MLE

Contents

1 MLE

1.1 Poisson

Let's define the Poisson log-likelihood

$$l = \sum_i y_i \log(\mu) - n\mu - \sum_i \log(y_i!)$$

```
poisson.lik<-function(mu,y){
  n<-nrow(y)
  logl<-sum(y)*log(mu)-n*mu
  return(-logl)
}
```

To compute the MLE of μ we need to maximize the log-likelihood. We use the optim R-function:

optim(starting values, log-likelihood, data)

starting values is a vector of starting values, log-likelihood is the name of the log-likelihood function that you seek to maximize, and data declares the data for the estimation. This specification causes R to use the Nelder-Mead algorithm. If you want to use the BFGS algorithm you should include the method="BFGS" option.

Imagine that we have a vector of 100 data points that consists of draws from a Poisson distribution with unknown μ.

```
npoint <- 100
x=rpois(npoint,2)
dat=data.frame(x)
```

We want to estimate this parameter and have already declared the log-likelihood function as poisson.lik.

Estimation using the BFGS algorithm is done as follows

```
optim(1,poisson.lik,y=dat,method="BFGS")
```

```
## $par
## [1] 2.02
##
## $value
## [1] 59.9743
```

```
##
## $counts
## function gradient
##       23        6
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Estimation using the CG algorithm is done as follows

```
optim(1,poisson.lik,y=dat,method="CG")
```

```
## $par
## [1] 2.02
##
## $value
## [1] 59.9743
##
## $counts
## function gradient
##       38        9
##
## $convergence
## [1] 0
##
## $message
## NULL
```

1.2 Normal

The normal log-likelihood is

$$-.5n\log(2\pi) - .5n\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_i (y_i - \mu)^2$$

we can program it as

```
normal.lik<-function(theta,y){
  mu<-theta[1]
  sigma2<-theta[2]
  n<-length(y)
  logl<- -.5*n*log(2*pi) -.5*n*log(sigma2) - (1/(2*sigma2))*sum((y-mu)**2)
  return(-logl)
}
```

Let's load real data to apply MLE on.

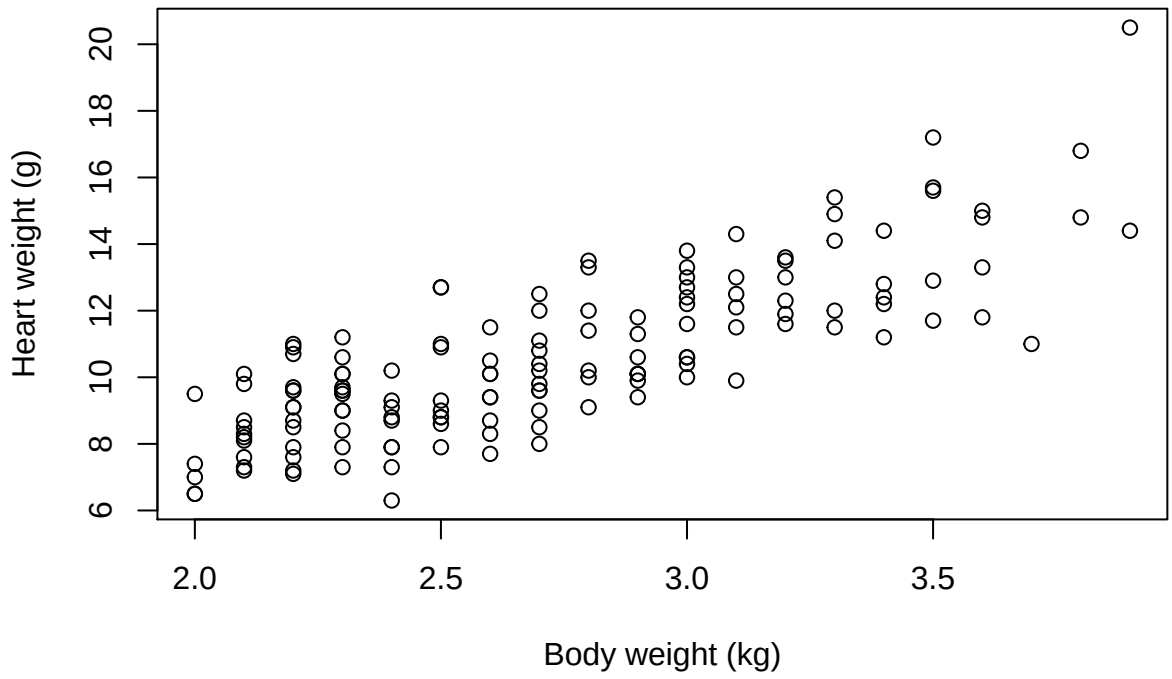
```
library(MASS)
data(cats)
names(cats)
```

