Assignment 06

Zhen Liu

2022-11-11

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

Contunuous random variables and limit laws

Simulating data with the uniform distribution

Q1

$$p_U(x) = \begin{cases} 1 & x \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$

$$P(X \in [a,b]) = \int_a^b p_U(x)dx = 1 * \text{length}([a,b] \cap [0,1]) = b - a$$

Q2

```
set.seed(0)
n <- 1000
sample_X <- data.frame(U=runif(n)) %>%
  mutate(X=case_when(
    (0<=U)&(U<0.25)~3,
    (0.25<=U)&(U<0.5)~10,
    (0.5<=U)&(U<=1)~0)) %>%
  pull(X)
```

Because the sample follows the distributions of Gaussian random variable, which is also called distribution

Q3

```
sample_X_0310<-function(a,b,n){
n <- n
sample_X_0310 <- data.frame(U=runif(n)) %>%
  mutate(X=case_when(
    (0<=U)&(U<a)~3,
    (a<=U)&(U<(a+b))~10,
    ((a+b)<=U)&(U<=1)~0))%>%
  pull(X)
}
```

Q4

```
sample_X_0310(0.5,0.1,10000)
```

```
average<- mean(sample_X_0310(0.5,0.1,10000))
average
```

```
## [1] 2.4775
```

Based on assignment 5 section 2.1, the value of Expectation X is 3*0.5 + 0.1*10 =2.5

The law if large numbers tells us that the sample average converges towards the expectation, for sequences of independent and identically distributed random varaibles.

Q5

```
var(sample_X_0310(0.5,0.1,10000))
```

```
## [1] 8.164148
```

Var(X) is: 9*0.5 +100*0.1-9*0.5*0.5-100*0.1*0.1-60*0.5*0.1 = 8.25

Q6

1.

```
beta<-seq(0.01,0.9,length = 100)

class(beta)
```

```
## [1] "numeric"
```

2.

```
sample_X_0310_new<-function(b){
n <- 100
sample_X_0310 <- data.frame(U=runif(n)) %>%
  mutate(X=case_when(
    (0<=U)&(U<0.1)~3,
    (0.1<=U)&(U<(0.1+b))~10,
    ((0.1+b)<=U)&(U<=1)~0))%>%
  pull(X)
}
```

```
Expectation<-function(b){
  return(mean(3*0.1 + b*10))
}
```

```
data_frame_X<-data.frame(beta)
```

```
data_frame_X%>%
  mutate(samplemean=map_dbl(.x=beta,.f=~mean(sample_X_0310_new(b=.x))))%>%
  mutate(Expectation=map_dbl(.x=beta,.f=~Expectation(b=.x)))
```

```
##          beta samplemean Expectation
## 1  0.01000000       0.44   0.4000000
## 2  0.01898990       0.33   0.4898990
## 3  0.02797980       0.83   0.5797980
## 4  0.03696970       0.86   0.6696970
## 5  0.04595960       0.56   0.7595960
## 6  0.05494949       0.76   0.8494949
## 7  0.06393939       0.89   0.9393939
## 8  0.07292929       1.26   1.0292929
## 9  0.08191919       1.24   1.1191919
## 10 0.09090909       1.04   1.2090909
## 11 0.09989899       1.46   1.2989899
## 12 0.10888889       0.99   1.3888889
## 13 0.11787879       1.16   1.4787879
## 14 0.12686869       1.41   1.5686869
## 15 0.13585859       1.99   1.6585859
## 16 0.14484848       1.27   1.7484848
## 17 0.15383838       1.98   1.8383838
## 18 0.16282828       1.86   1.9282828
## 19 0.17181818       2.17   2.0181818
## 20 0.18080808       1.87   2.1080808
## 21 0.18979798       2.13   2.1979798
## 22 0.19878788       2.77   2.2878788
## 23 0.20777778       3.11   2.3777778
## 24 0.21676768       2.07   2.4676768
## 25 0.22575758       2.60   2.5575758
## 26 0.23474747       1.78   2.6474747
## 27 0.24373737       2.53   2.7373737
## 28 0.25272727       2.53   2.8272727
## 29 0.26171717       2.28   2.9171717
## 30 0.27070707       2.80   3.0070707
## 31 0.27969697       2.90   3.0969697
## 32 0.28868687       3.83   3.1868687
## 33 0.29767677       2.70   3.2767677
## 34 0.30666667       2.83   3.3666667
## 35 0.31565657       3.00   3.4565657
## 36 0.32464646       2.96   3.5464646
## 37 0.33363636       3.07   3.6363636
## 38 0.34262626       4.36   3.7262626
## 39 0.35161616       4.54   3.8161616
## 40 0.36060606       4.03   3.9060606
## 41 0.36959596       4.08   3.9959596
## 42 0.37858586       3.53   4.0858586
## 43 0.38757576       4.04   4.1757576
## 44 0.39656566       3.53   4.2656566
## 45 0.40555556       4.33   4.3555556
## 46 0.41454545       4.50   4.4454545
## 47 0.42353535       3.99   4.5353535
## 48 0.43252525       4.74   4.6252525
## 49 0.44151515       5.38   4.7151515
## 50 0.45050505       4.69   4.8050505
## 51 0.45949495       4.50   4.8949495
## 52 0.46848485       4.70   4.9848485
## 53 0.47747475       4.85   5.0747475
## 54 0.48646465       6.04   5.1646465
```

