HW5-UrbanLandCover

2025-11-22

1. Data Gathering and Integration

Already done, got the data from UCI repository and did feature exploration - Only has numeric predictors but target labels are categorical - Include in the explanation why I really wanted ot use this dataset (GIS focused)

```
#Import Data sets
test <- read.csv("/Users/kurtis/Desktop/UrbanLandCover/testing.csv")
train <- read.csv("/Users/kurtis/Desktop/UrbanLandCover/training.csv")

#View both datasets
View(test)
View(train)

#Keep only the 'class' column and all columns ending in "_80"
train_subset <- train[, c("class", grep("_80$", names(train), value = TRUE))]
test_subset <- test[, c("class", grep("_80$", names(test), value = TRUE))]

#View the subsets
View(train_subset)
View(test_subset)

#Get the structure of the datasets
str(train_subset) #168 x 22
```

```
## 'data.frame':    168 obs. of  22 variables:
##  $ class       : chr  "car " "concrete " "concrete " "concrete " ...
##  $ BrdIndx_80  : num  1.33 4.68 2.05 3.5 2.53 4.15 3.16 1.25 1.16 2.17 ...
##  $ Area_80     : int  97 5544 4368 7622 8862 2517 2329 225 3120 657 ...
##  $ Round_80    : num  1.12 1.93 1.27 1.91 2.1 1.97 1.73 0.64 0.48 0.92 ...
##  $ Bright_80   : num  227 206 224 232 196 ...
##  $ Compact_80  : num  1.32 3.17 1.98 2.83 2.89 2.88 2.56 1.33 1.33 2.36 ...
##  $ ShpIndx_80  : num  1.42 5.12 2.22 4.19 6.57 4.39 3.5 1.33 1.17 2.24 ...
##  $ Mean_NIR_80 : num  204 179 203 205 181 ...
##  $ Mean_R_80   : num  237 218 234 246 204 ...
##  $ Mean_G_80   : num  240 220 236 245 205 ...
##  $ SD_NIR_80   : num  27.6 18.9 13.1 10.4 16 ...
##  $ SD_R_80     : num  28.36 19.94 13.26 9.36 16.5 ...
##  $ SD_G_80     : num  26.18 19.61 12.99 9.35 17.48 ...
##  $ LW_80       : num  2 2.26 2.19 3.38 24.83 ...
##  $ GLCM1_80    : num  0.5 0.74 0.75 0.67 0.84 0.81 0.88 0.76 0.87 0.88 ...
##  $ Rect_80     : num  0.85 0.48 0.71 0.62 0.59 0.62 0.57 0.87 0.91 0.74 ...
##  $ GLCM2_80    : num  6.29 8.14 7.16 7.01 7.97 8.19 8.46 7.17 7.38 8.14 ...
```

```
##  $ Dens_80     : num  1.67 1.28 1.61 1.16 0.43 1.5 1.3 1.96 2.34 2 ...
##  $ Assym_80    : num  0.7 0.78 0.82 0.93 1 0.71 0.88 0.44 0.11 0.35 ...
##  $ NDVI_80     : num  -0.08 -0.1 -0.07 -0.09 -0.06 0.3 -0.07 -0.1 -0.1 0.31 ...
##  $ BordLngth_80: int  56 1526 586 1464 2474 880 676 80 262 230 ...
##  $ GLCM3_80    : num  3806 1758 1245 1785 1130 ...
```

```
str(test_subset) #507 x 22
```

```
## 'data.frame':    507 obs. of  22 variables:
##  $ class       : chr  "concrete " "shadow " "shadow " "tree " ...
##  $ BrdIndx_80  : num  3.65 1.07 2.3 2.86 3.7 2.99 2.59 2.67 2.36 3.78 ...
##  $ Area_80     : int  1522 1377 2901 871 1585 1407 2650 3069 4905 1317 ...
##  $ Round_80    : num  2.7 0.46 2 2.22 1.48 1.5 1.36 1.55 1.15 2.24 ...
##  $ Bright_80   : num  181 49.8 46.6 90.2 109.1 ...
##  $ Compact_80  : num  5.48 1.07 2.78 3.34 2.7 2.52 1.86 2.42 1.85 4.29 ...
##  $ ShpIndx_80  : num  3.69 1.32 2.79 3.93 3.77 3.11 2.64 2.92 2.46 4.06 ...
##  $ Mean_NIR_80 : num  155.8 38.9 40 117.2 94.2 ...
##  $ Mean_R_80   : num  192.3 51.2 46.3 72.8 114.1 ...
##  $ Mean_G_80   : num  195 59.4 53.4 80.7 119 ...
##  $ SD_NIR_80   : num  34.9 10.1 14.5 42.1 27.3 ...
##  $ SD_R_80     : num  39.5 11.9 11.8 27.1 26.6 ...
##  $ SD_G_80     : num  39.2 12 12.3 27.4 25.8 ...
##  $ LW_80       : num  1.13 3.7 3.57 5.25 1.08 1.54 1.38 2.21 1.7 2 ...
##  $ GLCM1_80    : num  0.73 0.52 0.65 0.84 0.79 0.66 0.76 0.83 0.87 0.81 ...
##  $ Rect_80     : num  0.28 0.96 0.6 0.51 0.71 0.66 0.81 0.58 0.78 0.48 ...
##  $ GLCM2_80    : num  8.4 7.01 7.11 8.56 8.27 7.09 7.85 7.93 8.11 8.6 ...
##  $ Dens_80     : num  1.21 1.69 1.16 0.89 1.83 1.81 2.05 1.4 2.02 1.21 ...
##  $ Assym_80    : num  0.23 0.86 0.93 0.95 0.2 0.34 0.45 0.84 0.49 0.77 ...
##  $ NDVI_80     : num  -0.11 -0.14 -0.07 0.23 -0.1 -0.03 -0.13 -0.01 0.13 -0.08 ...
##  $ BordLngth_80: int  576 196 602 464 600 466 544 646 688 590 ...
##  $ GLCM3_80    : num  1790 2660 1432 1235 1397 ...
```

```
#Join the datasets together by rows
urban <- rbind(train_subset, test_subset)
View(urban)
str(urban) #675 x 22
```

```
## 'data.frame':    675 obs. of  22 variables:
##  $ class       : chr  "car " "concrete " "concrete " "concrete " ...
##  $ BrdIndx_80  : num  1.33 4.68 2.05 3.5 2.53 4.15 3.16 1.25 1.16 2.17 ...
##  $ Area_80     : int  97 5544 4368 7622 8862 2517 2329 225 3120 657 ...
##  $ Round_80    : num  1.12 1.93 1.27 1.91 2.1 1.97 1.73 0.64 0.48 0.92 ...
##  $ Bright_80   : num  227 206 224 232 196 ...
##  $ Compact_80  : num  1.32 3.17 1.98 2.83 2.89 2.88 2.56 1.33 1.33 2.36 ...
##  $ ShpIndx_80  : num  1.42 5.12 2.22 4.19 6.57 4.39 3.5 1.33 1.17 2.24 ...
##  $ Mean_NIR_80 : num  204 179 203 205 181 ...
##  $ Mean_R_80   : num  237 218 234 246 204 ...
##  $ Mean_G_80   : num  240 220 236 245 205 ...
##  $ SD_NIR_80   : num  27.6 18.9 13.1 10.4 16 ...
##  $ SD_R_80     : num  28.36 19.94 13.26 9.36 16.5 ...
##  $ SD_G_80     : num  26.18 19.61 12.99 9.35 17.48 ...
##  $ LW_80       : num  2 2.26 2.19 3.38 24.83 ...
##  $ GLCM1_80    : num  0.5 0.74 0.75 0.67 0.84 0.81 0.88 0.76 0.87 0.88 ...
```

```
##  $ Rect_80     : num  0.85 0.48 0.71 0.62 0.59 0.62 0.57 0.87 0.91 0.74 ...
##  $ GLCM2_80    : num  6.29 8.14 7.16 7.01 7.97 8.19 8.46 7.17 7.38 8.14 ...
##  $ Dens_80     : num  1.67 1.28 1.61 1.16 0.43 1.5 1.3 1.96 2.34 2 ...
##  $ Assym_80    : num  0.7 0.78 0.82 0.93 1 0.71 0.88 0.44 0.11 0.35 ...
##  $ NDVI_80     : num  -0.08 -0.1 -0.07 -0.09 -0.06 0.3 -0.07 -0.1 -0.1 0.31 ...
##  $ BordLngth_80: int  56 1526 586 1464 2474 880 676 80 262 230 ...
##  $ GLCM3_80    : num  3806 1758 1245 1785 1130 ...
```

```
#Extract Column names
col_names <- names(urban)
print(col_names)
```

```
##  [1] "class"        "BrdIndx_80"   "Area_80"      "Round_80"     "Bright_80"
##  [6] "Compact_80"   "ShpIndx_80"   "Mean_NIR_80"  "Mean_R_80"    "Mean_G_80"
## [11] "SD_NIR_80"    "SD_R_80"      "SD_G_80"      "LW_80"        "GLCM1_80"
## [16] "Rect_80"      "GLCM2_80"     "Dens_80"      "Assym_80"     "NDVI_80"
## [21] "BordLngth_80" "GLCM3_80"
```

```
#Create an ordering of the columns by the column names extracted earlier
new_order <- c(
  "class",

    # Spectral Variables
  "Bright_80", "Mean_G_80", "Mean_R_80", "Mean_NIR_80", "NDVI_80",

    # Shape Variables
  "Area_80", "BordLngth_80", "BrdIndx_80", "Round_80", "Compact_80",
  "ShpIndx_80", "LW_80", "Rect_80", "Dens_80", "Assym_80",

    # Texture Variables
  "SD_G_80", "SD_R_80", "SD_NIR_80", "GLCM1_80", "GLCM2_80", "GLCM3_80"
)

#Apply the desired ordering
urban <- urban[, new_order]
View(urban)
```

2. Data Exploration

- Using data exploration to understand what is happening is important throughout the pipeline, and is not limited to this step. However, it is important to use some exploration early on to make sure you understand your data.
- You must at least consider the distributions of each variable and at least some of the relationships between pairs of variables.

```
#Check for missing values
sum(is.na(urban)) #0 missing values
```

```
## [1] 0
```

```r
#Get the distribution of the target variable 'class'
table(urban$class)
```

```
##
##  asphalt  building       car  concrete     grass      pool    shadow      soil
##       59       122        36       116       112        29        61        34
##     tree
##      106
```

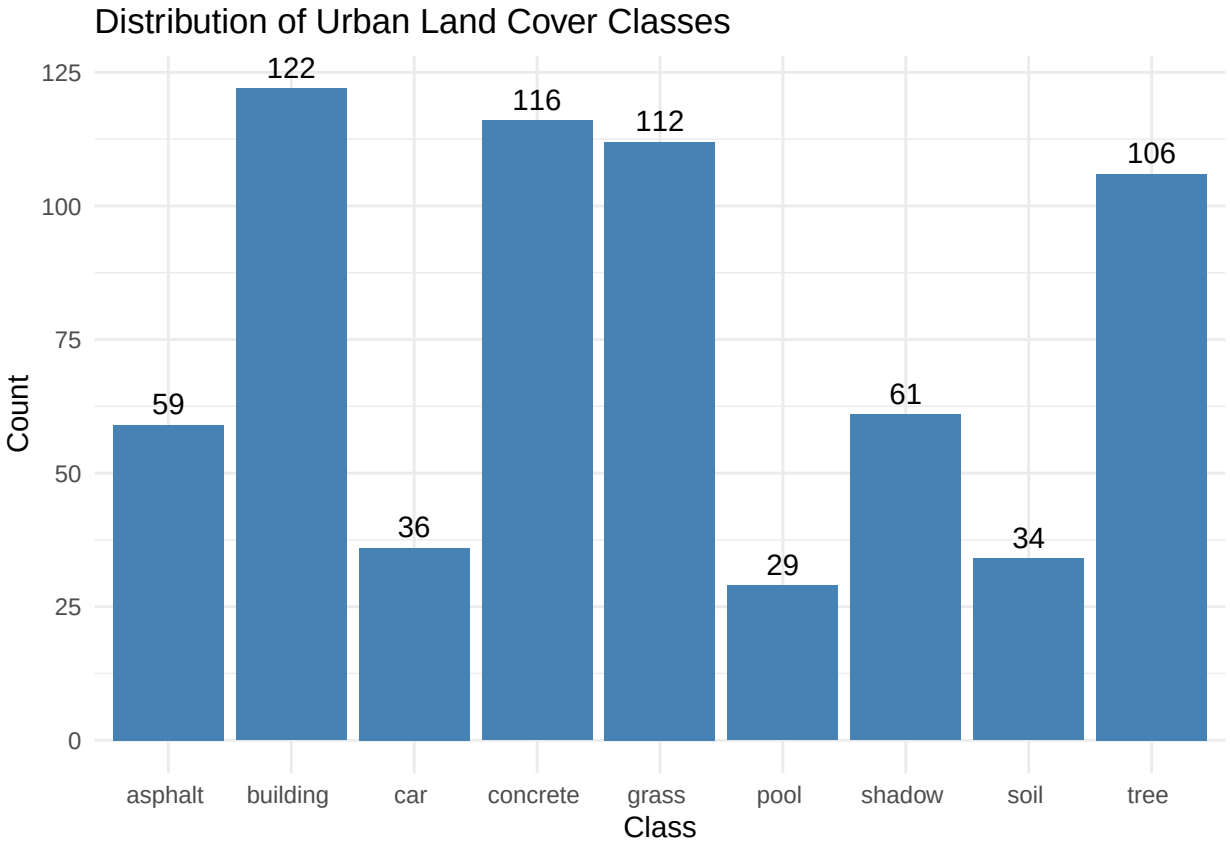
```r
#Count the number of instances in each class we have for the target
  # We have 9 classes total
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
class_counts <- urban %>%
  count(class)

#Plot the distribution of the target variable
library(ggplot2)
ggplot(class_counts, aes(x = class, y = n)) +
  geom_col(fill = "steelblue") +
  geom_text(aes(label = n), vjust = -0.5, size = 4) +
  labs(
    title = "Distribution of Urban Land Cover Classes",
    x = "Class",
    y = "Count"
  ) +
  theme_minimal()
```



- Not a very even distribution at all. Dominant classes are building, concrete, grass, tree - Roughly 3 levels of observation counts - High: building, concrete, grass, tree (100+) - Medium: asphalt, shadow (50-100) - Low: car, pool, soil(less than 50)

- Could be something to look into

Variable Correlations - within (spectral, shape, and texture) classifications

- Create three new datasets that subset the original dataset 'urban' by using the classifications of the variables (spectral, shape, and texture) variables based on the classifications we previouslt used. Then once those are created, I want you to create a correlation matrix for each of those datasets and visualize them using a heatmap plot. You can use any R package you like to create the heatmaps.

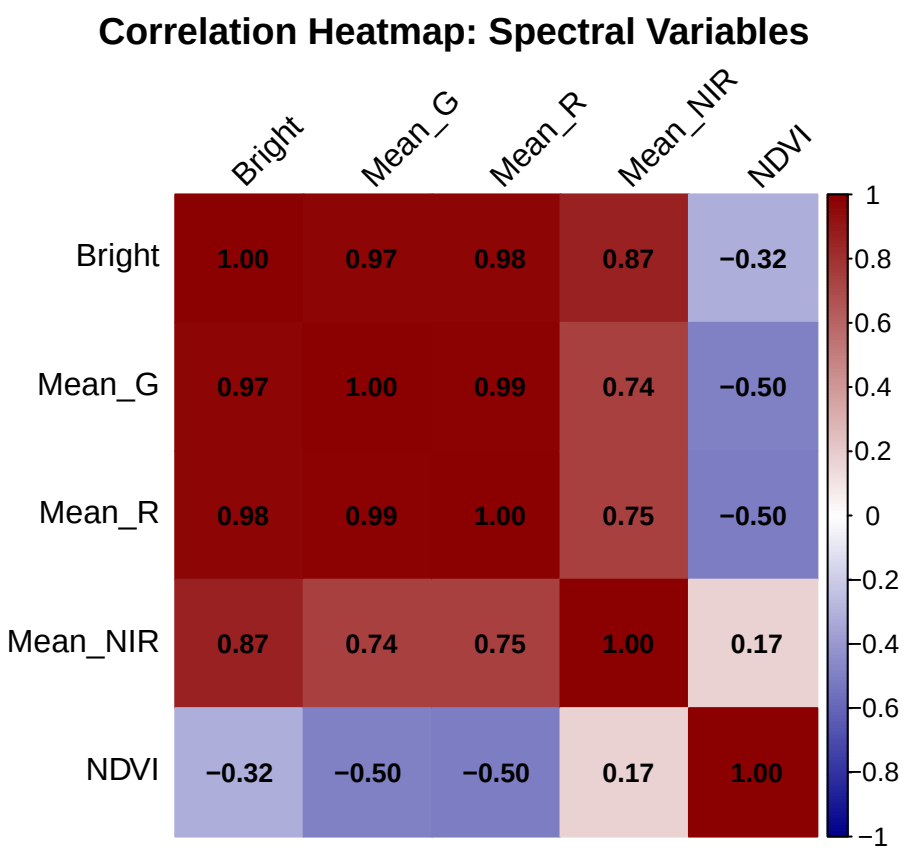
```
## ---- Packages ----
library(dplyr)
library(ggcorrplot)

#Before I go an further, remove the "_80" from the column names for easier reference
colnames(urban) <- gsub("_80", "", colnames(urban))
View(urban)

## ---- 1. Define variable groups ----

# Spectral variables
spectral_vars <- c(
```

```
  "Bright",
  "Mean_G",
  "Mean_R",
  "Mean_NIR",
  "NDVI"
)

# Shape variables
shape_vars <- c(
  "Area",
  "BordLngth",
  "BrdIndx",
  "Round",
  "Compact",
  "ShpIndx",
  "LW",
  "Rect",
  "Dens",
  "Assym"
)

# Texture variables
texture_vars <- c(
  "SD_G",
  "SD_R",
  "SD_NIR",
  "GLCM1",
  "GLCM2",
  "GLCM3"
)

## ---- 2. Create subset datasets ----

urban_spectral <- urban %>%
  select(all_of(spectral_vars))

urban_shape <- urban %>%
  select(all_of(shape_vars))

urban_texture <- urban %>%
  select(all_of(texture_vars))

library(dplyr)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

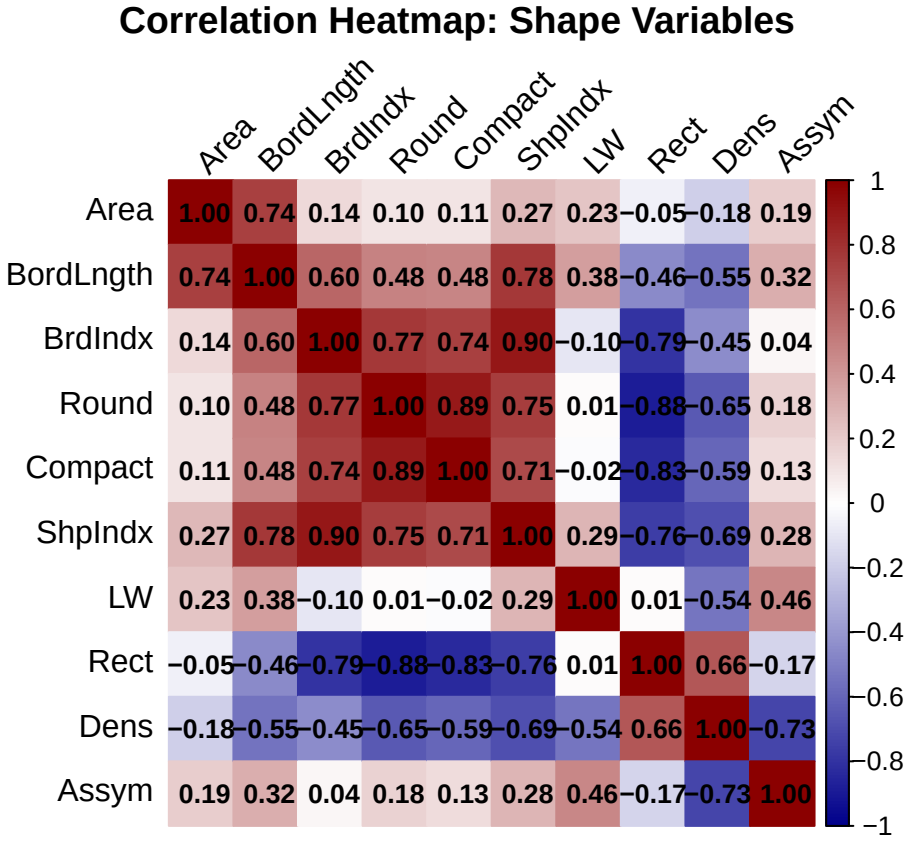
```
## ---- 3. Compute correlation matrices ----

cor_spectral <- cor(urban_spectral, use = "complete.obs")
cor_shape    <- cor(urban_shape,    use = "complete.obs")
cor_texture  <- cor(urban_texture,  use = "complete.obs")
```

```r
## ---- 4. Visualize using corrplot (upper triangle + diagonal) ----

# Color palette: blue (neg) → white → red (pos)
col_blue_red <- colorRampPalette(c("darkblue", "white", "darkred"))(200)


# Spectral heatmap
corrplot(
  cor_spectral,
  type = "full",        # upper triangle + diagonal
  method = "color",
  col = col_blue_red,
  tl.col = "black",
  tl.srt = 45,
  diag = TRUE,           # keep diagonal visible
  addCoef.col = "black",   # <-- prints correlation values
  number.cex = 0.8,        # <-- text size
  title = "Correlation Heatmap: Spectral Variables",
  mar = c(0,0,2,0)
)
```

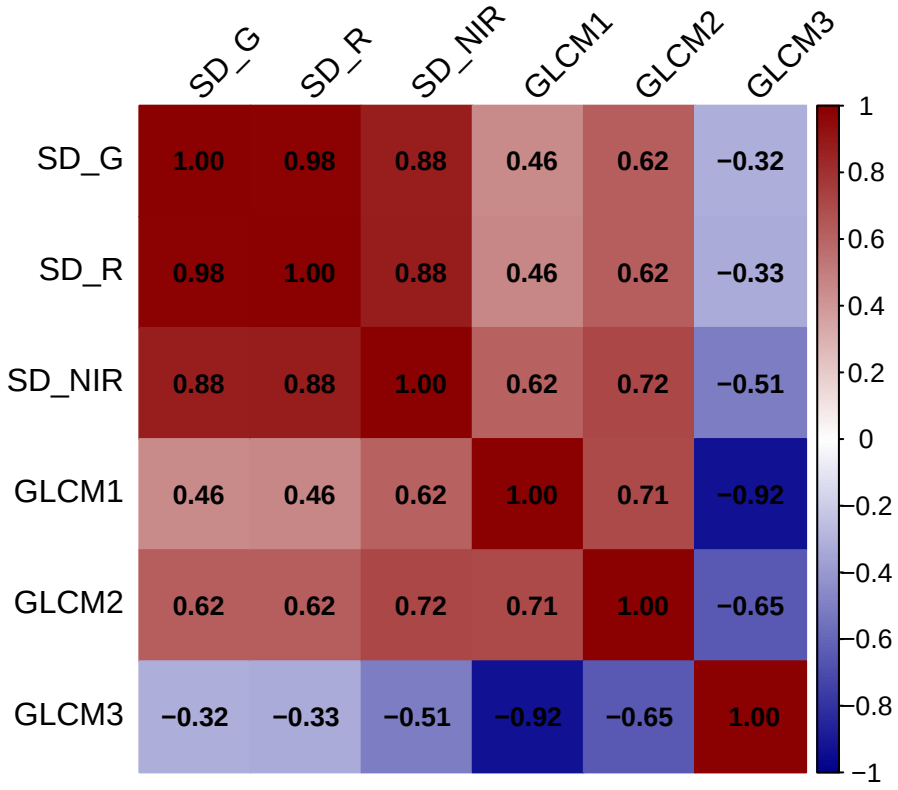



