R programming ~ Exercise 6-12

March 31, 2025

1 UNIVARIATE NORMAL DENSITY

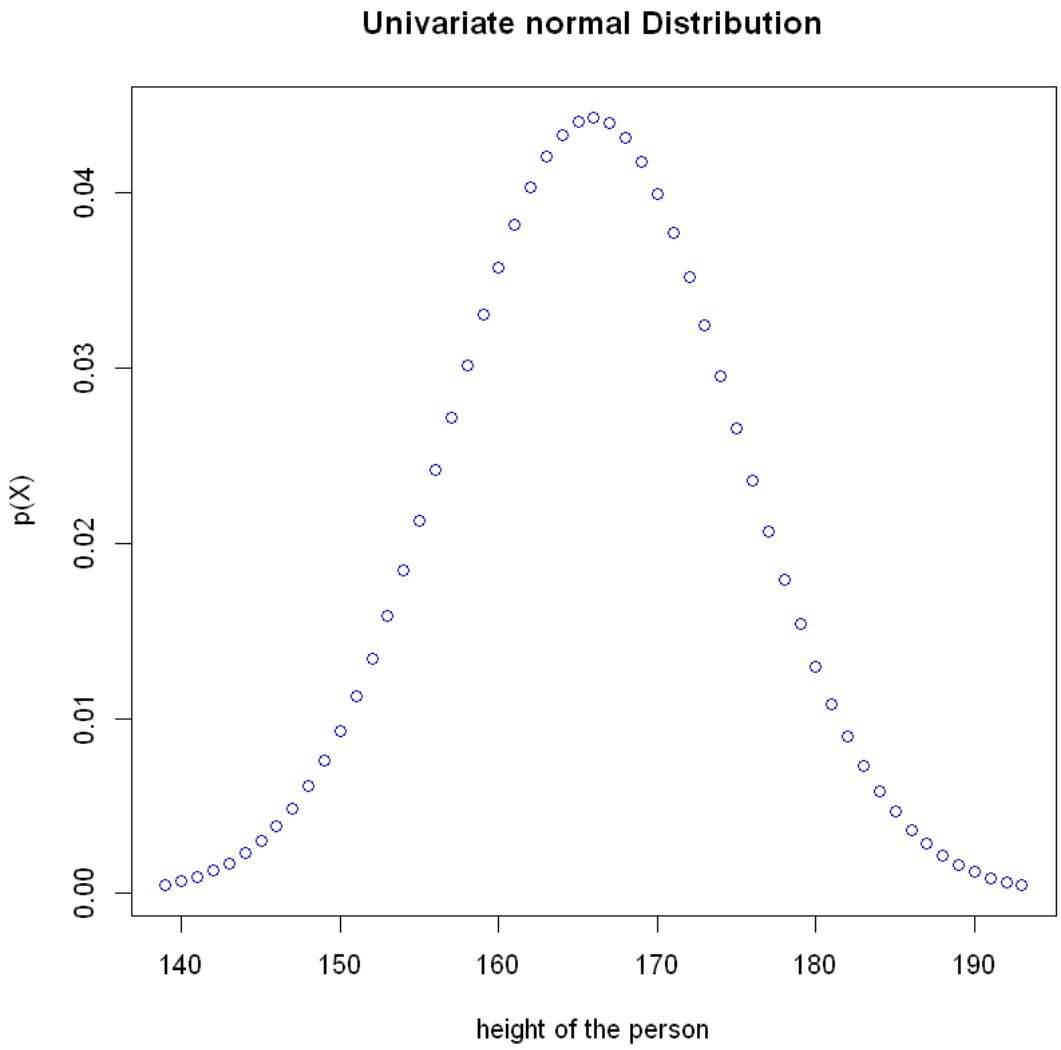
[1]:
```
uvtrain <- function(hdata)
{
    xv=vector(mode="numeric",length=0)
    pv=vector(mode="numeric",length=0)
    hmin=min(hdata)-15
    hmax=max(hdata)+15
    m=mean(hdata)
    v=var(hdata)
    cat("Mean: ",m,"\n")
    cat("Variance: ",v)
    for(x in hmin:hmax)
    {
        p=dnorm(x,m,sqrt(v))
        xv=c(xv,x)
        pv=c(pv,p)

    }
    plot(xv,pv,xlab="height of the person",ylab="p(X)",main="Univariate normal␣
 ↪Distribution",col="blue")
    return(c(m,v))
}
```

[2]:
```
uvtest <- function(m,v,ht)
{
  pt=dnorm(ht,m,sqrt(v))
  if (pt >= 0.00005)
  cat("the given height of the person is adult")
  else
  cat("the give height of the person is not adult")

}
```

[3]:
```
hdata <- c(165, 170, 160, 154, 175, 155, 167, 177, 158, 178)
```

[4]:
```
mv = uvtrain(hdata)
```

```
Mean:  165.9
Variance:  80.98889
```



[5]:
```
ht=as.numeric(readline(prompt="Enter the height of the person:"))
uvtest(mv[1],mv[2],ht)
```

```
the given height of the person is adult
```

2 MULTIVARIATE NORMAL DENSITY

[ ]:
```
install.packages("mvtnorm")
library(mvtnorm)
```

[ ]:
```
install.packages("rgl")
```

[ ]:
```
library(rgl)
mvtrain <- function(hwdata)
{
  hv=vector(mode="numeric",length=0)
  wv=vector(mode="numeric",length=0)
  pv=vector(mode="numeric",length=0)

  hmin=min(hwdata[,1])-15
  hmax=max(hwdata[,1])+15
  wmin=min(hwdata[,2])-15
  wmax=max(hwdata[,2])+15
  m=colMeans(hwdata)
  cv=cov(hwdata)
  cat("Mean Vector: ",m,"\n")
  cat("Covariance Matrix: ",cv)
  hv=rep(hmin:hmax,each=length(hmin:hmax))
  wv=rep(wmin:wmax,times=length(hmin:hmax))
  grid_points=cbind(hv,wv)
  pv=dmvnorm(grid_points,m,cv)
  plot3d(x=hv,y=wv,z=pv,xlab="height",ylab="weight",main="Multivariate normal␣
 ↪Distribution",col="blue")
  return(list(mv=m,cv=cv))
}
```

[8]:
```
mvtest <- function(mvdata,hwdata)
{
    mv=mvdata$mv
    cv=mvdata$cv
    pt=dmvnorm(hwdata,mv,cv)
    if(pt>=0.00005)
    cat("the given height and weight of the person is adult")
    else
    cat("the give height and weight of the person is not adult")
}
```

[9]:
```
hwdata <- cbind(
  c(165, 170, 160, 154, 175, 155, 167, 177, 158, 178),  # Heights
  c(78, 71, 60, 53, 72, 51, 64, 65, 55, 69)             # Weights
)
```

[10]:
```
mvdata = mvtrain(hwdata)
```

```
Mean Vector:  165.9 63.8
Covariance Matrix:  80.98889 58.64444 58.64444 80.17778
```

```
Warning message in cbind(hv, wv):
"number of rows of result is not a multiple of vector length (arg 1)"
```

[11]:
```
ht= as.numeric(readline(prompt ='Enter the Height of person ='))
wt= as.numeric(readline(prompt ='Enter the Weight of person ='))
hwdata = c(ht,wt)
mvtest(mvdata,hwdata)
```

```
the given height and weight of the person is adult
```

