Data Project Sample Lottery

Karim Adnane

Introduction: Set Up

Our goal is to analyze the lottery number distributions for the Cash4Life New York sample and provide data visualizations. The first step is to load the necessary R packages.

```
install.packages("tidyverse", repos = c('http://rforge.net', 'http://cran.rstudio.org'), type = 'source
```

```
## also installing the dependencies 'pillar', 'rlang', 'tibble'

## Warning in install.packages("tidyverse", repos = c("http://rforge.net", :
## installation of package 'rlang' had non-zero exit status

## Warning in install.packages("tidyverse", repos = c("http://rforge.net", :
## installation of package 'pillar' had non-zero exit status

## Warning in install.packages("tidyverse", repos = c("http://rforge.net", :
## installation of package 'tibble' had non-zero exit status

## Warning in install.packages("tidyverse", repos = c("http://rforge.net", :
## installation of package 'tidyverse' had non-zero exit status
```

```
install.packages("mosaic", repos = c('http://rforge.net', 'http://cran.rstudio.org'), type = 'source')

install.packages("ggplot2", repos = c('http://rforge.net', 'http://cran.rstudio.org'), type = 'source')

install.packages("dplyr", repos = c('http://rforge.net', 'http://cran.rstudio.org'), type = 'source')
```

```
## also installing the dependencies 'rlang', 'tidyselect', 'vctrs'

## Warning in install.packages("dplyr", repos = c("http://rforge.net", "http://
## cran.rstudio.org"), : installation of package 'rlang' had non-zero exit status

## Warning in install.packages("dplyr", repos = c("http://rforge.net", "http://
## cran.rstudio.org"), : installation of package 'vctrs' had non-zero exit status

## Warning in install.packages("dplyr", repos = c("http://rforge.net", "http://
## cran.rstudio.org"), : installation of package 'tidyselect' had non-zero exit
## status

## Warning in install.packages("dplyr", repos = c("http://rforge.net", "http://
## cran.rstudio.org"), : installation of package 'dplyr' had non-zero exit status
```

```
install.packages("stargazer", repos = c('http://rforge.net', 'http://cran.rstudio.org'), type = 'source

library(mosaic)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
##
## Attaching package: 'mosaic'
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
## The following object is masked from 'package:Matrix':
##
##     mean
## The following object is masked from 'package:ggplot2':
##
##     stat
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
library(stargazer)
```

```
##
## Please cite as:
##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.
##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
library(readxl)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v tibble  3.1.4     v purrr   0.3.4
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## Warning: package 'tibble' was built under R version 4.0.5
## Warning: package 'forcats' was built under R version 4.0.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x mosaic::count()            masks dplyr::count()
## x purrr::cross()             masks mosaic::cross()
## x mosaic::do()               masks dplyr::do()
## x tidyr::expand()            masks Matrix::expand()
## x dplyr::filter()            masks stats::filter()
## x ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## x dplyr::lag()               masks stats::lag()
## x tidyr::pack()              masks Matrix::pack()
## x mosaic::stat()             masks ggplot2::stat()
## x mosaic::tally()            masks dplyr::tally()
## x tidyr::unpack()            masks Matrix::unpack()
```

```
library(ggplot2)
library(dplyr)
```

Using the "Open Data NY" website, we download and upload the "Lottery Cash 4 Life Winning Numbers: Beginning 2014" excel sheet. For now, we wont be converting any data columns/rows.

Link: https://data.ny.gov/Government-Finance/Lottery-Cash-4-Life-Winning-Numbers-Beginning-2014/kwxv-fwze

```
Lottery_Cash_4_Life_Winning_Numbers_Beginning_2014_Test <- read_excel("C:/Users/KARIM/Downloads/Lottery_

data <- Lottery_Cash_4_Life_Winning_Numbers_Beginning_2014_Test
View(data)
```

To examine the numbers, we need to extract the numbers from the winning lottery tickets and separate them into five columns, one for each section. The Cash Balls column will simply be renamed and converted to numeric values. First, let's create 5 new columns from the separate lottery number values.

```
data[c('A', 'B', 'C', 'D', 'E')] <- str_split_fixed(data$`Winning Numbers`, ' ', 5)
View(data)

data$CB <- data$`Cash Ball`
```

Now we will convert the new columns into numeric values.

```
sapply(data, class)
```

```
## $`Draw Date`
## [1] "POSIXct" "POSIXt"
##
## $`Winning Numbers`
## [1] "character"
##
## $`Cash Ball`
## [1] "numeric"
##
## $A
## [1] "character"
##
## $B
## [1] "character"
##
## $C
## [1] "character"
##
## $D
## [1] "character"
##
## $E
## [1] "character"
##
## $CB
## [1] "numeric"
```

```
cols.num <- c("A","B", "C", "D", "E")
data[cols.num] <- sapply(data[cols.num],as.numeric)
```

```
sapply(data, class)
```

```
## $`Draw Date`
## [1] "POSIXct" "POSIXt"
##
## $`Winning Numbers`
## [1] "character"
##
## $`Cash Ball`
## [1] "numeric"
##
## $A
## [1] "numeric"
##
## $B
## [1] "numeric"
##
## $C
## [1] "numeric"
##
## $D
## [1] "numeric"
##
## $E
## [1] "numeric"
##
## $CB
## [1] "numeric"
```

Thankfully we had no corrupted data values or unclean data. We can check this by calculating the summary statistics for each of the columns and looking for any anomalies.

```
#Delete non-numeric columns (not needed)

data <- select(data, -c("Draw Date", "Winning Numbers", "Cash Ball"))
data
```

```
## # A tibble: 1,637 x 6
##        A     B     C     D     E    CB
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     7    20    22    30    46     3
##  2    26    27    32    47    55     1
##  3     4    30    44    52    55     4
##  4     4     6     7    17    46     1
##  5     5    22    23    44    59     4
##  6     6    10    18    41    50     2
##  7     8    28    35    46    57     4
##  8     2    10    15    22    29     4
##  9     1     5    11    21    33     4
## 10    14    30    34    40    48     2
## # ... with 1,627 more rows
```

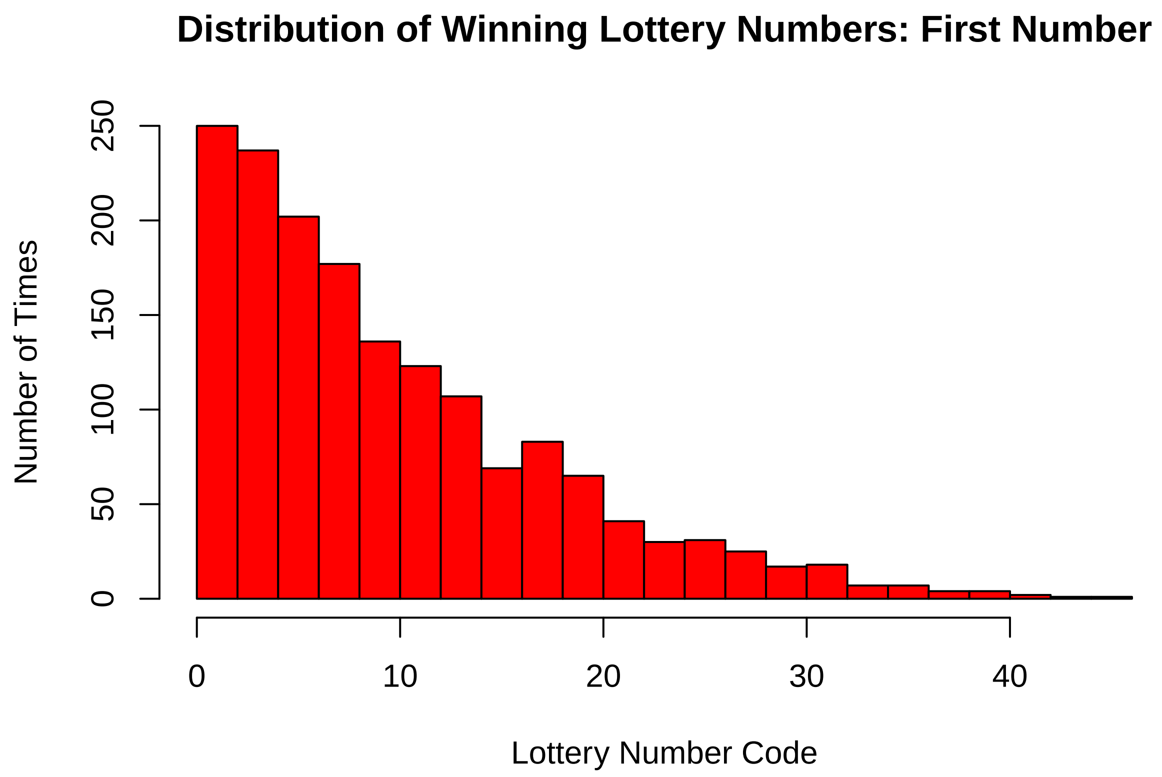
```
summary(data, type = "text", flip = TRUE)
```

```
##        A                B                C                D
##  Min.   : 1.00   Min.   : 2.00   Min.   : 5.00   Min.   : 9.00
##  1st Qu.: 4.00   1st Qu.:12.00   1st Qu.:22.00   1st Qu.:34.00  
##  Median : 8.00   Median :19.00   Median :30.00   Median :41.00  
##  Mean   :10.18   Mean   :20.15   Mean   :30.33   Mean   :40.44  
##  3rd Qu.:14.00   3rd Qu.:27.00   3rd Qu.:39.00   3rd Qu.:49.00  
##  Max.   :46.00   Max.   :56.00   Max.   :58.00   Max.   :59.00  
##        E               CB       
##  Min.   :14.00   Min.   :1.000  
##  1st Qu.:46.00   1st Qu.:1.000  
##  Median :53.00   Median :2.000  
##  Mean   :50.54   Mean   :2.471  
##  3rd Qu.:57.00   3rd Qu.:3.000  
##  Max.   :60.00   Max.   :4.000
```

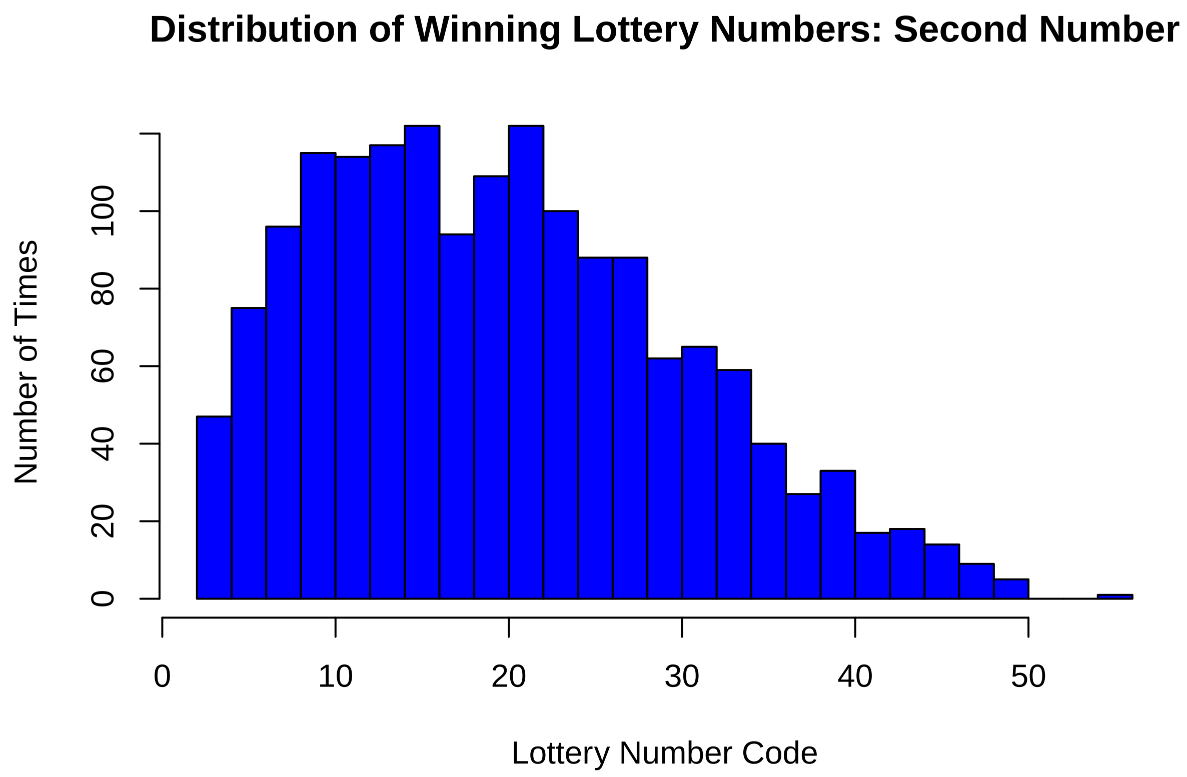
Already, we start to notice something interesting, particularly with the medians. Mathematically speaking, 50% of numbers in the distribution can be found below the median. For example, for the first section of the winning lottery numbers, 50% of winning numbers are below 8. We will examine this closer in the next sections.

