Homework #6

Nicholas J. Gotelli

February 21, 2018

This exercise teaches you how to compare a histogram of continuous (or integer) data to the probability density functions for different statistical distributions.

1. Set up a new `.Rmd` file for this exercise. Copy and paste the code below into different code chunks, and then read the text and run the code chunks one at a time to see what they do. You probably won't understand everything in the code, but this is a good start for seeing some realistic uses of `ggplot`. We will cover most of these details in the next few weeks.

2. Once the code is in and runs, try reading in your own `.csv` file into a data frame with this code chunk:

```
z <- read.table("MyDataFile.csv",header=TRUE,sep=",", stringsAsFactors=FALSE)
str(z)
summary(z)
```

Lauren will be able to help you if you are having trouble reading in the data.

3. Once your data are in, go ahead and comment out the "fake data" that are simulated in the chunk below. At that point, if you compile the entire file, it should run all of the code on your own data. Be sure to add comments to the code and commentary to the `.Rmd` file so that you can go back to it later and understand and use the code in your work.

4. Take a look at the second-to-last graph which shows the histogram of your data and 4 probability density curves (normal, uniform, exponential, gamma) that are fit to the data. The `beta` distribution in the final graph is somewhat special. It often fits the data pretty well, but that is because we have assumed the largest data point is the true upper bound, and everything is scaled to that. The fit of the uniform distribution also fixes the upper bound. The other curves (normal, exponential, and gamma) are more realistic because they do not have an upper bound. For most data sets, the gamma will probably fit best, but if you data set is small, it may be very hard to see much of a difference between the curves.

5. Using the best-fitting distribution, go back to the code and get the maximum likelihood parameters. Use those to simulate a new data set, with the same length as your original vector, and plot that in a histogram and add the probability density curve. Right below that, generate a fresh histogram plot of the original data, and also include the probability density curve.

How do the two histogram profiles compare? Do you think the model is doing a good job of simulating realistic data that match your original measurements? Why or why not?

If you have entered a large data frame with many columns, try running all of the code on a different variable to see how the simulation performs.

Once we get a little bit more R coding under our belts, we will return to the problem of simulating data and use some of this code again.

Open libraries

```
library(ggplot2) # for graphics
library(MASS) # for maximum likelihood estimation
```

Read in data vector

To illustrate, we will generate some fake data here:

```
# quick and dirty, a truncated normal distribution to work on the solution set

z <- rnorm(n=3000,mean=0.2)
z <- data.frame(1:3000,z)
names(z) <- list("ID","myVar")
z <- z[z$myVar>0,]
str(z)
```

```
## 'data.frame':    1713 obs. of  2 variables:
##  $ ID   : int  1 2 5 7 10 11 13 19 21 22 ...
##  $ myVar: num  0.0224 0.4623 0.3519 0.4852 0.4427 ...
```