3 LINEAR AND NON LINEAR ANALYSIS

1. Analysis of the Positive Relationship between Height and Weight of Women Using Correlation Coefficients

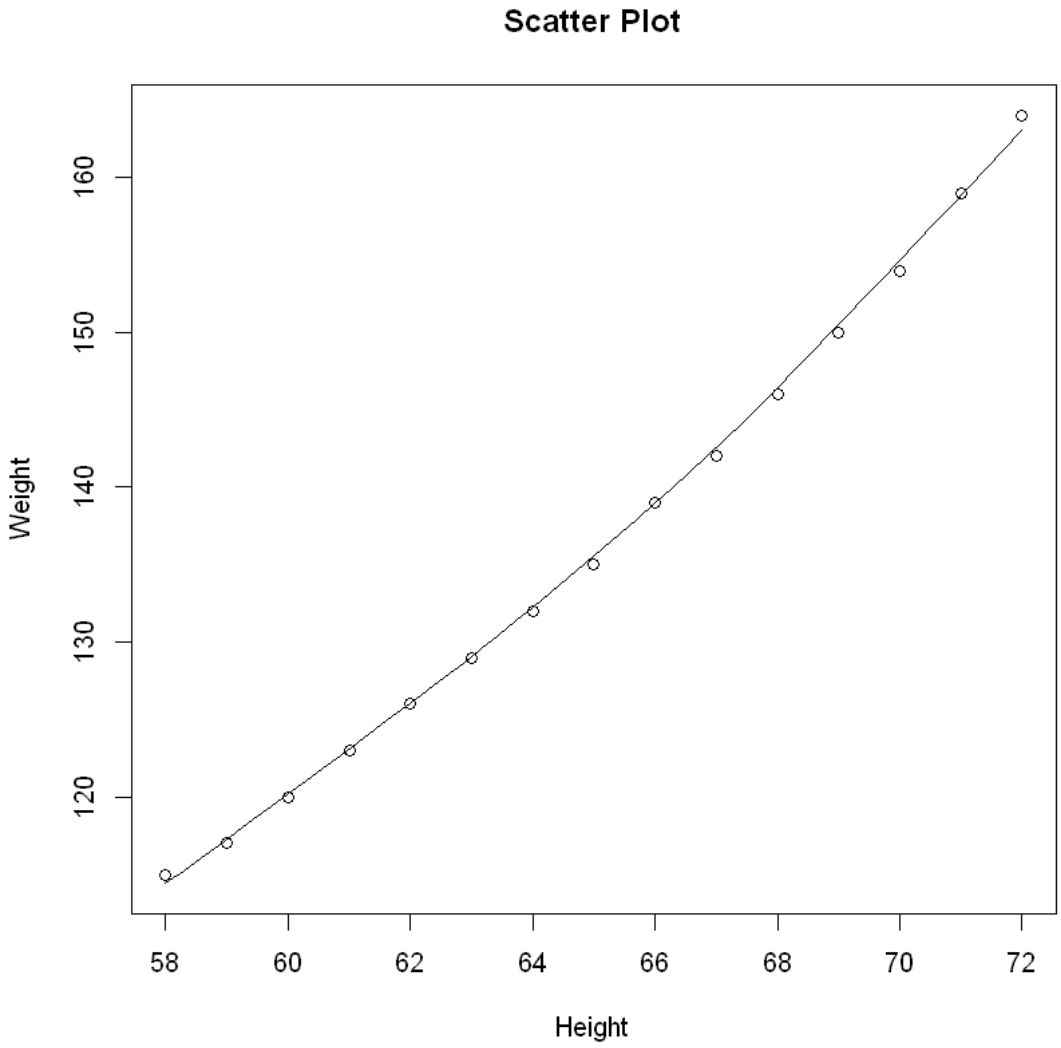
[12]:
```
# Load the Women's dataset (built-in in R)
data(women)

# Display the first 15 rows
head(women)
```

A data.frame: 6 × 2

	height <dbl>	weight <dbl>
1	58	115
2	59	117
3	60	120
4	61	123
5	62	126
6	63	129

[13]:
```
library(ggplot2)
scatter.smooth(women$height, women$weight,
               main = "Scatter Plot",
               xlab = "Height",
               ylab = "Weight")
```

```
Warning message:
"package 'ggplot2' was built under R version 4.4.3"
```



[14]:
```
# Another way to get the full covariance matrix
cm2 <- cov(women)
print("Full Covariance Matrix:")
print(cm2)
```

```
[1] "Full Covariance Matrix:"
       height   weight
height     20  69.0000
weight     69 240.2095
```

[15]:
```
cm2[1,2]
```

69

[16]:
```
# Compute the correlation matrix directly
cc2 <- cor(women)
print("Pearson's Correlation Matrix:")
print(cc2)
```

```
[1] "Pearson's Correlation Matrix:"
          height    weight
height 1.0000000 0.9954948
weight 0.9954948 1.0000000
```

[17]:
```
# Compute Spearman's rank correlation
cc3 <- cor(women, method = "spearman")
print("Spearman's Correlation Coefficients:")
print(cc3)
```

```
[1] "Spearman's Correlation Coefficients:"
       height weight
height      1      1
weight      1      1
```

[18]:
```
# Determine Relationship Type
if (cc2[1,2] > 0) {
  print("The relationship between Women's Height and Weight is Positive")
} else {
  print("The relationship between Women's Height and Weight is Negative")
}
```

```
[1] "The relationship between Women's Height and Weight is Positive"
```

2. Analysis of the Negative Relationship Between Weight of Cars and Mileage Using Correlation coefficients

[19]:
```
# Load the mtcars dataset (built-in in R)
data(mtcars)

# Display the dataset (first 32 rows)
head(mtcars,3)
```

A data.frame: 3 × 11

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1

[20]:
```
# Finding Covariance between Car Weight (wt) and Mileage (mpg)
co <- cov(mtcars$wt, mtcars$mpg)
cat("Covariance:",co)
```

```
Covariance: -5.116685
```

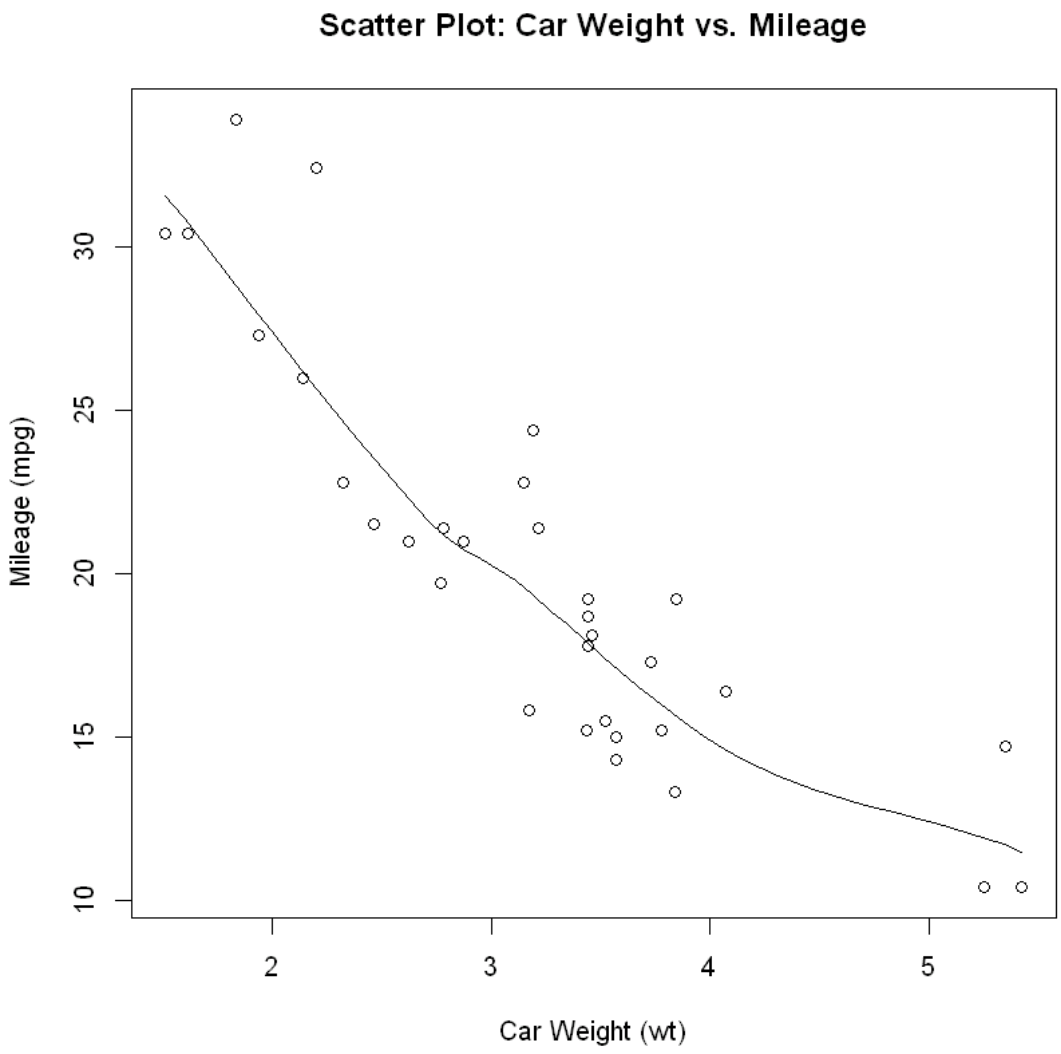
[21]:
```r
# Finding the Pearson Correlation Coefficient
cc <- cor(mtcars$wt, mtcars$mpg)
cat("Pearson's Correlation Coefficient:",cc)
```

```
Pearson's Correlation Coefficient: -0.8676594
```

