20INMCAL204- Lab Manual

Student[1,1,1,*], Siju K.S[b,2], Jobin Jose[b,2]

[a]*Department of Computer Applications RB402*
[b]*Department of Mathematics AB304*

Abstract

Experiments listed in the Lab Manual are successfully executed in the R version 4.1.0. Details of the experiments with input & output are summerized in the form of a report. Experiments are arranged in the form of sections. This report is prepared using the R-package `rticles` (Allaire et al., 2022).

Contents

*Corresponding author
Email addresses: `student@saintgits.org` (Student), `siju.swamy@saintgits.org` (Siju K.S), `jobin.jose@saintgits.org` (Jobin Jose)
[1]Student, 20INMCAL204.
[2]Course Faculty, 20INMCAL204.

1. Experiment 4: Statistical Summary and measure of normality of a dataset

1.1. Aim

1. To create the statistical summary of a data
2. To study normality of the data

1.2. Packages used and syntax of R methods

For statistical summary of a given dataset, the `rbase` package will be used. To calculate skewness and kurtosis of dataset, the `ACSWR` is used.

> *Note:* The functions `skewness` and `kurtosis` from the `e1071` package are more generic functions. Another resouse is `moments` package.

1.3. Algorithm

- Step 1: Load the dataset
- Step 2: Load necessary packages
- Step 3: Calculate statistical summaries
- Step 4: Calculate the `skewness` and `kurtosis` of the numerical data
- Step 5: Report the results

1.4. R code

```
#loading package
library(ACSWR)
```

```
## Warning: package 'ACSWR' was built under R version 4.1.3
```

```
#loading data
data(yb)
#view structure of data
str(yb)
```

```
## 'data.frame':    8 obs. of  2 variables:
##  $ Preparation_1: int  31 20 18 17 9 8 10 7
##  $ Preparation_2: int  18 17 14 11 10 7 5 6
```

```
# creating statistical summary

summary(yb)
```

```
##  Preparation_1   Preparation_2
##  Min.   : 7.00   Min.   : 5.00
##  1st Qu.: 8.75   1st Qu.: 6.75
##  Median :13.50   Median :10.50
##  Mean   :15.00   Mean   :11.00
##  3rd Qu.:18.50   3rd Qu.:14.75
##  Max.   :31.00   Max.   :18.00
```

```r
range(yb$Preparation_1); range(yb$Preparation_2) # list out ranges of data
```

```
## [1]  7 31
```

```
## [1]  5 18
```

```r
#skewness and kurtosis of preparation_1
skewcoeff(yb$Preparation_1); kurtcoeff(yb$Preparation_1)
```

```
## [1] 0.8548652
```

```
## [1] 2.727591
```

```r
#skewness and kurtosis of preparation_2
skewcoeff(yb$Preparation_2); kurtcoeff(yb$Preparation_2)
```

```
## [1] 0.2256965
```

```
## [1] 1.6106
```

1.5. Results & discussions

A distribution is normal then `mean=median=mode` and the skewness is 0 and kurtosis is 2. In this experiment statistical summaries of two variables are created. From the skewness and kurtosis measures, both the variables are positively skewed and `preparation_1` is lepto-kurtic and `preparation_2` is meso-kurtic. Based on the statistical summary and skewness and kurtosis measures, both the variables are different from a normal distribution.

2. Experiment 5- Implementation of Bayes Theorem

2.1. Aim

1. To calculate Bayes posterior probability using Bayes theorem

2.2. Packages used and syntax of R methods

Bayes posterior probability can be directly calculated using mathematical method or using the package `LaplacesDemon`.

2.3. Algorithm

- Step 1: Load the package, prior probabilities and conditionals
- Step 2: Calculate the Bayes posterior probability using the formula-$P(B_j|A) = \dfrac{P(A|B_j)P(B_j)}{\sum\limits_{j=1}^{m} P(A|B_j)P(B_j)}$
- Step 3: Calculate the same prior probability using `LaplaceDemon` package
- Step 4: Report the results

Case: Classical Problem from Hoel, Port, and Stone (1971). Suppose there are three tables with two drawers each. The first table has a gold coin in each of the drawers, the second table has a gold coin in one drawer and a silver coin in the other drawer, while the third table has silver coins in both of the drawers. A table is selected at random and a drawer is opened which shows a gold coin.

*Observation:*The problem is to compute the probability of the other drawer also showing a gold coin. The Bayes formula can be easily implemented in an R program.

2.4. R code

```
#loading data
prob_GC <- c(1,1/2,0)
priorprob_GC <- c(1/3,1/3,1/3)
```

```
#calculating postrior probability
post_GC <- prob_GC*priorprob_GC
post_GC/sum(post_GC)
```

```
## [1] 0.6666667 0.3333333 0.0000000
```

```
# do the same using LaplacesDemon` package
library(LaplacesDemon)
```

```
## Warning: package 'LaplacesDemon' was built under R version 4.1.3
```

```
BayesTheorem(prob_GC, priorprob_GC)
```

```
## [1] 0.6666667 0.3333333 0.0000000
## attr(,"class")
## [1] "bayestheorem"
```

2.5. Results & discussions

The Bayes theorem is used to calculate posterior probability of the Mathematical model of the given case. Also the result is verified using the `LaplacesDemon` package.

3. Experiment 6: Binomial Distribution

3.1. Aim

1. To calculate probability mass density, probability distribution and quantiles using binomial distribution

3.2. Packages used and syntax of R methods

Functions from `stat` package (which is loaded by default) are used.
The probability mass at a point x can be evaluated using the syntax:

`dbinom(x=x,size=n,p=p)`.

The probability distribution $P(X \leq x)$ is calculated using the `pbinom()` function. Syntax is:

`pbinom(x,size=n,p=p)`

The quantile for probability p can be evaluated using the `quantile()` function. Syntax is:

`qbinom(prob,size,p=p)`

3.3. Algorithm

- Step 1: Assign the inputs for required distribution
- Step 2: Calculate the required probabilities
- Step 3: Report the results

Case: Find the mass function of a binomial distribution with $n = 20, p = 0.4$. Also draw the graphs of the mass function and cumulative distribution function.

3.4. R code

```r
# create input parameters
n=20
p=0.4
x=0:20
```

3.4.1. Prbability distribution

```r
#calculating probability mass distribution and cumulative distribution
pmval=dbinom(x,size=n,prob=p)
pmval
```

```
##  [1] 3.656158e-05 4.874878e-04 3.087423e-03 1.234969e-02 3.499079e-02
##  [6] 7.464702e-02 1.244117e-01 1.658823e-01 1.797058e-01 1.597385e-01
## [11] 1.171416e-01 7.099488e-02 3.549744e-02 1.456305e-02 4.854351e-03
## [16] 1.294494e-03 2.696862e-04 4.230371e-05 4.700412e-06 3.298535e-07
## [21] 1.099512e-08
```

