Probability Distributions in R

Luke Daniels

2/21/2018

- The goal of this exercise was to use a data set and demonstrate uniform, normal, gamma, beta, and exponential distribution fits. The data was taken from an economic study. The observed car fatality rates in each U.S State.
- This is an homework assignment performed for Bio381 instructed by Dr. Gotelli, a graduate level biological computation class

Background

- This exercise will look at continuous distributions

Gamma Range: [0,infinity] Parameters: shape, scale Interpretation: mean=shape*scale , variance=shape*scale2 ; generates a variety of shapes (including normal and exponential) for positive continuous variables

With this distribution the two parameters are affecting the outcome.

Beta this model is good for large samples but not for small samples. We want to take into account the uncertainty in our analysis.

Range: [0,1] (can be rescaled to any range by simple multiplication and addition) Parameters: shape1, shape2 Interpretation: if shape1 and shape 2 are integers, interpret as a coin toss, with shape1 = # of successes + 1, shape2 = # of failures + 1. Gives distribution of value of p, estimated from data, which can range from exponential through uniform through normal (but all are bounded). Setting shape1 and shape2 <1 yields u-shaped distributions.

Loading the Data- Using Car Crash Fatality Data

The File is in dta coming from STATA so the "haven" package is necessary to convert. Ggplot2 is a graphical plot that will allow us to overlay the distributions onto the histograms.

```
library(MASS)
library(ggplot2)
library(haven)
driving <- read_dta("driving_2004-2.dta")
```

Plotting the Initial Histogram

The data set initially consisted of roughly 20 columns. For simplicity I selected only one column of the data and assigned it to a new vector. I renamed it "totalfatal" but I don't use this name much in the code. I plot the initial histogram. I have assigned simple parameters for the graphics such as color ect.

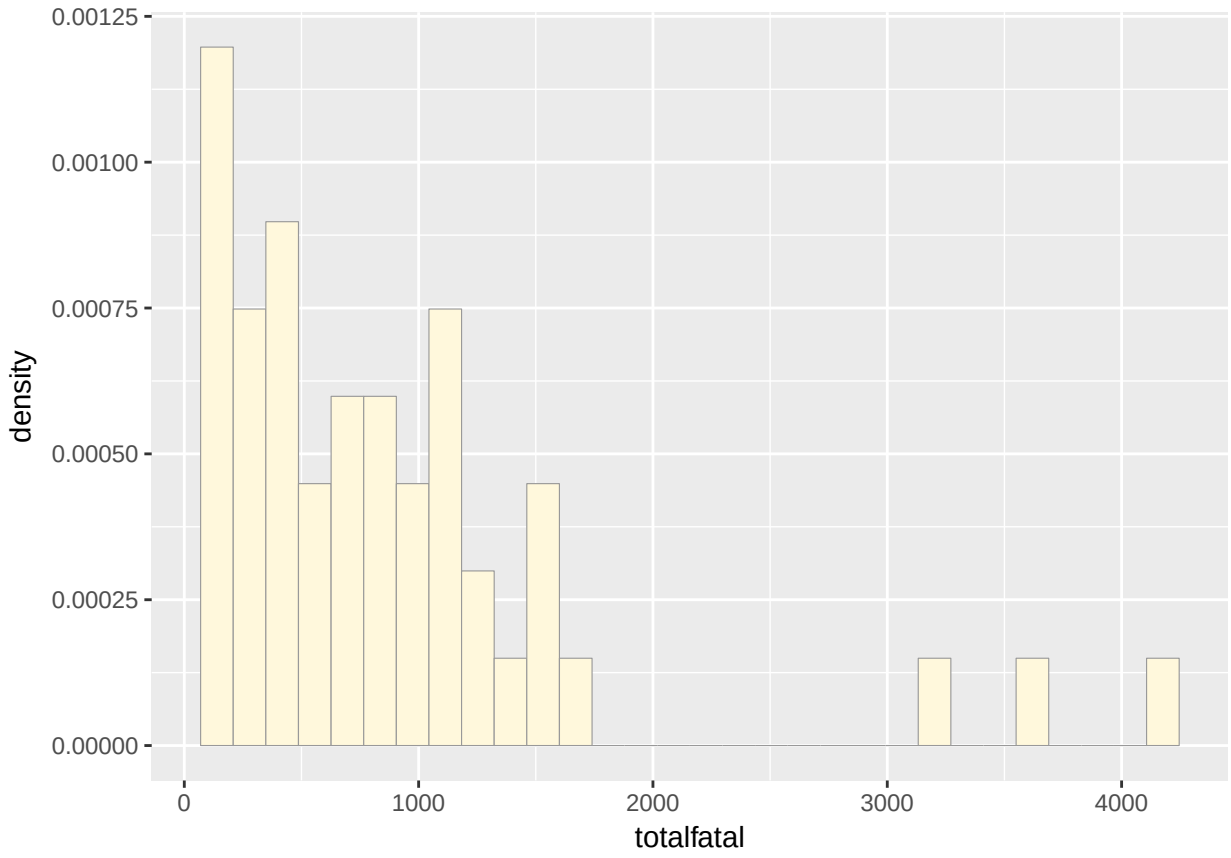
```
driving2 <- data.frame(driving$totfat)
                       names(driving2) <- c("totalfatal")

mean(driving2)
```

```
## Warning in mean.default(driving2): argument is not numeric or logical:
## returning NA
```

