post02

Albert Lee

11/21/2017

Title: Exploring different probability density functions, finding skewness, and measuring the degree fitness of line

Introduction

This post will explore interesting features associated with discrete and continuous probability density functions that have not been covered in lectures and lab. I was motivated to choose probability density functions as the main discussion point of the second post as I am currently taking Stat134 as well and thus want to apply R to the theoretical work. Furthermore, the post will make use of two new and advanced packages, also known as `fBasics` and `TimeSeries` to find out the skewness of density functions. To find out how probability density functions fit the random sample data generated by R, the post will also explore the algebraic and graphic techniques of measuring how good the fit is

Distributions

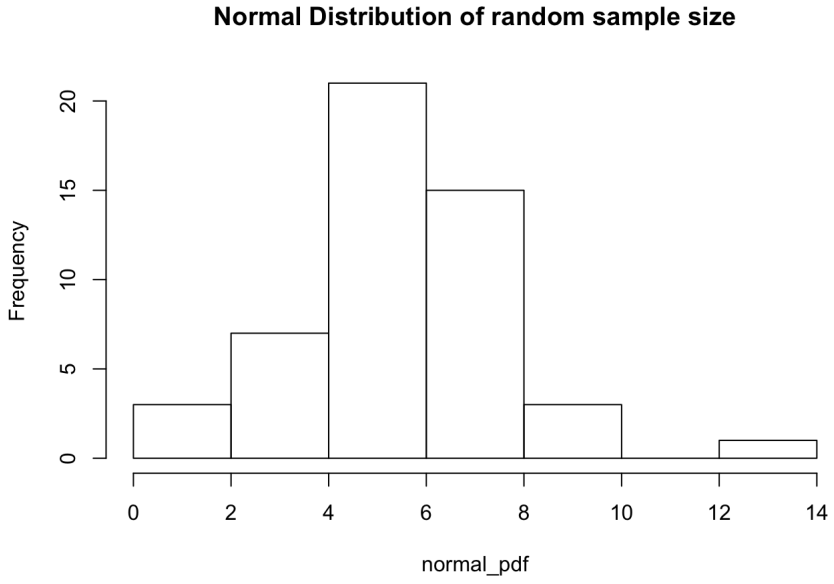
Normal Distribution

Normal distribution is a continuous probability density function that shows a distribution of a sample size 'n', with mean 'm', and standard deviation 'sd'. With normal distribution curve, I am going to explore the curve, density, and the **standard normal curve**

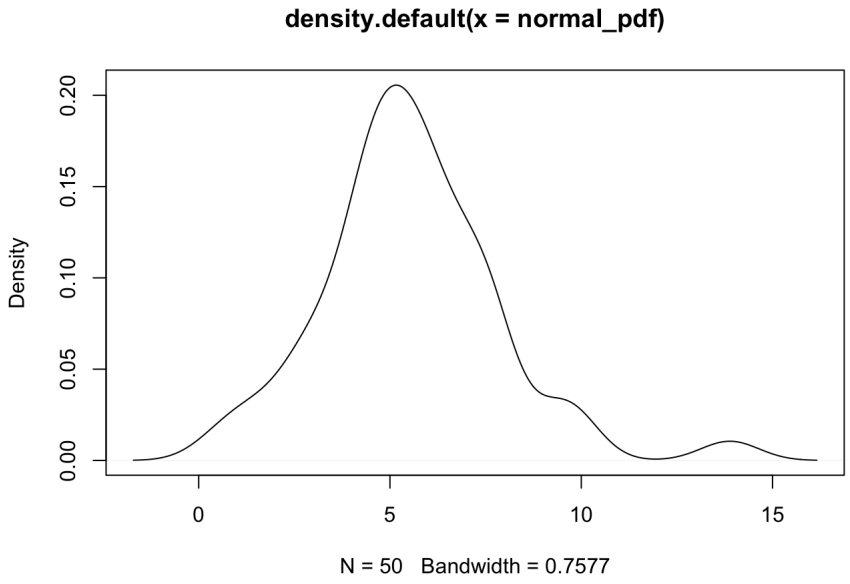
```
## vectorizing `normal_pdf` using a random sample size
normal_pdf <- rnorm(n = 50, m = 5, sd = 2)
normal_pdf
```

```
##  [1]  7.4189183  4.9763326 13.8851691  2.8055696  6.1242876  4.9769696
##  [7]  6.3903104  7.4654086  9.3865567  3.0066617  4.3032378  5.9654629
## [13]  5.0410931  0.5669997  7.4195728  7.9878133  7.4276914  1.6740628
## [19]  5.5154493  4.4141899  1.2595977  3.1176136  6.1683606  6.8785279
## [25]  2.4786516  5.2085195  9.8292027  5.8211683  5.1786956  5.1797291
## [31]  9.6113369  4.1682159  6.1049363  2.9042205  6.1660431  5.6506354
## [37]  7.4510284  5.1328750  5.8478859  4.2850763  4.5455337  7.3661273
## [43]  4.0845482  3.9940928  6.2963891  3.9758318  4.9852459  4.8401351
## [49]  4.6033576  7.5050069
```

