WEEK - 3

- How to work with data in R?
  - stored in R [data frame]
  - Read in data into R

- Simulate samples from a given distribution.

Recall - Week 1 and 2

- How to work with R and Rstudio?
- Data Visualisation :-
  - ggplot
  - plot ..

ggplot2

Recall :-

Layered Grammar of graphics

- Each template takes 7 statements / Parameters

```
ggplot(data = <DATA>)

+ <GEOM_FUNCTION>(

mapping = aes(<MAPPINGS>),

stat = <STAT>,

position = <POSITION>

) +

<COORDINATE_FUNCTION> +

<FACET_FUNCTION>
```

Specify data frame

geom_point, ... [Type of geometry in graph]

map variables to an aesthetic

statistical transformation

flip, polar

faceting

Data Types in R

```
> Course = "B.Sc."

> Number = 40

> Smart = TRUE

> mode(Course)
[1] "character"

> mode(Number)
[1] "numeric"

> mode(Smart)
[1] "logical"
```

R has many data types

- Focus on three

- character data ≡ Surrounded by double quotes
- logical data ≡ TRUE or FALSE
- numeric data
- Another important characteristic is class. For this we can use the class function.

Creating Vectors in R

```
> x = 3:7
> x
[1] 3 4 5 6 7
```

Vectors - variables

```
> s= seq(1,10, by=1)
> s
 [1]  1  2  3  4  5  6  7  8  9 10
> s10 = seq(1,10,by=0.5)
> s10
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0
[12]  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0
```

Can be created by seq -

Position of 6.5 in the vector

```
> rep(6,7)
[1] 6 6 6 6 6 6 6
> rep(x,3)
 [1] 3 4 5 6 7 3 4 5 6 7 3 4 5 6 7
```

Repeats the first argument 6, as many times as the 2nd argument 7.

first argument can be a vector

Vectors in R

Karnataka Covid 19 Bulletin

```
> KA_D = c(215,620,558, 1109,8813,350, 780, 420,
+                    144,478,816,242,1051,249,1238, 315,
+                    807,185,1993,515,1997,2886,371,156,
+                    589,1746,838,964,296,128)
```

- Entered it physically into R - as a vector

```
> KA_D[c(1,3,5)]
[1]  215  558 8813
> KA_D[-c(1:20)]
 [1] 1997 2886  371  156  589 1746  838  964  296  128
```

- Select
- Delete

```
> KA_Dp = (KA_D)/sum(KA_D)*100
> KA_Dp
 [1]  0.6964916  2.0084875  1.8076387  3.5926010 28.5496777
 [6]  1.1338236  2.5268068  1.3605883  0.4664874  1.5484791
[11]  2.6434287  0.7839580  3.4047102  0.8066345  4.0104960
[16]  1.0204412  2.6142732  0.5993067  6.4563154  1.6683404
[21]  6.4692734  9.3491853  1.2018530  0.5053614  1.9080631
[26]  5.6561599  2.7146976  3.1228741  0.9588908  0.4146555
```

- Apply a certain function to the vector

Vectors in R

```
> KA_D[KA_D < 1000]
 [1] 215 620 558 350 780 420 144 478 816 242 249 315 807 185
[15] 515 371 156 589 838 964 296 128
```

```
> sum(KA_D >2000)
[1] 2
```

```
> max(KA_D)
[1] 8813
```

```
> which(KA_D==max(KA_D))
[1] 5
```

logical operators can be used to perform selection and identification

function max

Missing Values in R

```
> x = c(1,45,6,7,NA,99,0)
> x
[1]  1 45  6  7 NA 99  0

> x==NA
[1] NA NA NA NA NA NA NA

> mean(x)
[1] NA

> mean(x,na.rm=TRUE)
[1] 26.33333

> is.na(x)
[1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE

> mean(x[!is.na(x)])
[1] 26.33333
```

Missing values

- remove them before Performing / applying functions on them

- logical operators cannot be used to identify them

- is.na or na.rm Commands to identify.