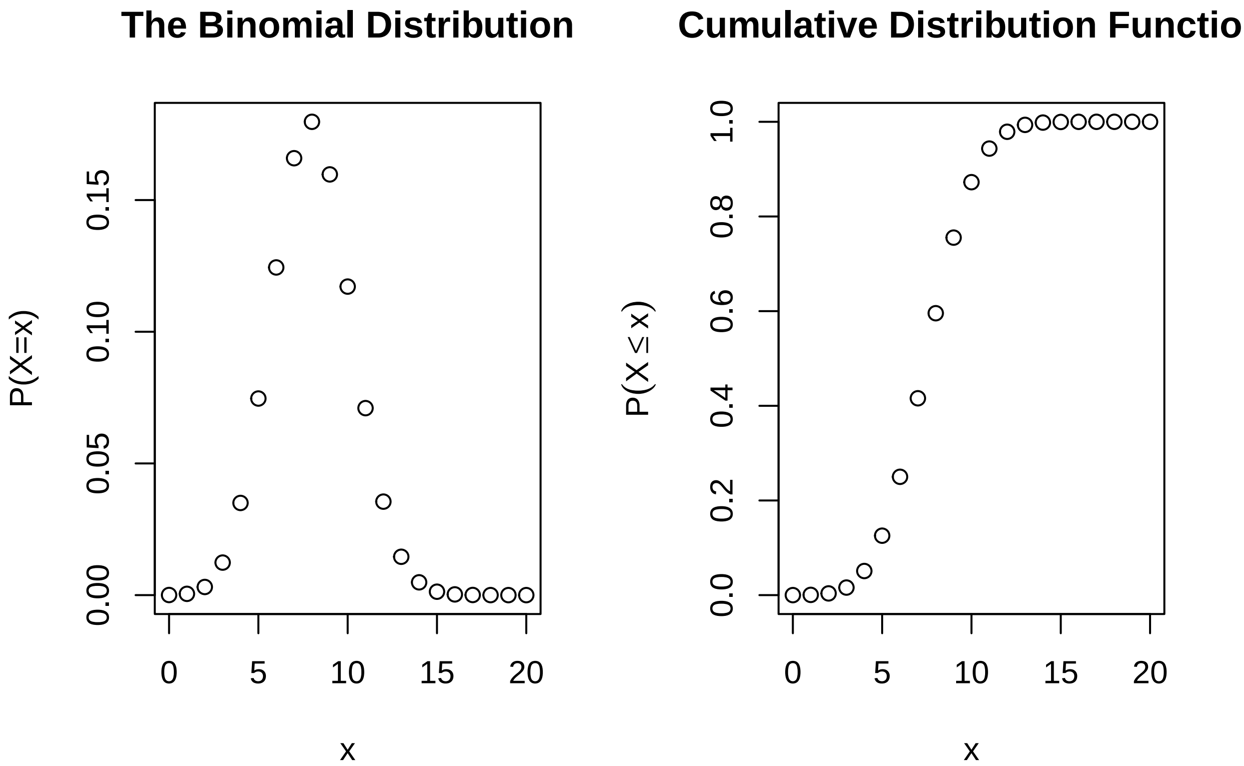
3.5. Cumulative probability distribution

```r
#calculating cumulative density
cdval=pbinom(x,size=n,prob=p)
cdval
```

```
##  [1] 3.656158e-05 5.240494e-04 3.611472e-03 1.596116e-02 5.095195e-02
##  [6] 1.255990e-01 2.500107e-01 4.158929e-01 5.955987e-01 7.553372e-01
## [11] 8.724788e-01 9.434736e-01 9.789711e-01 9.935341e-01 9.983885e-01
## [16] 9.996830e-01 9.999527e-01 9.999950e-01 9.999997e-01 1.000000e+00
## [21] 1.000000e+00
```

3.5.1. Plotting the `pmf` and `cdf`

```r
par(mfrow=c(1,2))
plot(x,pmval,xlab="x",ylab="P(X=x)", main="The Binomial Distribution")
plot(x,cdval,xlab="x",ylab=expression(P(X<=x)),main="Cumulative Distribution Function")
```



3.6. Results & discussions

The `pmf` and `cf` of the Binomial distribution for given input parameters are evaluated and create respective plots.

4. Experiment 7: Poisson Distribution

4.1. Aim

1. To calculate probability mass density, probability distribution and quantiles using Poisson distribution

4.2. Packages used and syntax of R methods

Functions from `stat` package (which is loaded by default) are used.
The probability mass at a point x can be evaluated using the syntax:

```
dpois(x=x,lambda=l).
```

The probability distribution $P(X \leq x)$ is calculated using the `ppois()` function. Syntax is:

```
pbinom(x,lambda=l)
```

The quantile for probability p can be evaluated using the `qpois()` function. Syntax is:

```
qpois(prob,lambda=l)
```

4.3. Algorithm

- Step 1: Assign the inputs for required distribution
- Step 2: Calculate the required probabilities
- Step 3: Report the results

Case: Given the data $n = 50$, mean, $\lambda = 25$, use appropriate function to find the mass function of a Poisson distribution. Also draw the graphs of the mass function and cumulative distribution function.

4.4. R code

```
# create input parameters
n <-50; l <- 25
x=0:50
```

4.4.1. Prbability distribution

```
#calculating probability mass distribution and cumulative distribution
pmval=dpois(x,lambda=l)
pmval
```

```
##  [1] 1.388794e-11 3.471986e-10 4.339982e-09 3.616652e-08 2.260408e-07
##  [6] 1.130204e-06 4.709182e-06 1.681851e-05 5.255784e-05 1.459940e-04
## [11] 3.649850e-04 8.295113e-04 1.728149e-03 3.323363e-03 5.934576e-03
## [16] 9.890961e-03 1.545463e-02 2.272739e-02 3.156582e-02 4.153397e-02
## [21] 5.191747e-02 6.180651e-02 7.023467e-02 7.634203e-02 7.952295e-02
## [26] 7.952295e-02 7.646438e-02 7.080035e-02 6.321460e-02 5.449534e-02
## [31] 4.541279e-02 3.662321e-02 2.861189e-02 2.167567e-02 1.593799e-02
## [36] 1.138428e-02 7.905751e-03 5.341723e-03 3.514292e-03 2.252751e-03
## [41] 1.407969e-03 8.585180e-04 5.110226e-04 2.971062e-04 1.688103e-04
## [46] 9.378351e-05 5.096930e-05 2.711133e-05 1.412048e-05 7.204329e-06
## [51] 3.602164e-06
```