```r
# Shape heatmap
corrplot(
  cor_shape,
  type = "full",
```

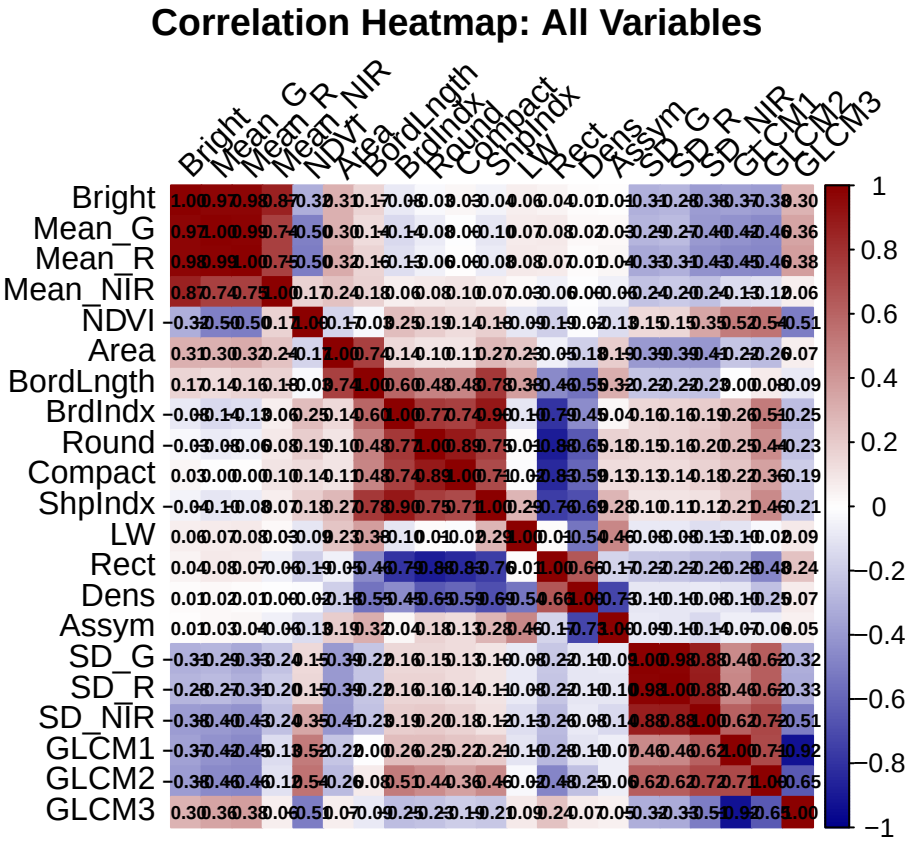
```
  method = "color",
  col = col_blue_red,
  tl.col = "black",
  tl.srt = 45,
  diag = TRUE,
  addCoef.col = "black",    # <-- prints correlation values
  number.cex = 0.8,         # <-- text size
  title = "Correlation Heatmap: Shape Variables",
  mar = c(0,0,2,0)
)
```




```
# Texture heatmap
corrplot(
  cor_texture,
  type = "full",
  method = "color",
  col = col_blue_red,
  tl.col = "black",
  tl.srt = 45,
  diag = TRUE,
  addCoef.col = "black",    # <-- prints correlation values
  number.cex = 0.8,         # <-- text size
  title = "Correlation Heatmap: Texture Variables",
  mar = c(0,0,2,0)
)
```




```r
# Now do a correlation plot of all the variables together
cor_all <- cor(urban[, -1], use = "complete.obs") # Exclude 'class' column

corrplot(
  cor_all,
  type = "full",
  method = "color",
  col = col_blue_red,
  tl.col = "black",
  tl.srt = 45,
  diag = TRUE,
  addCoef.col = "black",    # <-- prints correlation values
  number.cex = 0.6,         # <-- text size
  title = "Correlation Heatmap: All Variables",
  mar = c(0,0,2,0)
)
```



Observations: - Spectral Variables: - All Means have high positive correlations with all others - NDVI has negative correlations with all others - Shape Variables: - Rect and Dens have the most strong negative correlations with others - ShpIndx, BordLngth, Compact, BrdIndx have the most strong positive correlations with others - Texture Variables: - All the SD's have strong positive correlations with each other - GLCM3 has negative correlations with everything

From All plot: - Mean_R and Mean_G are basically the same variable, we can choose to jsut keep only one of them - Many of the strongest correlations are only within the classifications, as we see three main sub-groups of correlations - I would also argue that the negative are also pretty relevant, because there are many negative correlations between the spectral and the texture variables, those could make up a Principl Component - Maybe we

3. Data Cleaning

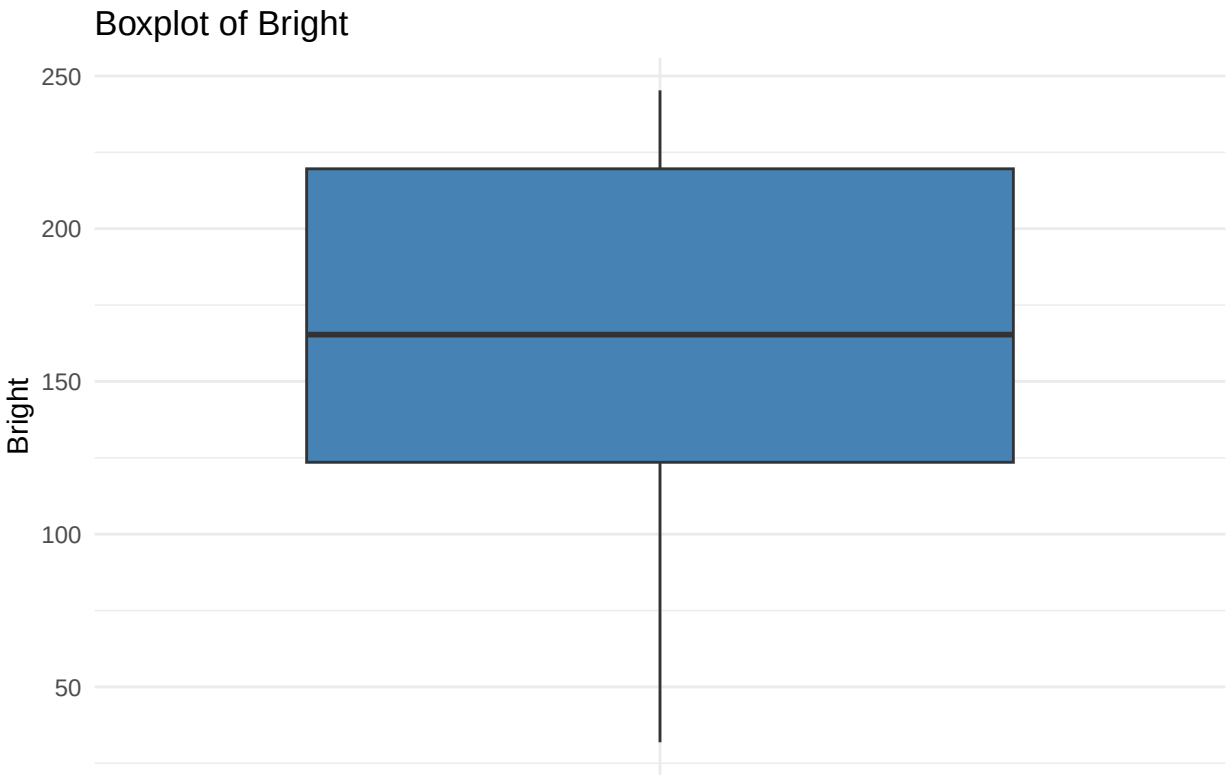
Don't forget – this can take a lot of the time of the whole process. Your cleaning process must ensure that there are no missing values and all outliers must be considered. It may be reasonable to just remove rows with missing values, however, if your data or small or that would change the distributions of the variables, that will not be adequate and you will need to consider other options, as discussed in the modules on cleaning. - Depending on your data and what you plan to do with it, you may also need to apply other processes we discussed. For example, clean up strings for consistency, deal with date formatting, change variable types between categorical and numeric, bin, smooth, group, aggregate or reshape. - Make the case with visualization or by showing resulting summary statistics that your data are clean enough to continue with your analysis.

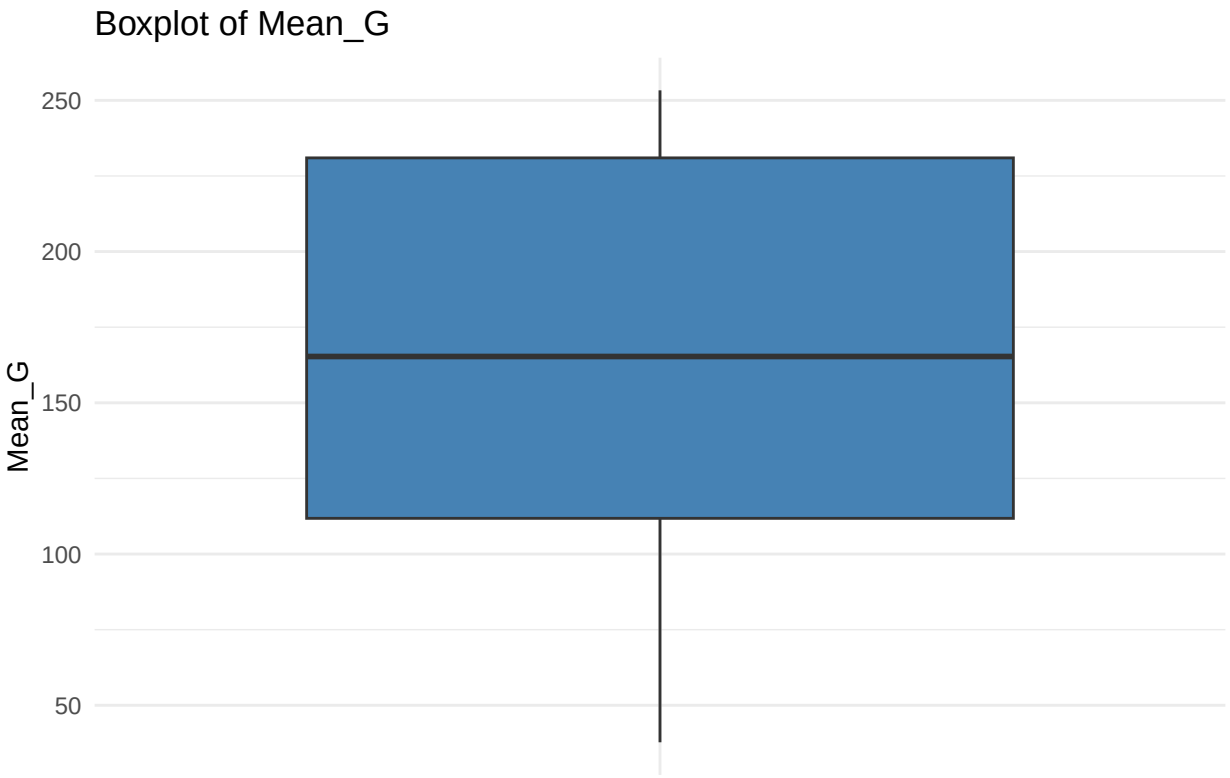
Feature Boxplots, Histograms to Examine Outliers

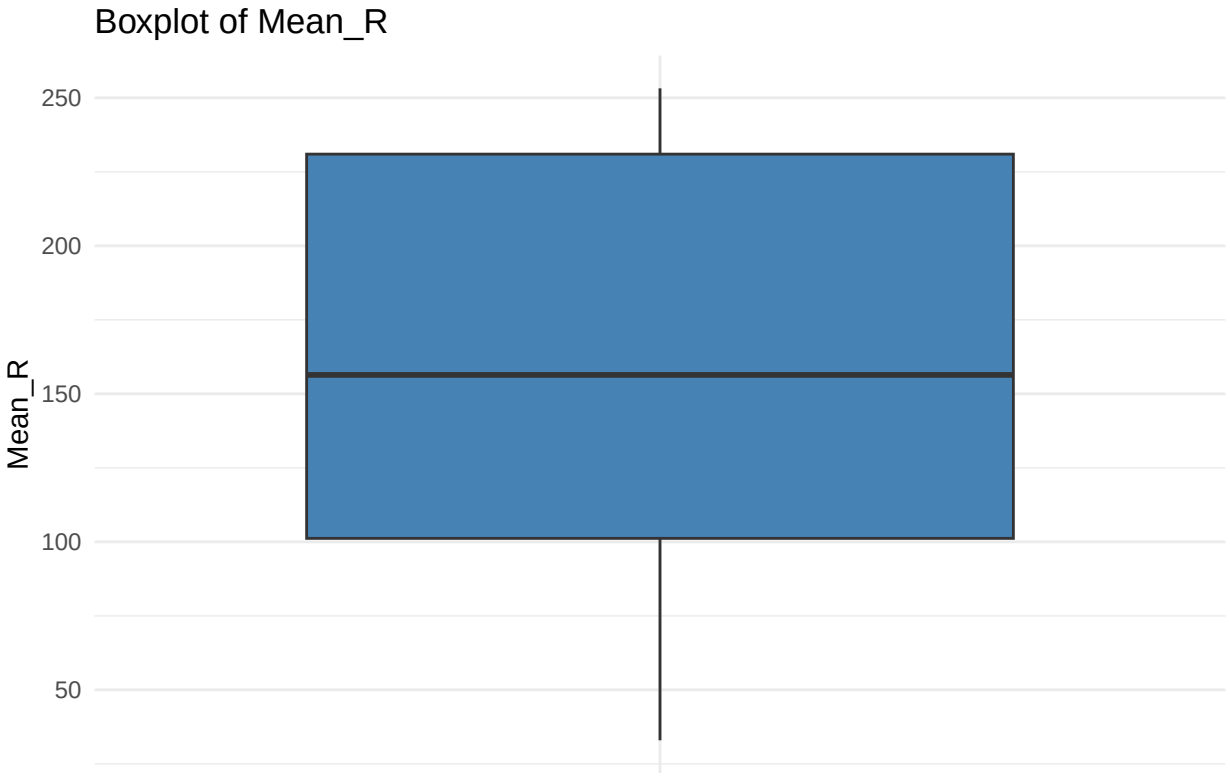
```
#Check for missing values again
sum(is.na(urban)) #0 missing values
```

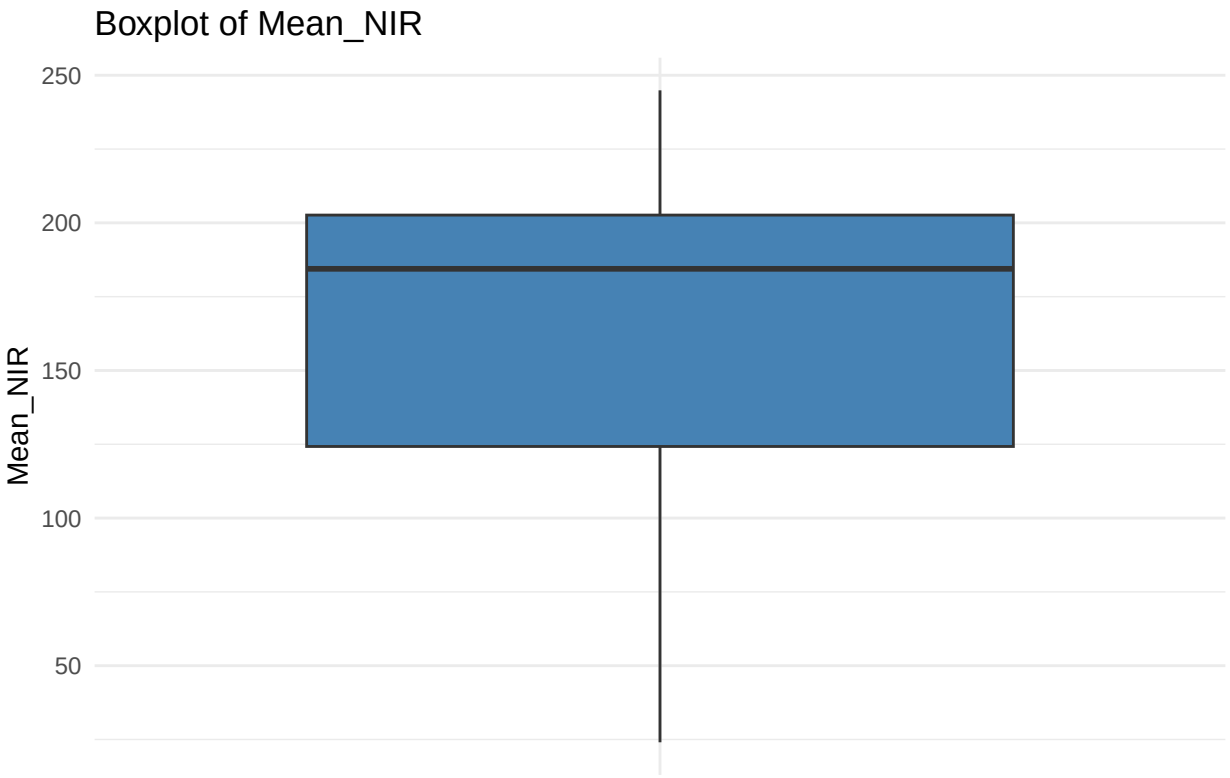
```
## [1] 0
```

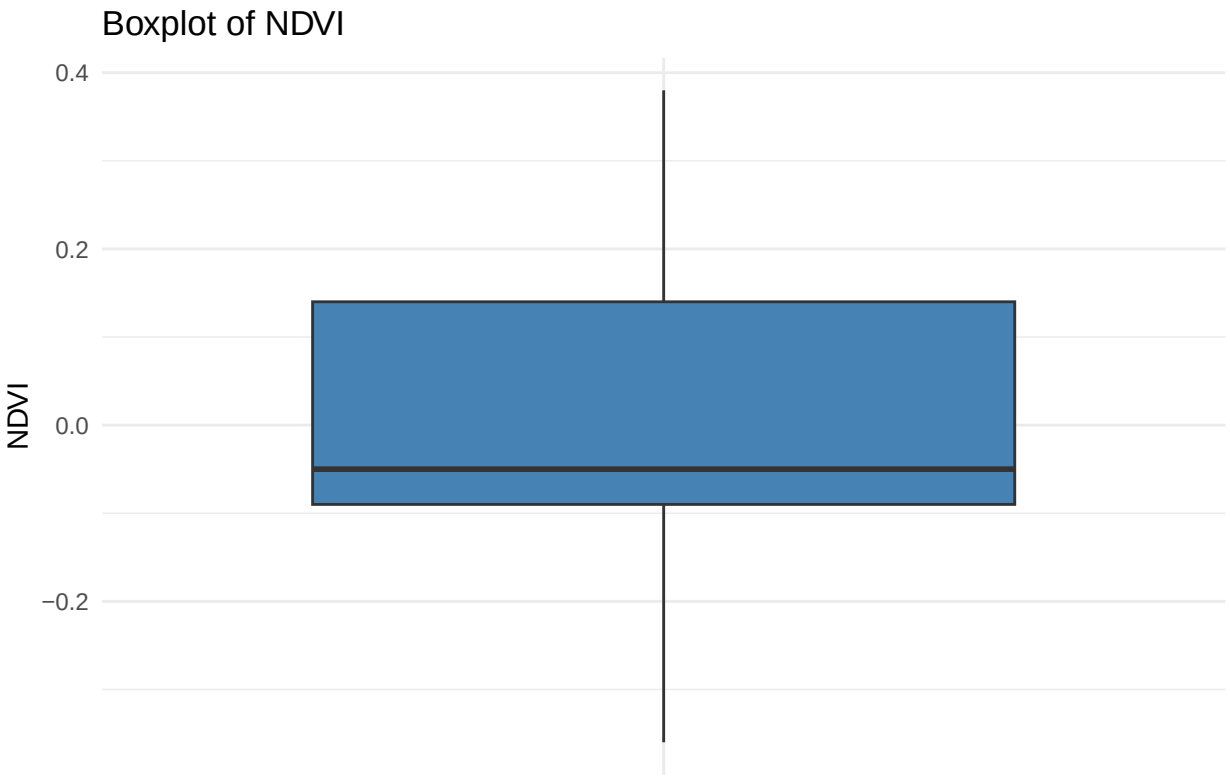
```
#Check for outliers using boxplots for each numeric variable
library(ggplot2)
library(dplyr)

# Select only numeric variables
numeric_vars <- urban %>%
  select(where(is.numeric), -class)  # remove class

# Loop through each numeric variable
for (var in names(numeric_vars)) {

  print(
    ggplot(urban, aes(x = "", y = .data[[var]])) +
      geom_boxplot(fill = "steelblue") +
      labs(
        title = paste("Boxplot of", var),
        x = "",
        y = var
      ) +
      theme_minimal()
  )