Modes in R

Vector :- each element is in the same mode

```
> x = c("Siva", "looser", "3", "5")
```

```
> mode(x)
[1] "character"
```

Computation would lead to an error.

```
> x[3] +x[4]
```

R has to be told to treat it as a numeric before one does the computation.

```
> as.numeric(x[3]) + as.numeric(x[4])
[1] 8
```

Matrices in R

```
> A  = matrix(seq(3,5, by=1/10), 7,3)
> A
     [,1] [,2] [,3]
[1,]  3.0  3.7  4.4
[2,]  3.1  3.8  4.5
[3,]  3.2  3.9  4.6
[4,]  3.3  4.0  4.7
[5,]  3.4  4.1  4.8
[6,]  3.5  4.2  4.9
[7,]  3.6  4.3  5.0
```

All elements in the matrix have the same mode

vector used to create matrix.

Specify rows & columns

```
> B  = matrix(seq(3,5, by=1/10), ncol=3)
```

fill the entries row wise.

```
> C  = matrix(seq(3,5, by=1/10), ncol=3, byrow=TRUE)
> A[4,1]
[1] 3.3
```

Data Frames in R

```
> xd = c("Siva", "looser", 3, 5)
```

In R will convert all elements of the vector to the same mode

```
> xd
[1] "Siva"   "looser" "3"      "5"
```

Data Frames in R

District names

```
> KA_District=c("Bagalakote","Ballari","Belagavi",
+ "Bengaluru Rural","Bengaluru Urban", "Bidar","Chamarajanagara",
+ "Chikkaballapura","Chikkamagaluru","Chitradurga", "Dakshina Kannada",
+ "Davanagere","Dharwada","Gadag","Hassana","Haveri","Kalaburagi",
+ "Kodagu","Kolara","Koppala","Mandya","Mysuru","Raichuru",
+ "Ramanagara","Shivamogga","Tumakuru","Udupi","Uttara Kannada",
+ "Vijayapura","Yadagiri")
```

Discharge Data

```
> KA_Discharge = data.frame(KA_District, KA_D)
> class(KA_Discharge)
[1] "data.frame"
> mode(KA_Discharge)
[1] "list"
```

sapply - simplifies "loop function"

```
> sapply(KA_Discharge,mode)
KA_District          KA_D
"character"   "numeric"
```

Data frame

- rectangular array
- elements can be in different mode

- 1st column — character
- 2nd column — numeric

Data Frames as Matrix in R

```
> names(KA_Discharge)=c("District", "Recovered")
```

change the names of the variables in the dataframe

```
> KA_Discharge$Recovered
 [1]  215  620  558 1109 8813  350  780  420  144  478  816
[12]  242 1051  249 1238  315  807  185 1993  515 1997 2886
[23]  371  156  589 1746  838  964  296  128

> KA_Discharge[3,2]
[1] 558

> KA_Discharge[3,]
  District Recovered
3 Belagavi       558

> KA_Discharge[,"Recovered"]
 [1]  215  620  558 1109 8813  350  780  420  144  478  816
[12]  242 1051  249 1238  315  807  185 1993  515 1997 2886
[23]  371  156  589 1746  838  964  296  128
```

various ways of selecting objects in a data frame

Data Frames as Matrix in R

```
> Deaths= c(346, 1712, 975, 903, 16593, 407,515, 446, 400,221,
+   1750,  611,1333,328,1291,652,856, 343, 647, 530, 673,
+   2494, 346, 338, 1105, 1172, 509, 793, 500, 206
+ )
```

Death counts from the bulletin

```
> KA_Discharge$Deaths = Deaths
```

- created a new variable
- name = Deaths
- assigned values from vector created earlier.

```
> head(KA_Discharge)
        District Recovered Deaths
1      Bagalakote       215    346
2         Ballari       620   1712
3        Belagavi       558    975
4 Bengaluru Rural      1109    903
5 Bengaluru Urban      8813  16593
6           Bidar       350    407
```

head( ) - displays of the dataframe

Reading Data Frames into R

```
> kabulldf=read.csv(file="KAbulletin.csv", header = TRUE)
```

```
> names(kabulldf)
[1] "District"            "Today.s.Positives"
[3] "Total.Positives"     "Today.s.Discharges"
[5] "Total.Discharges"    "Total.Active.Cases"
[7] "Today.s.Deaths"      "Deaths"
```

- R/R-studio: please ensure you set the correct working directory
  - where csv file is located.