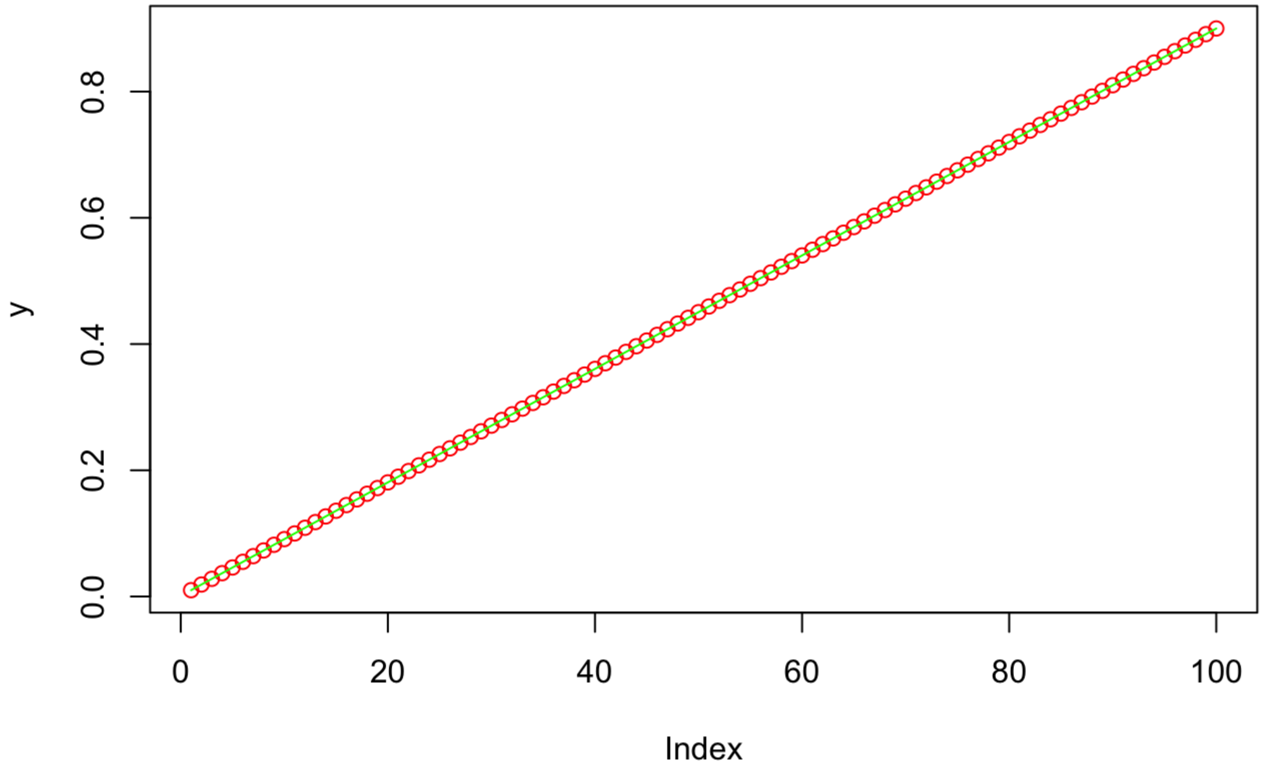
```
## 55  0.49545455       5.68   5.2545455
## 56  0.50444444       4.97   5.3444444
## 57  0.51343434       5.62   5.4343434
## 58  0.52242424       5.98   5.5242424
## 59  0.53141414       5.89   5.6141414
## 60  0.54040404       6.39   5.7040404
## 61  0.54939394       5.24   5.7939394
## 62  0.55838384       6.30   5.8838384
## 63  0.56737374       6.51   5.9737374
## 64  0.57636364       5.94   6.0636364
## 65  0.58535354       6.04   6.1535354
## 66  0.59434343       6.59   6.2434343
## 67  0.60333333       6.37   6.3333333
## 68  0.61232323       6.57   6.4232323
## 69  0.62131313       6.85   6.5131313
## 70  0.63030303       6.77   6.6030303
## 71  0.63929293       6.72   6.6929293
## 72  0.64828283       6.42   6.7828283
## 73  0.65727273       6.86   6.8727273
## 74  0.66626263       6.95   6.9626263
## 75  0.67525253       6.95   7.0525253
## 76  0.68424242       7.13   7.1424242
## 77  0.69323232       7.60   7.2323232
## 78  0.70222222       6.77   7.3222222
## 79  0.71121212       7.26   7.4121212
## 80  0.72020202       7.37   7.5020202
## 81  0.72919192       6.94   7.5919192
## 82  0.73818182       7.26   7.6818182
## 83  0.74717172       8.04   7.7717172
## 84  0.75616162       8.03   7.8616162
## 85  0.76515152       8.24   7.9515152
## 86  0.77414141       7.97   8.0414141
## 87  0.78313131       7.25   8.1313131
## 88  0.79212121       7.80   8.2212121
## 89  0.80111111       8.32   8.3111111
## 90  0.81010101       8.76   8.4010101
## 91  0.81909091       8.51   8.4909091
## 92  0.82808081       8.59   8.5808081
## 93  0.83707071       8.73   8.6707071
## 94  0.84606061       8.88   8.7606061
## 95  0.85505051       9.00   8.8505051
## 96  0.86404040       9.01   8.9404040
## 97  0.87303030       9.17   9.0303030
## 98  0.88202020       8.80   9.1202020
## 99  0.89101010       9.20   9.2101010
## 100 0.90000000       9.51   9.3000000
```