}
```

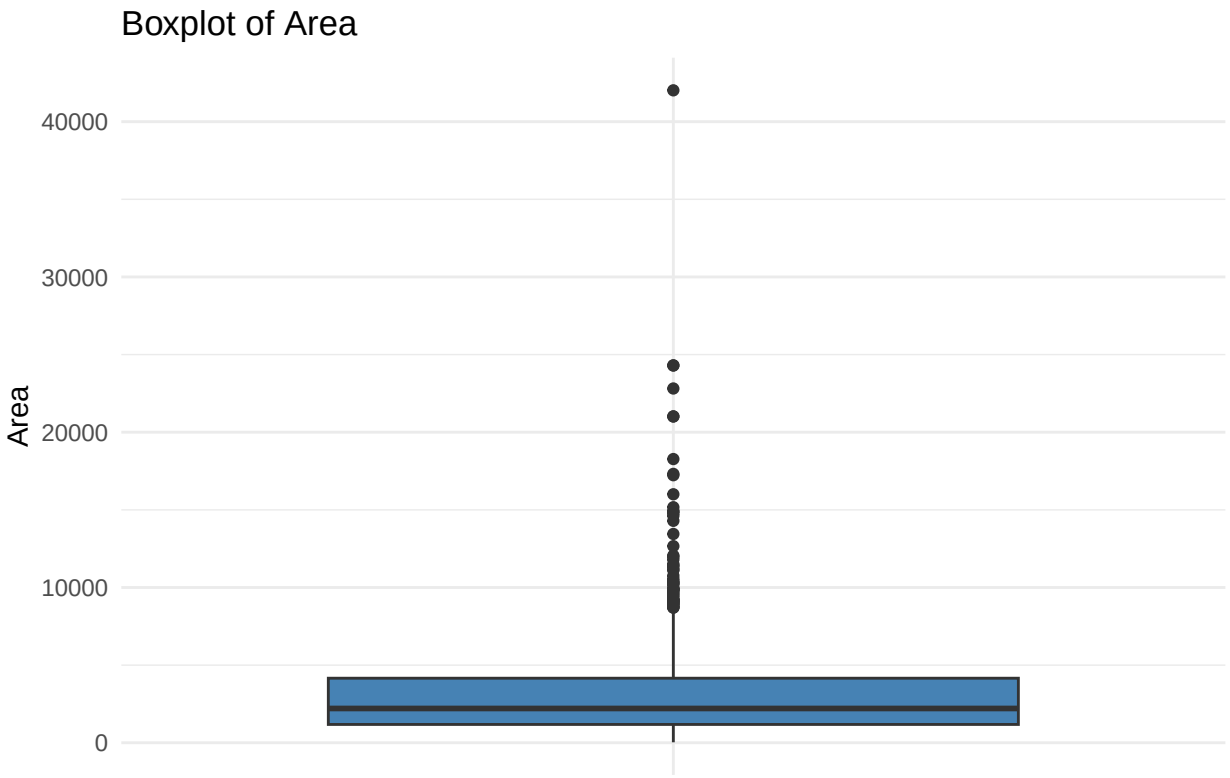

Boxplot of Bright
Bright
250
200
150
100
50

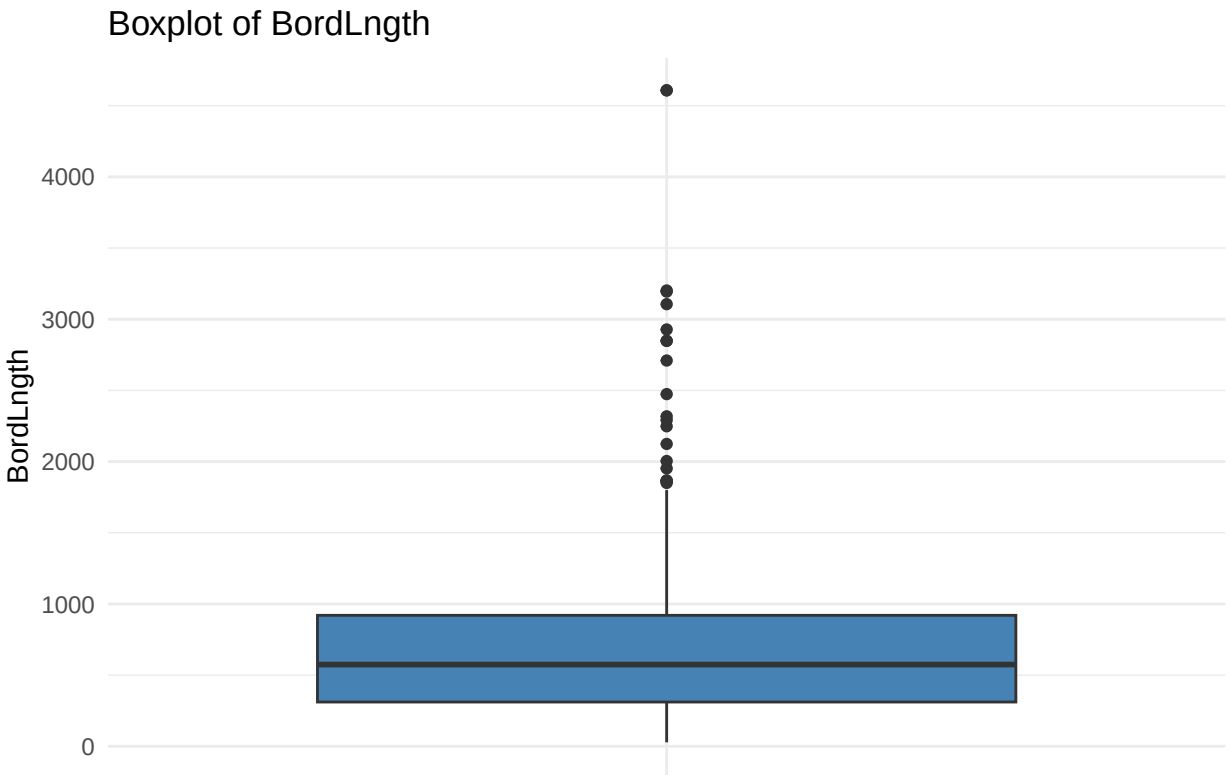

Boxplot of Mean_G
Mean_G
250
200
150
100
50

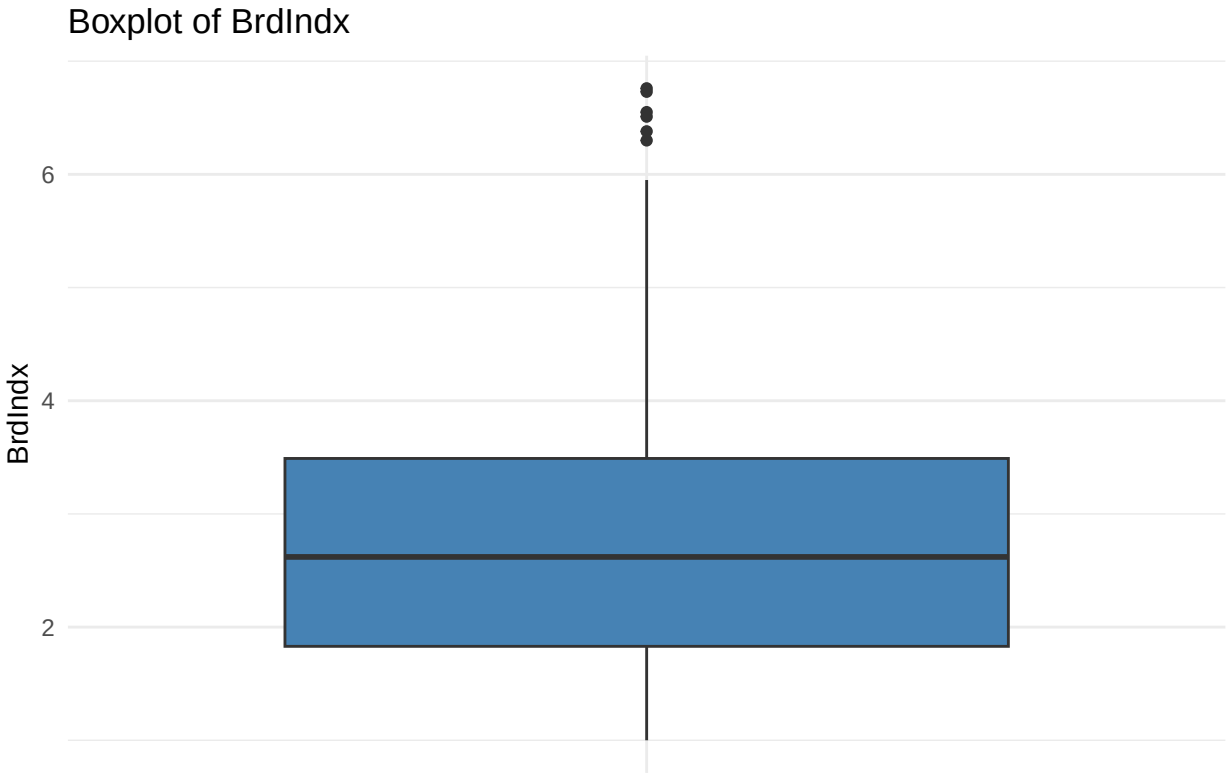

Boxplot of Mean_R
Mean_R
250
200
150
100
50

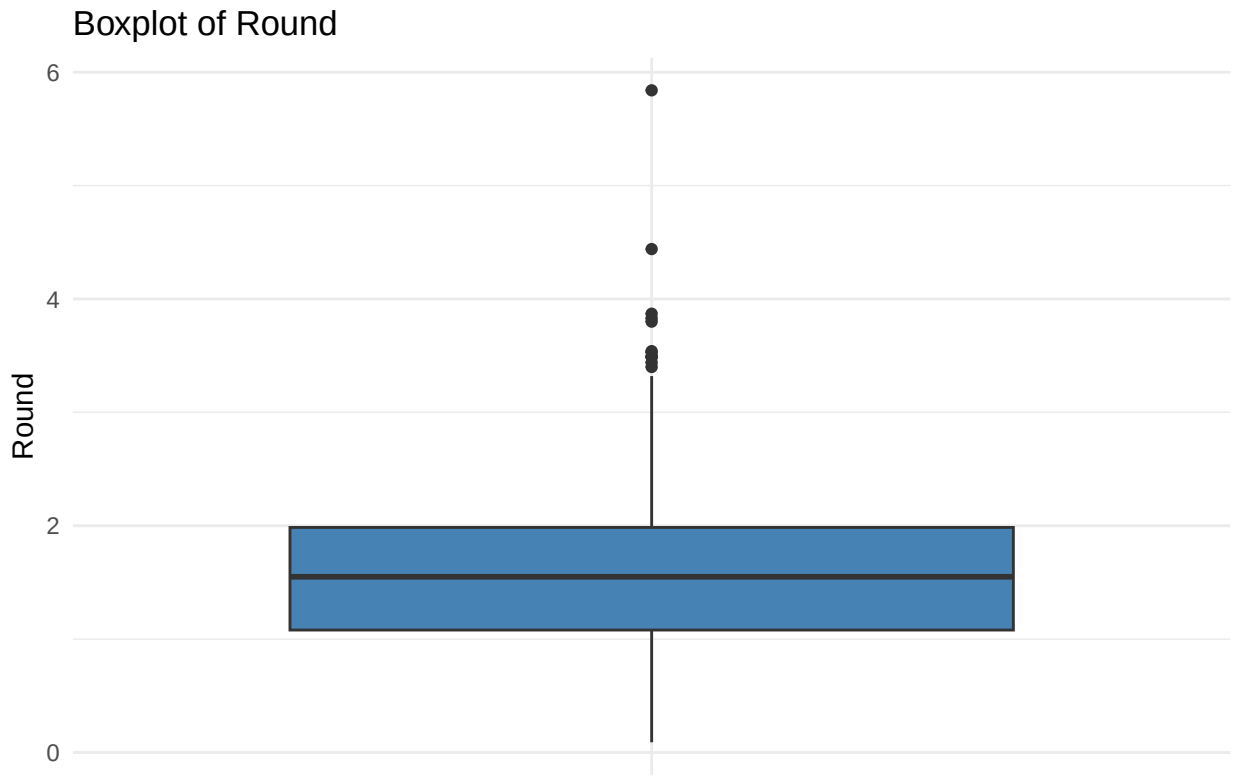

Boxplot of Mean_NIR
Mean_NIR
250
200
150
100
50

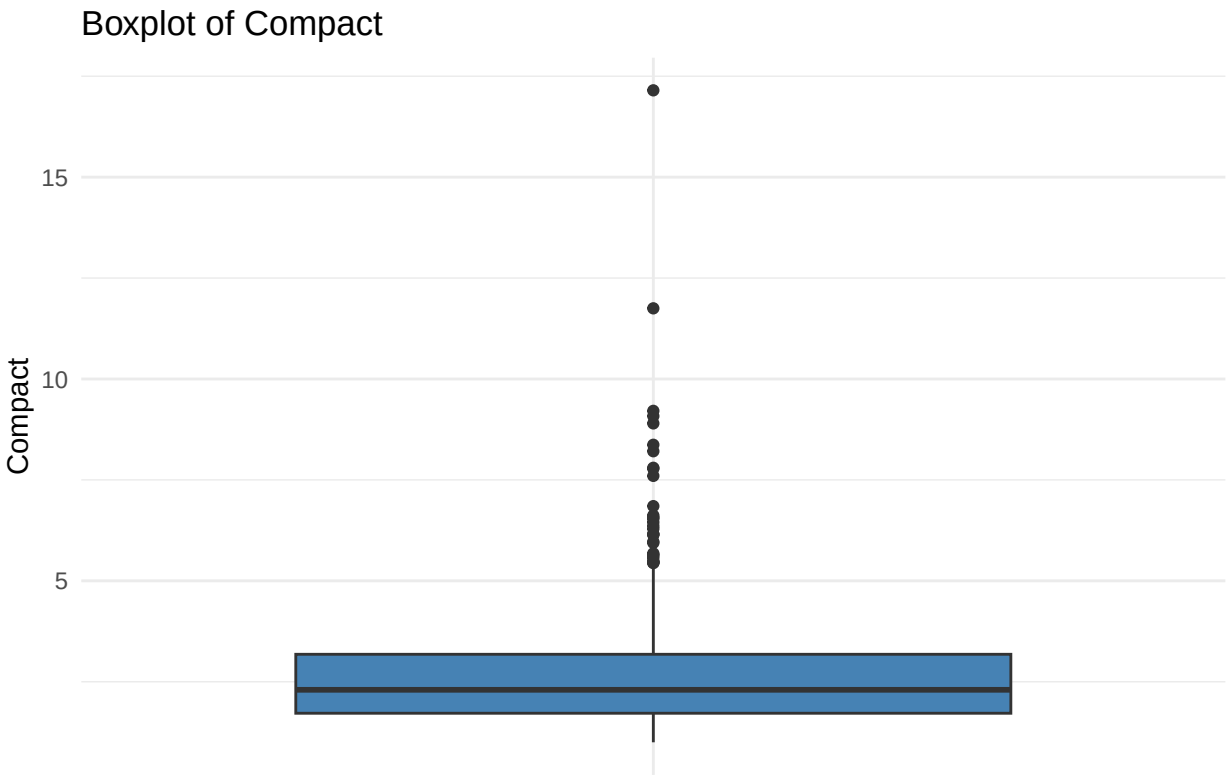

Boxplot of NDVI
NDVI
0.4
0.2
0.0
−0.2

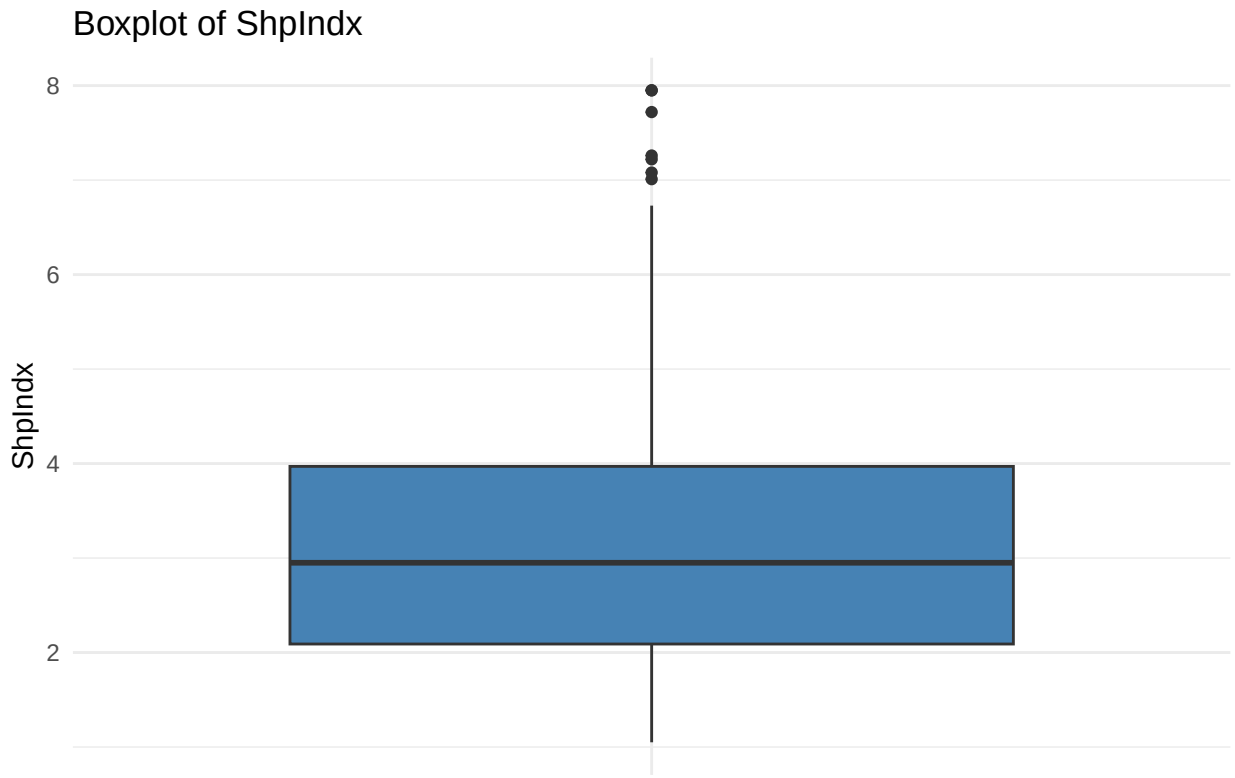

Boxplot of Area
Area
40000
30000
20000
10000
0

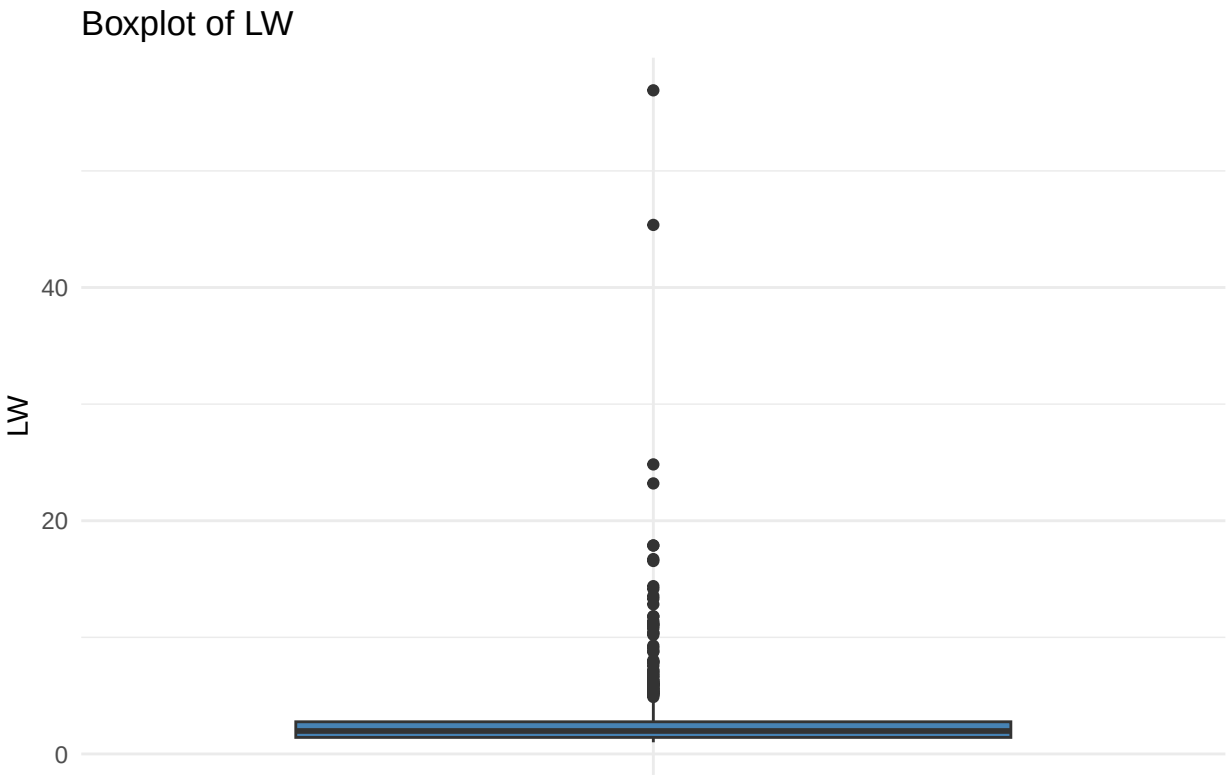

Boxplot of BordLngth
BordLngth
4000
3000
2000
1000
0

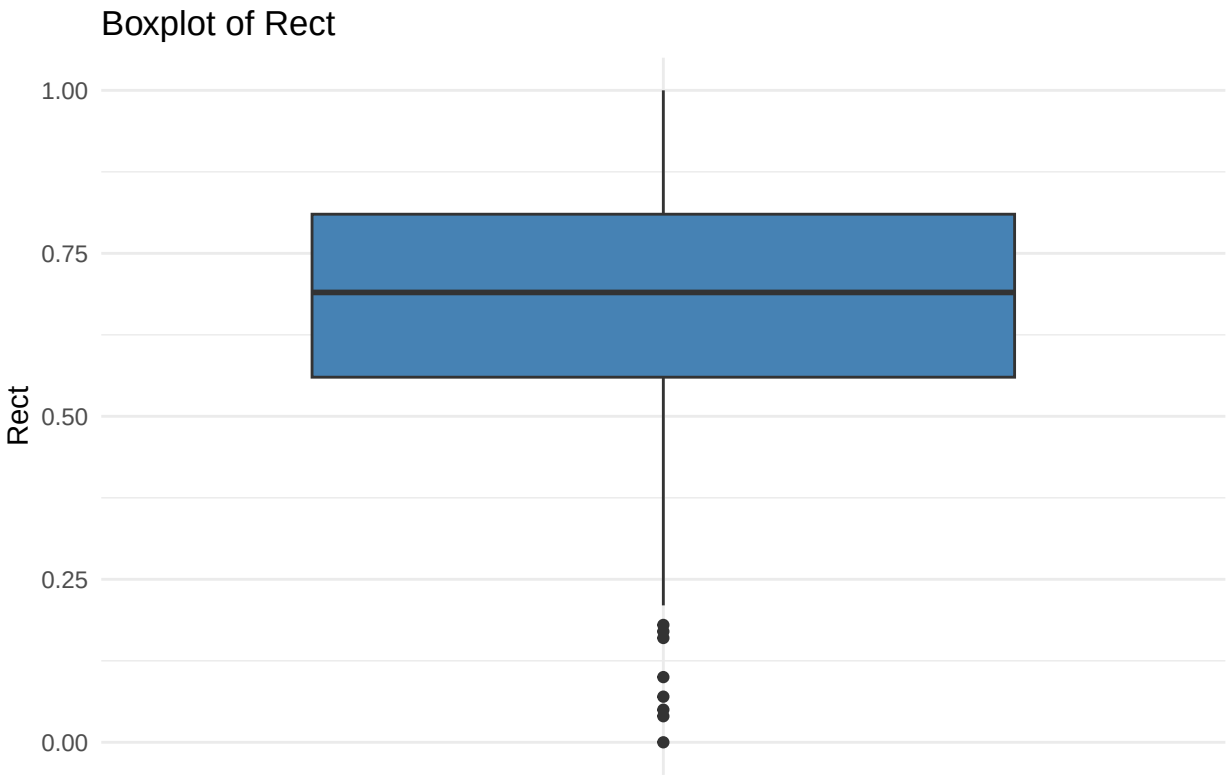

Boxplot of BrdIndx
BrdIndx
6
4
2

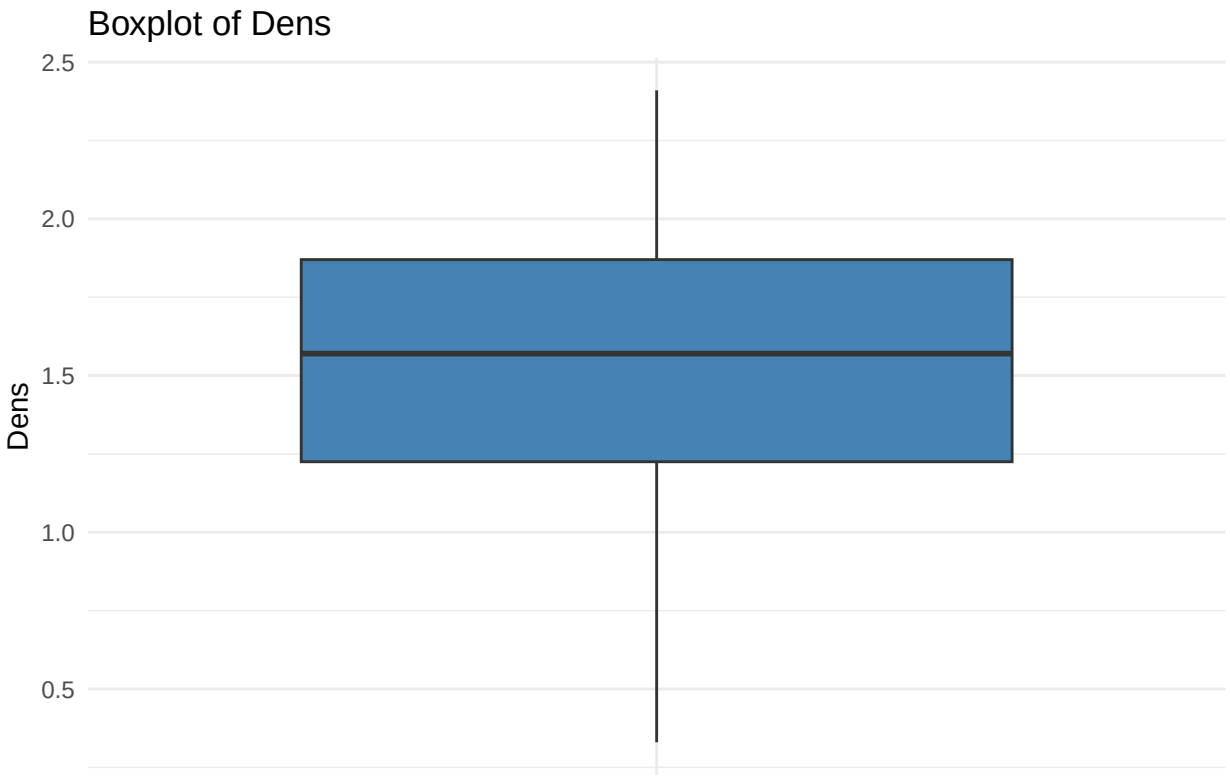

Boxplot of Round
Round
6
4
2
0

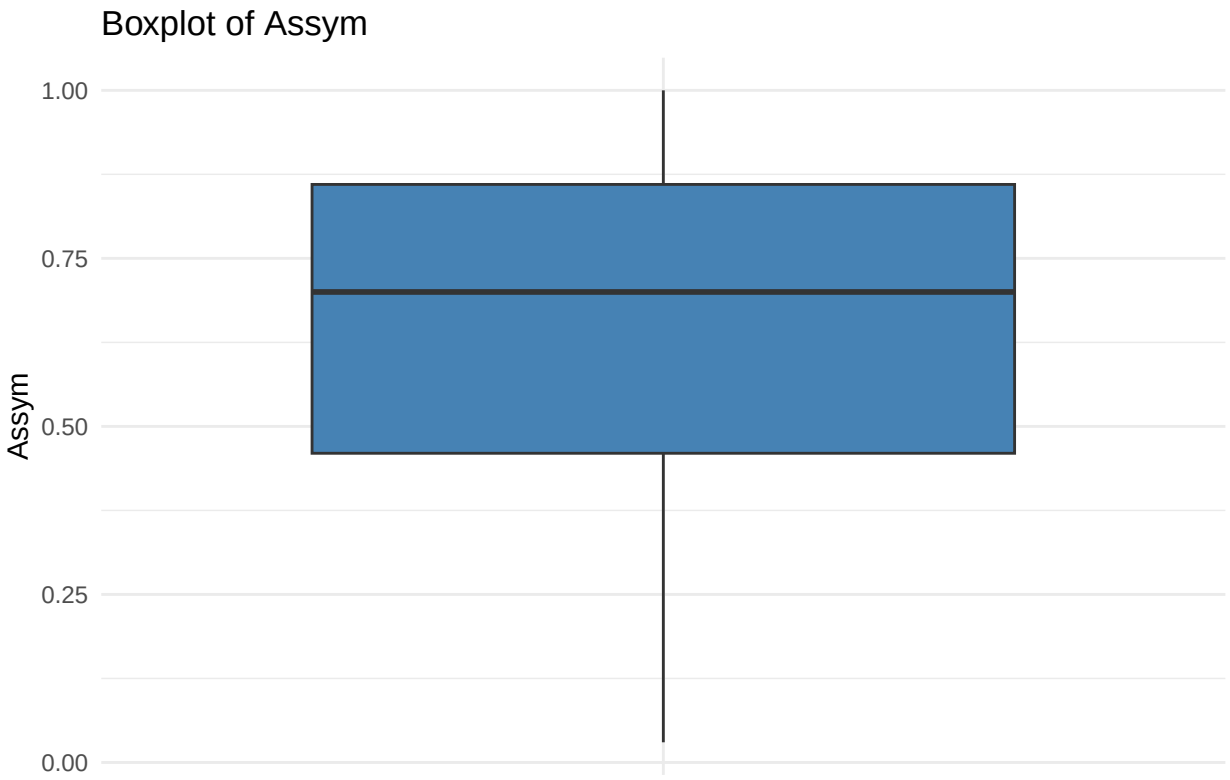

Boxplot of Compact
Compact