[22]:
```r
# Finding the Spearman Correlation Coefficient
ccs <- cor(mtcars$wt, mtcars$mpg, method = "spearman")
cat("Spearman's Correlation Coefficient:",ccs)
```

```
Spearman's Correlation Coefficient: -0.886422
```

[23]:
```r
# Scatter Plot
library(ggplot2)
scatter.smooth(mtcars$wt, mtcars$mpg,
               main = "Scatter Plot: Car Weight vs. Mileage",
               xlab = "Car Weight (wt)",
               ylab = "Mileage (mpg)")
```



[24]:
```
# Determine Relationship Type
if (cc > 0) {
  print("The relationship between Car Weight and Mileage is Positive")
} else {
  print("The relationship between Car Weight and Mileage is Negative")
}
```

```
[1] "The relationship between Car Weight and Mileage is Negative"
```

4 MULTIPLE CORRELATION COEFFICIENTS

[ ]:
```
install.packages("corrgram")
```

[26]:
```
# Load the built-in iris dataset
data(iris)

# Display first few rows of each species
head(iris[c(1:5,51:55,101:105),],n = 15)
```

A data.frame: 15 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.5	1.5	versicolor
53	6.9	3.1	4.9	1.5	versicolor
54	5.5	2.3	4.0	1.3	versicolor
55	6.5	2.8	4.6	1.5	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica
103	7.1	3.0	5.9	2.1	virginica
104	6.3	2.9	5.6	1.8	virginica
105	6.5	3.0	5.8	2.2	virginica

[27]:
```
# Remove species column to analyze numerical data only
iris.nospecies <- iris[, -5]
```

[28]:
```
# Construct the Covariance Matrix
coi <- cov(iris.nospecies)
cat("Covariance Matrix:")
coi
```

```
Covariance Matrix:
```

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.6856935	-0.0424340	1.2743154	0.5162707
Sepal.Width	-0.0424340	0.1899794	-0.3296564	-0.1216394
Petal.Length	1.2743154	-0.3296564	3.1162779	1.2956094
Petal.Width	0.5162707	-0.1216394	1.2956094	0.5810063

[29]:
```
# Compute Multiple Pearson's Correlation Coefficients
cci <- cor(iris.nospecies)
cat("Multiple Pearson's Correlation Coefficients:")
cci
```

```
Multiple Pearson's Correlation Coefficients:
```

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.0000000	-0.1175698	0.8717538	0.8179411
Sepal.Width	-0.1175698	1.0000000	-0.4284401	-0.3661259
Petal.Length	0.8717538	-0.4284401	1.0000000	0.9628654
Petal.Width	0.8179411	-0.3661259	0.9628654	1.0000000

[30]:
```
# Compute Multiple Spearman Correlation Coefficients
ccs <- cor(iris.nospecies, method = "spearman")
cat("Multiple Spearman's Correlation Coefficients:")
ccs
```

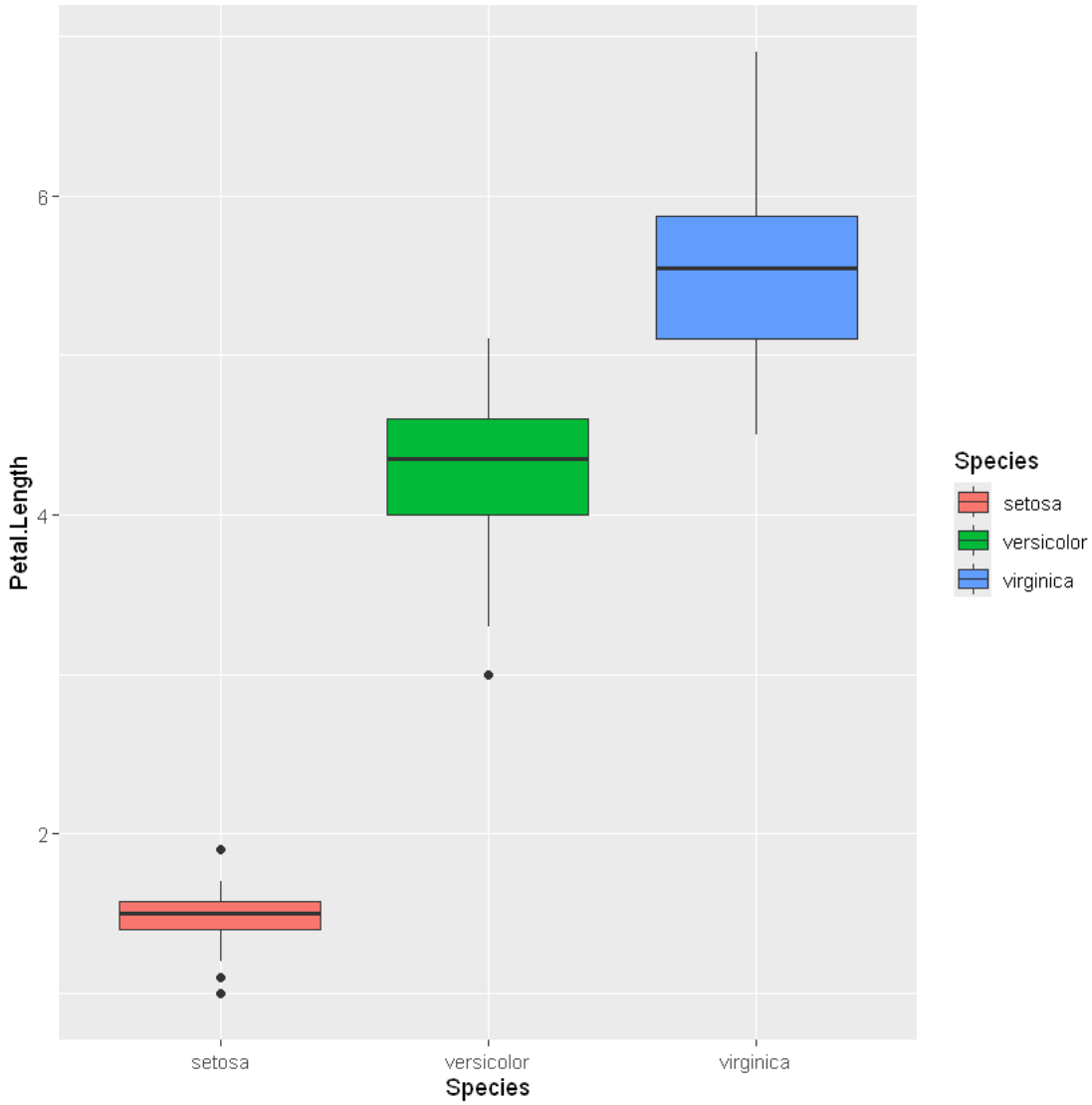
```
Multiple Spearman's Correlation Coefficients:
```

