Week-1

Rajesh Kalakoti

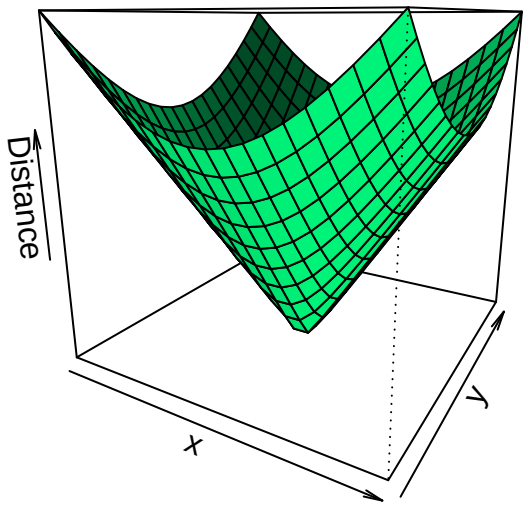
2023-08-03

```
# Include the script from the R directory
project_path <- here()
source(here("R", "utils.R"))
source(here("R","distance_functions.R"))
```

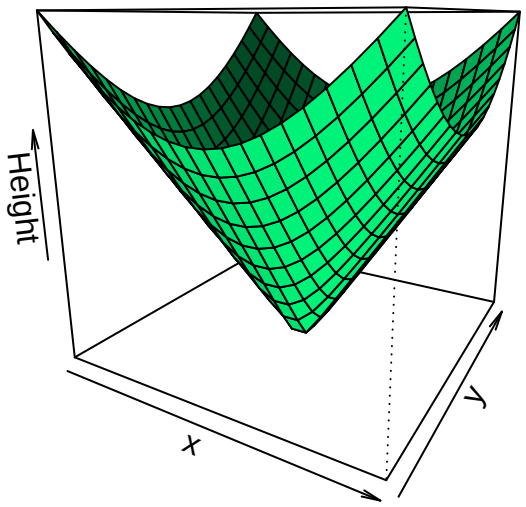
Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
euclidean_dist <- function(point1, point2) {
  squared_diff <- (point1 - point2)^2
  sqrt(sum(squared_diff))
}

x <- y <- seq(-1, 1, length = 20)
grid <- expand.grid(x = x, y = y)  # Create a grid of points
z <- matrix(0, nrow = length(x), ncol = length(y))  # Initialize the z matrix

for (i in 1:length(x)) {
  for (j in 1:length(y)) {
    z[i, j] <- euclidean_dist(c(x[i], y[j]), c(0, 0))
  }
}
persp(x, y, z,
      main = "3D Plot of Euclidean Distance",
      zlab = "Distance",
      theta = 30, phi = 15,
      col = "springgreen", shade = 0.5)
```

3D Plot of Euclidean Distance

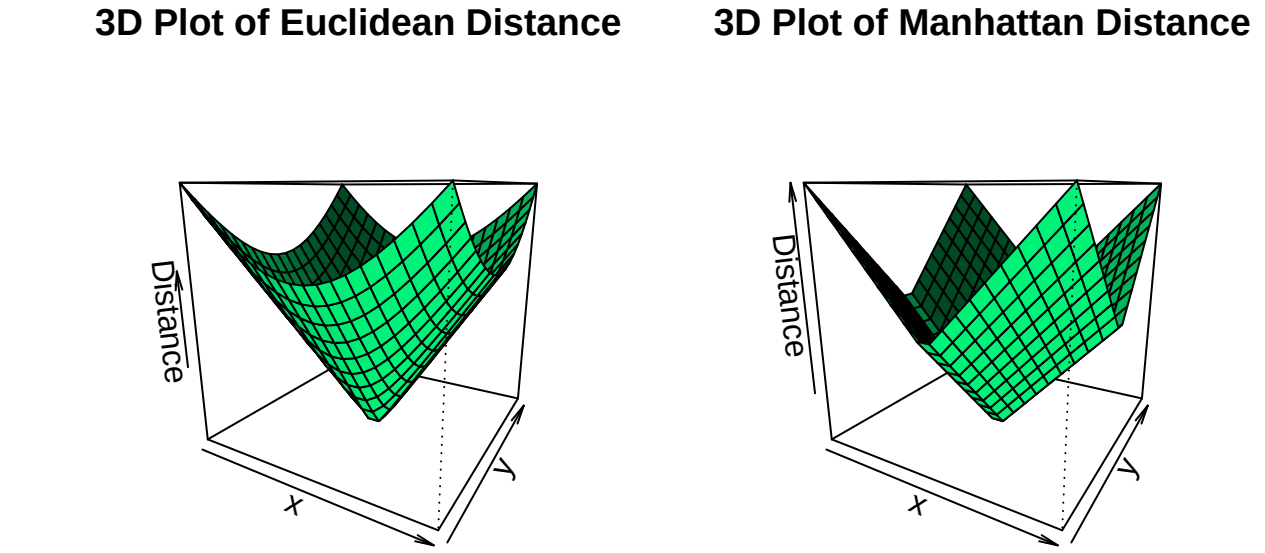



