Biostatistics: Theory and Applications in R (Virtual)

Week9_Session2_R_training9

Contents

Prof Dr Mohammed Abu Sayed Arfin Khan
Department of Forestry and Environmental Science
Shahjalal University of Science and Technology, Sylhet
+8801917174537, khan-for@sust.edu, nobelarfin@yahoo.com
Homepage | Google scholar | Researchgate | ORCID | Publons | BayCEER

```
#Set the working directory- getwd()/ setwd("Y:/")
getwd()
setwd("C:/Users/Fahmida Sultana/Desktop/R training/R_training_Class_09")


#install openxlsx package or xlsx package
library(openxlsx)
library(readxl)


#####import data set from xlsx file
study_1 <- read.xlsx("Tree_height.xlsx")
str(study_1)

study_2 <- read_excel("Tree_height.xlsx", sheet = "study_2")
str(study_2)
```

#ggpubr: 'ggplot2' Based Publication Ready Plots

```
library(ggplot2)
library(ggpubr)
```

Density plot with mean lines and marginal rug

```
# rug = in 2d display with the two 1d marginal distribution

str(study_1)

ggdensity(study_1, x = "biomass",
        add = "mean", rug = TRUE,
        color = "treatment", fill = "treatment",
        palette = c("#00AFBB", "#E7B800", "pink"))
```

Histogram plot with mean lines and marginal rug

```
gghistogram(study_1, x = "biomass",
        add = "mean", rug = TRUE,
        color = "treatment", fill = "treatment",
        palette = c("#00AFBB", "#E7B800", "pink"))
```

#Box plots and violin plots

```
# Box plots with jittered points

str(study_1)
p <- ggboxplot(study_1, x = "treatment", y = "biomass",
          color = "treatment", palette =c("#00AFBB", "#E7B800", "#FC4E07"),
          add = "jitter", shape = "treatment")

# Add p-values comparing groups

# Specify the comparisons you want

my_comparisons <- list( c("Control", "Drought"), c("Drought", "Rain"), c("Control", "Rain") )

p + stat_compare_means(comparisons = my_comparisons)+ # Add pairwise comparisons p-value
  stat_compare_means(label.y = 470)                  # Add global p-value
```

Violin plots with box plots inside

```
ggviolin(study_1, x = "treatment", y = "biomass",
          fill = "treatment", palette =c("#00AFBB", "#E7B800", "#FC4E07"),
          add = "boxplot", add.params = list(fill = "white"))+
      stat_compare_means(comparisons = my_comparisons)+ # Add pairwise comparisons p-value
      stat_compare_means(label.y = 470)                 # Add global p-value
```

#Bar plots with Demo data set

```
data("mtcars")
dfm <- mtcars

# Convert the cyl variable to a factor
dfm$cyl <- as.factor(dfm$cyl)

# Add the name colums
dfm$name <- rownames(dfm)

# Inspect the data
head(dfm[, c("name", "wt", "mpg", "cyl")])


#Ordered bar plots
```

```
#Sorting will be done globally, but not by groups.

ggbarplot(dfm, x = "name", y = "mpg",
        fill = "cyl",              # change fill color by cyl
        color = "white",           # Set bar border colors to white
        palette = "jco",           # jco journal color palett. see ?ggpar
        sort.val = "desc",         # Sort the value in dscending order
        sort.by.groups = FALSE,    # Don't sort inside each group
        x.text.angle = 90          # Rotate vertically x axis texts
)

#Sort bars inside each group. Use the argument sort.by.groups = TRUE.


ggbarplot(dfm, x = "name", y = "mpg",
        fill = "cyl",              # change fill color by cyl
        color = "white",           # Set bar border colors to white
        palette = "jco",           # jco journal color palett. see ?ggpar
        sort.val = "desc",         # Sort the value in dscending order
        sort.by.groups = TRUE,     # Don't sort inside each group
        x.text.angle = 90          # Rotate vertically x axis texts
)
```

