Computer Lab Week 7: solutions

STAT221

Histogram Bin Width Choice

In this question, use the inbuilt dataset from eruptions of the Old Faithful geyser in Yellowstone National Park, USA which is in the datasets library in R:

```
library(datasets)
data(faithful)
help(faithful)
```

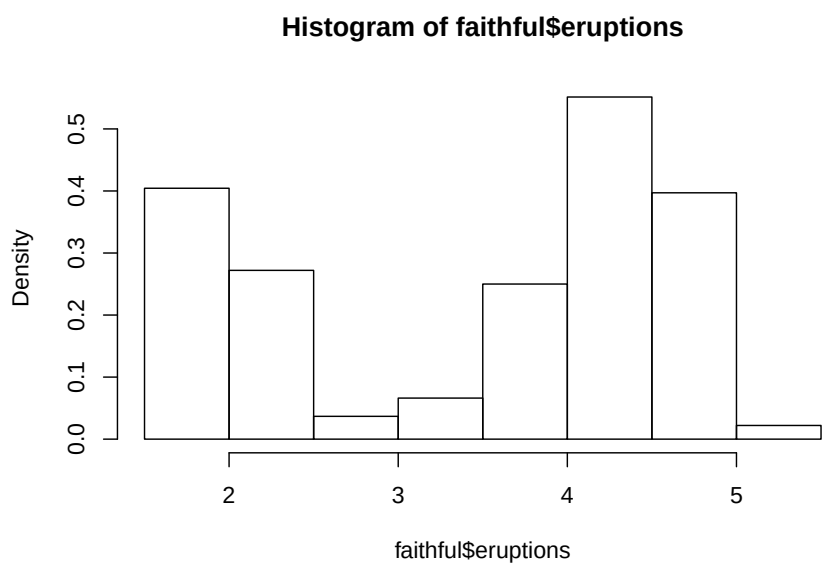
```
## starting httpd help server ... done
```

```
head(faithful, 6)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```


1. Produce a density histogram for the eruption times using the default settings for the bin width choice in R.

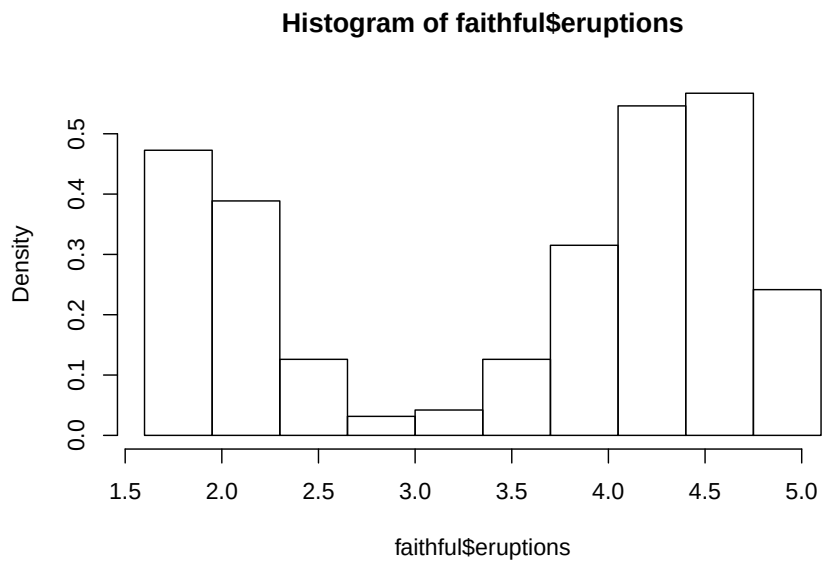
```
hist(faithful$eruptions, freq=FALSE)
```

2. Apply Sturge's Rule to this data and obtain the suggested bin width. Then redo the density histogram applying Sturge's Rule exactly to the eruptions data by defining the breakpoints yourself.

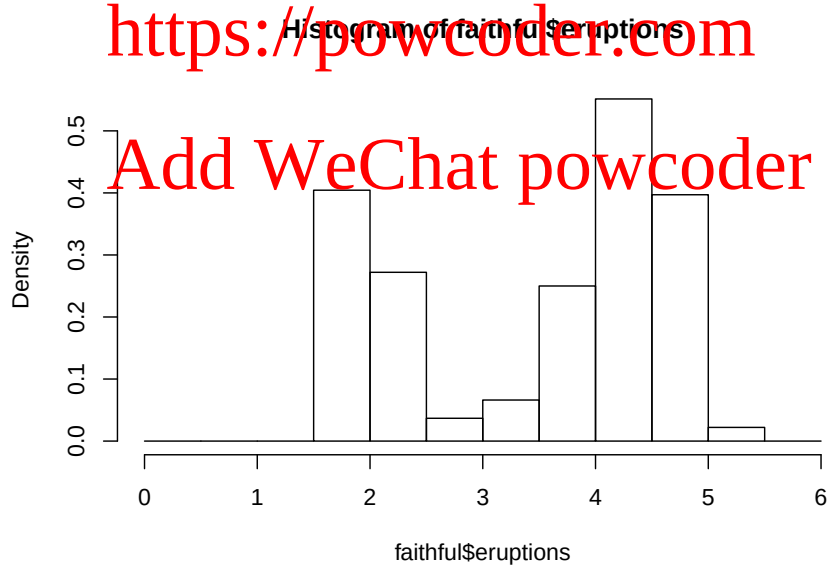
```
h <- diff(range(faithful$eruptions)) / ceiling(1 + log(nrow(faithful), base=2))
hist(faithful$eruptions, freq=FALSE,
```