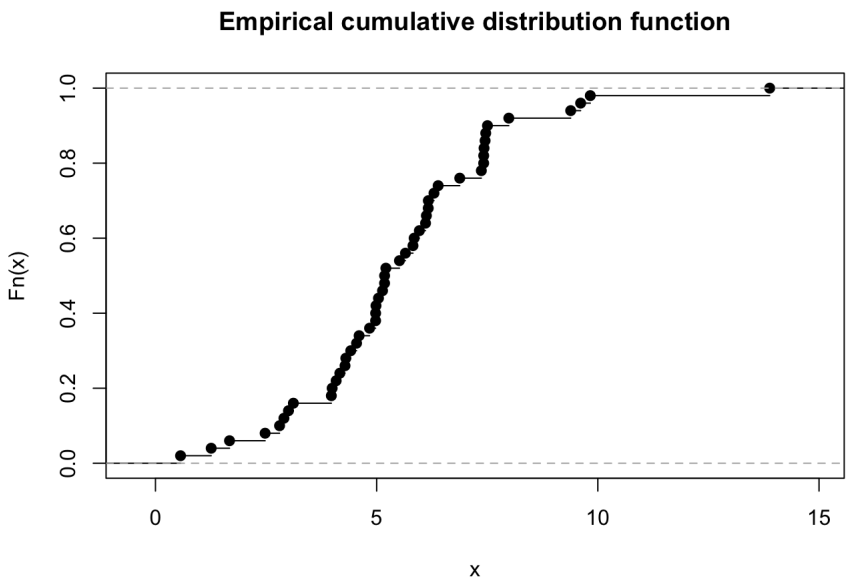
```
## histogram of `normal_pdf`
hist(normal_pdf, main = "Normal Distribution of random sample size")
```




```
## plotting density of `normal_pdf`
plot(density(normal_pdf))
```




```
## function `ecdf()` allows to draw the empirical cumulative distribution function
plot(ecdf(normal_pdf), main = 'Empirical cumulative distribution function')
```




```
## standardized pdf
standardnormal_pdf <- (normal_pdf - mean(normal_pdf)) / sd(normal_pdf)
standardnormal_pdf
```

```
##  [1]  0.80178860 -0.24488207  3.57263708 -1.17507415  0.24702731
##  [6] -0.24460909  0.36102057  0.82171013  1.64493991 -1.08890433
## [11] -0.53330944  0.17896945 -0.21713158 -2.13432213  0.80206904
## [16]  1.04556540  0.80554797 -1.65993531 -0.01386553 -0.48576542
## [21] -1.83753746 -1.04136039  0.26591302  0.57022628 -1.31516155
## [26] -0.14538781  1.83461784  0.11713788 -0.15816762 -0.15772477
## [31]  1.74126033 -0.59116754  0.23873512 -1.13280131  0.26491996
## [36]  0.04406296  0.81554805 -0.17780219  0.12858664 -0.54109180
## [41] -0.42948339  0.77916716 -0.62701994 -0.66578092  0.32077442
## [46] -0.67360593 -0.24106260 -0.30324396 -0.40470529  0.83867837
```

```
## Drawing the qqplot of standard normal distribution
qnorm(standardnormal_pdf)
```