15
10
5

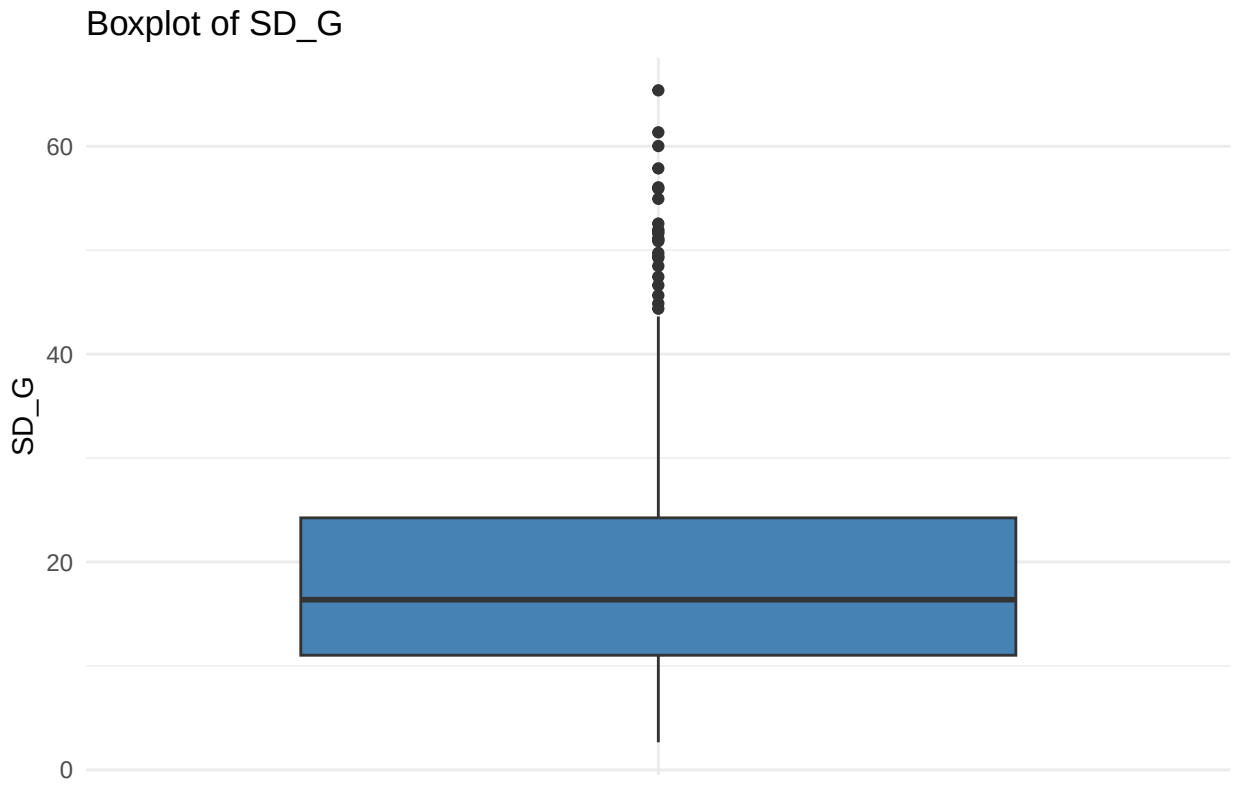

Boxplot of ShpIndx
ShpIndx
8
6
4
2

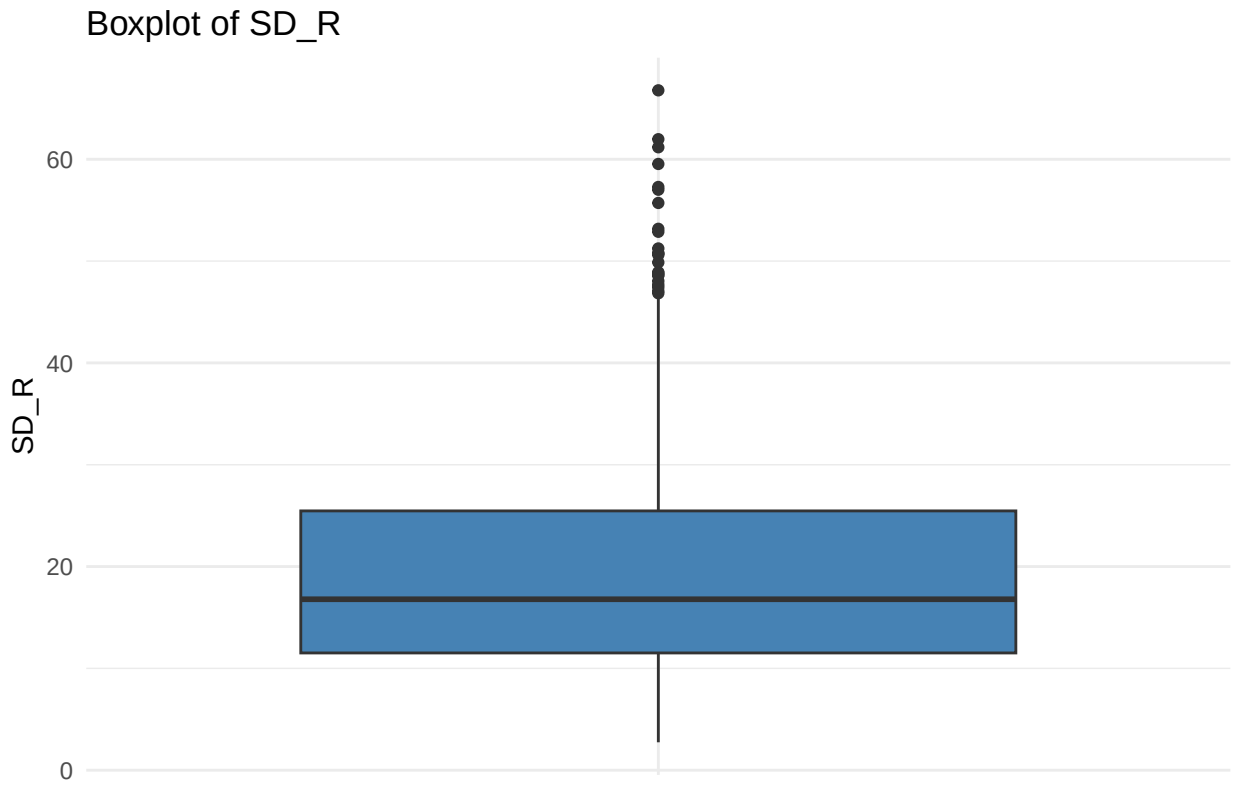

Boxplot of LW
LW
40
20
0

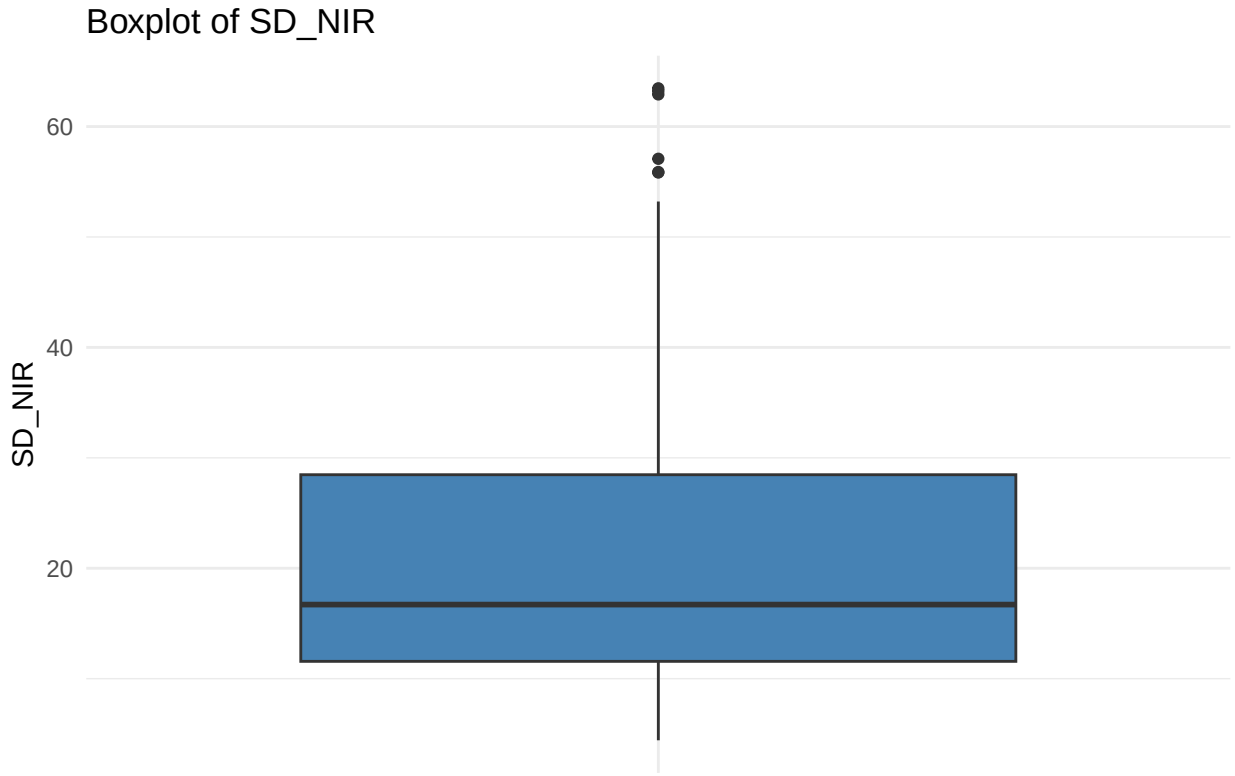

Boxplot of Rect
Rect
1.00
0.75
0.50
0.25
0.00

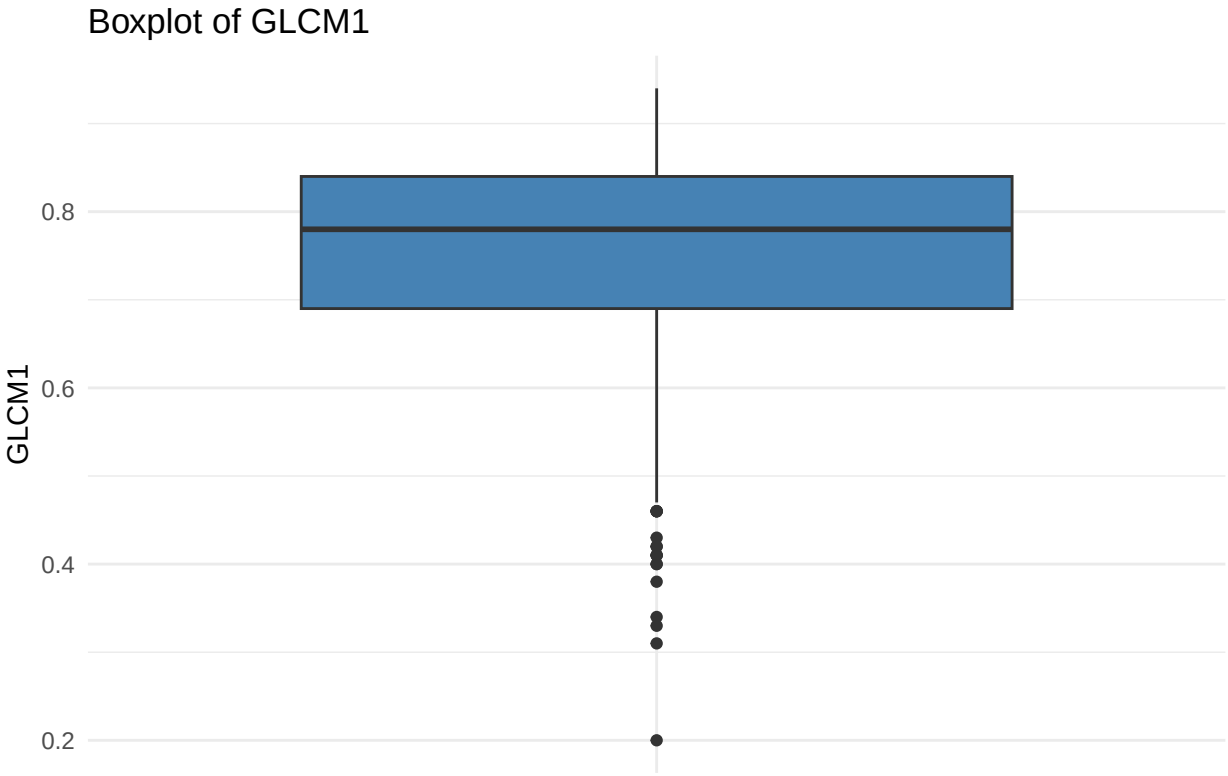

Boxplot of Dens
Dens
2.5
2.0
1.5
1.0
0.5

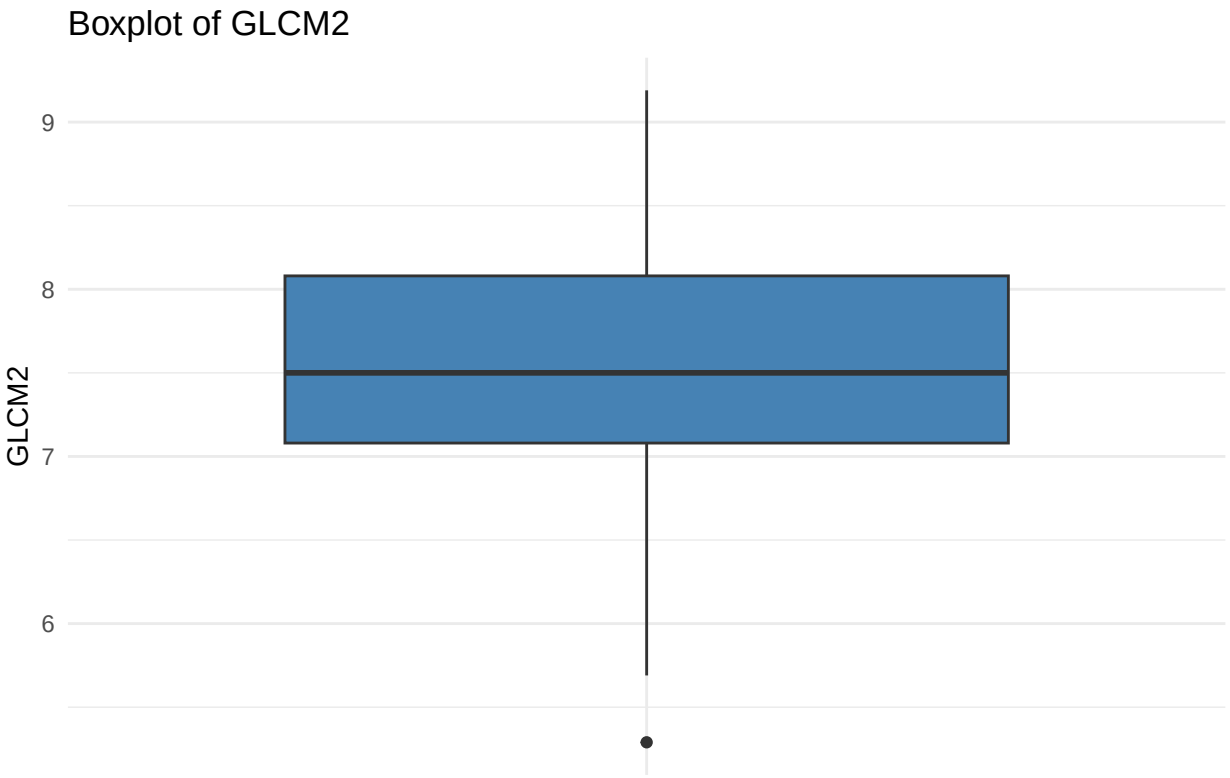

Boxplot of Assym
Assym
1.00
0.75
0.50
0.25
0.00

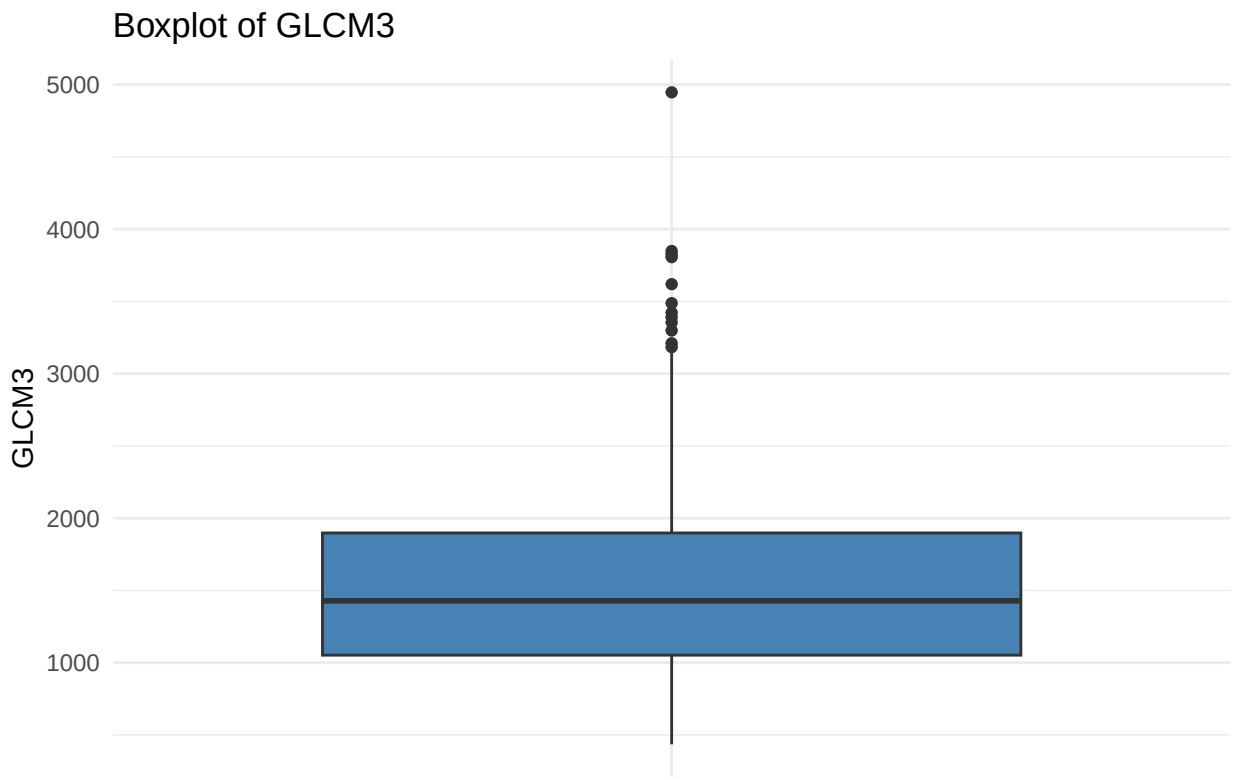

Boxplot of SD_G
SD_G
60
40
20
0


Boxplot of SD_R
SD_R
60
40
20
0

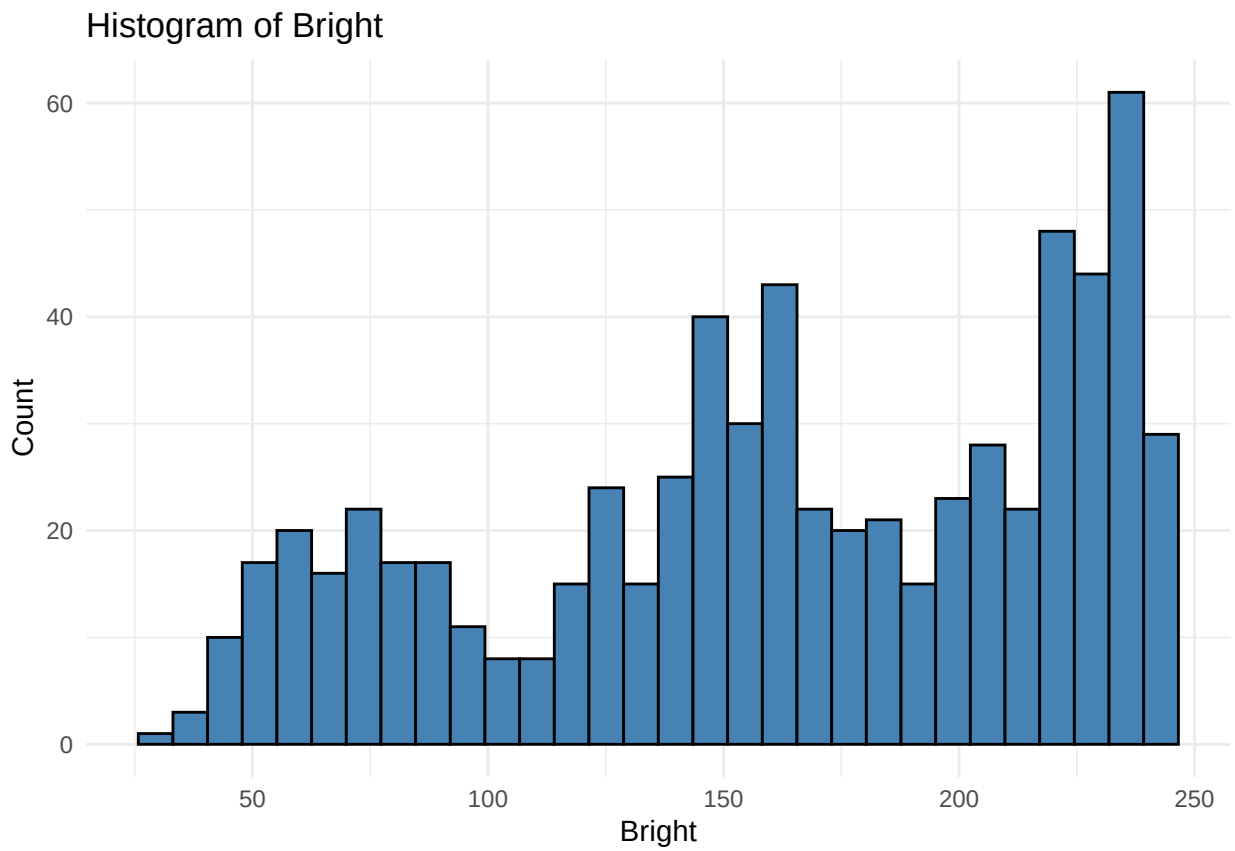

Boxplot of SD_NIR
SD_NIR
60
40
20

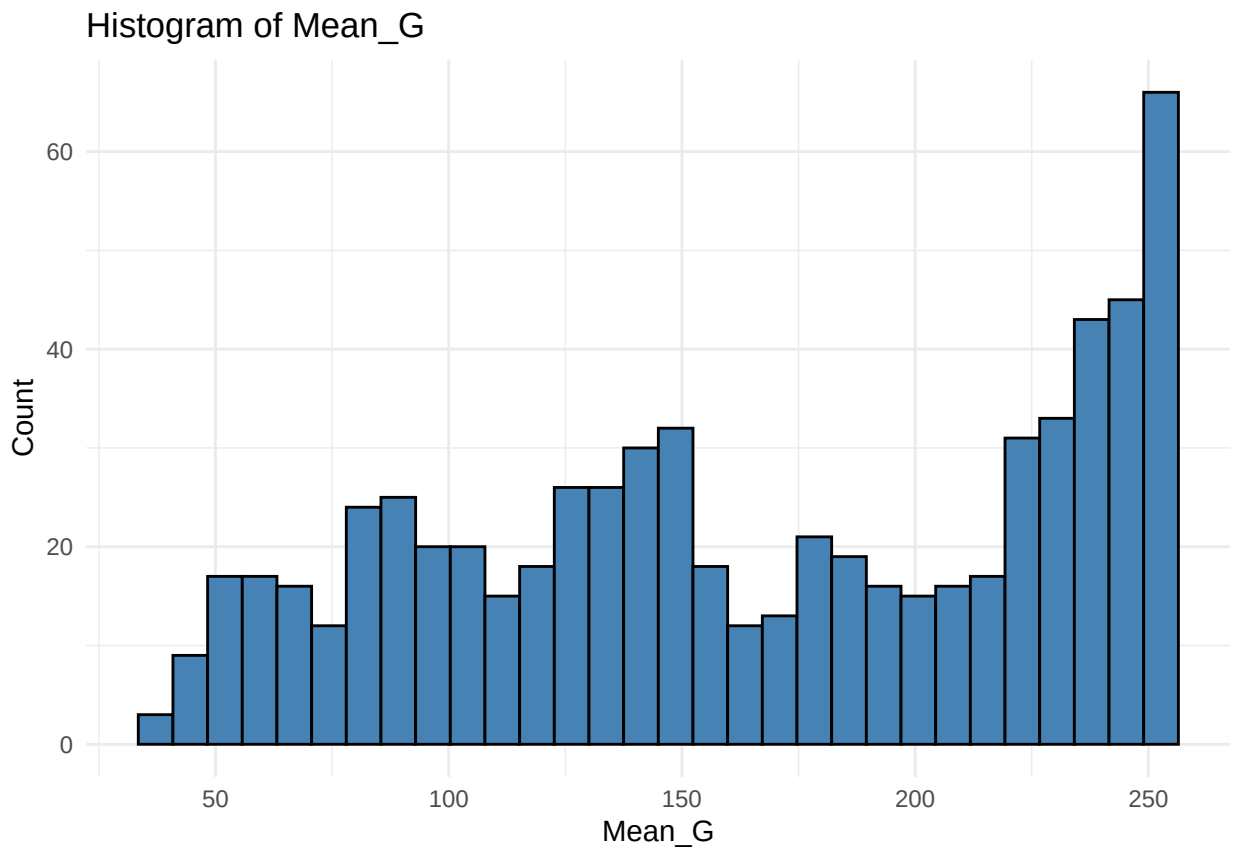

Boxplot of GLCM1
GLCM1
0.8
0.6
0.4
0.2

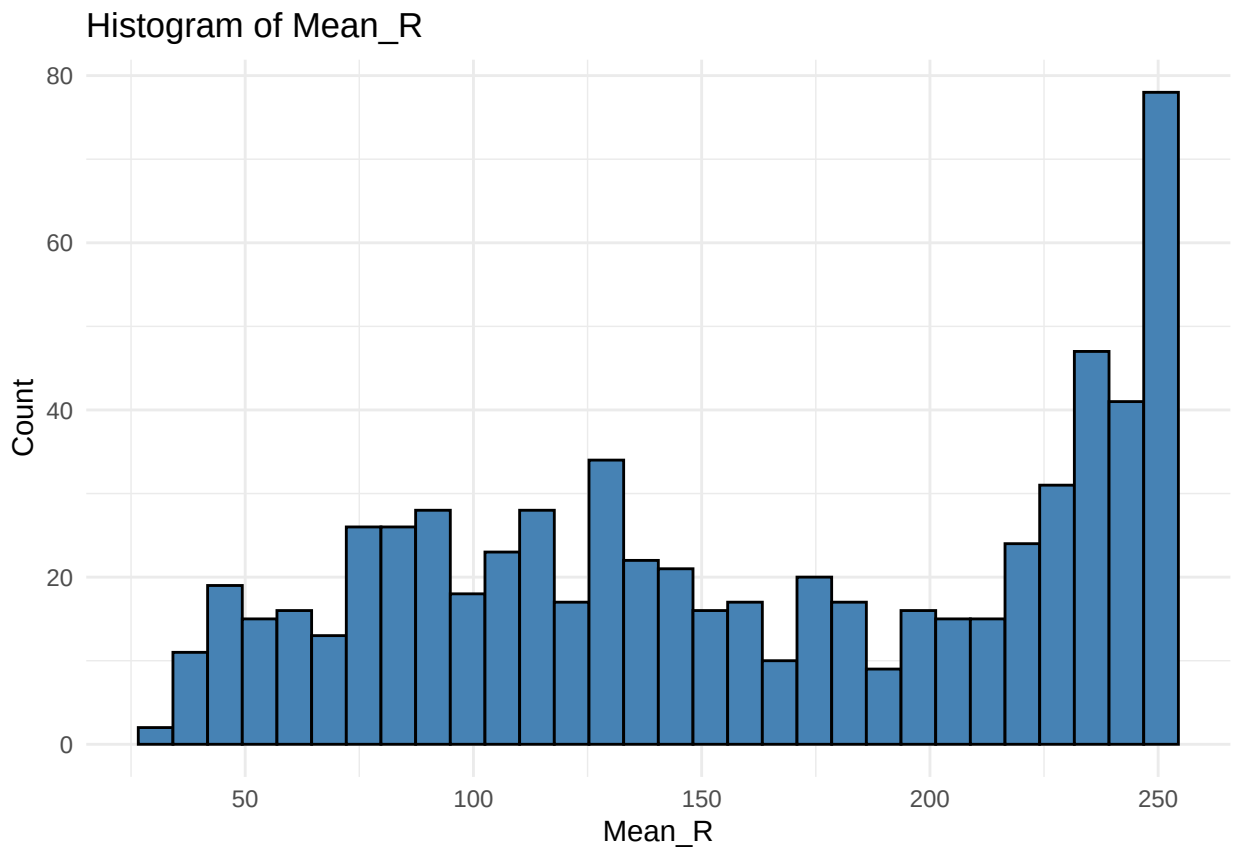

Boxplot of GLCM2
GLCM2
9
8
7
6

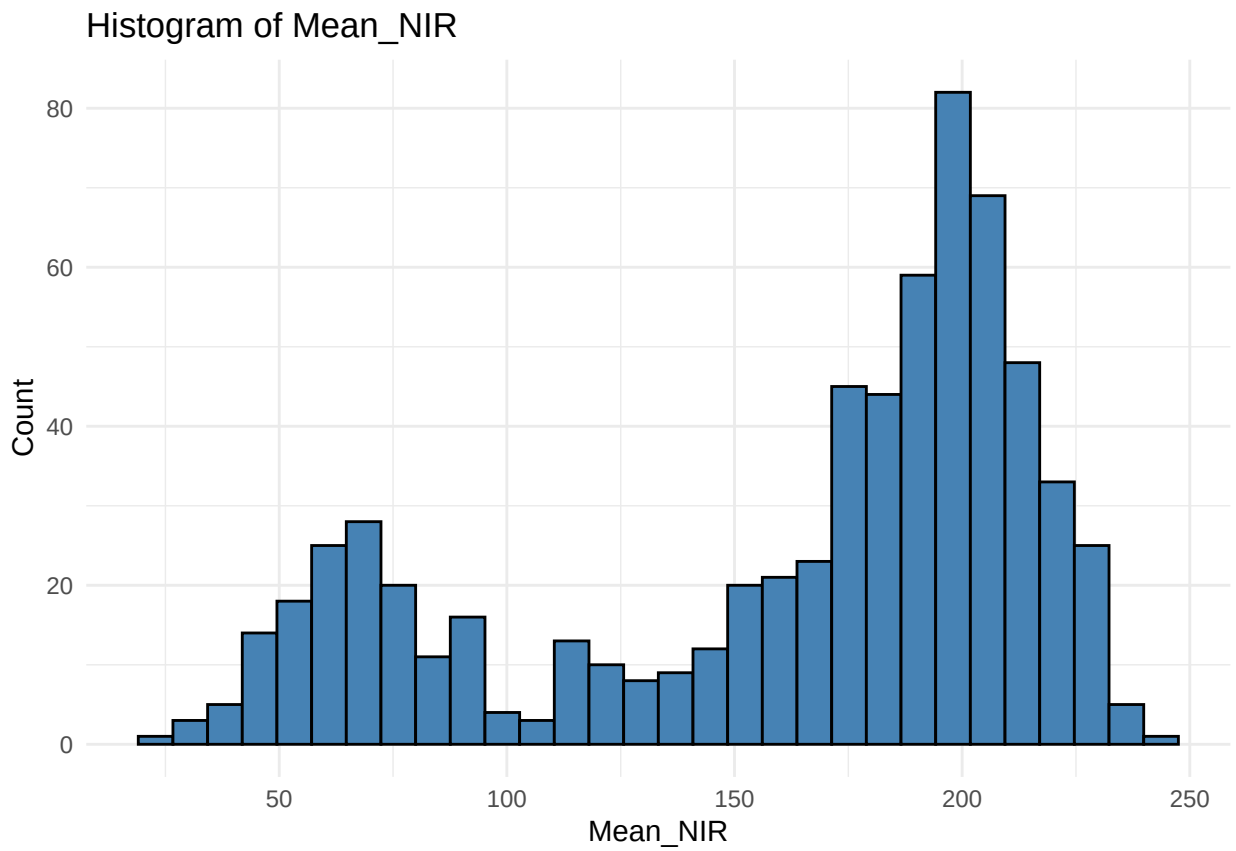


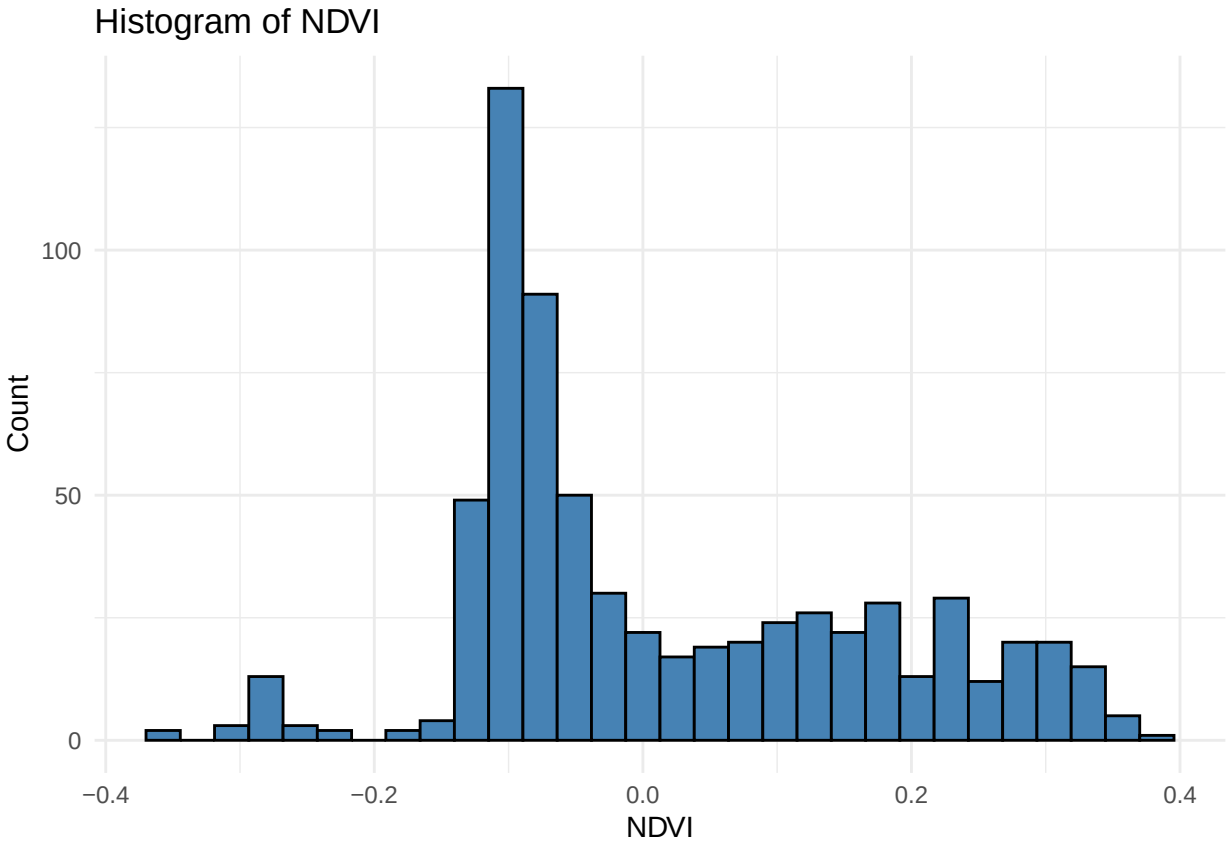