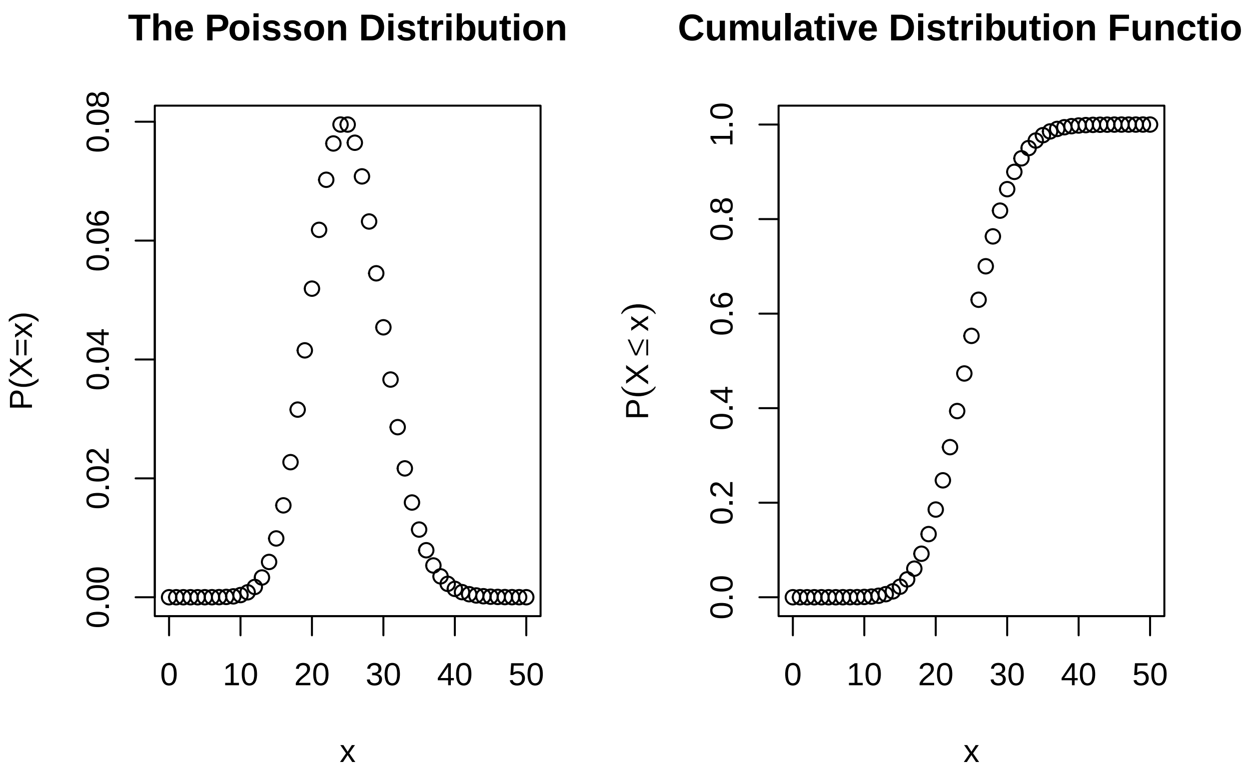
4.4.2. Cumulative probability distribution

```
#calculating cumulative density
cdval=ppois(x,lambda = l)
cdval
```

```
##  [1] 1.388794e-11 3.610865e-10 4.701069e-09 4.086759e-08 2.669083e-07
##  [6] 1.397112e-06 6.106294e-06 2.292480e-05 7.548264e-05 2.214766e-04
## [11] 5.864616e-04 1.415973e-03 3.144122e-03 6.467484e-03 1.240206e-02
## [16] 2.229302e-02 3.774765e-02 6.047504e-02 9.204086e-02 1.335748e-01
## [21] 1.854923e-01 2.472988e-01 3.175335e-01 3.938755e-01 4.733985e-01
## [26] 5.529214e-01 6.293858e-01 7.001861e-01 7.634007e-01 8.178961e-01
## [31] 8.633089e-01 8.999321e-01 9.285440e-01 9.502196e-01 9.661576e-01
## [36] 9.775419e-01 9.854477e-01 9.907894e-01 9.943037e-01 9.965564e-01
## [41] 9.979644e-01 9.988229e-01 9.993339e-01 9.996310e-01 9.997999e-01
## [46] 9.998936e-01 9.999446e-01 9.999717e-01 9.999858e-01 9.999930e-01
## [51] 9.999966e-01
```

*4.4.3. Plotting the **pmf** and **cdf***

```
par(mfrow=c(1,2))
plot(x,pmval,xlab="x",ylab="P(X=x)", main="The Poisson Distribution")
plot(x,cdval,xlab="x",ylab=expression(P(X<=x)),main="Cumulative Distribution Function")
```



4.5. Results & discussions

The `pmf` and `cf` of the Poisson distribution for given input parameters are evaluated and create respective plots.

5. Experiment 8: Normal Distribution

5.1. Aim

1. To calculate probability mass density, probability distribution and quantiles using Normal distribution

5.2. Packages used and syntax of R methods

Functions from `stat` package (which is loaded by default) is used.
The probability mass at a point x can be evaluated using the syntax:

`dnorm(x=x,mean=m,sd=s)`.

The probability distribution $P(X \leq x)$ is calculated using the `pnorm()` function. Syntax is:

`pnorm(x,mean=m,sd=s)`

The quantile for probability p can be evaluated using the `qnorm()` function. Syntax is:

`qnorm(prob,mean=m,sd=s)`

5.3. Algorithm

- Step 1: Assign the inputs for required distribution
- Step 2: Calculate the required probabilities
- Step 3: Report the results

Case: Generate and draw the cdf and pdf of a normal distribution with mean=10 and standard deviation=3. Use values of x from 0 to 20 in intervals of 1.

5.4. R code

```
# create input parameters
t=seq(0,20,1); mu=10;sd=3
```

5.4.1. Prbability distribution

```
#calculating probability mass distribution and cumulative distribution
pmval=dnorm(t,mean = mu,sd=sd)
pmval
```

```
##  [1] 0.000514093 0.001477283 0.003798662 0.008740630 0.017996989 0.033159046
##  [7] 0.054670025 0.080656908 0.106482669 0.125794409 0.132980760 0.125794409
## [13] 0.106482669 0.080656908 0.054670025 0.033159046 0.017996989 0.008740630
## [19] 0.003798662 0.001477283 0.000514093
```

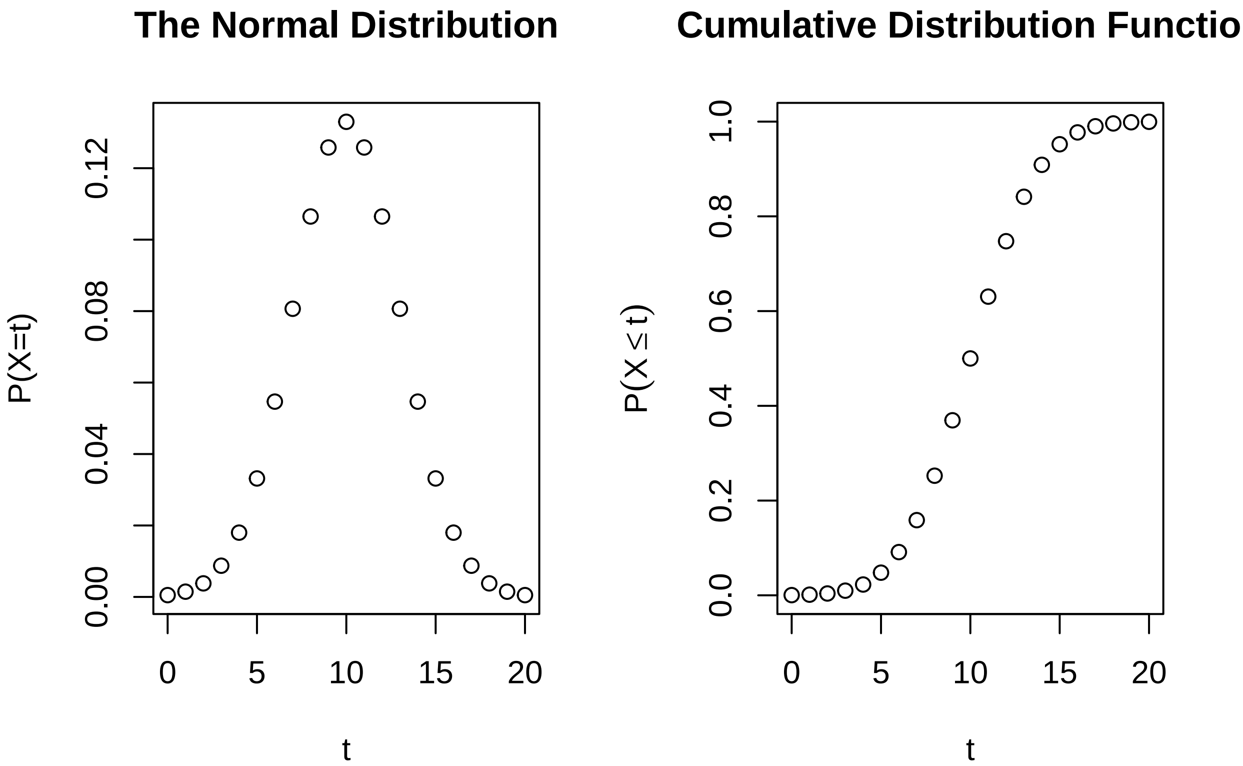
5.4.2. Cumulative probability distribution

```
#calculating cumulative density
cdval=pnorm(t,mean = mu,sd=sd)
cdval
```

```
##  [1] 0.0004290603 0.0013498980 0.0038303806 0.0098153286 0.0227501319
##  [6] 0.0477903523 0.0912112197 0.1586552539 0.2524925375 0.3694413402
## [11] 0.5000000000 0.6305586598 0.7475074625 0.8413447461 0.9087887803
## [16] 0.9522096477 0.9772498681 0.9901846714 0.9961696194 0.9986501020
## [21] 0.9995709397
```