Used to read csv files.

other read commands

- read.table
- read.xl ...

(require package)

1st row of csv file - provides names of variables

Reading Data Frames into R

```
> head(kabulldf)
         District Today.s.Positives Total.Positives
1      Bagalakote               325           39150
2         Ballari               502          111730
3        Belagavi               900           91717
4 Bengaluru Rural               517           77877
5 Bengaluru Urban             10692         1720890
6           Bidar                84           28865
  Today.s.Discharges Total.Discharges Total.Active.Cases
1                215            36541               2263
2                620           104268               5750
3                558            83958               6784
4               1109            72741               4233
5               8813          1570258             134038
6                350            27619                835
  Today.s.Deaths Deaths
1              0    346
2              5   1712
3              6    975
4              0    903
5             12  16593
6              0    407
```

head( )

- 8 variables

- Ex:

- mode (kabulldf)
- sapply (kabulldf, mode)

Selecting from Data Frames in R

```
> kabulldf[which.max(kabulldf$"Today.s.Positives"),]
   District Today.s.Positives Total.Positives
32    Total             24172         3809467
   Today.s.Discharges Total.Discharges Total.Active.Cases
32              30869          3526108             244331
   Today.s.Deaths Deaths
32             56  38998
```

Selects the district that has the maximum "Today.s.Positives"

☺ Total is largest

1st

2nd

```
> hpkabulldf = subset(kabulldf,           Today.s.Positives > 1000)

> head(hpkabulldf,2)
          District Today.s.Positives Total.Positives
5   Bengaluru Urban             10692         1720890
13         Dharwada              1044           80274
   Today.s.Discharges Total.Discharges Total.Active.Cases
5                8813          1570258             134038
13               1051            72514               6425
   Today.s.Deaths Deaths
5              12  16593
13              1   1333
```

Queries

- find objects that have certain properties

Two arguments

- 1st = dataframe
- 2nd = Condition that you want to use to create subset

Selecting from Data Frames in R

1st

```
> IRDkabulldf = subset(kabulldf,
+     select=c("Total.Positives", "Total.Discharges", "Deaths"))
```

2nd

Data frame

Specifies the columns to be subset-ed.

```
> head(IRDkabulldf)
  Total.Positives Total.Discharges Deaths
1           39150            36541    346
2          111730           104268   1712
3           91717            83958    975
4           77877            72741    903
5         1720890          1570258  16593
6           28865            27619    407
```

Data frame created by subset command retains the same names for the variables as the original dataframe.

Ordering Data Frames in R

```
> okabulldf = kabulldf[order(kabulldf$Today.s.Positives),]
> head(okabulldf, 4)
     District Today.s.Positives Total.Positives
31    Others*                 0              36
6       Bidar                84           28865
24 Ramanagara                84           29064
29 Vijayapura               121           39690
   Today.s.Discharges Total.Discharges Total.Active.Cases
31                  0               33                  0
6                 350            27619                835
24                156            27559               1167
29                296            38080               1110
   Today.s.Deaths Deaths
31              0      3
6               0    407
24              0    338
29              0    500
```

①

Re order the rows of the dataframe - corresponding to the order of one variable

① - variable = Today.s.Positives

check: options in order command

Generating Random data in R

$X \sim$ **Uniform**$(\{1, 2, \ldots, n\})$:

Let $n \geq 1$ be an integer. If X is a random variable such that

$$P(X = k) = \frac{1}{n} \text{ for all } 1 \leq k \leq n$$

```
> sample(1:6,10, replace=T)
 [1] 6 3 3 1 1 2 1 2 4 5
```

Rolling a dice 10 times

fair dice

```
> sample(c(0,1), 10, replace =TRUE, prob = c(0.3,0.7))
```

Tossing a biased(0.7) Coin 10 times

Goal :-

- Generate samples from a given distribution.

