Learning Outcomes

At the end of the session, you will be able to:

- Understanding correlation
- Plotting correlation with heatmap
- Understanding normalization with log transformation, standard scaling and min-max scaling

Activity

1. Understanding Correlation Analyses

1.1. Illustrate Pearson Correlation Testing using cor() method

- Write and run the following in R. Make your conclusion about the code:

```
x = c(1, 2, 3, 4, 5, 6, 7)
y = c(1, 3, 6, 2, 7, 4, 5)

# Calculating correlation coefficient using cor() method
result = cor(x, y, method = "pearson")

cat("Pearson correlation coefficient is:", result)
```

1.2. Illustrate Pearson Correlation Testing using cor.test() method

- Write and run the following in R. Make your conclusion about the code:

```
# Calculating correlation coefficient using cor.test() method
result = cor.test(x, y, method = "pearson")

# Print the result
print(result)
```

1.3. Visualize Correlation Matrix using Correlogram

- Write and run the following in R. Make your conclusion about the code:

```
install.packages("corrplot")
install.packages("RColorBrewer")
library(corrplot)
library(RColorBrewer)

M <-cor(mtcars)

?corrplot
corrplot(M, type="upper")
corrplot(M, type="upper", order="hclust")
corrplot(M, type="upper", order="hclust",col=brewer.pal(n=8,
name="RdYlBu"))
```

2. Plotting Correlation with Heatmap

2.1. Create and reorder correlation matrix

```
install.packages("lattice")
library(lattice)
```

```
data <-environmental
head(data)

corr_mat <- round(cor(data),2)
head(corr_mat)

# reorder corr matrix using corr coefficient as distance metric
dist <- as.dist((1-corr_mat)/2)

# hierarchical clustering the dist matrix
hc <- hclust(dist)
corr_mat <-corr_mat[hc$order, hc$order]

install.packages("reshape2")
library(reshape2)

# reduce the size of correlation matrix
melted_corr_mat <- melt(corr_mat)
head(melted_corr_mat)
```

2.2. Correlation Heatmap using ggplot2.

- Write and run the following in R. Make your conclusion about the code:

```
install.packages("ggplot2")
library(ggplot2)

ggplot(data = melted_corr_mat, aes(x=Var1, y=Var2,
fill=value)) + geom_tile() + geom_text(aes(Var2, Var1, label =
value), color = "white", size = 4)
```

2.3. Correlation Heatmap using heatmaply

- Write and run the following in R. Make your conclusion about the code:

```
install.packages("heatmaply")
library(heatmaply)

# plotting corr heatmap
heatmaply_cor(x = cor(data), xlab = "Features", ylab = "Features",
k_col = 2, k_row = 2)
```

2.4. Correlation Heatmap using ggcorplot

- Write and run the following in R. Make your conclusion about the code:

```
install.packages("ggcorplot")
library(ggcorrplot)
```

```
# plotting corr heatmap
ggcorrplot::ggcorrplot(cor(data))
```

3. Understanding Normalization

3.1. Normalize data with log transformation in base R

- Write and run the following in R. Make your conclusion about the code:

```
# Create data
mydata <- c(244, 753, 596, 645, 874, 141,639, 465, 999, 654)

# normalizing data
scaled_data <- log(mydata)
print(scaled_data)
```

3.2. Normalize Data with Standard Scaling in R

- Write and run the following in R. Make your conclusion about the code:

```
scaled_data2<- as.data.frame(scale(mydata))
print(scaled_data2)
```

3.3. Normalize Data using Min-Max Scaling

- Write and run the following in R. Make your conclusion about the code:

```
install.packages("caret")
library(caret)

minmax <- preProcess(as.data.frame(mydata), method=c("range"))
scaled_data3 <- predict(minmax, as.data.frame(mydata))
print(scaled_data3)
```

3.4. Compare and contrast the summary of the raw data vs normalized data

- Write and run the following in R. Make your conclusion about the code:

```
summary(mydata)
summary(scaled_data1)
summary(scaled_data2)
summary(scaled_data3)
```