5.4.3. Plotting the `pmf` and `cdf`

```
par(mfrow=c(1,2))
plot(t,pmval,xlab="t",ylab="P(X=t)", main="The Normal Distribution")
plot(t,cdval,xlab="t",ylab=expression(P(X<=t)),main="Cumulative Distribution Function")
```



5.5. Results & discussions

The `pmf` and `cf` of the Normal distribution for given input parameters are evaluated and create respective plots.

6. Experiment 9: Fitting of Distribution

6.1. Aim

1. To fit a binomial distribution to the given data

6.2. Packages used and syntax of R methods

Functions from `stat` package (which is loaded by default) is used.

Any observed distribution can be fitted to a theoretical distribution using the formula $N \times p(X = x)$, where N is the pdf of the target distribution.

6.3. Algorithm

- Step 1: Assign the inputs for required fitting model
- Step 2: Calculate the theoretical frequencies
- Step 3: Visualize the observed distribution and theoretical distribution
- Step 4: Compare the difference in orginal and fitted distributions

Case: The following data shows the result of throwing 12 fair dice 4,096 times; a throw of 4,5, or 6 being called success.

Success(X) :	0	1	2	3	4	5	6	7	8	9	10	11	12
Frequency(f) :	0	7	60	198	430	731	948	847	536	257	71	11	0

Fit a binomial distribution and find the expected frequencies. Compare the graphs of the observed frequency and theoretical frequency.

6.4. R code

```
# create input parameters
p=3/6
Observed=c(0 ,7, 60,198,430,731,948,847,536, 257, 71,11, 0)
success=seq(0,12,1)
N=sum(Observed)
```

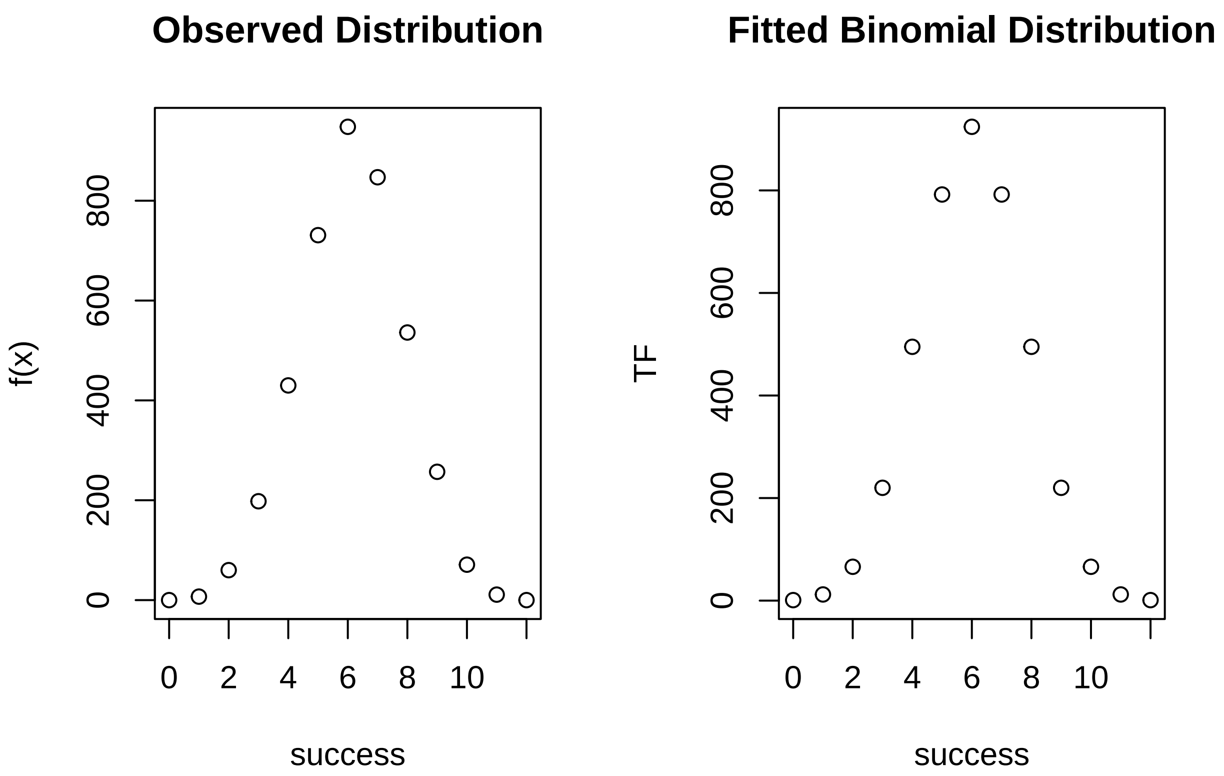
6.4.1. Theoretical distribution

```
#calculating theoretical frequencies
TF=round(N*dbinom(success,size=12,prob=p),2)
TF
```

```
## [1]   1  12  66 220 495 792 924 792 495 220  66  12   1
```

6.4.2. Plotting the observed and theoretical frequencies

```
par(mfrow=c(1,2))
plot(success,Observed,xlab="success",ylab="f(x)")
title("Observed Distribution")
plot(success,TF,xlab="success",ylab="TF")
title("Fitted Binomial Distribution")
```



6.4.3. *Fitting table*

```
Fit=data.frame(Success=success,Observed,Fitted_binom=TF)
knitr::kable(
  Fit, caption = 'The binomial fitting table',
  booktabs = TRUE)
```

Table 1: The binomial fitting table

Success	Observed	Fitted_binom
0	0	1
1	7	12
2	60	66
3	198	220
4	430	495
5	731	792
6	948	924
7	847	792
8	536	495
9	257	220
10	71	66
11	11	12
12	0	1

6.5. Results & discussions

Given arbitrary distribution is mapped into the framework of a binomial distribution.

7. Experiment 10: Correlation analysis- Karl-Pearson Coefficient of Correlation

7.1. Aim

1. To find the correlation coefficient of the given bivariate data

7.2. Packages used and syntax of R methods

Functions from `stat` package (which is loaded by default).

Pearson correlation (r), which measures a linear dependence between two variables (x and y). It's also known as a parametric correlation test because it depends to the distribution of the data. It can be used only when x and y are from normal distribution. The plot of $y = f(x)$ is named the linear regression curve. The Pearson correlation formula is:

$$r = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}$$

where m_x and m_y are means of the distributions x and y respectively.