```
    breaks=seq(min(faithful$eruptions), max(faithful$eruptions), by=h))
```



3. Adapt the code to use breakpoints defined by seq(0, 6, 0.01). The bin width of 0.01 is clearly too small. Try bin widths of 0.1, 0.5, 1 and 2. Decide on the best bin width in your opinion.


```
hist(faithful$eruptions, freq=FALSE, breaks=seq(0, 6, 0.5))
```

Comparing data with a Normal distribution

In statistics, we often need to compare the distribution of a sample of data to a true Normal distribution. An example is after fitting a regression model when we plot a histogram of the residuals and want to make an assessment of whether or not they appear to follow a Normal distribution, or whether there seems to be a problem with the assumption that the residuals are approximately Normal.

In this question you will write a function that produces a histogram for a given sample of data and which also plots on the histogram a smooth Normal distribution with the same mean and standard deviation as

the data itself.

The key to this task is working out how to scale the vertical axis of the smooth Normal distribution.

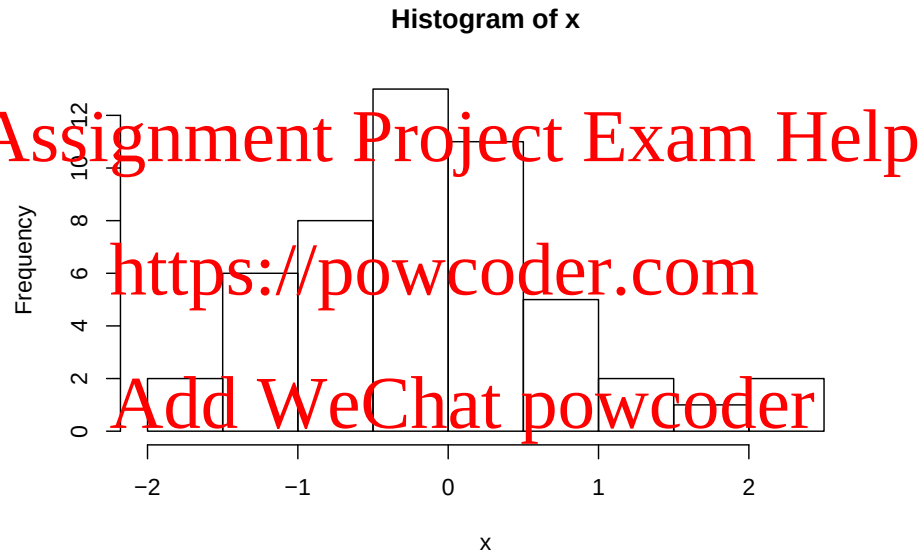
1. Start by writing a very short function (called `my.hist.plot`) that plots a histogram of some data in a vector (called `x`):

