Fitting a Probability Model

David Dalpiaz

STAT 3202, Autumn 2018, OSU

- Fitbit Sleep Data
  - Hours of Sleep, Normal Model
  - Awakenings, Poisson Model
- New Zealand River Data
  - River Length, Gamma Model
- Cocaine Seizure Data
  - Cocaine Potency, Beta Model
  - Cocaine Price, Some Model?
- QQ Plots
- Process Recap

> All models are wrong, but some models are useful.
>
> – George Box (https://en.wikipedia.org/wiki/All_models_are_wrong)

In this document we will discuss how to use (well-known) probability distributions to model univariate data (a single variable) in R. We will call this process "fitting" a model.

For the purpose of this document, the variables that we would like to model are assumed to be a random sample from some population. That is, we are considering models of the form

$$X_1, X_2, \ldots, X_n \overset{iid}{\sim} f(x \mid \theta)$$

Models of this form assume that the data we obtained can be modeled by random variables that are *independent* and *identically distributed* according to some distribution with parameter θ. It is important to be aware that these are **assumptions** of the model that we are using. For now, we will use these (often implicit) assumptions, but not necessarily spend much time challenging them. (Specifically, we won't criticize the independence and identical assumptions. If our data truly is a random sample, there shouldn't be much of an issue.) For other more explicit assumptions (the distribution assumption) that we make, we will learn how to criticize them.

Some natural questions should arise when modeling a variable:

- What probability models are appropriate? Specifically, what distribution should be used?
- How do you fit a particular model? That is, how do we fit a chosen probability distribution to some data?
- How do we know if a model that we fit is *good*?

To fit a probability model and answer these questions, we will generally use the following

procedure:

- **Look** at the data.
- **Assume** a probability distribution.
- **Estimate** the model parameters, that is, the parameters of the chosen distribution.
- **Check** how well the estimated model fits using a QQ-Plot.

To illustrate this process and introduce some new concepts such as a QQ-Plot, let's look a several examples.

Fitbit Sleep Data

The file `fitbit-sleep.csv` (https://daviddalpiaz.github.io/stat3202-au18/data/fitbit-sleep.csv) contains data relevant to Dave's (Professor Dalpiaz) sleep. Note that we are reading the data directly form the web into a variable named `fitbit_sleep`. Here we use the `read_csv()` function form the `readr` package which technically makes `fitbit_sleep` a `tibble` (https://cran.r-project.org/web/packages/tibble/vignettes/tibble.html), which is much like a data frame, but with a few differences we won't notice here.

```
library(readr)
fitbit_sleep = read_csv("https://daviddalpiaz.github.io/stat3202-au18/data
/fitbit-sleep.csv")
```

After reading in the data, we need to **look** at the data. One way to do this is to simply output the entire dataset. Since it is a tibble, it does so in an easy to read fashion.

```
fitbit_sleep
```

start <chr>	end <chr>	min_sle... <int>	min_aw... <int>	num_a... <int>	total_b <i
2018-09-05 11:13PM	2018-09-06 8:31AM	495	63	54	5
2018-09-05 12:04AM	2018-09-05 7:48AM	399	65	27	4
2018-09-03 11:34PM	2018-09-04 8:15AM	469	52	34	5
2018-09-02 11:24PM	2018-09-03 8:32AM	485	63	41	5
2018-09-02 12:25AM	2018-09-02 8:43AM	456	42	35	4
2018-08-31 11:59PM	2018-09-01 8:15AM	430	66	33	4
2018-08-30 11:21PM	2018-08-31 7:45AM	449	55	33	5
2018-08-29 10:16PM	2018-08-30 7:23AM	474	73	38	5
2018-08-28 10:56PM	2018-08-29 7:33AM	447	70	37	5
2018-08-28 12:09AM	2018-08-28 8:05AM	404	72	34	4

1-10 of 36 rows | 1-8 of 9 columns Previous **1** 2 3 4 Next

In RStudio, you could use `View(fitbit_sleep)` to view the data in a data viewer window.

When we **look** at the data in this manner, what we're really looking for is overall structure of the data. We like to know:

- What are the rows (observations) in this dataset?
- What are the columns (variables) in this dataset? Why type of variables are they?

Normally these questions can be answered is some form of documentation. If we are dealing with a dataset from `R` or an `R` package, often `?dataset_name` will answer these questions.

For this dataset, each row represents a night of sleep recorded by a Fitbit Versa. The columns are:

- **`start`** - date and time Fitbit recognized beginning of sleep
- **`end`** - date and time Fitbit recognized end of sleep
- **`min_sleep`** - total time spent asleep in minutes
- **`min_awake`** - total time spent awake in minutes
- **`num_awake`** - number of awakenings during sleep
- **`total_bed`** - total time in bed in minutes, including both asleep and awake
- **`min_REM`** - time spent in the REM sleep stage in minutes
- **`min_light`** - time spent in the light sleep stages (NREM 1 and 2) in minutes
- **`min_deep`** - time spent in the deep sleep stages (NREM 3 and 4) in minutes

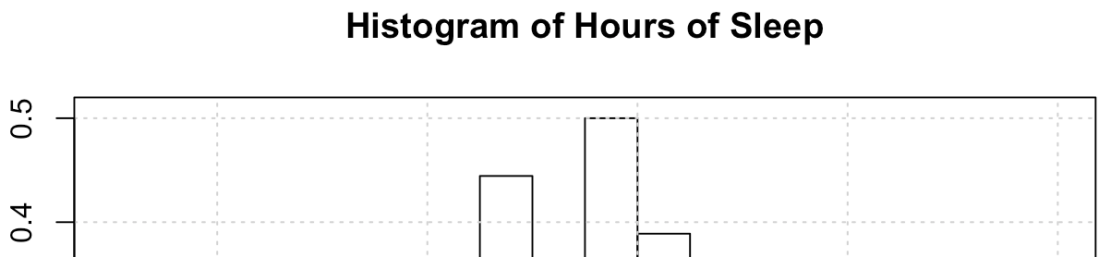
Note that technically these are all simply what the Fitbit has measured. There is certainly a lot of measurement error. (These quantities are being estimated by the Fitbit based on movement and heart rate.)

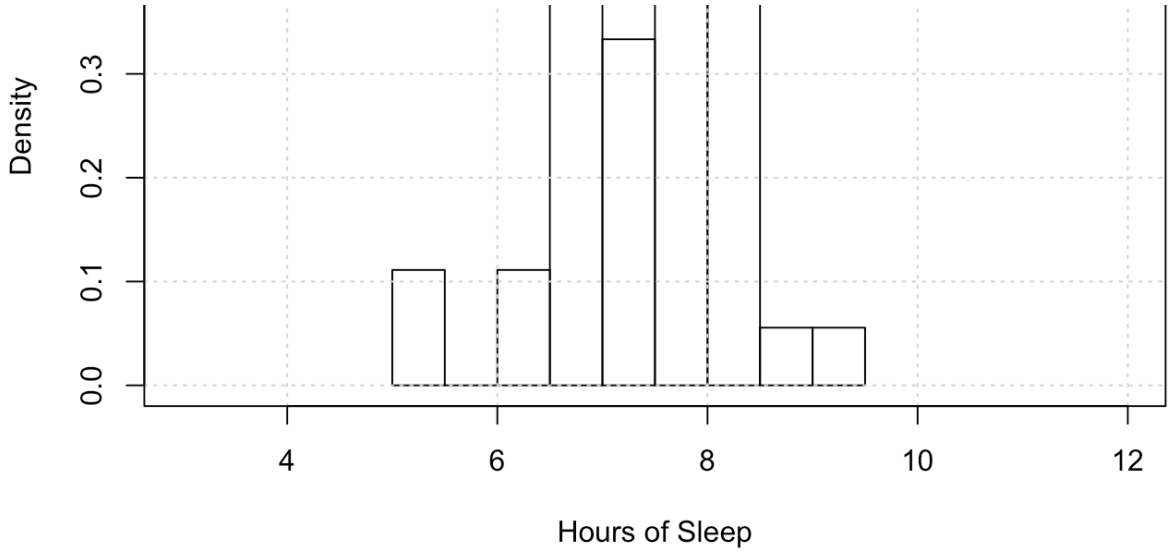
Hours of Sleep, Normal Model

Let's create a new variable called `hour_sleep` that measures the hours of sleep each night that we will attempt to model.

```
hour_sleep = fitbit_sleep$min_sleep / 60
```

```
hist(hour_sleep, probability = TRUE, xlim = c(3, 12),
     main = "Histogram of Hours of Sleep",
     xlab = "Hours of Sleep")
grid()
box()
```





This histogram gives some insight into why we wouldn't simply use the empirical distribution. In some sense, we can think of using the empirical distribution as "overfitting." It places zero probability on too much of the range of the variable, and places too much probability on the observations we have observed. By fitting a probability model, we will "smooth" away these issues.

Here, we can argue that a continuous distribution would make the most sense. While the Fitbit only records sleep to the nearest minute, in reality sleep time could take on any positive real number between 0 and 24 hours.

Without giving it too much though, we'll attempt to model this data with a normal distribution. While a normal distribution does allow for the possibility of any real number, the vast majority of the probability is within three standard deviation of the mean.

To "fit" the normal model, we simply need to estimate its two parameters, μ and $^2. Recall that the probability density function for a normal random variable is given by

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\left(\frac{x-\mu}{\sigma}\right)^2\right], \quad x \in \mathbb{R}, \ \mu \in \mathbb{R}, \ \sigma^2 > 0$$

The maximum likelihood estimators for μ and σ^2 are given by

$$\hat{\mu} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Notice that $\bar{x}$ can be calculated with the `mean()` function in `R`. However, the `var()` function in `R` calculates

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

While this is not the MLE (or MoM), the difference is so small that we'll use it since it makes our life easier.

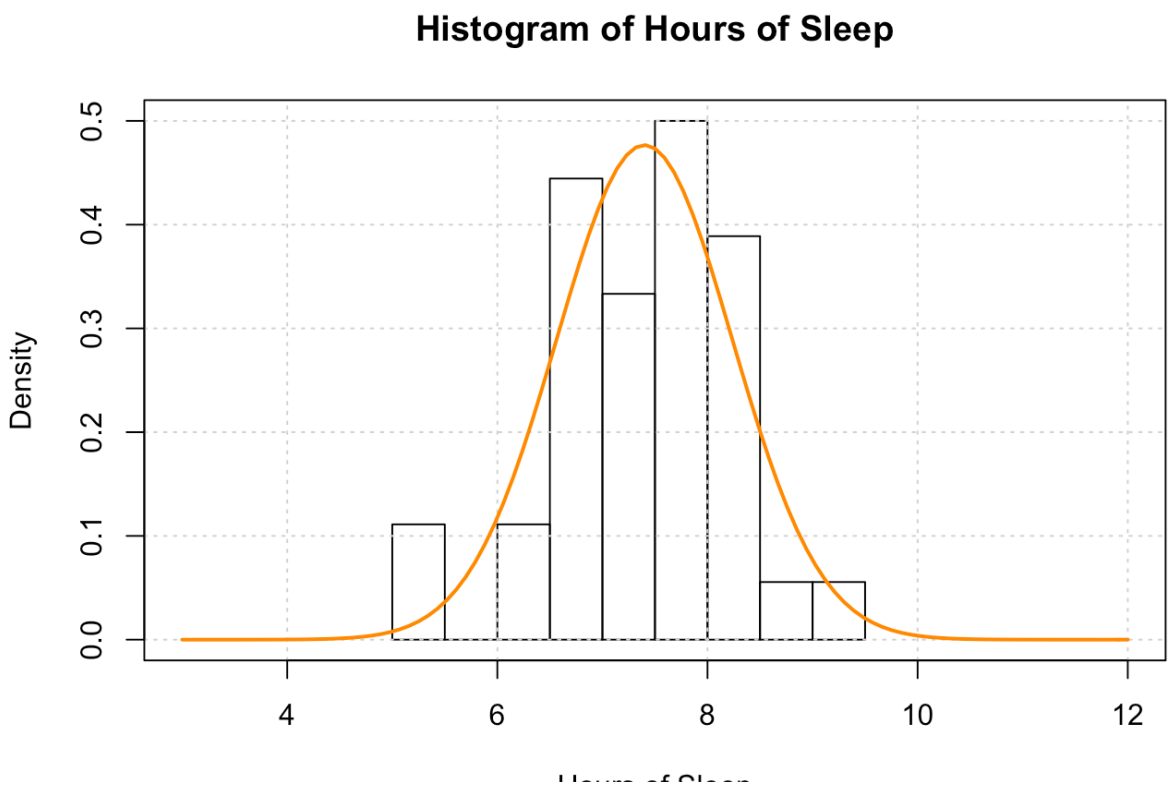
```
hsleep_xbar = mean(hour_sleep)
hsleep_s2   = var(hour_sleep)
c(hsleep_xbar, hsleep_s2)
```

```
## [1] 7.400000 0.700254
```

Here we see these two estimates. In this dataset, Dave sleeps on average 7.4 hours, with a standard deviation of about 0.7 hours.