Data Visualization

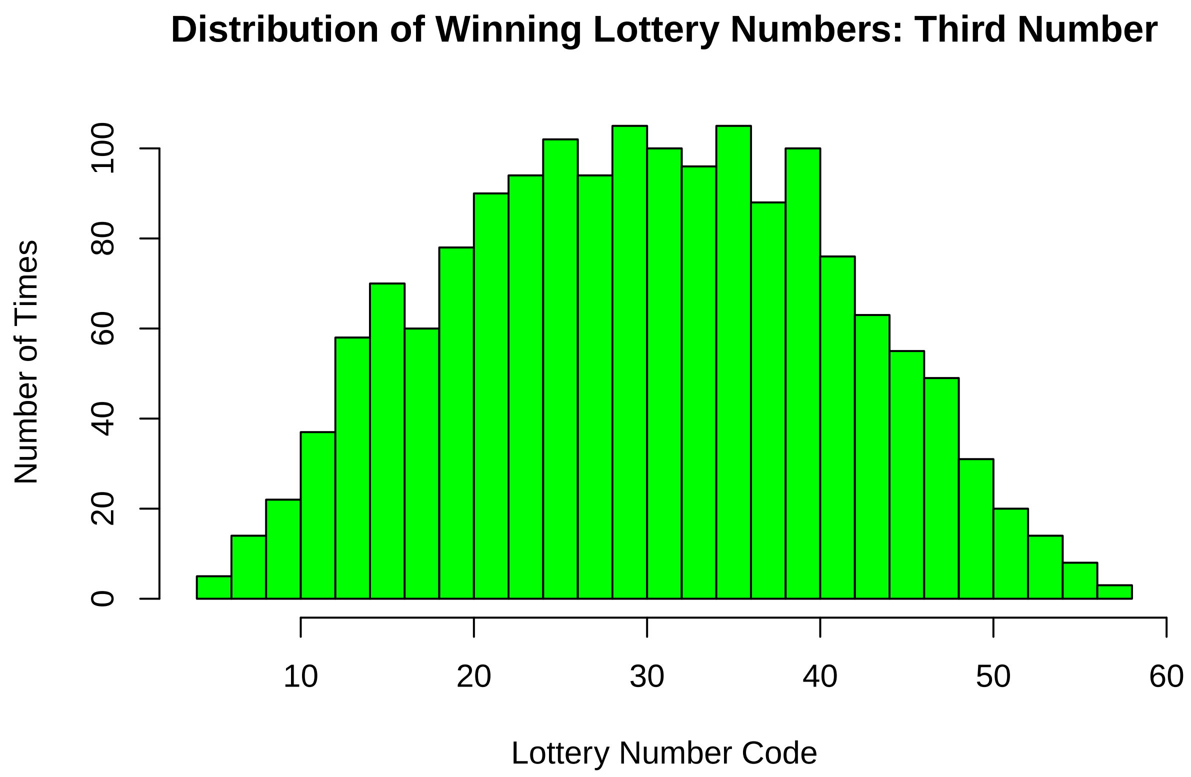
```
A <- data$A
B <- data$B
C <- data$C
D <- data$D
E <- data$E
CB <- data$CB
hist(A, main="Distribution of Winning Lottery Numbers: First Number", col='red', xlab= "Lottery Number (
```



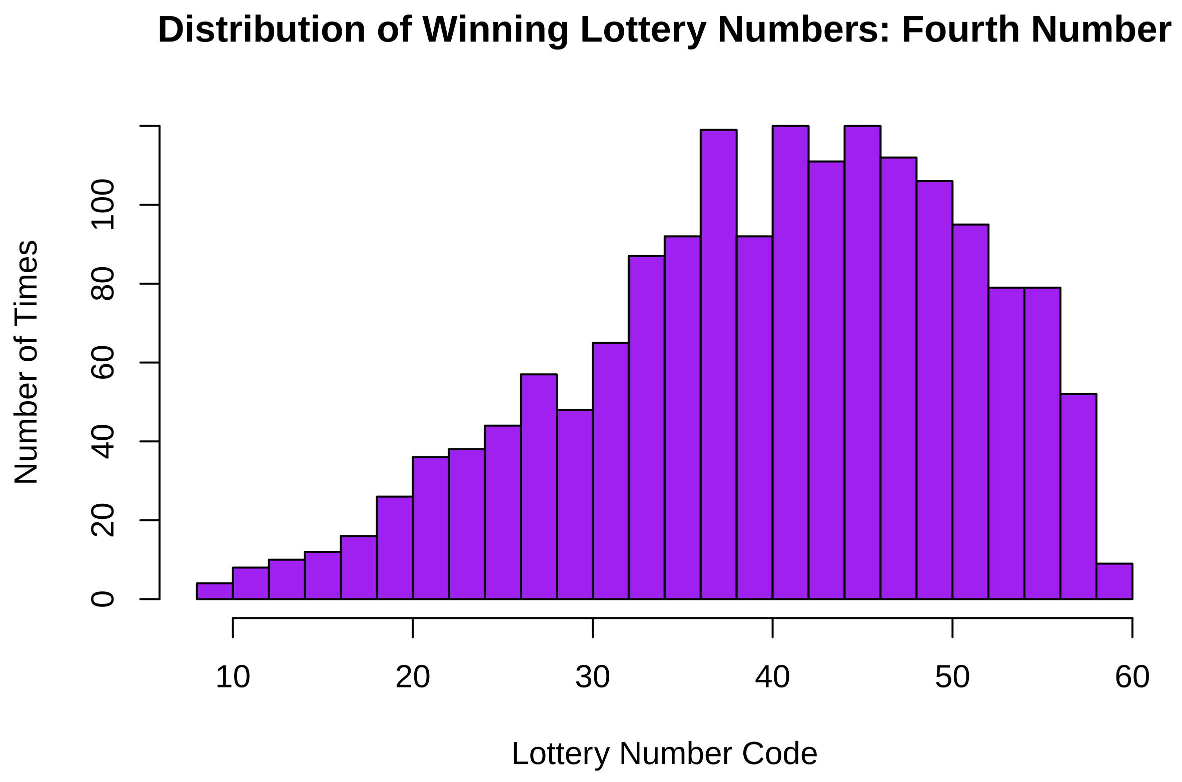
```
hist(B, main="Distribution of Winning Lottery Numbers: Second Number", col='blue', xlab= "Lottery Numbe
```



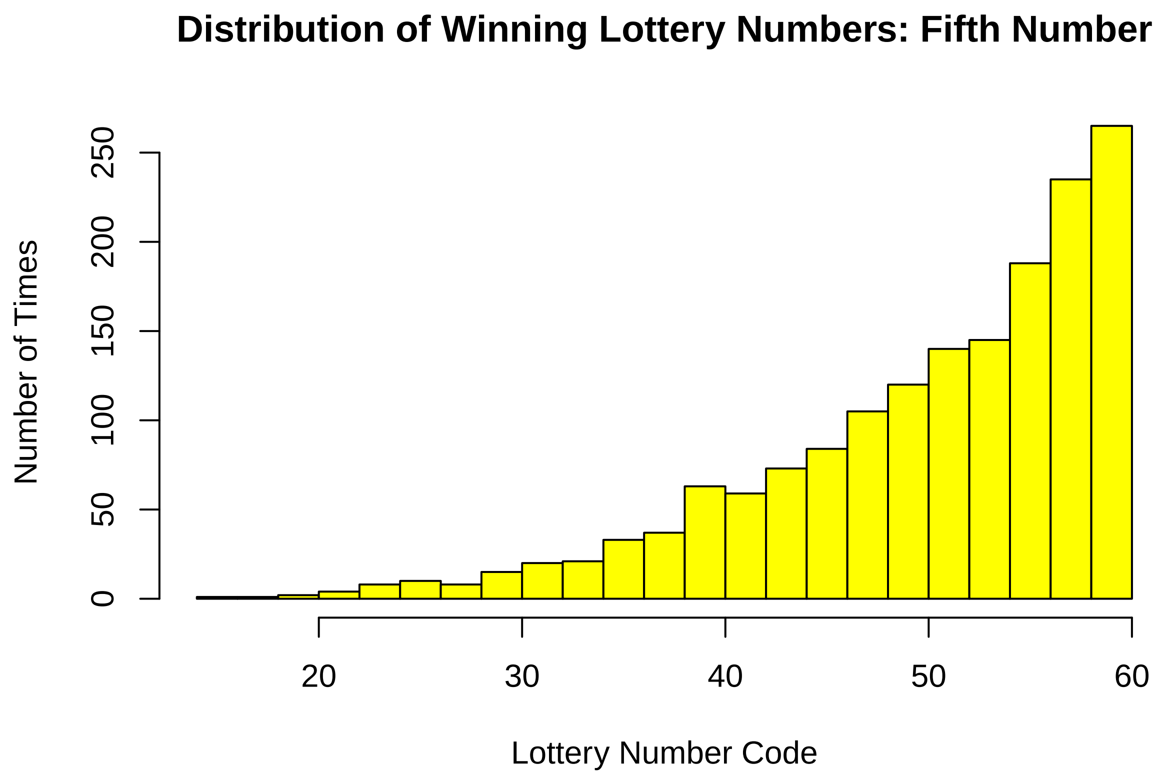
```
hist(C, main="Distribution of Winning Lottery Numbers: Third Number", col='green', xlab= "Lottery Numbe
```



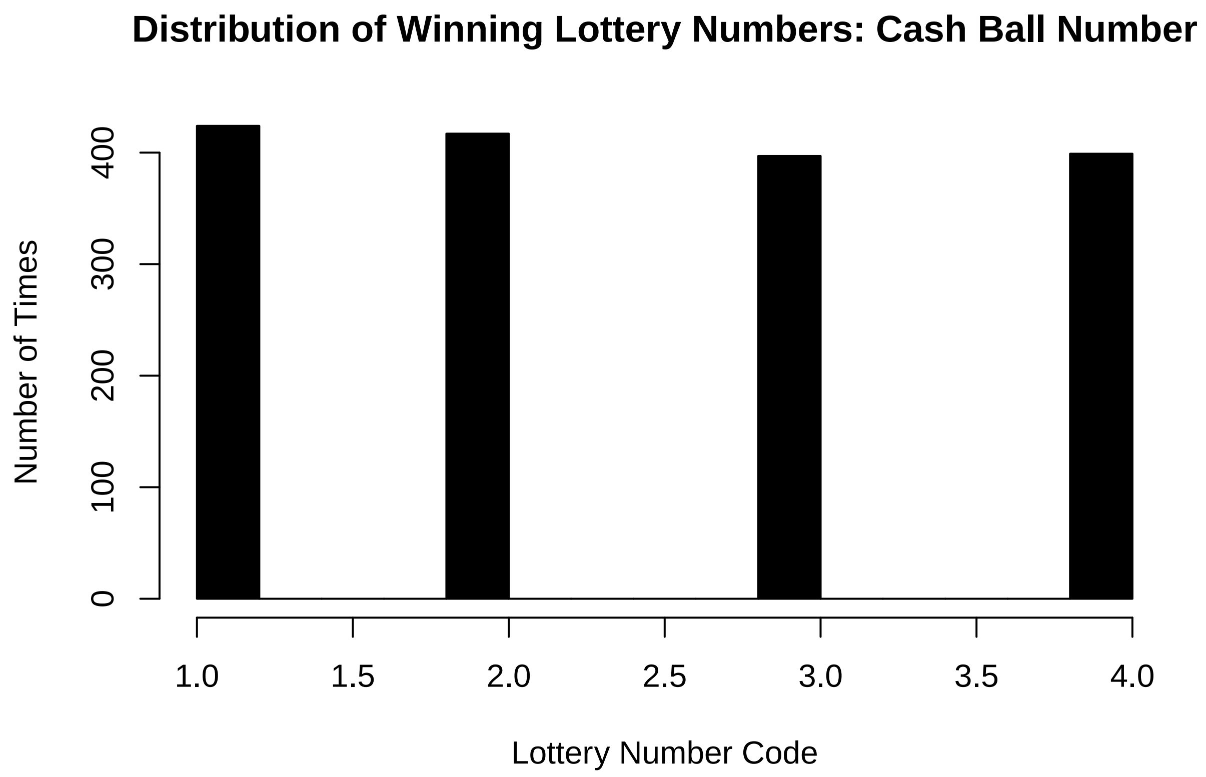
```
hist(D, main="Distribution of Winning Lottery Numbers: Fourth Number", col='purple', xlab= "Lottery Numl
```



```
hist(E, main="Distribution of Winning Lottery Numbers: Fifth Number", col='yellow', xlab= "Lottery Numbe
```



```
hist(CB, main="Distribution of Winning Lottery Numbers: Cash Ball Number", col='black', xlab= "Lottery N
```



Each section of the winning lottery numbers follows variations of normal distributions. In probability and statistics, we call these the skew-normal distributions, and we will prove this in the later sections. It is important to note that for the Cash4Life Lottery, you choose numbers ranging from 1 to 60. Cash Ball is 1 to 4.

Section A (first number), follows a right-skewed distribution, where 50% of numbers are below 8.

Section B (second number), follows a slightly right-skewed distribution, where 50% of numbers are below 19

Section C (third number), follows a near-perfect normal distribution, where 50% of numbers are below 30.

Section D (fourth number), follows a slightly left-skewed distribution, where 50% of numbers are below 41.

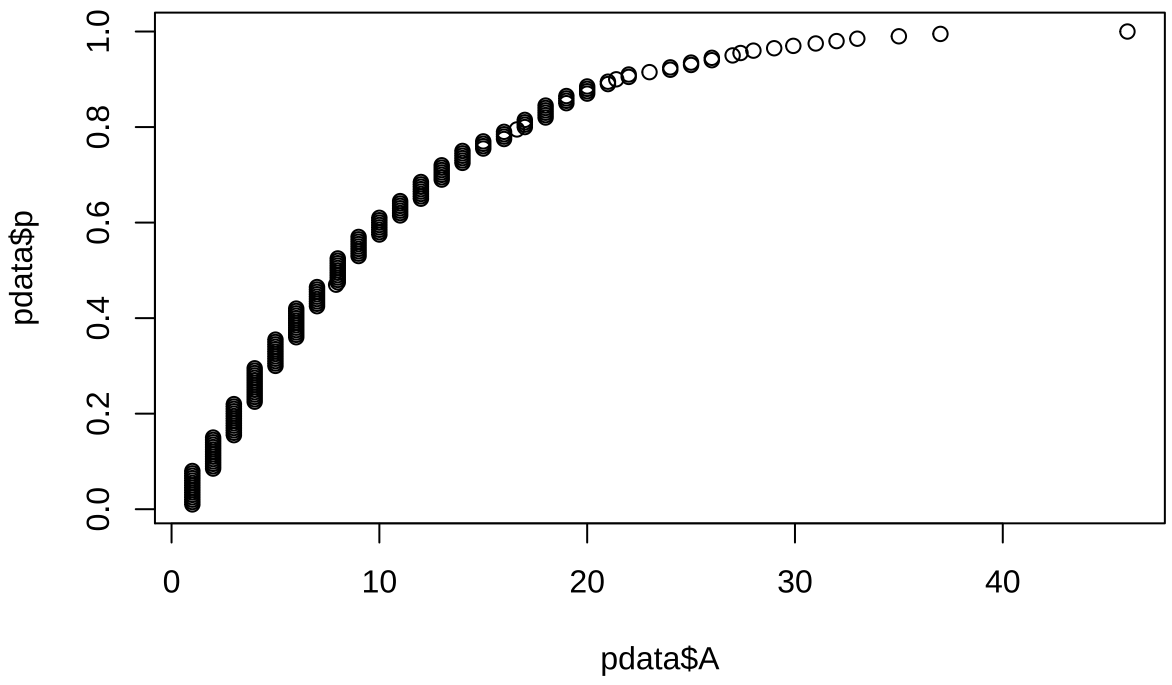
Section E (fifth number), follows a left-skewed distribution, where 50% of numbers are below 53.