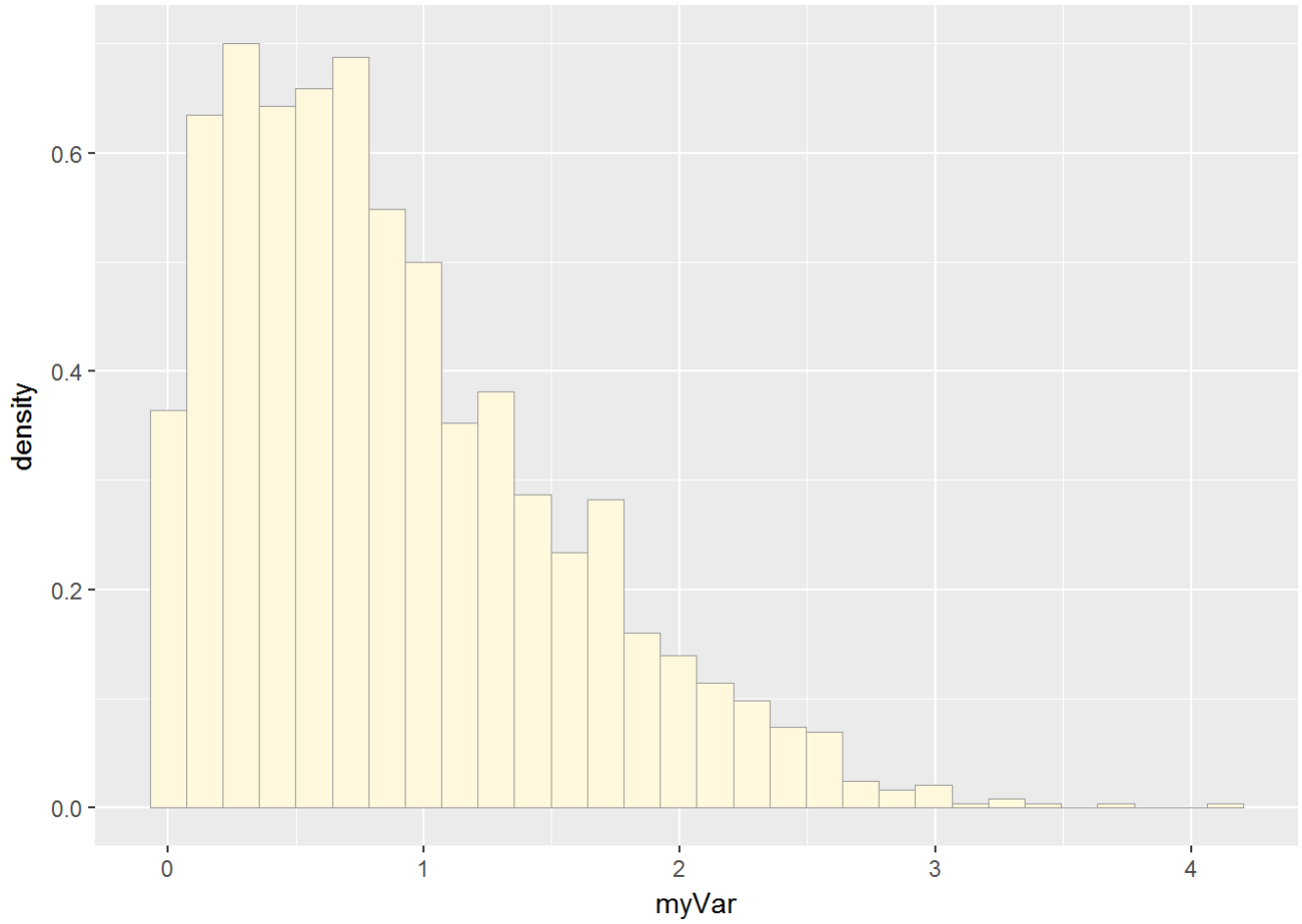
```
summary(z$myVar)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000298 0.371928 0.752248 0.886893 1.284160 4.134398
```

Plot histogram of data

Plot a histogram of the data, using a modification of the code from lecture. Here we are switching from `qplot` to `ggplot` for more graphics options. We are also rescaling the y axis of the histogram from counts to density, so that the area under the histogram equals 1.0.