Correlation coefficient can be computed using the functions `cor()` or `cor.test()`:

Syntax:

`cor(x, y, method = c("pearson", "kendall", "spearman"))` `cor.test(x, y, method=c("pearson", "kendall", "spearman"))` Note: If the data contain missing values, use the following R code to handle missing values by case-wise deletion. `cor(x, y, method = "pearson", use = "complete.obs")`

7.3. Algorithm

- Step 1: Assign the inputs for required correlation model
- Step 2: Calculate the correlation coefficient
- Step 3: Verify the result using direct calculation
- Step 4: Interpret the result

Case: From the following data, compute Karl Pearson's coefficient of correlation.

Price(Rupees) :	10	20	30	40	50	60	70
Supply(Units) :	8	6	14	16	10	20	24

7.4. R code

```r
# loading data
price=c(10,20,30,40,50,60,70)
supply=c(8,6,14,16,10,20,24)
```

7.4.1. Calculating correlation coefficient

```
resp1=cor.test(price,supply,method='pearson')
resp1
```

```
##
##  Pearson's product-moment correlation
##
## data:  price and supply
## t = 3.6145, df = 5, p-value = 0.01531
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2707625 0.9774828
## sample estimates:
##       cor
## 0.8504201
```

7.4.2. Direct calculation

Formula is:

$$r = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}$$

```
#direct calculation
rho=(sum(((price-mean(price))*(supply-mean(supply)))))/(sqrt(sum((price-mean(price))^2)*(sum((supply-m
rho
```

```
## [1] 0.8504201
```

7.4.3. Plotting the variables

```
my_data=data.frame(price=price,supply=supply)
head(my_data)
```

```
##   price supply
## 1    10      8
## 2    20      6
## 3    30     14
## 4    40     16
## 5    50     10
## 6    60     20
```

```
library(ggpubr)
```

```
## Loading required package: ggplot2
```

```
ggscatter(my_data, x = "price", y = "supply",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Price", ylab = "Supply")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

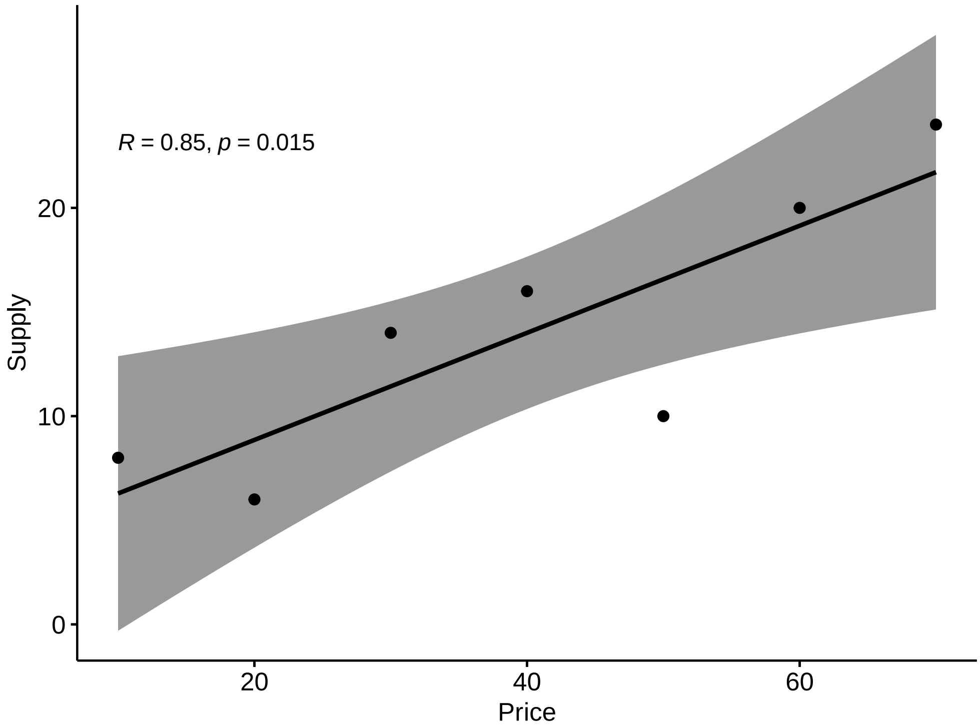


Figure 1: Scatter plot with smooth fit curve

7.5. Results & discussions

Correlation coefficient of given bivariate data is calculated using built-in function in `stats` package. The result is verified using direct calculation.

Since the Pearson coefficient is 0.8504201. Also p-value is 0.015308 <0.05. So the null hypothesis is accepted. So it is statistically reasonable to conclude that there is a significant positive correlation between the price and supply based on the sample.

8. Experiment 11: Correlation analysis- Karl-Pearson Coefficient of Correlation

8.1. Aim

1. To find the correlation coefficient of the given bivariate data

8.2. Packages used and syntax of R methods

Functions from `stat` package (which is loaded by default)is used.

Pearson correlation (r), which measures a linear dependence between two variables (x and y). It's also known as a parametric correlation test because it depends to the distribution of the data. It can be used only when x and y are from normal distribution. The plot of $y = f(x)$ is named the linear regression curve. The Pearson correlation formula is:

$$r = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}$$

where m_x and m_y are means of the distributions x and y respectively.

Correlation coefficient can be computed using the functions `cor()` or `cor.test()`:

Syntax:

cor(x, y, method = c("pearson", "kendall", "spearman")) cor.test(x, y, method=c("pearson", "kendall", "spearman")) Note: If the data contain missing values, use the following R code to handle missing values by case-wise deletion. cor(x, y, method = "pearson", use = "complete.obs")

8.3. Algorithm

- Step 1: Assign the inputs for required correlation model
- Step 2: Calculate the correlation coefficient
- Step 3: Verify the result using direct calculation
- Step 4: Interpret the result

Case: From the following data compute correlation between height of father and height of daughters by Karl Pearson's coefficient of correlation.

Height of Father(Cms)	65	66	67	67	68	69	71	73
Height of Daughter(Cms)	67	68	64	69	72	70	69	73

8.4. R code

```
# loading data
height_F=c(65,66,67,67,68,69,71,73)
height_D=c(67,68,64,69,72,70,69,73)
```

8.4.1. Calculating correlation coefficient

```
resp2=cor.test(height_F,height_D,method='pearson')
resp2
```

```
##
##  Pearson's product-moment correlation
##
## data:  height_F and height_D
## t = 2.0717, df = 6, p-value = 0.08369
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1080788  0.9281049
## sample estimates:
##       cor
## 0.6457766
```

8.4.2. Direct calculation

Formula is:

$$r = \frac{\sum(x-m_x)(y-m_y)}{\sqrt{\sum(x-m_x)^2\sum(y-m_y)^2}}$$

```
#direct calculation
rho=(sum(((height_F-mean(height_F))*(height_D-mean(height_D)))))/(sqrt(sum((height_F-mean(height_F))^2
rho
```