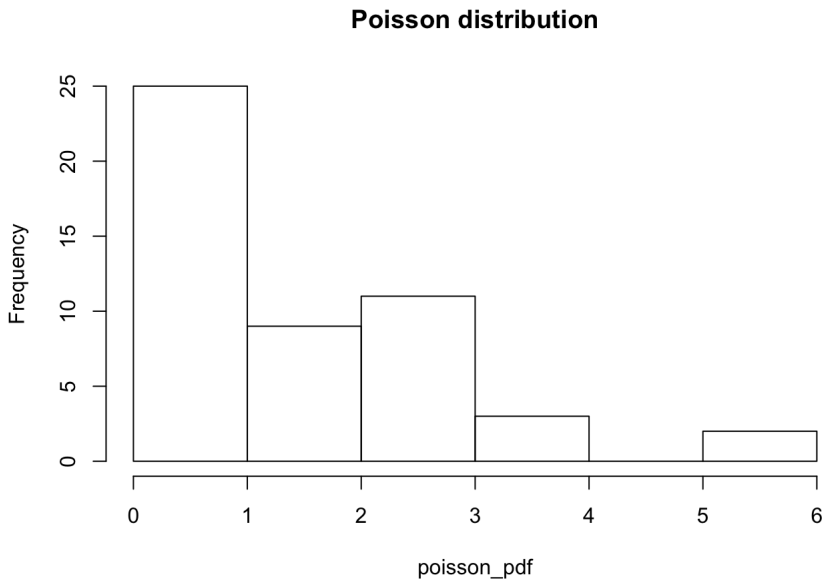
```
## Warning in qnorm(standardnormal_pdf): NaNs produced
```

```
##  [1]  0.8480272        NaN        NaN        NaN -0.6838742        NaN
##  [7] -0.3557322  0.9219019        NaN        NaN        NaN -0.9192996
## [13]        NaN        NaN  0.8490348        NaN  0.8616066        NaN
## [19]        NaN        NaN        NaN        NaN -0.6252210  0.1769503
## [25]        NaN        NaN        NaN -1.1894166        NaN        NaN
## [31]        NaN        NaN -0.7103772        NaN -0.6282504 -1.7053674
## [37]  0.8985284        NaN -1.1330976        NaN        NaN  0.7693835
## [43]        NaN        NaN -0.4655343        NaN        NaN        NaN
## [49]        NaN  0.9890407
```

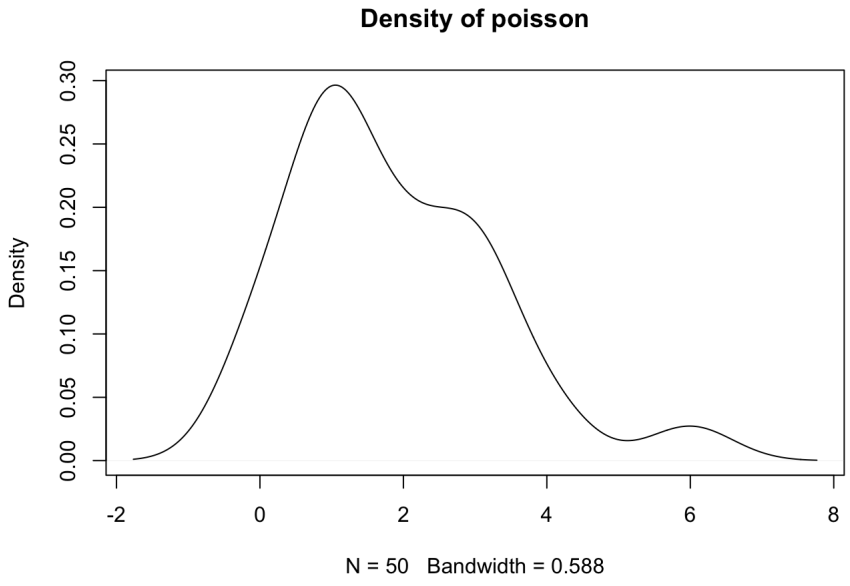
Poisson Distribution

Poisson Distribution is useful when there is a distribution of rare events in a large population. For instance, mutation of a cell, which is a rare event but could happen in a large population, can be modelled by poisson distribution