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

```
p1 <- ggplot(data=driving2, aes(x=totalfatal, y=..density..)) +
  geom_histogram(color="grey60",fill="cornsilk",size=0.2)
print(p1)
```

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

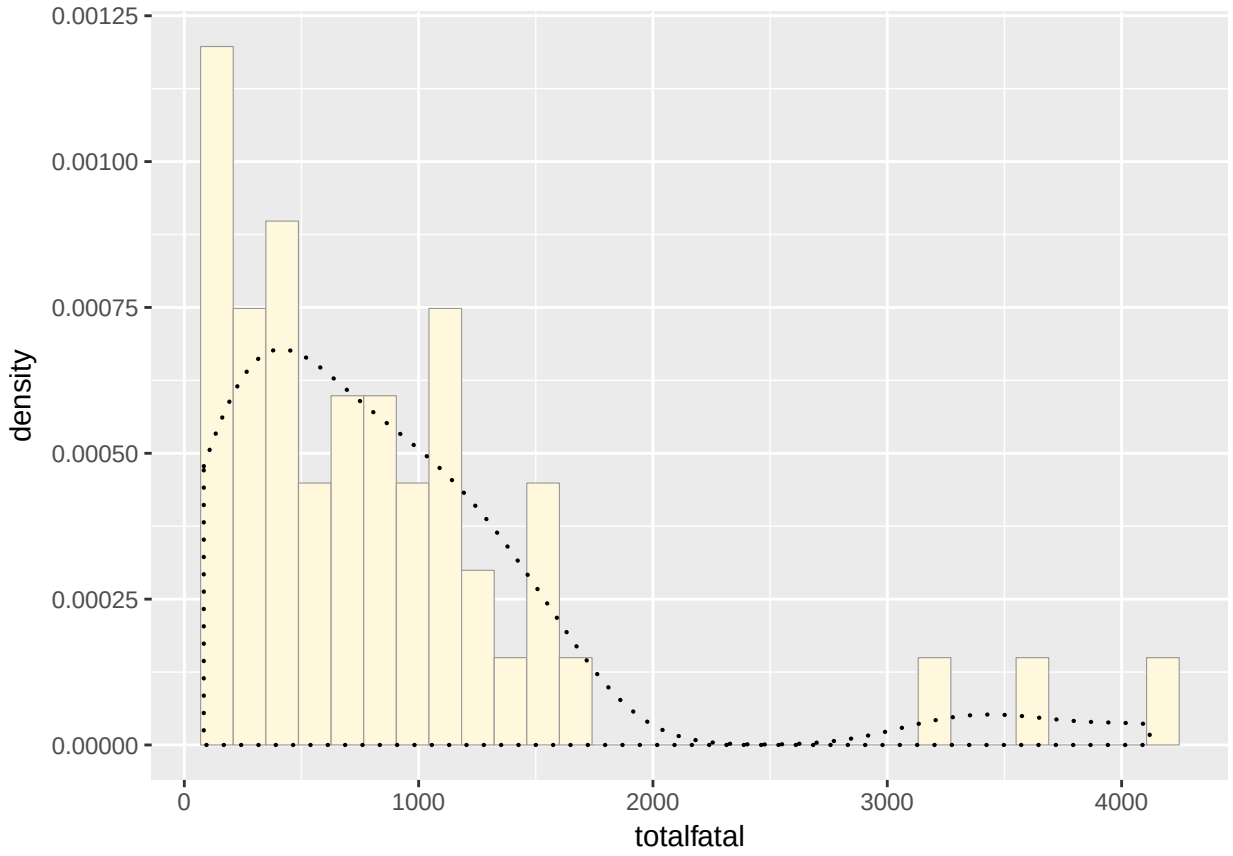


Adding the Empirical Density Curve

This data was randomly selected so the distribution is odd. Here I assigned a density distribution.

```
p1 <-  p1 +  geom_density(linetype="dotted",size=0.75)
print(p1)
```

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



Get Maximum Likelihood Parameters for Normal

I wanted to understand the parameters of the distribution. This will print mean and sd. These parameters will be used later in the code when generating a random set of data to match the parameters.

```
normPars <- fitdistr(driving$totfat,"normal")
print(normPars)
```

```
##      mean         sd
##   882.29167   841.90899
##  (121.51910) ( 85.92698)
```

```
normPars$estimate["mean"]
```

```
##     mean
## 882.2917
```

```
str(normPars)
```

```
## List of 5
##  $ estimate: Named num [1:2] 882 842
##   ..- attr(*, "names")= chr [1:2] "mean" "sd"
##  $ sd      : Named num [1:2] 121.5 85.9
##   ..- attr(*, "names")= chr [1:2] "mean" "sd"
##  $ vcov    : num [1:2, 1:2] 14767 0 0 7383
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "mean" "sd"
```