```
## [1] 0.6457766
```

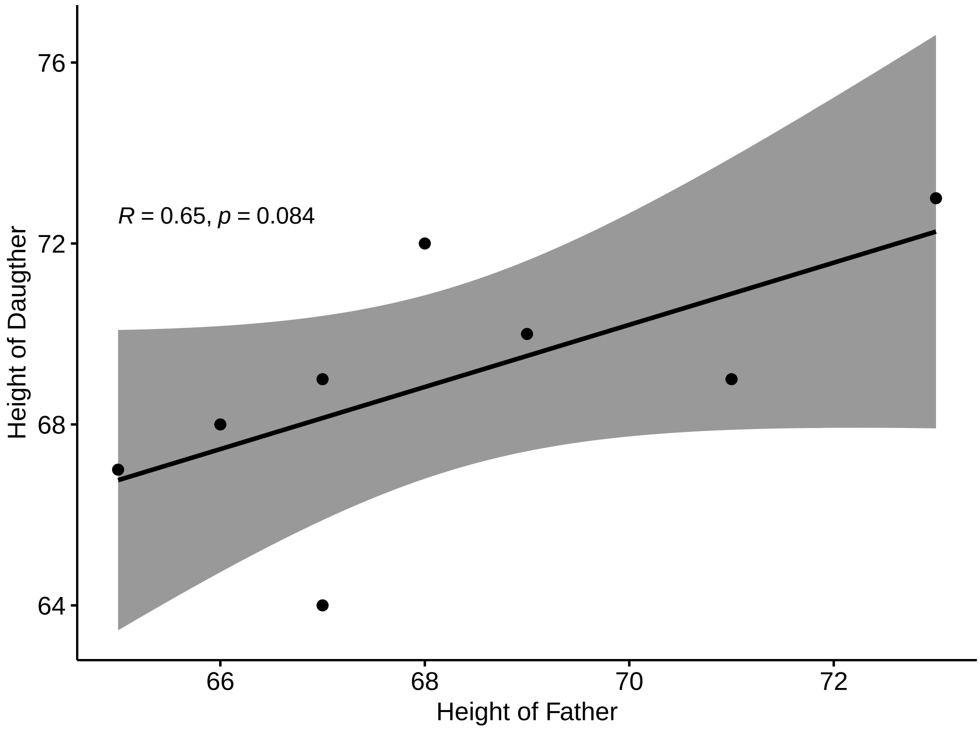


Figure 2: Scatter plot with smooth fit curve

8.4.3. Plotting the variables

```
my_data=data.frame(height_F=height_F,height_D=height_D)
head(my_data)
```

```
##   height_F height_D
## 1       65       67
## 2       66       68
## 3       67       64
## 4       67       69
## 5       68       72
## 6       69       70
```

```
library(ggpubr)
ggscatter(my_data, x = "height_F", y = "height_D",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Height of Father", ylab = "Height of Daugther")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

8.5. Results & discussions

Correlation coefficient of given bivariate data is calculated using built-in function in `stats` package. The result is verified using direct calculation.

The Pearson coefficient is 0.6457766. Also p-value is 0.0836865 >0.05. So the null hypothesis is rejected. So it is statistically reasonable to conclude that there is no significant positive correlation between the ages of the samples.

9. Experiment 12: Correlation analysis- Spearman Rank Correlation Coefficient

9.1. Aim

1. To find the Spearman rank correlation coefficient of the given bivariate data

9.2. Packages used and syntax of R methods

Functions from `stat` package (which is loaded by default) is used.

Pearson correlation (r), which measures a linear dependence between two variables (x and y). It's also known as a parametric correlation test because it depends to the distribution of the data. It can be used only when x and y are from normal distribution. The plot of $y = f(x)$ is named the linear regression curve. The Spearman correlation method computes the correlation between the rank of x and the rank of y variables.

$$rho = \frac{\sum(x' - m_{x'})(y'_i - m_{y'})}{\sqrt{\sum(x' - m_{x'})^2 \sum(y' - m_{y'})^2}}$$

where $x' = rank(x)$ and $y' = rank(y)$.

Correlation coefficient can be computed using the functions `cor()` or `cor.test()`:

Syntax:

`cor(x, y, method = c("pearson", "kendall", "spearman"))` `cor.test(x, y, method=c("pearson", "kendall", "spearman"))` Note: If the data contain missing values, use the following R code to handle missing values by case-wise deletion. `cor(x, y, method = "pearson", use = "complete.obs")`

9.3. Algorithm

- Step 1: Assign the inputs for required correlation model
- Step 2: Calculate the correlation coefficient
- Step 3: Interpret the result

Case: The scores for nine students in history and algebra are as follows:

History :	35	23	47	17	10	43	9	6	28
Algebra :	30	33	45	23	8	49	12	4	31

Compute the Spearman rank correlation.

9.4. R code

```r
# loading data
History=c(35,23,47,17,10,43,9,6,28)
Algebra=c(30,33,45,23,8,49,12,4,31)
```

9.4.1. Calculating correlation coefficient

```
ress3=cor.test(History,Algebra,method='spearman')
ress3
```

```
##
##  Spearman's rank correlation rho
##
## data:  History and Algebra
## S = 12, p-value = 0.002028
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
## rho
## 0.9
```