Q7

```
library(ggplot2)
```

```
x<-data_frame_X$beta
y<-data_frame_X$samplemean
y_1<-data_frame_X$Expectation

plot(x,y,col="red")
lines(x,y_1,col="green")
```



Exponential distribution

Q1

$$P_\lambda(x) = \int_{-\infty}^{\infty} p_\lambda(x)dx = 1 - 0 = 1$$

Q2

```
my_cdf_exp<-function(x,lambda){
  if(x<0){
  return(0)
  }
  else{
    return(1-10^-(lambda*x))
  }
}
```

```
lambda <- 1/2
map_dbl(.x=seq(-1,4), .f=~my_cdf_exp(x=.x,lambda=lambda) )
```

```
## [1] 0.0000000 0.0000000 0.6837722 0.9000000 0.9683772 0.9900000
```

```
test_inputs <- seq(-1,10,0.1)
my_cdf_output <- map_dbl(.x=test_inputs, .f=~my_cdf_exp(x=.x,lambda=lambda))
```

Q3

```
my_quantile_exp<-function(x,lambda){

  y<-1/my_cdf_exp(x,lambda)

}
```

```
test_inputs <- seq(0.01,0.99,0.01)
my_cdf_output <- map_dbl(.x=test_inputs, .f=~my_quantile_exp(x=.x,lambda=lambda))
```

Q4

The mean of population is ?

The Binomial distribution and the central limit theorem

Q1

$$P(Z = z) = p^z * (1 - p)^(1 - z), z = 0, 1$$

$$E(Z) = \sum_{z=0}^{1} zP(z) = \sum_{z=0}^{1} zP^z(1 - p)^(1 - z) = 0 + p = p$$

$$E(Z^2) = \sum_{z=0}^{1} z^2P(z) = \sum_{z=0}^{1} z^2p^z(1 - p)^(1 - z) = 0 + p = p$$

$$Var(Z) = E(Z^2) - (E(Z))^2 = p - p^2$$

Q2

1.

```
x <- seq(0,50,by =1)

pmf<- dbinom(x,size = 50,prob = 0.7)

binom_df<-data.frame(x,pmf)
```