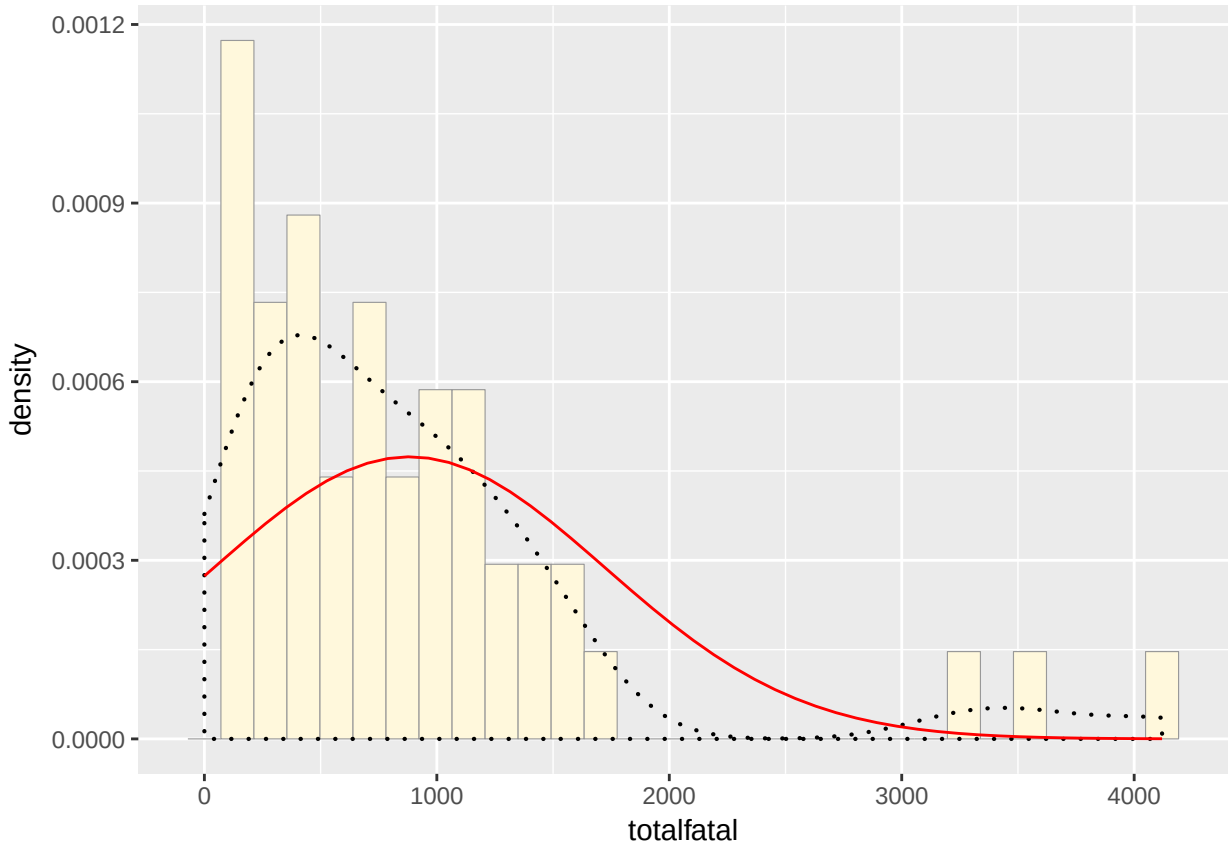
```
##   .. ..$ : chr [1:2] "mean" "sd"
##  $ n       : int 48
##  $ loglik  : num -391
##  - attr(*, "class")= chr "fitdistr"
```

Normal Distribution

The first distribution is the most common. Here I assign the parameters above to the variables meanML and sdML. The stat_function is a command in ggplot2 that allows us to overlay distribution lines onto histograms. The English language is flexible in R – for example look at "color" spelt "colour"

```
meanML <- normPars$estimate["mean"]
sdML <- normPars$estimate["sd"]

xval <- seq(0,max(driving$totfat),len=length(driving$totfat))

stat <- stat_function(aes(x = xval, y = ..y..), fun = dnorm, colour="red", n = length(driving$totfat), a
print(p1 + stat)
```