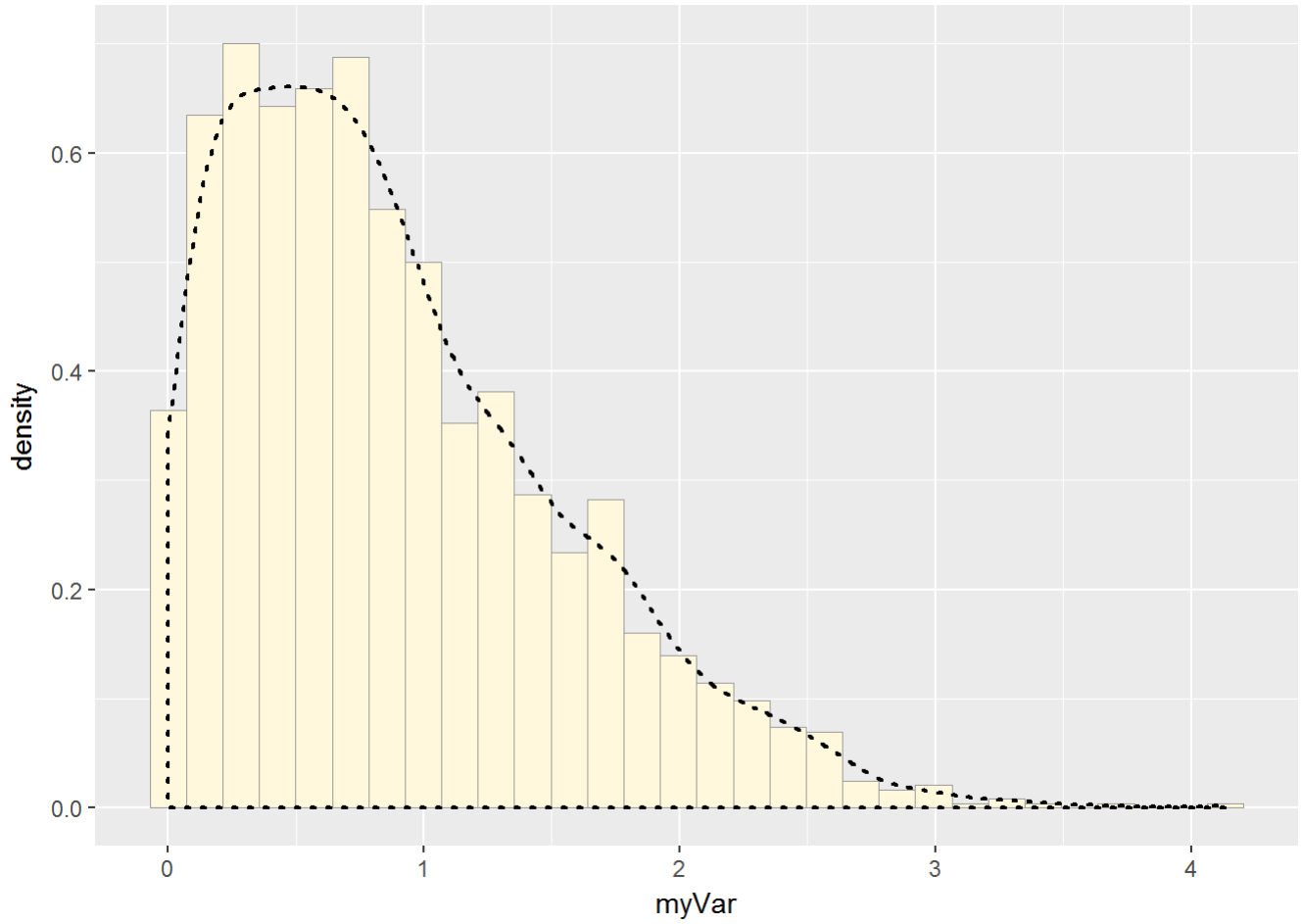
```
p1 <- ggplot(data=z, aes(x=myVar, y=..density..)) +
  geom_histogram(color="grey60",fill="cornsilk",size=0.2)
print(p1)
```



Add empirical density curve

Now modify the code to add in a kernel density plot of the data. This is an empirical curve that is fitted to the data. It does not assume any particular probability distribution, but it smooths out the shape of the histogram:

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



Get maximum likelihood parameters for `normal`

Next, fit a normal distribution to your data and grab the maximum likelihood estimators of the two parameters of the normal, the mean and the variance:

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

```
##       mean          sd
##   0.88689316   0.65666629
##  (0.01586595) (0.01121892)
```

```
str(normPars)
```

```
## List of 5
##  $ estimate: Named num [1:2] 0.887 0.657
##   ..- attr(*, "names")= chr [1:2] "mean" "sd"
##  $ sd      : Named num [1:2] 0.0159 0.0112
##   ..- attr(*, "names")= chr [1:2] "mean" "sd"
##  $ vcov    : num [1:2, 1:2] 0.000252 0 0 0.000126
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "mean" "sd"
##   .. ..$ : chr [1:2] "mean" "sd"
##  $ n       : int 1713
##  $ loglik  : num -1710
##  - attr(*, "class")= chr "fitdistr"
```