```
# Histogram / bar-style distribution plot
for (var in names(numeric_vars)) {
  print(
    ggplot(urban, aes(x = .data[[var]])) +
      geom_histogram(fill = "steelblue", bins = 30, color = "black") +
      labs(
        title = paste("Histogram of", var),
        x = var,
        y = "Count"
      ) +
      theme_minimal()
  )
}
```

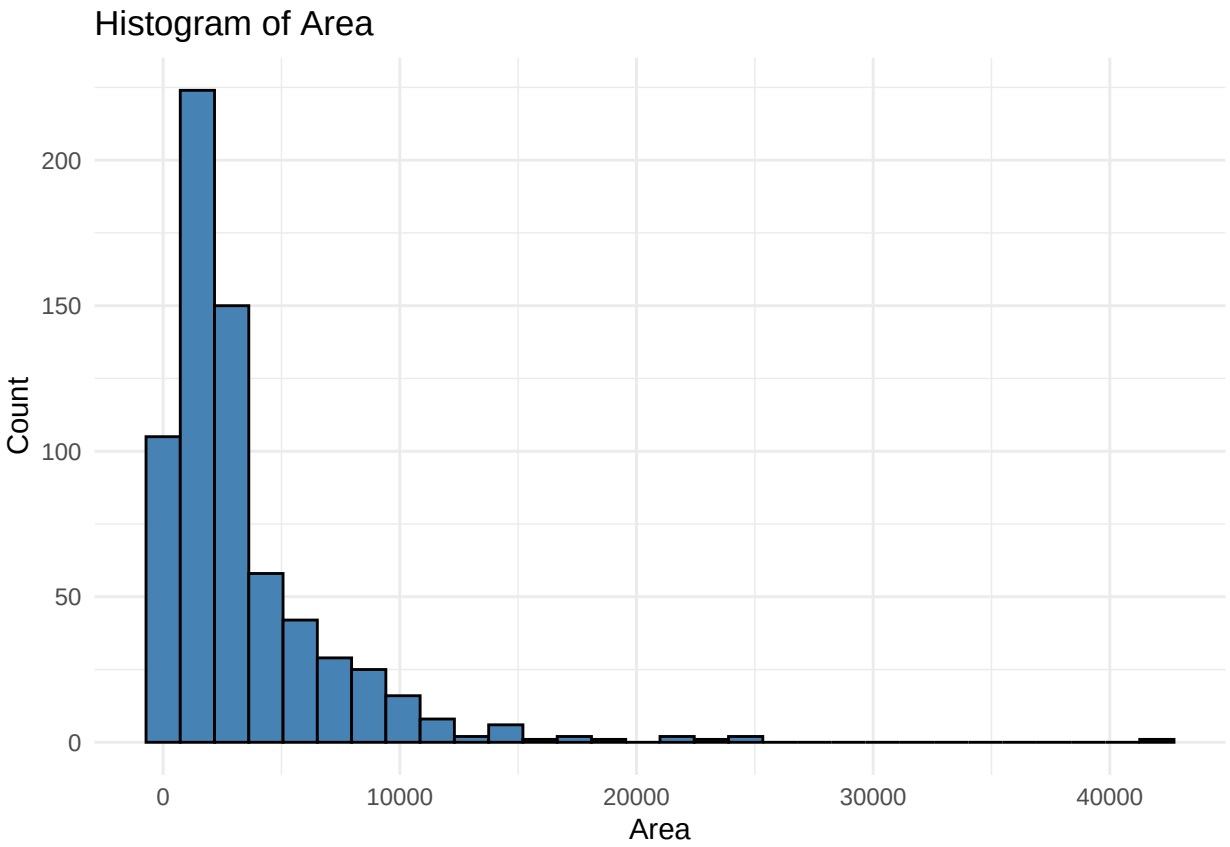

Histogram of Bright
Count
60
40
20
0
50
100
150
200
250
Bright

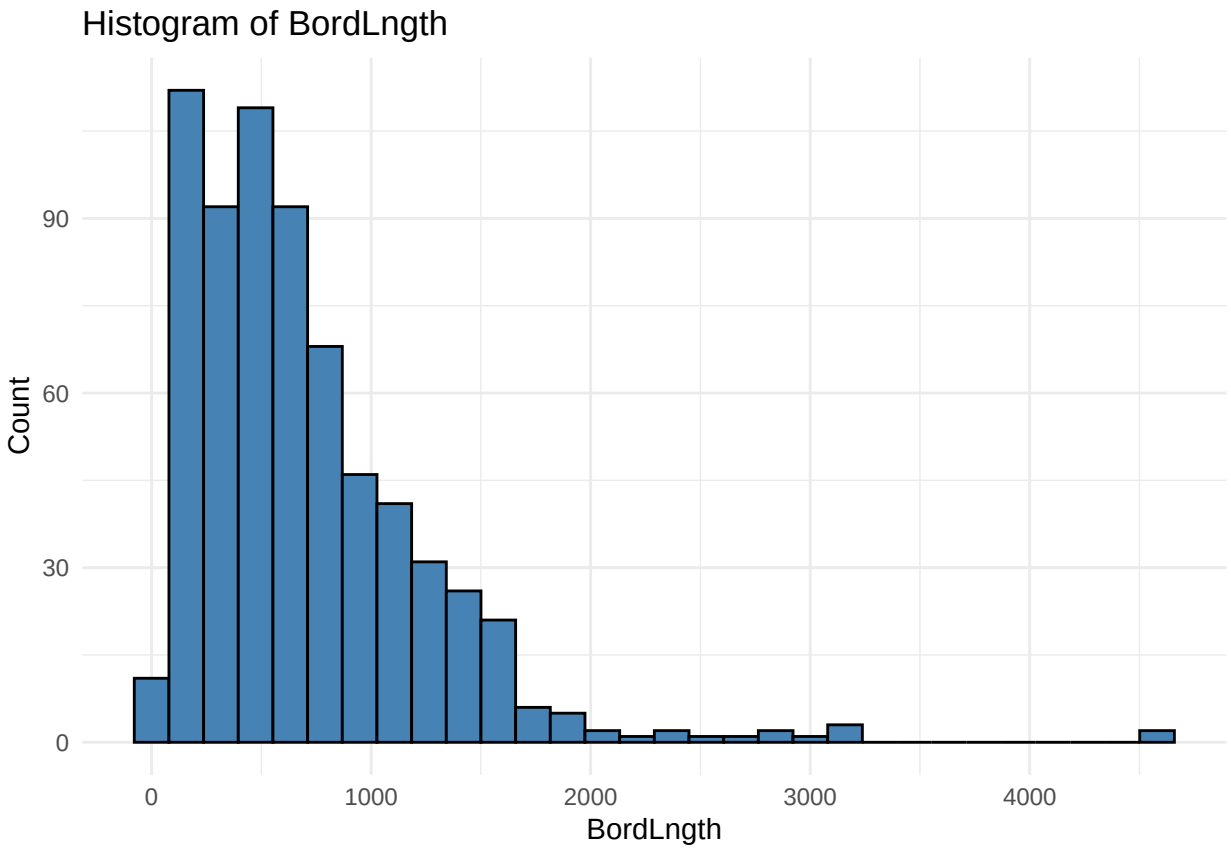

Histogram of Mean_G
Count
60
40
20
0
50
100
150
200
250
Mean_G

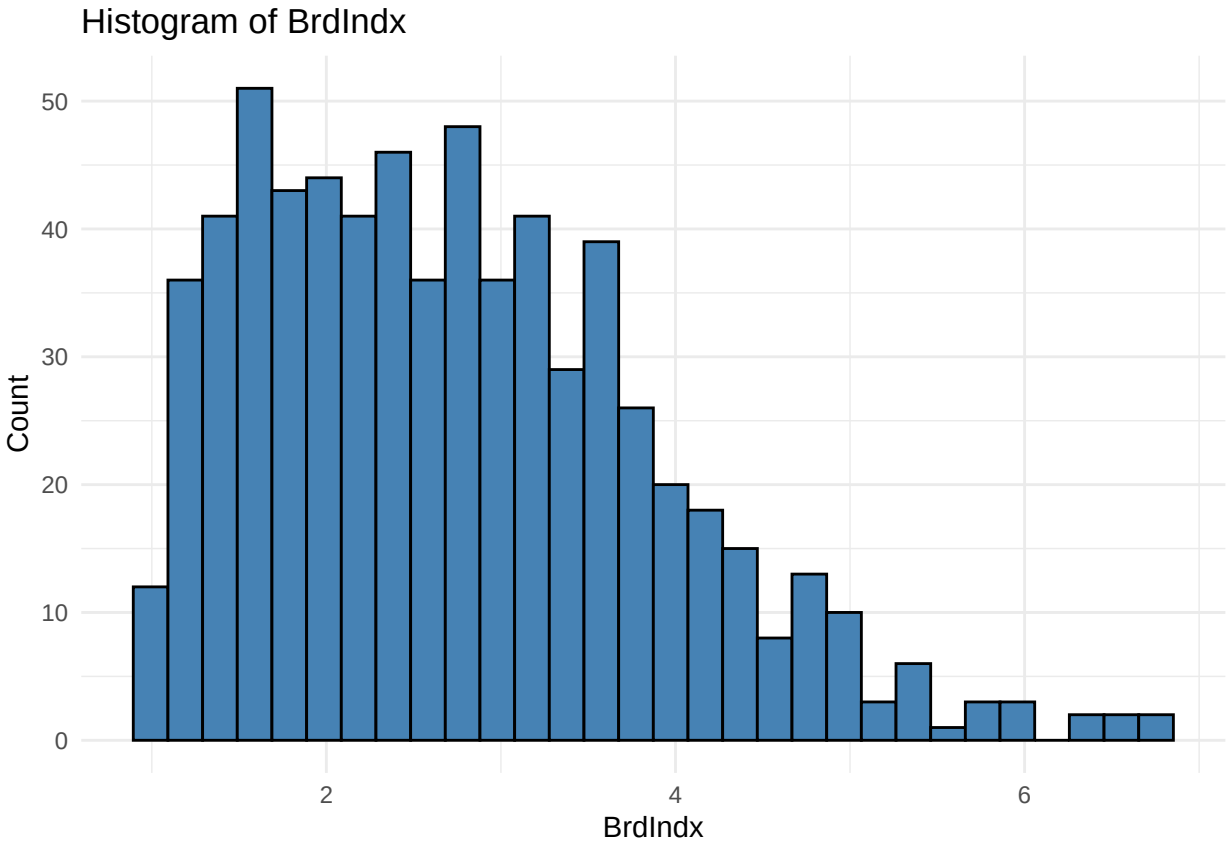

Histogram of Mean_R
Count
Mean_R
0
20
40
60
80
50
100
150
200
250

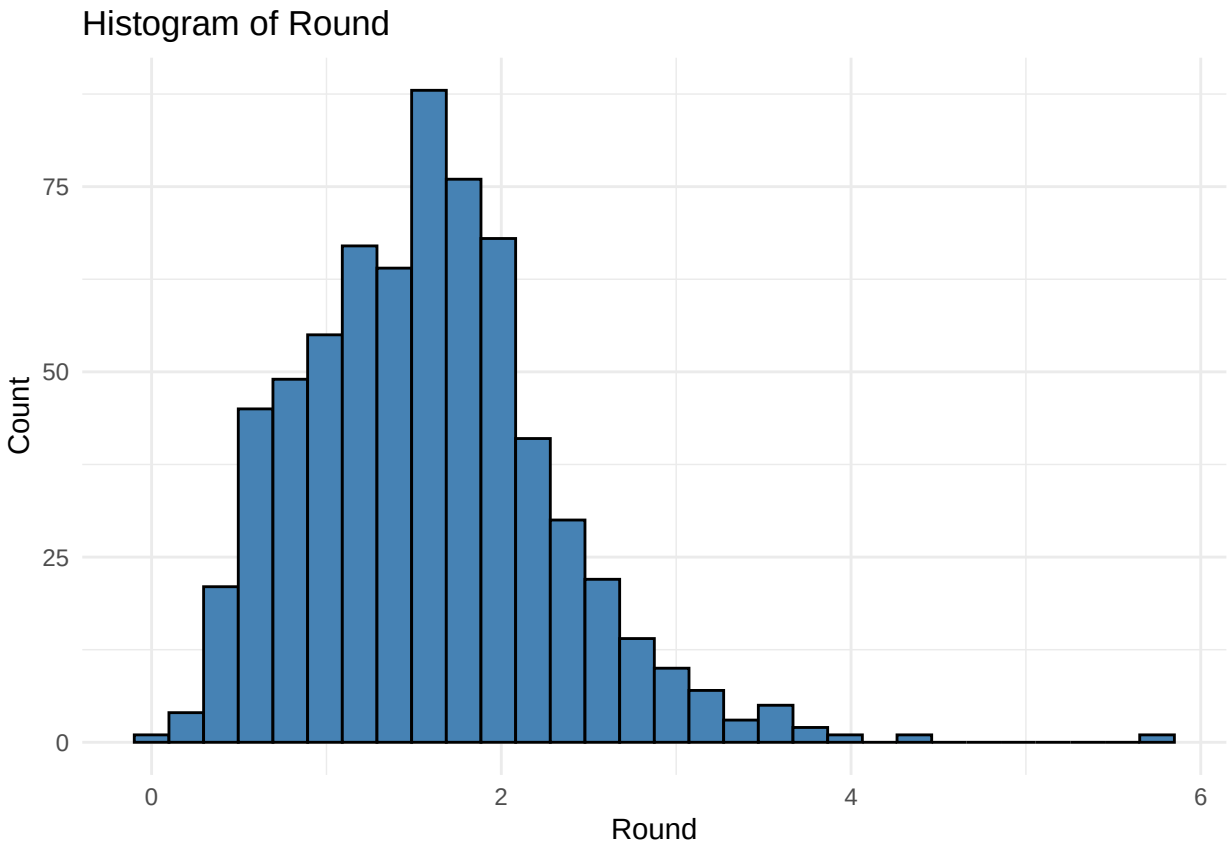

Histogram of Mean_NIR
Count
Mean_NIR
0
20
40
60
80
50
100
150
200
250

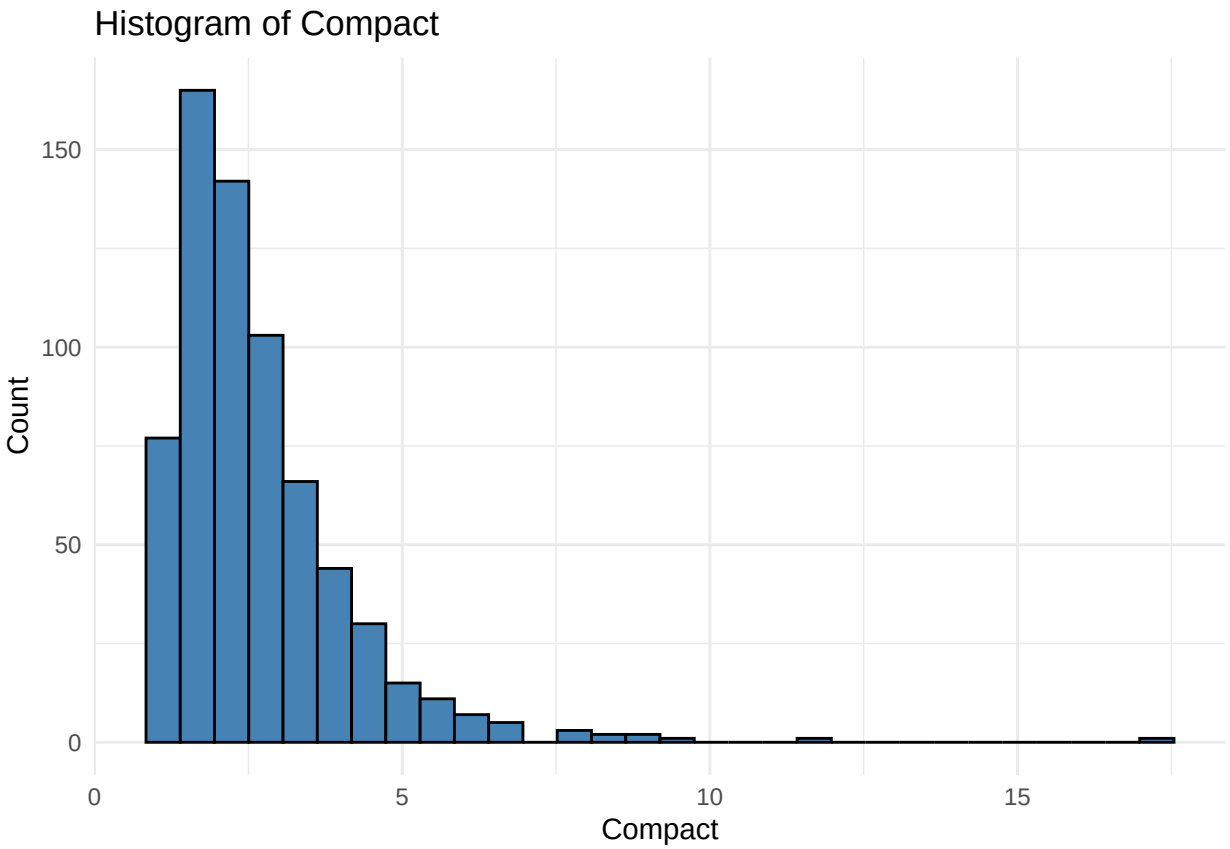

Histogram of NDVI
Count
100
50
0
−0.4
−0.2
0.0
0.2
0.4
NDVI

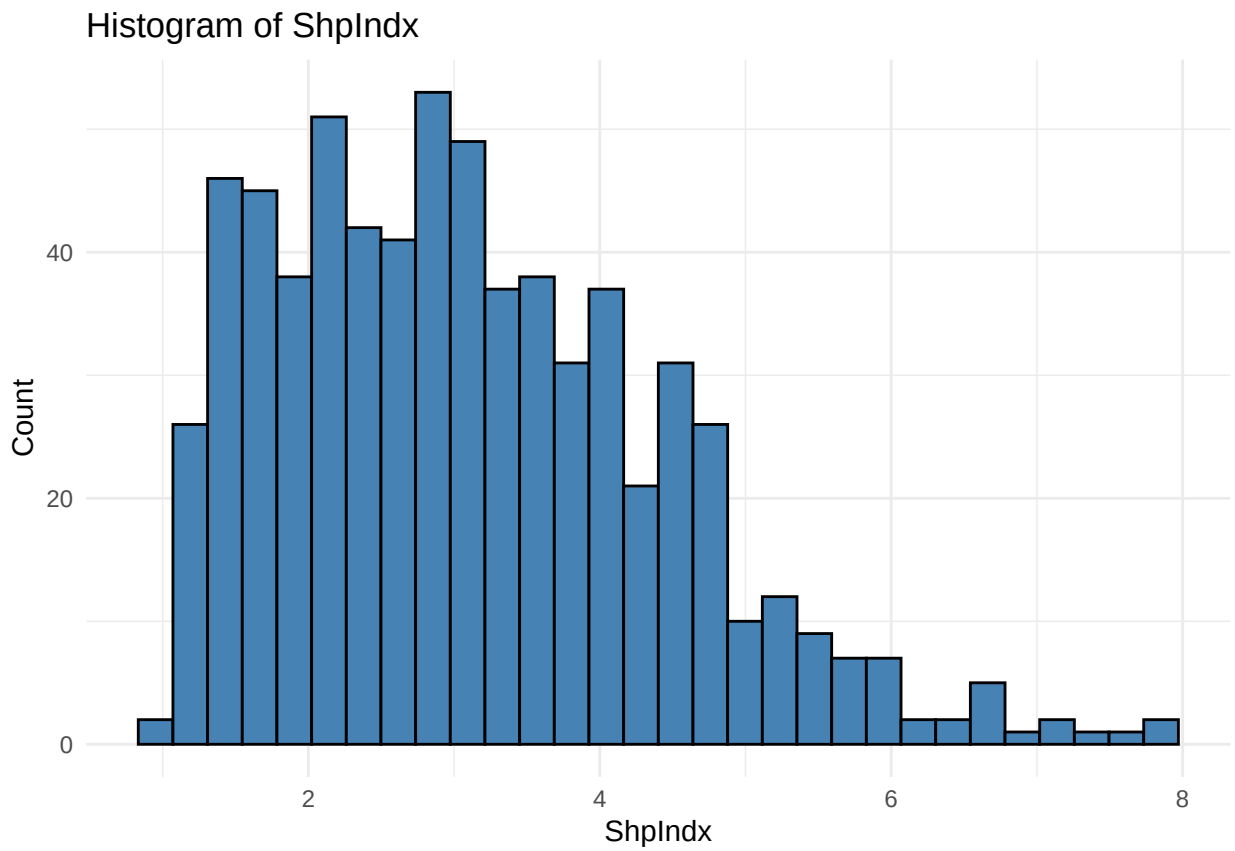

Histogram of Area
Count
200
150
100
50
0
0
10000
20000
30000
40000
Area

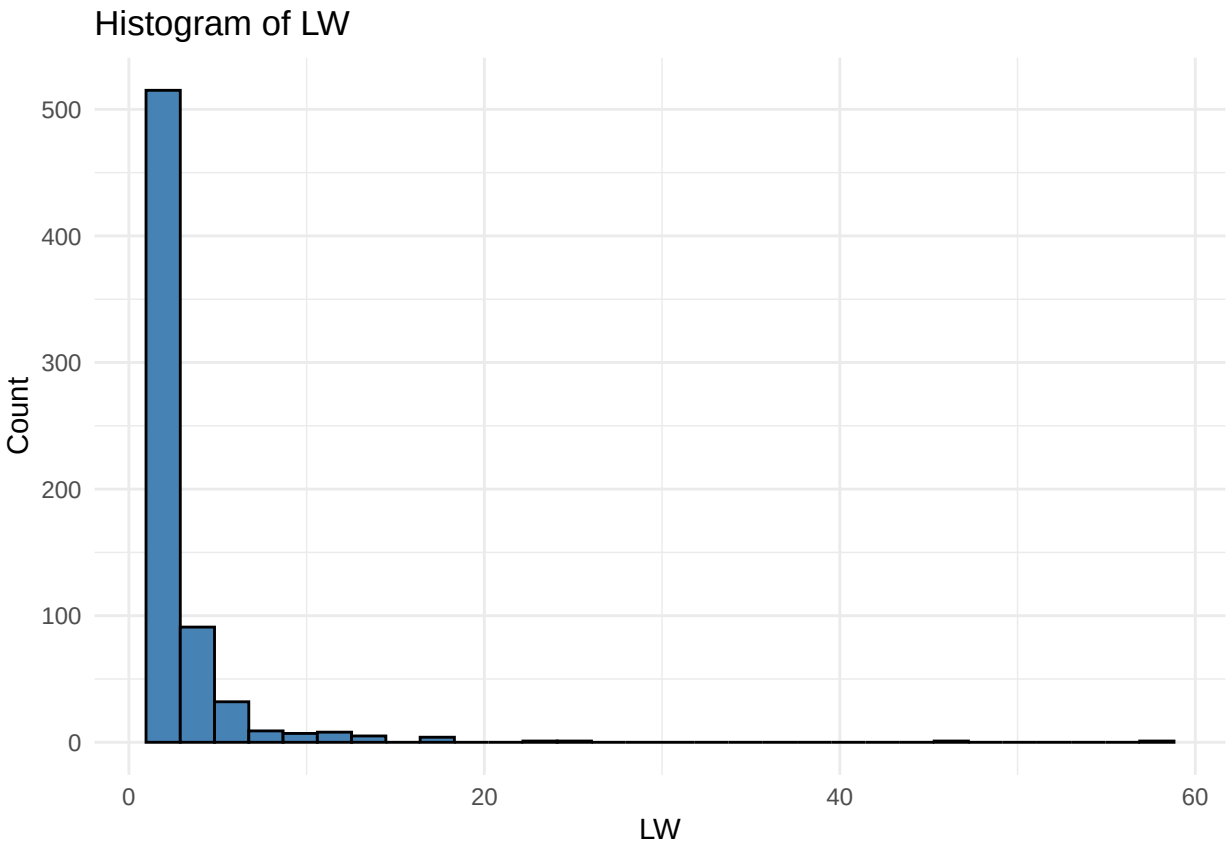

Histogram of BordLngth
Count
90
60
30
0
0
1000
2000
3000
4000
BordLngth

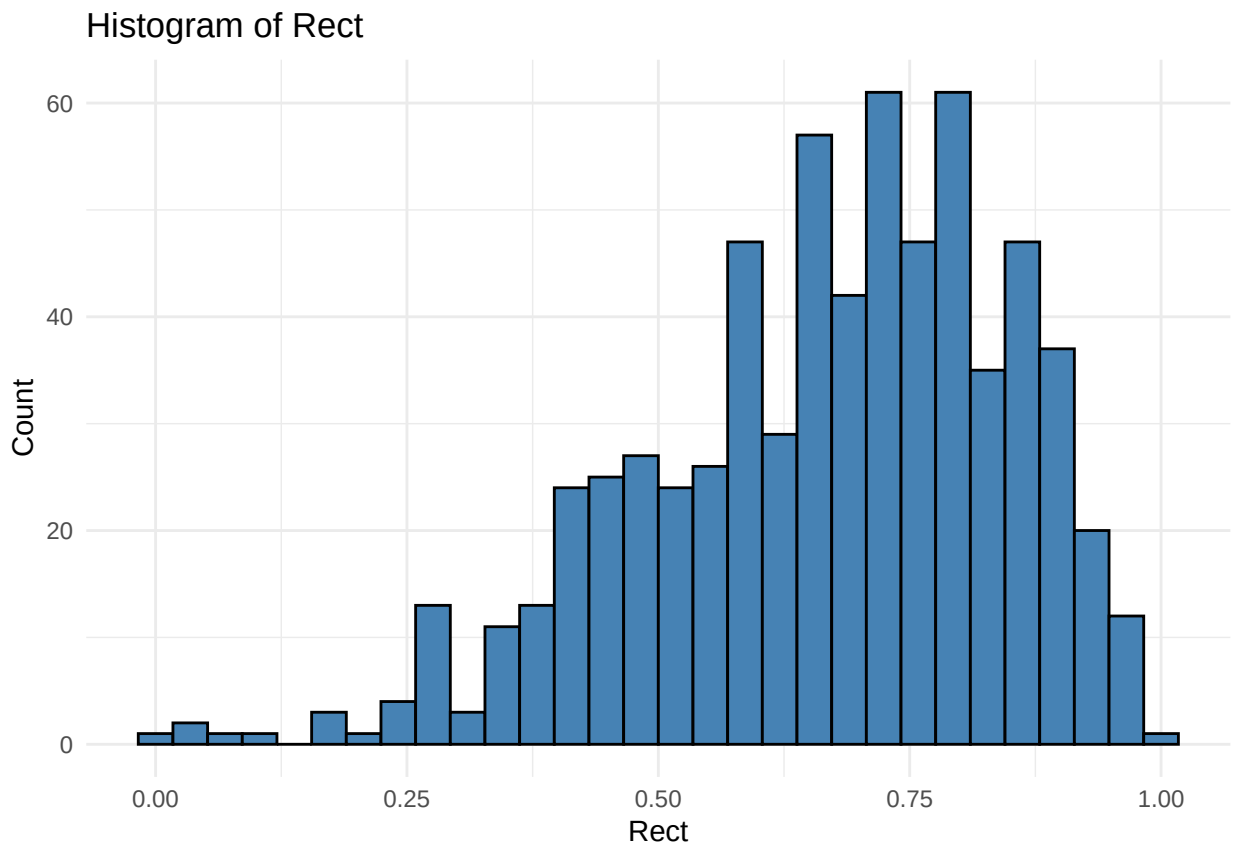