- n=6 - Experiment is rolling a fair dice

vector to sample from

$\{1,2,3,4,5,6\}$

# of samples to generate

10

Sample with replacement

- 0 with probability 0.3
- 1 with probability 0.7

Generating Random data in R

$X \sim$ **Binomial**(n, p): Let $0 \leq p \leq 1$ and let $n \geq 1$ be an integer. If X is a random variable taking values in $\{0, 1, \ldots, n\}$ having a probability mass function

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for all $0 \leq k \leq n$.

- n Bernoulli trials
  - (Tossing a coin n times)
- # of successes
  (# of Heads or 1's)

Binomial Experiment

```
> rbinom(10, 6, 0.5)
 [1] 3 4 5 5 2 3 4 2 3 2
```

```
> rbinom(m, size,prob)
```

# of samples required

# of Bernoulli trials (n)

Prob of success (p)

```
> rbinom(10, 30,0.3)
 [1] 13 11  8  8  8  7 11  9  6 11
```

Ex:-

check

dbinom( )

qbinom( )

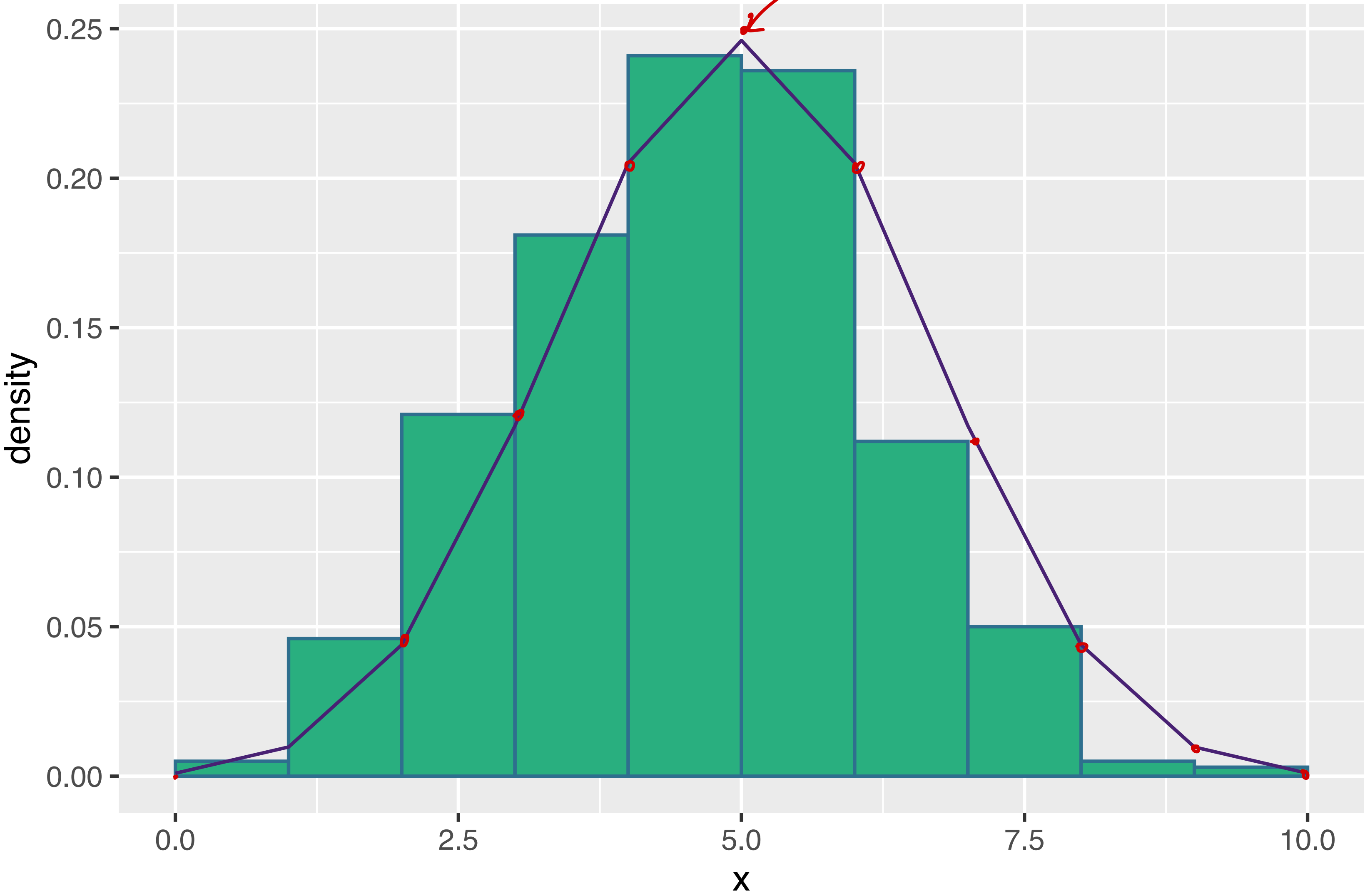
pbinom( )