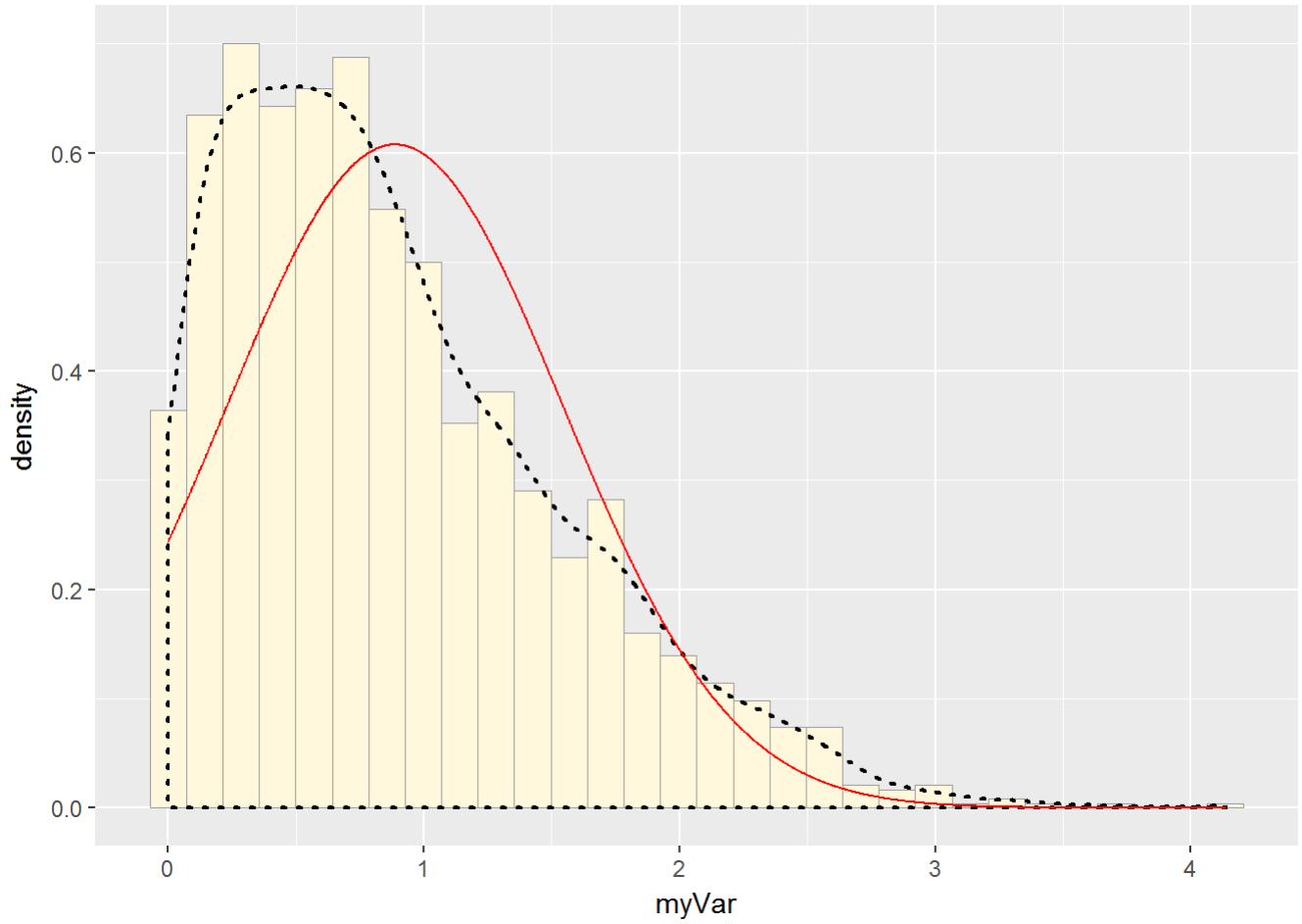
```
normPars$estimate["mean"] # note structure of getting a named attribute
```

```
##      mean
## 0.8868932
```

Plot `normal` probability density

Now let's call the `dnorm` function inside ggplot's `stat_function` to generate the probability density for the normal distribution. Read about `stat_function` in the help system to see how you can use this to add a smooth function to any ggplot. Note that we first get the maximum likelihood parameters for a normal distribution fitted to thse data by calling `fitdistr`. Then we pass those parameters (`meanML` and `sdML` to `stat_function`:

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

xval <- seq(0,max(z$myVar),len=length(z$myVar))