#Deviation graphs

```
# Calculate the z-score of the mpg data
dfm$mpg_z <- (dfm$mpg -mean(dfm$mpg))/sd(dfm$mpg)
dfm$mpg_grp <- factor(ifelse(dfm$mpg_z < 0, "low", "high"),
                levels = c("low", "high"))

# Inspect the data
head(dfm[, c("name", "wt", "mpg", "mpg_z", "mpg_grp", "cyl")])

#Create an ordered barplot, colored according to the level of mpg:


ggbarplot(dfm, x = "name", y = "mpg_z",
        fill = "mpg_grp",          # change fill color by mpg_level
        color = "white",           # Set bar border colors to white
        palette = "jco",           # jco journal color palett. see ?ggpar
        sort.val = "asc",          # Sort the value in ascending order
        sort.by.groups = FALSE,    # Don't sort inside each group
        x.text.angle = 90,         # Rotate vertically x axis texts
        ylab = "MPG z-score",
        xlab = FALSE,
        legend.title = "MPG Group"
)
```

```
#Rotate the plot: use rotate = TRUE and sort.val = “desc”


ggbarplot(dfm, x = "name", y = "mpg_z",
        fill = "mpg_grp",          # change fill color by mpg_level
        color = "white",           # Set bar border colors to white
        palette = "jco",           # jco journal color palett. see ?ggpar
        sort.val = "desc",         # Sort the value in descending order
        sort.by.groups = FALSE,    # Don't sort inside each group
        x.text.angle = 90,         # Rotate vertically x axis texts
        ylab = "MPG z-score",
        legend.title = "MPG Group",
        rotate = TRUE,
        ggtheme = theme_minimal()
)
```

#Dot charts

```
#Lollipop chart
#Lollipop chart is an alternative to bar plots,
#when you have a large set of values to visualize.

ggdotchart(dfm, x = "name", y = "mpg",
        color = "cyl",                      # Color by groups
        palette = c("#00AFBB", "#E7B800", "#FC4E07"), # Custom color palette
        sorting = "ascending",                  # Sort value in descending order
        add = "segments",                       # Add segments from y = 0 to dots
        ggtheme = theme_pubr()                  # ggplot2 theme
)


#Sort in decending order. sorting = “descending”.

#Rotate the plot vertically, using rotate = TRUE.
#Sort the mpg value inside each group by using group = “cyl”.
#Set dot.size to 6.
#Add mpg values as label. label = “mpg” or label = round(dfm$mpg).

ggdotchart(dfm, x = "name", y = "mpg",
        color = "cyl",                      # Color by groups
        palette = c("#00AFBB", "#E7B800", "#FC4E07"), # Custom color palette
        sorting = "descending",                 # Sort value in descending order
        add = "segments",                       # Add segments from y = 0 to dots
        rotate = TRUE,                        # Rotate vertically
        group = "cyl",                        # Order by groups
        dot.size = 6,                       # Large dot size
        label = round(dfm$mpg),                 # Add mpg values as dot labels
        font.label = list(color = "white", size = 9,
```

```
                vjust = 0.5),           # Adjust label parameters
        ggtheme = theme_pubr()                  # ggplot2 theme
)


#Deviation graph:

# Use y = “mpg_z”
#Change segment color and size: add.params = list(color = “lightgray”, size = 2)

ggdotchart(dfm, x = "name", y = "mpg_z",
        color = "cyl",                  # Color by groups
        palette = c("#00AFBB", "#E7B800", "#FC4E07"), # Custom color palette
        sorting = "descending",                 # Sort value in descending order
        add = "segments",                       # Add segments from y = 0 to dots
        add.params = list(color = "lightgray", size = 2), # Change segment color and size
        group = "cyl",                  # Order by groups
        dot.size = 6,                   # Large dot size
        label = round(dfm$mpg_z,1),                 # Add mpg values as dot labels
        font.label = list(color = "white", size = 9,
                vjust = 0.5),           # Adjust label parameters
        ggtheme = theme_pubr()                  # ggplot2 theme
)+
  geom_hline(yintercept = 0, linetype = 2, color = "lightgray")
```

#Principal Component Analysis in R

```
library(devtools)
library(ggbiplot)
library(scales)
library(grid)
#install.packages("devtools")
#install_github("vqv/ggbiplot")


str(study_2)

study_2.pca<- prcomp(study_2[,c(3:6)], center = TRUE,scale. = TRUE)
summary(study_2.pca)
str(study_2.pca)



ggbiplot(study_2.pca)
```

#ellipse around each group.

```
ggbiplot(study_2.pca,ellipse=TRUE, groups=study_2$study_area)


#rownames as labels
ggbiplot(study_2.pca,ellipse=TRUE,labels=rownames(study_2), groups=study_2$study_area)
```