Section CB (Cash Ball) does not follow any variation of the normal distribution. Instead, it is a uniform distribution of numbers 1, 2, 3, and 4.

For sections A-E, to show that these follow skew-normal distributions, first, we need to create a data frame of percentile values for each section and plot them. This will give us a cumulative distribution plot.

```
#Create a sequence of percentiles

p <- seq(from = 0.01, to = 1, by = 0.005)
pdata <- as.data.frame(p)

#Calculate percentiles of each section using the dataframe

pdata$A <- quantile(A, pdata$p)
pdata$B <- quantile(B, pdata$p)
```

```
pdata$C <- quantile(C, pdata$p)
pdata$D <- quantile(D, pdata$p)
pdata$E <- quantile(E, pdata$p)
View(pdata)

#Example Plot and Histogram

plot(pdata$A, pdata$p)
```



Discussion: Removing Points

Now that we have the cumulative distribution plots, we can interpolate a cumulative distribution function correct? Not yet.

An issue that we can see from the example plot is the stacking of points, for example, the number "1" has multiple Y percentile values. If we were to use any polynomial interpolation method, we would not get accurate results. For the best interpolating method for our data, if we keep all the points, the polynomial function line would go through the average of each X value (Winning Lottery Number Code) and this would not provide accurate results. In laymen's terms, there are too many stacking points and we can not estimate a "best fit" line until we get rid of the points.

To solve this issue, we need to create five separate data frames, for each section. This is to make sure we don't remove important data rows when we remove rows based on one column. Here is an example of what we are trying to do. Suppose we have the following dataset

0.10 - 1

0.20 - 1

0.30 - 1

0.40 - 2

0.50 - 2

0.60 - 3

Based on this dataset, 30% of numbers are 1's. So, we want to remove the unnecessary duplicates.

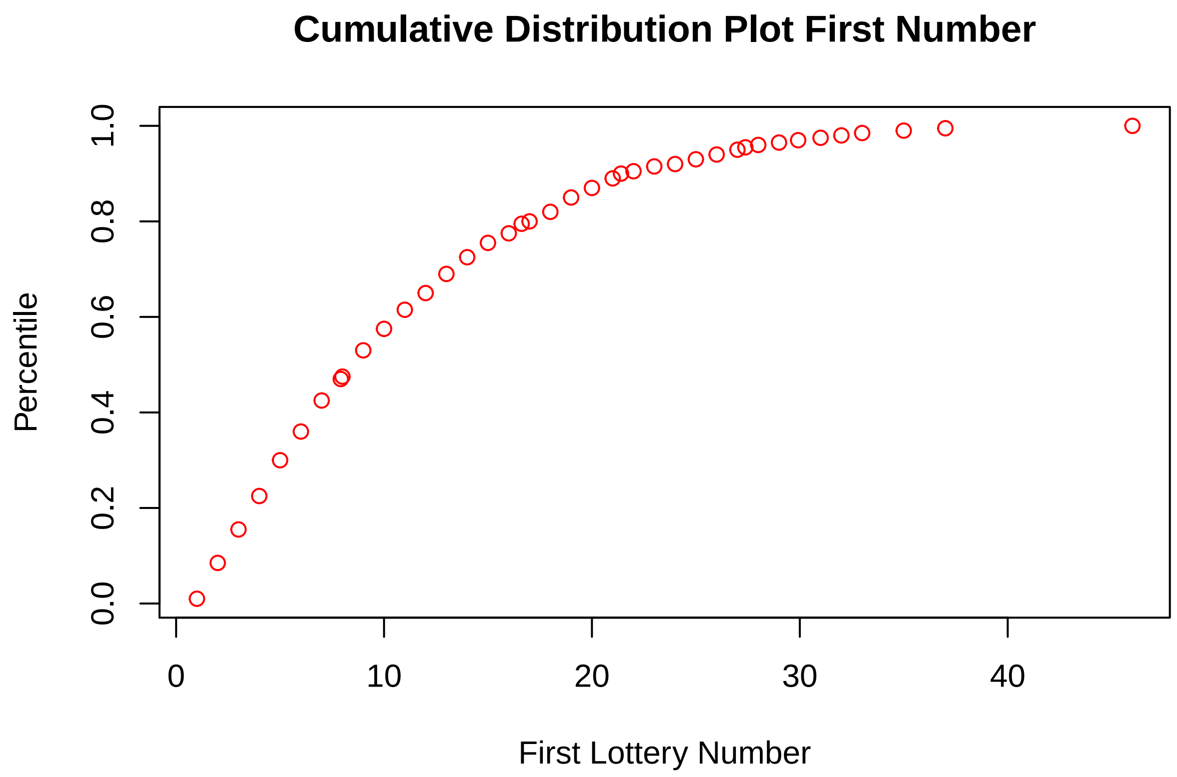
0.10 - 1

0.40 - 2

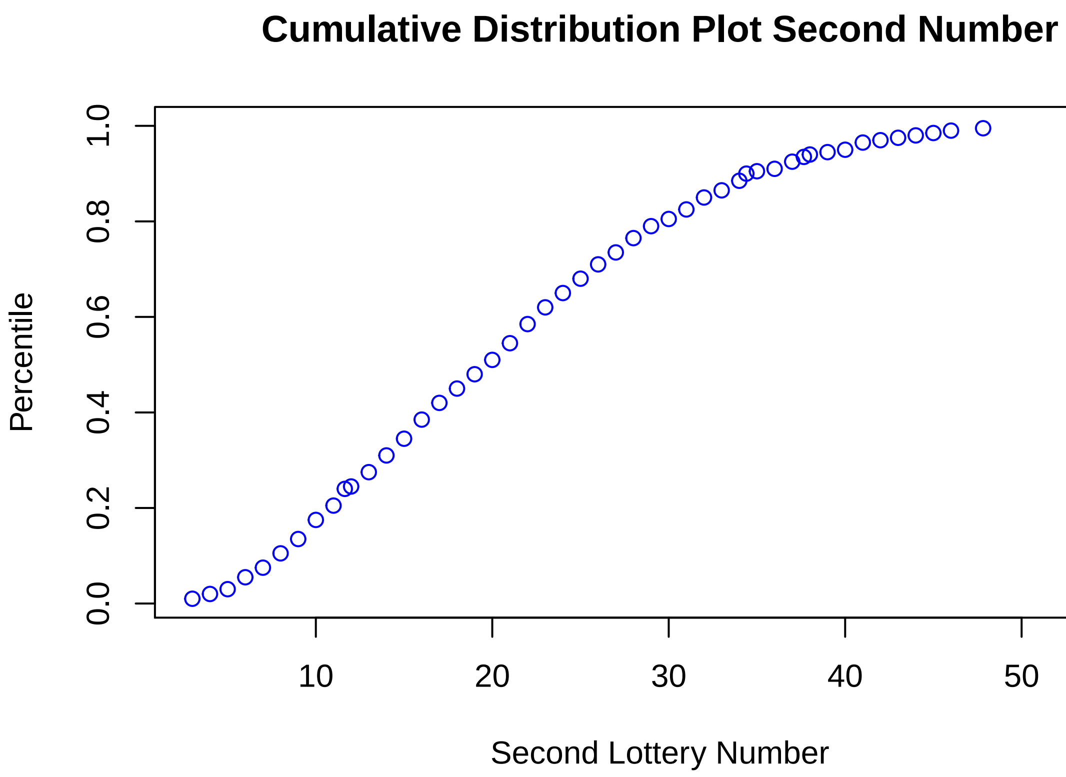
0.60 - 3

Did we remove the fact that 30% of numbers are 1's? No. We have the starting percentile of 0.40 for 2's. Since we know that 1's start at 0.10, to calculate what percent of numbers are 1's, subtract 0.10 from 0.40 and we get 0.30. Thankfully, since we calculated 221 percentiles, our results will be more accurate.

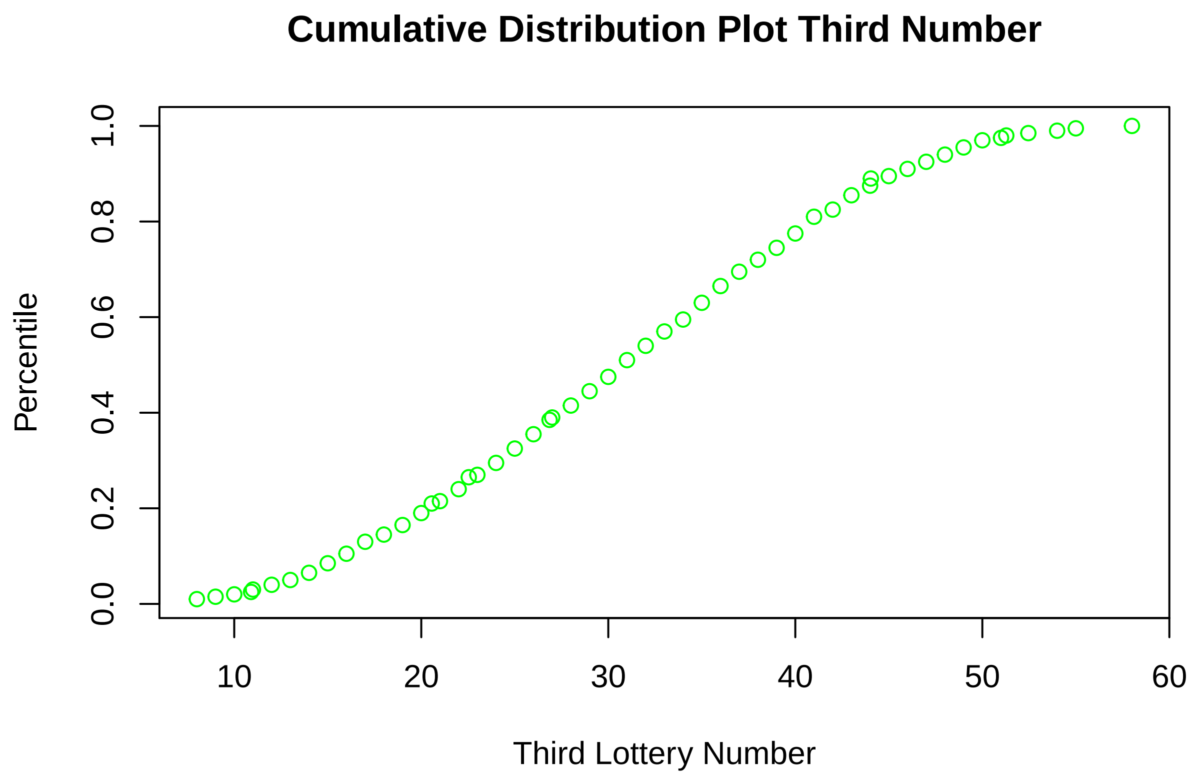
```r
dataA <- cbind.data.frame(pdata$p, pdata$A)
dataB <- cbind.data.frame(pdata$p, pdata$B)
dataC <- cbind.data.frame(pdata$p, pdata$C)
dataD <- cbind.data.frame(pdata$p, pdata$D)
dataE <- cbind.data.frame(pdata$p, pdata$E)

dataA <- dataA[!duplicated(dataA$`pdata$A`), ]
dataB <- dataB[!duplicated(dataB$`pdata$B`), ]
dataC <- dataC[!duplicated(dataC$`pdata$C`), ]
dataD <- dataD[!duplicated(dataD$`pdata$D`), ]
dataE <- dataE[!duplicated(dataE$`pdata$E`), ]

dataA$percentiles <- dataA$`pdata$p`
dataA$A <- dataA$`pdata$A`

dataB$percentiles <- dataB$`pdata$p`
dataB$B <- dataB$`pdata$B`

dataC$percentiles <- dataC$`pdata$p`
dataC$C <- dataC$`pdata$C`

dataD$percentiles <- dataD$`pdata$p`
dataD$D <- dataD$`pdata$D`

dataE$percentiles <- dataE$`pdata$p`
dataE$E <- dataE$`pdata$E`

#Example Plots

plot(dataA$A, dataA$percentiles, xlab="First Lottery Number", ylab="Percentile", main="Cumulative Distri
```



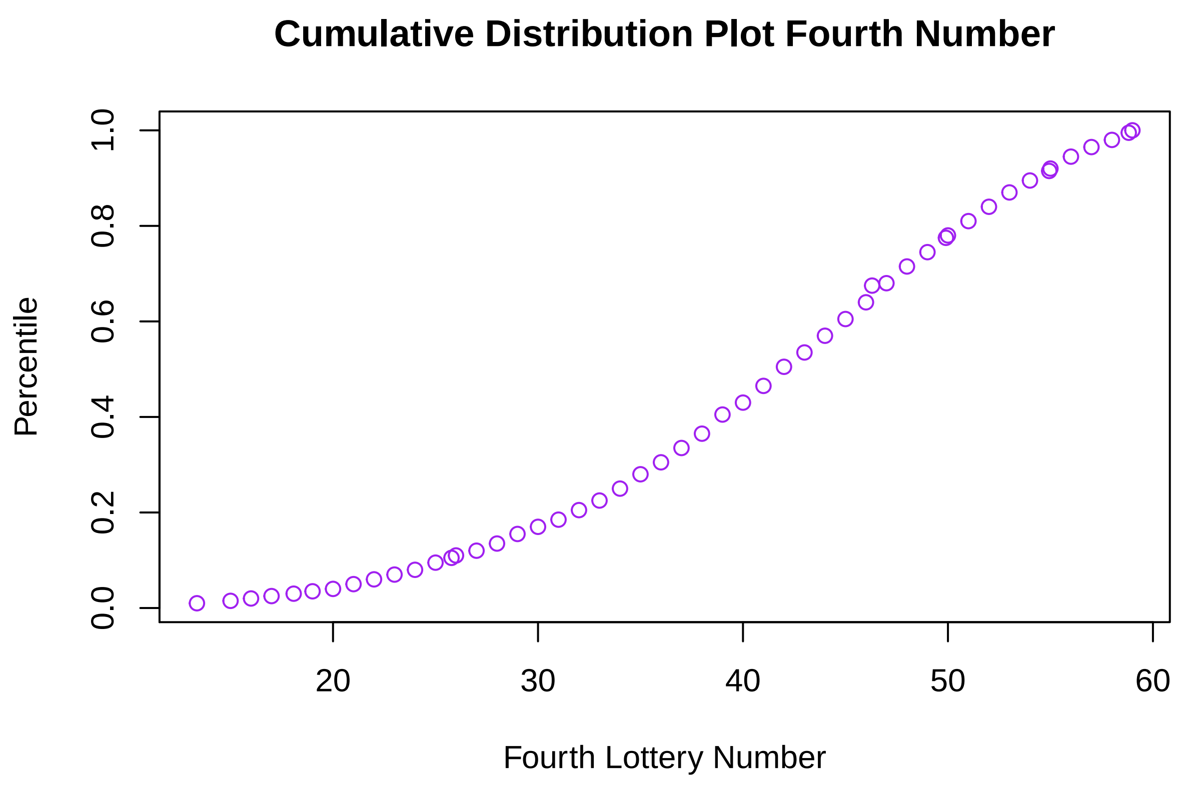
```
plot(dataB$B, dataB$percentiles, xlab="Second Lottery Number", ylab="Percentile", main="Cumulative Dist
```