```
## [1] "Sex" "Bwt" "Hwt"
```

```
head(cats)
```

```
##   Sex Bwt Hwt
## 1   F 2.0 7.0
## 2   F 2.0 7.4
## 3   F 2.0 9.5
## 4   F 2.1 7.2
## 5   F 2.1 7.3
## 6   F 2.1 7.6
```

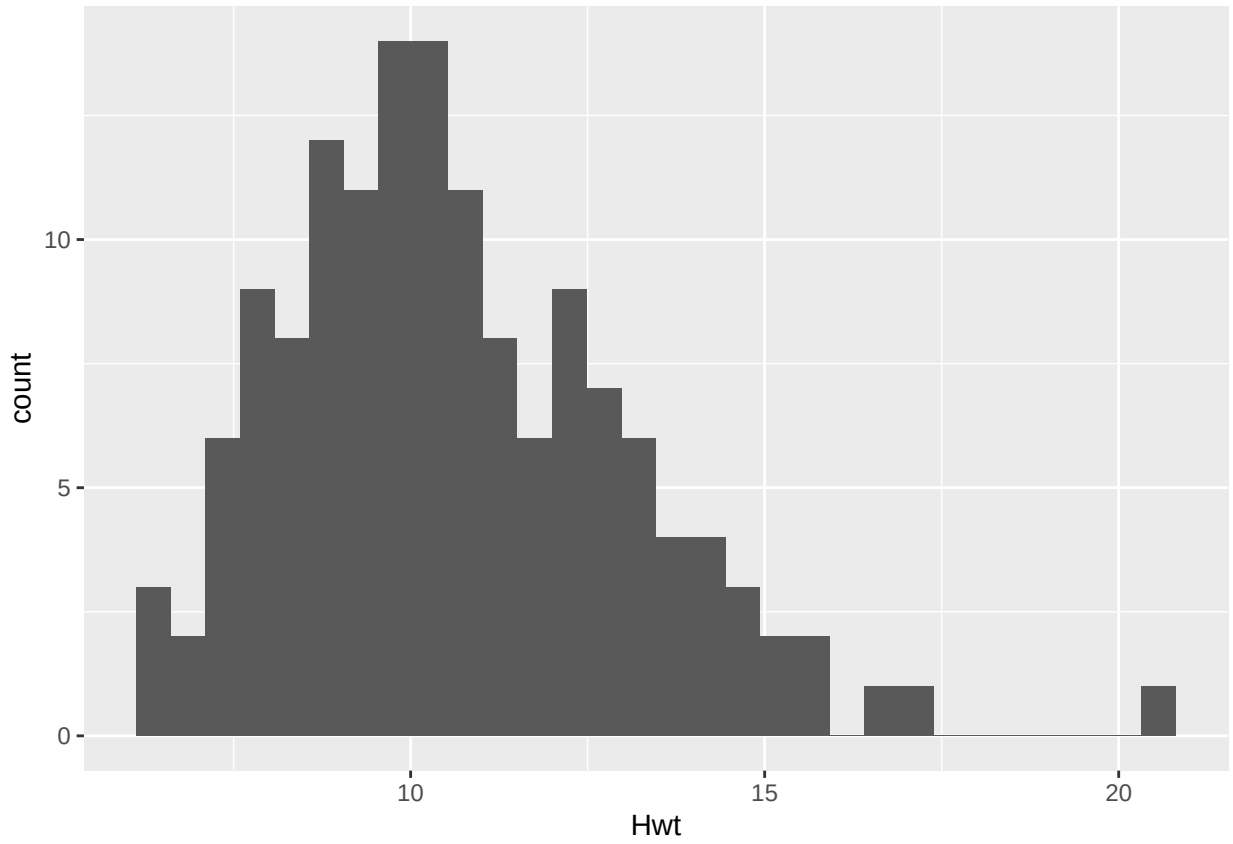
Let's plot Hwt vs Bwt:



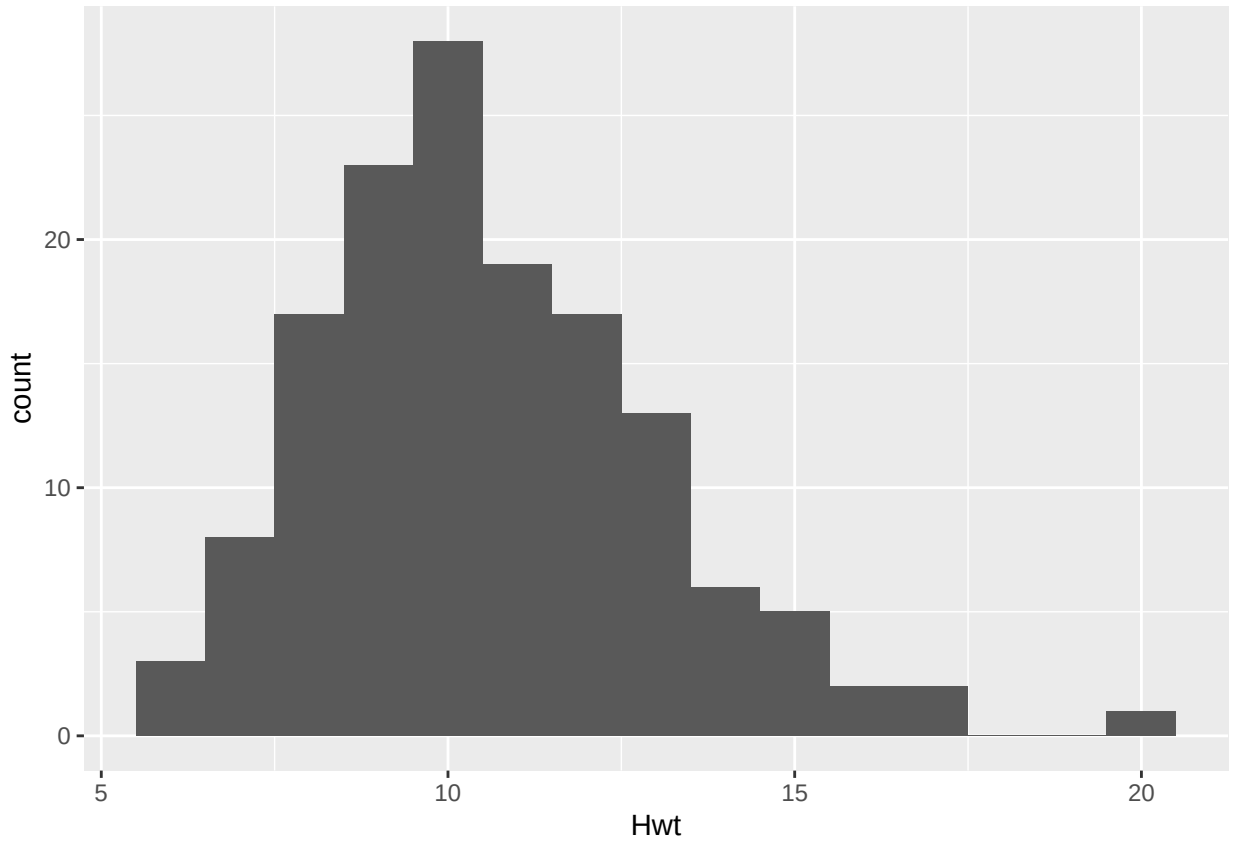
Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
library(ggplot2)
# Basic histogram
ggplot(cats, aes(x=Hwt)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

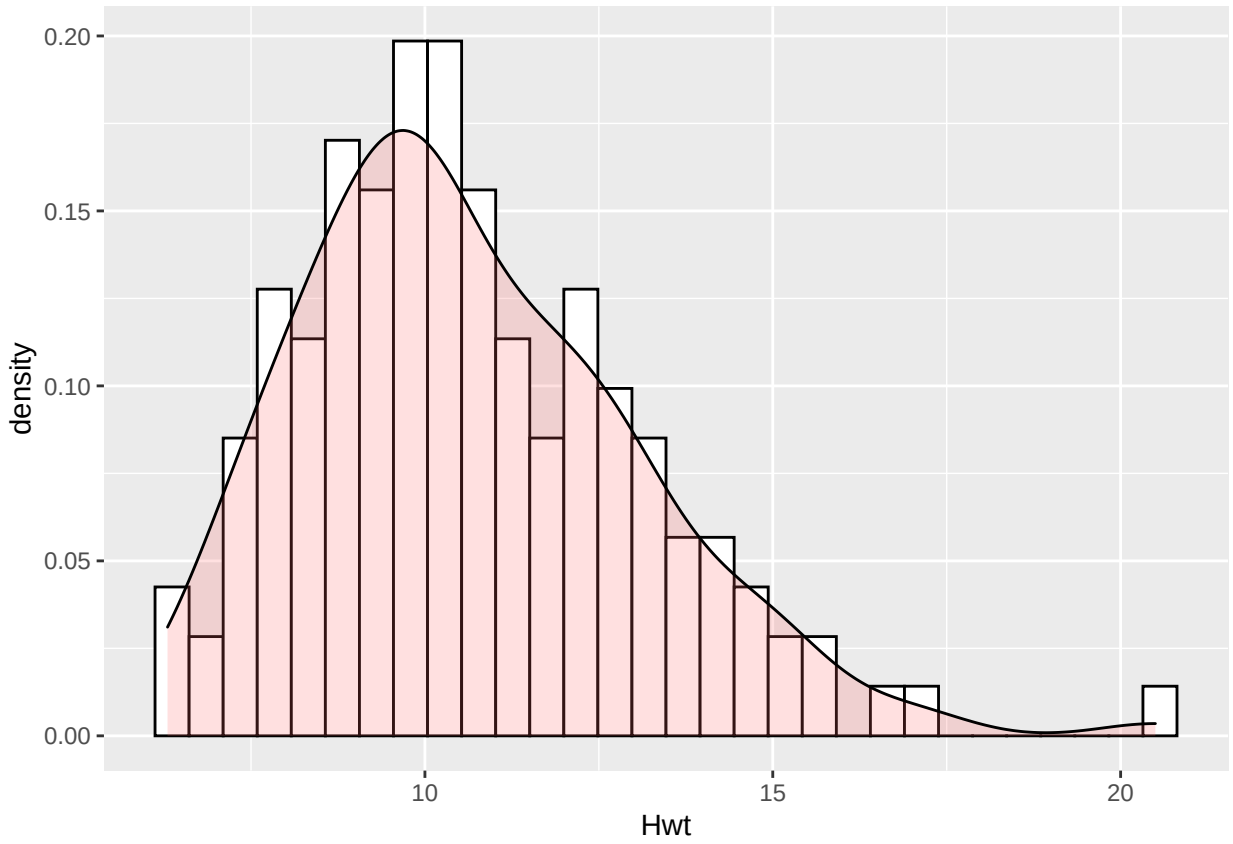



