LAB PROGRAM

1. **Visualizing Amounts and Distributions**

```
# Load data and prepare
data("AirPassengers")
air_df <- data.frame(
  Passengers = as.numeric(AirPassengers),
  Year = factor(floor(time(AirPassengers))),
  Month = factor(cycle(AirPassengers)),
  MonthName = factor(month.abb[cycle(AirPassengers)], levels = month.abb)
)
```

 # 1. Bar Plot (Yearly totals)

```
year_totals <- aggregate(Passengers ~ Year, air_df, sum)
barplot(year_totals$Passengers, names.arg = year_totals$Year,
        main = "Total Passengers per Year", col = "skyblue")
```

 # 2. Grouped Bar Plot

```
library(ggplot2)
ggplot(air_df, aes(x = MonthName, y = Passengers, fill = Year)) +
  geom_col(position = "dodge") + ggtitle("Monthly Passengers by Year")
```

 # 3. Stacked Bar Plot

```
ggplot(air_df, aes(x = Year, y = Passengers, fill = MonthName)) +
  geom_col() + ggtitle("Yearly Passengers Stacked by Month")
```

 # 4. Dot Plot

```
plot(air_df$Passengers, pch = 19, col = "red",
     main = "Dot Plot: Monthly Passengers", ylab = "Passengers")
```

 # 5. Heatmap

```
pass_matrix <- matrix(air_df$Passengers, nrow = 12)
heatmap(pass_matrix, Rowv = NA, Colv = NA, main = "Heatmap: Months × Years")
```

 # 6. Violin Plot

```
ggplot(air_df, aes(x = MonthName, y = Passengers, fill = MonthName)) +
  geom_violin() + ggtitle("Violin Plot by Month")
```

 # 7. Ridgeline Plot

```
library(ggridges)
ggplot(air_df, aes(x = Passengers, y = MonthName, fill = MonthName)) +
  geom_density_ridges() + ggtitle("Ridgeline Plot by Month")
# 8. Histogram
hist(air_df$Passengers, col = "lightgreen",
     main = "Histogram of Passengers", xlab = "Passengers")
# 9. Density Plot
plot(density(air_df$Passengers), main = "Density Plot", col = "blue", lwd = 2)
# 10. Box Plot
boxplot(Passengers ~ Year, data = air_df, col = "yellow",
        main = "Box Plot by Year", las = 2)
```

2. Visualizing Proportions

```
# Load and prepare data
data("AirPassengers")
air_df <- data.frame(
  Passengers = as.numeric(AirPassengers),
  Year = factor(floor(time(AirPassengers))),
  Month = factor(cycle(AirPassengers)),
  MonthName = factor(month.abb[cycle(AirPassengers)], levels = month.abb)
)
# Aggregate data for proportions
year_totals <- aggregate(Passengers ~ Year, air_df, sum)
month_totals <- aggregate(Passengers ~ MonthName, air_df, sum)
# 1. PIE CHART
pie(year_totals$Passengers,
    labels = year_totals$Year,
    main = "Pie Chart: Proportion by Year",
    col = rainbow(nrow(year_totals)))
# 2. STACKED BAR (Proportional)
library(ggplot2)
# Stacked bar showing proportions (100% stacked)
```

```
ggplot(air_df, aes(x = Year, y = Passengers, fill = MonthName)) +
  geom_bar(position = "fill", stat = "identity") +
  labs(title = "Stacked Bar (100% Proportion)",
       y = "Proportion") +
  scale_y_continuous(labels = scales::percent)
# 3. TREEMAP
# install.packages("treemap")
library(treemap)
treemap(air_df,
        index = c("Year", "MonthName"),
        vSize = "Passengers",
        title = "Treemap: Passengers by Year & Month",
        palette = "Set3")
# 4. SUNBURST CHART
# install.packages("plotly")
library(plotly)
# Create hierarchical data
air_df$path <- paste(air_df$Year, air_df$MonthName, sep = "/")
plot_ly(
  labels = c(unique(air_df$Year), air_df$MonthName),
  parents = c(rep("", 12), rep(air_df$Year, each = 12)[1:144]),
  values = c(aggregate(Passengers ~ Year, air_df, sum)$Passengers, air_df$Passengers),
  type = 'sunburst',
  branchvalues = 'total'
) %>% layout(title = "Sunburst Chart")
# 5. PARALLEL SETS
# install.packages("ggparallel")
library(ggparallel)
# Simplify: Use first 3 years for clarity
air_subset <- air_df[air_df$Year %in% c("1949", "1950", "1951"), ]
ggparallel(list("Year", "MonthName"),
           data = air_subset,
```

```
        weight = "Passengers",
        title = "Parallel Sets: Year to Month Flow")
```