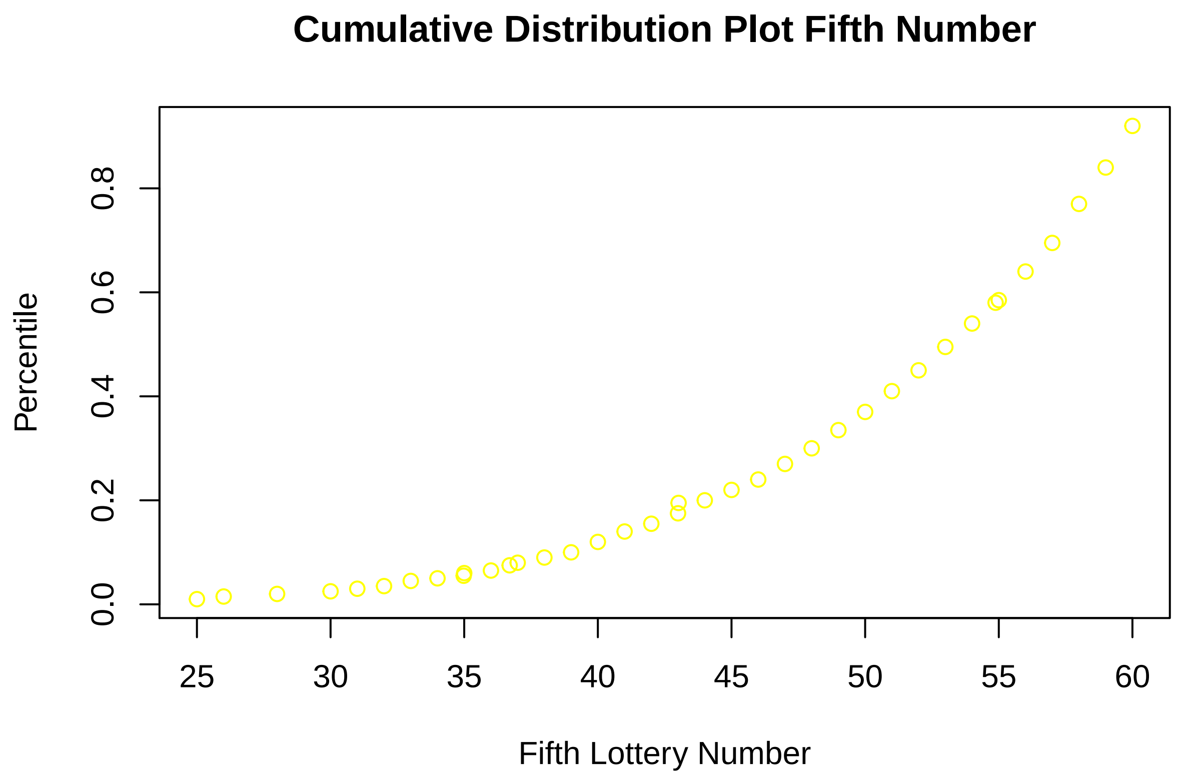
```
plot(dataC$C, dataC$percentiles, xlab="Third Lottery Number", ylab="Percentile", main="Cumulative Distr
```



```
plot(dataD$D, dataD$percentiles, xlab="Fourth Lottery Number", ylab="Percentile", main="Cumulative Dist
```

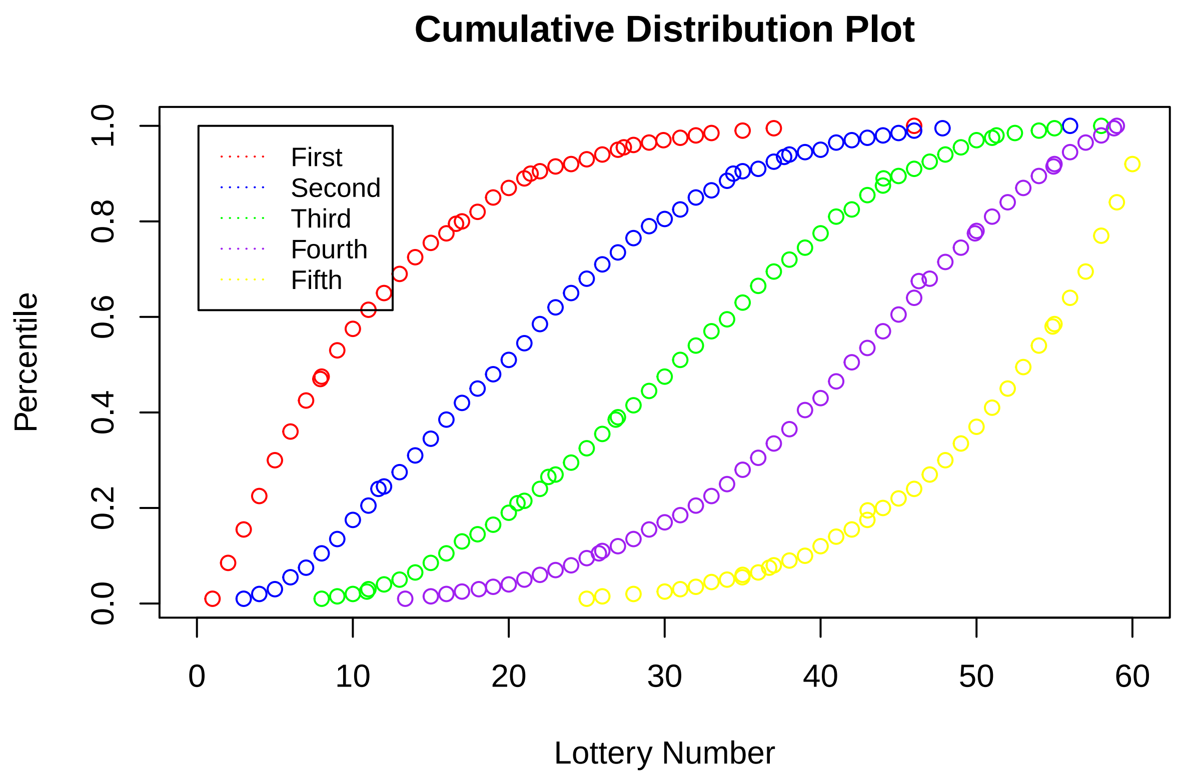


```
plot(dataE$E, dataE$percentiles, xlab="Fifth Lottery Number", ylab="Percentile", main="Cumulative Distr
```



Lets combine the plots

```
plot(dataA$A, dataA$percentiles, xlab="Lottery Number", ylab="Percentile", main="Cumulative Distributio
points(dataB$B, dataB$percentiles, col="blue")
points(dataC$C, dataC$percentiles, col="green")
points(dataD$D, dataD$percentiles, col="purple")
points(dataE$E, dataE$percentiles, col="yellow")


legend(0.1, 1, legend=c("First", "Second", "Third", "Fourth", "Fifth"),
       col=c("red", "blue", "green", "purple", "yellow"), lty=3, cex=0.8)
```



Polynomial Fitting & Interpolation Using Regression

The best method to fit a line through these points is to perform a polynomial regression of a certain degree. This will also give us an estimate of a cumulative distribution function for each number section of winning lottery numbers.

What I found was that degrees 4, and 5 provide the best estimates.

Y = B0 + B1*x B2*x^2 + B3*x^3 + B4*x^4 + B5*x^5 + B6*x^6

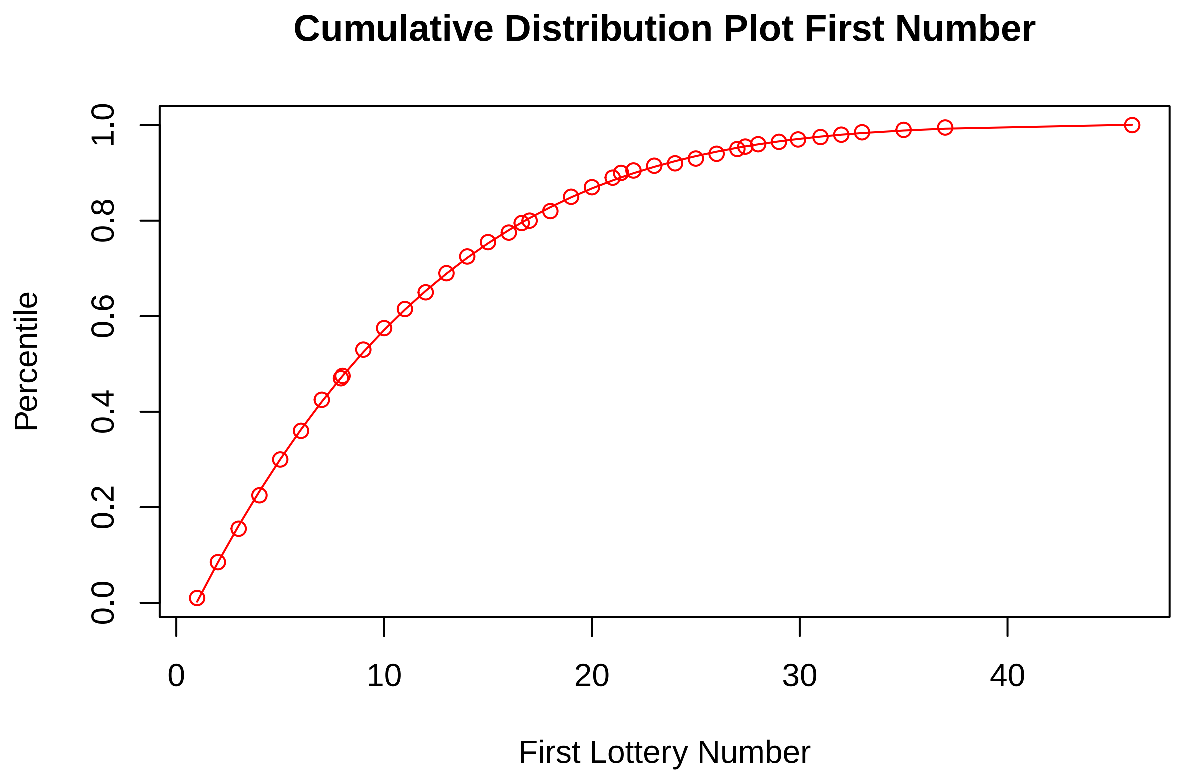
B4, B5, B6 might be removed depending on the Adjusted R2 value and large P-values for each section. With this method, we will be able to derive a CDF for each section, and the PDF (probability density function) is the derivative of the CDF. Note, these are only estimates of the true function, so there will be small errors when we compare our findings to the percentile data frame. We will examine this closely after getting our regression equation.

```
dataA$A2 <- dataA$A^2
dataA$A3 <- dataA$A^3
dataA$A4 <- dataA$A^4
dataA$A5 <- dataA$A^5

regression_A <- lm(percentiles ~ A+A2+A3+A4, data = dataA)
summary(regression_A)
```

```
##
## Call:
## lm(formula = percentiles ~ A + A2 + A3 + A4, data = dataA)
##
```

```
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0082848 -0.0026499  0.0001049  0.0024683  0.0095654
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.546e-02  3.732e-03 -22.902  < 2e-16 ***
## A            9.057e-02  1.107e-03  81.855  < 2e-16 ***
## A2          -2.883e-03  9.782e-05 -29.477  < 2e-16 ***
## A3           4.133e-05  3.219e-06  12.839 8.42e-15 ***
## A4          -2.237e-07  3.473e-08  -6.442 2.03e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.004293 on 35 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9998
## F-statistic: 4.217e+04 on 4 and 35 DF,  p-value: < 2.2e-16
```

```
plot(dataA$A, dataA$percentiles, xlab="First Lottery Number", ylab="Percentile", main="Cumulative Distri
lines(sort(dataA$A),
      fitted(regression_A)[order(dataA$A)],
      col = "red",
      type = "l")
```



Let's perform the other regressions and plot the lines. The best degree was found in advance for each number so we are skipping the same steps as the first number regression.

```r
dataB$B2 <- dataB$B^2
dataB$B3 <- dataB$B^3
dataB$B4 <- dataB$B^4
dataB$B5 <- dataB$B^5

dataC$C2 <- dataC$C^2
dataC$C3 <- dataC$C^3
dataC$C4 <- dataC$C^4
dataC$C5 <- dataC$C^5

dataD$D2 <- dataD$D^2
dataD$D3 <- dataD$D^3
dataD$D4 <- dataD$D^4
dataD$D5 <- dataD$D^5

dataE$E2 <- dataE$E^2
dataE$E3 <- dataE$E^3
dataE$E4 <- dataE$E^4


regression_B <- lm(percentiles ~ B+B2+B3+B4, data = dataB)
summary(regression_B)
```

```
##
## Call:
## lm(formula = percentiles ~ B + B2 + B3 + B4, data = dataB)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0104504 -0.0037233  0.0002564  0.0031755  0.0125849
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.436e-02  6.344e-03  -5.416 2.42e-06 ***
## B            4.890e-03  1.354e-03   3.611 0.000777 ***
## B2           2.137e-03  8.947e-05  23.883  < 2e-16 ***
## B3          -5.889e-05  2.272e-06 -25.920  < 2e-16 ***
## B4           4.479e-07  1.942e-08  23.071  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005379 on 44 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9997
## F-statistic: 4.702e+04 on 4 and 44 DF,  p-value: < 2.2e-16
```

```r
regression_C <- lm(percentiles ~ C+C2+C3+C4, data = dataC)
summary(regression_C)
```

```
##
## Call:
## lm(formula = percentiles ~ C + C2 + C3 + C4, data = dataC)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0082045 -0.0029354 -0.0008143  0.0017026  0.0185729
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.966e-02  1.515e-02   5.918 3.13e-07 ***
## C           -2.305e-02  2.474e-03  -9.315 2.01e-12 ***
## C2           1.830e-03  1.343e-04  13.626  < 2e-16 ***
## C3          -2.185e-05  2.942e-06  -7.427 1.45e-09 ***
## C4           3.016e-08  2.241e-08   1.346    0.185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005013 on 49 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9998
## F-statistic: 6.441e+04 on 4 and 49 DF,  p-value: < 2.2e-16
```