TO get a sense of how well this model fits, we add its density to the histogram of the data. Note that the `dnorm()` function takes the standard deviation, not the variance as an argument.

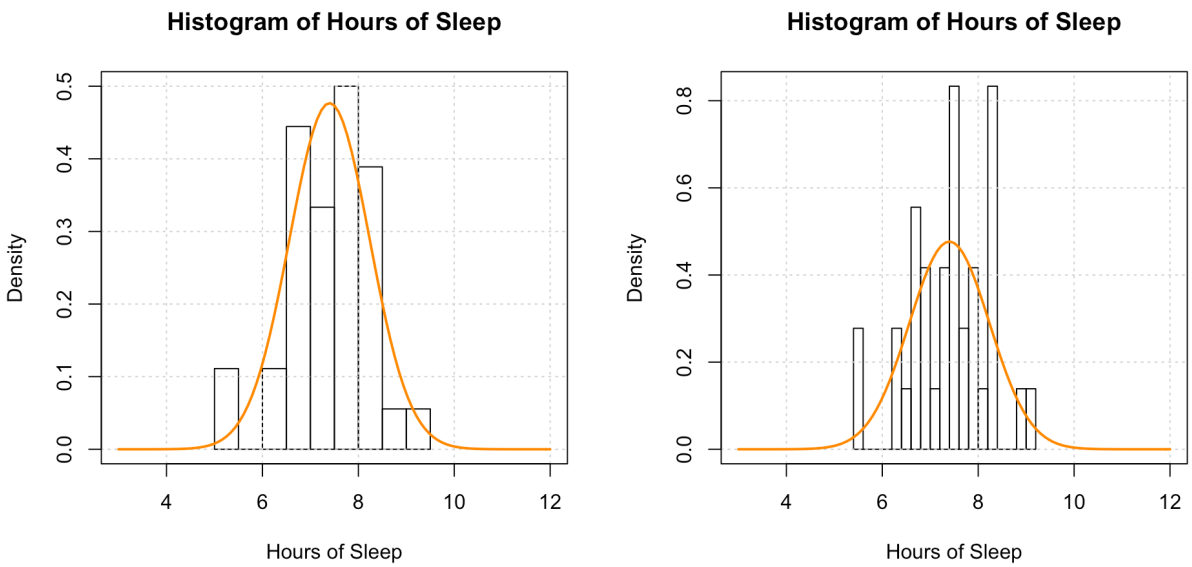
```
hist(hour_sleep, probability = TRUE, xlim = c(3, 12),
     main = "Histogram of Hours of Sleep",
     xlab = "Hours of Sleep")
grid()
box()
curve(dnorm(x, mean = mean(hour_sleep), sd = sd(hour_sleep)),
      add = TRUE, col = "darkorange", lwd = 2)
```



Hours of Sleep

It turns out that it's difficult to tell how well this fits using only a histogram. It actually becomes really difficult when you realize that histograms are very sensitive to bin width selection.

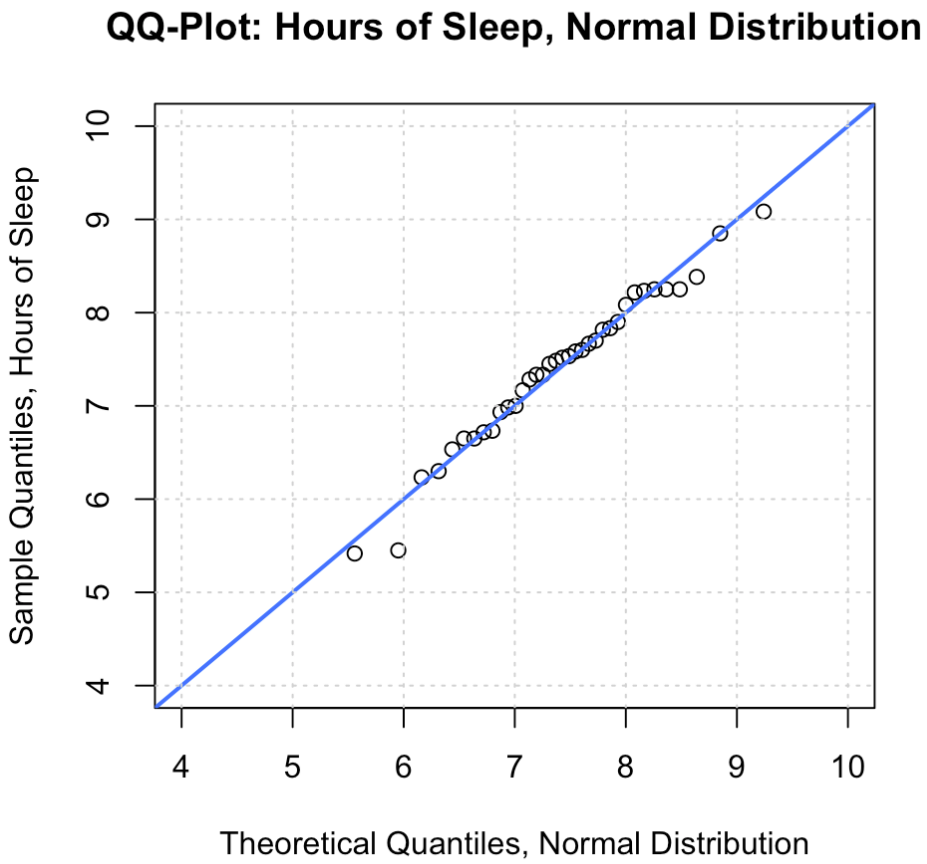
```
par(mfrow = c(1, 2))

# default histogram
hist(hour_sleep, probability = TRUE, xlim = c(3, 12),
     main = "Histogram of Hours of Sleep",
     xlab = "Hours of Sleep")
grid()
box()
curve(dnorm(x, mean = mean(hour_sleep), sd = sd(hour_sleep)),
      add = TRUE, col = "darkorange", lwd = 2)

# modified bin selection
hist(hour_sleep, probability = TRUE, xlim = c(3, 12), breaks = 20,
     main = "Histogram of Hours of Sleep",
     xlab = "Hours of Sleep")
grid()
box()
curve(dnorm(x, mean = mean(hour_sleep), sd = sd(hour_sleep)),
      add = TRUE, col = "darkorange", lwd = 2)
```



Instead of using a histogram to check how well our model fits, we'll instead use a quantile-quantile plot, often simply called a **QQ plot**. We'll save the details for later, but the idea of a QQ plot is simple. One one axis, normally the y axis, we plot the actual data that we observed. On the other axis, the x axis, we plot where we would expect the data to be if it was sampled from the distribution that we fit. We then add a line for $y = x$. If the model fits well, the points should fall close to the line. If the model fits poorly, the points will

deviate form the line.

```
qqplot(x = qnorm(ppoints(hour_sleep),
                 mean = mean(hour_sleep), sd = sd(hour_sleep)),
       y = hour_sleep,
       xlim = c(4, 10), ylim = c(4, 10),
       main = "QQ-Plot: Hours of Sleep, Normal Distribution",
       xlab = "Theoretical Quantiles, Normal Distribution",
       ylab = "Sample Quantiles, Hours of Sleep")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```



Without being an “expert” in QQ plots it might not immediately be clear if these points are close enough to the line. Reading QQ plots is actually a bit of an art. For now, we’ll say that this is pretty good! (More on “good” and “bad” QQ plots later.)

To reiterate why we like this model better than using the empirical distribution, we calculate the probability that Dave sleeps between 5.5 and 6 hours two ways.

```
# using empirical distribution
mean(5.5 < hour_sleep & hour_sleep < 6)
```

```
## [1] 0
```

```
# using normal model
diff(pnorm(q = c(5.5, 6), mean = mean(hour_sleep), sd = sd(hour_sleep)))
```

```
## [1] 0.03557407
```

Note that the smallest observations of `hour_sleep` in this dataset is `min(hour_sleep)`. Thus it seems reasonable to place positive probability on sleeping between 5.5 and 6 hours. This is accomplished by the normal model, but not by using the empirical distribution.

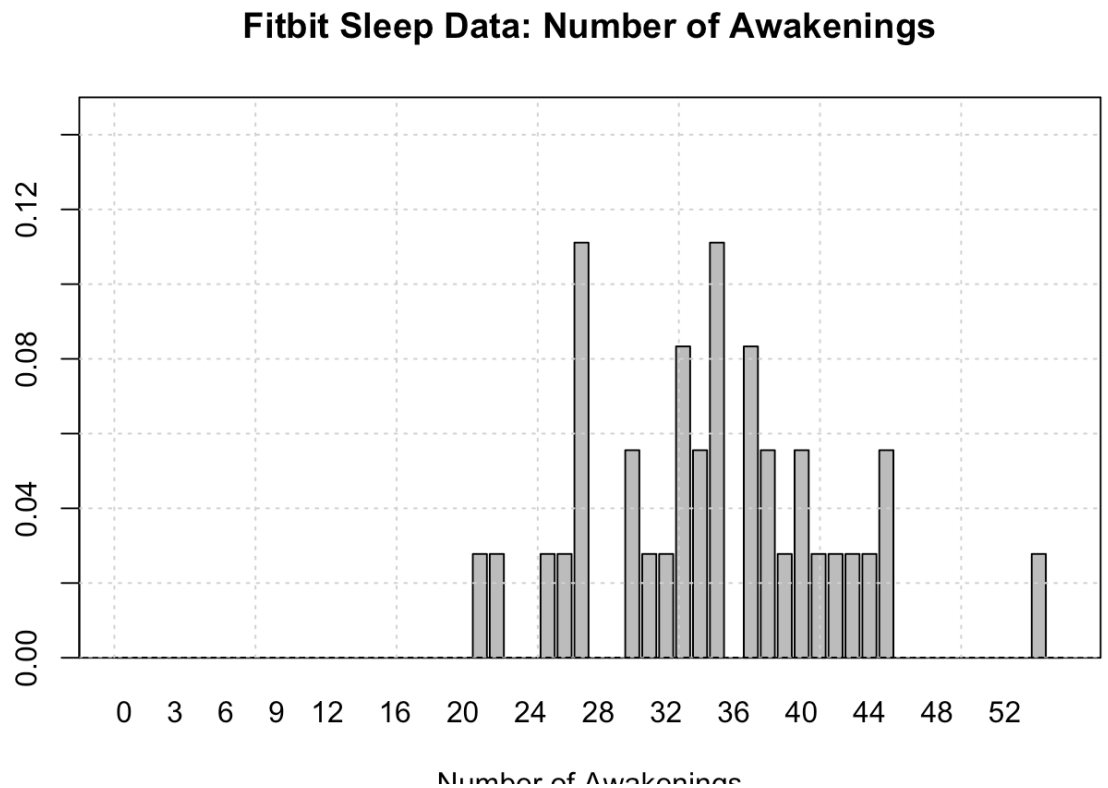
Awakenings, Poisson Model

Now let's try to model the number of times Dave wakes up during the night.

```
awakes = fitbit_sleep$num_awake
```

Since this is "count" data, it actually is discrete. To visualize a discrete distribution, we often use a barplot instead of a histogram.