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

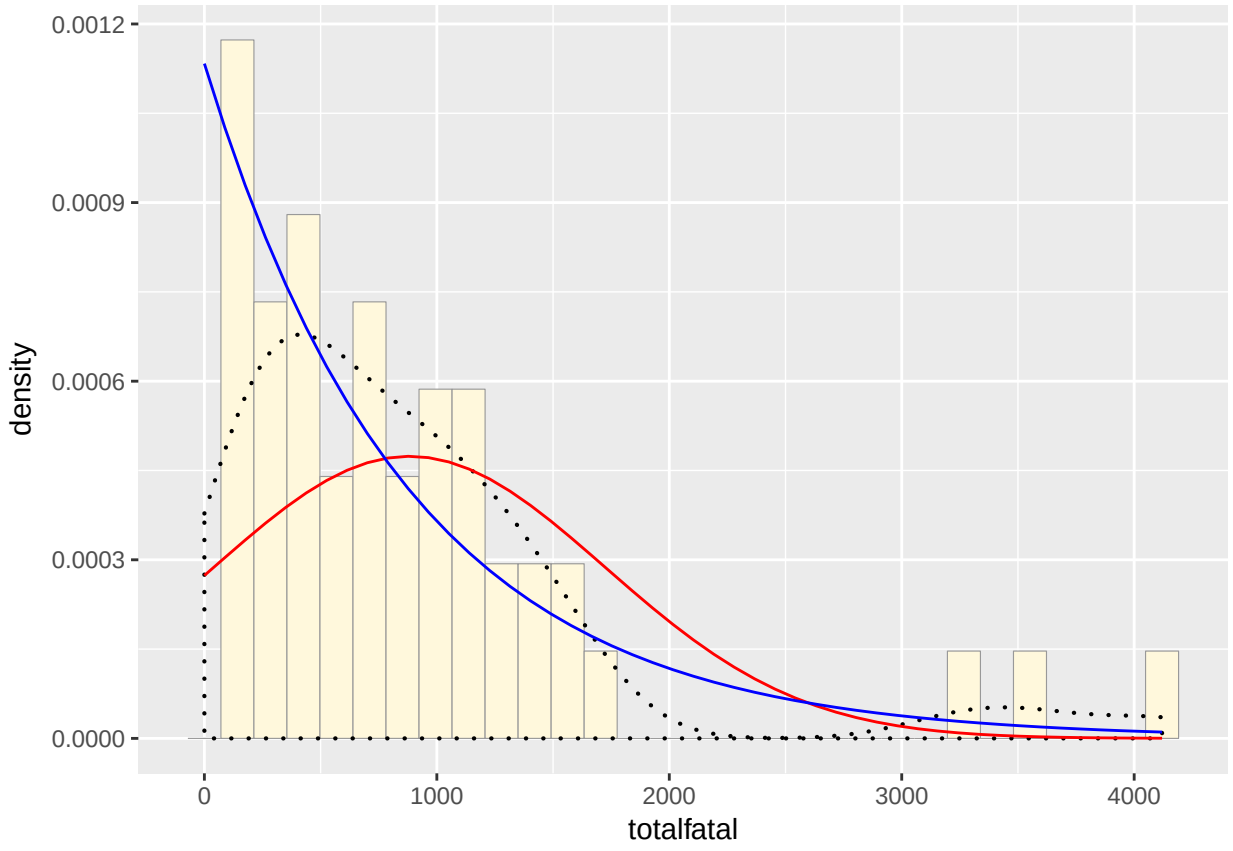


Plot Exponential Probability Distribution

We are setting new parameters for an exponential distribution. Assign it stat2 so that it does not replace the distribution line stat1

```
expoPars <- fitdistr(driving$totfat,"exponential")
rateML <- expoPars$estimate["rate"]

stat2 <- stat_function(aes(x = xval, y = ..y..), fun = dexp, colour="blue", n = length(driving$totfat),
print(p1 + stat + stat2)
```

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

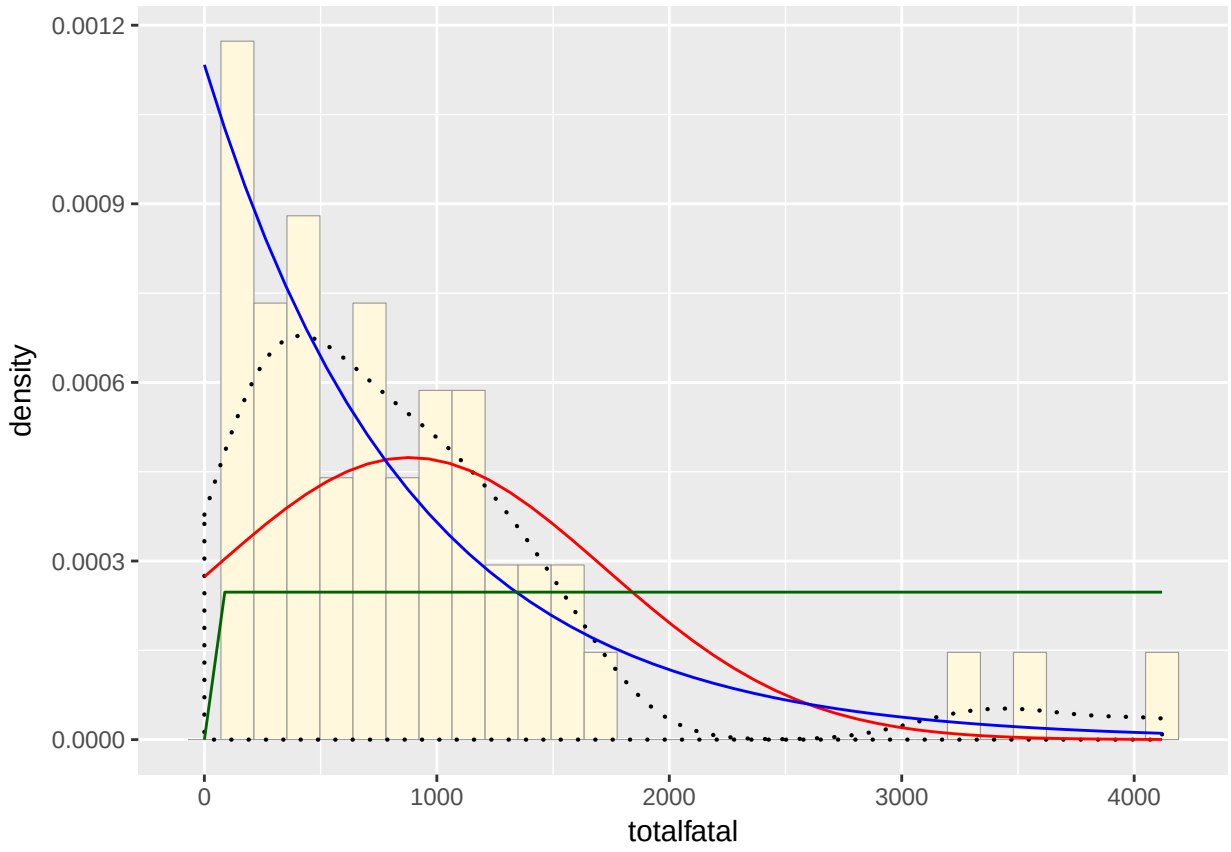


Uniform Probability Distribution

Similar logic. Notice how were are changing the line colors and compiling all lines on top of one another.

```
stat3 <- stat_function(aes(x = xval, y = ..y..), fun = dunif, colour="darkgreen", n = length(driving$to
p1 + stat + stat2 + stat3
```