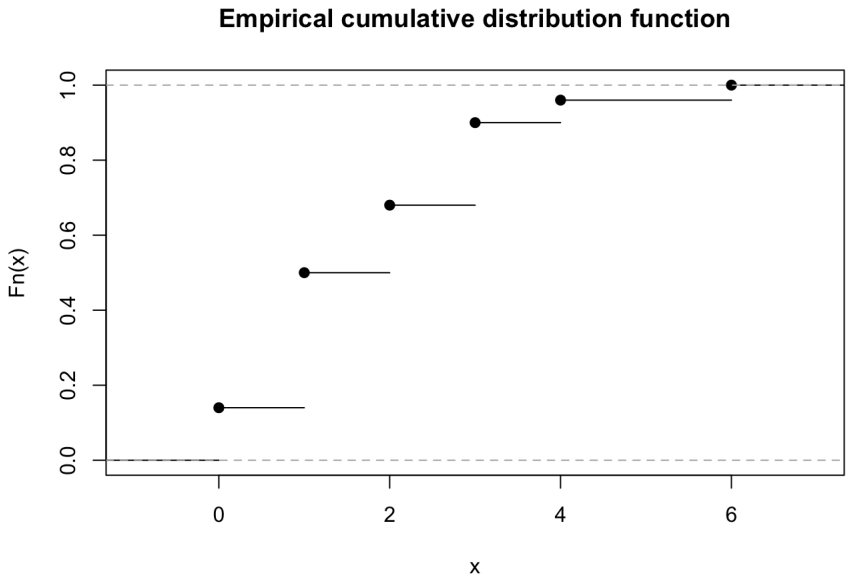
```r
## Vectorizing poisson_pdf as a poisson distribution
poisson_pdf <- rpois(n = 50, lambda = 2)

## Creating a histogram of `poisson_pdf`
hist(poisson_pdf, main = 'Poisson distribution')
```




```r
## Plotting density of `poisson_pdf`
plot(density(poisson_pdf), main = "Density of poisson")
```



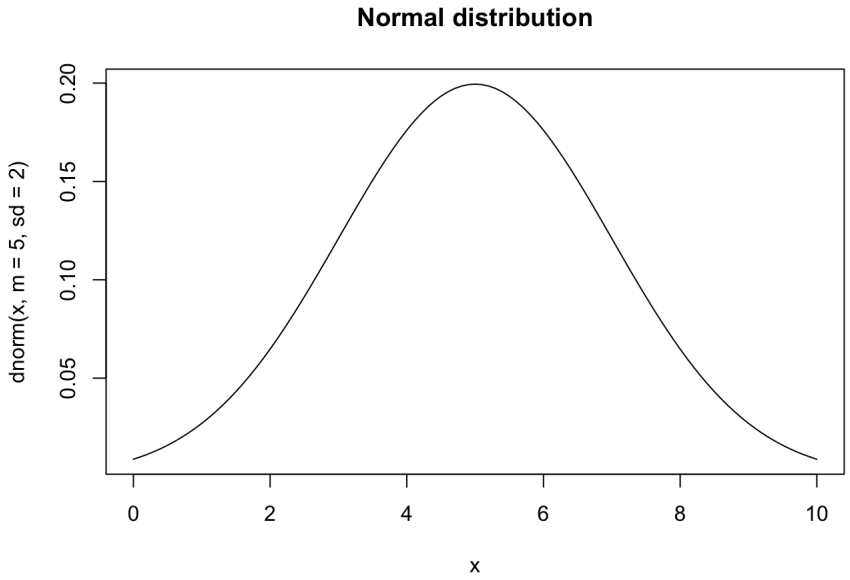

```r
## Plotting cumulative distribution function of `poisson_pdf`
plot(ecdf(poisson_pdf), main = 'Empirical cumulative distribution function')
```