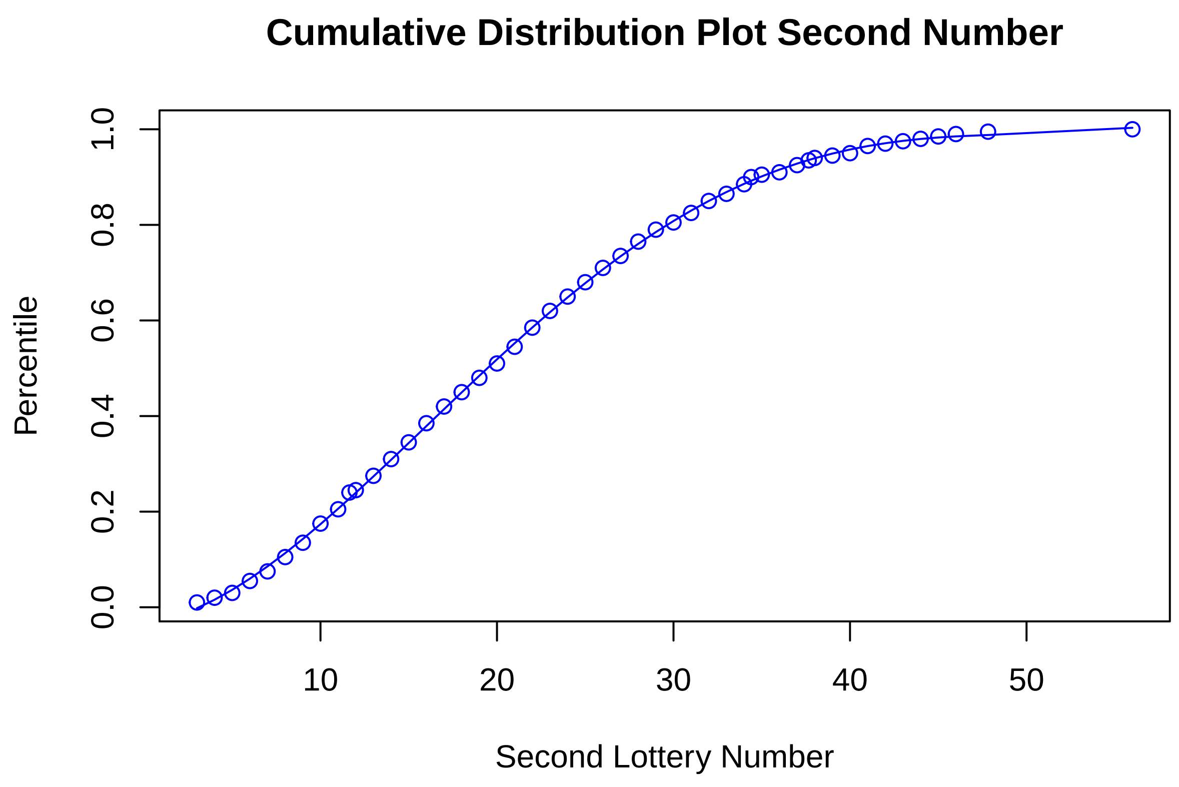
```
regression_D <- lm(percentiles ~ D+D2+D3+D4, data = dataD)
summary(regression_D)
```

```
##
## Call:
## lm(formula = percentiles ~ D + D2 + D3 + D4, data = dataD)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0077722 -0.0029086 -0.0009698  0.0028890  0.0233885
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.538e-01  4.132e-02  -6.143 1.77e-07 ***
## D            4.303e-02  5.407e-03   7.959 3.40e-10 ***
## D2          -2.646e-03  2.475e-04 -10.694 4.61e-14 ***
## D3           7.407e-05  4.734e-06  15.644  < 2e-16 ***
## D4          -6.017e-07  3.224e-08 -18.664  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005162 on 46 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9998
## F-statistic: 5.552e+04 on 4 and 46 DF,  p-value: < 2.2e-16
```

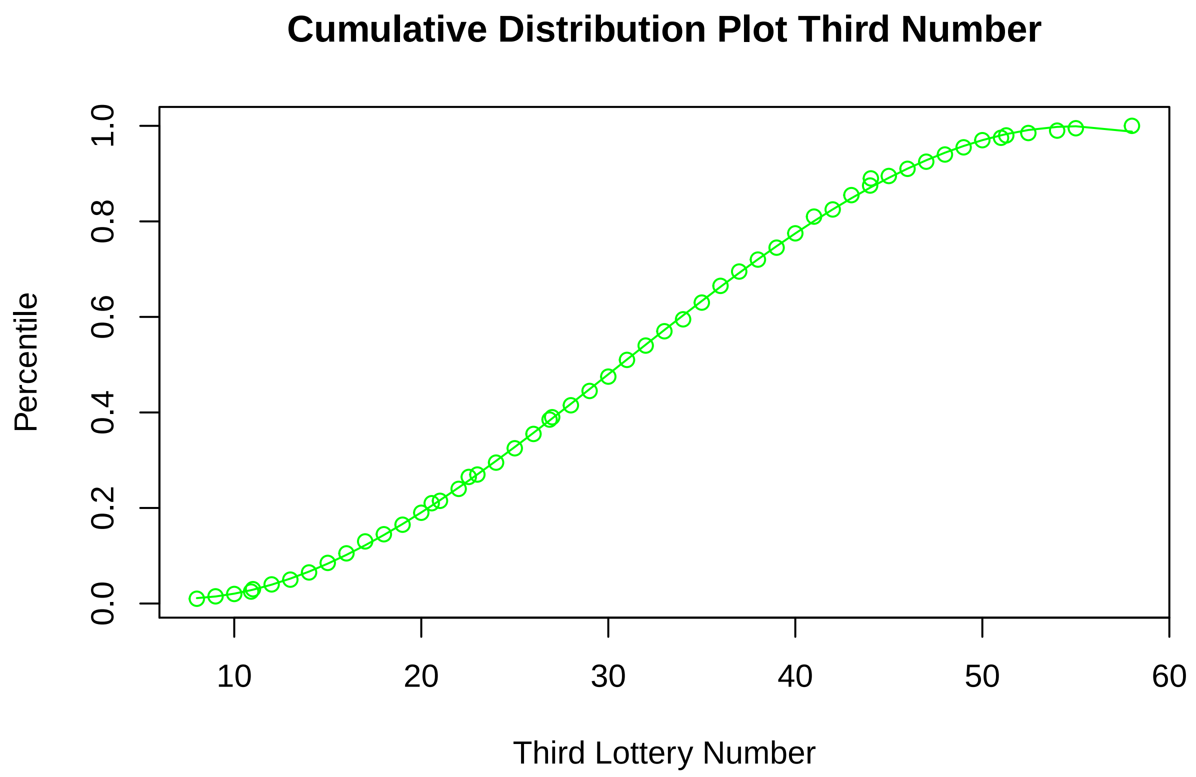
```
regression_E <- lm(percentiles ~ E+E2+E3+E4, data = dataE)
summary(regression_E)
```

```
##
## Call:
## lm(formula = percentiles ~ E + E2 + E3 + E4, data = dataE)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0095529 -0.0026410  0.0006065  0.0022716  0.0182431
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.003e+00  2.885e-01   3.478 0.001441 **
## E           -1.051e-01  2.917e-02  -3.604 0.001020 **
## E2           4.072e-03  1.077e-03   3.780 0.000626 ***
## E3          -7.146e-05  1.726e-05  -4.140 0.000226 ***
## E4           5.395e-07  1.014e-07   5.322 7.14e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.004924 on 33 degrees of freedom
## Multiple R-squared:  0.9997, Adjusted R-squared:  0.9996
## F-statistic: 2.587e+04 on 4 and 33 DF,  p-value: < 2.2e-16
```

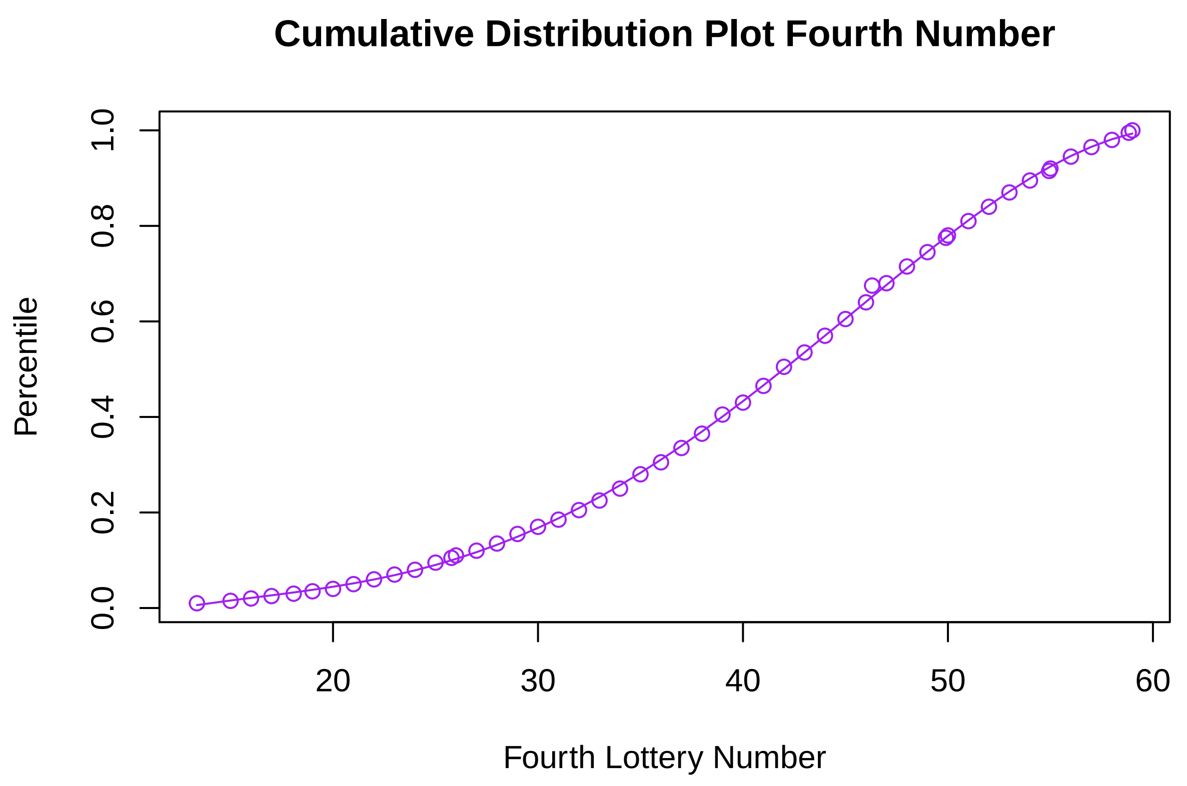
```
plot(dataB$B, dataB$percentiles, xlab="Second Lottery Number", ylab="Percentile", main="Cumulative Dist
lines(sort(dataB$B),
      fitted(regression_B)[order(dataB$B)],
      col = "blue",
      type = "l")
```



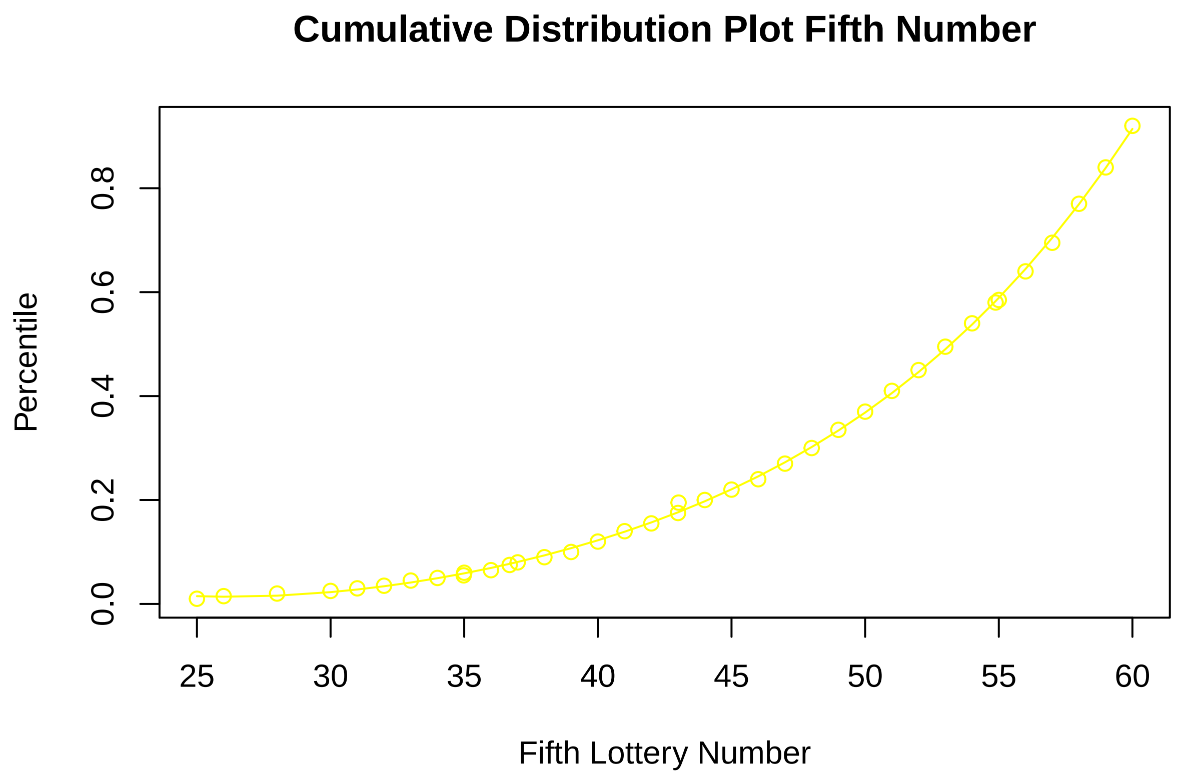
```
plot(dataC$C, dataC$percentiles, xlab="Third Lottery Number", ylab="Percentile", main="Cumulative Distr
lines(sort(dataC$C),
      fitted(regression_C)[order(dataC$C)],
      col = "green",
      type = "l")
```



```
plot(dataD$D, dataD$percentiles, xlab="Fourth Lottery Number", ylab="Percentile", main="Cumulative Dist
lines(sort(dataD$D),
      fitted(regression_D)[order(dataD$D)],
      col = "purple",
      type = "l")
```