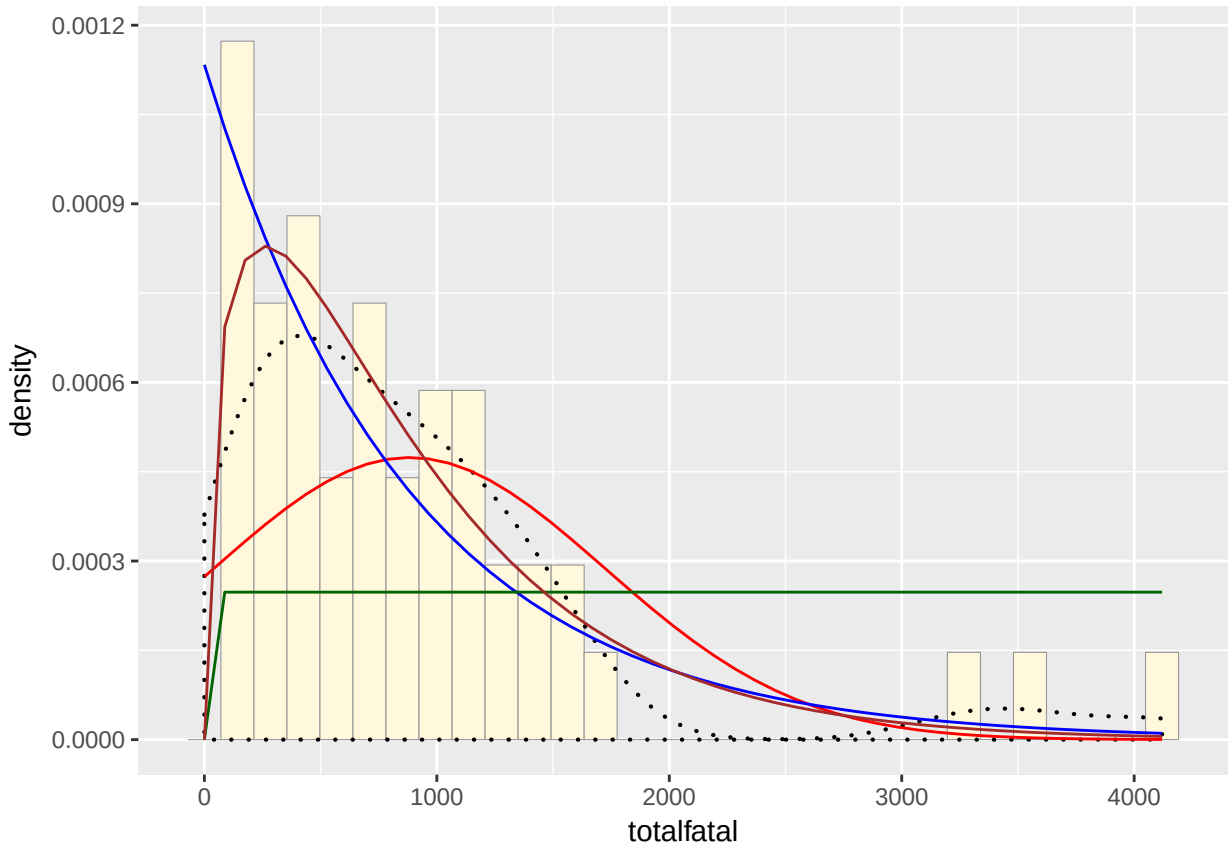
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Gamma Distribution

```
gammaPars <- fitdistr(driving$totfat,"gamma")
shapeML <- gammaPars$estimate["shape"]
rateML <- gammaPars$estimate["rate"]

stat4 <- stat_function(aes(x = xval, y = ..y..), fun = dgamma, colour="brown", n = length(driving$totfa
p1 + stat + stat2 + stat3 + stat4
```