```
barplot(table(factor(awakes, levels = 0:55)) / length(awakes), ylim = c(0,
0.15),
        names.arg = as.character(0:55),
        xlab = "Number of Awakenings",
        main = "Fitbit Sleep Data: Number of Awakenings")
box()
grid()
```



Number of Awakenings

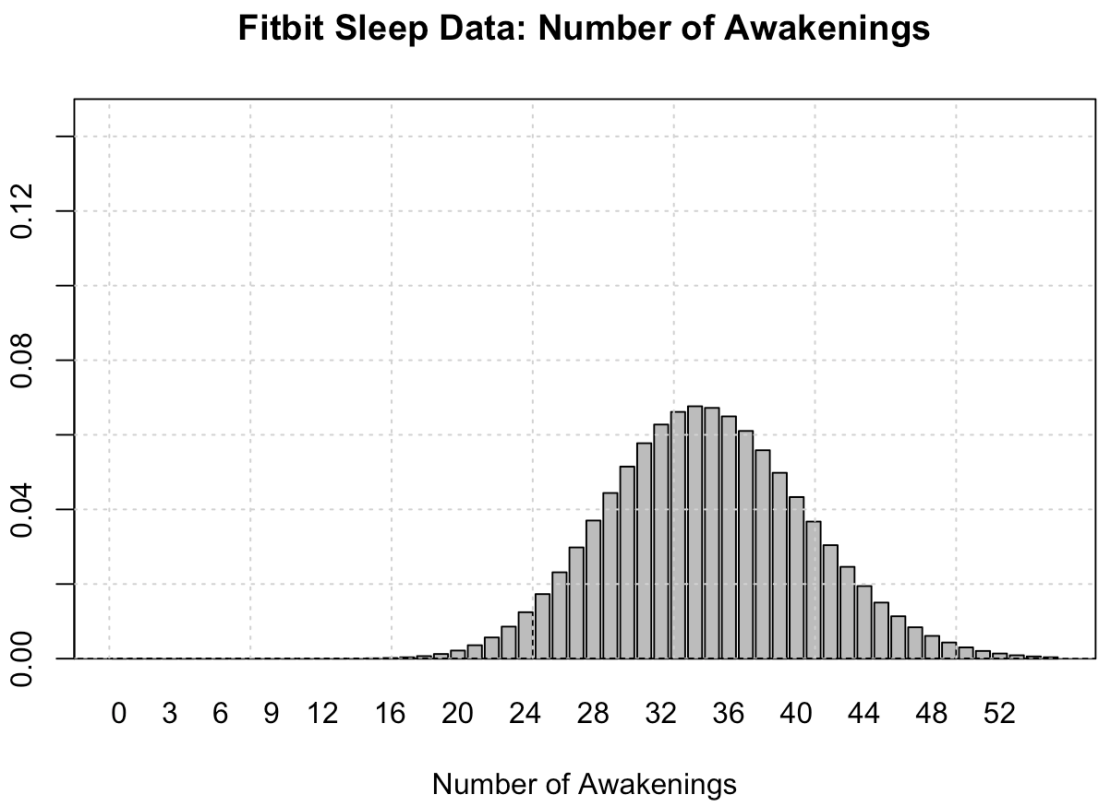
Let's try a Poisson model. To "fit" the model, we need to estimate λ, which we have repeatedly seen can be done using the sample mean, $\bar{x}$.

```
mean(awakes)
```

```
## [1] 34.77778
```

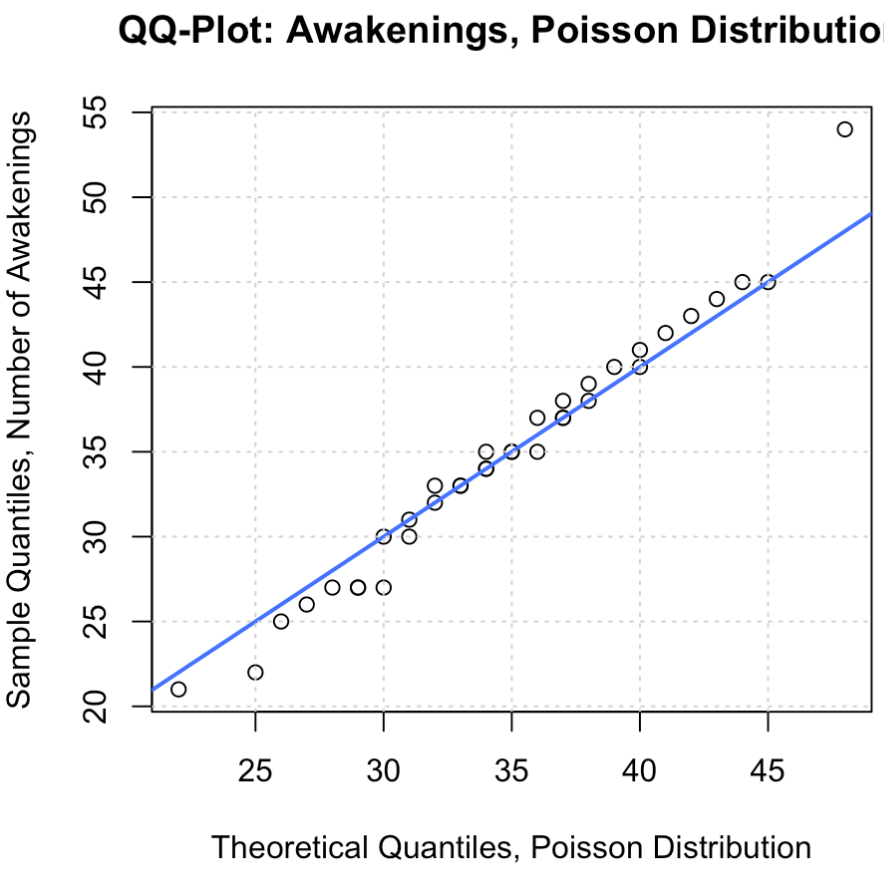
When we represent this distribution as a barplot, we obtain the following.

```
barplot(dpois(0:55, lambda = mean(awakes)), ylim = c(0, 0.15),
        names.arg = as.character(0:55),
        xlab = "Number of Awakenings",
        main = "Fitbit Sleep Data: Number of Awakenings")
box()
grid()
```



It's hard to tell how well this "fits," so we create a QQ plot.

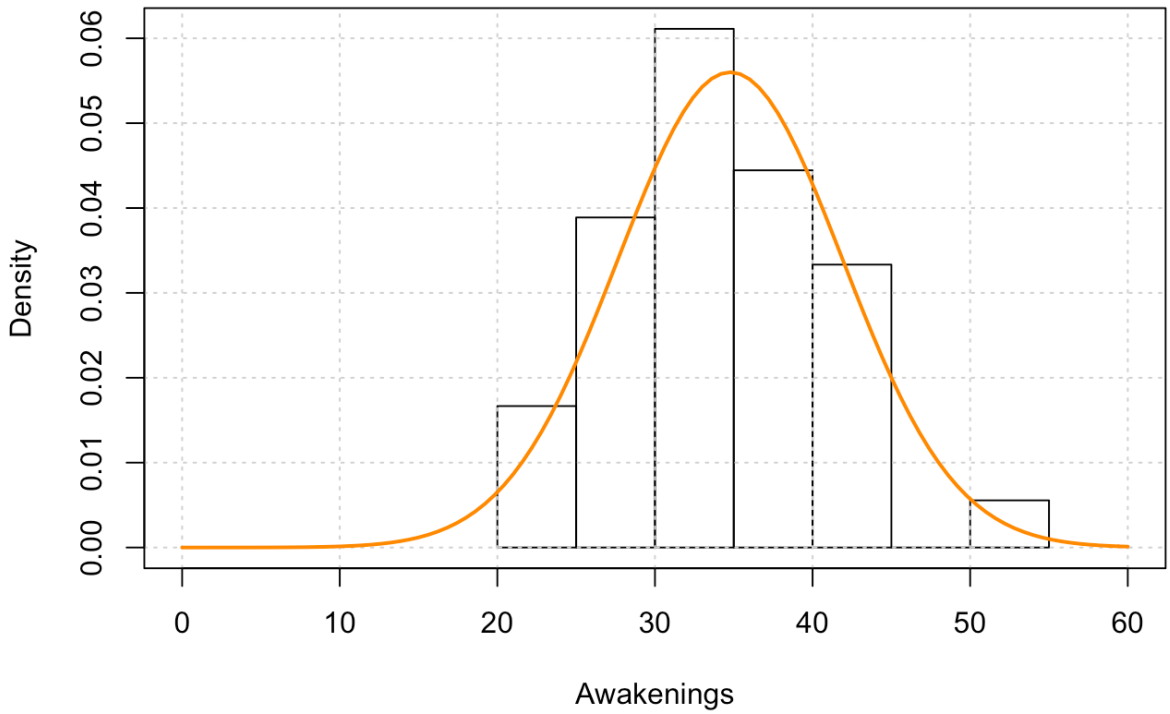
```
qqplot(x = qpois(ppoints(awakes), lambda = mean(awakes)),
       y = awakes,
       main = "QQ-Plot: Awakenings, Poisson Distribution",
       xlab = "Theoretical Quantiles, Poisson Distribution",
       ylab = "Sample Quantiles, Number of Awakenings")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```



This QQ plot looks pretty good!

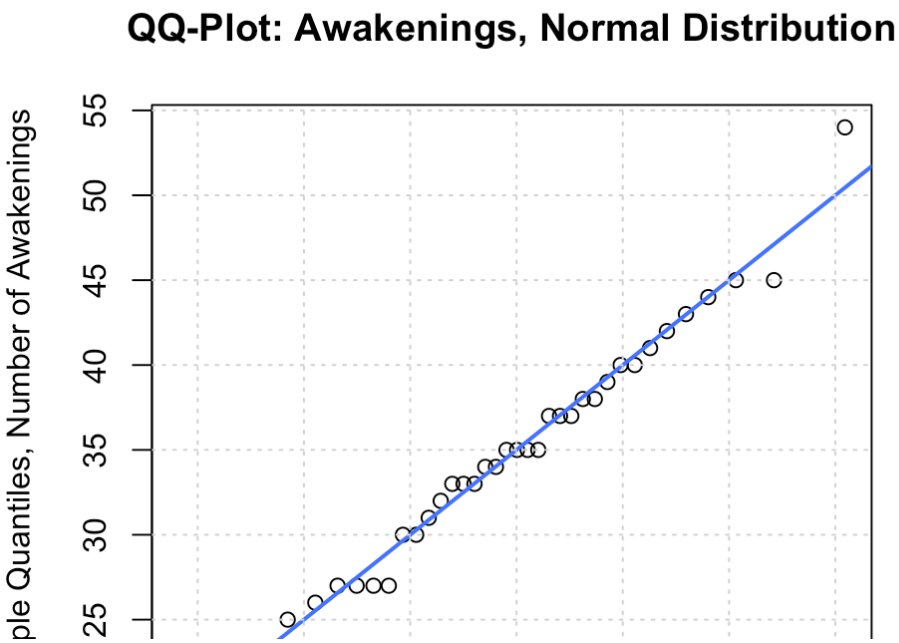
However, when we represented the fitted Poisson model as a barplot, its shape may have been very familiar. Even though it is a continuous model, let's try a normal model.

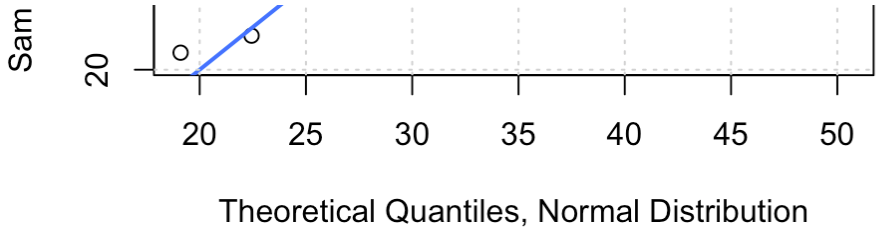
```
hist(awakes, probability = TRUE, xlim = c(0, 60),
     main = "Histogram of Number of Awakenings",
     xlab = "Awakenings")
grid()
box()
curve(dnorm(x, mean = mean(awakes), sd = sd(awakes)),
      add = TRUE, col = "darkorange", lwd = 2)
```



The fitted normal model on top of the histogram looks pretty good...

```
qqplot(x = qnorm(ppoints(awakes),
                 mean = mean(awakes), sd = sd(awakes)),
       y = awakes,
       main = "QQ-Plot: Awakenings, Normal Distribution",
       xlab = "Theoretical Quantiles, Normal Distribution",
       ylab = "Sample Quantiles, Number of Awakenings")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```





QQ plot looks good as well! Both of these models are reasonable. This is largely because a normal distribution is a decent approximation of a Poisson distribution, especially for "large" values of λ.

New Zealand River Data

The data stored in `nz-rivers.txt` (https://daviddalpiaz.github.io/stat3202-au18/data/nz-rivers.txt) records information about rivers in New Zealand.

```
nz_rivers = read_table2("https://daviddalpiaz.github.io/stat3202-au18/data
/nz-rivers.txt")
```

Notice that here we used the `read_table2()` function to import the data. This is because this is *not* a `.csv` file. It is actually delimited by whitespace. If you use the Import Dataset tool in RStudio, it will automatically detect this.