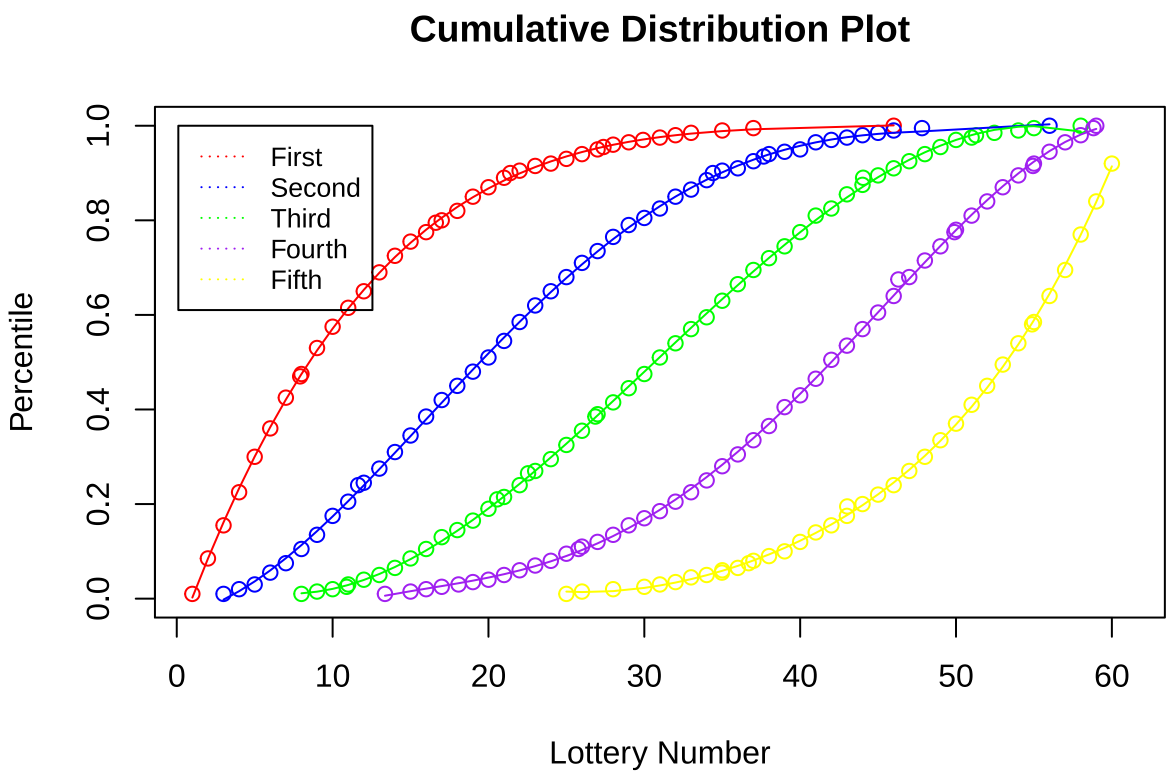


```
plot(dataE$E, dataE$percentiles, xlab="Fifth Lottery Number", ylab="Percentile", main="Cumulative Distri
lines(sort(dataE$E),
      fitted(regression_E)[order(dataE$E)],
      col = "yellow",
      type = "l")
```



```
plot(dataA$A, dataA$percentiles, xlab="Lottery Number", ylab="Percentile", main="Cumulative Distributio
points(dataB$B, dataB$percentiles, col="blue")
points(dataC$C, dataC$percentiles, col="green")
points(dataD$D, dataD$percentiles, col="purple")
points(dataE$E, dataE$percentiles, col="yellow")
lines(sort(dataA$A),
      fitted(regression_A)[order(dataA$A)],
      col = "red",
      type = "l")
lines(sort(dataB$B),
      fitted(regression_B)[order(dataB$B)],
      col = "blue",
      type = "l")
lines(sort(dataC$C),
      fitted(regression_C)[order(dataC$C)],
      col = "green",
      type = "l")
lines(sort(dataD$D),
      fitted(regression_D)[order(dataD$D)],
      col = "purple",
      type = "l")
lines(sort(dataE$E),
      fitted(regression_E)[order(dataE$E)],
      col = "yellow",
      type = "l")
```

```
legend(0.1, 1, legend=c("First", "Second", "Third", "Fourth", "Fifth"),
       col=c("red", "blue", "green", "purple", "yellow"), lty=3, cex=0.8)
```

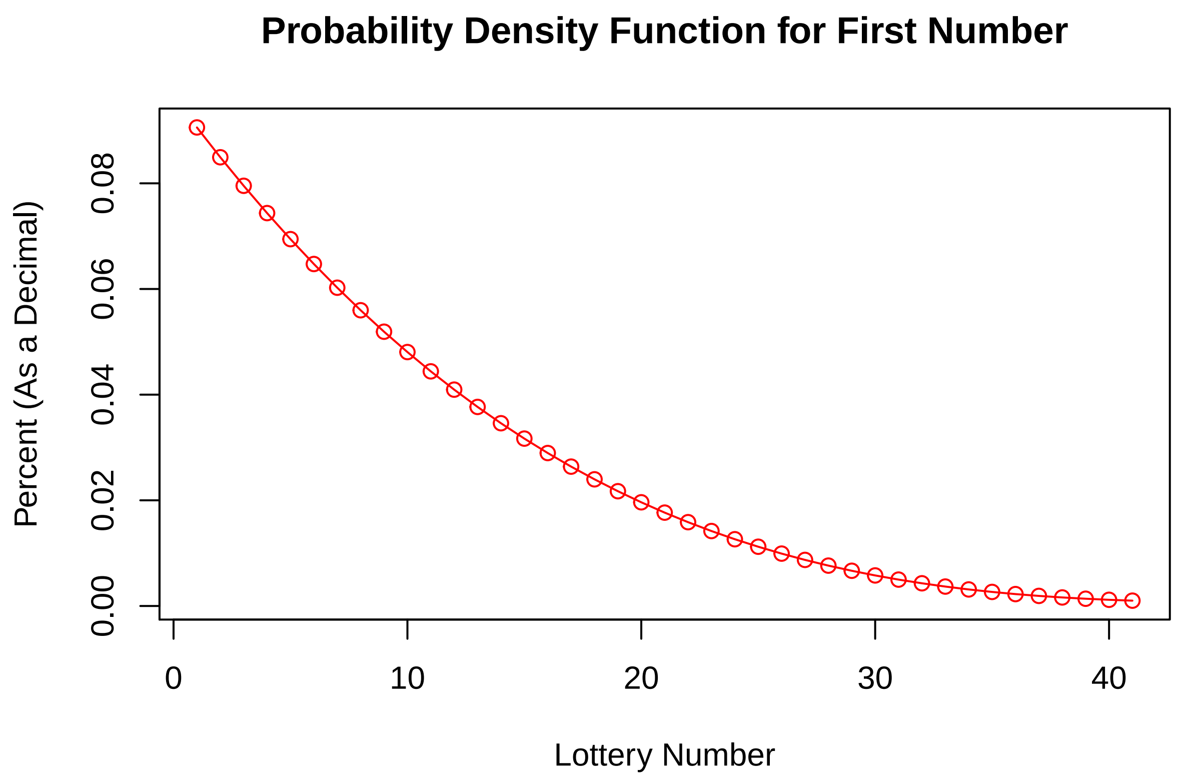


Now we want to derive the derivatives of our regression equations. Remember, the regression equations are best-fit estimates of cumulative distribution functions. The derivative will provide us an estimate of probability density functions, which will look similar to the respected histograms of each section.

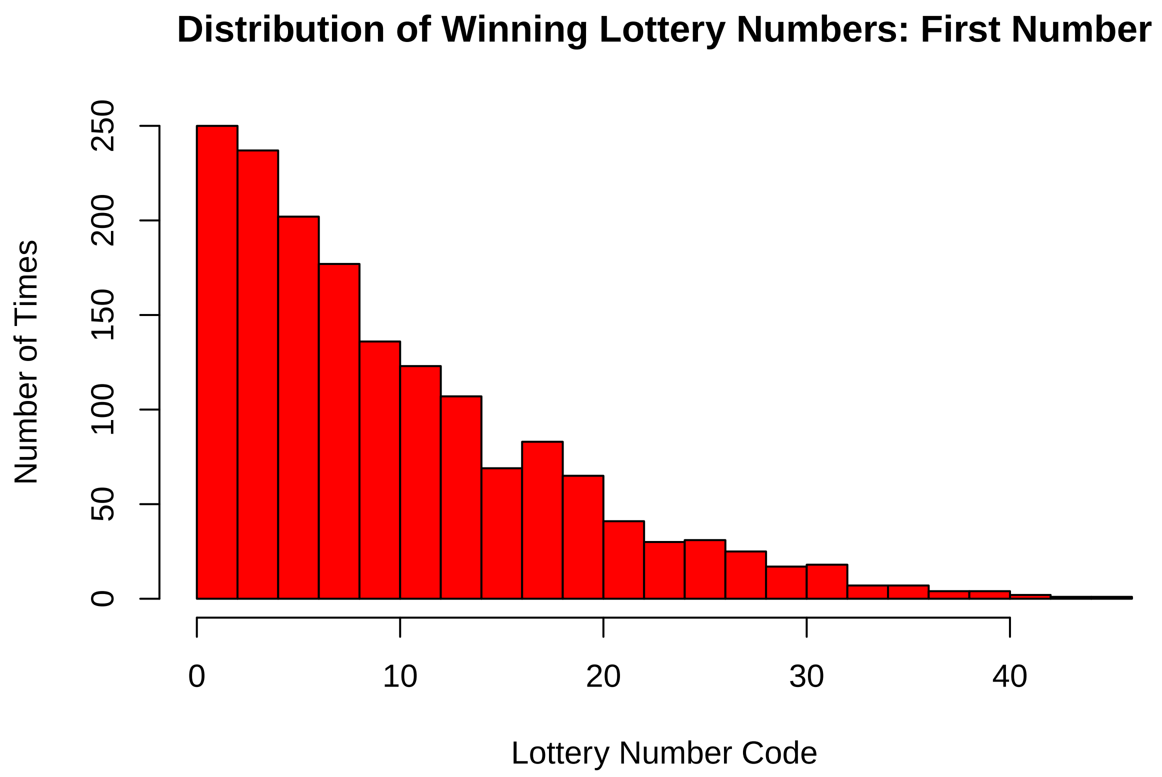
```
AB1 <- as.numeric(regression_A$coefficients[2])
AB2 <- as.numeric(regression_A$coefficients[3])
AB3 <- as.numeric(regression_A$coefficients[4])
AB4 <- as.numeric(regression_A$coefficients[5])




#Derivative of First Regression is AB1 + AB2*(2* x) + AB3*(3*x^2) + AB4*(4 * x^3). ABi represents the c

ap <- function(x){
  AB1 + AB2*(2 * x) + AB3*(3*x^2) + AB4*(4*x^3)
}

#Checking if the area under the PDF is approximately 1
integrate(ap, 1, max(dataA$A))
```

```
## 0.9985099 with absolute error < 1.1e-14
```

```
#Plot PDF and line
plot(ap(seq(0, 40, by=1)), main = "Probability Density Function for First Number", xlab = "Lottery Numbe
lines(ap(seq(0, 40, by=1)), col = "red")
```



```
#Compare with original histogram

hist(A, main="Distribution of Winning Lottery Numbers: First Number", col='red', xlab= "Lottery Number (
```



Using the function we just derived from the regression equation, we can figure out what percent of numbers show up in the winning lottery numbers for each section. We will try some example problems, and compare three methods:

Integrating the Probability Density Function (Finding the Area Under the Curve).

Subtracting Percentiles from Data Frame.