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



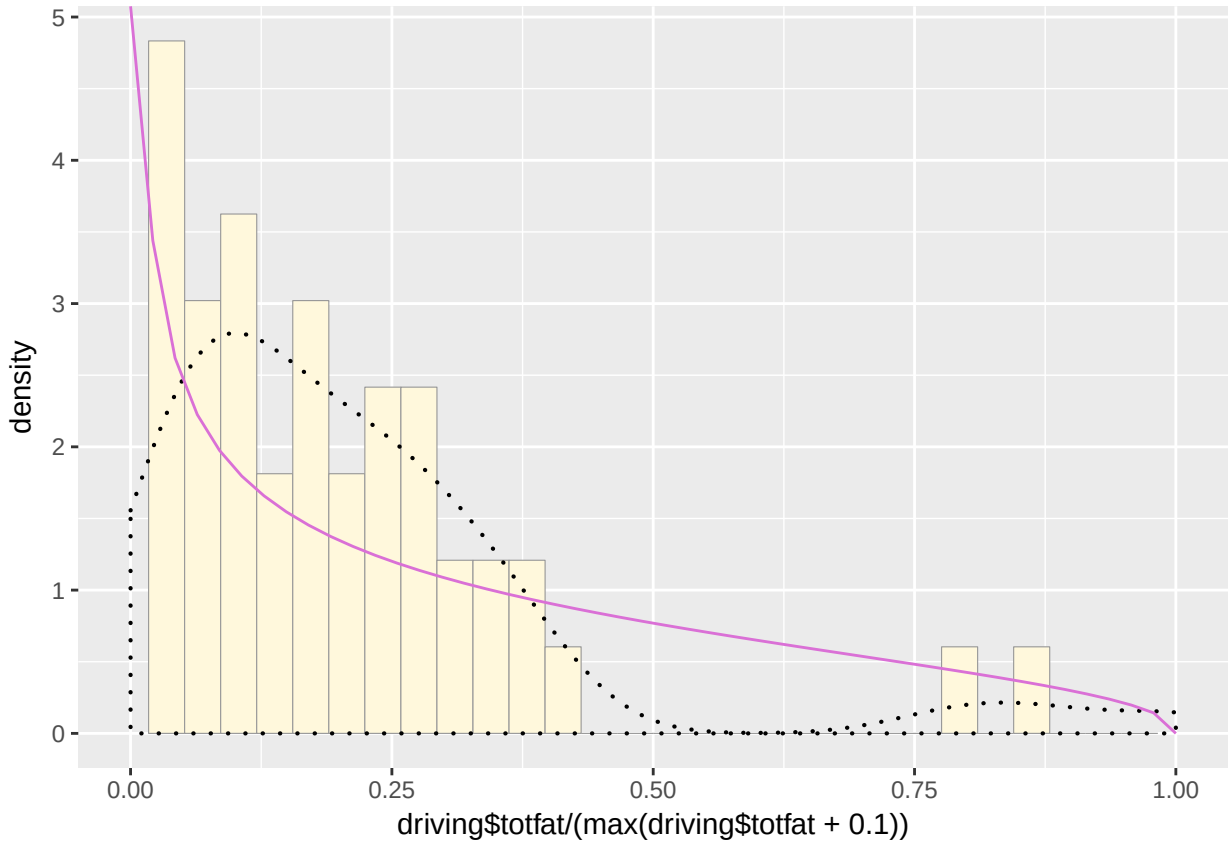
Beta Distribution

```
pSpecial <- ggplot(data=driving2, aes(x=driving$totfat/(max(driving$totfat + 0.1)), y=..density..)) +
  geom_histogram(color="grey60",fill="cornsilk",size=0.2) +
  xlim(c(0,1)) +
  geom_density(size=0.75,linetype="dotted")

betaPars <- fitdistr(x=driving$totfat/max(driving$totfat + 0.1),start=list(shape1=1,shape2=2),"beta")
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
shape1ML <- betaPars$estimate["shape1"]
shape2ML <- betaPars$estimate["shape2"]

statSpecial <- stat_function(aes(x = xval, y = ..y..), fun = dbeta, colour="orchid", n = length(driving$
pSpecial + statSpecial
```

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



Compare Log Likelihoods

We want to find which distribution best fits the data. The higher the number the better the fit.

```
normPars$loglik
```

```
## [1] -391.4213
```

```
gammaPars$loglik
```

```
## [1] -371.9341
```

```
expoPars$loglik
```

```
## [1] -373.5611
```

Gamma is the Best Fit

Create Simulated Data

Here we create simulated based on the parameters of the data above.