Histogram of BrdIndx
Count
BrdIndx
0
10
20
30
40
50
2
4
6

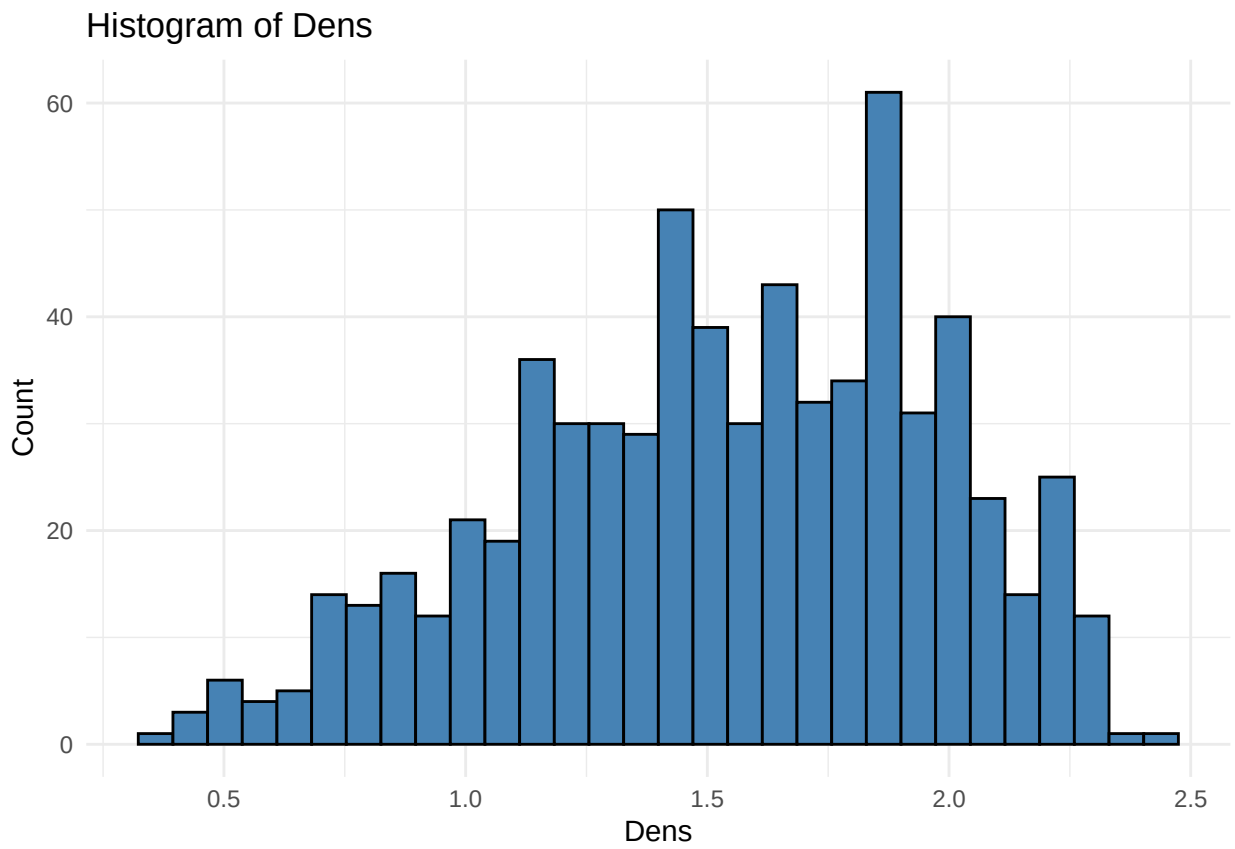

Histogram of Round
Count
75
50
25
0
0
2
4
6
Round

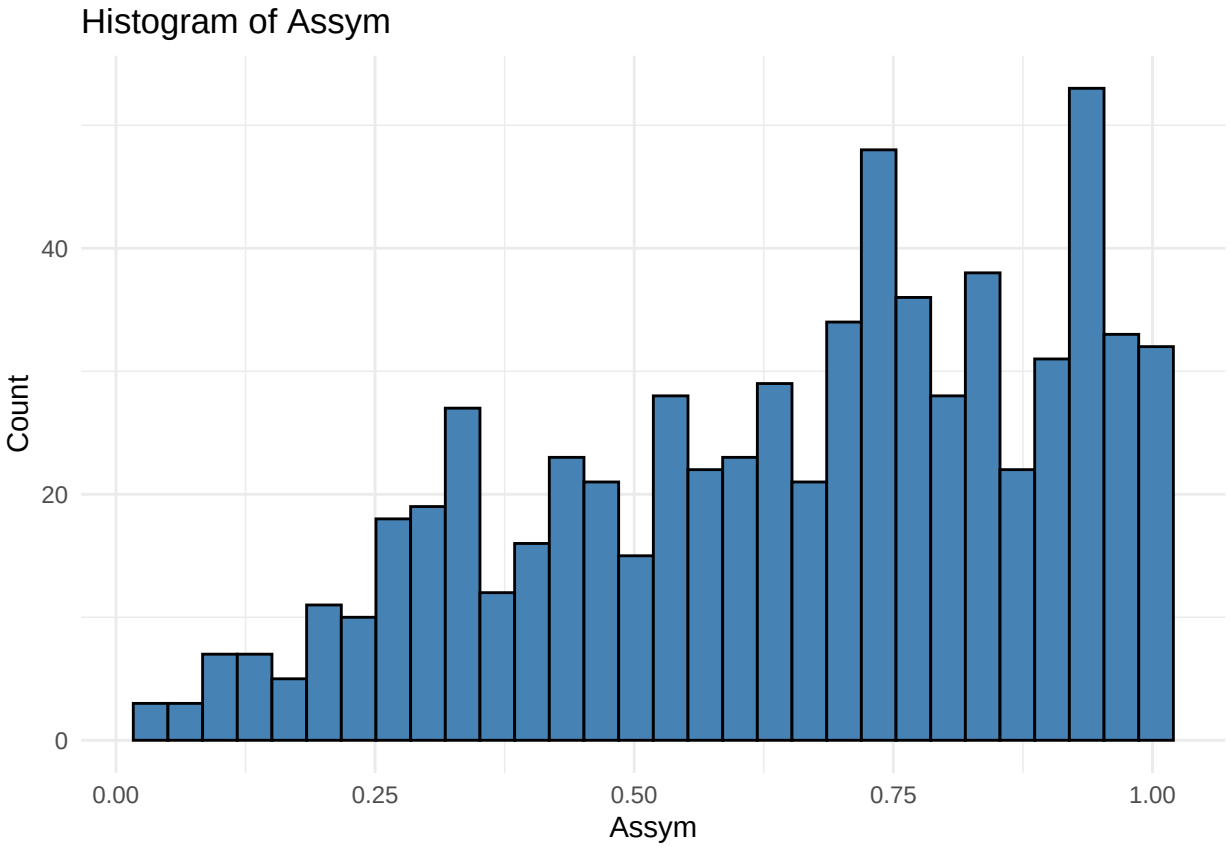

Histogram of Compact
Count
150
100
50
0
0
5
10
15
Compact

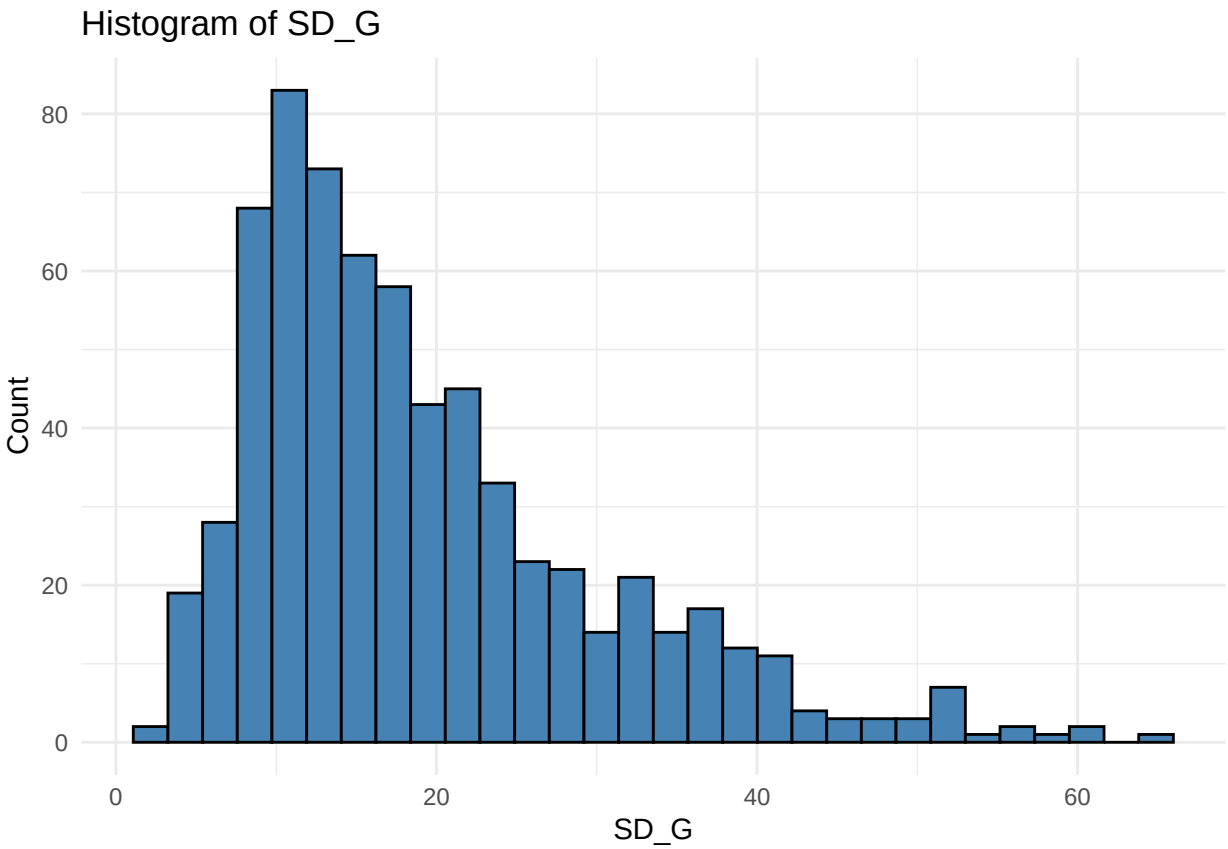

Histogram of ShpIndx
Count
0
20
40
2
4
6
8
ShpIndx

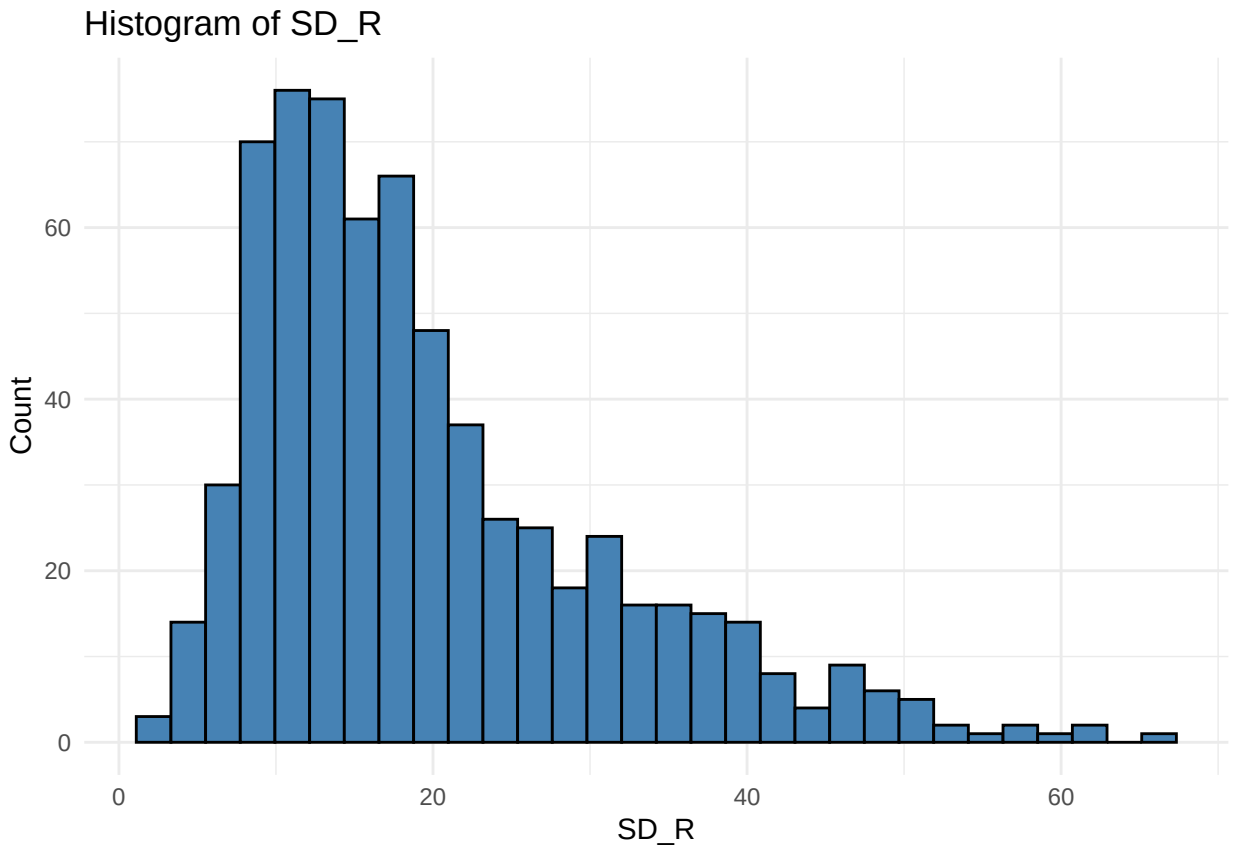

Histogram of LW
Count
500
400
300
200
100
0
0
20
40
60
LW

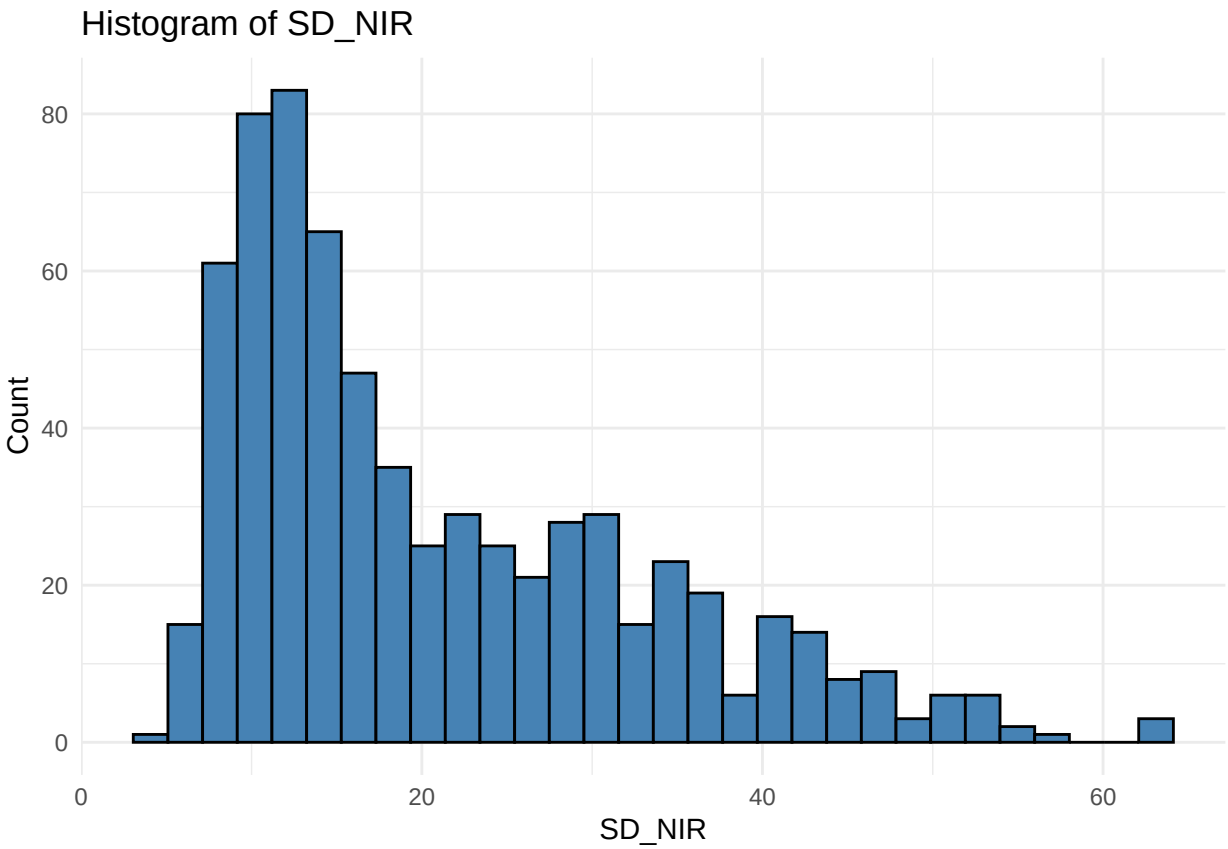

Histogram of Rect
Count
60
40
20
0
0.00
0.25
0.50
0.75
1.00
Rect

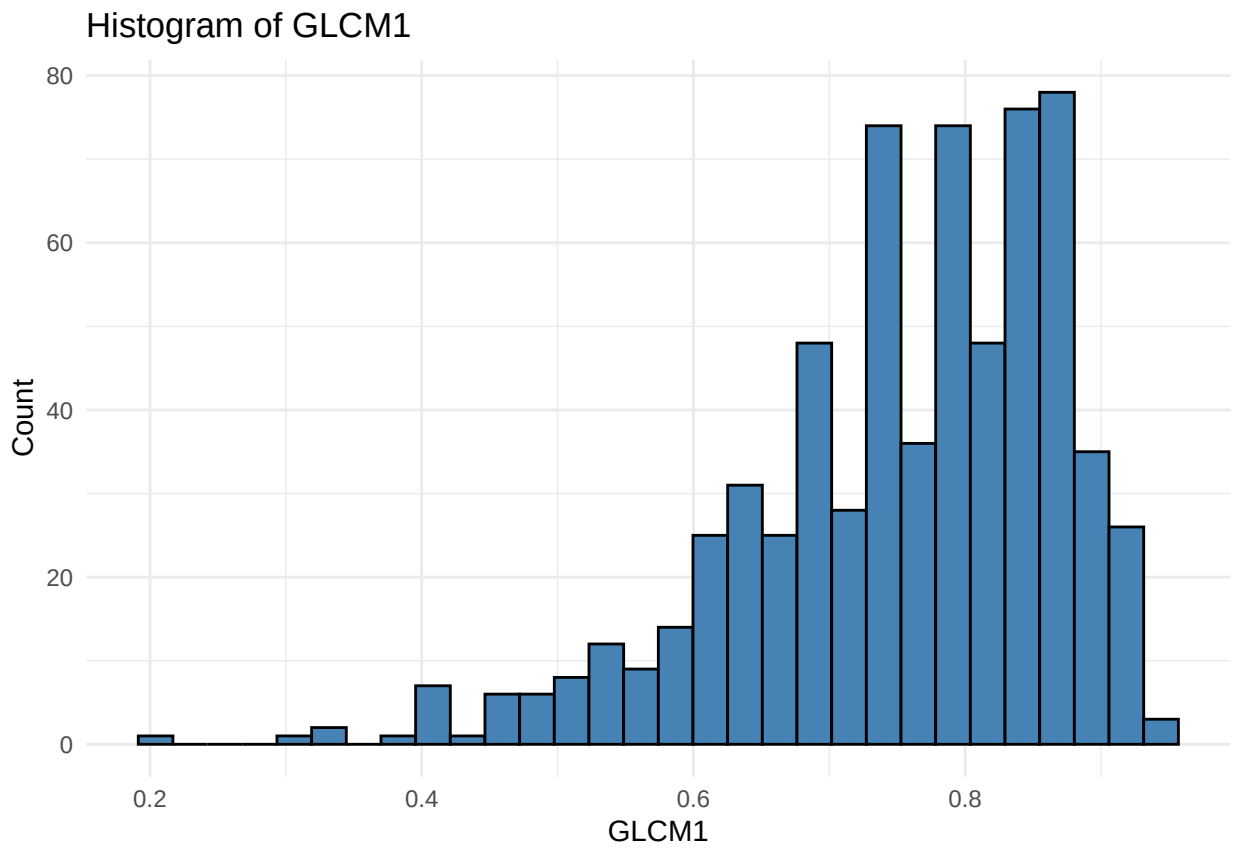

Histogram of Dens
Count
60
40
20
0
0.5
1.0
1.5
2.0
2.5
Dens

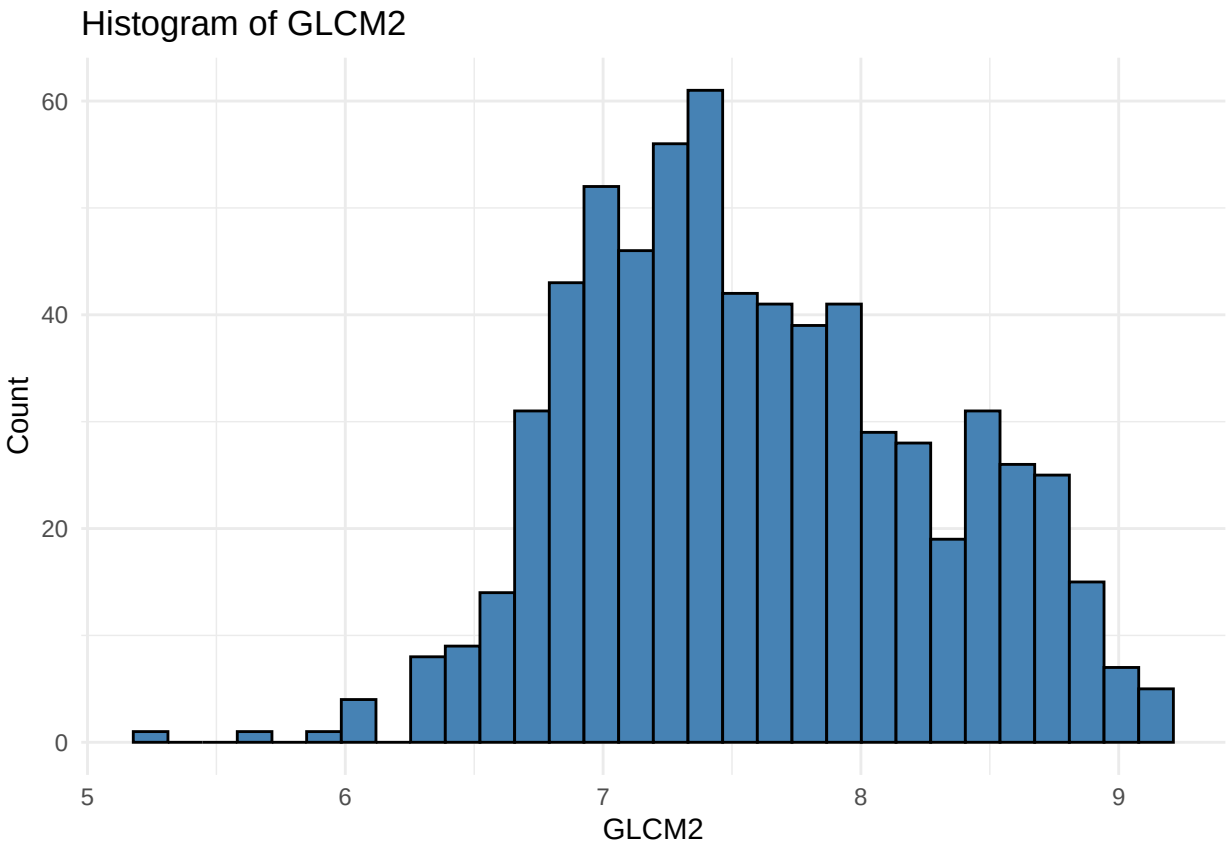

Histogram of Assym
Count
40
20
0
0.00
0.25
0.50
0.75
1.00
Assym

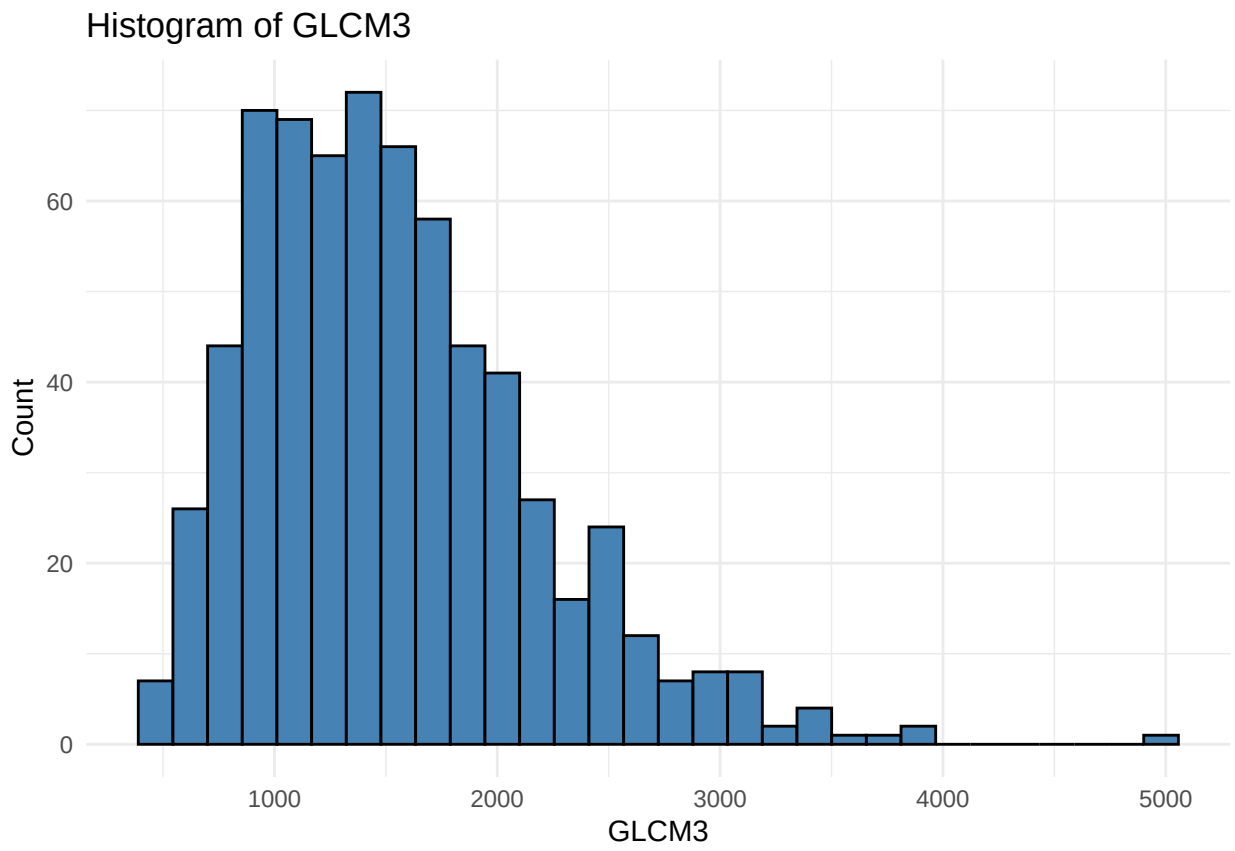

Histogram of SD_G
Count
SD_G
0
20
40
60
80
0
20
40
60


Histogram of SD_R
Count
0
20
40
60
0
20
40
60
SD_R


Histogram of SD_NIR
Count
SD_NIR
0
20
40
60
0
20
40
60
80


Histogram of GLCM1
Count
80
60
40
20
0
0.2
0.4
0.6
0.8
GLCM1


Histogram of GLCM2
Count
60
40
20
0
5
6
7
8
9
GLCM2

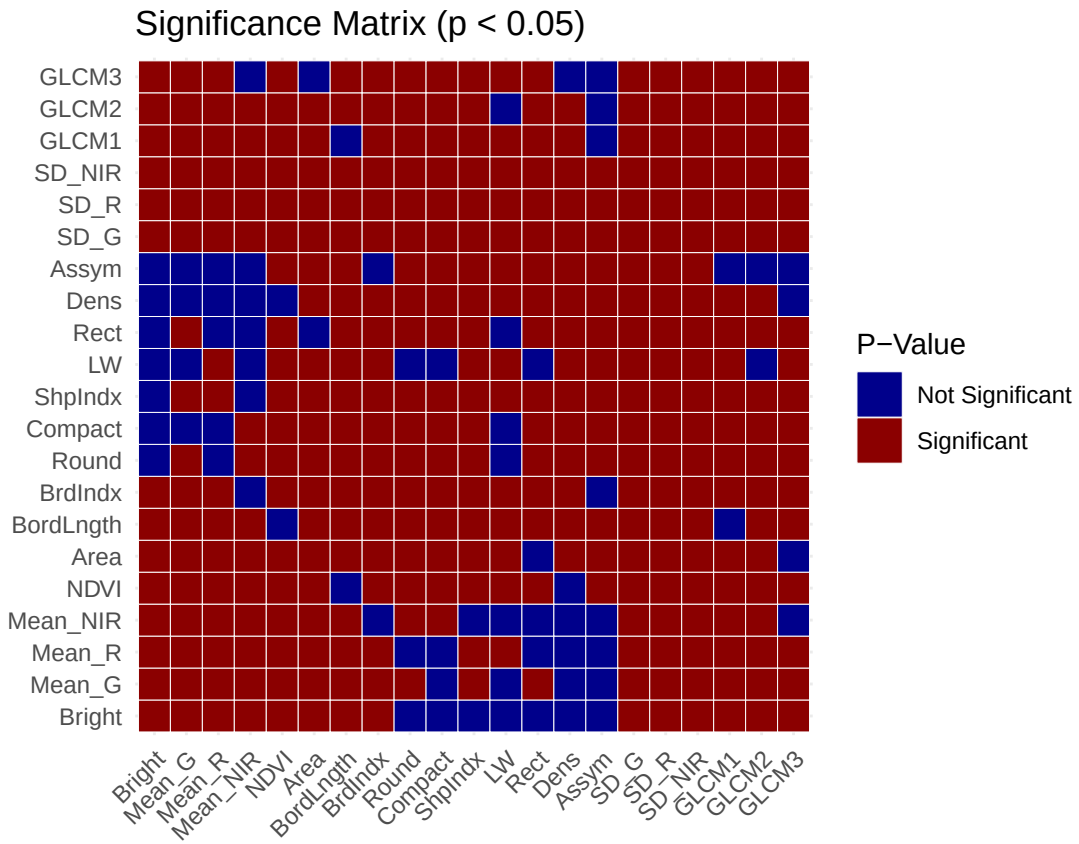


Correlation Significance