```
# Change the width of bins
ggplot(cats, aes(x=Hwt)) +
  geom_histogram(binwidth=1)
```



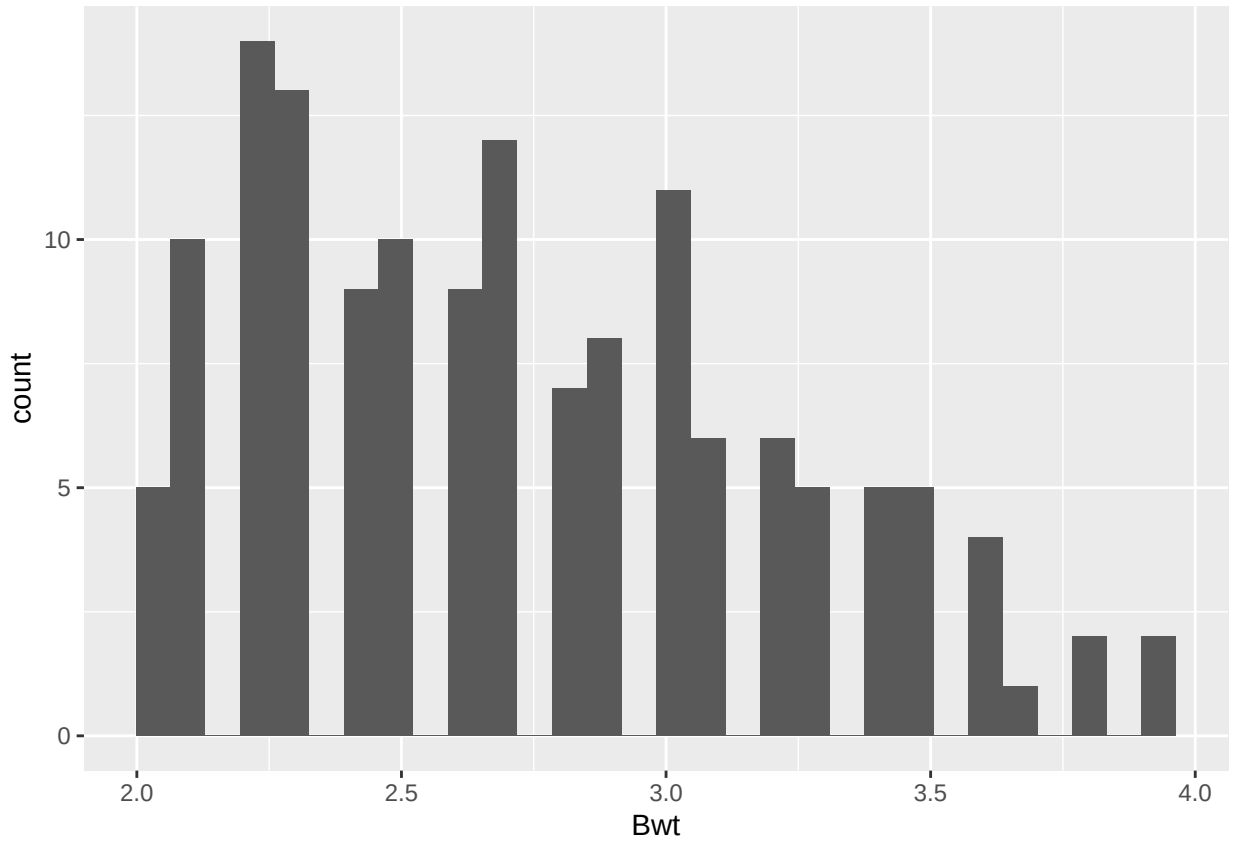