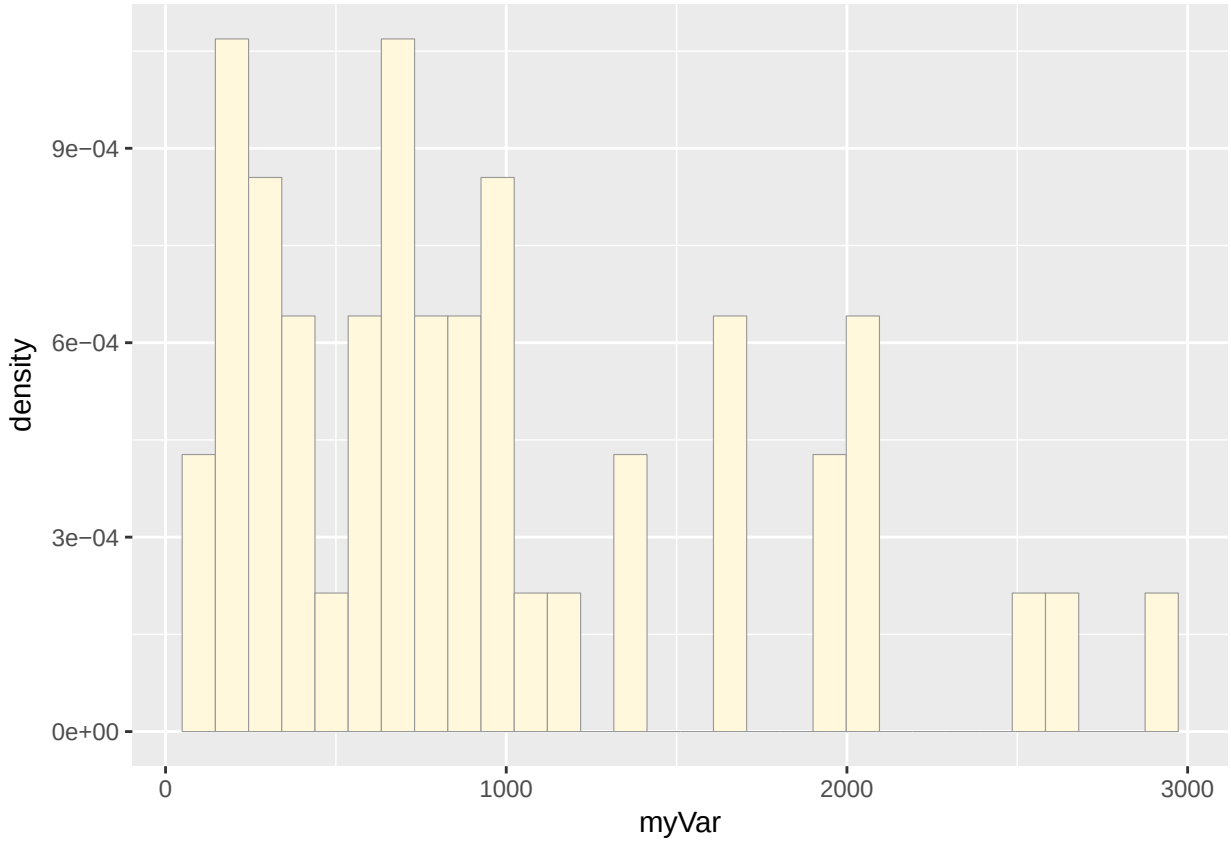
```
z <- rgamma(n= 48, shape=shapeML, rate=rateML)
z <- data.frame(1:48,z)

names(z) <- list("ID","myVar")
```

```
z <- z[z$myVar>0,]
str(z)
```

```
## 'data.frame':    48 obs. of  2 variables:
##  $ ID   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ myVar: num  1644 2901 1197 104 728 ...
```

```
p5 <- ggplot(data=z, aes(x=myVar, y=..density..)) +
  geom_histogram(color="grey60",fill="cornsilk",size=0.2)
print(p5)
```

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




```
normPars <- fitdistr(z$myVar,"normal")
print(normPars)
```

```
##      mean        sd
##   956.8606   718.2452
##  (103.6698) ( 73.3056)
```

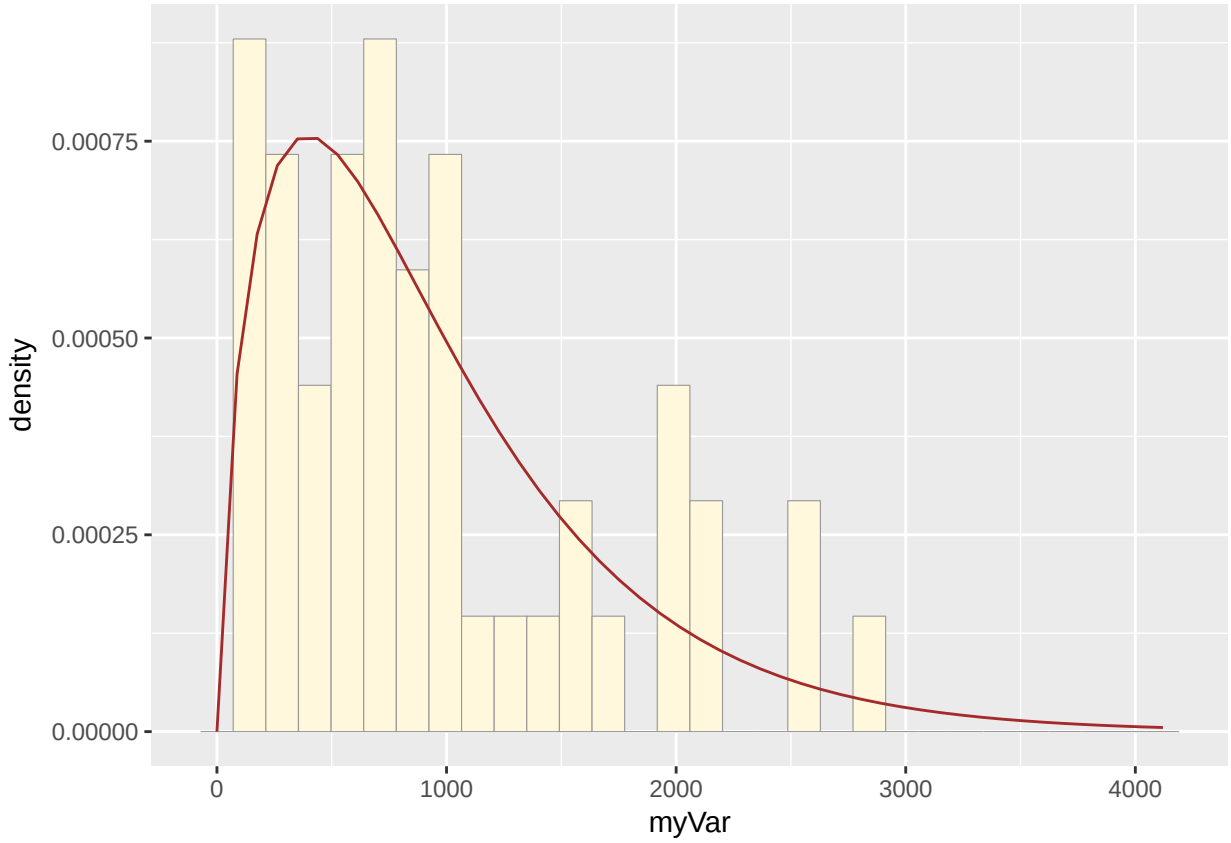
```
str(normPars)
```

```
## List of 5
##  $ estimate: Named num [1:2] 957 718
##   ..- attr(*, "names")= chr [1:2] "mean" "sd"
##  $ sd      : Named num [1:2] 103.7 73.3
##   ..- attr(*, "names")= chr [1:2] "mean" "sd"
##  $ vcov    : num [1:2, 1:2] 10747 0 0 5374
##   ..- attr(*, "dimnames")=List of 2
```

```
##   .. ..$ : chr [1:2] "mean" "sd"
##   .. ..$ : chr [1:2] "mean" "sd"
##  $ n       : int 48
##  $ loglik  : num -384
##  - attr(*, "class")= chr "fitdistr"
```