```
nz_rivers
```

River <chr>	Length <int>	FlowsInto <chr>
Clarence	209	Pacific
Conway	48	Pacific
Waiau	169	Pacific
Hurunui	138	Pacific
Waipara	64	Pacific
Ashley	97	Pacific
Waimakariri	161	Pacific
Selwyn	95	Pacific
Rakaia	145	Pacific
Ashburton	90	Pacific
1-10 of 41 rows	Previous **1** 2 3 4 5 Next	

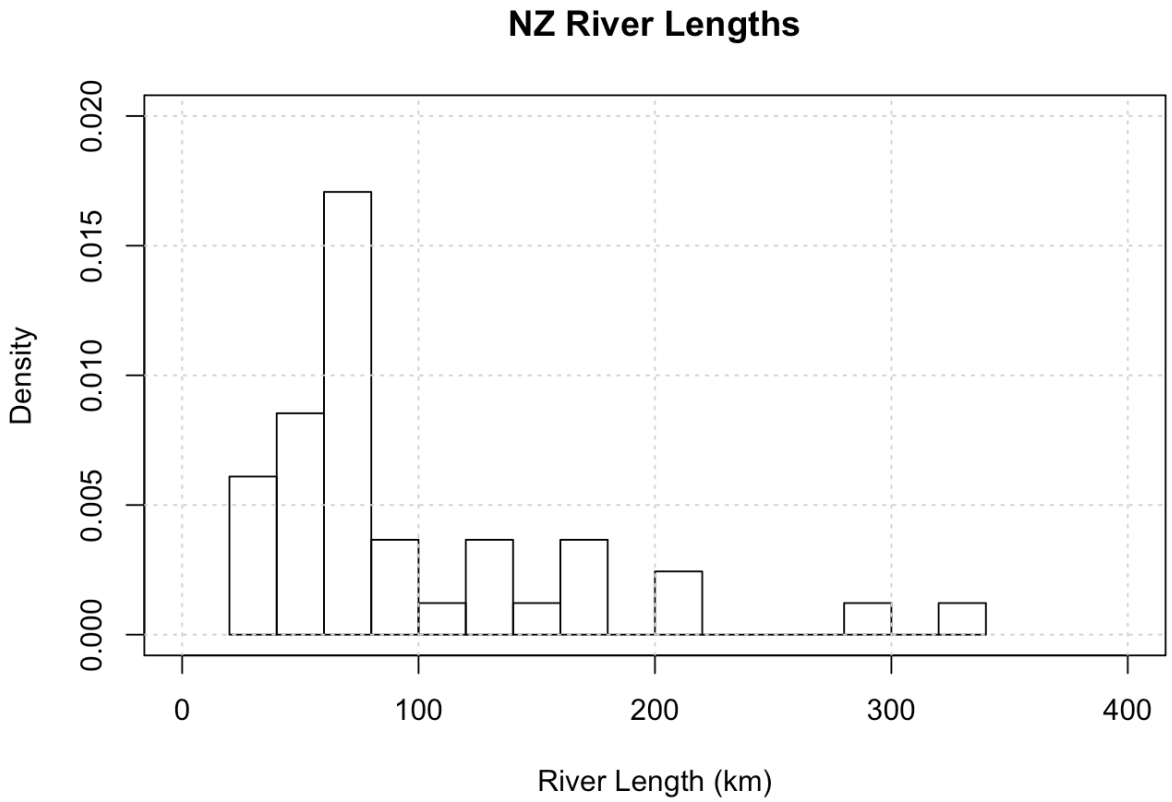
This dataset contains information on rivers in New Zealand including their name, length,

and the body of water the river flows into.

River Length, Gamma Model

We will try to model the length of the rivers, which in this dataset is given in kilometers.

```
hist(nz_rivers$Length, probability = TRUE,
     main = "NZ River Lengths", xlab = "River Length (km)",
     ylim = c(0, 0.020), xlim = c(0, 400), breaks = 12)
box()
grid()
```



By looking at the histogram, and simply understanding that rivers will have some positive length, we would like to consider probability distributions that places density on positive numbers. A number of probability distributions have this feature, including: chi-square, F, exponential, and gamma. Let's try gamma.

Recall that the probability density function of a gamma random variable is given by

$$f(x \mid \alpha, \beta) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta}, x > 0, \alpha > 0, \beta > 0$$

Note that `R` will use `shape` instead of α and `scale` instead of β.

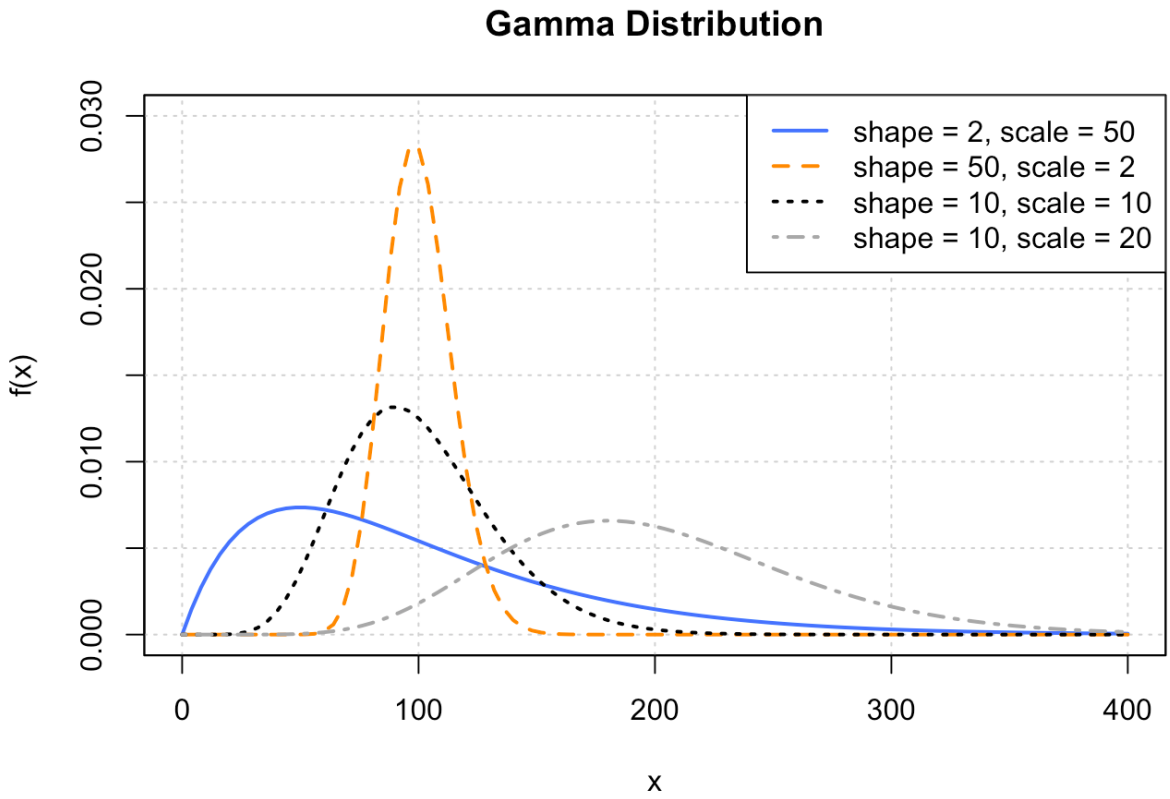
Also, the gamma distribution has

$$\mathrm{E}[X] = \alpha\beta$$

and

$$\mathrm{Var}[X] = \alpha\beta^2$$

Below are plots of gamma with various values for the parameters. By estimating these parameters from the data, hopefully we can find a gamma distribution that matches the shape of the above histogram.



While it is not possible to find an analytic solution for the MLE, given $X_1, X_2, \ldots X_n$ assumed to be a random sample from a gamma distribution, we can use method of moments to obtain estimators:

$$\tilde{\alpha} = \frac{\bar{x}^2}{\overline{x^2} - \bar{x}^2}$$

$$\tilde{\beta} = \frac{\overline{x^2} - \bar{x}^2}{\bar{x}}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\overline{x^2} = \frac{1}{n}\sum_{i=1}^{n} x_i^2$$

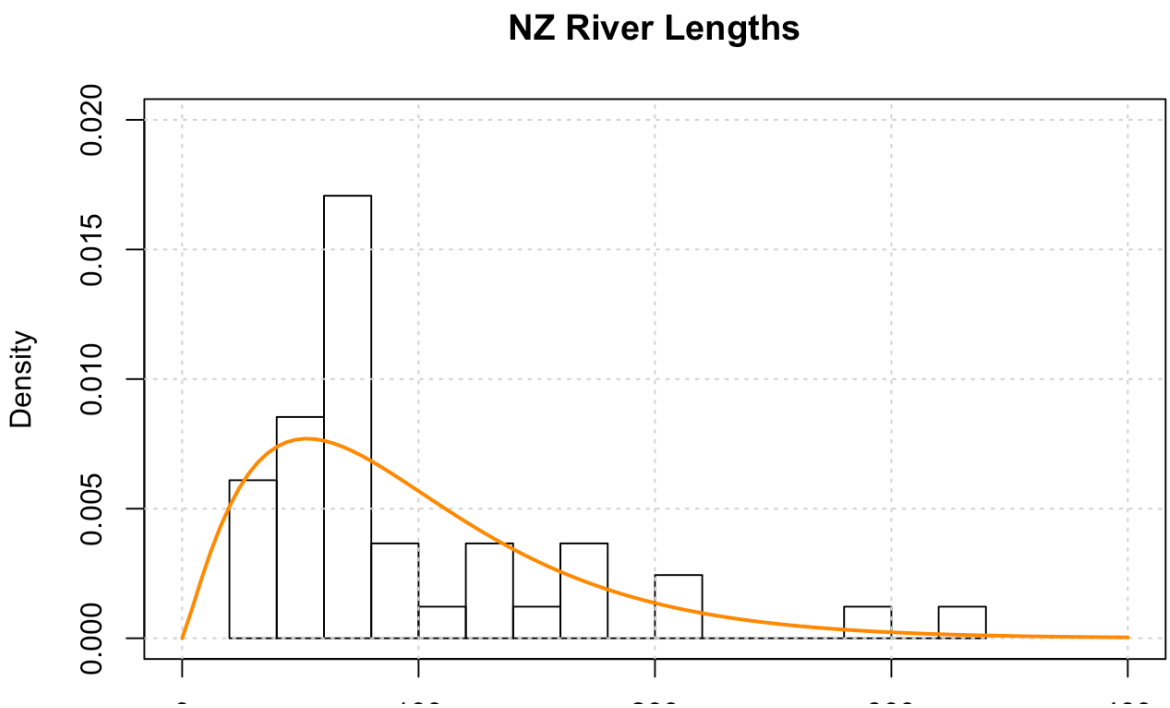
We can apply these estimators to this data in `R`.

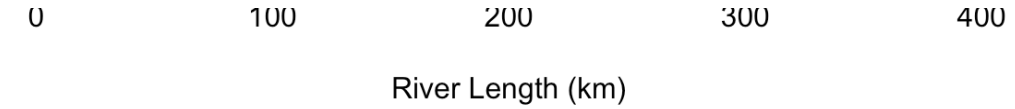
```
# calculating sample moments
len_samp_moment_1 = mean(nz_rivers$Length)
len_samp_moment_2 = mean(nz_rivers$Length ^ 2)

# method of moments estimators
len_alpha_mom = len_samp_moment_1 ^ 2 / (len_samp_moment_2 - len_samp_mome
nt_1 ^ 2)
len_beta_mom  = len_samp_moment_1 / len_alpha_mom

# estimates for this dataset
c(len_alpha_mom, len_beta_mom)
```

```
## [1]  2.189716 44.342529
```

We then add this fitted gamma model to the histogram.

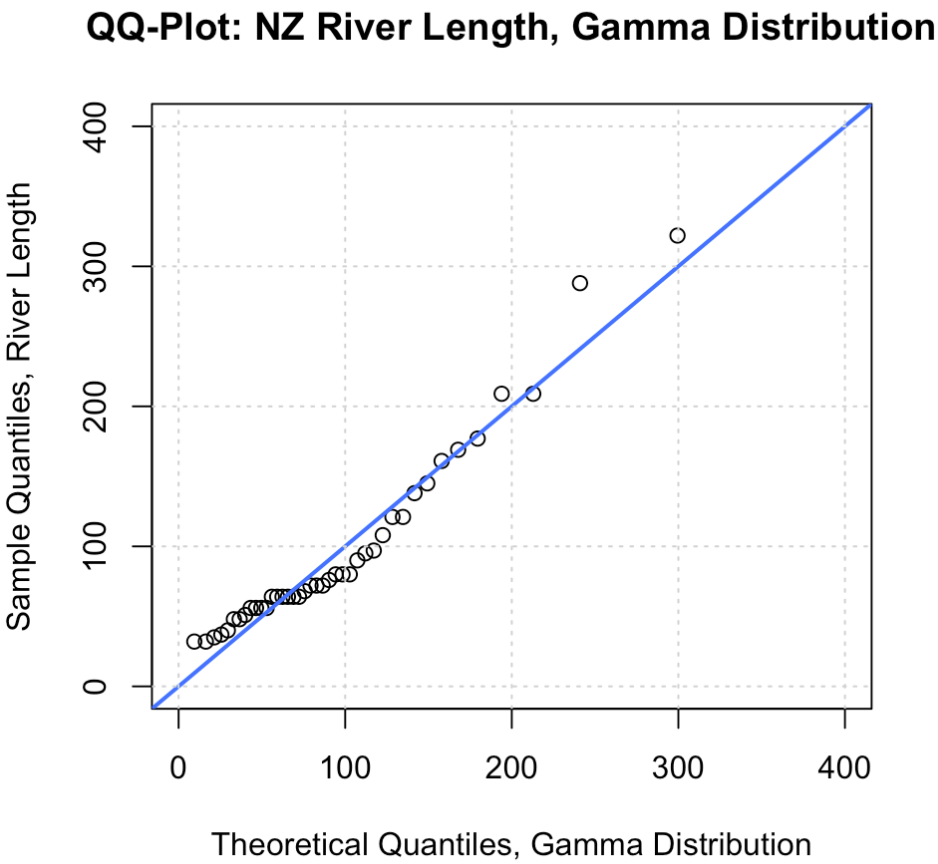
```
hist(nz_rivers$Length, probability = TRUE,
     main = "NZ River Lengths", xlab = "River Length (km)",
     ylim = c(0, 0.020), xlim = c(0, 400), breaks = 12)
box()
grid()
curve(dgamma(x, shape = len_alpha_mom, scale = len_beta_mom),
      add = TRUE, col = "darkorange", lwd = 2)
```





Then to better verify the fit, we create a QQ plot.