```
cone1 <- function(x, y){
sqrt(x^2+y^2)
}

x <- y <- seq(-1, 1, length= 20)
z <- outer(x, y, cone1)

persp(x, y, z,
main="Perspective Plot of a Cone",
zlab = "Height",
theta = 30, phi = 15,
col = "springgreen", shade = 0.5)
```




```r
euclidean_dist <- function(point1, point2) {
  squared_diff <- (point1 - point2)^2
  sqrt(sum(squared_diff))
}

manhattan_distance <- function(point1, point2) {
  if (length(point1) != length(point2)) {
    stop("Both points should have the same number of dimensions.")
  }

  abs_diff <- abs(point1 - point2)
  distance <- sum(abs_diff)
  return(distance)
}

x <- y <- seq(-1, 1, length = 20)
grid <- expand.grid(x = x, y = y)  # Create a grid of points
z_euclidean <- matrix(0, nrow = length(x), ncol = length(y))  # Initialize the z matrix for Euclidean d
z_manhattan <- matrix(0, nrow = length(x), ncol = length(y))   # Initialize the z matrix for Manhattan

for (i in 1:length(x)) {
  for (j in 1:length(y)) {
    z_euclidean[i, j] <- euclidean_dist(c(x[i], y[j]), c(0, 0))
    z_manhattan[i, j] <- manhattan_distance(c(x[i], y[j]), c(0, 0))
  }
}
```

```r
# Create a layout of subplots to show both Euclidean and Manhattan distances
par(mfrow = c(1, 2))

# Plot for Euclidean distance
persp(x, y, z_euclidean,
      main = "3D Plot of Euclidean Distance",
      zlab = "Distance",
      theta = 30, phi = 15,
      col = "springgreen", shade = 0.5)

# Plot for Manhattan distance
persp(x, y, z_manhattan,
      main = "3D Plot of Manhattan Distance",
      zlab = "Distance",
      theta = 30, phi = 15,
      col = "springgreen", shade = 0.5)
```

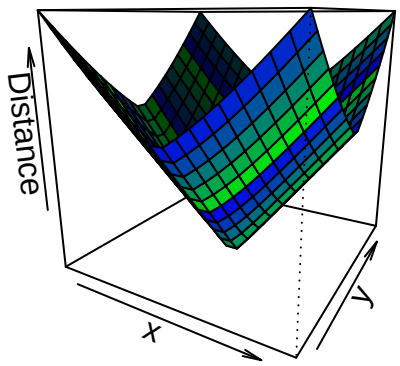



```r
# Reset the layout
par(mfrow = c(1, 1))
```

```r
euclidean_dist <- function(point1, point2) {
  squared_diff <- (point1 - point2)^2
  sqrt(sum(squared_diff))
}
```

```r
manhattan_distance <- function(point1, point2) {
  if (length(point1) != length(point2)) {
    stop("Both points should have the same number of dimensions.")
  }

  abs_diff <- abs(point1 - point2)
  distance <- sum(abs_diff)
  return(distance)
}

x <- y <- seq(-5, 5, length = 20)
grid <- expand.grid(x = x, y = y)  # Create a grid of points

z_euclidean <- matrix(0, nrow = length(x), ncol = length(y))  # Initialize the z matrix for Euclidean d
z_manhattan <- matrix(0, nrow = length(x), ncol = length(y))   # Initialize the z matrix for Manhattan

for (i in 1:length(x)) {
  for (j in 1:length(y)) {
    z_euclidean[i, j] <- euclidean_dist(c(x[i], y[j]), c(0, 0))
    z_manhattan[i, j] <- manhattan_distance(c(x[i], y[j]), c(0, 0))
  }
}

# Combine the distances and choose different colors for each
combined_distances <- z_euclidean + z_manhattan
color_palette <- colorRampPalette(c("blue", "green"))(100)  # Choose colors for mapping distances

# Create a layout of subplots
layout(matrix(c(1, 2), nrow = 1))

# Plot both distances on the same 3D plane with different colors
persp(x, y, combined_distances,
      main = "3D Plot of Combined Distances",
      zlab = "Distance",
      theta = 30, phi = 15,
      col = color_palette, shade = 0.5)