Generating Random data in R

```
> rbinom(1000, 10,0.5)
```

$X \sim \text{Binomial}(10, 0.5)$

$\mathbb{P}(X = 5)$



Work sheet Exercise

- Generate 1000 samples of Binomial (10,0.5)
- Histogram of generated data (proportions)
- line plot of the true Binomial probabilities

Generating Random data in R

$X \sim$ **Normal**(μ, σ^2): Let $\mu \in \mathbb{R}$ and let $\sigma > 0$. Then X is said to be normally distributed with parameters μ and σ^2 if it has the density

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

for all $x \in \mathbb{R}$.

```
> rnorm(1,10,5)
[1] 5.865734
```

```
> rnorm(n, mean, sd)
```

# of samples

standard deviation of Normal random variable

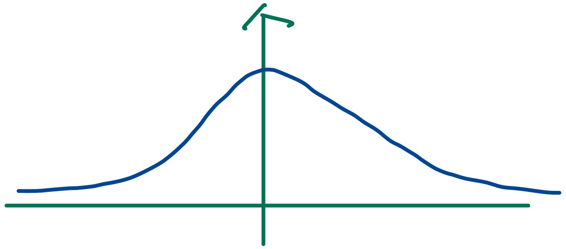
mean

```
> rnorm(10, 3, 5)
 [1]  7.529100  7.972982 -7.149743  6.655984  2.105155
 [6]  1.114047  9.126808  7.754853  7.459944  4.858321
```

Normal distribution

$\sigma = 1, \mu = 0$

- $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$



- Ex:-

dnorm()

pnorm()

qnorm()