```
qqplot(x = qgamma(ppoints(nz_rivers$Length),
                  shape = len_alpha_mom, scale = len_beta_mom),
       y = nz_rivers$Length,
       xlim = c(0, 400), ylim = c(0, 400),
       main = "QQ-Plot: NZ River Length, Gamma Distribution",
       xlab = "Theoretical Quantiles, Gamma Distribution",
       ylab = "Sample Quantiles, River Length")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```



This QQ plot isn't the best, but it isn't terrible.

Cocaine Seizure Data

This dataset comes from STRIDE, the System to Retrieve Information from Drug Evidence. It contains all concaine seizures in the US from 2007 that have a known weight. For a full dataset description, use `?cocaine` after loading the `ggvis` package.

```
cocaine = ggvis::cocaine
cocaine
```

state <chr>	potency <dbl>	weight <dbl>	month <int>	price <dbl>
WA	77	217	1	5000
CT	51	248	1	4800
FL	68	43	1	3500
OH	69	123	1	3500
MS	75	118	1	3400
VA	73	127	1	3000
FL	54	50	1	3000
IL	58	140	1	2800
GA	77	127	1	2600
MS	49	74	1	2600

1-10 of 3,380 rows Previous **1** 2 3 4 5 6 … 338 Next

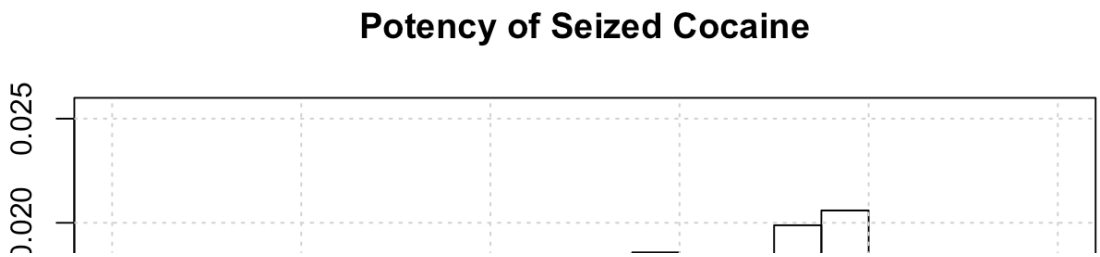
Cocaine Potency, Beta Model

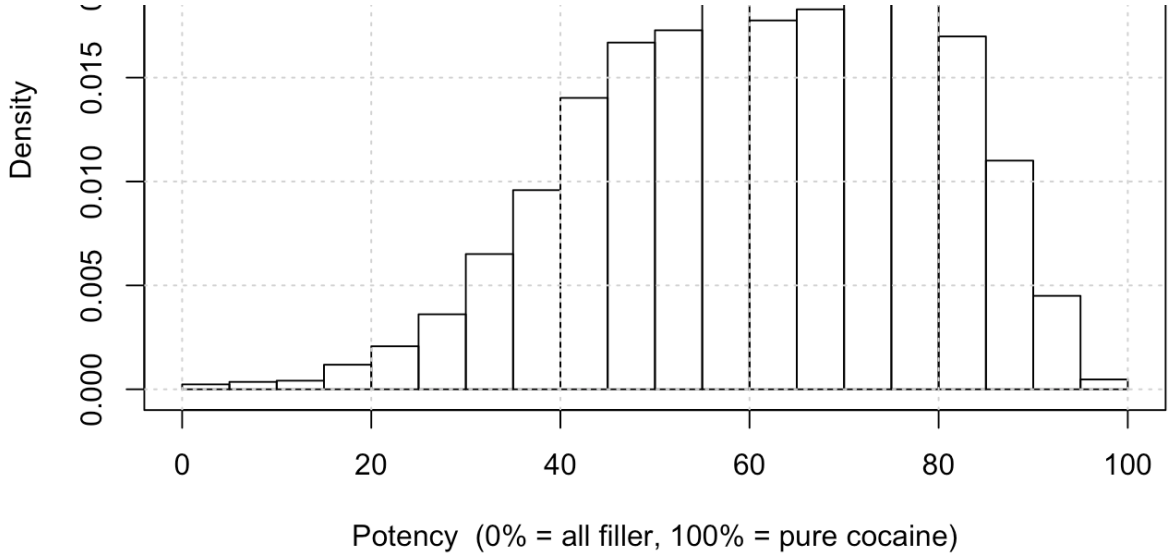
First, let's model the potency of the cocaine seized.

```
range(cocaine$potency)
```

```
## [1]  2 98
```

```
hist(cocaine$potency, probability = TRUE,
     main = "Potency of Seized Cocaine", xlab = "Potency  (0% = all fille
r, 100% = pure cocaine)",
     ylim = c(0, 0.025), xlim = c(0, 100), breaks = 15)
box()
grid()
```





We see that potency can range from 0 (all filler) to 100 (pure cocaine). We aren't aware of any distributions on the interval from 0 to 100, but we do know one on the interval from 0 to 100, the beta distribution.

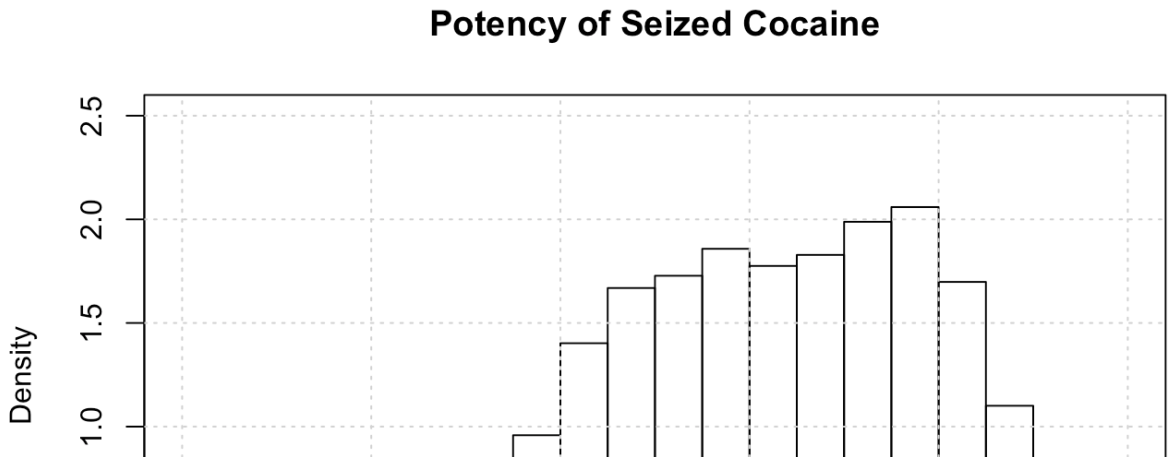
In order to use the beta distribution, we'll first perform a quick transformation.

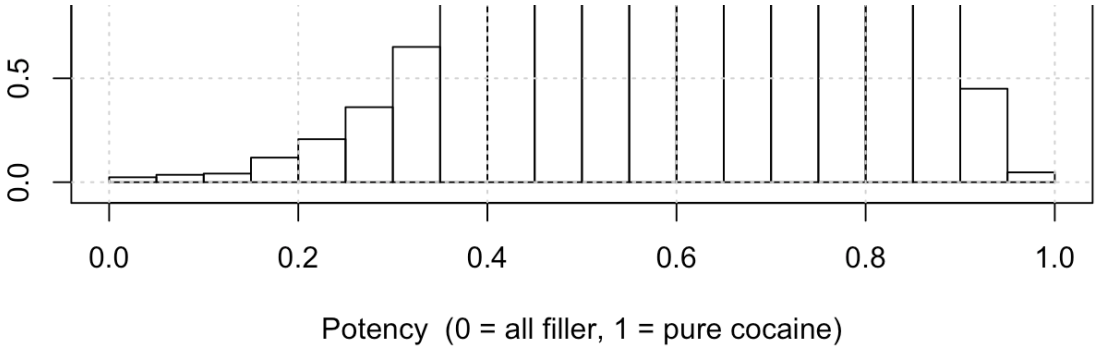
```
coke_pot = cocaine$potency / 100
```

```
range(coke_pot)
```

```
## [1] 0.02 0.98
```

```
hist(coke_pot, probability = TRUE,
     main = "Potency of Seized Cocaine", xlab = "Potency  (0 = all filler,
1 = pure cocaine)",
     ylim = c(0, 2.5), xlim = c(0, 1), breaks = 15)
box()
grid()
```





Notice that the shape stays exactly the same.

Recall, that a random variable X that follows a beta distribution has probability density function

$$f(x \mid \alpha, \beta) = \left[\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right] x^{\alpha-1}(1-y)^{\beta-1}, 0 \leq x \leq 1, \alpha > 0, \beta > 0$$

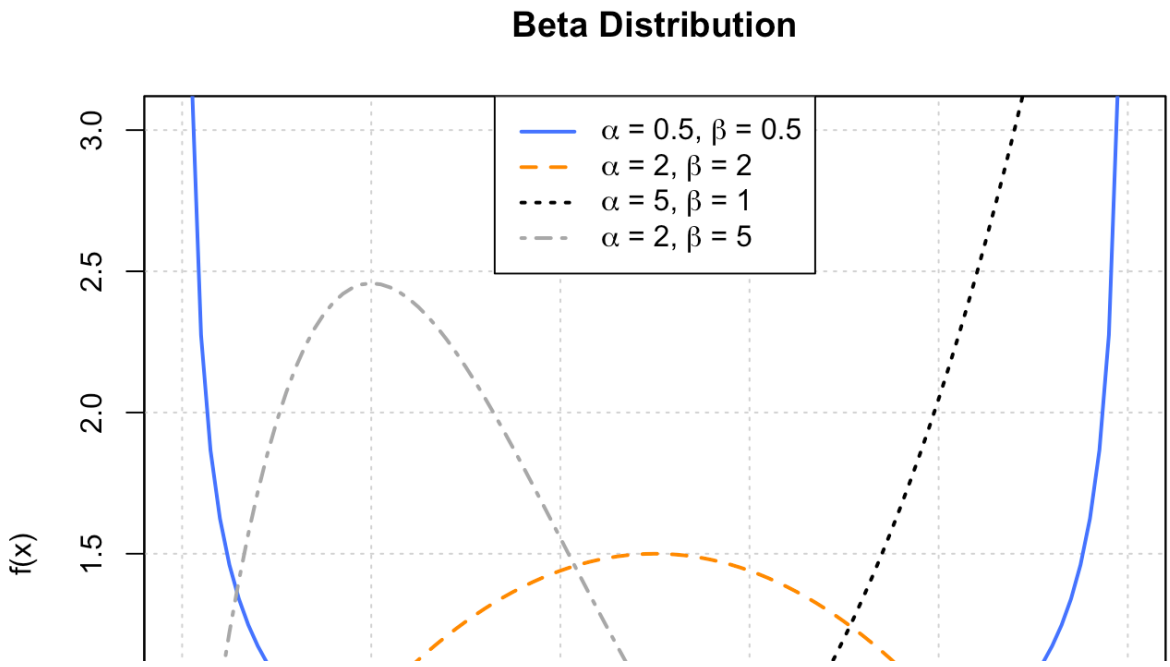
Also, the beta distribution has

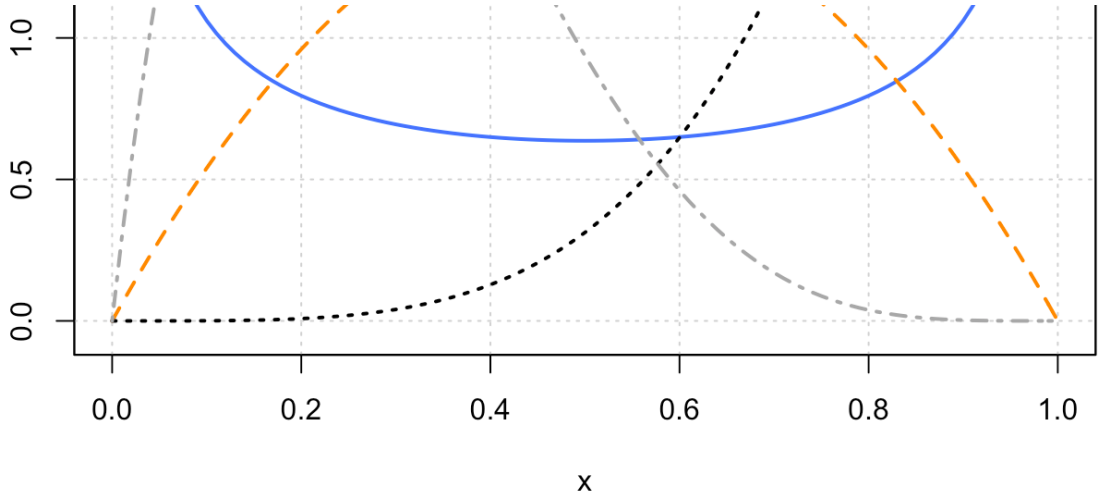
$$\mathrm{E}[X] = \frac{\alpha}{\alpha+\beta}$$

and

$$\mathrm{Var}[X] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Below are plots of beta with various values for the parameters. By estimating these parameters from the data, hopefully we can find a gamma distribution that matches the shape of the above histogram.