A matrix: 4 × 4 of type dbl

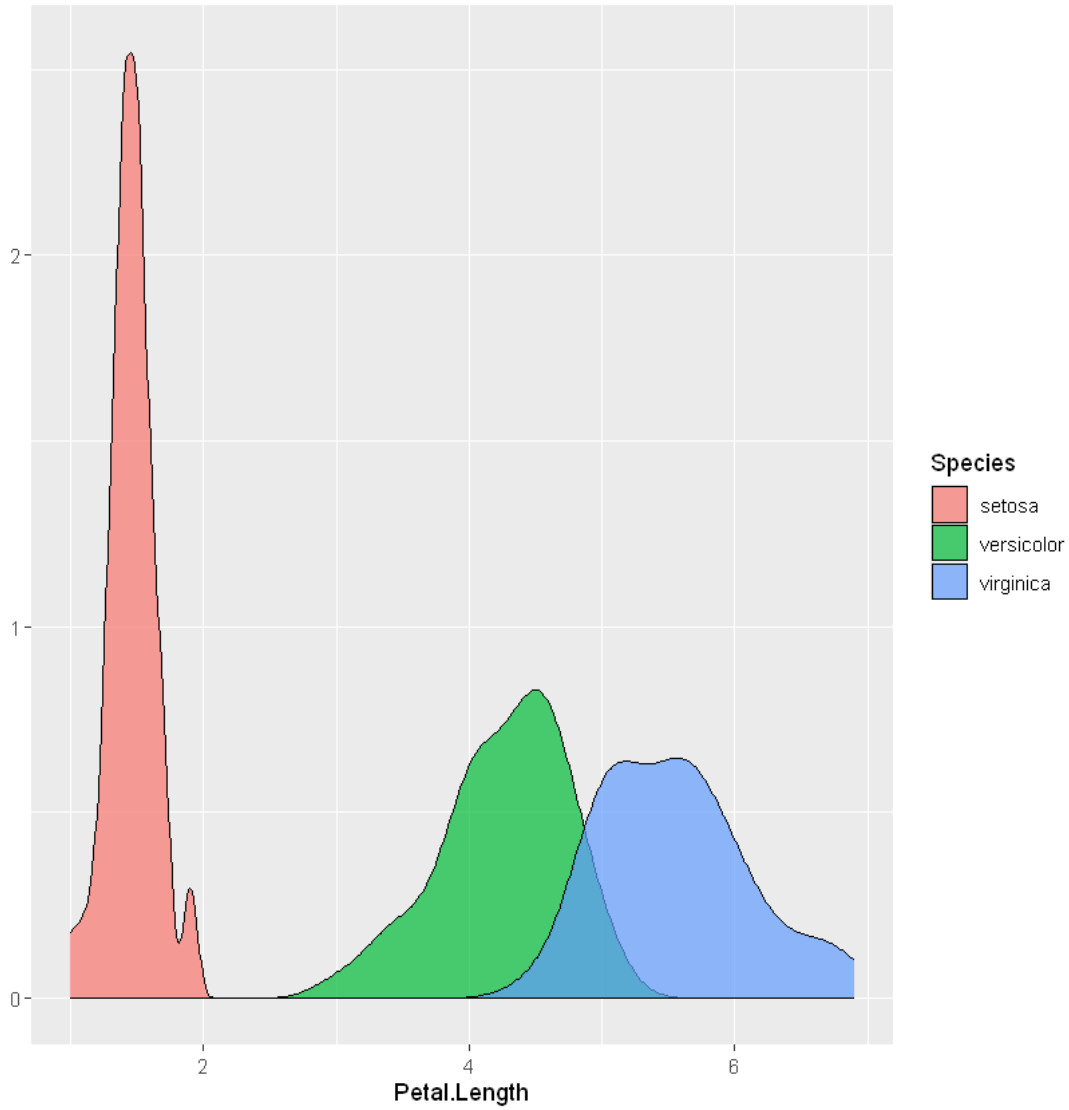
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.0000000	-0.1667777	0.8818981	0.8342888
Sepal.Width	-0.1667777	1.0000000	-0.3096351	-0.2890317
Petal.Length	0.8818981	-0.3096351	1.0000000	0.9376668
Petal.Width	0.8342888	-0.2890317	0.9376668	1.0000000

[31]:
```
#Analysis of Iris Data Using Box Plot
qplot(Species, Petal.Length, data=iris, geom="boxplot", fill=Species)
```

```
Warning message:
"`qplot()` was deprecated in ggplot2 3.4.0."
```



[32]:
```
#Analysis of Iris Data Using Normal Density
qplot (Petal.Length, data=iris, geom="density", alpha=I(.7),fill=Species)
```