 stat <- stat_function(aes(x = xval, y = ..y..), fun = dnorm, colour="red", n = length(z$myVar), a
rgs = list(mean = meanML, sd = sdML))
 p1 + stat
```

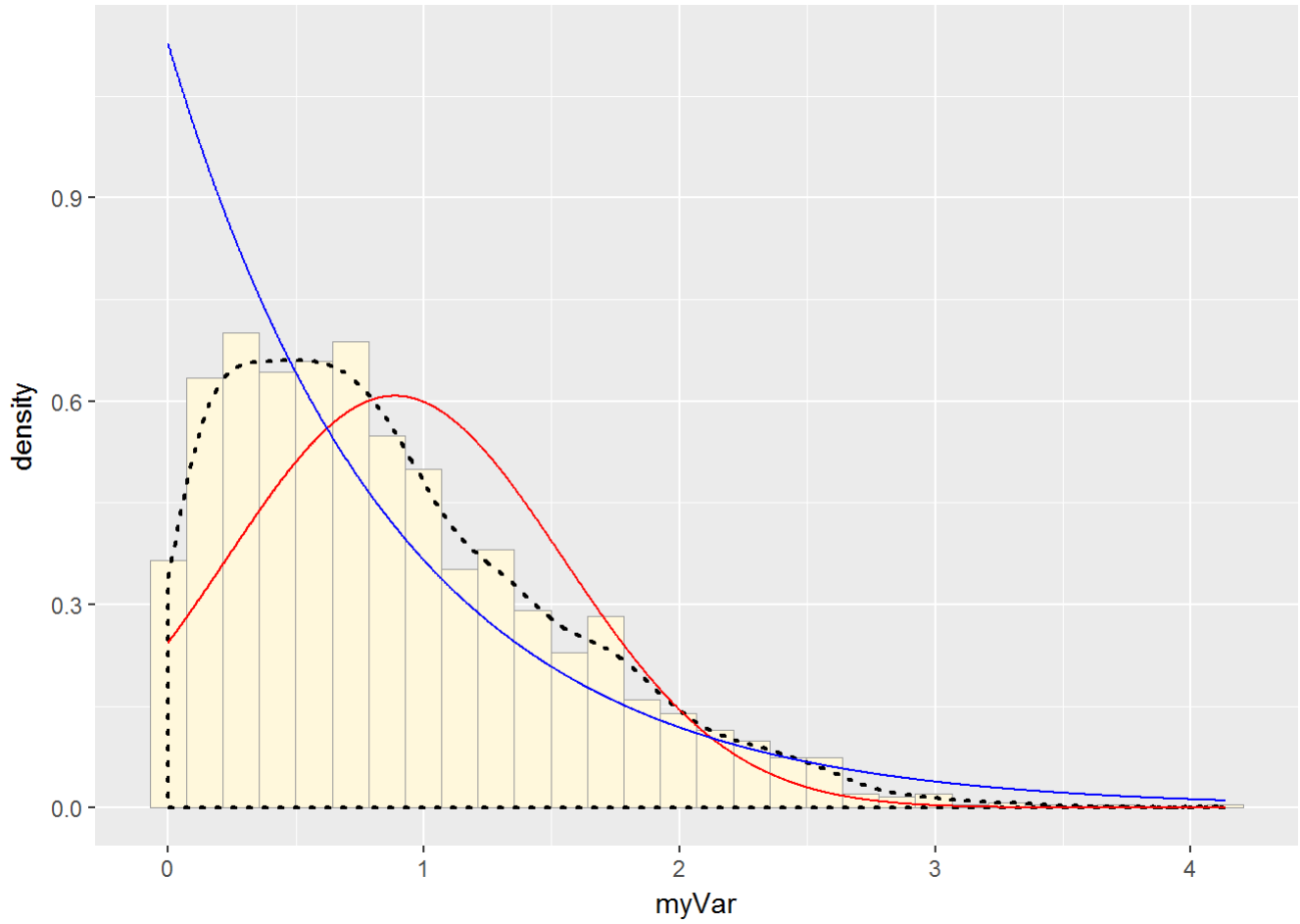


Notice that the best-fitting normal distribution (red curve) for these data actually has a biased mean. That is because the data set has