Counting and Dividing (True Answer).

```
##Q1 What percent of winning lottery numbers have numbers 1 to 5 in the first section
#Percentiles

View(pdata)

#Answer: 35.5%

#Integrating PDF function

integrate(ap, 1, 6)
```

```
## 0.3605425 with absolute error < 4e-15
```

```
#Answer: 36%

#Counting Method

sum(data$A > 0.9 & data$A < 5.1)/sum(data$A > 0)
```

```
## [1] 0.3585828
```

```
#True Answer: 35.85828%

##Q2 What percent of winning lottery numbers have numbers 5 to 8 in the first section
#Percentiles Method

0.525 - 0.295
```

```
## [1] 0.23
```

```
#Answer: 23%

#Integrating PDF Function

integrate(ap, 5, 9)
```

```
## 0.2244556 with absolute error < 2.5e-15
```

```
#Answer: 22.4%

#Counting Method

sum(data$A > 4.9 & data$A < 8.1)/sum(data$A > 0)
```

```
## [1] 0.2315211
```

```
#True Answer: 23%

##Q3 What percent of winning lottery numbers have numbers 1, 3 and 5 in the first section
#Percentiles Method

0.08 + (0.220-0.150) + (0.355-0.295)
```

```
## [1] 0.21
```

```
#Answer: 21%

#Integrating PDF Function

integrate(ap, 1, 2)
```

```
## 0.08220907 with absolute error < 9.1e-16
```

```
0.08220907
```

```
## [1] 0.08220907
```

```
integrate(ap, 3, 4)
```

```
## 0.07187931 with absolute error < 8e-16
```

```
0.07187931
```

```
## [1] 0.07187931
```

```
integrate(ap, 5, 6)
```

```
## 0.06246633 with absolute error < 6.9e-16
```

```
0.06246633
```

```
## [1] 0.06246633
```

```
0.08220907+0.07187931+0.06246633
```

```
## [1] 0.2165547
```

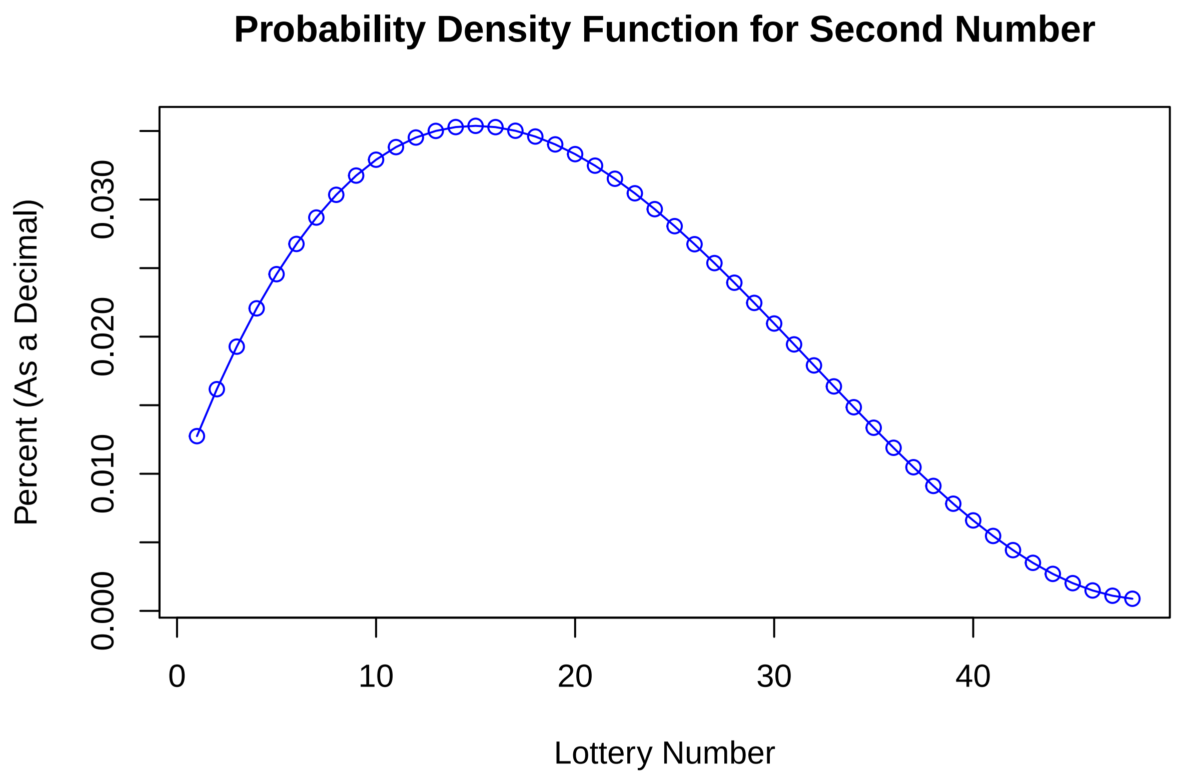
```
#Answer: 21.6%

#Counting Method

(sum(data$A > 0.9 & data$A < 1.9) + sum(data$A > 2.9 & data$A < 3.9) + sum(data$A > 4.9 & data$A < 5.9)
```

```
## [1] 0.2113622
```

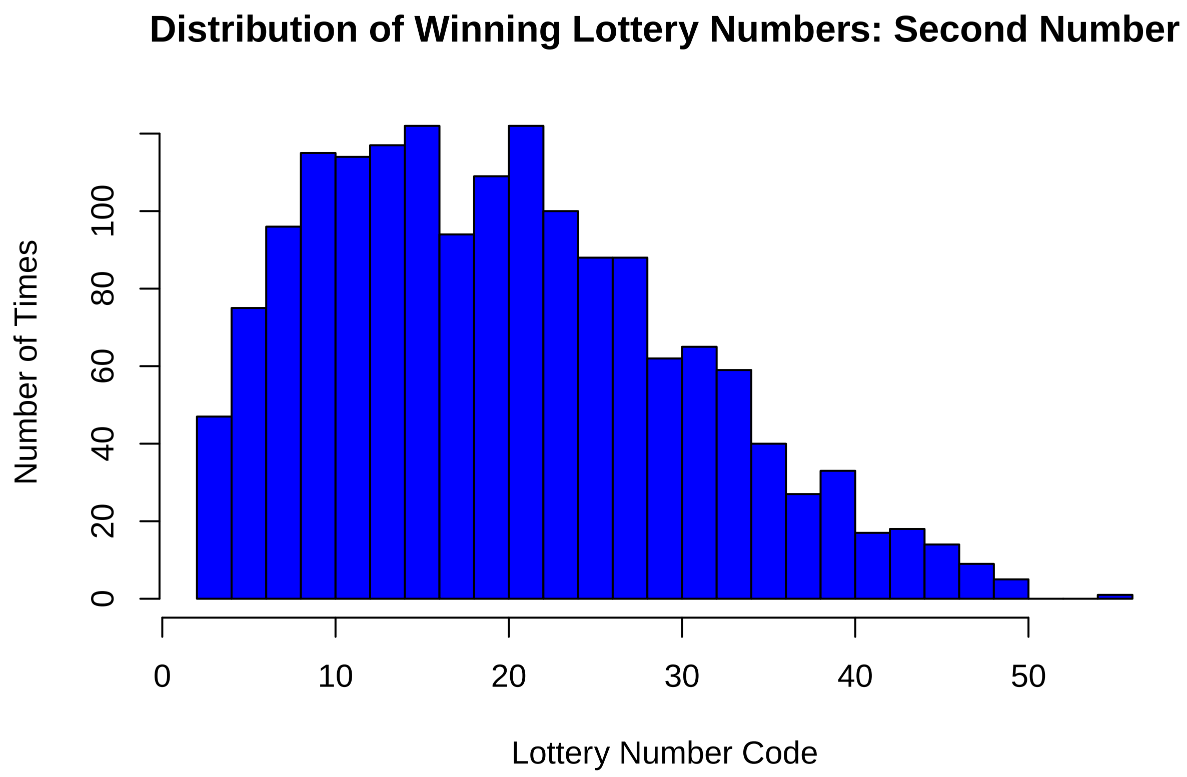
```
#True Answer: 21.13622%
```

We can see that each method provides answers close to each other with a small error difference. It's important to note that if we generated more percentiles, our answers would be more accurate to the true value, which can be derived by counting the number of numbers that satisfy the condition and dividing by the sample size. Let's gather the rest of the probability density functions.

```
BB1 <- as.numeric(regression_B$coefficients[2])
BB2 <- as.numeric(regression_B$coefficients[3])
BB3 <- as.numeric(regression_B$coefficients[4])
BB4 <- as.numeric(regression_B$coefficients[5])


bp <- function(x){
  BB1 + BB2*(2*x) + BB3*(3*x^2) + BB4*(4*x^3)
}

plot(bp(seq(min(data$B), 49, by=1)), main = "Probability Density Function for Second Number", xlab = "L
lines(bp(seq(min(data$B), 49, by=1)), col = "blue")
```

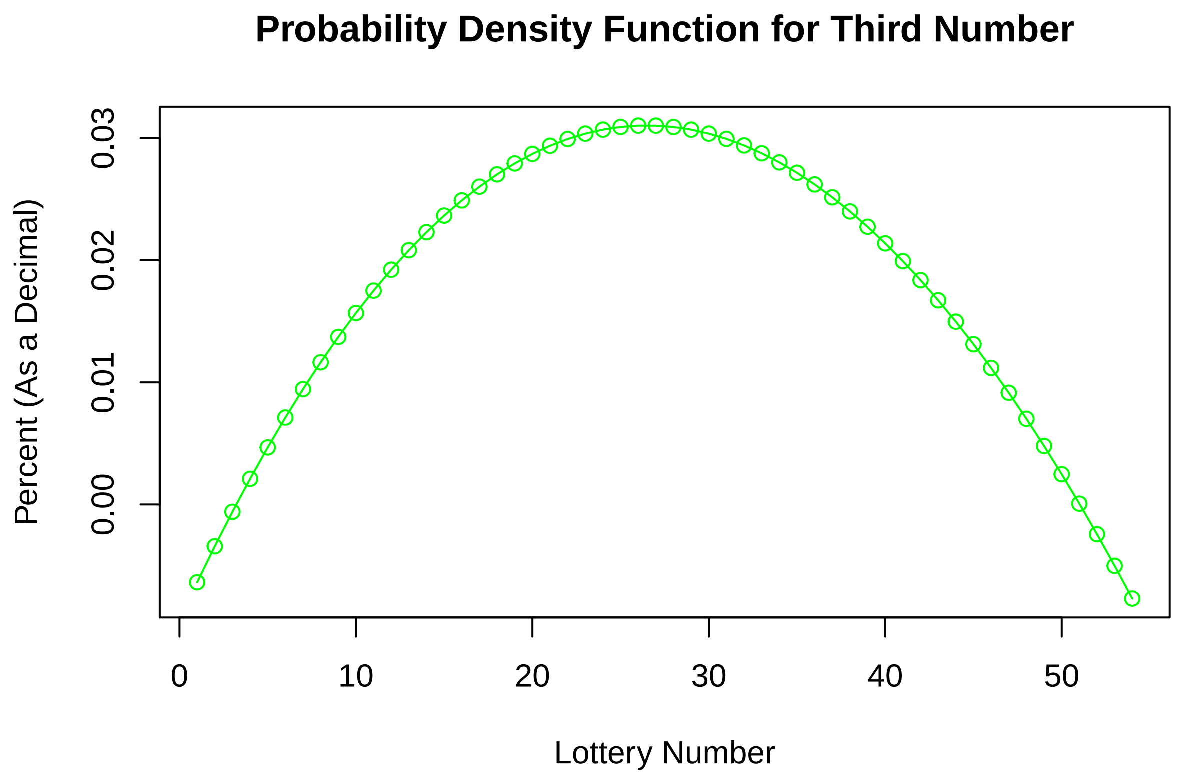


```
hist(B, main="Distribution of Winning Lottery Numbers: Second Number", col='blue', xlab= "Lottery Numbe
```

