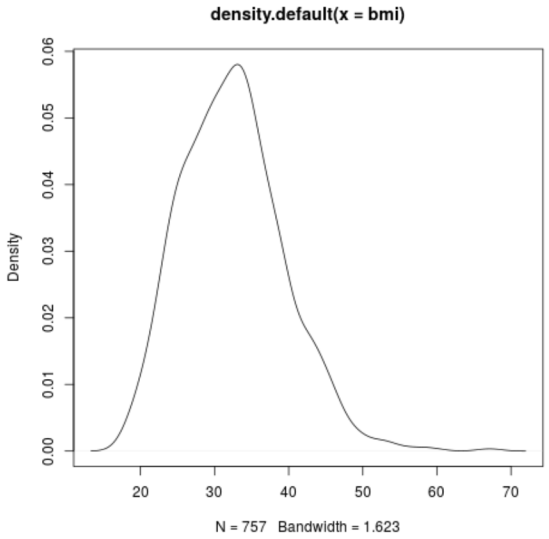

Figure 8.4: Kernel density estimate, BMI data

8.4 Parameter Estimation

To fit a parametric model such as the gamma to our data, the question then arises as to how to estimate the parameters. Two common methods for estimating the parameters of a density are the Method of Moments (MM) and Maximum Likelihood Estimation (MLE).[4] We'll introduce these via examples.

8.4.1 Method of Moments

MM gets its name from the fact that quantities like mean and variance are called *moments*. $E(X^k)$ is the k^{th} moment of X, with $E[(X - EX)^k]$ being termed the k^{th} *central* moment. if we have an m-parameter family, we "match" m moments, as follows.

[4]These methods are used more generally as well, not just for estimating density parameters.

8.4.2 Example: BMI Data

From Section 6.7.4.1, we know that for a gamma-distributed X,

$$EX = r/\lambda \tag{8.14}$$

and

$$Var(X) = r/\lambda^2 \tag{8.15}$$

In MM, we simply replace population values by sample estimates in the above equations, yielding

$$\overline{X} = \widehat{r}/\widehat{\lambda} \tag{8.16}$$

and

$$s^2 = \widehat{r}/\widehat{\lambda}^2 \tag{8.17}$$

Dividing the first equation by the second, we obtain

$$\widehat{\lambda} = \overline{X}/s^2 \tag{8.18}$$

and thus from the first equation,

$$\widehat{r} = \overline{X}\widehat{\lambda} = \overline{X}^2/s^2 \tag{8.19}$$

Let's see how well the model fits, at least visually:

```
xb <- mean(bmi)
s2 <- var(bmi)
lh <- xb/s2
ch <- xb^2/s2
hist(bmi,freq=FALSE,breaks=20)
curve(dgamma(x,ch,lh),0,70,add=TRUE)
```

The plot is shown in Figure 8.5. Visually, the fit looks fairly good. Be sure to keep in mind the possible sources of discrepancy between the fitted model and the histogram:

- Sampling variation: We are of course working with sample data, not the population. It may be that with a larger sample, the discrepancy may be lesser.

- Model bias: As the quote from George Box reminds us, a model is just that, a simplying model of reality. Most models are imperfect, e.g. the assumed massless, frictionless string from physics computations, but are often good enough for our purposes.

- Choice of number of bins: The parametric model here might fit even better with a different choice than our 20 for the number of bins.

8.4.3 The Method of Maximum Likelihood

To see how MLE works, consider the following game. I toss a coin until I accumulate r heads. I don't tell you what value I've chosen for r, but I do tell you K, the number of tosses I needed. You then must guess the value of r. Well, K has a negative binomial distribution (Section 5.4.3), so

$$P(K = u) = \binom{u-1}{r-1} 0.5^u, \ u = r, r+1, ... \tag{8.20}$$

Say I tell you $K = 7$. Then what you might do is find the value of r that maximizes

$$\binom{6}{r-1} 0.5^7 \tag{8.21}$$

You are asking, "What value of r would have made our data ($K = 7$) most likely?" Trying $r = 1, 2, ..., 7$, one finds that $r = 4$ maximizes (8.21), so we would take that as our guess.[5]

Now consider our parametric density setting. For "likelihood" with continous data, we don't have probabilities, but it is defined in terms of densities, as follows.

[5]By the way, here is how the Method of Moments approach would work here. For the negative binomial distribution it is known that $E(K) = r/p$, where p is the probability of "success," in this setting meaning heads. So $E(K) = 2r$. Under MM, we would set $\overline{K} = 2\widehat{r}$, where the left-hand side is the average of all values of K in our data. We only did the "experiment" once, so $\overline{K} = 6$ and we guess r to be 3.