no negative values, so the normal distribution (which is symmetric) is not working well.

Plot `exponential` probability density

Now let's use the same template and add in the curve for the exponential:

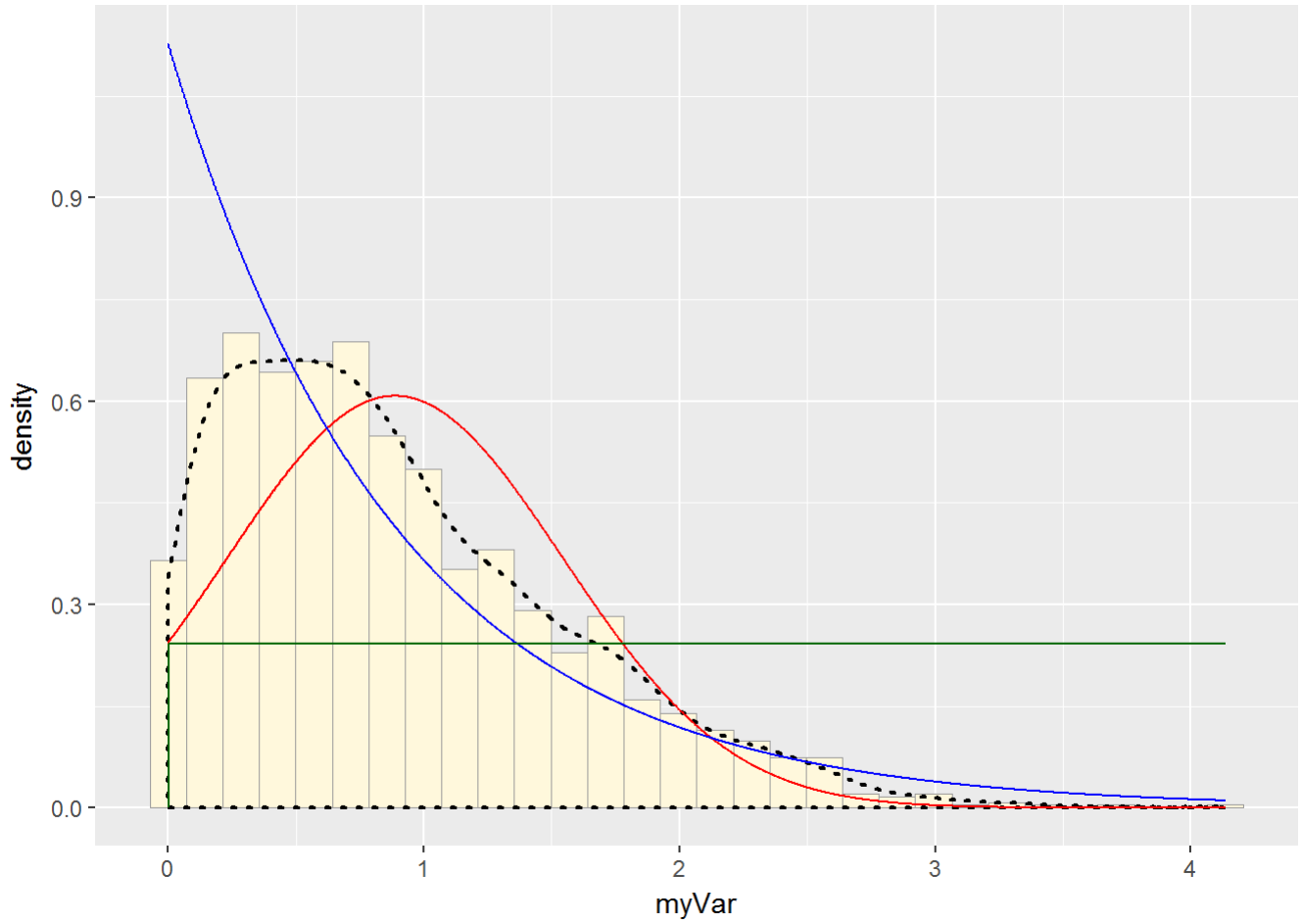
```
expoPars <- fitdistr(z$myVar,"exponential")
rateML <- expoPars$estimate["rate"]