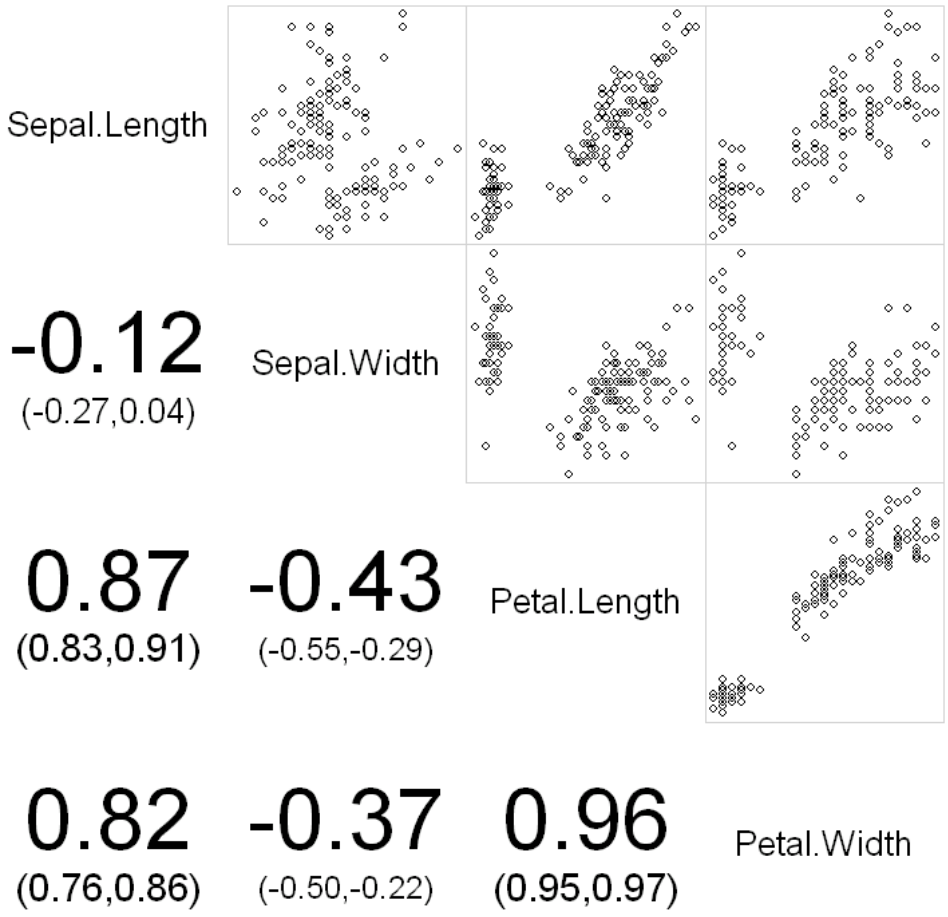


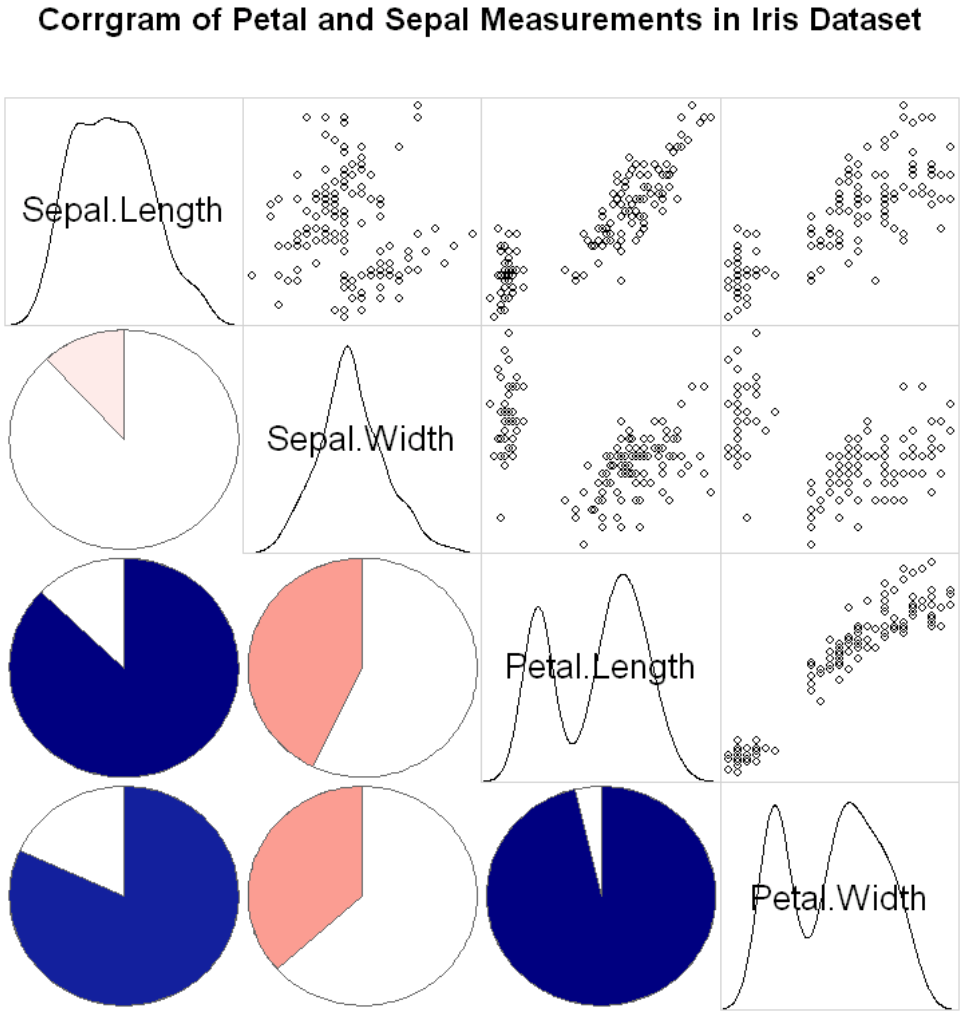
[33]:
```r
# Checking relationships based on correlation matrix
if (cci[4, 1] > 0) {
  print("Relationship between Petal Width and Sepal Length is Positive")
} else {
  print("Relationship between Petal Width and Sepal Length is Negative")
}

if (cci[2, 1] > 0) {
  print("Relationship between Sepal Width and Sepal Length is Positive")
} else {
  print("Relationship between Sepal Width and Sepal Length is Negative")
}
```

```
[1] "Relationship between Petal Width and Sepal Length is Positive"
[1] "Relationship between Sepal Width and Sepal Length is Negative"
```

[34]:
```r
library(corrgram)
# Correlation Matrix Visualization
corrgram(iris, lower.panel = panel.conf, upper.panel = panel.pts)

# Overlapping Density Plot for Three Species
corrgram(iris, lower.panel = panel.pie, upper.panel = panel.pts,
         diag.panel = panel.density,
         main = "Corrgram of Petal and Sepal Measurements in Iris Dataset")
```

```
Warning message:
"package 'corrgram' was built under R version 4.4.3"
Warning message in par(usr):
"argument 1 does not name a graphical parameter"
Warning message in par(usr):
"argument 1 does not name a graphical parameter"
Warning message in par(usr):
"argument 1 does not name a graphical parameter"
Warning message in par(usr):
"argument 1 does not name a graphical parameter"
Warning message in par(usr):
"argument 1 does not name a graphical parameter"
Warning message in par(usr):
"argument 1 does not name a graphical parameter"
Warning message in par(usr):
"argument 1 does not name a graphical parameter"
```


Sepal.Length
-0.12
(-0.27,0.04)
Sepal.Width
0.87
(0.83,0.91)
-0.43
(-0.55,-0.29)
Petal.Length
0.82
(0.76,0.86)
-0.37
(-0.50,-0.22)
0.96
(0.95,0.97)
Petal.Width



5 BAYE'S RULE

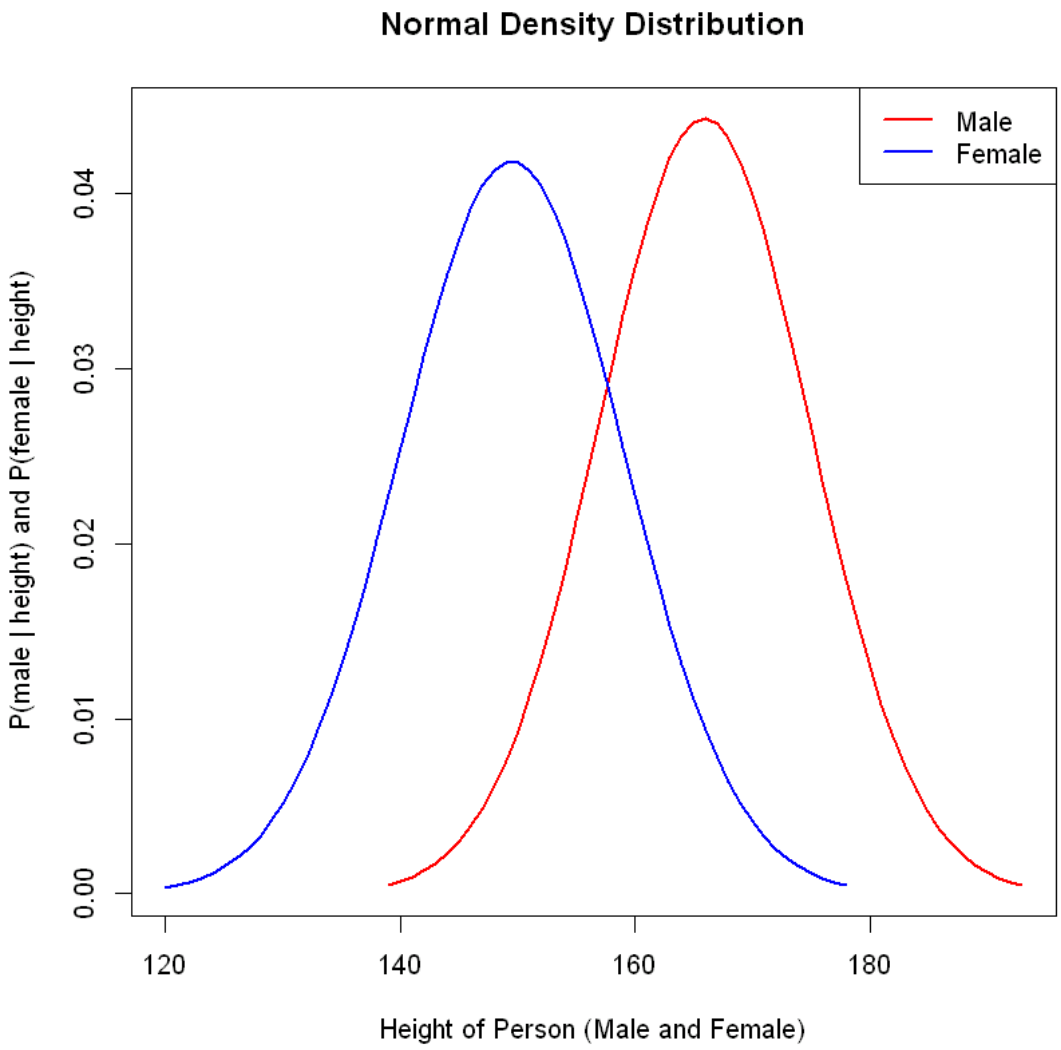
[35]:
```r
# Function to train the model based on height distributions of males and females
uvtrain <- function(hm, hf) {
  hmin_m <- min(hm) - 15
  hmax_m <- max(hm) + 15
  hmin_f <- min(hf) - 15
  hmax_f <- max(hf) + 15

  mm <- mean(hm)  # Mean of male heights
  vm <- var(hm)   # Variance of male heights
  cat("Mean of Male Height:", mm, "\n")
```

```
  cat("Variance of Male Height:", vm, "\n")

  mf <- mean(hf)  # Mean of female heights
  vf <- var(hf)   # Variance of female heights
  cat("Mean of Female Height:", mf, "\n")
  cat("Variance of Female Height:", vf, "\n")

  # Generate height ranges
  hmm <- hmin_m:hmax_m
  hff <- hmin_f:hmax_f

  # Use dnorm() to compute probabilities
  pmh <- dnorm(hmm, mean = mm, sd = sqrt(vm))
  pfh <- dnorm(hff, mean = mf, sd = sqrt(vf))

  # Plot the normal densities
  plot(hmm, pmh, type = "l", col = "red", lwd = 2,
       xlim = c(min(hff), max(hmm)),
       xlab = "Height of Person (Male and Female)",
       ylab = "P(male | height) and P(female | height)",
       main = "Normal Density Distribution")
  lines(hff, pfh, col = "blue", lwd = 2)
  legend("topright", legend = c("Male", "Female"), col = c("red", "blue"), lty␣
 ↪= 1, lwd = 2)

return(c(mm,vm,mf,vf))
}
```