While it is not possible to find an analytic solution for the MLE, given $X_1, X_2, \ldots X_n$ assumed to be a random sample from a beta distribution, we can use method of moments to obtain estimators:

$$\tilde{\alpha} = \bar{x}\left(\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right)$$

$$\tilde{\beta} = (1-\bar{x})\left(\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Let's apply these estimators to this dataset.

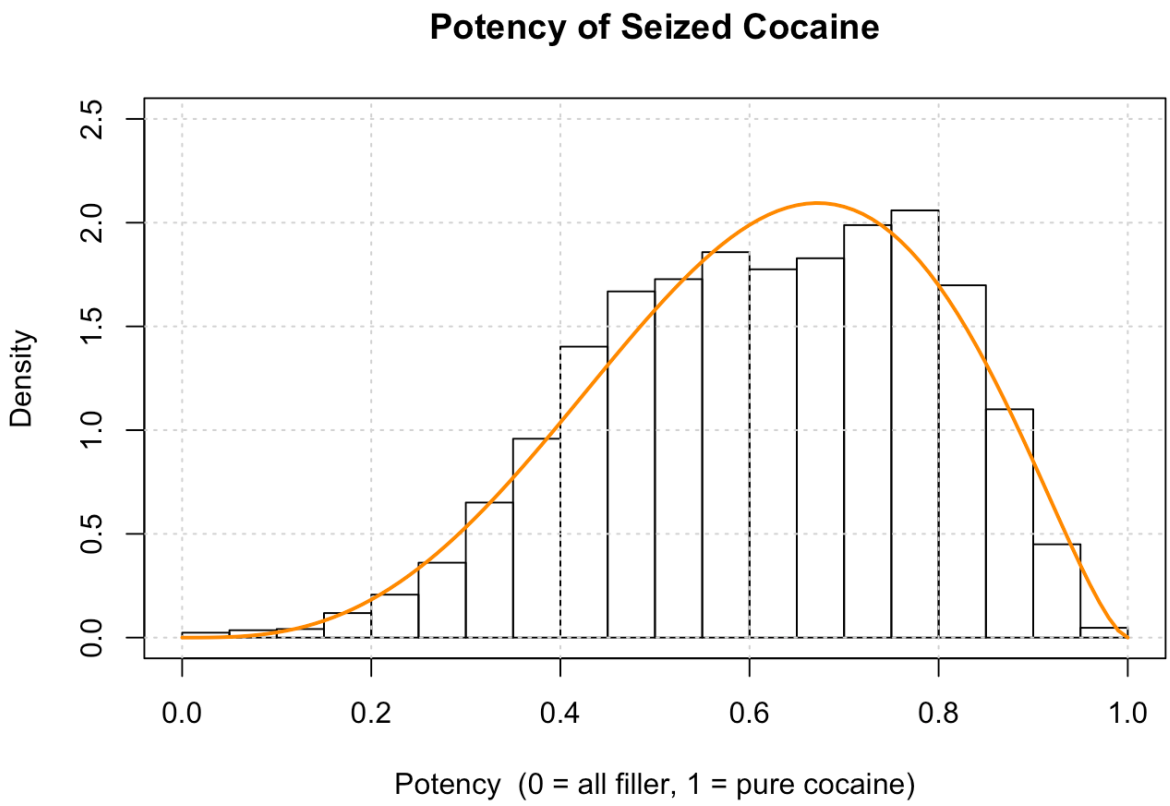
```
coke_pot_alpha_hat = mean(coke_pot) * ((mean(coke_pot) * (1 - mean(coke_po
t))) / (var(coke_pot)) - 1)
coke_pot_beta_hat = (1 - mean(coke_pot)) * ((mean(coke_pot) * (1 - mean(co
ke_pot))) / (var(coke_pot)) - 1)
```

```
c(coke_pot_alpha_hat, coke_pot_beta_hat)
```

```
## [1] 4.138140 2.531346
```

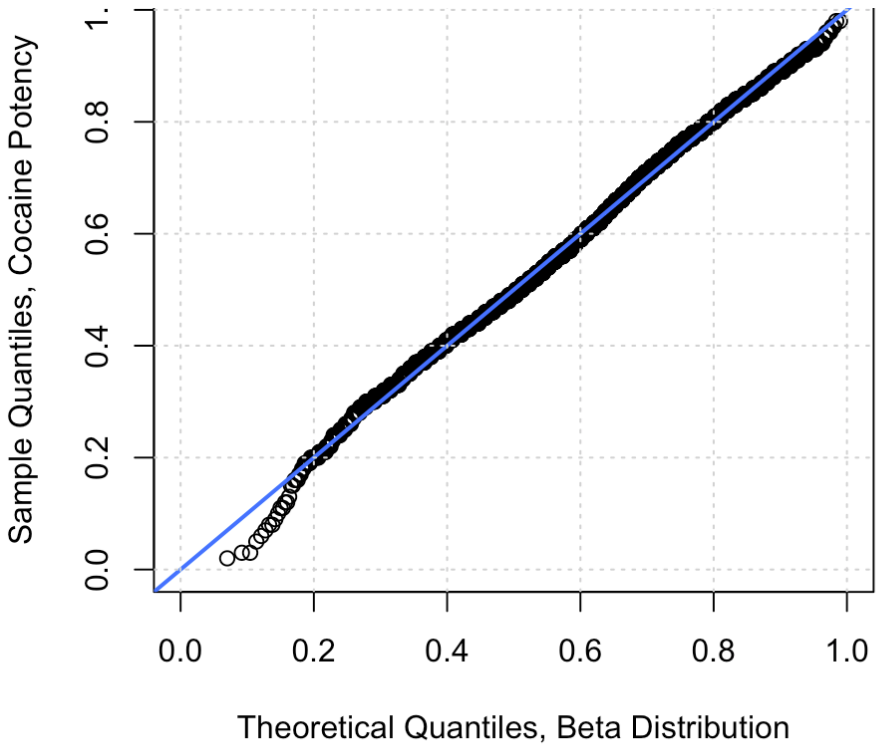
Note that `R` will use `shape1` instead of α and `shape2` instead of β.

```
hist(coke_pot, probability = TRUE,
     main = "Potency of Seized Cocaine", xlab = "Potency  (0 = all filler,
1 = pure cocaine)",
     ylim = c(0, 2.5), xlim = c(0, 1), breaks = 15)
box()
grid()
curve(dbeta(x, shape1 = coke_pot_alpha_hat, shape2 = coke_pot_beta_hat),
      add = TRUE, col = "darkorange", lwd = 2)
```



This looks pretty good, but let's verify with a QQ plot.

```
qqplot(x = qbeta(ppoints(coke_pot), shape1 = coke_pot_alpha_hat, shape2 =
coke_pot_beta_hat),
       y = coke_pot,
       xlim = c(0, 1), ylim = c(0, 1),
       main = "QQ-Plot: Cocaine Potency, Beta Distribution",
       xlab = "Theoretical Quantiles, Beta Distribution",
       ylab = "Sample Quantiles, Cocaine Potency")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```

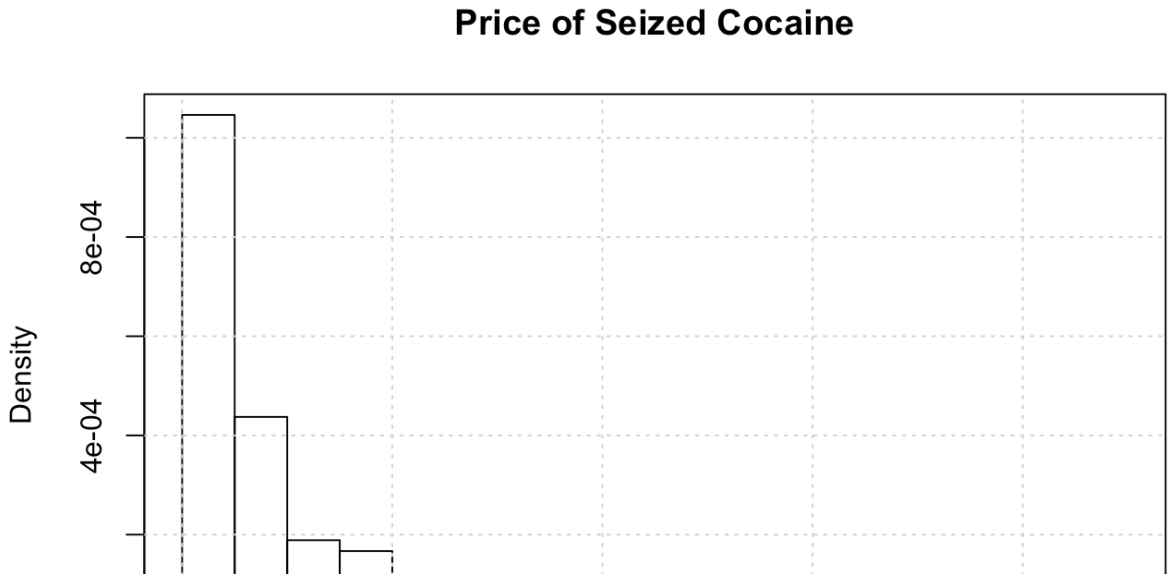


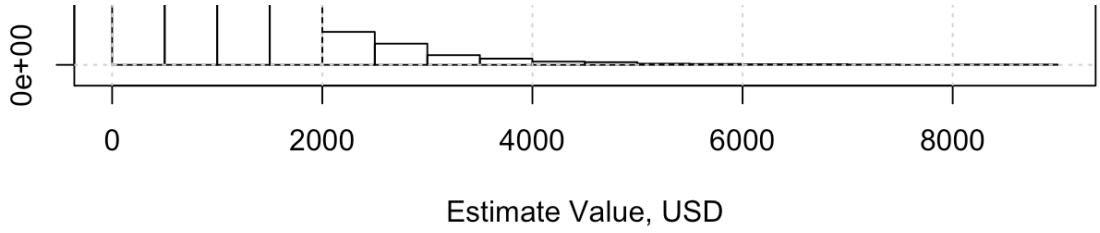
The left tail isn't perfect, but again, this look pretty good.

Cocaine Price, Some Model?

Now let's try to model the price of the cocaine seized.