 # 6. SANKEY DIAGRAM

```
# install.packages("networkD3")
library(networkD3)
# Create nodes
nodes <- data.frame(name = c(levels(air_df$Year), levels(air_df$MonthName)))
# Create links (Year → Month connections)
links <- data.frame(
  source = as.numeric(air_df$Year) - 1,
  target = as.numeric(air_df$Month) + 11,
  value = air_df$Passengers
)
sankeyNetwork(Links = links, Nodes = nodes, Source = "source",
              Target = "target", Value = "value", NodeID = "name",
              fontSize = 12, nodeWidth = 20,
              title = "Sankey Diagram: Passengers Flow")
```

3. Visualizing Relationships and Associations

```
# Load and prepare data
data("AirPassengers")
air_df <- data.frame(
  Passengers = as.numeric(AirPassengers),
  Year = factor(floor(time(AirPassengers))),
  Month = factor(cycle(AirPassengers)),
  MonthName = factor(month.abb[cycle(AirPassengers)], levels = month.abb),
  Time = as.numeric(time(AirPassengers)),
  Lag1 = c(NA, as.numeric(AirPassengers)[-length(AirPassengers)])  # Lagged variable
)
# Remove NA for plots
air_complete <- na.omit(air_df)
# 1. SCATTERPLOT
# Passengers vs Time (with lag)
```

```
plot(air_complete$Time, air_complete$Passengers,
     pch = 19, col = "blue",
     main = "Scatterplot: Passengers vs Time",
     xlab = "Time", ylab = "Passengers")
# 2. BUBBLE PLOT
# Install if needed: install.packages("ggplot2")
library(ggplot2)

# Add a size variable (passenger count normalized)
air_complete$Size <- scale(air_complete$Passengers)
ggplot(air_complete, aes(x = Time, y = Passengers, size = Passengers, color = Year)) +
  geom_point(alpha = 0.7) +
  scale_size(range = c(1, 10)) +
  labs(title = "Bubble Plot: Passengers over Time",
       x = "Time", y = "Passengers") +
  theme_minimal()
# 3. HEXBIN PLOT
# Install if needed: install.packages("hexbin")
library(hexbin)
hexbinplot(Passengers ~ Time, data = air_complete,
           main = "Hexbin Plot: Density of Points",
           xlab = "Time", ylab = "Passengers",
           colramp = function(n) heat.colors(n))
# 4. CORRELOGRAM
# Install if needed: install.packages("corrplot")
library(corrplot)
# Create correlation matrix (select numeric columns)
numeric_data <- air_complete[, c("Passengers", "Time", "Lag1")]
cor_matrix <- cor(numeric_data)

corrplot(cor_matrix, method = "color",
         title = "Correlogram: Variable Correlations",
```

```
      mar = c(0, 0, 1, 0))
# 5. NETWORK GRAPH
# Install if needed: install.packages("igraph")
library(igraph)
# Create simple network: Months connected to years
edges <- data.frame(
  from = air_complete$Year,
  to = air_complete$MonthName,
  weight = air_complete$Passengers
)
# Create graph
g <- graph_from_data_frame(edges[1:30, ], directed = FALSE)  # First 30 for clarity
plot(g, main = "Network Graph: Year-Month Connections",
     vertex.color = "lightblue",
     vertex.size = 15,
     edge.width = E(g)$weight/100)  # Scale edge width
# 6. PCA (Principal Component Analysis)
# Create matrix with lagged variables
air_matrix <- cbind(
  air_complete$Passengers,
  air_complete$Lag1,
  as.numeric(air_complete$Month)
)
colnames(air_matrix) <- c("Passengers", "Lag1", "Month")
# Perform PCA
pca_result <- prcomp(air_matrix, scale = TRUE)
# Plot PCA
plot(pca_result$x[, 1:2], pch = 19, col = "darkgreen",
     main = "PCA Plot",
     xlab = "PC1", ylab = "PC2")
text(pca_result$x[, 1:2], labels = air_complete$Year, pos = 3, cex = 0.7)
# 7. t-SNE (Alternative to PCA)
```

```
# Install if needed: install.packages("Rtsne")
library(Rtsne)
# Run t-SNE
set.seed(123)
tsne_result <- Rtsne(air_matrix, dims = 2, perplexity = 5)
# Plot t-SNE
plot(tsne_result$Y, pch = 19, col = as.numeric(air_complete$Year),
    main = "t-SNE Plot",
    xlab = "Dimension 1", ylab = "Dimension 2")
# 8. UMAP
# Install if needed: install.packages("umap")
library(umap)
# Run UMAP
umap_result <- umap(air_matrix)
# Plot UMAP
plot(umap_result$layout, pch = 19, col = "purple",
    main = "UMAP Plot",
    xlab = "UMAP1", ylab = "UMAP2")
```