```
library(ggplot2)

#Compute p-value matrix
cor_test_results <- cor.mtest(urban[, -1], conf.level = 0.95)
p_values <- cor_test_results$p

#Convert p-values to a binary factor
sig_matrix <- ifelse(p_values < 0.05, "Significant", "Not Significant")

#Convert to long for ggplot
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
sig_df <- as.data.frame(as.table(sig_matrix))
colnames(sig_df) <- c("Var1", "Var2", "Significance")

#Plot with the bins
ggplot(sig_df, aes(x = Var1, y = Var2, fill = Significance)) +
  geom_tile(color = "white") +
  scale_fill_manual(
    values = c("Significant" = "darkred", "Not Significant" = "darkblue")
  ) +
  coord_fixed() +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.title = element_blank()
  ) +
  labs(
    title = "Significance Matrix (p < 0.05)",
    fill = "P-Value"
  )
```




```r
# # Convert p-value matrix to long format
# pval_df <- as.data.frame(as.table(p_values))
# colnames(pval_df) <- c("Var1", "Var2", "p_value")
#
# ggplot(pval_df, aes(x = Var1, y = Var2, fill = p_value)) +
```

```
#   geom_tile() +
#   scale_fill_gradientn(
#     colours = c("darkred", "white", "darkblue"),
#     limits = c(0, 1),                 # <--- explicitly 0 to 1
#     name = "p-value"
#   ) +
#   coord_fixed() +
#   theme_minimal() +
#   theme(
#     axis.text.x = element_text(angle = 45, hjust = 1)
#   ) +
#   labs(
#     title = "P-Value Matrix (0-1 Color Scale)",
#     x = "",
#     y = ""
#   )

# #Calculate significance of the correlation matrix "All Variables" from previous section
#  library(corrplot)
#
# # Function from corrplot documentation:
# cor_test_results <- cor.mtest(urban[, -1], conf.level = 0.95)
#
# p_values <- cor_test_results$p   # Extract p-value matrix
#
# # Color palette from 0 (significant) to 1 (not significant)
# pval_palette <- colorRampPalette(c("darkred", "white", "darkblue"))(200)
#
# # Visualize significance on the correlation plot
# corrplot(
#   p_values,
#   method = "color",
#   col = pval_palette,
#   tl.col = "black",
#   tl.srt = 45,
#   cl.lim = c(0, 1),       # <--- sets the legend range explicitly
#   title = "P-Value Matrix for Correlations",
#   mar = c(0,0,2,0)
# )
```

Feature Removal

- Remove highly correlated features that are redundant based on the correlation analysis done previously. Explain why you are removing each feature.