[36]:
```
# Bayes Rule Testing Function Using Normal Density
uvtest <- function(mm, vm, mf, vf, ht) {
  # Probability of Male given Height using dnorm()
  pm <- dnorm(ht, mean = mm, sd = sqrt(vm))

  # Probability of Female given Height using dnorm()
  pf <- dnorm(ht, mean = mf, sd = sqrt(vf))

  if (pm > pf) {
    print("The given Height of the Person is Male")
  } else {
    print("The given Height of the Person is Female")
  }
}
```

[37]:
```
# Train Function Call
hm <- c(165, 170, 160, 154, 175, 155, 167, 177, 158, 178)
hf <- c(140, 145, 149, 152, 157, 135, 139, 160, 155, 163)
mv <- uvtrain(hm, hf)
```

```
Mean of Male Height: 165.9
Variance of Male Height: 80.98889
Mean of Female Height: 149.5
Variance of Female Height: 90.72222
```



[38]:
```
# Testing Function Call
ht <- as.numeric(readline(prompt = 'Enter the height of the person for␣
 ↪prediction: '))
mm <- mv[1]
vm <- mv[2]
mf <- mv[3]
vf <- mv[4]
uvtest(mm, vm, mf, vf, ht)
```

```
[1] "The given Height of the Person is Male"
```

6 RAINFALL PREDICTION

[39]:
```r
# Load necessary library
library(ggplot2)
```

[40]:
```r
# Check precip data
head(precip)
```

Mobile 67 **Juneau** 54.7 **Phoenix** 7 **Little Rock** 48.5 **Los Angeles** 14 **Sacramento** 17.2

[41]:
```r
is.vector(precip)
```

TRUE

[42]:
```r
mean(precip)
```

34.8857142857143

[43]:
```r
# Function to compute t-statistic
t.statistic <- function(thesample, thepopulation) {
  numerator <- mean(thesample) - thepopulation
  denominator <- sd(thesample) / sqrt(length(thesample))
  t.stat <- numerator / denominator
  return(t.stat)
}
```

[44]:
```r
# Generate a large population with mean 38
population.precipitation <- rnorm(100000, mean = 38,sd = 10)  # Added standard␣
 ↪deviation
```

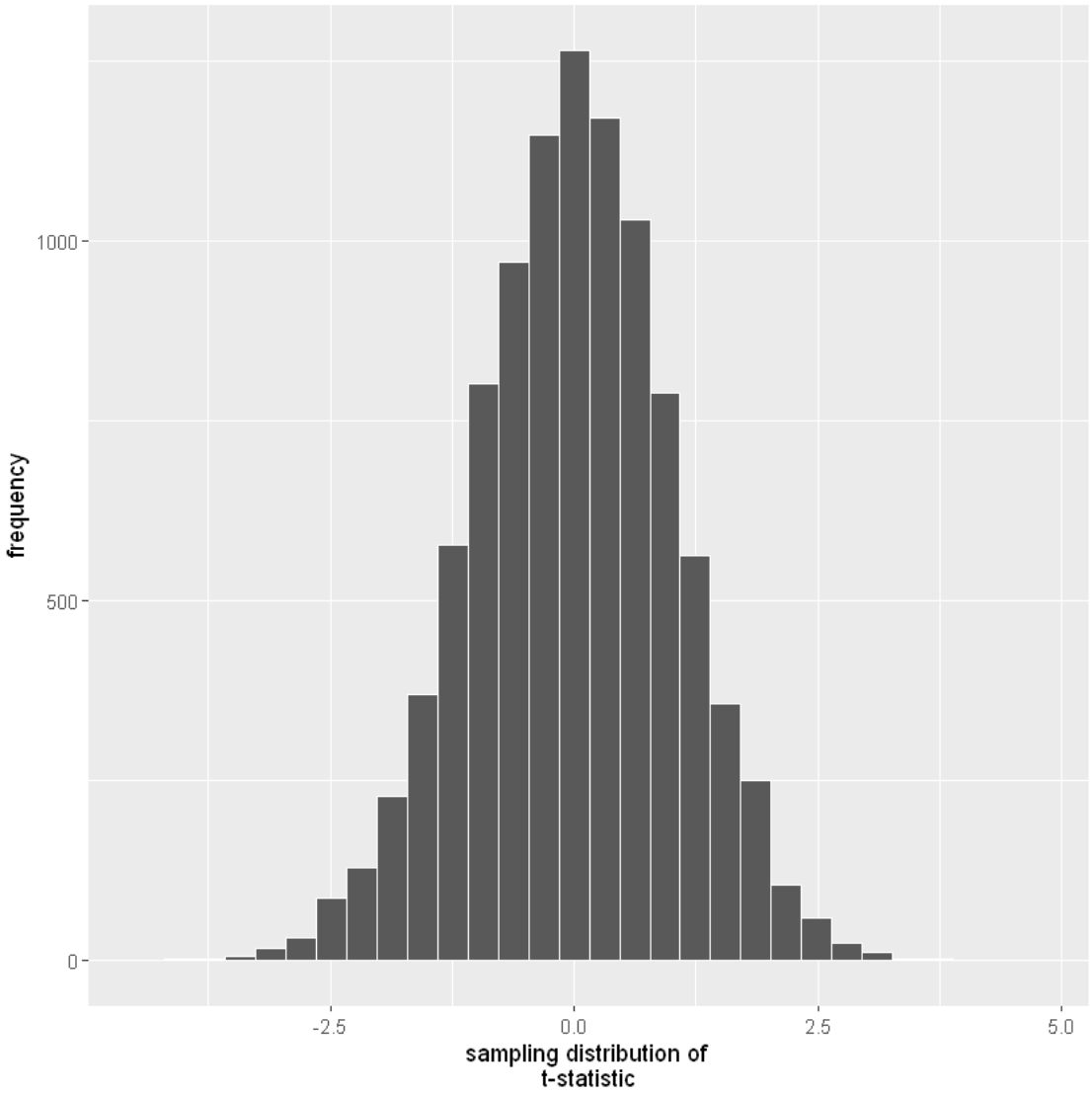
[45]:
```r
# Initialize vector for t-statistics
t.stats <- numeric(10000)
```

[46]:
```r
# Compute t-statistics using t.test() for 10,000 samples
for (i in 1:10000) {
  a.sample <- sample(population.precipitation, 70)
  t.stats[i] <- t.test(a.sample, mu = 38)$statistic  # Using inbuilt function
}
```

[47]:
```r
# Create histogram of t-statistics
tmpdata <- data.frame(vals = t.stats)
```

[48]:
```r
qplot(vals, data=tmpdata, geom="histogram",color=I("white"),xlab="sampling␣
 ↪distribution of
t-statistic",ylab="frequency")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



[49]:
```
# Perform t-tests on actual precip data
t.test(precip, mu = 38)  # Perform one-sample t-test
```

```
        One Sample t-test

data:  precip
t = -1.901, df = 69, p-value = 0.06148
alternative hypothesis: true mean is not equal to 38
95 percent confidence interval:
 31.61748 38.15395
sample estimates:
mean of x
```