3. 4. Visualizing Time Series / Trends

```
# Load and prepare data
data("AirPassengers")
air_ts <- AirPassengers
# 1. LINE PLOT
plot(air_ts,
    main = "Line Plot: Air Passengers Over Time",
    xlab = "Year", ylab = "Passengers (thousands)",
    col = "blue", lwd = 2)
# 2. MULTIPLE LINE PLOT
# Decompose by year
years <- start(air_ts)[1]:end(air_ts)[1]
matplot(years, matrix(air_ts, nrow = 12, byrow = TRUE),
```

```
      type = "l", lty = 1, col = rainbow(12),
      main = "Multiple Lines: Yearly Patterns",
      xlab = "Year", ylab = "Passengers")
legend("topleft", legend = month.abb, col = rainbow(12), lty = 1, cex = 0.7
# 3. DOSE-RESPONSE CURVE (Seasonal pattern)
# Average by month
monthly_avg <- aggregate(air_ts, by = list(month = cycle(air_ts)), FUN = mean)
plot(monthly_avg$month, monthly_avg$x,
    type = "b", pch = 19, col = "red",
    main = "Dose-Response: Average by Month",
    xlab = "Month", ylab = "Average Passengers",
    xaxt = "n")
axis(1, at = 1:12, labels = month.abb)
# 4. SEASONAL DECOMPOSITION (STL)
# Install if needed: install.packages("ggplot2")
library(ggplot2)
# Using stl() for decomposition
air_stl <- stl(air_ts, s.window = "periodic")
plot(air_stl,
    main = "STL Decomposition: Trend, Seasonal, Residual")
# 5. SMOOTHING / MOVING AVERAGE
# Simple moving average (12-month window)
if (!require("zoo")) install.packages("zoo")
library(zoo)
ma_12 <- rollmean(air_ts, k = 12, align = "center", fill = NA)
plot(air_ts, col = "gray",
    main = "Moving Average Smoothing (12-month)",
    xlab = "Year", ylab = "Passengers")
lines(ma_12, col = "red", lwd = 2)
legend("topleft", legend = c("Original", "12-month MA"),
      col = c("gray", "red"), lty = 1)
# 6. SEASONAL SUB-SERIES PLOT (Extra)
```

```
# Install if needed: install.packages("forecast")
library(forecast)
ggseasonplot(air_ts, year.labels = TRUE, year.labels.left = TRUE) +
  ggtitle("Seasonal Sub-Series Plot") +
  ylab("Passengers") +
  xlab("Month")
# 7. TREND EXTRACTION (Extra)
# Using decompose()
air_decompose <- decompose(air_ts)
plot(air_decompose)
```