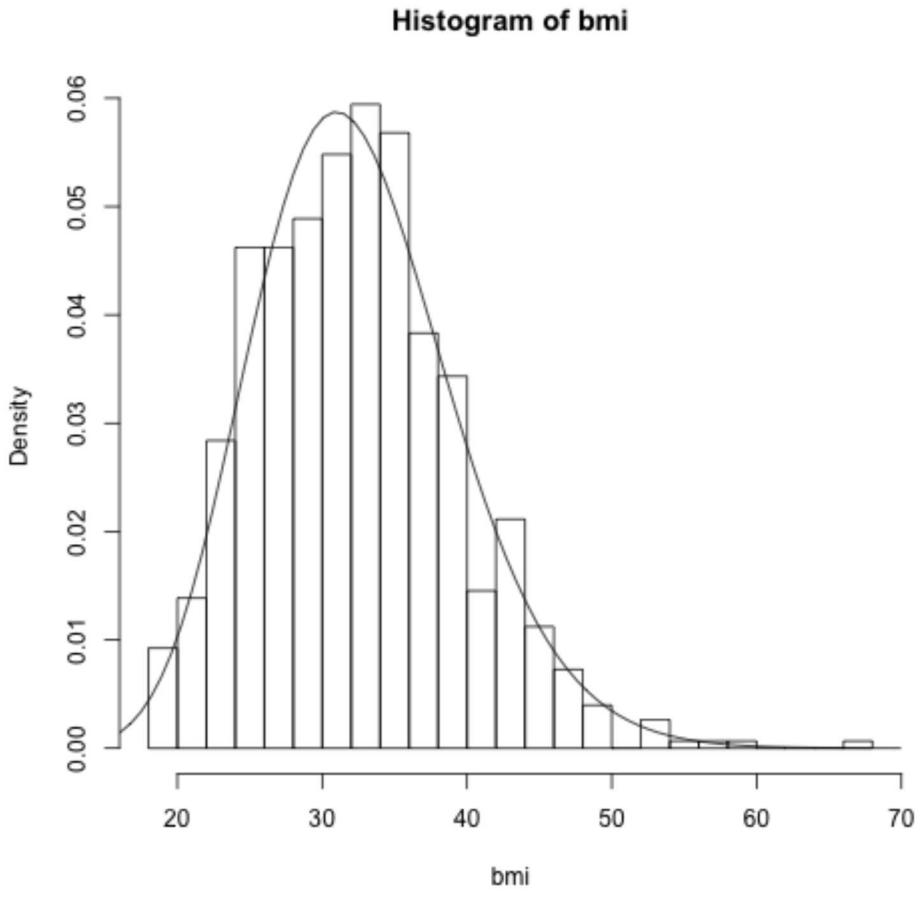

Figure 8.5: BMI, histogram and gamma fit

Say $g(t,\theta)$ is our parametric density, with θ being the parameter (possibly vector-valued). The likelihood is defined as

$$L = \Pi_i^n g(X_i, \theta) \tag{8.22}$$

We will take $\widehat{\theta}$ to be the value that maximizes L, but it's usually easier to equivalently maximize

$$l = \log L = \Sigma_i^n \log g(X_i, \theta) \tag{8.23}$$

Typically the equations have no closed-form solution, and thus must be solved numerically. R's **mle()** function does this for us.

8.4.4 Example: Humidity Data

This is from the Bike Sharing dataset on the UC Irvine Machine Learning Repository [12]. We are using the **day** data, one column of which is for humidity.

Since the humidity values are in the interval (0,1), a natural candidate for a parametric model would be a beta distribution (Section 6.7.5). Here is the code and output:

```
> bike <- read.csv('day.csv',header=TRUE)
> hum <- bike$hum
> hum <- hum[hum > 0]
> library(stats4)
> ll <- function(a,b)
+    sum(-log(dbeta(hum,a,b)))
> z <- mle(minuslogl=ll,start=list(a=1,b=1))
> z
...
Coefficients:
       a        b
6.439144 3.769841
```

The R function **mle()** has two main arguments. The first specifies a function that computes the log-likelihood, our function **ll()** here. The arguments to that function must be the parameters, which I have named **a** and **b** for "alpha" and "beta."