```
hist(cocaine$price, probability = TRUE,
     main = "Price of Seized Cocaine", xlab = "Estimate Value, USD",
     breaks = 15)
box()
grid()
```





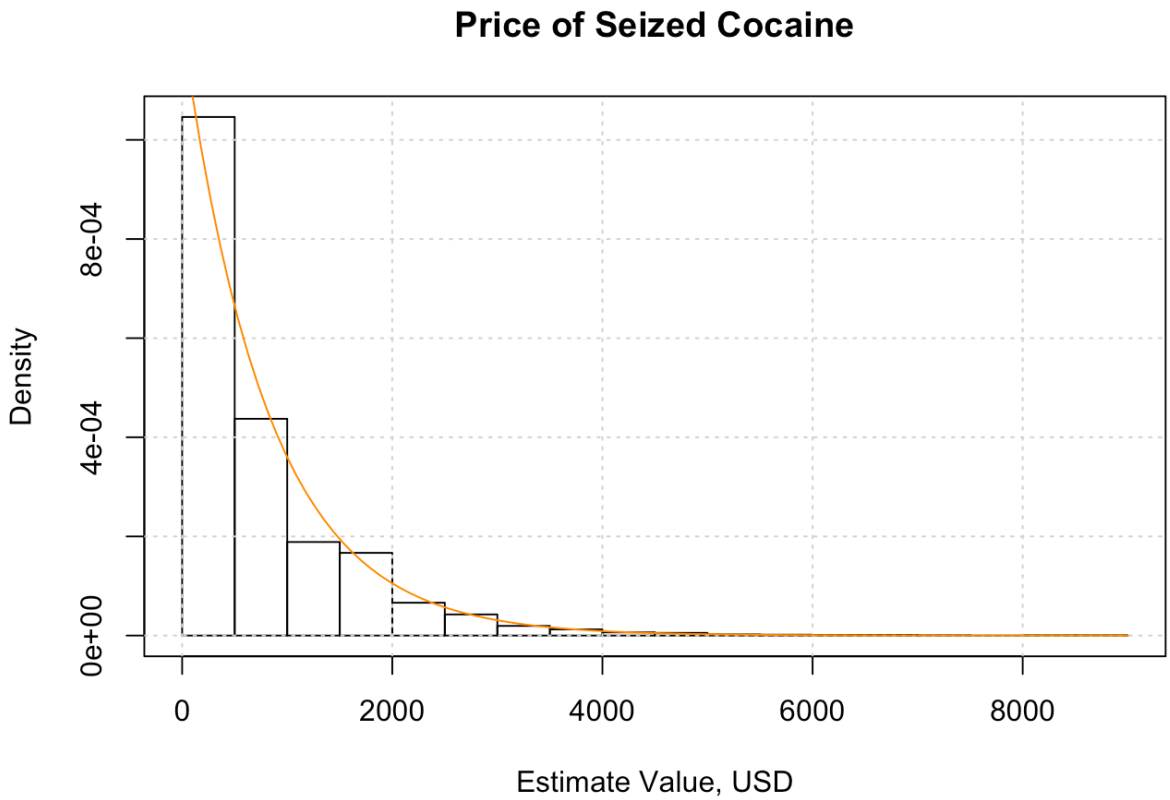
This is a very right-skewed distribution. We know a number of distributions that are valide for positive numbers. Let's try exponential.

Note that `R` uses `rate` in place of $1/\theta$ if the probability density function

$$f(x \mid \theta) = \frac{1}{\theta} e^{-x/\theta}, \quad x > 0, \ \theta > 0$$

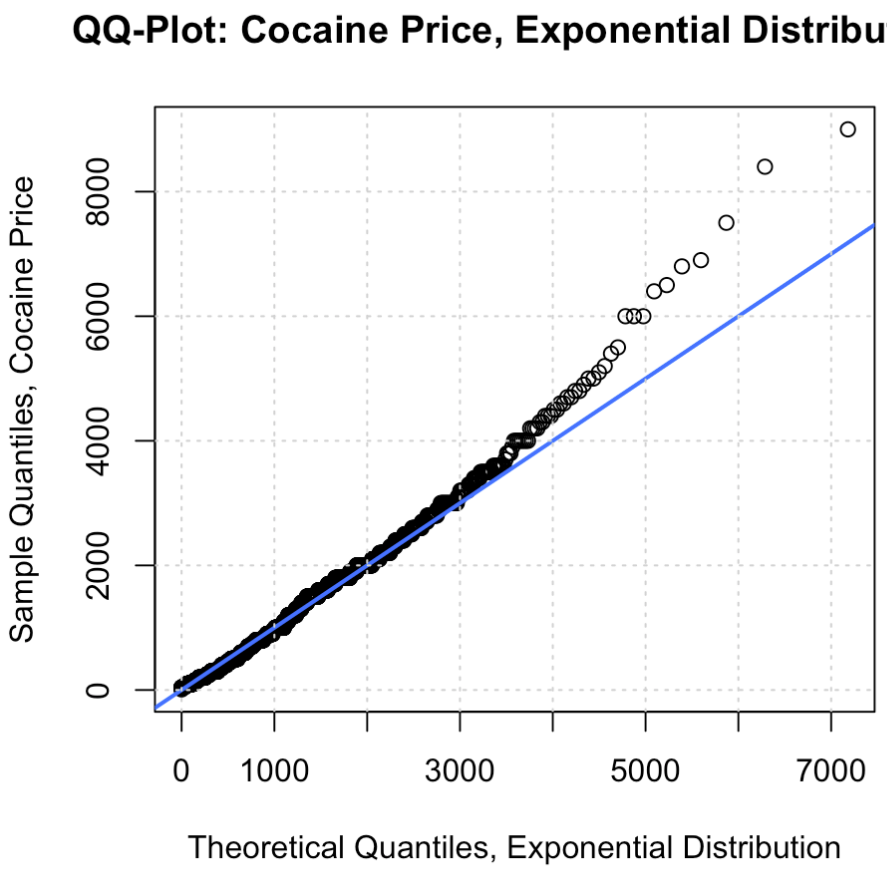
Also note that the MLE for θ is $\bar{X}$.

```
hist(cocaine$price, probability = TRUE,
     main = "Price of Seized Cocaine", xlab = "Estimate Value, USD",
     breaks = 15)
box()
grid()
curve(dexp(x, rate = 1 / mean(cocaine$price)),
      add = TRUE, col = "darkorange")
```



This looks OK, but let's check with a QQ plot.

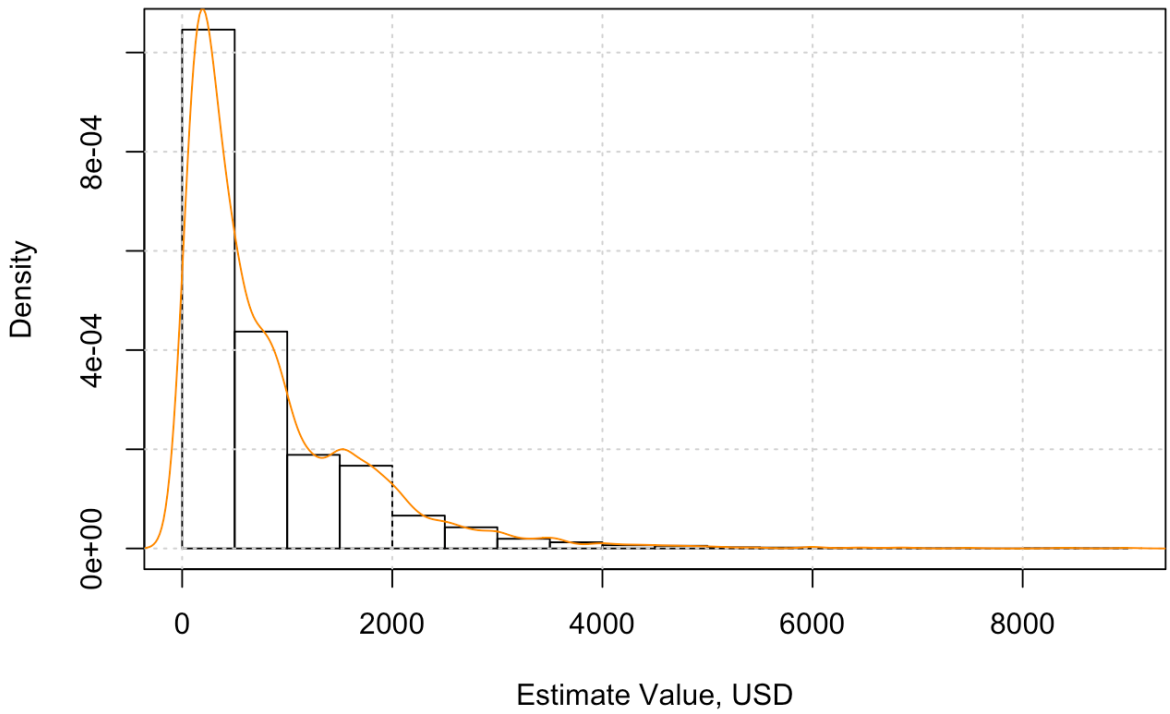
```
qqplot(x = qexp(ppoints(cocaine$price), rate = 1 / mean(cocaine$price)),
       y = cocaine$price,
       main = "QQ-Plot: Cocaine Price, Exponential Distribution",
       xlab = "Theoretical Quantiles, Exponential Distribution",
       ylab = "Sample Quantiles, Cocaine Price")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```



This actually highlights an issue. It seems that this distribution actually fits poorly in the right tail.

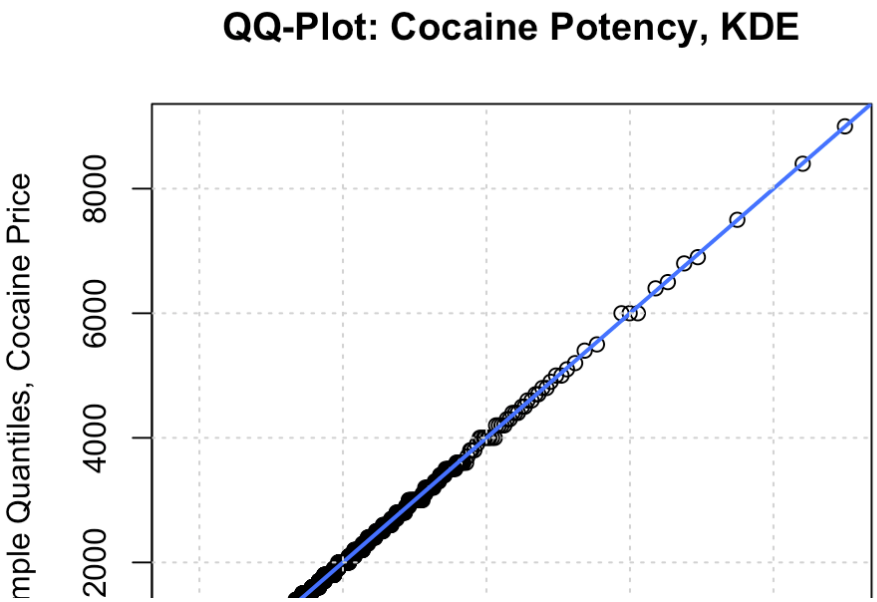
We could try some other distributions, but let's cheat a bit a use something called a kernel density estimate.

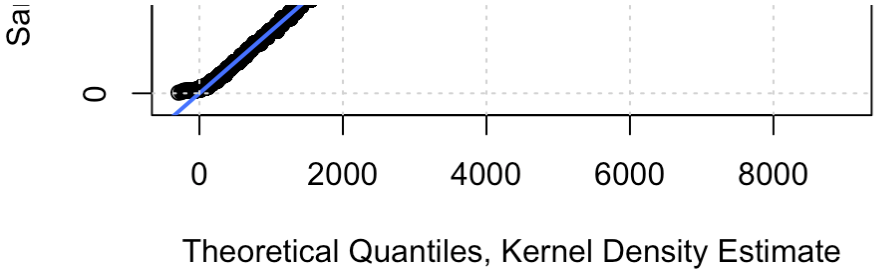
```
hist(cocaine$price, probability = TRUE,
     main = "Price of Seized Cocaine", xlab = "Estimate Value, USD",
     breaks = 15)
box()
grid()
lines(density(cocaine$price), col = "darkorange")
```