5. **Visualizing Geospatial Data**

```
# We'll use built-in US datasets for geospatial examples

# 1. CHOROPLETH MAP
# Install if needed: install.packages(c("maps", "ggplot2", "viridis"))
library(maps)
library(ggplot2)
library(viridis)

# US state data
us_states <- map_data("state")

# Create sample data (crime rates by state)
data(USArrests)
USArrests$region <- tolower(rownames(USArrests))

# Merge with map data
map_data <- merge(us_states, USArrests, by = "region", all.x = TRUE)

# Choropleth map
ggplot(map_data, aes(x = long, y = lat, group = group, fill = Murder)) +
  geom_polygon(color = "white") +
  scale_fill_viridis(name = "Murder Rate") +
  theme_void() +
  ggtitle("Choropleth Map: US Murder Rates by State")

# 2. (SPATIAL) DENSITY PLOT
# Using crime data with latitude/longitude (simulated)
set.seed(123)
n_points <- 1000
crime_sim <- data.frame(
  long = runif(n_points, -125, -65),  # US longitudes
  lat = runif(n_points, 25, 50)       # US latitudes
)
```

```
# Density plot
ggplot(crime_sim, aes(x = long, y = lat)) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +
  geom_point(alpha = 0.1, size = 0.5) +
  scale_fill_viridis(name = "Density") +
  theme_void() +
  ggtitle("Spatial Density Plot: Simulated Crime Density")

# 3. 3D GEOSPATIAL
# Install if needed: install.packages("plotly")
library(plotly)

# 3D surface plot (elevation example)
data(volcano)  # Maunga Whau volcano in New Zealand

plot_ly(z = ~volcano, type = "surface") %>%
  layout(title = "3D Geospatial: Volcano Elevation",
         scene = list(
           xaxis = list(title = "X"),
           yaxis = list(title = "Y"),
           zaxis = list(title = "Elevation")
         ))

# 4. INTERACTIVE MAPS
# Install if needed: install.packages("leaflet")
library(leaflet)

# Create interactive map with markers
leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap tiles
  addMarkers(
    lng = -74.0060, lat = 40.7128,  # New York
    popup = "New York City"
  ) %>%
  addMarkers(
    lng = -118.2437, lat = 34.0522,  # Los Angeles
    popup = "Los Angeles"
  ) %>%
  addMarkers(
    lng = -87.6298, lat = 41.8781,  # Chicago
    popup = "Chicago"
  ) %>%
  setView(lng = -96, lat = 37.8, zoom = 4) %>%
  addControl("Interactive US Map", position = "topright")

# 5. SIMPLE POINT MAP (Extra)
# Plot US cities
data(us.cities)
big_cities <- subset(us.cities, pop > 500000)
```

```
plot(big_cities$long, big_cities$lat,
    pch = 19, col = "red", cex = 0.5,
    main = "Point Map: Major US Cities",
    xlab = "Longitude", ylab = "Latitude")
maps::map("state", add = TRUE)
```

6. 6. Visualizing Uncertainty

```
# Load AirPassengers dataset (already in R)
data("AirPassengers")
ap <- as.numeric(AirPassengers)
time_index <- 1:length(ap)

# 1. CONFIDENCE INTERVAL
plot(time_index, ap, pch=19, main="AirPassengers: Confidence Interval", xlab="Time", ylab="Passengers")
mod <- lm(ap ~ time_index)
abline(mod, col="red", lwd=2)
ci <- predict(mod, data.frame(time_index=time_index), interval="confidence")
lines(time_index, ci[,2], col="red", lty=2)
lines(time_index, ci[,3], col="red", lty=2)

# 2. BOOTSTRAPPING
hist(replicate(500, mean(sample(ap, replace=TRUE))),
    main="AirPassengers: Bootstrapped Means",
    xlab="Mean Passengers", col="lightblue")

# 3. JITTER & TRANSPARENCY
plot(jitter(time_index), jitter(ap),
    pch=19, col=rgb(0.2,0.5,0.8,0.3), cex=0.8,
    main="AirPassengers: Jitter & Transparency",
    xlab="Time", ylab="Passengers")

# 4. 2D HISTOGRAM
```

```
smoothScatter(time_index, ap,
              main="AirPassengers: 2D Histogram",
              xlab="Time", ylab="Passengers")

# 5. CONTOUR LINES
library(MASS)
kde <- kde2d(time_index, ap)
contour(kde, main="AirPassengers: Contour Lines",
        xlab="Time", ylab="Passengers")
```