```
# Histogram with density plot
ggplot(cats, aes(x=Hwt)) +
 geom_histogram(aes(y=..density..), colour="black", fill="white")+
 geom_density(alpha=.2, fill="#FF6666")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

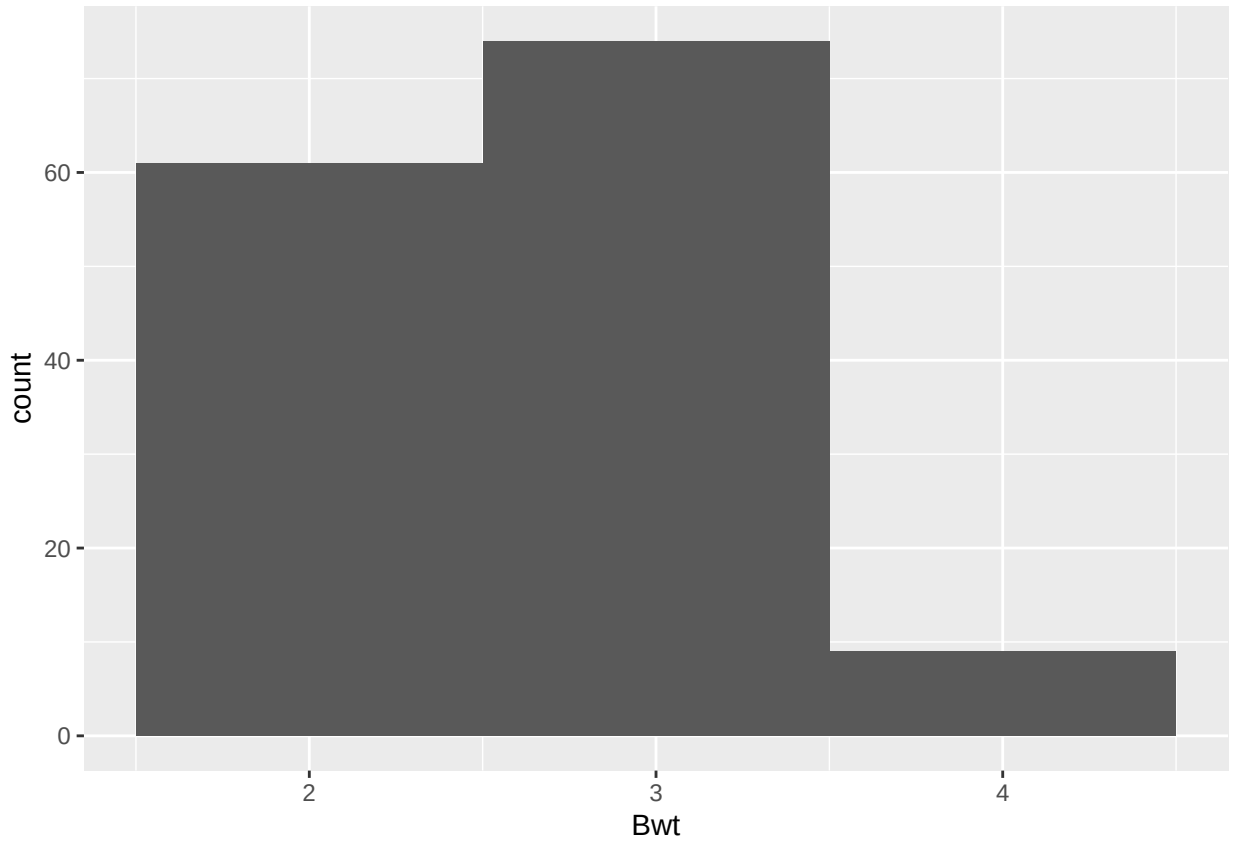



```
library(ggplot2)
# Basic histogram
ggplot(cats, aes(x=Bwt)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

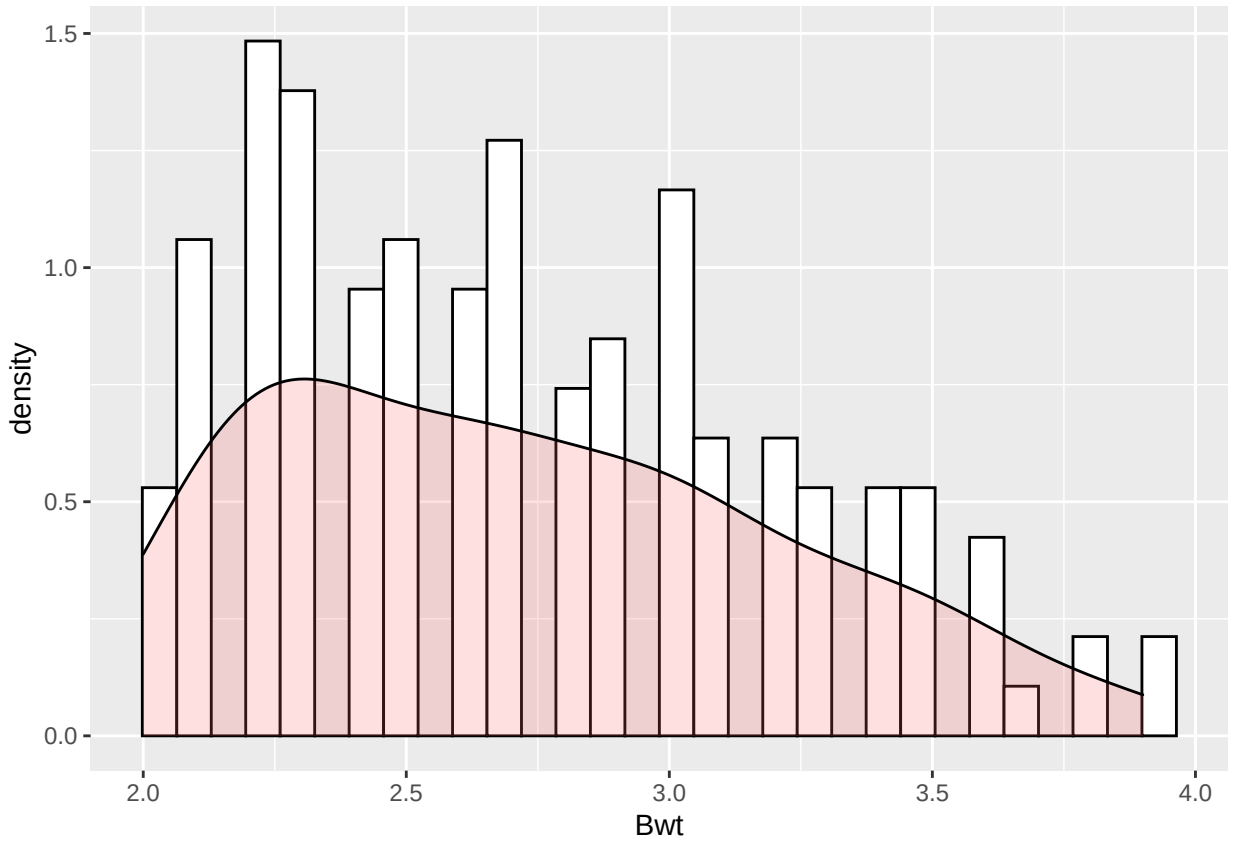