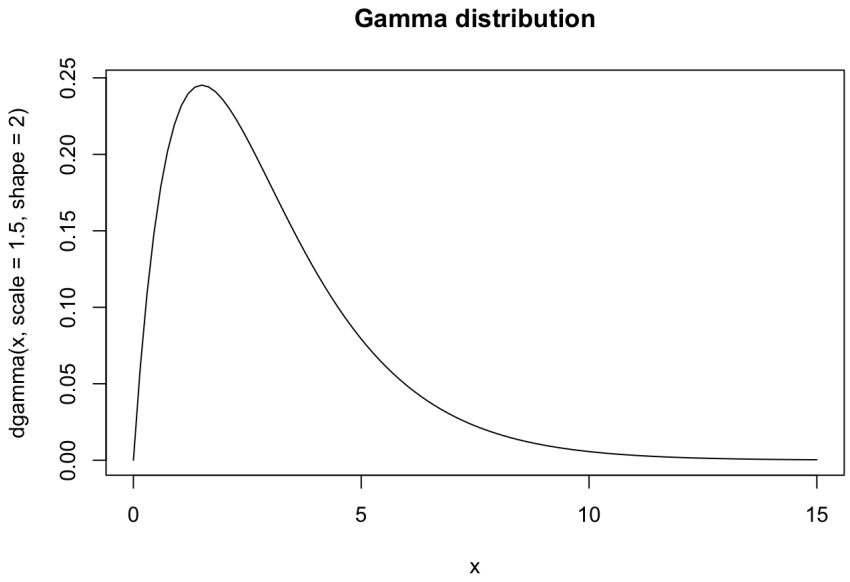
Gaussian and Gamma Distribution

Guassian distribution is just an another name for normal distribution. Gamma distribution in this case can be simply understood as a continuous distribution of discrete poisson distribution (in fact it is poisson process but the theoretical element will not be dealt in this post)

```
## normal (gaussian) distribution
curve(dnorm(x, m = 5, sd = 2), from = 0, to = 10, main = "Normal distribution")
```




```
## gamma distribution
gamma_distribution <- curve(dgamma(x, scale = 1.5, shape = 2), from = 0, to = 15, main = "Gamma distribution")
```




```
gamma_distribution
```

```
## $x
##   [1]  0.00  0.15  0.30  0.45  0.60  0.75  0.90  1.05  1.20  1.35  1.50
##  [12]  1.65  1.80  1.95  2.10  2.25  2.40  2.55  2.70  2.85  3.00  3.15
##  [23]  3.30  3.45  3.60  3.75  3.90  4.05  4.20  4.35  4.50  4.65  4.80
##  [34]  4.95  5.10  5.25  5.40  5.55  5.70  5.85  6.00  6.15  6.30  6.45
##  [45]  6.60  6.75  6.90  7.05  7.20  7.35  7.50  7.65  7.80  7.95  8.10
##  [56]  8.25  8.40  8.55  8.70  8.85  9.00  9.15  9.30  9.45  9.60  9.75
##  [67]  9.90 10.05 10.20 10.35 10.50 10.65 10.80 10.95 11.10 11.25 11.40
##  [78] 11.55 11.70 11.85 12.00 12.15 12.30 12.45 12.60 12.75 12.90 13.05
##  [89] 13.20 13.35 13.50 13.65 13.80 13.95 14.10 14.25 14.40 14.55 14.70
## [100] 14.85 15.00
##
## $y
##   [1] 0.0000000000 0.0603224945 0.1091641004 0.1481636441 0.1787520123
##   [6] 0.2021768866 0.2195246544 0.2317398084 0.2396421142 0.2439417958
##  [11] 0.2452529608 0.2441054614 0.2409553695 0.2361942206 0.2301571663
##  [16] 0.2231301601 0.2153562859 0.2070413273 0.1983586659 0.1894535843
##  [21] 0.1804470443 0.1714389996 0.1625112989 0.1537302270 0.1451487253
##  [26] 0.1368083310 0.1287408689 0.1209699229 0.1135121169 0.1063782254
##  [31] 0.0995741367 0.0931016849 0.0869593685 0.0811429683 0.0756460786
##  [36] 0.0704605613 0.0655769339 0.0609846986 0.0566726220 0.0526289698
##  [41] 0.0488417037 0.0452986461 0.0419876151 0.0388965358 0.0360135304
##  [46] 0.0333269896 0.0308256296 0.0284985349 0.0263351906 0.0243255047
##  [51] 0.0224598233 0.0207289383 0.0191240900 0.0176369651 0.0162596914
##  [56] 0.0149848286 0.0138053579 0.0127146687 0.0117065450 0.0107751496
##  [61] 0.0099150087 0.0091209954 0.0083883133 0.0077124801 0.0070893110
##  [66] 0.0065149032 0.0059856194 0.0054980732 0.0050491140 0.0046358130
##  [71] 0.0042554492 0.0039054966 0.0035836119 0.0032876220 0.0030155136
##  [76] 0.0027654219 0.0025356206 0.0023245129 0.0021306219 0.0019525826
##  [81] 0.0017891340 0.0016391113 0.0015014395 0.0013751264 0.0012592570
##  [86] 0.0011529874 0.0010555399 0.0009661977 0.0008843007 0.0008092410
##  [91] 0.0007404588 0.0006774392 0.0006197083 0.0005668302 0.0005184041
##  [96] 0.0004740616 0.0004334639 0.0003962999 0.0003622838 0.0003311529
## [101] 0.0003026662
```