2.

```
head(binom_df,3)
```

```
##   x          pmf
## 1 0 7.178980e-27
## 2 1 8.375477e-25
## 3 2 4.787981e-23
```

Q3

```
x<-seq(0,50,by =0.01)

pdf<-dnorm(x,mean = 35,sd = 3.24)

gaussian_df<-data.frame(x,pdf)

head(gaussian_df)
```

```
##      x          pdf
## 1 0.00 5.632852e-27
## 2 0.01 5.823795e-27
## 3 0.02 6.021153e-27
## 4 0.03 6.225140e-27
## 5 0.04 6.435976e-27
## 6 0.05 6.653890e-27
```

Q4

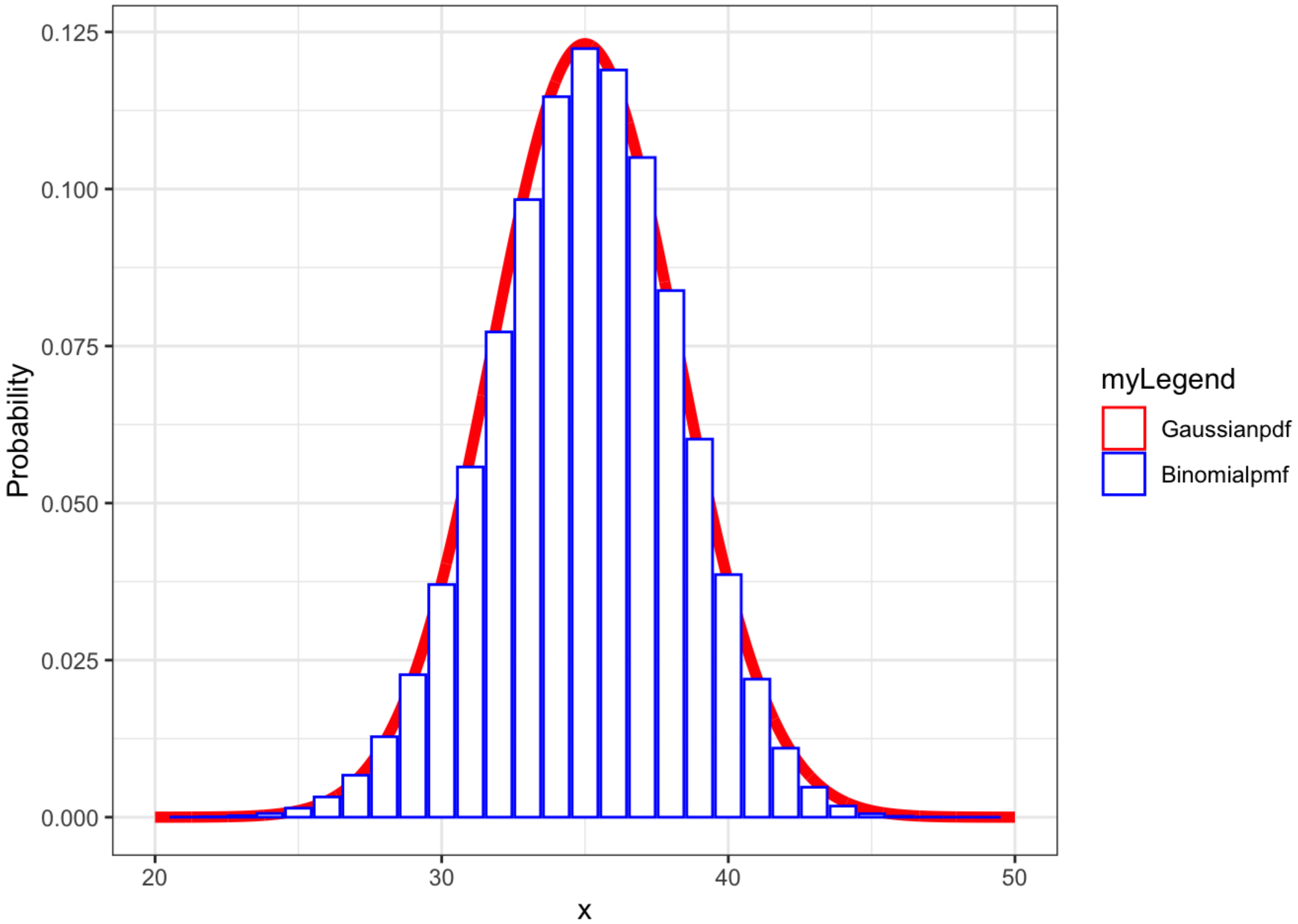
```
colors<-c("Gaussianpdf"="red","Binomialpmf"="blue")
fill<-c("Gaussianpdf"="white","Binomialpmf"="white")


ggplot()+labs(x="x",y="Probability")+theme_bw()+
  geom_line(data=gaussian_df,aes(x,y=pdf,color="Gaussianpdf"),size=2)+
  geom_col(data=binom_df,aes(x=x,y=pmf,color="Binomialpmf",fill="Binomialpmf"))+
  scale_color_manual(name="myLegend",values=colors)+ scale_fill_manual(name="myLegen
d",values=fill)+ xlim(c(20,50))
```

```
## Warning: Removed 20 rows containing missing values (position_stack).
```

```
## Warning: Removed 2000 row(s) containing missing values (geom_path).
```