9.4.2. Plotting the variables

```
my_data=data.frame(History=History,Algebra=Algebra)
head(my_data)
```

```
##   History Algebra
## 1      35      30
## 2      23      33
## 3      47      45
## 4      17      23
## 5      10       8
## 6      43      49
```

```
library(ggpubr)
ggscatter(my_data, x = "History", y = "Algebra",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "spearman",
          xlab = "Marks in History", ylab = "Marks in Algebra")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

9.5. Results & discussions

Correlation coefficient of given bivariate data is calculated using built-in function in `stats` package.

The Spearman rank correlation coefficient is 0.9. Also p-value is 0.0020282 >0.05. So the null hypothesis is rejected. So it is statistically reasonable to conclude that there is significant positive correlation between the marks of the samples.

10. Experiment 13: Ordinary Linear Regression using R

10.1. Aim

1. To fit a linear regression to the given data

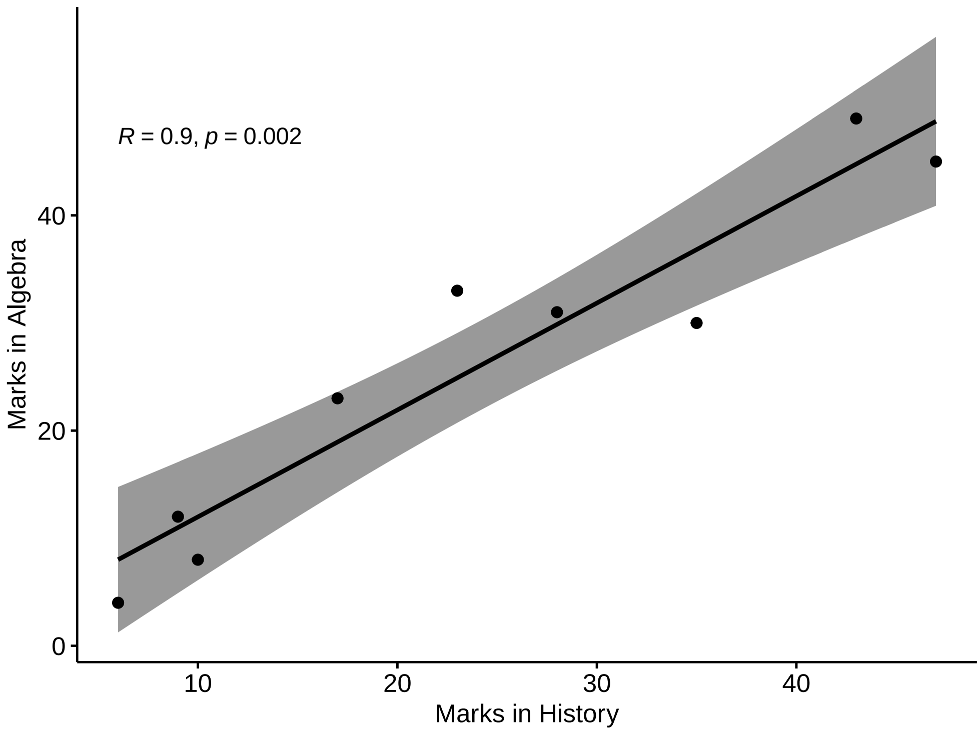


Figure 3: Scatter plot with smooth fit curve

10.2. Packages used and syntax of R methods

Functions from `stats` package (which is loaded by default).

Regression assumptions

Linear regression makes several assumptions about the data, such as :

1. Linearity of the data. The relationship between the predictor (x) and the outcome (y) is assumed to be linear.
2. Normality of residuals. The residual errors are assumed to be normally distributed.
3. Homogeneity of residuals variance. The residuals are assumed to have a constant variance (homoscedasticity)
4. Independence of residuals error terms.

One should check whether or not these assumptions hold true. Potential problems include:

1. Non-linearity of the outcome - predictor relationships
2. Heteroscedasticity: Non-constant variance of error terms.
3. Presence of influential values in the data that can be:
4. Outliers: extreme values in the outcome (y) variable
5. High-leverage points: extreme values in the predictors (x) variable
6. All these assumptions and potential problems can be checked by producing some diagnostic plots visualizing the residual errors.

When build a regression model, one need to assess the performance of the predictive model. In other words, we need to evaluate how well the model is in predicting the outcome of a new test data that have not been used to build the model.

Two important metrics are commonly used to assess the performance of the predictive regression model:

Root Mean Squared Error, which measures the model prediction error. It corresponds to the average difference between the observed known values of the outcome and the predicted value by the model. RMSE is computed as $RMSE = mean((observeds - predicteds)^2)$

We read this as "y is modeled as beta1 (b_1) times x, plus a constant beta0 (b_0), plus an error term e."

When we have multiple predictor variables, the equation can be written as $y = b_0 + b_1 * x + e$, where: * b_0 is the intercept, $b_1, b_2, \ldots, b_n$ are the regression weights or coefficients associated with the predictors $x_1, x_2, \ldots, x_n$. * e is the error term (also known as the residual errors), the part of y that can be explained by the regression model.

10.3. Algorithm

- Step 1: Assign the inputs for required regression model
- Step 2: Create the regression model using linear model- `lm()` function in R
- Step 3: Report the summary coefficients
- Step 4: Interpret the model summary

Case: Build a simple linear model to predict sales units based on the advertising budget spent on youtube. The sales data with corresponding expenditure on advertisment in Youtube is shown below:

Sales(Y) :	2	4	6	9	12	34	45
Add expense(Youtube) :	1	2	4	7	9	11	15

Fit a OLS model and predict the sales for an add expence ($) $1000, 2000, 3500$. Also interpret the the linear model with suitable fit measures.

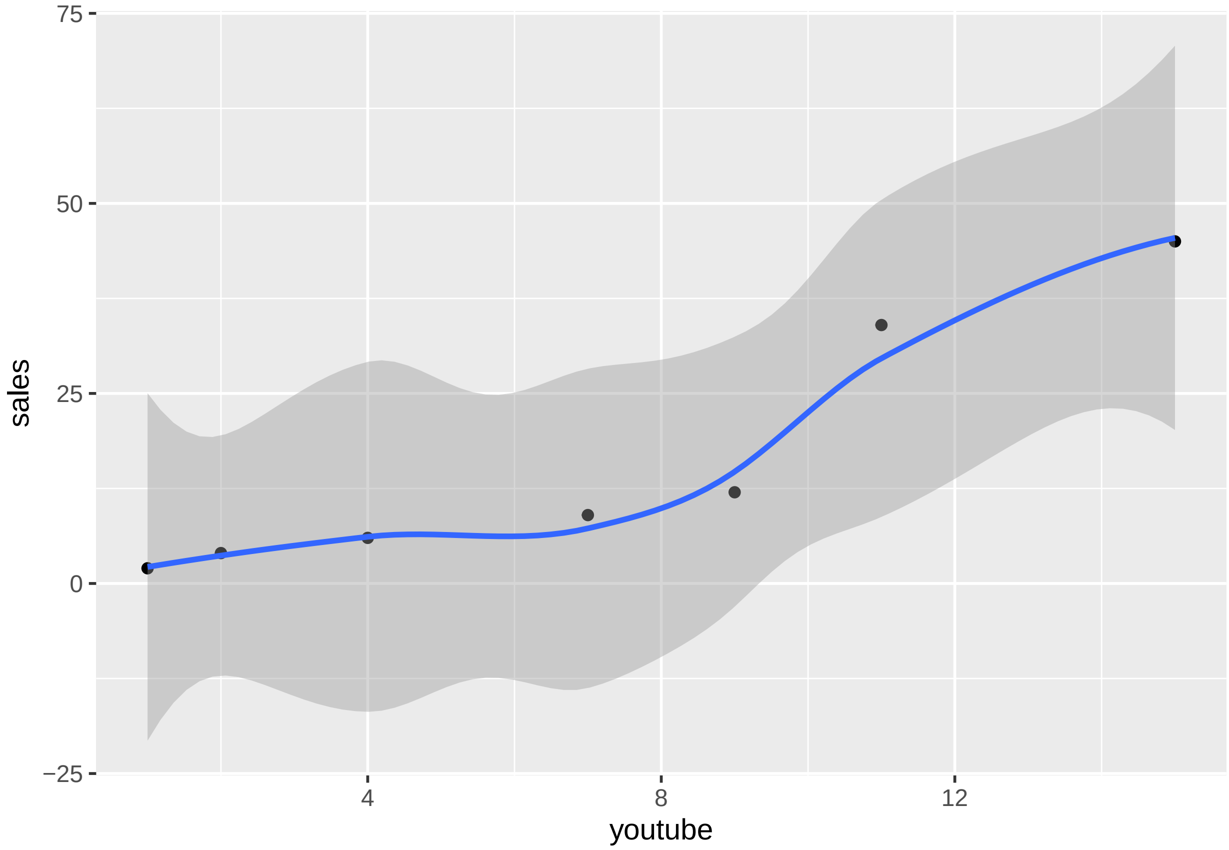
10.4. R code

```
# create input parameters
marketing=data.frame(sales=c(2,4,6,9,12,34,45),youtube=c(1,2,4,7,9,11,15))
```

10.4.1. Plotting the data as a scatter diagram

```
library(ggplot2)
ggplot(marketing, aes(x = youtube, y = sales)) +
  geom_point() +
  stat_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



10.4.2. Checking Association

```
cor.test(marketing$sales,marketing$youtube)
```

```
##
##  Pearson's product-moment correlation
##
## data:  marketing$sales and marketing$youtube
## t = 5.5176, df = 5, p-value = 0.002677
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5751106 0.9893519
## sample estimates:
##       cor
## 0.9267856
```

10.4.3. Create the OLS model

```
#fit a linear model
model <- lm(sales ~ youtube, data = marketing)
```

10.4.4. Model summary

```
summary(model)$coef
```

```
##              Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -5.363636  4.6607638 -1.150806 0.30185618
## youtube      3.051948  0.5531309  5.517587 0.00267729
```

10.4.5. Prediction using fitted models

```
library(dplyr)
newdata <- data.frame(youtube = c(1000,  2000, 3500))
model %>% predict(newdata)
```

```
##         1         2         3
##  3046.584  6098.532 10676.455
```

10.5. Results & discussions

From the given data, an OLR model is created using R function `lm()`. In our example, the fitted linear model is

$$Sales = -5.363636 + 3.051948(Youtube)$$

it can be seen that p-value of the F-statistic is 0.002, which is highly significant. This means that, at least, one of the predictor variables is significantly related to the outcome variable. So it is statistically reasonable to conclude that advertisement in Youtube significantly impact the sales. Finally the predicted returns are 3046.5846098.53210676.455 for respective investment 1000, 2000 and 3500 respectively.