Computing skewness index

To compute skewness index, we can use the function `skewness()` included in `fBasics` package. Sometimes you might need to download `TimeSeries` in order to use `fBasics`

```
# loading li
library(fBasics)
```

```
## Loading required package: timeDate
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'default/
## America/Los_Angeles'
```

```
## Loading required package: timeSeries
```

```r
library(timeSeries)

skewness(normal_pdf)
```

```
## [1] 0.7341945
## attr(,"method")
## [1] "moment"
```

```r
skewness(poisson_pdf)
```

```
## [1] 0.899389
## attr(,"method")
## [1] "moment"
```

```r
skewness(gamma_distribution)
```

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

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

Measures of degree of fitness of line

Analaysing from poisson distribution dataset algebraically

This algebraic test allows us to find the absolute value of the degree of fitness of line of poisson model

```r
## Estimate of the `lambda` parameter
lambda_estimate <- mean(poisson_pdf)

## A table with empirical values of frequencies in Poisson distribution
table_freq <- table(poisson_pdf)
table_freq
```

```
## poisson_pdf
##  0  1  2  3  4  6
##  7 18  9 11  3  2
```

```r
## Vectorizing `freq_vector`
freq_vector <- vector()

## Vector of empirical frequencies
for (i in 1: length(table_freq)) freq_vector [i] <- table_freq [[i]]

## Vectorizing `expected_frequencies`
expected_freq <- (dpois(0:max(poisson_pdf), lambda = lambda_estimate)*50)

## return `table_freq`
table_freq
```

```
## poisson_pdf
##  0  1  2  3  4  6
##  7 18  9 11  3  2
```

```r
## return `expected_freq`
expected_freq
```

```
## [1]  7.7836315 14.4775546 13.4641258  8.3477580  3.8817075  1.4439952
## [7]  0.4476385
```

```r
## calculating absolute value of the degree of fitness of line
abs_fitness <- mean(abs(freq_vector - trunc(expected_freq)))
```

```
## Warning in freq_vector - trunc(expected_freq): longer object length is not
## a multiple of shorter object length
```

```r
abs_fitness
```

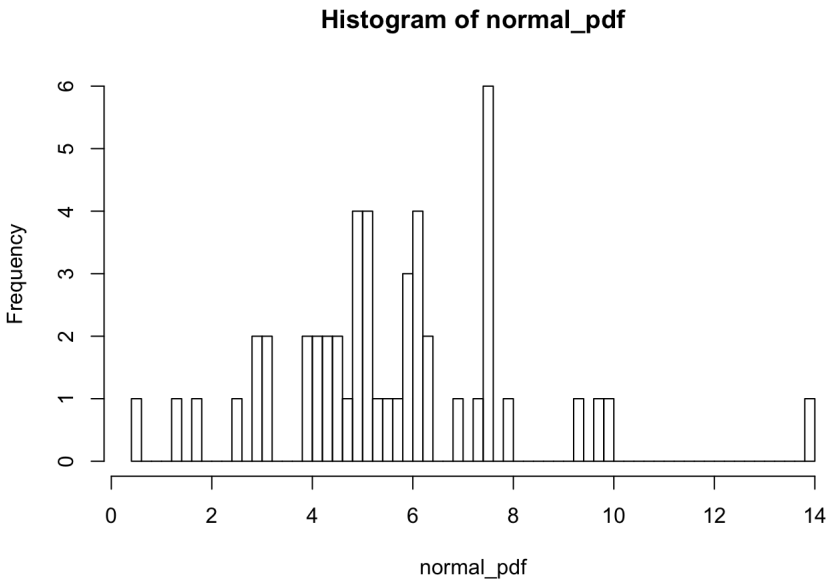
```
## [1] 2.714286
```

```r
## calculating percentage of degree of fitness of line index
abs_fitness / mean(freq_vector) * 100
```