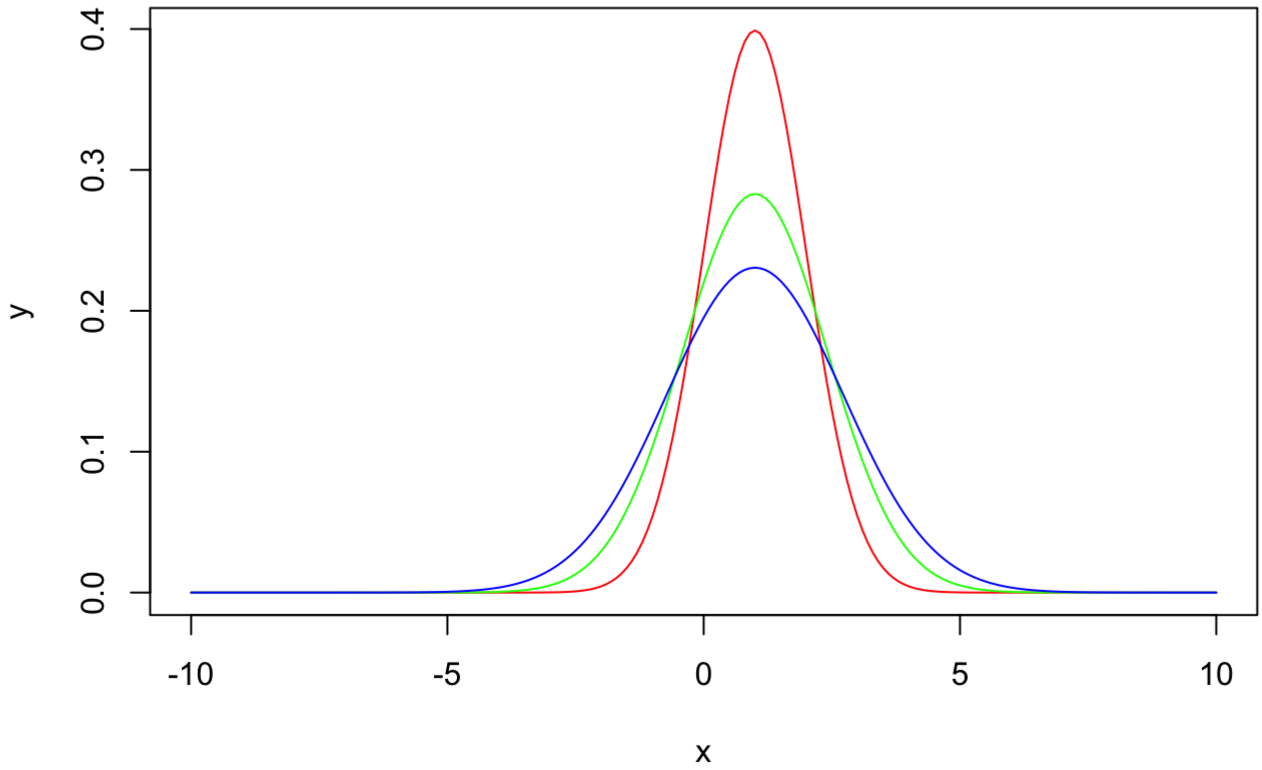
```
## Warning: Removed 2 rows containing missing values (geom_col).
```



