test follows some fairly simple logic when we are testing a point null hypothesis $H_0 : \theta = \theta_0$ versus the alternative $H_A : \theta \neq \theta_0$. The test statistic λ follows and asymptotic χ^2 distribution with degrees of freedom the same as the dimension of θ. For this to work θ_0 has to be in the interior of Θ, the support of the parameter θ. Otherwise, the asymptotics break down.

Now, let's just look at this a bit more carefully when there is just one scalar parameter θ. We will expand $\ell(\theta_0)$ about the point $\theta = \hat{\theta}$ using a polynomial expansion. We will make use of the fact that the log-likelihood function is approximately quadratic when evaluated at the MLE. We are also assuming that $\theta = \theta_0$ actually generates the data.

$$\ell(\theta_0) \approx \ell\left(\hat{\theta}\right) - \frac{1}{2}H\left(\hat{\theta}\right)\left(\hat{\theta} - \theta_0\right)^2 \tag{4}$$

where $H(\hat{\theta})$ is the curvature and provides the precision, which is the inverse of the variance. We get

$$\lambda = 2\left(\ell\left(\hat{\theta}\right) - \ell(\theta_0)\right) \approx H\left(\hat{\theta}\right)\left(\hat{\theta} - \theta_0\right)^2. \tag{5}$$

When $\theta = \theta_0$, we know that the MLE has Central Limit Theorem

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \xrightarrow{Dist} N\left(0, I(\theta_0)^{-1}\right). \tag{6}$$

Because $H\left(\hat{\theta}\right)$ converges to $nI\left(\theta_0\right)$ almost surely, we can conclude that λ converges to a χ^2_1. Those of you who love linear algebra and real analysis can go ahead and do this same thing for matrix parameters and when there are extra parameters ϕ. Otherwise, trust me? The only change is that the degrees of freedom changes from 1 to the dimension of θ. Now you can see why being in the interior matters. Otherwise we cut the parabola in half and things get weird.

When we have a small sample size, we can use the parametric bootstrap to try to compute the p-value. This is often also referred to as simulating the p-value. When there are no extra parameters ϕ, we can directly simulate from the null hypothesis. When there is a ϕ, we can do either 1) fix it at its MLE under the null and simulate or 2) replicate the dataset, fit ϕ for the replicate, and simulate. When doing the latter, as we would for any bootstrap, we should respect different structures (like stratifications) in the population.

1.2.1 More examples

Here we present two examples of testing normal means. The first is comparing means of two groups when the standard deviations of the groups are assumed to be the same. The second is the same comparison when the standard deviations are assumed to be different. These are similar in spirit to comparing the proportions in two groups that we did before for Bernoulli data.

Let's assume two sets of random variables $Y_1, \ldots, Y_n$ and $X_1, \ldots, X_m$ with realizations $y_1, \ldots, y_n$ and $x_1, \ldots, x_m$. Assume that the Y_i are iid normal with mean μ_Y and standard deviation σ_Y and that the X_j are iid normal with mean μ_X and standard deviation σ_X. Additionally assume that each Y_i is indepdendent of each X_j.