```
my.hist.plot = function(x) {
  hist(x)
}
```

Keep your function and working in an R script file so that you can easily edit and save it as you go along.

Check it works by simulating some data from a Normal distribution and testing it (by running `my.hist.plot(x)`). Test it with data from a range of different Normals, so you are sure what you are working with.

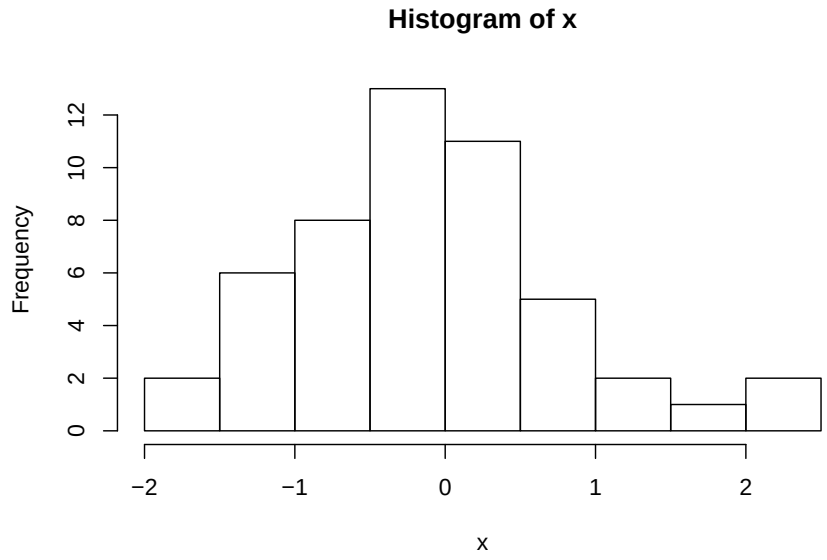
```
set.seed(12)
x = rnorm(50)
my.hist.plot(x)
```



2. Gradually modify your function to include the things you need.

Start by adding some code that calculates the mean, standard deviation (which is the square root of the variance, given by `sqrt(var(x))`), and the number of observations in the dataset `x`.

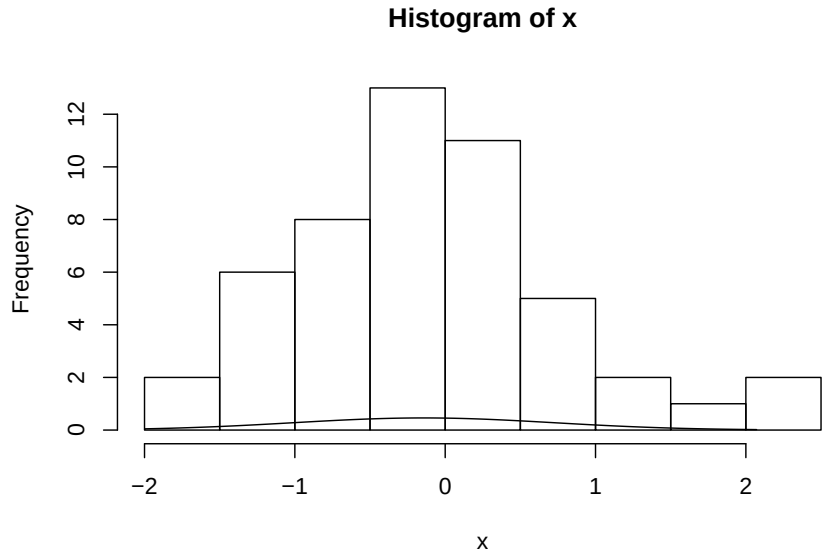
```
my.hist.plot = function(x) {
  x.mean = mean(x)
  x.sd = sqrt(var(x))
  x.n = length(x)
  hist(x)
}
my.hist.plot(x)
```



Then add a Normal curve to your histogram that has the same mean and sd as the data. You will need to use `seq`, as in the lecture notes, and `lines`. When you first do this the curve will not have the correct height - you will sort this out next.

The best approach is to create just one set of sample data and focus on getting that working before trying to generalise your code.


```
my.hist.plot = function(x) {
  x.mean = mean(x)
  x.sd = sqrt(var(x))
  x.n = length(x)
  hist(x)
  x.curve = seq(min(x), max(x), 0.001)
  y.curve = dnorm(x.curve, mean=x.mean, sd=x.sd)
  lines(x.curve, y.curve)
}
my.hist.plot(x)
```

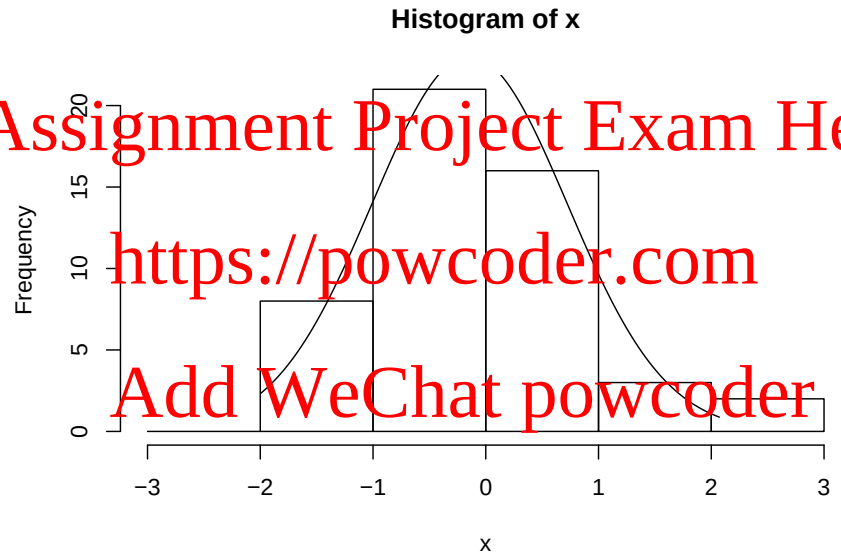


3. The last thing to figure out is the vertical scaling. Remember that the height of a histogram depends on:

- the total sample size, n (the more samples there are, the taller the bars)
- the standard deviation, s (the greater the standard deviation, the lower the bars)
- the break points of the bars (bigger bins are taller).

You will need to specify the breaks in the histogram to get this to work. Again, the best approach is to create just one set of sample data and focus on getting that working before trying to generalise your code.

```
my.hist.plot = function(x) {
  x.mean = mean(x)
  x.sd = sqrt(var(x))
  x.n = length(x)
  hist(x, breaks = -3:3)
  x.curve = seq(min(x), max(x), 0.001)
  y.curve = dnorm(x.curve, mean=x.mean, sd=x.sd)
  lines(x.curve, x.n * y.curve) # works when bins have a width of one
}
my.hist.plot(x)
```



Once you get it working for one set of data, you can keep on playing around with this and improving it further . . .

If you get this far, you are definitely an R programmer for real!