```
# Change the width of bins
ggplot(cats, aes(x=Bwt)) +
  geom_histogram(binwidth=1)
```




```
# Histogram with density plot
ggplot(cats, aes(x=Bwt)) +
 geom_histogram(aes(y=..density..), colour="black", fill="white")+
 geom_density(alpha=.2, fill="#FF6666")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Let's assume that Hwt is normal and fit the normal likelihood to these data to obtain the MLE's of μ and σ^2:

```
y=cats$Hwt
mu=mean(y)
sig=sd(y)
optim(c(mu,sig),normal.lik,y=y,method="BFGS")
```

```
## Warning in log(sigma2): NaNs produced

## Warning in log(sigma2): NaNs produced
```

```
## $par
## [1] 10.630556  5.886325
##
## $value
## [1] 331.9562
##
## $counts
## function gradient
##       26        8
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Let's try with normal data:

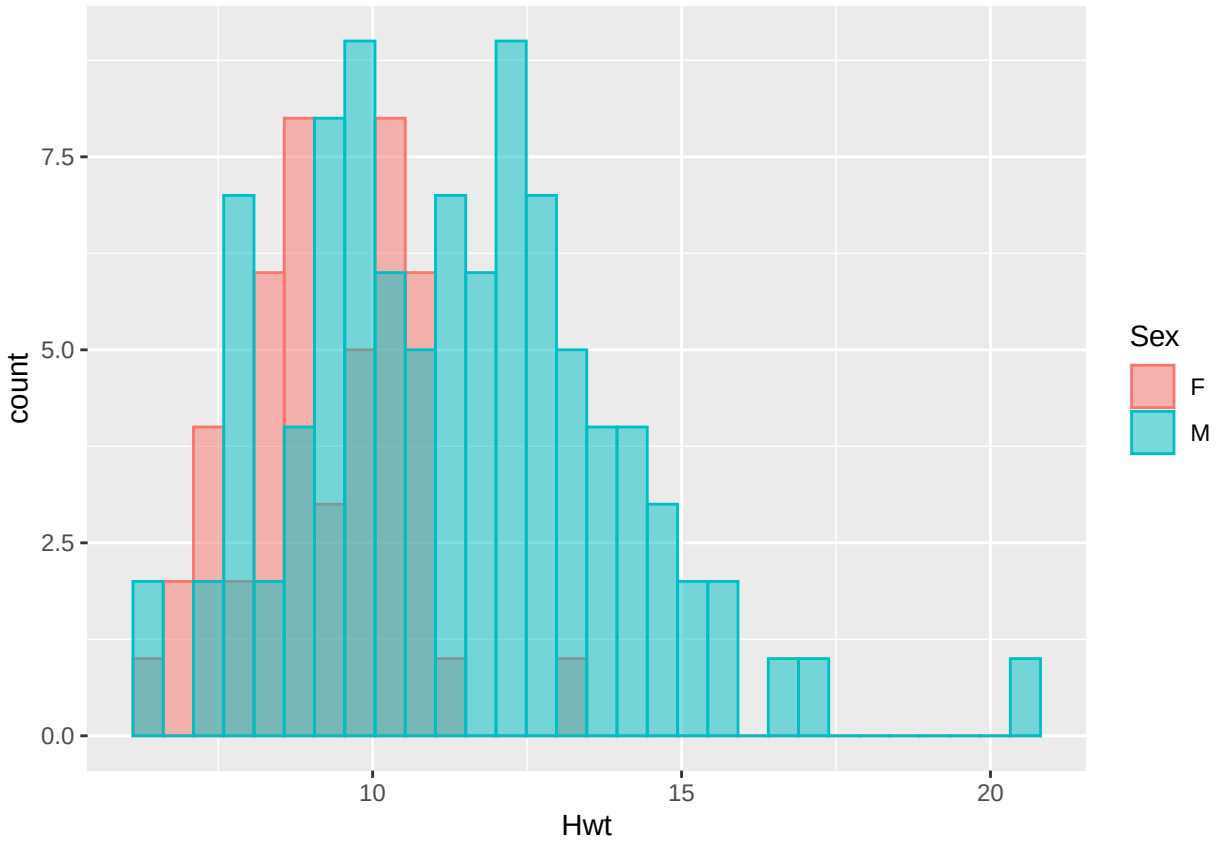
```r
y=rnorm(100)
mu=mean(y)
sig=sd(y)
optim(c(mu,sig),normal.lik,y=y,method="BFGS")
```