The Guassian distribution

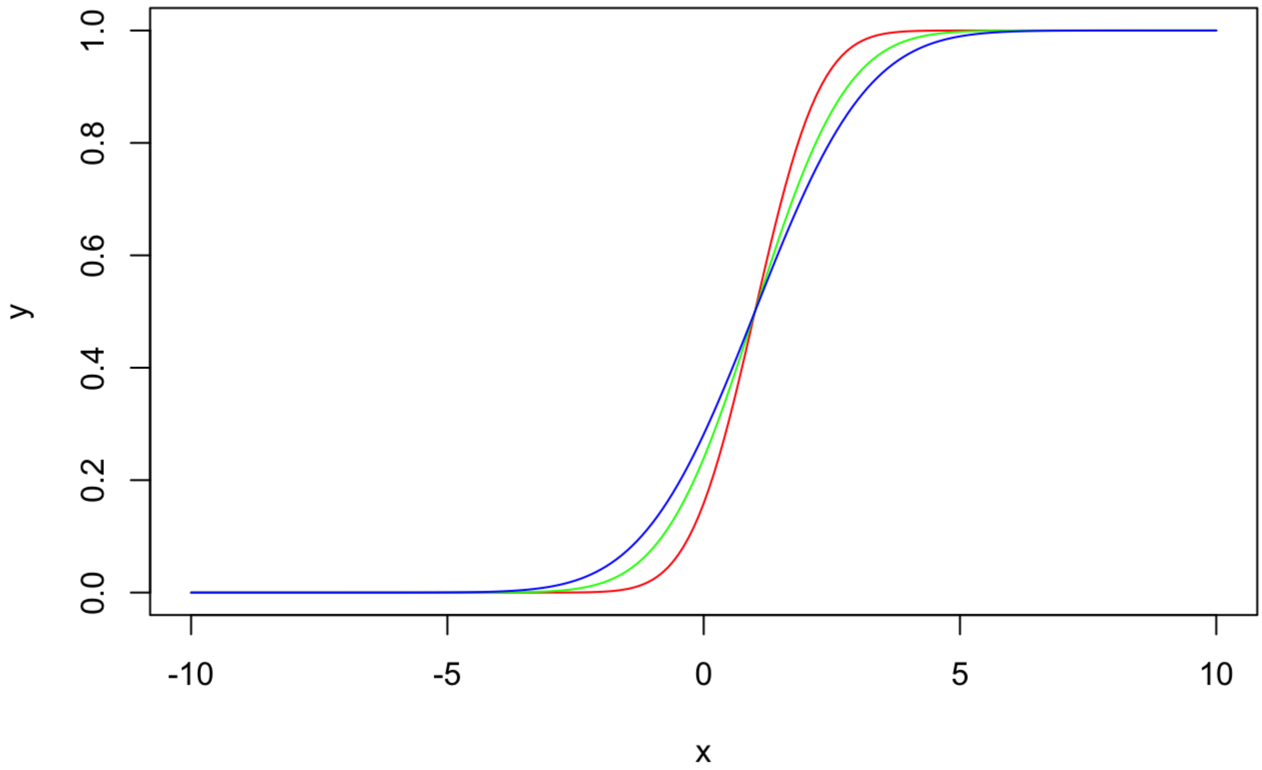
Q1

```
x <- seq(-10, 10, by = .1)
y <- dnorm(x, mean=1, sd=1)
y_1<-dnorm(x, mean=1, sd=1.41)
y_2<-dnorm(x, mean=1, sd=1.73)
plot(x, y,type = 'l',col='red')
lines(x, y_1,col='green')
lines(x, y_2,col='blue')
```