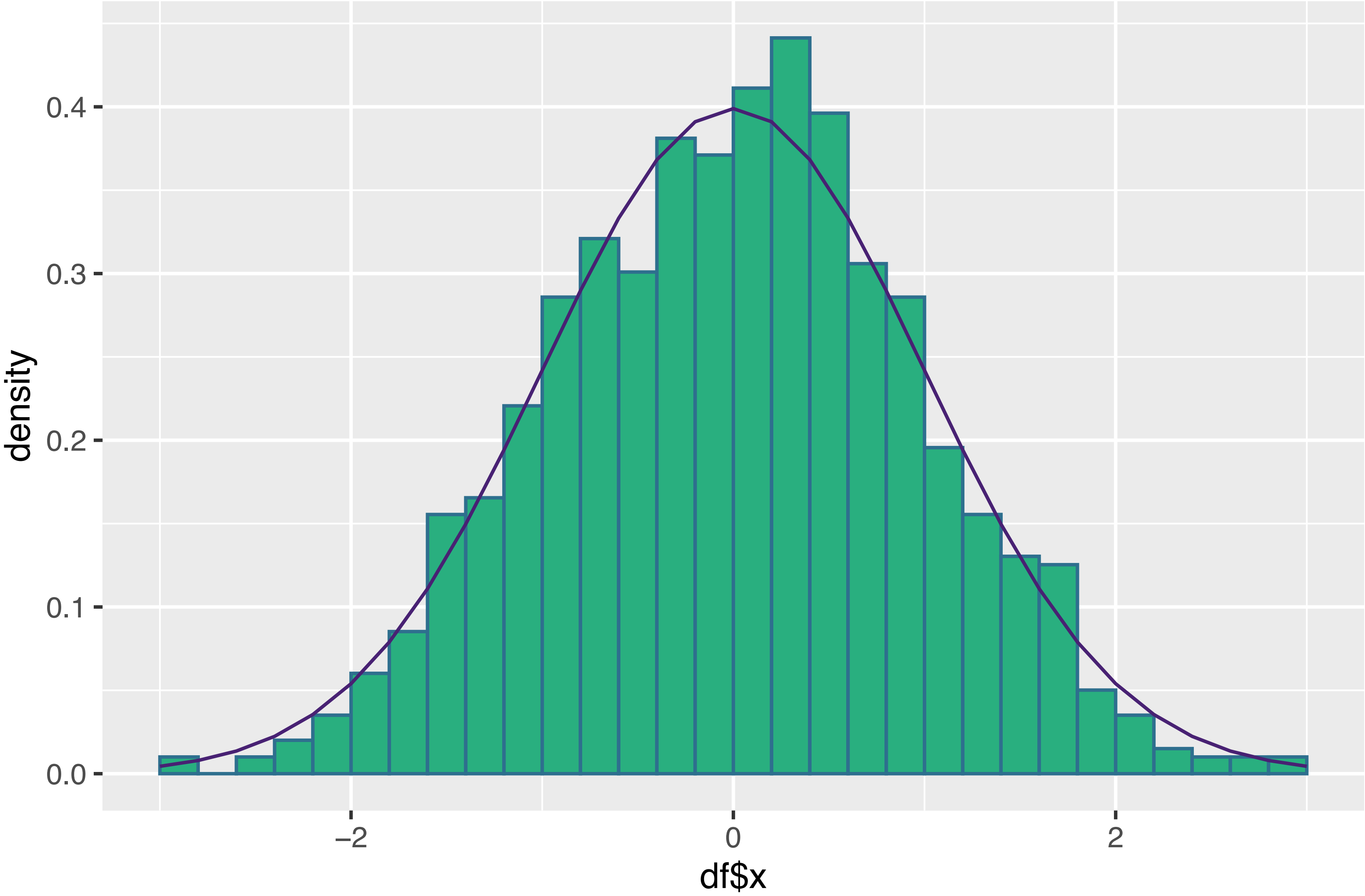
check

Generating Random data in R

Work sheet

Exercise

```
> rnorm(1000, 0,1)
```



- Generate 1000 samples of Normal (0,1)

Histogram of generated data (proportions)

- line plot of the true Normal density

Generating Random data in R

$X \sim \mathbf{Exp}(\lambda)$: Suppose $\lambda > 0$. If X is a random variable with its probabilty density function given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

it is said to be distributed exponentially with parameter λ.

```
> rexp(10, 1/2500)
 [1] 10091.04499  3826.35655   537.16355  1443.33491
 [5]  5863.52814  2368.87468  2256.52472  4008.94340
 [9]  1390.32837    83.46584
```

# of samples

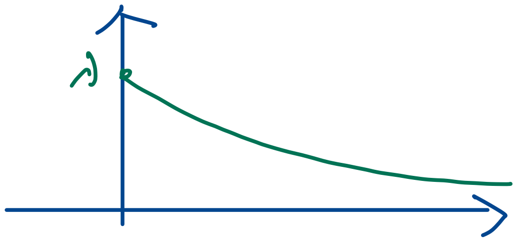
```
> rexp(n,rate)
```

rate

```
> rexp(10, 3)
 [1] 0.35294149 0.19999367 0.44155155 0.35013407 0.19273441
 [6] 0.07736766 0.39486037 0.08252493 0.03475531 0.10853122
```

Exponential distribution

rate $\equiv \lambda$



Ex:

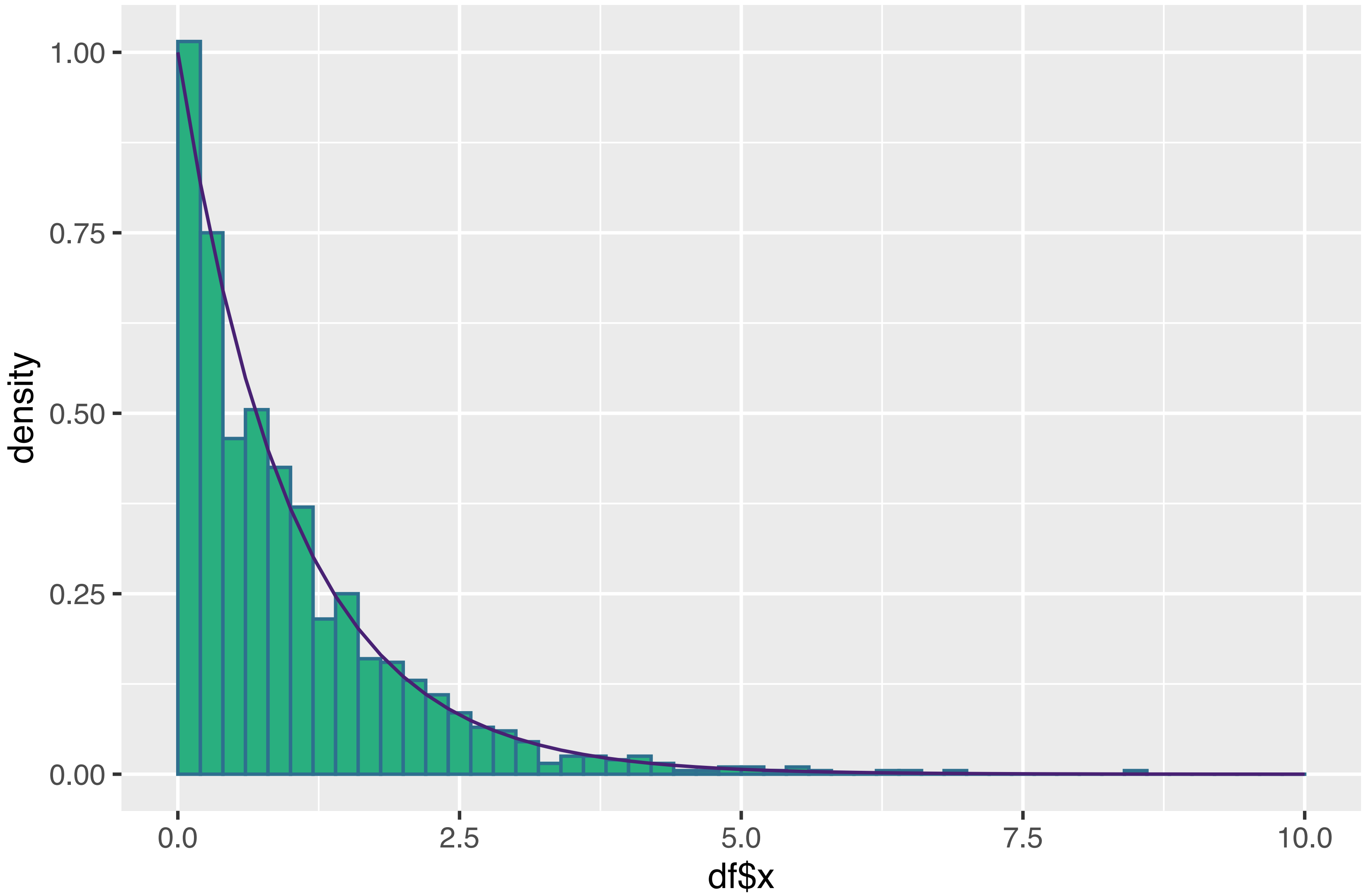
check dexp()

pexp()

qexp()

Generating Random data in R

```
> rexp(1000, 1)
```



Work sheet-

Exercise

- Generate 1000 samples of Exponential(1)

Histogram of generated data

(proportions)

- line plot of the true exponential density.