In our first case, we will assume that $\sigma_Y = \sigma_X$ and in our second case we will not make this restriction. In either case, the null hypothesis is $H_0 : \mu_y = \mu_X$ and the alternative is $H_A : \mu_Y \neq \mu_X$. We will write one density function for getting the log-likelihood under the least restrictive situation and use it in four different MLE functions.

```
d_norm_twogroups = function(y,x,mu_y,sigma_y,mu_x,sigma_x,log=FALSE){
  out = c(dnorm(y,mu_y,sigma_y,TRUE), dnorm(x,mu_x,sigma_x,TRUE))
  if(log){out}else{exp(out)}
}

# Sigma same
```

```r
mle_norm_twogroups_same_mean_same_sd = function(y,x){
  f_trans = function(theta_trans){
    mu_y = theta_trans[1]
    sigma_y = exp(theta_trans[2])
    mu_x = mu_y
    sigma_x = sigma_y
    sum(d_norm_twogroups(y,x,mu_y,sigma_y,mu_x,sigma_x,TRUE))
  }
  control = list(fnscale=-1,maxit=1000,reltol=1e-10)
  opt_trans = optim(c(0,0),f_trans,method="BFGS",control=control,hessian=FALSE)
  mu = opt_trans$par[1]
  sigma = exp(opt_trans$par[2])
  theta = c(mu=mu,sigma=sigma)
  loglik = opt_trans$value
  return(list(theta=theta,loglik=loglik))
}

mle_norm_twogroups_diff_mean_same_sd = function(y,x){
  f_trans = function(theta_trans){
    mu_y = theta_trans[1]
    sigma_y = exp(theta_trans[2])
    mu_x = theta_trans[3]
    sigma_x = sigma_y
    sum(d_norm_twogroups(y,x,mu_y,sigma_y,mu_x,sigma_x,TRUE))
  }
  control = list(fnscale=-1,maxit=1000,reltol=1e-10)
  opt_trans = optim(c(0,0,0),f_trans,method="BFGS",control=control,hessian=FALSE)
  mu_y = opt_trans$par[1]
  sigma = exp(opt_trans$par[2])
  mu_x = opt_trans$par[3]
  theta = c(mu_y=mu_y,sigma=sigma,mu_x=mu_x)
  loglik = opt_trans$value
  return(list(theta=theta,loglik=loglik))
}

mle_norm_twogroups_same_mean_diff_sd = function(y,x){
  f_trans = function(theta_trans){
    mu_y = theta_trans[1]
    sigma_y = exp(theta_trans[2])
    mu_x = mu_y
    sigma_x = exp(theta_trans[3])
    sum(d_norm_twogroups(y,x,mu_y,sigma_y,mu_x,sigma_x,TRUE))
  }
  control = list(fnscale=-1,maxit=1000,reltol=1e-10)
  opt_trans = optim(c(0,0,0),f_trans,method="BFGS",control=control,hessian=FALSE)
  mu = opt_trans$par[1]
  sigma_y = exp(opt_trans$par[2])
  sigma_x = exp(opt_trans$par[3])
  theta = c(mu=mu,sigma_y=sigma_y,sigma_x=sigma_x)
  loglik = opt_trans$value
  return(list(theta=theta,loglik=loglik))
}

mle_norm_twogroups_diff_mean_diff_sd = function(y,x){
  f_trans = function(theta_trans){
    mu_y = theta_trans[1]
    sigma_y = exp(theta_trans[2])
    mu_x = theta_trans[3]
    sigma_x = exp(theta_trans[4])
    sum(d_norm_twogroups(y,x,mu_y,sigma_y,mu_x,sigma_x,TRUE))
  }
  control = list(fnscale=-1,maxit=1000,reltol=1e-10)
  opt_trans = optim(c(0,0,0,0),f_trans,method="BFGS",control=control,hessian=FALSE)
  mu_y = opt_trans$par[1]
  sigma_y = exp(opt_trans$par[2])
  mu_x = opt_trans$par[3]
  sigma_x = exp(opt_trans$par[4])
  theta = c(mu_y=mu_y,sigma_y=sigma_y,mu_x=mu_x,sigma_x=sigma_x)
  loglik = opt_trans$value
  return(list(theta=theta,loglik=loglik))
}
```

Let’s get a dataset and do these two tests (the one with common variance and the one without).

```
library(MASS)
data(cats) # in MASS
y = cats$Bwt[cats$Sex=="M"]
x = cats$Bwt[cats$Sex=="F"]
fit_same_mean_same_sd = mle_norm_twogroups_same_mean_same_sd(y,x)
fit_diff_mean_same_sd = mle_norm_twogroups_diff_mean_same_sd(y,x)
lambda = 2*fit_diff_mean_same_sd$loglik-2*fit_same_mean_same_sd$loglik
p_val = pchisq(lambda,1,lower.tail=FALSE)
print(p_val)
```

```
## [1] 1.058387e-11
```

```
fit_same_mean_diff_sd = mle_norm_twogroups_same_mean_diff_sd(y,x)
fit_diff_mean_diff_sd = mle_norm_twogroups_diff_mean_diff_sd(y,x)
lambda = 2*fit_diff_mean_diff_sd$loglik-2*fit_same_mean_diff_sd$loglik
p_val = pchisq(lambda,1,lower.tail=FALSE)
print(p_val)
```

```
## [1] 7.156528e-15
```

We reject the null hypothesis in both cases for any reasonable significance level. We shoud try to determine whether the standard deviations are the same when we are already assuming the means are not the same. This is another likelihood ratio test.

```r
lambda = 2*fit_diff_mean_diff_sd$loglik-2*fit_diff_mean_same_sd$loglik
p_val = pchisq(lambda,1,lower.tail=FALSE)
print(p_val)
```

```
## [1] 7.065559e-05
```

Because this p-value is small compared to any reasonable significance level, we can reasonably conclude that male cats and female cats have both different standard deviations of weight. Now, if we tried to test the difference in standard deviations with the means being the same (which we have already seen that we should reject), then we might make a mistake.

```r
lambda = 2*fit_same_mean_diff_sd$loglik-2*fit_same_mean_same_sd$loglik
p_val = pchisq(lambda,1,lower.tail=FALSE)
print(p_val)
```

```
## [1] 0.2275846
```

The mistake here is that a lot of the full data x and y variance is actually explained by the difference in group means and our ability to see the difference between the groups might come from the fact that the data x is a lot tighter than the data y. When we assume common means, we lose the center for the two groups and so the contribution to the overall variance for both is wrong. Something like 25% of the variation in the data is explained by the difference in group means.