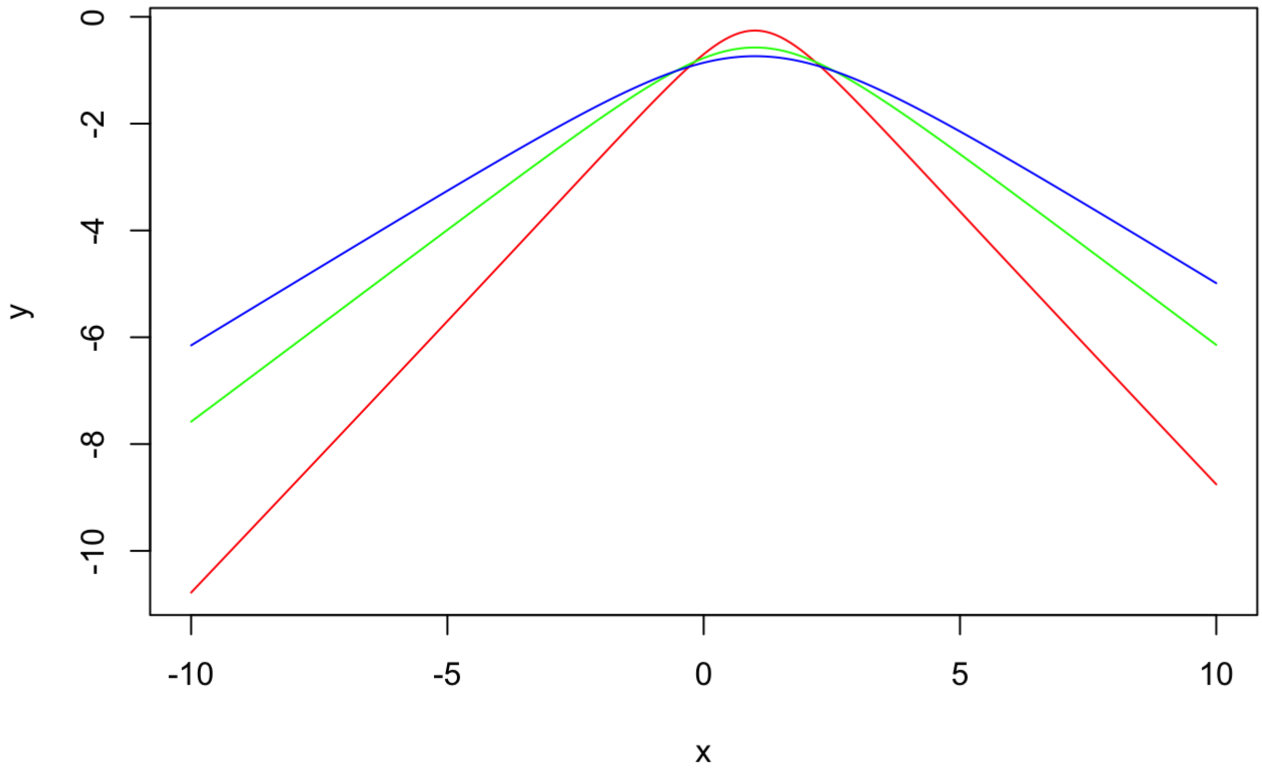
Q2

```
x <- seq(-10, 10, by = .1)
y <- pnorm(x, mean=1, sd=1)
y_1<-pnorm(x, mean=1, sd=1.41)
y_2<-pnorm(x, mean=1, sd=1.73)
plot(x, y,type = 'l',col='red')
lines(x, y_1,col='green')
lines(x, y_2,col='blue')
```



Q3

```
x <- seq(-10, 10, by = .1)
y <- qnorm(dnorm(x, mean=1, sd=1))
y_1<-qnorm(dnorm(x, mean=1, sd=1.41))
y_2<-qnorm(dnorm(x, mean=1, sd=1.73))
plot(x, y,type = 'l',col='red')
lines(x, y_1,col='green')
lines(x, y_2,col='blue')
```



reciprocal

Q4

```
set.seed(0)
standardGaussianSample<-c(rnorm(100, mean = 0,sd=1))
```

Q5

Q6

Q7

Location estimators with Gaussian data

```
set.seed(0)
num_trials_per_sample_size <- 1000
min_sample_size <- 30
max_sample_size <- 500
sample_size_inc <- 5
mu_0 <- 1
sigma_0 <- 3


simulation_df<-crossing(trial=seq(num_trials_per_sample_size),
                        sample_size=seq(min_sample_size,
                                        max_sample_size,sample_size_inc)) %>%
  mutate(simulation=pmap(.l=list(trial,sample_size),
                         .f=~rnorm(.y,mean=mu_0,sd=sigma_0))) %>%
  mutate(sample_md=map_dbl(.x=simulation,.f=median)) %>%
  group_by(sample_size)%>%
  summarise(msq_error_md=mean((sample_md-mu_0)^2))
```