```
34.88571
```

[50]:
```
qt(.025, df=69)
```

```
-1.99494541510724
```

[51]:
```
t.test(precip, mu = 38, alternative = "less")  # One-tailed t-test (left-tailed)
```

```
        One Sample t-test

data:  precip
t = -1.901, df = 69, p-value = 0.03074
alternative hypothesis: true mean is less than 38
95 percent confidence interval:
     -Inf 37.61708
sample estimates:
mean of x
 34.88571
```

[ ]:
```
t.statistic(precip, population.precipitation)
```

7 TESTING TWO MEANS

[ ]:
```
install.packages("heplots")
```

[ ]:
```
# Load necessary libraries
library(heplots)
library(ggplot2)
```

[54]:
```
# Display first few rows
head(WeightLoss)

# Count the number of observations in each group
table(WeightLoss$group)
```

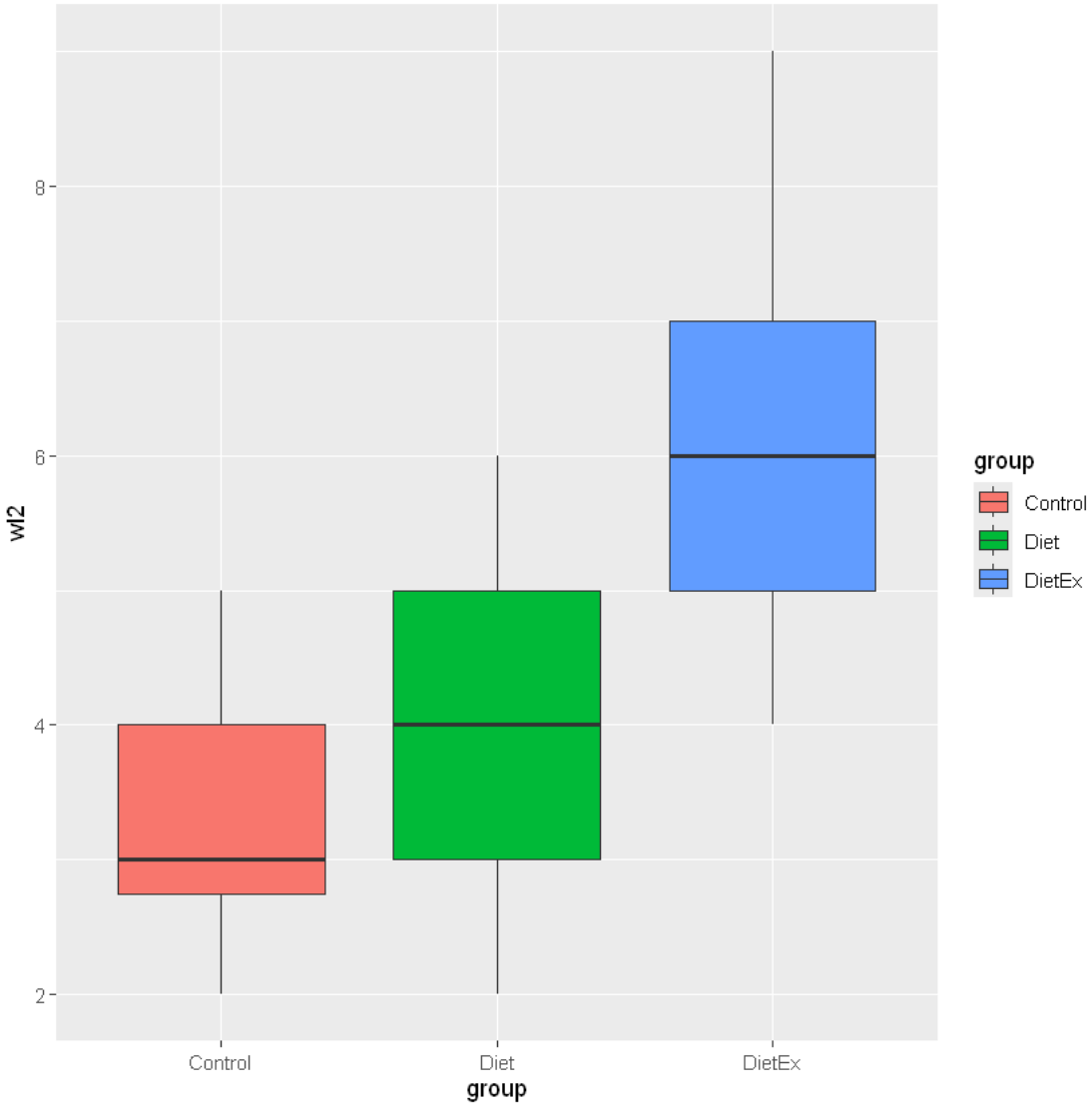
A data.frame: 6 × 7

	group <fct>	wl1 <int>	wl2 <int>	wl3 <int>	se1 <int>	se2 <int>	se3 <int>
1	Control	4	3	3	14	13	15
2	Control	4	4	3	13	14	17
3	Control	4	3	1	17	12	16
4	Control	3	2	1	11	11	12
5	Control	5	3	2	16	15	14
6	Control	6	5	4	17	18	18

```
Control    Diet  DietEx
```

```
12       12       10
```

[55]:
```
qplot(group, wl2, data=WeightLoss, geom="boxplot", fill=group)
```



[56]:
```
the.anova <- aov(wl2 ~ group, data=WeightLoss)
```

[57]:
```
summary(the.anova)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
group        2  45.28  22.641   13.37 6.49e-05 ***
Residuals   31  52.48   1.693
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[58]:
```
pairwise.t.test(WeightLoss$wl2, WeightLoss$group, p.adjust.method =␣
↪"bonferroni")
```

```
        Pairwise comparisons using t tests with pooled SD

data:  WeightLoss$wl2 and WeightLoss$group

       Control Diet
Diet   0.8418  -
DietEx 7.1e-05 0.0014

P value adjustment method: bonferroni
```

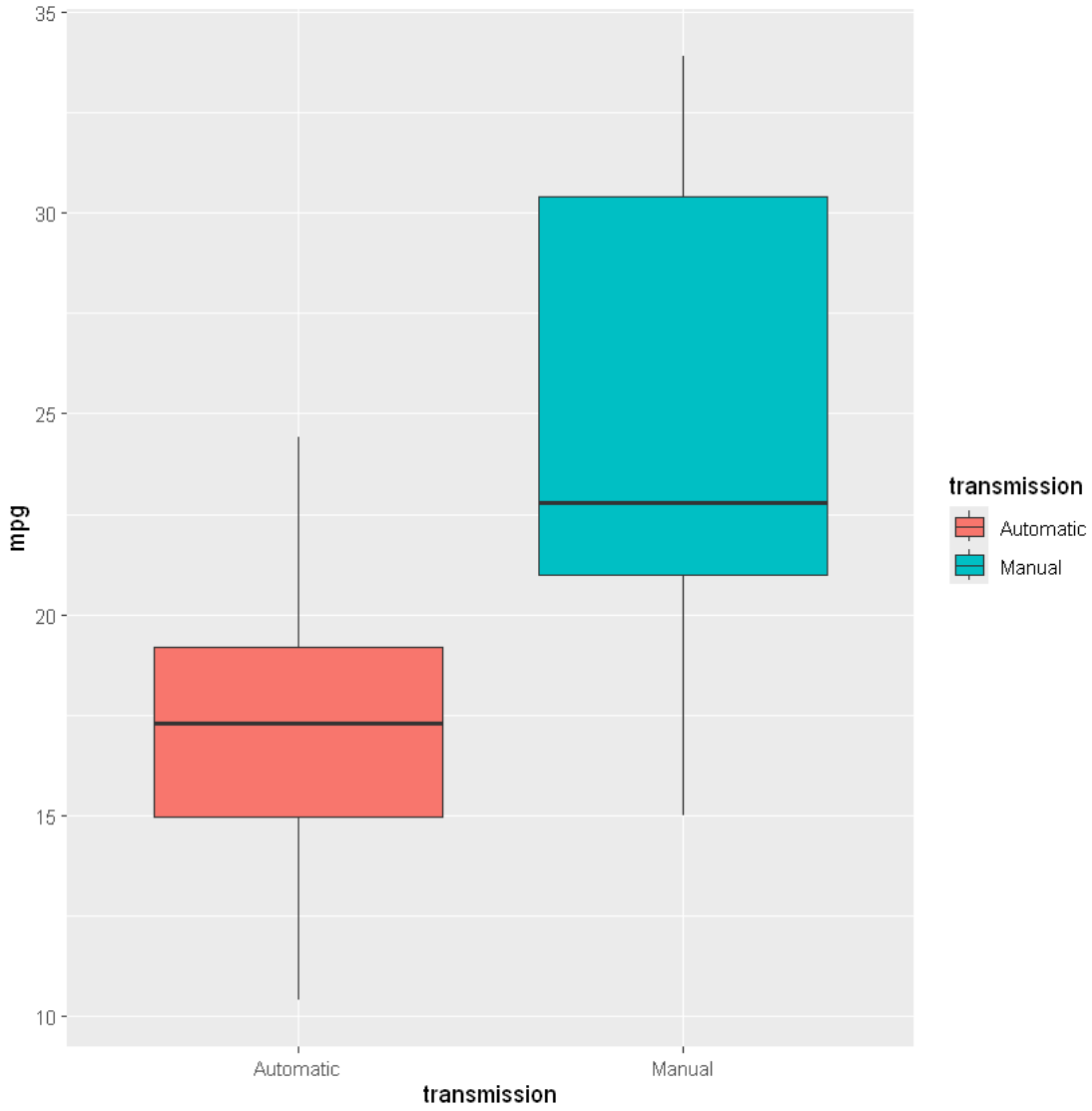
[59]:
```
# Analyzing the mtcars dataset
# Compute mean mpg for automatic and manual transmissions
mean(mtcars$mpg[mtcars$am == 0])  # Automatic
mean(mtcars$mpg[mtcars$am == 1])  # Manual
```