2 Akaike Information Criterion

The idea of the AIC is very simple. We begin with an assumption that ALL MODELS ARE WRONG. So, while we have the model $f(y;\theta)$ that we are using to analyze the data, we have an actual distribution of the data simply given by $g(y)$. Generally, g is not $f(y;\theta)$. The AIC makes sense for comparing models that are non-nested (one is not a simpler version of another), which makes it very useful.

When we are maximizing the likelihood, we are approximating the expectation

$$E\left[\log(f(Y;\theta));g\right] \approx \frac{\ell(\theta;y_1,\dots,y_n)}{n}. \tag{7}$$

In order to see how good the parametric model is, we want to maximize the left hand side. This happens at the value θ^*. We do not know θ^*, so the best we can do is plug in $\hat{\theta}$ to see how good our model is at approximating g. Plugging in $\hat{\theta}$ on the right hand side, we get an approximation of the thing we want. However, it is biased. The AIC is a first order bias correction. It turns out that the bias corrected approximation is

$$E\left[\log\left(f\left(Y;\hat{\theta}\right)\right);g\right] \approx \frac{\ell\left(\hat{\theta};y_1,\dots,y_n\right)}{n} - \frac{p}{n} \tag{8}$$

where $p = dim(\theta)$. We can convert this to an information criterion that judges the quality of the model by taking $-2n$ times the right hand side, which gives

$$AIC(f) = -2\ell\left(\hat{\theta};y_1,\dots,y_n\right) + 2p. \tag{9}$$

The bias correction can be thought of as a penalization for model complexity. As models get more complex (use more parameters), they get penalized for each additionally parameter used.

Model comparison is achieved through the AIC by choosing the model with the smallest AIC as the model that best approximates the true data generation process. An alternative to choosing a model is to make model weights (think of these as probabilities). The weight (unnormalized probability) for model f is defined as

$$w(f) = \exp\left(-\frac{AIC(f)}{2}\right). \tag{10}$$

Often, choosing a model is insufficient for a problem (like making predictions from a complicated process) and the AIC weights are used to randomly draw models from the space of models.

2.1 Caveats

1. Parameter value must be on interior of its support.
2. The use of p can be refined. Using p is really assuming that g is a restriction of f, so that f is correct but just has too many parameters. The refinement was done by Takeuchi and is called the TIC and uses a somewhat complicated formula for "dimension."
3. There are a lot of asymptotic arguments that give the penalty. There are versions of AIC with corrections for small sample size.
4. There are extensions for non-iid data. For instance, there are AICs for comparing time series data.
5. We can use the bootstrap to evaluate the bias instead of the asymptotic arguments. This is probably a better thing to do in small samples.

2.2 Examples of AIC

First we use the AIC to compare the four models we had for cat weights.

```r
AIC_same_mean_same_sd = 2*2-2*fit_same_mean_same_sd$loglik
AIC_diff_mean_same_sd = 2*3-2*fit_diff_mean_same_sd$loglik
AIC_same_mean_diff_sd = 2*3-2*fit_same_mean_diff_sd$loglik
AIC_diff_mean_diff_sd = 2*4-2*fit_diff_mean_diff_sd$loglik
print(c(
  AIC_same_mean_same_sd,
  AIC_diff_mean_same_sd,
  AIC_same_mean_diff_sd,
  AIC_diff_mean_diff_sd
))
```

```
## [1] 203.4342 159.2169 203.9783 145.4237
```

The AIC provides the same inference we had made before from doing tests.

Second, we use the bootstrap to compute the difference in AIC for the comparison of probabilities of success for Bernoulli trials from the `nodal` data. We need to double the bias output from the bootstrap in order to match the AIC theory.

```r
twice_log_like_diff_Bern = function(data,ind,grouping){
  data=data[ind]; grouping = grouping[ind]
  y = data[grouping==0]; x = data[grouping==1]
  n_y = length(y); n_x = length(x); n = n_x + n_y
  theta_same_prob = mean(c(y,x))
  loglik_same_prob = n*(theta_same_prob*log(theta_same_prob)+
                          (1 - theta_same_prob)*log(1 - theta_same_prob))
  theta_y = mean(y); theta_x = mean(x)
  loglik_diff_prob =  n_y*(theta_y*log(theta_y)+
                             (1 - theta_y)*log(1 - theta_y)) +
    n_x*(theta_x*log(theta_x)+
           (1 - theta_x)*log(1 - theta_x))
  2*loglik_diff_prob-2*loglik_same_prob
}

boot_reps = boot(nodal$stage,twice_log_like_diff_Bern,10000,
                 strata=nodal$aged,grouping=nodal$aged)

AIC_bias = 2*(mean(boot_reps$t)-boot_reps$t0)
AIC_diff_boot = boot_reps$t0-AIC_bias
print(c(AIC_bias=AIC_bias, AIC_diff_boot=AIC_diff_boot))
```

```
##      AIC_bias AIC_diff_boot
##      2.005179     -1.822698
```

The difference in AIC is negative, which means that the smaller model is better, which is what we concluded from the LRT.