11. Experiment 14: Construct a scatter plot to investigate the relationship between two variables.

11.1. Aim

1. To investigate relationship between two given variables.

11.2. Packages used and syntax of R methods

Functions from `stats` package (which is loaded by default). For plottting the scatter points, the `geom_point()` function from `ggplot2` package and `cor.test()` function from `stats` package are used.

11.3. Algorithm

- Step 1: Assign the inputs for required comparison
- Step 2: Create scatter plot using- `plot()` function in R
- Step 3: Report the observations
- Step 4: Confirm the observation using correlation analysis.

Case: Identify the relationship between sales units based on the advertising budget spent on youtube. The sales data with corresponding expenditure on advertisment in Youtube is shown below:

Sales(Y) :	2	4	6	9	12	34	45
Add expense(Youtube) :	1	2	4	7	9	11	15

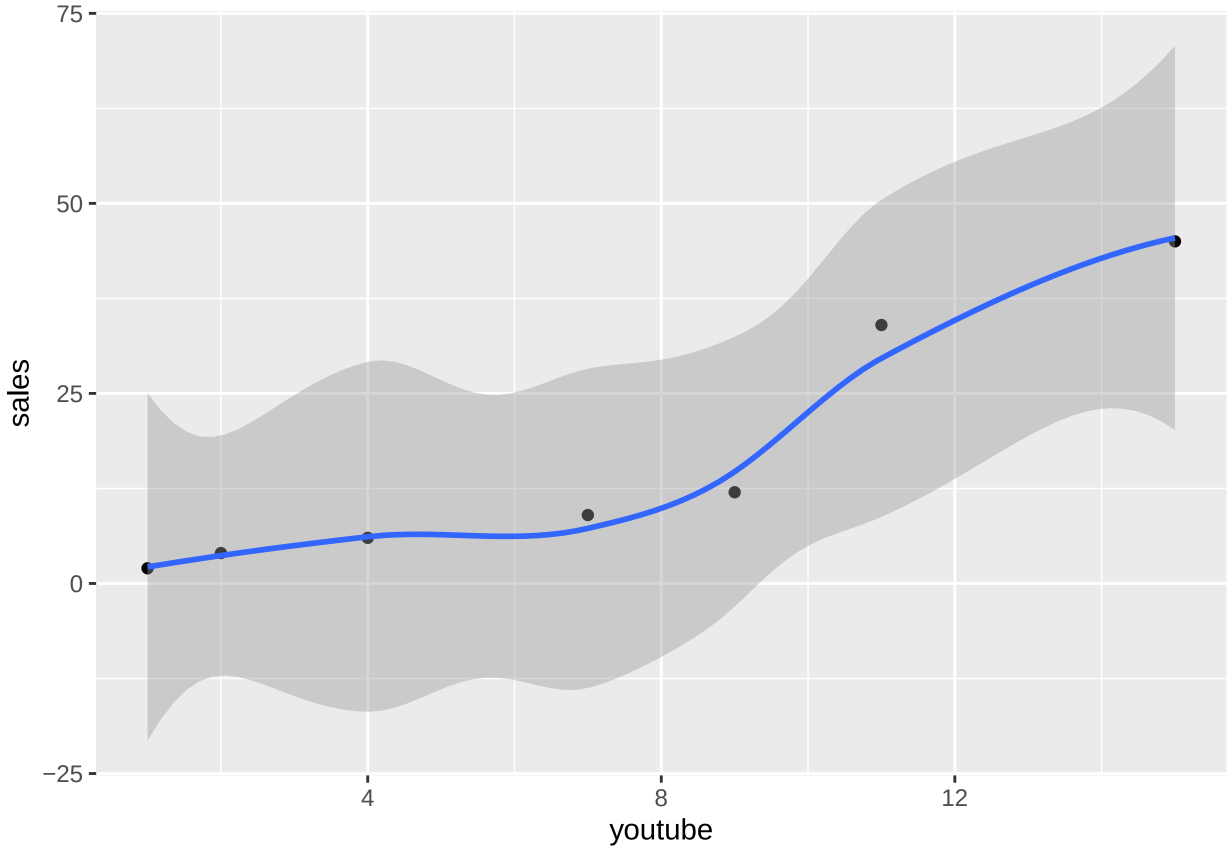
11.4. R code

```r
# create input parameters
marketing=data.frame(sales=c(2,4,6,9,12,34,45),youtube=c(1,2,4,7,9,11,15))
```

11.4.1. Plotting the data as a scatter diagram

```r
library(ggplot2)
ggplot(marketing, aes(x = youtube, y = sales)) +
  geom_point() +
  stat_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



11.4.2. Checking Association

```r
cor.test(marketing$sales,marketing$youtube)
```

```
##
##  Pearson's product-moment correlation
##
## data:  marketing$sales and marketing$youtube
## t = 5.5176, df = 5, p-value = 0.002677
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
```

```
##  0.5751106 0.9893519
## sample estimates:
##       cor
## 0.9267856
```

11.5. Results & discussions

The given variables are plotted using `ggplot2` package. From the plot it is clear that there is a linear relationship between the variables. The correlation coefficient is 0.9267856. So the linear association is highly significant at 5% significance level.

12. Experiment 15: Compute confidence intervals for the mean when the standard deviation is known

12.1. Aim

1. To Compute confidence intervals for the mean of a given dataset.

12.2. Packages used and syntax of R methods

Functions from `stats` package (which is loaded by default) are used.

12.3. Algorithm

- Step 1: Assign the inputs
- Step 2: Find the mean , variance and SE
- Step 3: Calculate a 95% confidence interval for mean using the formula $\bar{X} \pm 1.96SE$. Here $SE = \frac{SD}{\sqrt{n}}$
- Step 4: Return the confidence intervals (LCL and UCL)

Case: Calculate a 95% confidence interval for mean if $\bar{X} = 12, SD = 3, n = 30$

12.4. R code

```r
# create input parameters
xbar <- 12
stddev <- 3
n <- 30
```

12.4.1. Calculating 95% confidence interval for the mean

```r
SE=stddev/sqrt(n)
error <- qnorm(0.975)*SE
lower_bound <- xbar - error
upper_bound <- xbar + error
```

12.4.2. Display the LCL and UCL

```r
lower_bound
```

```
## [1] 10.92648
```

```
upper_bound
```

```
## [1] 13.07352
```

12.5. Results & discussions

The 95% confidence interval for mean is calculated as $(10.9264835, 13.0735165)$.

References

JJ Allaire, Yihui Xie, Christophe Dervieux, R Foundation, Hadley Wickham, Journal of Statistical Software, Ramnath Vaidyanathan, Association for Computing Machinery, Carl Boettiger, Elsevier, Karl Broman, Kirill Mueller, Bastiaan Quast, Randall Pruim, Ben Marwick, Charlotte Wickham, Oliver Keyes, Miao Yu, Daniel Emaasit, Thierry Onkelinx, Alessandro Gasparini, Marc-Andre Desautels, Dominik Leutnant, MDPI, and Taylor and Francis. *rticles: Article Formats for R Markdown*, 2022. URL https://CRAN.R-project.org/package=rticles. R package version 0.23.