17.1473684210526

24.3923076923077

[60]:
```
# Create a copy of mtcars and add a factor variable for transmission
mtcars.copy <- mtcars
mtcars.copy$transmission <- factor(ifelse(mtcars$am == 0, "Automatic",␣
↪"Manual"))
```

[61]:
```
qplot(transmission, mpg, data=mtcars.copy,geom="boxplot", fill=transmission)
```



[62]:
```
# Alternative way to perform t-test directly on the dataset
t.test(mpg ~ am, data = mtcars, alternative = "less")
```

```
        Welch Two Sample t-test

data:  mpg by am
t = -3.7671, df = 18.332, p-value = 0.0006868
alternative hypothesis: true difference in means between group 0 and group 1 is␣
 ↪less than 0
95 percent confidence interval:
      -Inf -3.913256
sample estimates:
```

```
mean in group 0 mean in group 1
       17.14737        24.39231
```

8 SIMPLE LINEAR REGRESSION

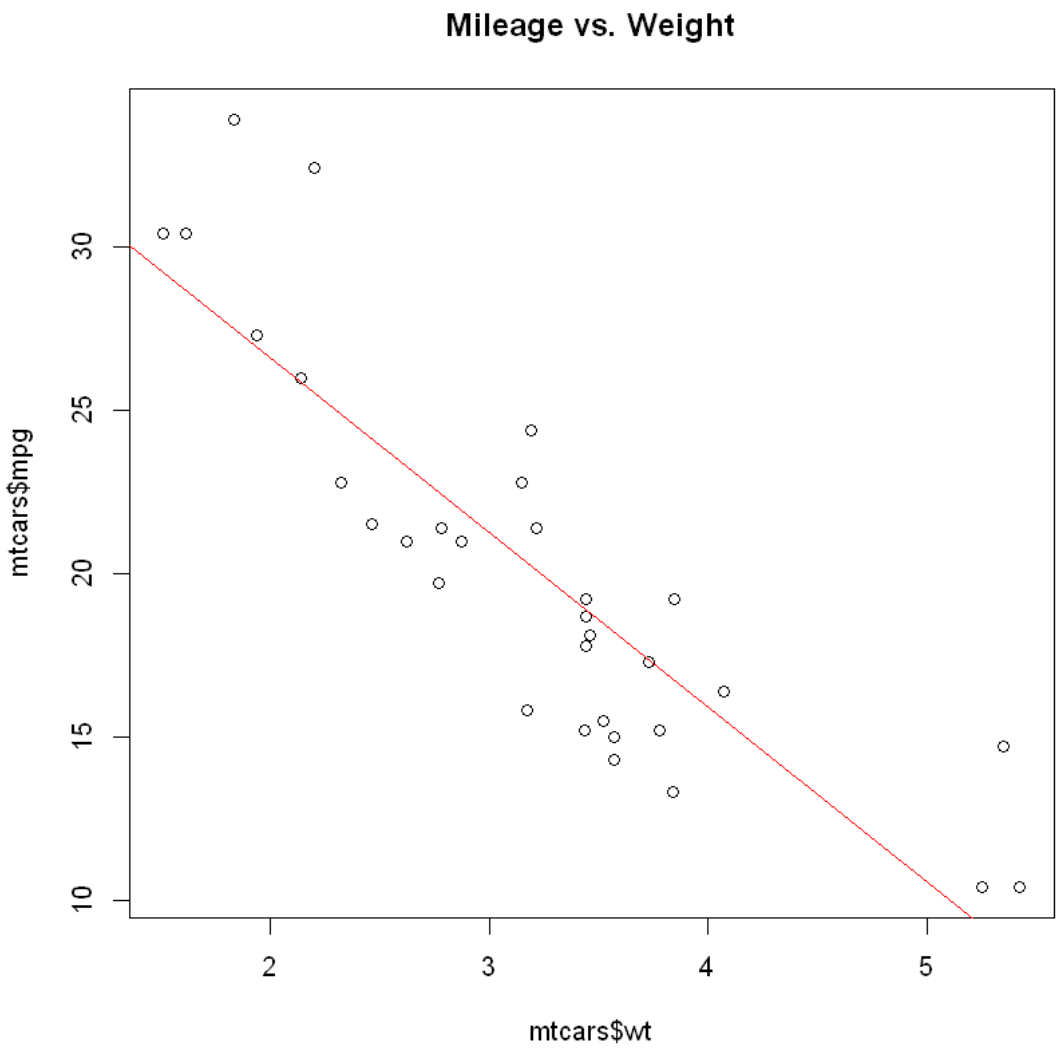
[63]:
```r
# Load dataset
head(mtcars,n = 3)
```

A data.frame: 3 × 11

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1

[64]:
```r
# Simple Linear Regression
model1 <- lm(mpg ~ wt, data = mtcars)
```

[65]:
```r
plot(mtcars$wt, mtcars$mpg, main = "Mileage vs. Weight")
abline(model1, col = "red")
```



[66]:
```
summary(model1)
```

```
Call:
lm(formula = mpg ~ wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
```

```
wt           -5.3445     0.5591  -9.559 1.29e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,        Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

[67]:
```r
# Prediction for weight = 6
pred_mpg <- predict(model1, newdata = data.frame(wt = 6))
cat("Predicted MPG:", pred_mpg, "\n")
```

```
Predicted MPG: 5.218297
```

[68]:
```r
# Model Coefficients
cat("Intercept:", coef(model1)[1], "\n")
cat("Slope:", coef(model1)[2], "\n")
```

```
Intercept: 37.28513
Slope: -5.344472
```

[69]:
```r
# Multiple Linear Regression
model2 <- lm(mpg ~ wt + hp, data = mtcars)
summary(model2)
```

```
Call:
lm(formula = mpg ~ wt + hp, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max
-3.941 -1.600 -0.182  1.050  5.854

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
hp          -0.03177    0.00903  -3.519  0.00145 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,        Adjusted R-squared:  0.8148
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

[70]:
```r
# Predictions for (wt = 2.5, hp = 275)
predict(model2, newdata = data.frame(wt = 2.5, hp = 275))
```

1: 18.7951328403412