# Reset the layout
layout(1)
```

3D Plot of Combined Distances




```
# Include the script from the R directory
project_path <- here()
source(here("R", "utils.R"))
```

```
?entropy
```

```
## No documentation for 'entropy' in specified packages and libraries:
## you could try '??entropy'
```

```
library("car")
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```r
library("rgl")
```

```
## This build of rgl does not include OpenGL functions.  Use
##  rglwidget() to display results, e.g. via options(rgl.printRglwidget = TRUE).
```

```r
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 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
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
sep.l <- iris$Sepal.Length
sep.w <- iris$Sepal.Width
pet.l <- iris$Petal.Length
```

```r
library("car")
library("rgl")
data(iris)
sep.l <- iris$Sepal.Length
sep.w <- iris$Sepal.Width
pet.l <- iris$Petal.Length

save <- getOption("rgl.useNULL")
options(rgl.useNULL = TRUE)
scatter3d(x = sep.l, y = pet.l, z = sep.w, groups = iris$Species,
          surface = FALSE, ellipsoid = TRUE)
```

```
## Loading required namespace: mgcv
```

```
## Loading required namespace: MASS
```

```r
widget <- rglwidget()
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead
```

```
## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```

```r
# Explicitly set the elementId property
widget$elementId <- "my-rgl-plot"
```

```
## Warning in widget$elementId <- "my-rgl-plot": Coercing LHS to a list
```

```
widget
```

```
## [[1]]
## [1] "../../../../../../tmp/RtmpPZ1OsT/file6e29568b48b8.png"
##
## $elementId
## [1] "my-rgl-plot"
```

```
#
```

```
library(rgl)

# Load the Iris dataset
data(iris)

# Create an interactive 3D scatter plot
scatter3d(x = iris$Sepal.Length, y = iris$Petal.Length, z = iris$Sepal.Width,
          groups = iris$Species, surface = FALSE, ellipsoid = TRUE)

# Display the interactive plot
# rglwidget()
```

```
library(rgl)

# Adding Titles and Labeling Axes to Plot
cone <- function(x, y){
sqrt(x ^ 2 + y ^ 2)
}

# prepare variables.
x <- y <- seq(-1, 1, length = 30)
z <- outer(x, y, cone)

# plot the 3D surface
# Adding Titles and Labeling Axes to Plot
persp3d(x, y, z,col = "orange")

# add animation

# add animation
play3d(spin3d(axis = c(0, 0, 1)), duration = 10)
# Display the interactive plot using rglwidget()
rglwidget()
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead
```

```
## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```

```r
library(rgl)

# Function to compute cone height
cone <- function(x, y) {
  sqrt(x ^ 2 + y ^ 2)
}

# Prepare variables
x <- y <- seq(-1, 1, length = 30)
z <- outer(x, y, cone)

# Create the 3D plot
persp3d(x, y, z,
        type = "surface",
        col = "orange")

# Add rotation animation
play3d(spin3d(axis = c(0, 0, 1)), duration = 10)

# Display the interactive plot using rglwidget()
rglwidget()
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead
```

```
## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```

```r
library(rgl)

# Function to compute cone height using Chebyshev distance
chebyshev_cone <- function(x, y) {
  max(abs(x), abs(y))
}

# Prepare variables
x <- y <- seq(-1, 1, length = 30)

# Calculate z values using Euclidean distance
z_euclidean <- outer(x, y, function(x, y) sqrt(x^2 + y^2))

# Calculate z values using Chebyshev distance
z_chebyshev <- matrix(0, nrow = length(x), ncol = length(y))
for (i in seq_along(x)) {
  for (j in seq_along(y)) {
    z_chebyshev[i, j] <- chebyshev_cone(x[i], y[j])
  }
}

# Scale the z values for better visibility
```

```r
z_euclidean <- z_euclidean * 0.5
z_chebyshev <- z_chebyshev * 0.5

# Create the 3D plot for Euclidean distance cone
plot3d(x, y, z_euclidean,
       type = "surface",
       col = "orange")

# Add the Chebyshev cone to the same plot
plot3d(x, y, z_chebyshev,
       type = "surface",
       col = "blue",  # Using a different color for the Chebyshev cone
       add = TRUE)    # Adding to the existing plot

# Add rotation animation
play3d(spin3d(axis = c(0, 0, 1)), duration = 10)

# Display the interactive plot using rglwidget()
rglwidget()
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead
```

```
## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```