```
## $par
## [1] 0.0820075 1.0420268
##
## $value
## [1] 143.9522
##
## $counts
## function gradient
##       17        4
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Let's see if there is any difference in the two sexes:

```r
# Change histogram plot line colors by groups
# Use semi-transparent fill
p<-ggplot(cats, aes(x=Hwt, fill=Sex,color=Sex)) +
  geom_histogram(position="identity", alpha=0.5)
p
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



1.3 The Cauchy Location-Scale Family

The (standard) Cauchy Distribution is the continuous univariate distribution having density

$$f(x) = \frac{1}{\pi} \frac{1}{1 + x^2}, \quad -\infty < x < \infty$$

The standard Cauchy distribution has no parameters, but it induces a two-parameter location-scale family having densities

$$f_{\mu,\sigma^2}(x) = \frac{1}{\sigma} f\left(\frac{x - \mu}{\sigma}\right), \quad -\infty < x < \infty$$

If f is any distribution having mean zero and variance 1, then $f_{\mu,\sigma^2}(x)$ has mean μ and variance σ^2. But the Cauchy distribution has neither mean nor variance. Thus we call μ the location parameter and σ the scale parameter.

Since the standard Cauchy distribution is clearly symmetric about zero, the Cauchy(μ, σ) distribution is symmetric about μ. Hence μ is the population median and a "good" estimate is the sample median. A robust scale estimator analogous to the sample median is the interquartile range (IQR). The IQR of the standard Cauchy distribution is

```
qcauchy(3/4) - qcauchy(1/4)
```

```
## [1] 2
```

The population IQR of the Cauchy(μ, σ) distribution is 2σ, and hence a "good" estimate of σ is the sample IQR divided by 2.

```
cauchy.lik<-function(mu,y){
  logl<-sum(log(1 + (x - mu)^2))
  return(-logl)
}
```

or as

```
mlogl <- function(mu, x) {
  logl=sum(-dcauchy(x, location = mu, log = TRUE))
  return(logl)
}
```

Either produces the same results on the simulated data

```
n <- 30
set.seed(42)
x <- rcauchy(n)
```

Here the true "unknown" μ is zero, but we pretend we do not know that and see how good the MLE is as an estimator.

The following does the estimation

```
mu.start <- median(x)
mu.start
```

```
## [1] -0.1955062
```

```
optim(mu.start,cauchy.lik,y=x,method="BFGS")
```

```
## $par
## [1] -53769.35
##
## $value
## [1] -653.547
##
## $counts
## function gradient
##       29       28
##
## $convergence
## [1] 0
##
## $message
## NULL
```

And the following does the estimation using the other "minus log likelihood" function

```
optim(mu.start,mlogl,x=x,method="BFGS")
```

```
## $par
## [1] -0.1816496
##
## $value
## [1] 77.47065
##
## $counts
## function gradient
##       11        3
```

```
##
## $convergence
## [1] 0
##
## $message
## NULL
```

1.3.1 A Simulation Study

We see for these data, the MLE is slightly better than the sample median. But this is just one data set. For random data sometimes the MLE will be better and sometimes the sample median will be better. We are interested in the sampling distributions of the two estimators, which we can easily study by simulation.

```
nsim <- 100
mu <- 0
mu.hat <- double(nsim)
mu.med <- double(nsim)
for (i in 1:nsim) {
     xsim <- rcauchy(n, location = mu)
     mu.start <- median(xsim)
     out<-optim(mu.start,mlogl,x=xsim,method="BFGS")
     mu.hat[i] <- out$par
     mu.med[i] <- mu.start
}

mean((mu.hat - mu)^2)
```

```
## [1] 0.06203156
```

```
mean((mu.med - mu)^2)
```

```
## [1] 0.08242236
```

The two numbers reported from the simulation are the mean square errors (MSE) of the two estimators. Their ratio

```
mean((mu.hat - mu)^2)/mean((mu.med - mu)^2)
```

```
## [1] 0.752606
```

is the **asymptotic relative efficiency (ARE)** of the estimators. We see the MLE is more accurate, as theory says it must be.

1.3.2 Two Parameters

The two parameter Cauchy log-likelihood is

$$l = -n \log(\sigma\pi) - \sum_i \left(1 + \left(\frac{x_i - \mu}{\sigma}\right)^2\right)$$

```
cauchy.lik2<-function(theta,y) {
  mu<-theta[1]
  sigma<-theta[2]
  n<-length(y)
  logl <- -n* log(sigma * pi) - sum(log(1+((y-mu)/sigma)^2))
  return(-logl)
}
```

and the MLE calculated by

```
theta1=median(x)
theta2=IQR(x)/2
theta.start=c(theta1,theta2)
out <- optim(c(theta1,theta2),cauchy.lik2,y=x,method="BFGS")
theta.hat <- out$par
theta.hat
```

```
## [1] -0.1809269  0.7605517
```

1.4 Fisher Information

1.4.1 Observed Fisher Information

```
out <- optim(c(theta1,theta2),cauchy.lik2,y=x,hessian=T,method="BFGS")
```

We can now invert the Hessian to obtain the observed Fisher information matrix, the asymptotic variance matrix of the MLE

```
fish.observed <- out$hessian
OI<-solve(out$hessian)
```

The square root of the diagonal elements are then the standard errors,