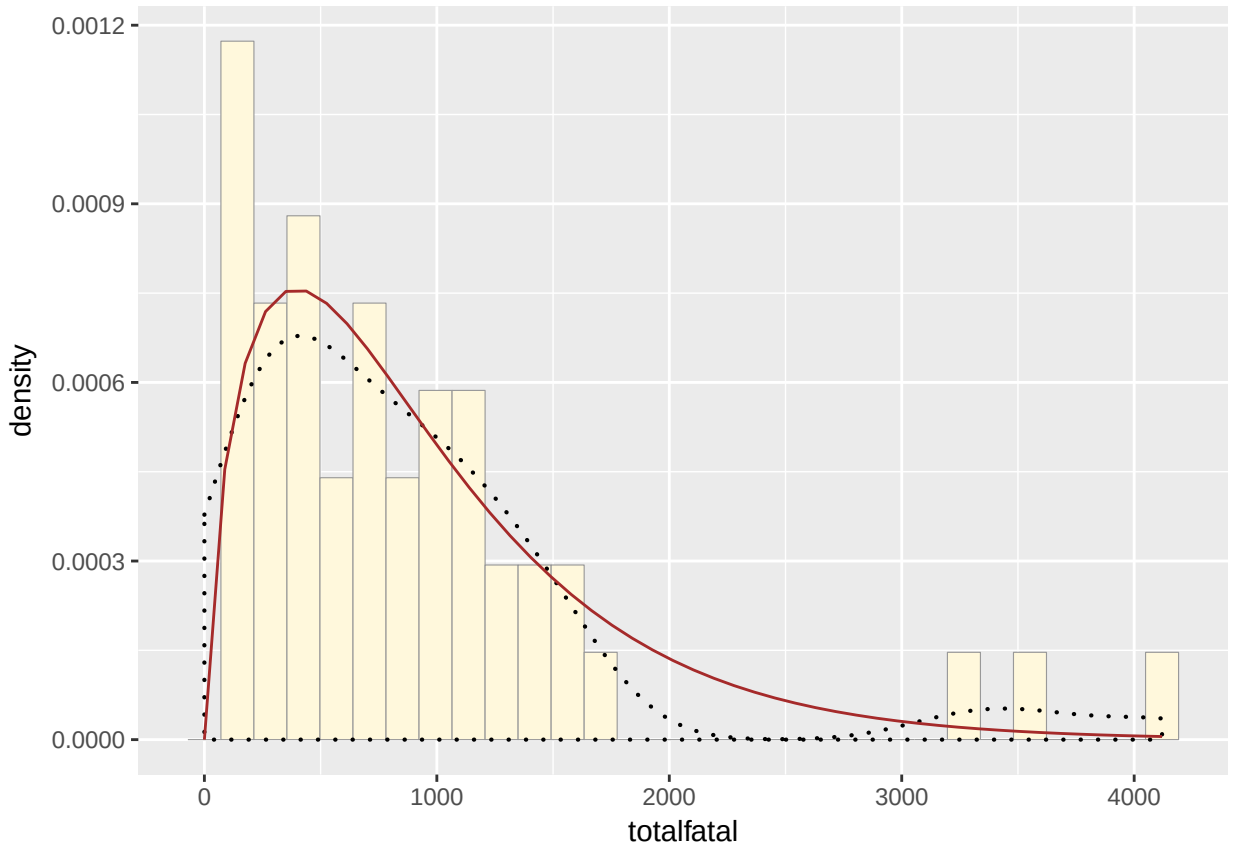
```
gammaPars <- fitdistr(z$myVar,"gamma")
shapeML <- gammaPars$estimate["shape"]
rateML <- gammaPars$estimate["rate"]

stat4 <- stat_function(aes(x = xval, y = ..y..), fun = dgamma, colour="brown", n = length(z$myVar), arg
p5 + stat4
```

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




```
p1 + stat4
```

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



We can see that the simulated data given the parameters doesn't exactly match our data set. However, it is fairly good for simulated data!