```r
library(rgl)

# Create two example vectors of length 100
x <- rnorm(100)
y <- rnorm(100)

# Calculate Euclidean and Manhattan distances
euclidean_distance <- sqrt((x - y)^2)
manhattan_distance <- abs(x - y)

# Create a 3D scatter plot
scatter3d(x = x, y = y, z = euclidean_distance, col = "blue", size = 2)
scatter3d(x = x, y = y, z = manhattan_distance, col = "red", size = 2, add = TRUE)
rglwidget()
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead
```

```
## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```

```r
library(rgl)
euclidean_dist <- function(point1, point2) {
  squared_diff <- (point1 - point2)^2
  sqrt(sum(squared_diff))
}

manhattan_distance <- function(point1, point2) {
  if (length(point1) != length(point2)) {
    stop("Both points should have the same number of dimensions.")
  }

  abs_diff <- abs(point1 - point2)
  distance <- sum(abs_diff)
  return(distance)
}

x <- y <- seq(-5, 5, length = 20)
grid <- expand.grid(x = x, y = y)  # Create a grid of points

z_euclidean <- matrix(0, nrow = length(x), ncol = length(y))  # Initialize the z matrix for Euclidean d
z_manhattan <- matrix(0, nrow = length(x), ncol = length(y))   # Initialize the z matrix for Manhattan

for (i in 1:length(x)) {
  for (j in 1:length(y)) {
    z_euclidean[i, j] <- euclidean_dist(c(x[i], y[j]), c(0, 0))
    z_manhattan[i, j] <- manhattan_distance(c(x[i], y[j]), c(0, 0))
```

```
  }
}

# Combine the distances and choose different colors for each
combined_distances <- z_euclidean + z_manhattan
color_palette <- colorRampPalette(c("blue", "green"))(100)  # Choose colors for mapping distances

# Create a layout of subplots
layout(matrix(c(1, 2), nrow = 1))

# Plot both distances on the same 3D plane with different colors
persp3d(x, y, combined_distances,
      main = "3D Plot of Combined Distances",
      zlab = "Distance",
      theta = 30, phi = 15,
      col = color_palette, shade = 0.5)

# Reset the layout
layout(1)
rglwidget()
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead
```

```
## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```

```r
library(rgl)

euclidean_dist <- function(point1, point2) {
  squared_diff <- (point1 - point2)^2
  sqrt(sum(squared_diff))
}

manhattan_distance <- function(point1, point2) {
  if (length(point1) != length(point2)) {
    stop("Both points should have the same number of dimensions.")
  }

  abs_diff <- abs(point1 - point2)
  distance <- sum(abs_diff)
  return(distance)
}

x <- y <- seq(-5, 5, length = 20)
grid <- expand.grid(x = x, y = y)  # Create a grid of points

z_euclidean <- matrix(0, nrow = length(x), ncol = length(y))  # Initialize the z matrix for Euclidean d
z_manhattan <- matrix(0, nrow = length(x), ncol = length(y))   # Initialize the z matrix for Manhattan

for (i in 1:length(x)) {
  for (j in 1:length(y)) {
    z_euclidean[i, j] <- euclidean_dist(c(x[i], y[j]), c(0, 0))
```

```
    z_manhattan[i, j] <- manhattan_distance(c(x[i], y[j]), c(0, 0))
  }
}

# Combine the distances and choose different colors for each
combined_distances <- z_euclidean + z_manhattan
color_palette <- colorRampPalette(c("blue", "green"))(100)  # Choose colors for mapping distances

# Create a layout of subplots
layout(matrix(c(1, 2), nrow = 1))

# Plot both distances on the same 3D plane with different colors
persp3d(x, y, z_euclidean,
        main = "euclidean",
        zlab = "Distance",
        theta = 30, phi = 15,
        col = color_palette, shade = 0.5)

persp3d(x, y, z_manhattan,
        main = "Manhattan distance",
        zlab = "Distance",
        theta = 30, phi = 15,
        col = color_palette, shade = 0.5)

# Add annotations to serve as legends
text3d(x = 0, y = 0, z = max(combined_distances) + 10, text = "Euclidean", adj = c(0.5, 0))
text3d(x = 0, y = 0, z = max(combined_distances) + 5, text = "Manhattan", adj = c(0.5, 0))

# Reset the layout
layout(1)
rglwidget()
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot =
## TRUE requires the webshot2 package and Chrome browser; using rgl.snapshot()
## instead
```

```
## Warning in rgl.snapshot(filename, fmt, top): this build of rgl does not support
## snapshots
```