Wow, look at that, it fits great, which we can verify with a QQ plot.

```
library(spatstat)
qqplot(x = quantile.density(density(cocaine$price), ppoints(cocaine$pric
e)),
       y = cocaine$price,
       main = "QQ-Plot: Cocaine Potency, KDE",
       xlab = "Theoretical Quantiles, Kernel Density Estimate",
       ylab = "Sample Quantiles, Cocaine Price")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```





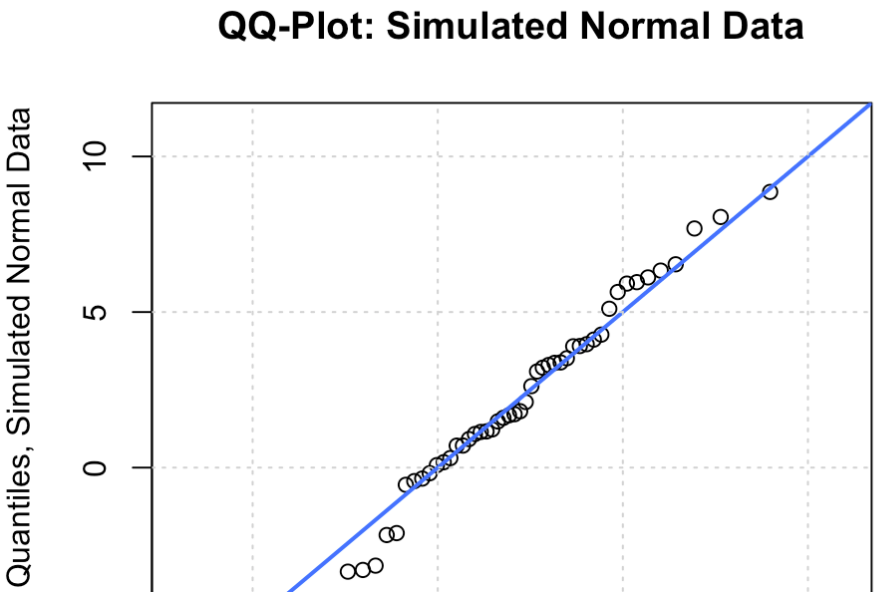
That's a big improvement! Why not just use this all the time. There are actually a few reasons, most of which are hard to explain currently. Once we start doing hypothesis testing, we'll see that the lack of a model parameter here would make things difficult. (Also, there is technically a tuning (hyper) parameter being used here, we just a using the default value.)

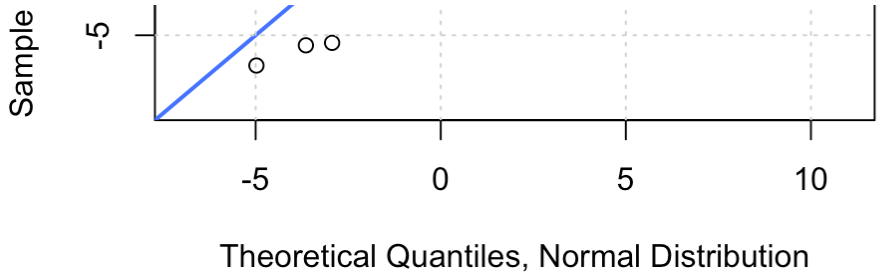
Kernel density estimating is an example of a nonparametric method, again notice that we didn't estimate a parameter. We'll discuss some related methods later in the course.

QQ Plots

To get a better sense of how QQ plots work, we provide four examples.

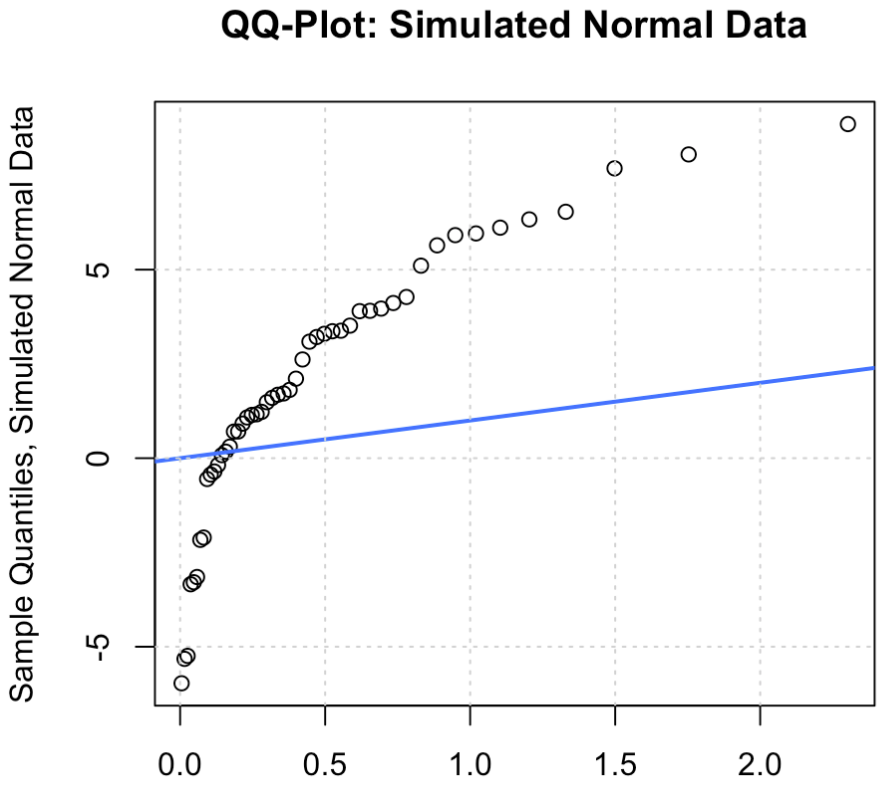
```
set.seed(42)
sim_norm_data = rnorm(n = 50, mean = 2, sd = 3)
qqplot(x = qnorm(ppoints(sim_norm_data), mean = 2, sd = 3),
       y = sim_norm_data,
       xlim = c(-7, 11), ylim = c(-7, 11),
       main = "QQ-Plot: Simulated Normal Data",
       xlab = "Theoretical Quantiles, Normal Distribution",
       ylab = "Sample Quantiles, Simulated Normal Data")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```

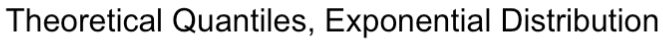




- **Example 1**
  - Data Simulated From: Normal Distribution (`mean = 2, sd = 3`)
  - Model Fit: Normal Distribution (`mean = 2, sd = 3`)
  - Comments: Here we are fitting the exact correct distribution, so this QQ plot is great. If you rerun this chunk repeatedly, but without the code for setting a seed for the randomization (`set.seed(42)`) you can see a number of good QQ plots.

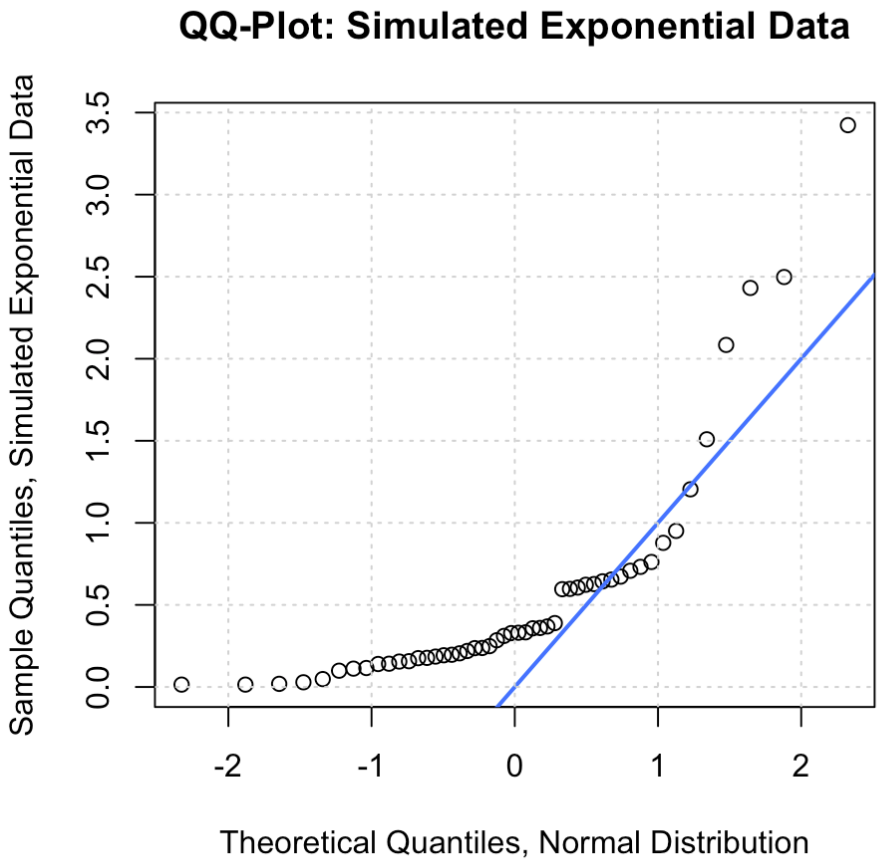
```
set.seed(42)
sim_norm_data = rnorm(n = 50, mean = 2, sd = 3)
qqplot(x = qexp(ppoints(sim_norm_data), rate = 2),
       y = sim_norm_data,
       main = "QQ-Plot: Simulated Normal Data",
       xlab = "Theoretical Quantiles, Exponential Distribution",
       ylab = "Sample Quantiles, Simulated Normal Data")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```





- **Example 2**
  - Data Simulated From: Normal Distribution (`mean = 2, sd = 3`)
  - Model Fit: Exponential Distribution (`rate = 2`)
  - Comments: Here we are fitting an exponential distribution to a normal distribution. The QQ plot is terrible.

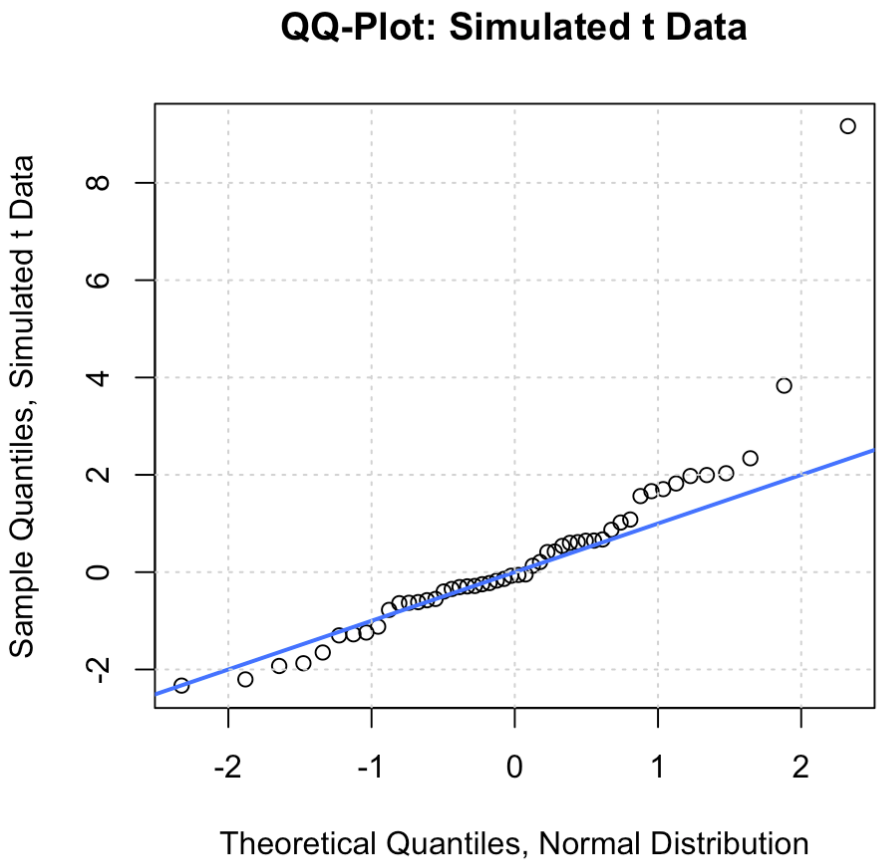
```
set.seed(42)
sim_exp_data = rexp(n = 50, rate = 2)
qqplot(x = qnorm(ppoints(sim_exp_data), mean = 0, sd = 1),
       y = sim_exp_data,
       main = "QQ-Plot: Simulated Exponential Data",
       xlab = "Theoretical Quantiles, Normal Distribution",
       ylab = "Sample Quantiles, Simulated Exponential Data")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```



- **Example 3**
  - Data Simulated From: Exponential Distribution (`rate = 2`)
  - Model Fit: Normal Distribution (`mean = 0, sd = 1`)
  - Comments: Here we are fitting a normal distribution to an exponential

distribution. The QQ plot is terrible.

```
set.seed(42)
sim_t_data = rt(n = 50, df = 3)
qqplot(x = qnorm(ppoints(sim_t_data), mean = 0, sd = 1),
       y = sim_t_data,
       main = "QQ-Plot: Simulated t Data",
       xlab = "Theoretical Quantiles, Normal Distribution",
       ylab = "Sample Quantiles, Simulated t Data")
abline(a = 0, b = 1, col = "dodgerblue", lwd = 2)
grid()
```



- **Example 4**
  - Data Simulated From: t Distribution (`rate = 2`)
  - Model Fit: Normal Distribution (`df = 3`)
  - Comments: Here we are fitting a normal distribution to a t distribution. The QQ plot is not great. You should especially notice the large deviation from the line at the right extreme. However, if you rerun this code repeatedly without `set.seed(42)` , you'll notice that sometimes the QQ plot looks OK. This is because these two distributions are very similar, except for the tails, which sometimes might not be captured in the sampled data.

Process Recap

- **Look** at the data.
  - What are the rows (observations) in the dataset?
  - What are the columns (variables) in the dataset? What are their types?
  - Is the variable of interest best modeled as continuous or discrete?
  - What is the range of possible values of the variable according to the data?
  - What is the range of possible values of the variable that *could* appear in the data?
  - Plot the data with a histogram (continuous data) or barplot (discrete data).
- **Assume** a probability distribution.
  - Is there a distribution that accepts the same input values as the range of your data?
  - Is there a distribution that has the potential to match the shape of your data?
- **Estimate** the model parameters, that is, the parameters of the chosen distribution.
  - Use maximum likelihood if possible, otherwise, use method of moments.
- **Check** how well the estimated model fits using a QQ-Plot.
  - Consider a different model if your current model does not fit well.