#other example of PCA graph

```
library(factoextra)
library(ggfortify)

#install.packages("factoextra")
#install.packages("ggfortify")

fviz_pca_var(study_2.pca,
             col.var = "contrib",
             gradient.cols = c("red","gold","green3","royalblue"))
```

#ggcorrplot: Visualization of a correlation matrix using ggplot2

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


# Compute a correlation matrix
str(study_2)
corr <- round(cor(study_2[,c(3:6)]), 1)
head(corr[, 1:4])

# Compute a matrix of correlation p-values
p.mat <- cor_pmat(study_2[,c(3:6)])
head(p.mat[, 1:4])
```

Visualize the correlation matrix

```
# method = "square" (default)

ggcorrplot(corr)
```

```
# method = "circle"
ggcorrplot(corr, method = "circle")


# Reordering the correlation matrix
# using hierarchical clustering

ggcorrplot(corr, hc.order = TRUE, outline.color = "white")
```

Types of correlogram layout

```
# Get the lower triangle
ggcorrplot(corr,
        hc.order = TRUE,
        type = "lower",
        outline.color = "white")


# Get the upper triangle
ggcorrplot(corr,
        hc.order = TRUE,
        type = "upper",
        outline.color = "white")


# Add correlation coefficients
# argument lab = TRUE
ggcorrplot(corr,
        hc.order = TRUE,
        type = "lower",
        lab = TRUE)
```

Add correlation significance level

```
# Argument p.mat
# Barring the no significant coefficient
ggcorrplot(corr,
        hc.order = TRUE,
        type = "lower",
        p.mat = p.mat)
```

#ggtext: Improved text rendering support for ggplot2

```
library(tidyverse)
library(ggtext)
library(glue)

#install.packages("tidyverse")
#install.packages("ggtext")
#install.packages("glue")
#employ images as axis labels
```

#download images from website and use it in the figure

```
labels <- c(
  p2 = "<img
src='https://upload.wikimedia.org/wikipedia/commons/thumb/8/86/Iris_setosa.JPG/180px-
Iris_setosa.JPG'
    width='100' /><br>*I. setosa*",
  p4 = "<img src='https://upload.wikimedia.org/wikipedia/commons/thumb/3/38/Iris_virginica_-
_NRCS.jpg/320px-Iris_virginica_-_NRCS.jpg'
    width='100' /><br>*I. virginica*"
)

str(study_1)
ggplot(study_1, aes(plant, biomass)) +
  geom_boxplot() +
  scale_x_discrete(
    name = NULL,
    labels = labels
  ) +
  theme(
    axis.text.x = element_markdown(color = "black", size = 11)
  )
```

#using lagre amount of text in the figure

```
library(ggplot2)
library(ggpubr)
library(tidyverse)
library(ggtext)
library(glue)
```

```
ggviolin(study_1, x = "treatment", y = "biomass",
       fill = "treatment", palette =c("#00AFBB", "#E7B800", "#FC4E07"),
       add = "boxplot", add.params = list(fill = "white"))+
 labs(
  title = "<b>Plant biomass production vs. climate treatment</b><br>
  <span style = 'font-size:10pt'>Plant biomass *under control*
  not significantly vary with **'drought treatment'** in our
  experiment. <span style = 'color:red;'>However, under rain</span> plant biomass was
  slightly higher than control.</span>",
  x = "Treatment",
  y = "Biomass(mg)<br><span style = 'font-size:8pt'>A measure of
  the plant productivity.</span>"
 ) +
 theme(
  plot.title.position = "plot",
  plot.title = element_textbox_simple(
   size = 13,
   lineheight = 1,
   padding = margin(5.5, 5.5, 5.5, 5.5),
   margin = margin(0, 0, 5.5, 0),
   fill = "cornsilk"
  ),
  axis.title.x = element_textbox_simple(
   width = NULL,
   padding = margin(4, 4, 4, 4),
   margin = margin(4, 0, 0, 0),
   linetype = 1,
   r = grid::unit(8, "pt"),
   fill = "azure1"
  ),
  axis.title.y = element_textbox_simple(
   hjust = 0,
   orientation = "left-rotated",
   minwidth = unit(1, "in"),
   maxwidth = unit(2, "in"),
   padding = margin(4, 4, 2, 4),
   margin = margin(0, 0, 2, 0),
   fill = "lightsteelblue1"
  )
 )
```