```
se<-sqrt(diag(OI))
se
```

```
## [1] 0.1753908 0.2272959
```

From it, we can construct asymptotic confidence intervals

```
conf.level <- 0.95
crit <- qnorm((1 + conf.level)/2)

theta.hat[1] + c(-1, 1) * crit * sqrt(se)
```

```
## [1] -1.0017537  0.7534969
```

```
theta.hat[2] + c(-1, 1) * crit * sqrt(se)
```

```
## [1] -0.06027512  1.69497551
```

1.4.2 Expected Fisher Information

The first derivatives are

$$\frac{\partial l(\mu,\sigma)}{\partial \mu} = \sum_i \frac{2(x_i-\mu)}{\sigma^2+(x_i-\mu)^2}$$

$$\frac{\partial l(\mu,\sigma)}{\partial \sigma} = \frac{n}{\sigma} - \sum_i \frac{2\sigma}{\sigma^2+(x_i-\mu)^2}$$

R does not compute analytic integrals. But it does numerical integrals, which is all we need to compute Fisher information.

```
theta.hat
```

```
## [1] -0.1809269  0.7605517
```

```
mu <- theta.hat[1]
sigma <- theta.hat[2]
grad1 <- function(x) 2 * (x - mu)/(sigma^2 + (x - mu)^2)
grad2 <- function(x) (1/sigma - 2 * sigma/(sigma^2 + (x - mu)^2))

fish.exp <- matrix(NA, 2, 2)
fish.exp[1, 1] <- integrate(function(x) grad1(x)^2 * dcauchy(x,
    mu, sigma), -Inf, Inf)$value
fish.exp[2, 2] <- integrate(function(x) grad2(x)^2 * dcauchy(x,
    mu, sigma), -Inf, Inf)$value
fish.exp[1, 2] <- integrate(function(x) grad1(x) * grad2(x) *
    dcauchy(x, mu, sigma), -Inf, Inf)$value
fish.exp[2, 1] <- fish.exp[1, 2]

round(n * fish.exp, 10)
```

```
##          [,1]     [,2]
## [1,] 25.93187  0.00000
## [2,]  0.00000 25.93187
```

```
fish.observed
```

```
##             [,1]        [,2]
## [1,] 32.50771347  0.01763678
## [2,]  0.01763678 19.35606168
```

The matrix fish.observed is observed Fisher information evaluated at the MLE, that is, $J(\hat{\theta})$. The matrix fish.exact is expected Fisher information evaluated at the MLE, that is, $I(\hat{\theta})$. Neither is "exact": both are approximations to the asymptotic variance of the MLE.

1.5 Function maxlogL

Let's use maxlogL in package EstimationTools:

```
library(EstimationTools)
```

```
## Loading required package: DEoptim
## Loading required package: parallel
##
## DEoptim package
## Differential Evolution algorithm in R
## Authors: D. Ardia, K. Mullen, B. Peterson and J. Ulrich
## Loading required package: survival
## Loading required package: stringr
## Loading required package: BBmisc
##
## Attaching package: 'BBmisc'
## The following object is masked from 'package:base':
##
##     isFALSE
## >>>>>>>>>>>>>>>>>>>>>>>>>   EstimationTools Version 2.1.0   <<<<<<<<<<<<<<<<<<<<<<<<<
##   Feel free to report bugs in https://github.com/Jaimemosg/EstimationTools/issues
```

```r
# Example 1: estimation with one fixed parameter
x <- rnorm(n = 10000, mean = 160, sd = 6)
theta_1 <- maxlogL(x = x, dist = 'dnorm', control = list(trace = 1),
                   link = list(over = "sd", fun = "log_link"),
                   fixed = list(mean = 160))
```

```
##   0:     43899.659:  1.00000
##   1:     32533.557:  2.00000
##   2:     32319.293:  1.92253
##   3:     32186.239:  1.77096
##   4:     32178.418:  1.80245
##   5:     32178.291:  1.79899
##   6:     32178.291:  1.79889
##   7:     32178.291:  1.79889
```

```r
summary(theta_1)
```

```
## ------------------------------------------------------------------
## Optimization routine: nlminb
## Standard Error calculation: Hessian from optim
## ------------------------------------------------------------------
##        AIC      BIC
##   64356.58 64356.58
## ------------------------------------------------------------------
##    Estimate  Std. Error Z value Pr(>|z|)
## sd   6.04294    0.04273   141.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## ------------------------------------------------------------------
## Note: p-values valid under asymptotic normality of estimators
## ---
```

Both parameters of normal distribution mapped with logarithmic function:

```r
theta_2 <- maxlogL(x = x, dist = "dnorm",
                   link = list(over = c("mean","sd"),
                               fun = c("log_link","log_link")))
summary(theta_2)
```

```
## ------------------------------------------------------------------
## Optimization routine: nlminb
## Standard Error calculation: Hessian from optim
## ------------------------------------------------------------------
##        AIC      BIC
##   64359.98 64374.41
## ------------------------------------------------------------------
##      Estimate  Std. Error Z value Pr(>|z|)
## mean 159.95328    0.06043  2647.0   <2e-16 ***
## sd     6.04276    0.04273   141.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## ------------------------------------------------------------------
## Note: p-values valid under asymptotic normality of estimators
## ---
```