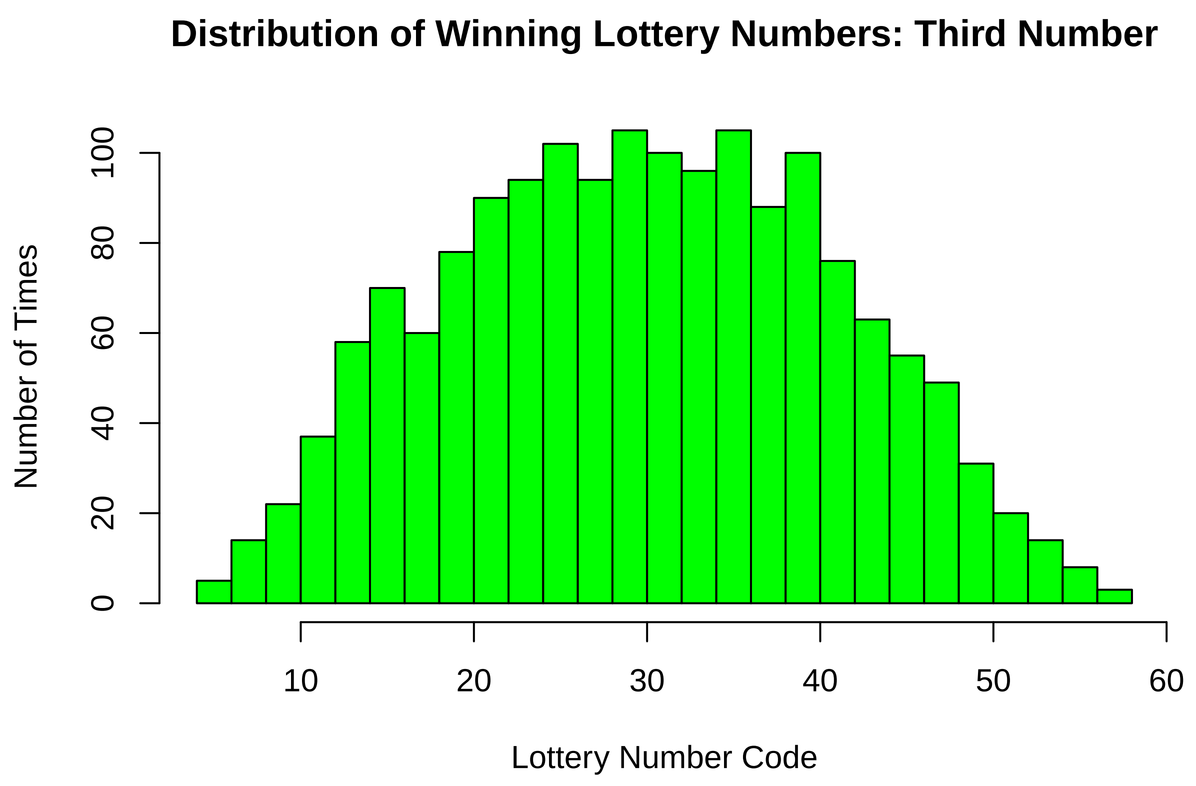


```
#Outlier 56 is removed
```

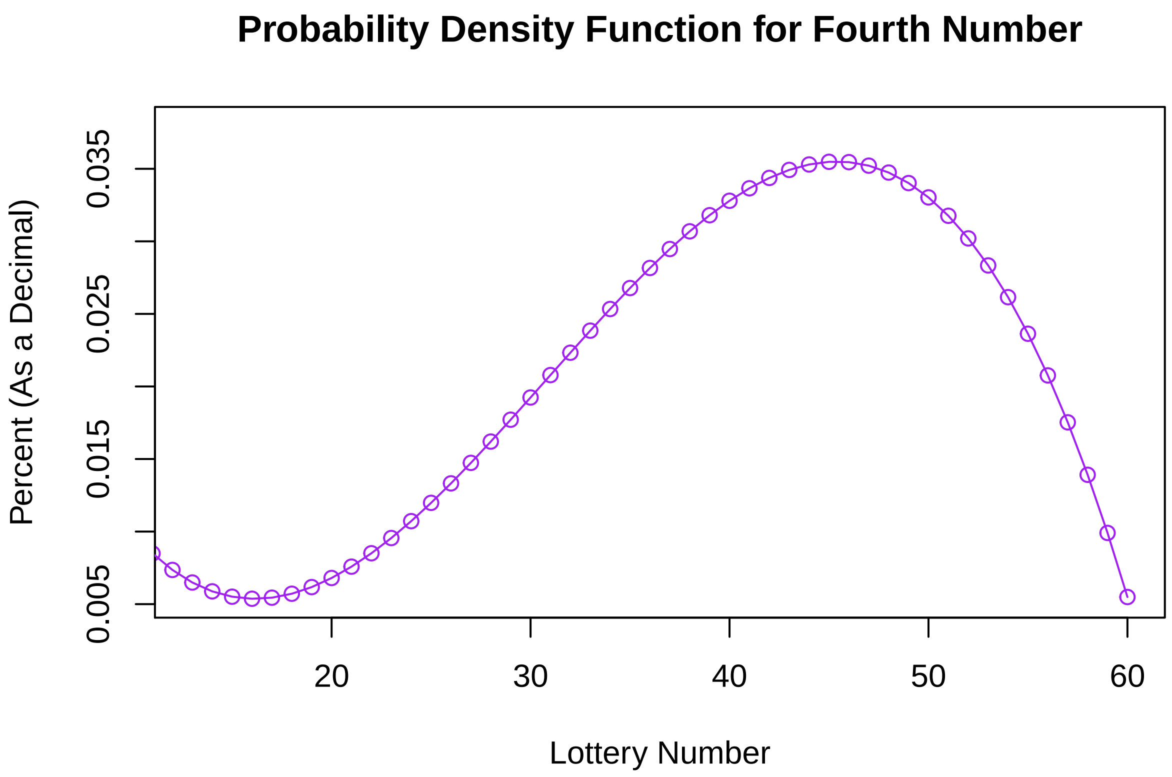
```
CB1 <- as.numeric(regression_C$coefficients[2])
CB2 <- as.numeric(regression_C$coefficients[3])
CB3 <- as.numeric(regression_C$coefficients[4])
CB4 <- as.numeric(regression_C$coefficients[5])


cp <- function(x){
  CB1 + CB2*(2*x) + CB3*(3*x^2) + CB4*(4*x^3)
}

plot(cp(seq(min(data$C), max(data$C), by=1)), main = "Probability Density Function for Third Number", x
lines(cp(seq(min(data$C), max(data$C), by=1)), col = "green")
```



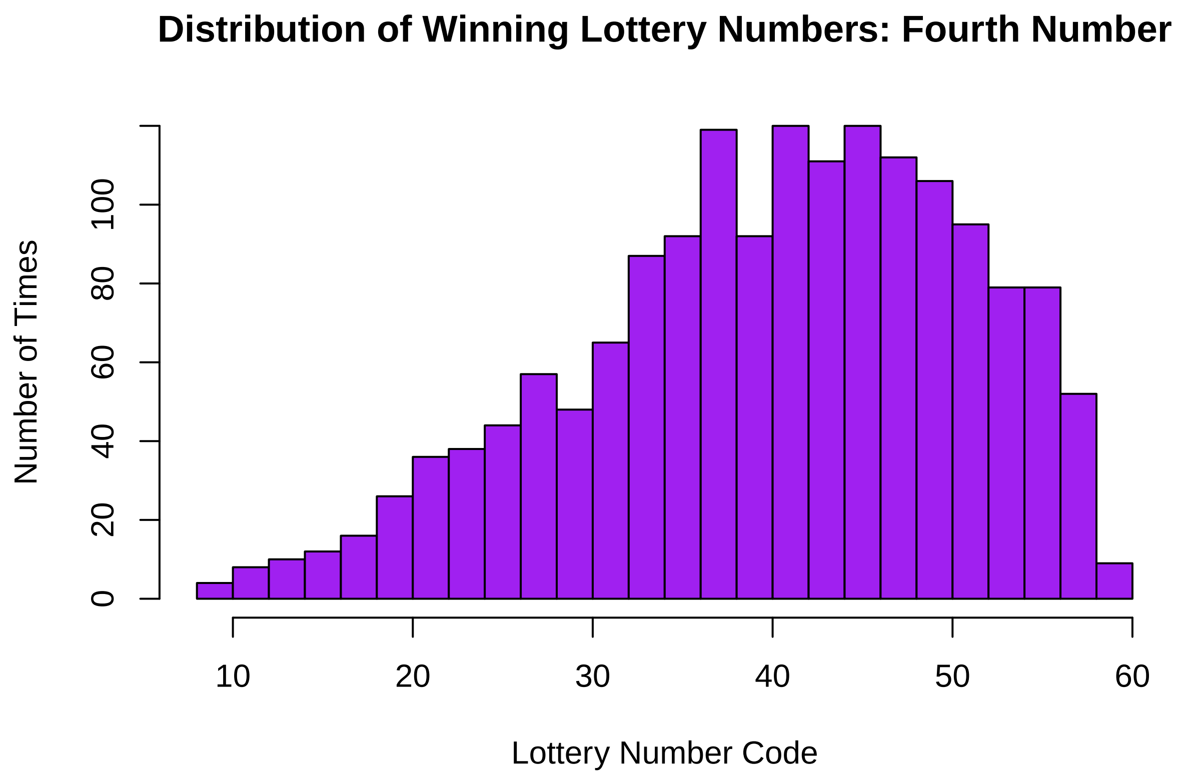
```
hist(C, main="Distribution of Winning Lottery Numbers: Third Number", col='green', xlab= "Lottery Numbe
```



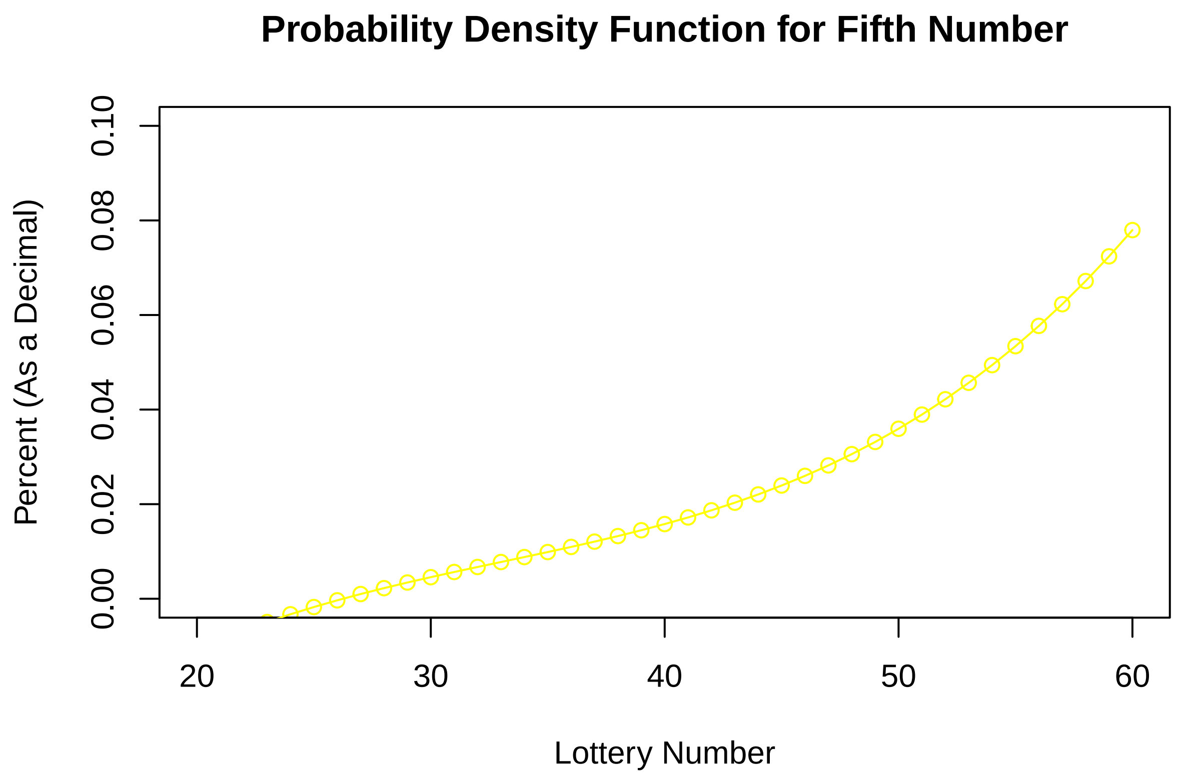
```r
DB1 <- as.numeric(regression_D$coefficients[2])
DB2 <- as.numeric(regression_D$coefficients[3])
DB3 <- as.numeric(regression_D$coefficients[4])
DB4 <- as.numeric(regression_D$coefficients[5])


dp <- function(x){
  DB1 + DB2*(2*x) + DB3*(3*x^2) + DB4*(4*x^3)
}

plot(dp(seq(1, 60, by=1)), main = "Probability Density Function for Fourth Number", xlab = "Lottery Numl
lines(dp(seq(1, 60, by=1)), col = "purple")
```



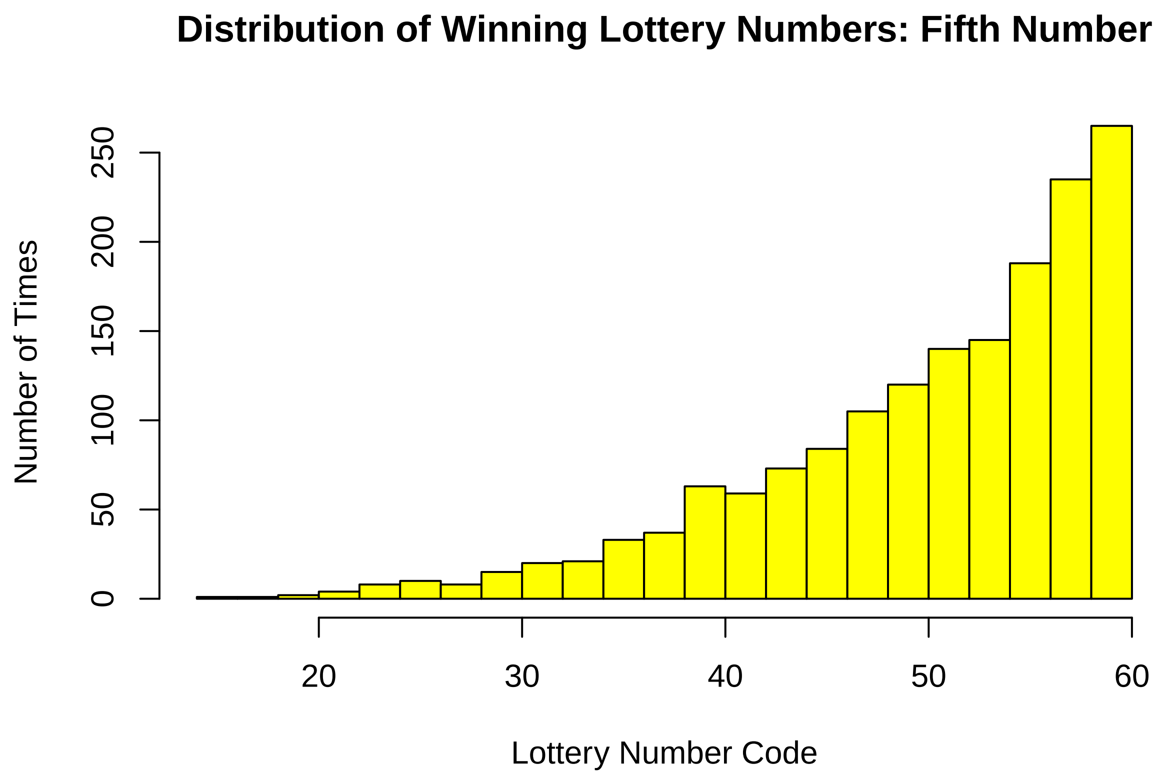
```
hist(D, main="Distribution of Winning Lottery Numbers: Fourth Number", col='purple', xlab= "Lottery Numb
```



```
EB1 <- as.numeric(regression_E$coefficients[2])
EB2 <- as.numeric(regression_E$coefficients[3])
EB3 <- as.numeric(regression_E$coefficients[4])
EB4 <- as.numeric(regression_E$coefficients[5])


ep <- function(x){
  EB1 + EB2*(2*x) + EB3*(3*x^2) + EB4*(4*x^3)
}

plot(ep(seq(1, 60, by=1)), main = "Probability Density Function for Fifth Number", xlab = "Lottery Numbe
lines(ep(seq(1, 60, by=1)), col = "yellow")
```



```
hist(E, main="Distribution of Winning Lottery Numbers: Fifth Number", col='yellow', xlab= "Lottery Numbe
```



We now have estimates for the probability density functions for each section of the winning lottery numbers, and we can see that each follows a skew-normal distribution of some degree. The only exceptions are the cash ball, holding random nature of a uniform distribution. The functions we have derived allow us to estimate further and answer questions that require convolution of probability distributions, such as the probability that a winning number will be 1 10 20 30 40 50. While in a presumed random setting, this would easy to solve by simply dividing the number of combinations over the total, we can not do that with the case of the Cash4Life lottery. It does not follow a uniform distribution, which people would assume. Another example that we can prove using the functions, is the combination 1 10 20 30 40 50 is more likely to appear than the combination 60 60 60 60 60. Through the visualizations, we can see why this is the case. The number 60, has almost 0% chance of showing up in the first two sections of the lottery number. It's important to note that due to the nature of linear/polynomial regressions, probability functions have a restricted range. We can not use the first section function to figure out the percent chance that the numbers 50-60 will show up, since those numbers are not in the percentile range that we regressed on. With more data after clean up, we can correlate these findings to sales, purchases, and distribution of numbers of tickets played.

This project uses real-life data to present a lax demonstration of how we can use polynomial regressions to interpolate cumulative distribution functions, allowing us to gain rough estimates of probability density functions. I'm researching this concept, along with advanced probability topics, in the applications of income mobility, government data, social networks, and financial markets.