Data Mining Lab (CSEN 3155)

(Statistical Analysis in R)
(Explore Classification data file)
(ggplot2)

Statistical Analysis in R

1. Mean

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Find Mean.
result.mean<- mean(x)
print(result.mean)
```

When trim = 0.3, 3 values from each end will be dropped from the calculations to find mean.
In this case the sorted vector is (−21, −5, 2, 3, 4.2, 7, 8, 12, 18, 54) and the values removed from the vector for calculating mean are (−21,−5,2) from left and (12,18,54) from right.

```
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Find Mean.
result.mean<-  mean(x,trim = 0.3)
print(result.mean)
```

2. Median

```
# Create the vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Find the median.
median.result<- median(x)
print(median.result)
```

3. Mode

The mode is the value that has highest number of occurrences in a set of data. Unlike mean and median, mode can have both numeric and character data. R does not have a standard in-built function to calculate mode. So we create a user function to calculate mode of a data set in R. This function takes the vector as input and gives the mode value as output.

```
# Create the function.
getmode<- function(v) {
uniqv<- unique(v)
uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Create the vector with numbers.
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)
# Calculate the mode using the user function.
result<- getmode(v)
print(result)
# Create the vector with characters.
charv<- c("o","it","the","it","it")
# Calculate the mode
mode = which.max(table(charv))
print(mode)
# Calculate the mode using the user function.
result<- getmode(charv)
print(result)
```

4. Covariance and Correlation

- Covariance and Correlation are terms used in statistics to measure relationships between two random variables.
- Both of these terms measure linear dependency between a pair of random variables or bivariate data.
- The correlation coefficient is measured on a scale that varies from + 1 through 0 to - 1.
- When one variable increases as the other increases the correlation is positive;
- When one decreases as the other increases it is negative. Complete absence of correlations represented by 0.
- Covariance and correlation are two mathematical concepts which are commonly used in statistics. When comparing data samples from different populations, covariance is used to determine how much two random variables vary together, whereas correlation is used to determine when a change in one variable can result in a change in another.

Correlation in R

Syntax: cor(x, y, method)
where x and y represents the data vectors,method defines the type of method to be used to compute correlation. Default is “pearson”.

Example:

```
# Data vectors
x <- c(1, 3, 5, 10)
y <- c(2, 4, 6, 20)
# Print correlation using different methods
print(cor(x, y))
print(cor(x, y, method = "pearson"))
print(cor(x, y, method = "kendall"))
print(cor(x, y, method = "spearman"))
```

Output:

```
[1] 0.9724702
[1] 0.9724702
[1] 1
[1] 1
```

Covariance Syntax in R

Syntax: cov(x, y, method)
Where x and y represents the data vectors, method defines the type of method to be used to compute covariance. Default is “pearson”.

Example

```
x <- c(1, 3, 5, 10)
y <- c(2, 4, 6, 20)
# Print covariance using different methods
print(cov(x, y))
print(cov(x, y, method = "pearson"))
print(cov(x, y, method = "kendall"))
print(cov(x, y, method = "spearman"))
```

Output:

```
[1] 30.66667
[1] 30.66667
[1] 12
[1] 1.666667
```

5. rnorm()

   - The rnorm in R is a built-in function that generates a vector of normally distributed random numbers.
   - The rnorm() function takes a sample size as input and generates many random numbers.
   - The rnorm() function helps to generate random numbers whose distribution is normal.

   Syntax
   rnorm(n, mean, sd)

   Parameters
   n: It is the number of observations(sample size).
   mean: It is the mean value of the sample data. Its default value is zero.
   sd: It is the standard deviation. Its default value is 1.

   Example:
   rnorm(10)
   -2.16711645 -1.44174333 0.27413377 0.02784403 -1.63774104 1.24496373
   -1.12228873 0.29122357 0.46364074 0.49827003

6. Skewness
   - Skewness refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data.
   - If the curve is shifted to the left or to the right, it is said to be skewed.
   - Skewness can be quantified as a representation of the extent to which a given distribution varies from a normal distribution.
   - A normal distribution has a skew of zero, while a lognormal distribution, for example, would exhibit some degree of right-skew.
   - Now, we know that the skewness is the measure of asymmetry and its types are distinguished by the side on which the tail of probability distribution lies.

   Example:
   ```
   install.packages ("moments")
   library (moments)
   # Defining data vector
   x <- c(35, 37, 40, 41, 42, 43, 55)
   # Print skewness of distribution
   print(skewness(x))
   # Histogram of distribution
   hist(x)
   ```

7. Calculate mean, median and mode of the data {1,2,1,3,2,4,4,4,3,2,1,5,5,4,5}.

8. First create a data frame containing 4 columns as follows:

	col1	col2	col3	col4
1	1	11	30	3
2	2	12	29	5
3	3	13	28	2
4	4	14	27	1
5	5	15	26	7
6	6	16	25	6
7	7	17	24	9
8	8	18	23	2
9	9	19	22	3
10	10	20	21	4

9. Find the correlation between col1, col2; col1, col3; col1, col4; col3, col4
10. Find the pairwise correlation between col1, col4 and col2 and display as a two-dimensional matrix. Similarly, do it for col2, col3, col4.
11. Find the pairwise correlation between all the variables and display as a two-dimensional matrix.
12. Write a R program to create bell curve of a random normal distribution. Calculate the skewness of it.
13. Create a random vector containing n values ranging from min to max using Uniform distribution generator runif(n, min, max)) and find the Nth highest value of the vector in R.

On Classification data analysis

14. Read the file "ClassificationSimpleLab.csv".
15. Display the file
16. Display the effective input variables
17. Display the class variable
18. For every variable, display the number of distinct values and their counts in the file.
19. For every variable, display the number of distinct values and corresponding counts of the class values.

ggplot2 is a powerful graphics package with a ggplot() function.

In the lab book, explain what happens when you execute each of the commands in R.

1. install.packages("ggplot2")
2. library(ggplot2)
3. auto = read.csv("auto-mpg.csv")
4. auto$cylinders <- factor(auto$cylinders,
   levels = c(3,4,5,6,8),
   labels = c("3cyl", "4cyl", "5cyl", "6cyl", "8cyl"))

Single variable plots
#Scatter plot

5. ggplot(auto, aes(x=weight, y=mpg)) + geom_point()

#Line Graph
6. ggplot(auto, aes(x=weight, y=mpg)) + geom_line()
7. ggplot(auto, aes(x=weight, y=mpg)) + geom_line() + geom_point()
8. ggplot(auto, aes(x=cylinders, y=horsepower)) + geom_col(fill="red")
9. ggplot(auto, aes(x=mpg, y=horsepower)) + geom_col(fill="green")
10. ggplot(auto, aes(x=cylinders, y=horsepower)) + geom_col()
11. ggplot(auto, aes(x=factor(cylinders),y=horsepower)) + geom_col()

Similar to geom_col()
12. ggplot(auto, aes(x=cylinders, y=horsepower)) + geom_bar(stat="identity")
13. ggplot(auto, aes(x=cylinders)) + geom_bar()

#histogram
14. ggplot(auto, aes(x=cylinders)) + geom_histogram()
15. auto$cylinders = as.numeric(auto$cylinders)
16. ggplot(auto, aes(x=cylinders)) + geom_histogram()
17. ggplot(auto, aes(x=cylinders)) + geom_histogram(bin=30)
18. ggplot(auto, aes(x=cylinders)) + geom_histogram(binwidth=4)
19. ggplot(auto, aes(x=cylinders)) + geom_histogram(binwidth=30)

```
20. ggplot(auto, aes(x=cylinders, y=horsepower)) + geom_col()
21. auto$cylinders <- factor(auto$cylinders,
                             levels = c(3,4,5,6,8),
                             labels = c("3cyl", "4cyl", "5cyl", "6cyl", "8cyl"))

22. ggplot(auto, aes(x=cylinders, y=horsepower)) + geom_col()

#boxplot
23. ggplot(auto, aes(x=cylinders, y=horsepower)) + geom_boxplot()

#Plotting a function curve
# Plot a user-defined function
24. myfun <- function(xvar) {
    1 / (1 + exp(-xvar + 10))
    }

25. myfun1 <- function(xvar) {
    (xvar^3-5*xvar)
    }
26. ggplot(data.frame(x = c(-4, 4)), aes(x = x)) +
    stat_function(fun = myfun1, geom = "line")
27. ggplot(data.frame(x = c(0, 20)), aes(x = x)) +
    stat_function(fun = myfun, geom = "line")
28. ggplot(data.frame(x = c(0, 20)), aes(x, colour="orange")) +
    stat_function(fun = myfun, geom = "line")
```