Q1

```
median(simulation_df$msq_error_md)
```

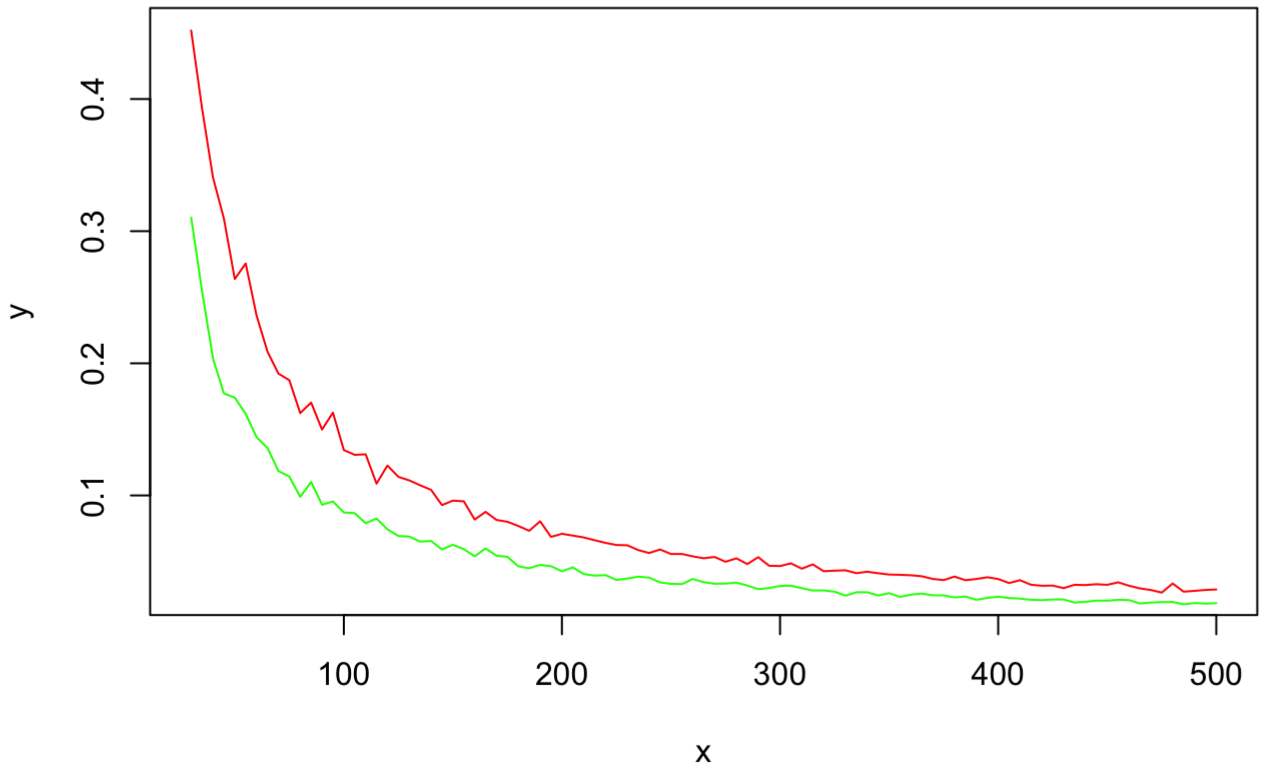
```
## [1] 0.05288549
```

Q2

```
simulation_df<-crossing(trial=seq(num_trials_per_sample_size),
                        sample_size=seq(min_sample_size,
                                        max_sample_size,sample_size_inc)) %>%
  mutate(simulation=pmap(.l=list(trial,sample_size),
                         .f=~rnorm(.y,mean=mu_0,sd=sigma_0))) %>%
  mutate(sample_md=map_dbl(.x=simulation,.f=median)) %>%
  group_by(sample_size)%>%
  summarise(msq_error_md=mean((sample_md-mu_0)^2))
```

```
simulation_df_mean<-crossing(trial=seq(num_trials_per_sample_size),
                        sample_size=seq(min_sample_size,
                                        max_sample_size,sample_size_inc)) %>%
  mutate(simulation=pmap(.l=list(trial,sample_size),
                         .f=~rnorm(.y,mean=mu_0,sd=sigma_0))) %>%
  mutate(sample_mn=map_dbl(.x=simulation,.f=mean)) %>%
  group_by(sample_size)%>%
  summarise(msq_error_mn=mean((sample_mn-mu_0)^2))
```

```
x <- simulation_df$sample_size
y <- simulation_df$msq_error_md
y_1<- simulation_df_mean$msq_error_mn
plot(x, y,type = 'l',col='red')
lines(x,y_1,col ='green')
```


0.1
0.2
0.3
0.4
y
100
200
300
400
500
x