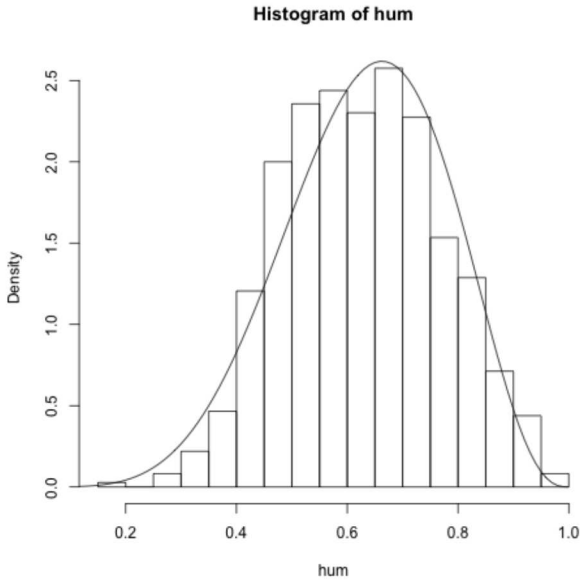

Figure 8.6: Humidity, histogram + fitted beta density

The calculation uses an iterative process, starting with an initial guess for the MLEs, then successsively refining the guess until convergence to the MLEs. The second argument to **mle()**, **start**, specifies our initial guess.

Let's plot this fitted density against the histogram:

```
> hist(hum,breaks=20,freq=FALSE)
> a <- coef(z)[1]
> b <- coef(z)[2]
> curve(dbeta(x,a,b),0,1,add=TRUE)
```

The result is shown in Figure 8.6. The caveats at the end of Section 8.4.2 apply here as well.

By the way, **mle()** also provides standard errors for the estimates α and β:

```
> vcov(z)
           a          b
a 0.11150334 0.05719833
b 0.05719833 0.03616982
```

This is the *covariance* matrix, with variances on the diagonal and covariances in the off-diagonal slots. So the standard error of $\widehat{\alpha}$ is $\sqrt{0.11150334}$, or about 0.334.

8.5 MM vs. MLE

MM and MLE are both powerful techniques, but which is better? On the one hand, MLEs can be shown to asymptotically optimal (smallest standard errors). On the other hand, MLEs require more assumptions. As with many things in data science, the best tool may depend on the setting.

8.6 Assessment of Goodness of Fit

In our examples above, we can do a visual assessment of how well our model fits the data, but it would be nice to have a quantitative measure of goodness of fit.

The classic assessment tool is the Chi-Squared Goodness of Fit Test. It is one of the oldest statistical methods (1900!), and thus in wide use. But Professor Box's remark suggests that this procedure is not the best way to gauge model fit, as the test answers the yes–or–no question, e.g. "Is the population distribution *exactly* gamma?" — of dubious relevance, given that we know *a priori* that the answer is No.[6]

A more useful measure is the *Kolmogorov-Smirnov (KS) statistic*. It actually gives us the size of the discrepancy between the fitted model family and the true population distribution. To make matters concrete, say we are fitting a beta model, with the humidity data above..

K-S is based on cdfs. Of course, the **pbeta()** function gives us the cdf for the beta family, but we also need a model-free estimate of F_X, the true population cdf of X. For the latter, we use the *empirical cdf* of X, defined as

$$\widehat{F}_X(t) = \frac{M(t)}{n} \tag{8.24}$$

where $M(t)$ is simply a count of the number of X_i that are $\leq t$. The R function **ecdf()** calculates this for us:

[6] This is a problem with significance tests in general, to be discussed in Chapter 10.

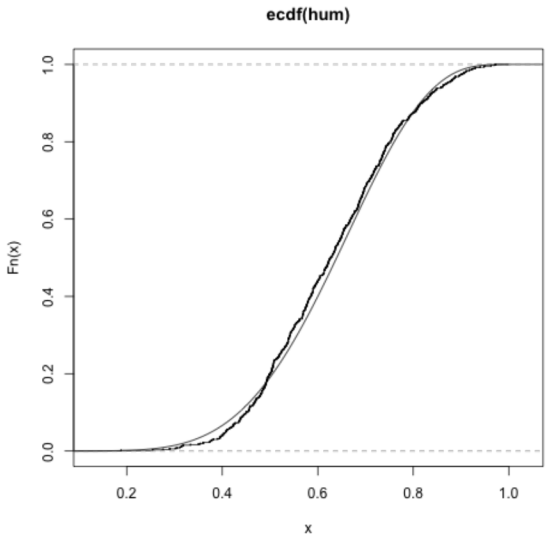

Figure 8.7: K-S analysis, humidity data

```
> ehum <- ecdf(hum)
> plot(ehum,cex=0.1)
> curve(pbeta(x,a,b),0,1,add=TRUE)
```

(The values of **a** and **b** had been previously computed.) The plot is in Figure 8.7.

So that's the visual, showing a good fit. Now to actually quantify the situation, the K-S statistic measures the fit, in terms of the maximum discrepancy, i.e. the largest difference between the empirical cdf and the fitted cdf:

```
> fitted.beta <- pbeta(hum,a,b)
> eh <- ecdf(hum)
> ks.test(eh,fitted.beta)$statistic
         D
0.04520548
...
```

Since cdf values range in [0,1], that maximum discrepancy of 0.045 is pretty good.

Of course, we mustn't forget that this number is subject to sampling vari-