stat2 <- stat_function(aes(x = xval, y = ..y..), fun = dexp, colour="blue", n = length(z$myVar), a
rgs = list(rate=rateML))
 p1 + stat + stat2
```



Plot `uniform` probability density

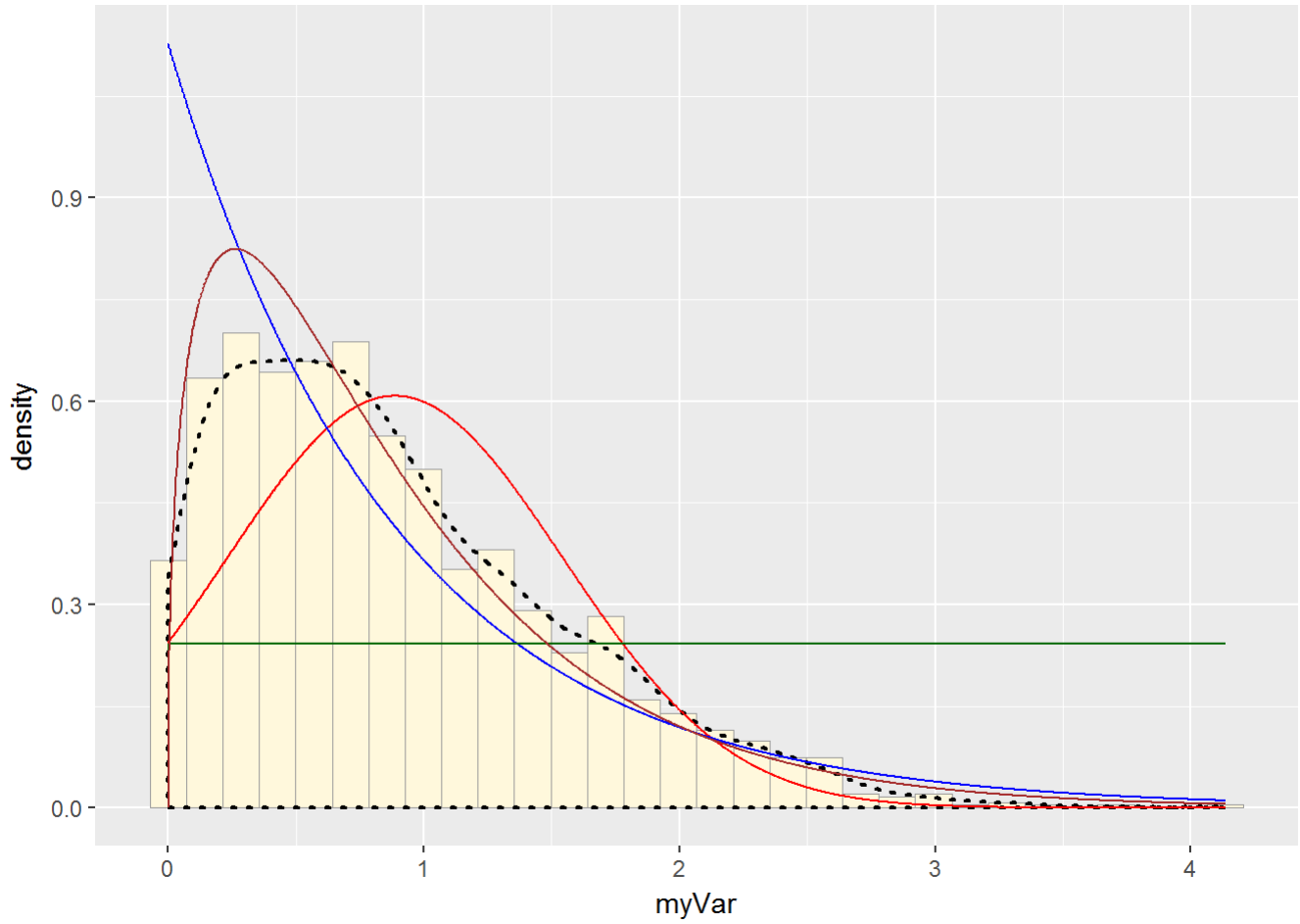
For the uniform, we don't need to use `fitdistr` because the maximum likelihood estimators of the two parameters are just the minimum and the maximum of the data:

```
stat3 <- stat_function(aes(x = xval, y = ..y..), fun = dunif, colour="darkgreen", n = length(z$myV
ar), args = list(min=min(z$myVar), max=max(z$myVar)))
 p1 + stat + stat2 + stat3
```



Plot gamma probability density

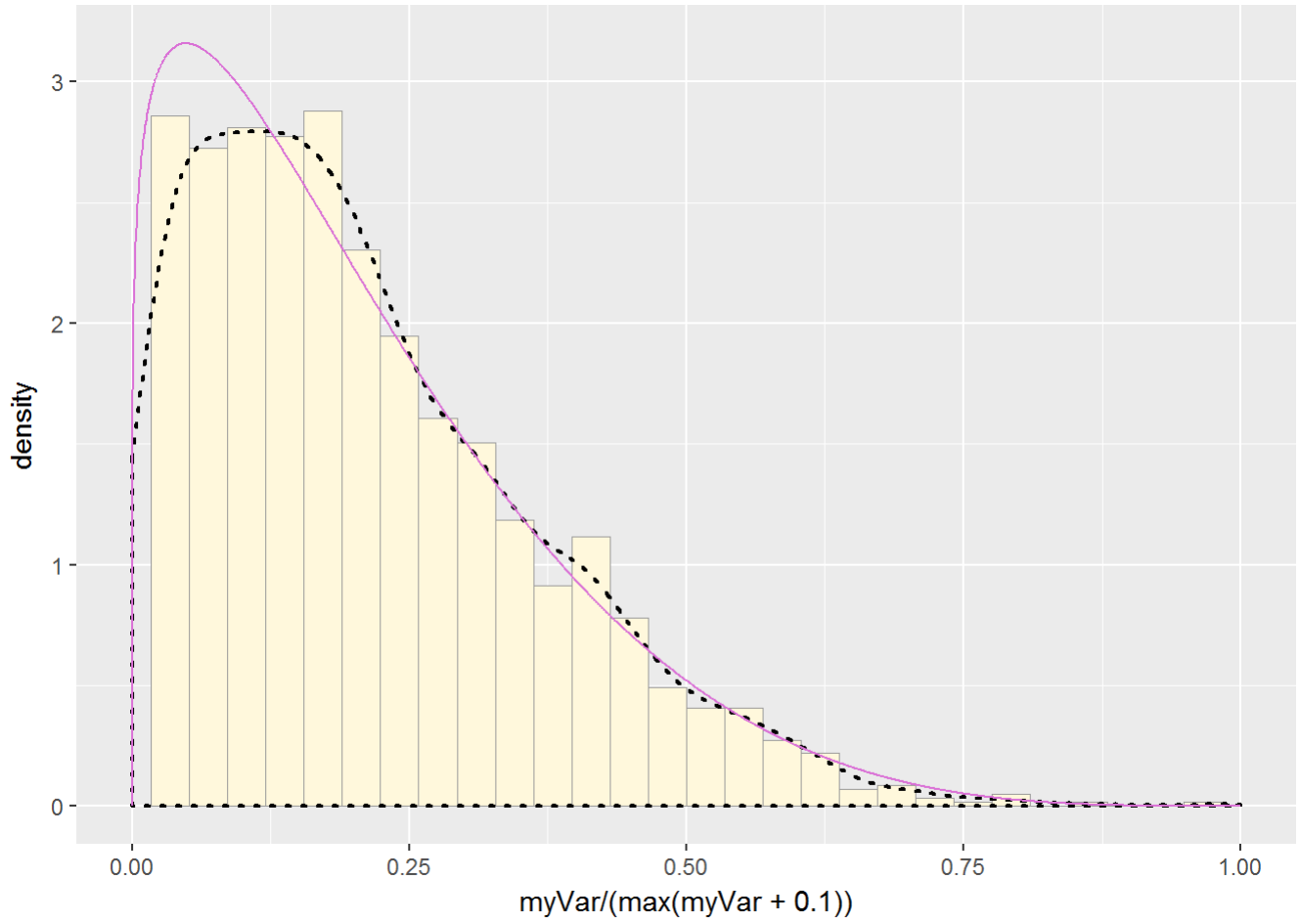
```
gammaPars <- fitdistr(z$myVar,"gamma")
shapeML <- gammaPars$estimate["shape"]
rateML <- gammaPars$estimate["rate"]

stat4 <- stat_function(aes(x = xval, y = ..y..), fun = dgamma, colour="brown", n = length(z$myVa
r), args = list(shape=shapeML, rate=rateML))
 p1 + stat + stat2 + stat3 + stat4
```



Plot `beta` probability density

This one has to be shown in its own plot because the rawdata must be rescaled so they are between 0 and 1, and then they can be compared to the beta.

```
pSpecial <- ggplot(data=z, aes(x=myVar/(max(myVar + 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=z$myVar/max(z$myVar + 0.1),start=list(shape1=1,shape2=2),"beta")
shape1ML <- betaPars$estimate["shape1"]
shape2ML <- betaPars$estimate["shape2"]

statSpecial <- stat_function(aes(x = xval, y = ..y..), fun = dbeta, colour="orchid", n = length(z$
myVar), args = list(shape1=shape1ML,shape2=shape2ML))
pSpecial + statSpecial
```


density
3
2
1
0
0.00
0.25
0.50
0.75
1.00
myVar/(max(myVar + 0.1))