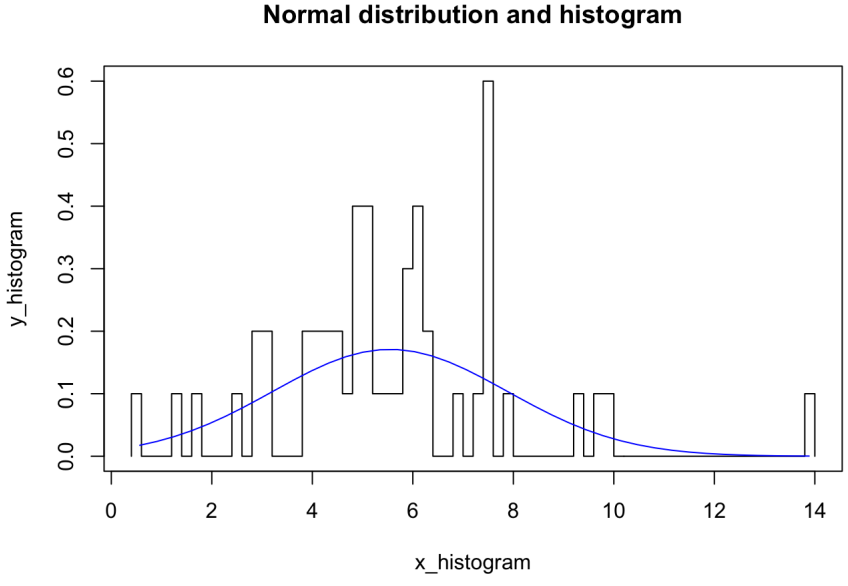
```
## [1] 32.57143
```

Analyzing using graphical technique on normal distribution

```r
## creating a histogram of normal probability density function `hist_normalpdf`
hist_normalpdf <- hist(normal_pdf, breaks = 50)
```




```r
## creating `x_histogram`
x_histogram <- c(min(hist_normalpdf$breaks), hist_normalpdf$breaks)

## creating `y_histogram`
y_histogram <- c(0, hist_normalpdf$density, 0)

## measuring degree of fitness of line of x
x_goodfitness <- seq(min(normal_pdf), max(normal_pdf), length = 40)

## measuring degree of fitness of line of y
y_goodfitness <- dnorm(x_goodfitness, mean = mean(normal_pdf), sd = sd(normal_pdf))

## plotting a finalized histogram
plot(x_histogram, y_histogram, type = 's', ylim = c(0, max(y_histogram, y_goodfitness)), main = "Normal distributi
on and histogram")

## drawing a line of comparison
lines(x_goodfitness, y_goodfitness, col = "Blue")
```



Messages/Conclusion

This post explored different distributions and functions you can use to find out simple and advanced aspects of distributions that you might or might not have covered before. The packages are also useful in that you do not have to write a complex code to find skewness but rather use the package to find the skewness index. Degree of fitness of line index also allows you to also match whether the randomly generated dataset matches the theoretical values and it shows that although there might be a slight correlation, it does not imply causation and thus require a larger sample size of fixed values.

References

- Reference for my inspiration behind this post
  - Source 1: The Normal Distribution
  - Source 2: R Tutorial
  - Source 3: The Gamma Distribution
  - Source 4: fBasics: Rmetrics
  - Source 5: Degree of fitness of line test in R | R-blogger
  - Source 6: Poisson Regression
  - Source 7: Elementary-Statistics