```
# Find feature pairs that have correaltion > abs(0.9)
high_cor_pairs <- which(abs(cor_all) > 0.9 & abs(cor_all) < 1, arr.ind = TRUE)

# Get unique features to remove (keep only one from each pair)
features_to_remove <- unique(rownames(high_cor_pairs)[high_cor_pairs[,1] < high_cor_pairs[,2]])
print(features_to_remove) #"Bright"  "Mean_G"  "BrdIndx" "SD_G"    "GLCM1"
```

```
## [1] "Bright"  "Mean_G"  "BrdIndx" "SD_G"    "GLCM1"
```

```
  # Justification for removal:
  # "Bright": Highly correlated with Mean_R and Mean_NIR, which are more informative
  # "Mean_G": Highly correlated with Mean_R, so we keep Mean_R as it may be more relevant for urban lan
  # "BrdIndx": Highly correlated with Compact and ShpIndx, which provide better shape information
  # "SD_G": Highly correlated with SD_R, so we keep SD_R for texture representation
  # "GLCM1": Highly correlated with GLCM2 and GLCM3, so we keep GLCM2 for texture representation

# Remove the identified features from the dataset
urban_trimmed <- urban %>%
  select(-all_of(features_to_remove))

View(urban_trimmed)
```

Feature Log Transformation + Normalization

- Normalize the features as they are all in different scales

```
# Vector of variables to log10(1 + x) transform
vars_to_log <- c(
  "Compact",
  "LW",
  "Area",
  "BordLngth",
  "SD_R",
  "SD_NIR",
  "Round",
  "GLCM3"
)

# Create a copy of the "urban_trimmmed" dataset (will be the transformed version)
urban_transformed <- urban_trimmed

# Apply log10(1 + x) to each selected variable in the *new* dataset
# Loop, create new log variables, remove original
for (var in vars_to_log) {

  # Create new variable name "log_" prefix
  new_name <- paste0("log_", var)

  # Log transform and assign
  urban_transformed[[new_name]] <- log10(1 + urban_transformed[[var]])

  # Remove old untransformed variable
  urban_transformed[[var]] <- NULL
}

### Now 8/16 predictors have been log transformed

#Standardize all variables prior to running a PCA
  #Z-Score normalization
urban_scaled <- urban_transformed
urban_scaled[, -1] <- scale(urban_scaled[, -1]) #Exclude class column
```

```r
View(urban_scaled)
```

Summary Statistics for Cleaned Data

```r
#Ensure all predictors are numeric
str(urban_scaled)  #Yes are currently numeric
```

```
## 'data.frame':    675 obs. of  17 variables:
##  $ class        : chr  "car " "concrete " "concrete " "concrete " ...
##  $ Mean_R       : num  1.127 0.842 1.087 1.249 0.639 ...
##  $ Mean_NIR     : num  0.75 0.304 0.731 0.778 0.334 ...
##  $ NDVI         : num  -0.636 -0.768 -0.571 -0.702 -0.505 ...
##  $ ShpIndx      : num  -1.287 1.527 -0.679 0.82 2.63 ...
##  $ Rect         : num  1.016 -1.045 0.236 -0.265 -0.432 ...
##  $ Dens         : num  0.305 -0.598 0.166 -0.875 -2.564 ...
##  $ Assym        : num  0.191 0.516 0.678 1.126 1.41 ...
##  $ GLCM2        : num  -1.91 0.808 -0.632 -0.852 0.558 ...
##  $ log_Compact  : num  -1.233 0.575 -0.461 0.313 0.361 ...
##  $ log_LW       : num  -0.2114 -0.0309 -0.0781 0.6103 4.463 ...
##  $ log_Area     : num  -2.92 0.926 0.699 1.23 1.373 ...
##  $ log_BordLngth: num  -2.705 1.312 0.144 1.261 1.902 ...
##  $ log_SD_R     : num  0.9104 0.2762 -0.4448 -1.0444 -0.0606 ...
##  $ log_SD_NIR   : num  0.7729 0.0772 -0.5783 -0.9718 -0.2174 ...
##  $ log_Round    : num  -0.583 0.598 -0.334 0.573 0.803 ...
##  $ log_GLCM3    : num  2.414 0.529 -0.313 0.566 -0.549 ...
```

```r
#Ensure 'class' variable is treated as a factor with 9 classifications
urban_scaled$class <- as.factor(urban_scaled$class)
str(urban_scaled$class)  # Factor w/ 9 levels
```

```
##  Factor w/ 9 levels "asphalt ","building ",..: 3 4 4 4 4 9 3 3 2 9 ...
```

```r
#Get counts for each class
table(urban_scaled$class)
```

```
##
##  asphalt  building       car  concrete     grass      pool    shadow      soil
##       59       122        36       116       112        29        61        34
##     tree
##      106
```

```r
#Summary statistics of the cleaned dataset
summary(urban_scaled)
```

```
##        class          Mean_R            Mean_NIR            NDVI
##  building :122   Min.   :-1.85854   Min.   :-2.4531   Min.   :-2.4752
##  concrete :116   1st Qu.:-0.86137   1st Qu.:-0.6688   1st Qu.:-0.7020
##  grass    :112   Median :-0.05424   Median : 0.4028   Median :-0.4393
```

```
##  tree     :106   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  shadow   : 61   3rd Qu.: 1.03619   3rd Qu.: 0.7268   3rd Qu.: 0.8085
##  asphalt  : 59   Max.   : 1.36108   Max.   : 1.4793   Max.   : 2.3847
##  (Other)  : 99
##     ShpIndx             Rect              Dens              Assym
##  Min.   :-1.5687   Min.   :-3.7184   Min.   :-2.79547   Min.   :-2.5324
##  1st Qu.:-0.7777   1st Qu.:-0.5992   1st Qu.:-0.72496   1st Qu.:-0.7847
##  Median :-0.1235   Median : 0.1249   Median : 0.07317   Median : 0.1908
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.6523   3rd Qu.: 0.7933   3rd Qu.: 0.76720   3rd Qu.: 0.8411
##  Max.   : 3.6797   Max.   : 1.8516   Max.   : 2.01645   Max.   : 1.4101
##
##      GLCM2          log_Compact          log_LW           log_Area
##  Min.   :-3.3795   Min.   :-1.6909   Min.   :-1.0917   Min.   :-4.01660
##  1st Qu.:-0.7493   1st Qu.:-0.7425   1st Qu.:-0.6779   1st Qu.:-0.54712
##  Median :-0.1321   Median :-0.1464   Median :-0.2332   Median : 0.04923
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.7201   3rd Qu.: 0.5827   3rd Qu.: 0.2789   3rd Qu.: 0.65224
##  Max.   : 2.3511   Max.   : 5.1114   Max.   : 6.2160   Max.   : 2.85640
##
##  log_BordLngth        log_SD_R          log_SD_NIR        log_Round
##  Min.   :-3.5308   Min.   :-2.95640   Min.   :-2.3926   Min.   :-3.00966
##  1st Qu.:-0.6285   1st Qu.:-0.68976   1st Qu.:-0.7927   1st Qu.:-0.65239
##  Median : 0.1184   Median :-0.02974   Median :-0.1407   Median : 0.09078
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.6939   3rd Qu.: 0.71562   3rd Qu.: 0.8279   3rd Qu.: 0.66535
##  Max.   : 2.6612   Max.   : 2.48020   Max.   : 2.3164   Max.   : 3.69019
##
##    log_GLCM3
##  Min.   :-2.87784
##  1st Qu.:-0.72491
##  Median : 0.02094
##  Mean   : 0.00000
##  3rd Qu.: 0.71505
##  Max.   : 3.05387
##
```

4. Data Pre Processing

- Transformed variables already prior to normalization
- Normalization was included as a Data Preprocessing step on the rubric

5. Clustering

- Remove target label
- Determine proper clustering algorithm to use

1. Try k-means with k=9 since I have nine class labels
2. Try hierarchical clustering to get an idea for the number X of clusters that "naturally occur"

- Use "wards distance" as the linkage metrics

3. Try running k-means with X clusters that resulted from Hierarchical
4. Determine optimal number of clusters using..

- Silhouette Score
- Elbow Plot

5. Run K-means with the "optimal" number of clusters

Try k-means with k=9 since I have nine class labels

```
#Remove target label
urban_km_scaled <- urban_scaled[, -1] #Remove class column
View(urban_km_scaled)

# 1. K-means, with K=9 (because we have 9 classes)
set.seed(947829)  # for reproducibility

km9 <- kmeans(
  x = urban_km_scaled,
  centers = 9,
  nstart = 200,   # multiple random starts for a better solution
  iter.max = 100
)

# 2. View K-means results
km9$size   # Cluster sizes
```

```
## [1]  50 101  63  99 122  48  35 101  56
```

```
km9$tot.withinss  # Total within-cluster sum of squares (compactness measure, lower=compacter)
```

```
## [1] 4276.098
```

```
km9$betweenss / km9$totss # Proportion of variance explained by each cluster ()
```

```
## [1] 0.6034775
```

```
# 3. Attach the cluster labels to a data frame that HAS class labels already
urban_scaled$cluster_km9 <- factor(km9$cluster)

# 4. Compare clusters to true class labels
table(
  Cluster = urban_scaled$cluster_km9,
  Class   = urban_scaled$class
)
```

```
##         Class
## Cluster asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##       1       12         4    0         0      0     0      31     0     3
```

```
##       2        0       73    0        9     11      2       0     6     0
##       3        1        6    0       48      0      0       0     7     1
##       4        1        3    1        1     35      3       3     3    49
##       5        6        1   16        3     34      1       5     3    53
##       6        0        2   17        0      2     23       4     0     0
##       7        2        0    0       23      7      0       0     3     0
##       8        0       33    1       32     23      0       0    12     0
##       9       37        0    1        0      0      0      18     0     0
```

Determine Optimal K from K-means

- Use a large number of starts to get optimal solution

```r
library(cluster)

# Use the same predictor matrix as your k-means runs (scaled, no class)
urban_km_scaled <- urban_scaled[, -1]

# Store silhouette scores
k_range <- 2:20
sil_scores_kmeans <- numeric(length(k_range))

# Compute silhouette for each k
for (i in seq_along(k_range)) {
  k <- k_range[i]

  # Run k-means
  km_tmp <- kmeans(
    x = urban_km_scaled,
    centers = k,
    nstart = 300,  # Run the entire algorithm "nstart" times, keeping the best starting spots
    iter.max = 200
  )

  # Compute silhouette
  sil <- silhouette(km_tmp$cluster, dist(urban_km_scaled))

  sil_scores_kmeans[i] <- mean(sil[, "sil_width"])
}

# Combine into a dataframe
sil_results_kmeans <- data.frame(
  k = k_range,
  avg_silhouette = sil_scores_kmeans
)

sil_results_kmeans
```

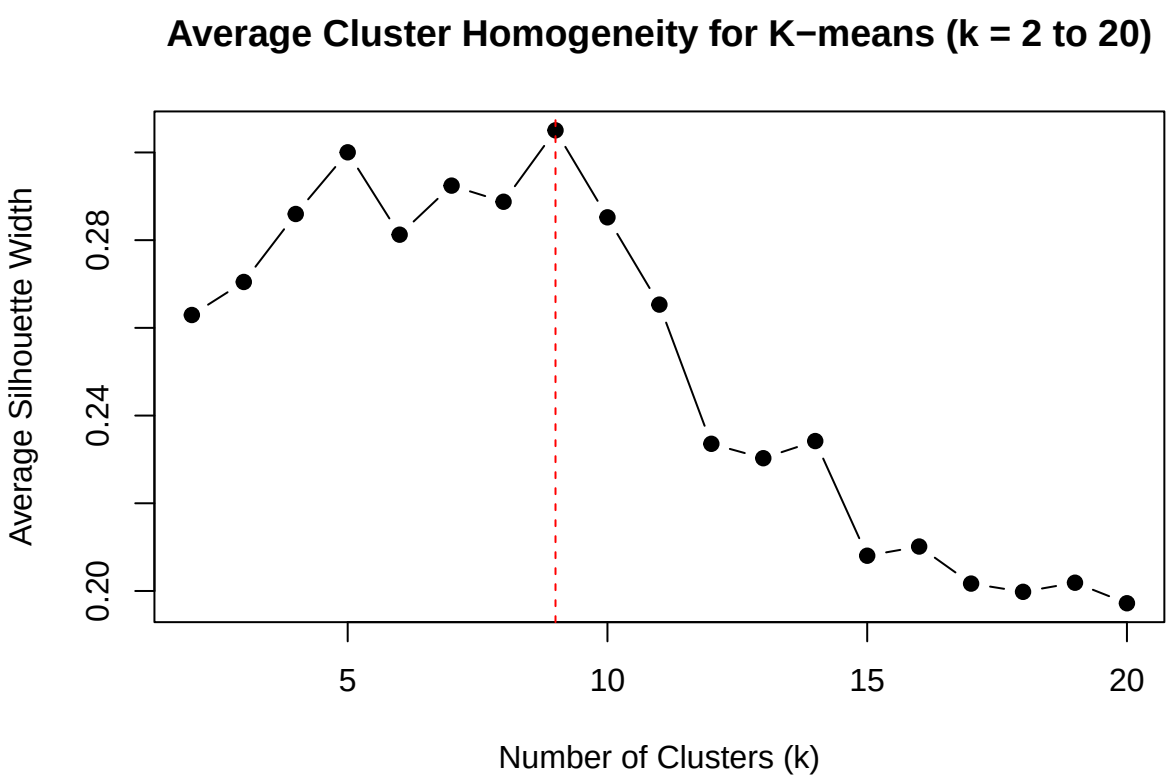
```
##    k avg_silhouette
## 1  2      0.2629307
## 2  3      0.2704651
## 3  4      0.2859599
## 4  5      0.3000246
```

```
## 5   6      0.2812415
## 6   7      0.2924305
## 7   8      0.2887514
## 8   9      0.3050361
## 9  10      0.2852008
## 10 11      0.2653023
## 11 12      0.2335608
## 12 13      0.2302749
## 13 14      0.2341818
## 14 15      0.2080365
## 15 16      0.2101337
## 16 17      0.2016838
## 17 18      0.1998056
## 18 19      0.2018854
## 19 20      0.1972023
```

```r
# Best k based on silhouette
best_k_kmeans <- sil_results_kmeans$k[which.max(sil_results_kmeans$avg_silhouette)]
best_k_kmeans
```

```
## [1] 9
```

```r
# Plot silhouette vs k
plot(
  sil_results_kmeans$k,
  sil_results_kmeans$avg_silhouette,
  type = "b",
  pch = 19,
  xlab = "Number of Clusters (k)",
  ylab = "Average Silhouette Width",
  main = "Average Cluster Homogeneity for K-means (k = 2 to 20)"
)

abline(v = best_k_kmeans, col = "red", lty = 2)
```




```
##############################################################################################################
# 1. Run K-means again using the optimal k from silhouette
##############################################################################################################
set.seed(37920472)

km_best <- kmeans(
  x = urban_km_scaled,
  centers = best_k_kmeans,
  nstart = 300,
  iter.max = 200
)

# 2. View K-means results for the chosen k
km_best$size                      # Cluster sizes
```

```
## [1]  35 122 101 101  63  48  56  99  50
```

```
km_best$tot.withinss              # Total within-cluster sum of squares (compactness)
```

```
## [1] 4276.098
```

```
km_best$betweenss / km_best$totss  # Proportion of variance explained
```

```
## [1] 0.7115966
```

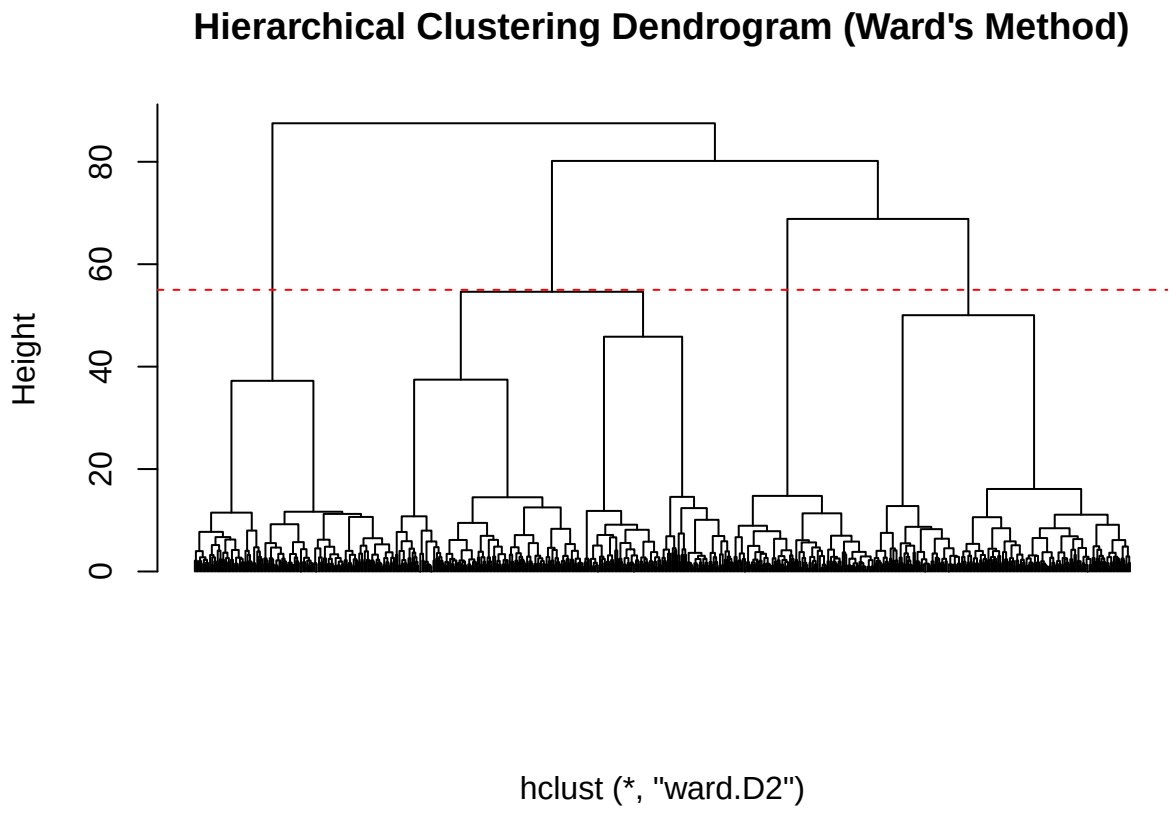
```r
# 3. Attach cluster labels to a data frame that HAS class labels
urban_scaled$cluster_km_best <- factor(km_best$cluster)

# 4. Compare clusters to true class labels
table(
  Cluster = urban_scaled$cluster_km_best,
  Class   = urban_scaled$class
)
```

```
##        Class
## Cluster asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##       1        2         0    0        23      7     0       0     3     0
##       2        6         1   16         3     34     1       5     3    53
##       3        0        73    0         9     11     2       0     6     0
##       4        0        33    1        32     23     0       0    12     0
##       5        1         6    0        48      0     0       0     7     1
##       6        0         2   17         0      2    23       4     0     0
##       7       37         0    1         0      0     0      18     0     0
##       8        1         3    1         1     35     3       3     3    49
##       9       12         4    0         0      0     0      31     0     3
```

Try hierarchical clustering to get an idea for the number X of clusters that "naturally occur"

```r
#Hierarchical Clustering to determine natural number of clusters
#-----------------------------------------
# 1. Prepare data for hierarchical clustering
#    (use the same scaled predictors as k-means)
#-----------------------------------------

# urban_scaled already has class in column 1, predictors in cols 2:...
urban_hc_data <- urban_scaled[, -1]   # remove class column

#-----------------------------------------
# 2. Compute distance matrix (Euclidean)
#-----------------------------------------
dist_mat <- dist(urban_hc_data, method = "euclidean")

#-----------------------------------------
# 3. Hierarchical clustering with Ward's method
#   ("ward.D2" is the recommended Ward linkage)
#-----------------------------------------
hc_ward <- hclust(dist_mat, method = "ward.D2")

#-----------------------------------------
# 4. Plot the dendrogram
#-----------------------------------------
plot(
  hc_ward,
  labels = FALSE,          # don't plot every label (too many points)
  hang   = -1,
```

```
  main   = "Hierarchical Clustering Dendrogram (Ward's Method)",
  xlab   = "",
  ylab   = "Height"
)

# Optional: add a horizontal line to visually inspect a cut
# (adjust 'h' after you look at the dendrogram)
abline(h = 55, col = "red", lty = 2)
```




```
#Extract number of clusters from where the line hit
  # Cut the tree at height h
clusters_h <- cutree(hc_ward, h = 55)

# Number of clusters at that cut
Y <- length(unique(clusters_h))
Y
```

```
## [1] 4
```

```
#----------------------------------------
# 5. (Optional) Try cutting the tree at X clusters once you decide X
#----------------------------------------
# Can cut based on height (red line) or number of clusters
X <- 13  # <--- set based on dendrogram inspection
```

```r
clusters_hc_X <- cutree(hc_ward, k = X)

# Attach to data for later comparison
urban_scaled$cluster_hc_X <- factor(clusters_hc_X)
table(urban_scaled$cluster_hc_X, urban_scaled$class)
```

```
##
##      asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##   1        0         2    7         0      2     0       0     0     0
##   2        1         6    0        48      0     0       0     7     1
##   3        0        23    0        24      0     0       0     0     0
##   4        2         0    0        23      7     0       0     3     0
##   5        2         1    9         0     12     1       4     1    16
##   6        0         1   10         0      0    25       5     0     0
##   7        0        23    0         6     10     0       0     5     0
##   8        1         2    1         1     35     1       2     3    49
##   9        0        50    0         3      1     2       0     1     0
##   10      37         0    1         0      0     0      18     0     0
##   11       0        10    1         8     23     0       0    12     0
##   12      12         4    0         0      0     0      31     0     3
##   13       4         0    7         3     22     0       1     2    37
```

Determine optimal K from Hierarchical

- 13 ended up being optimal here

```r
library(cluster)

# 1. Data & distance used for hierarchical clustering
# (Assumes urban_scaled has class in column 1, predictors in cols 2+:)
urban_hc_data <- urban_scaled[, -1]

# Distance matrix (same you would have used for hc_ward)
dist_mat <- dist(urban_hc_data, method = "euclidean")

# 2. Hierarchical clustering with Ward's method (if not already done)
hc_ward <- hclust(dist_mat, method = "ward.D2")

# 3. Evaluate average silhouette (homogeneity) for k = 2 to 15
k_range   <- 2:20
avg_sil   <- numeric(length(k_range))

for (i in seq_along(k_range)) {
  k  <- k_range[i]
  cl <- cutree(hc_ward, k = k)          # cluster assignment from hierarchical

  sil <- silhouette(cl, dist_mat)       # silhouette object
  avg_sil[i] <- mean(sil[, "sil_width"])  # average silhouette width for this k
}

# 4. Put results in a data frame
sil_results <- data.frame(
```

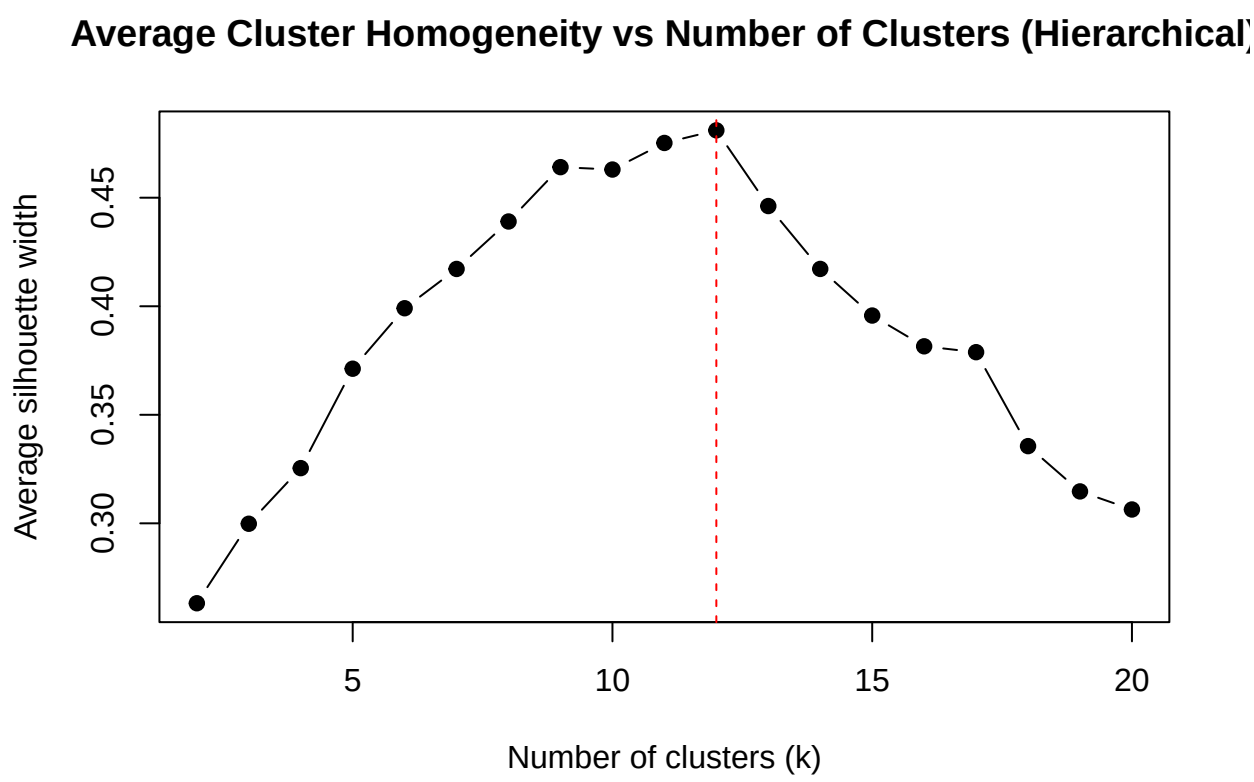
```
  k       = k_range,
  avg_sil = avg_sil
)

sil_results
```

```
##     k   avg_sil
## 1   2 0.2631956
## 2   3 0.2998035
## 3   4 0.3254279
## 4   5 0.3712221
## 5   6 0.3990895
## 6   7 0.4171823
## 7   8 0.4390431
## 8   9 0.4640787
## 9  10 0.4629746
## 10 11 0.4752164
## 11 12 0.4810247
## 12 13 0.4461551
## 13 14 0.4171968
## 14 15 0.3957145
## 15 16 0.3815485
## 16 17 0.3788468
## 17 18 0.3355715
## 18 19 0.3147034
## 19 20 0.3063771
```

```
# 5. Find the k with the best (highest) homogeneity
best_k <- sil_results$k[which.max(sil_results$avg_sil)]
best_k
```

```
## [1] 12
```

```
# 6. (Optional) Plot average silhouette vs k
plot(
  sil_results$k, sil_results$avg_sil,
  type = "b",
  xlab = "Number of clusters (k)",
  ylab = "Average silhouette width",
  main = "Average Cluster Homogeneity vs Number of Clusters (Hierarchical)",
  pch  = 19
)
abline(v = best_k, col = "red", lty = 2)
```



Try running each algorithm with the optimal K ### K-means with K=13

```
# Predictor matrix (scaled, no class)
urban_km_scaled <- urban_scaled[, -1]

set.seed(947829)

km13 <- kmeans(
  x        = urban_km_scaled,
  centers  = 13,      # from hierarchical optimal k
  nstart   = 300,
  iter.max = 200
)

# Inspect metrics
km13$size                          # cluster sizes
```

```
##  [1] 35 76 50 57 54 56 99 11 63 37 44 47 46
```

```
km13$tot.withinss                  # total within-cluster SSE
```

```
## [1] 3849.246
```

```
km13$betweenss / km13$totss        # proportion of variance explained
```

```
## [1] 0.8600272
```

```r
# Attach labels back to data with class
urban_scaled$cluster_km13 <- factor(km13$cluster)

# Compare to true class labels
table(
  Cluster = urban_scaled$cluster_km13,
  Class   = urban_scaled$class
)
```

```
##         Class
## Cluster asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##       1        2         0    0        23      7     0       0     3     0
##       2        4         0    7         3     22     0       1     2    37
##       3       12         4    0         0      0     0      31     0     3
##       4        0        50    0         3      1     2       0     1     0
##       5        0        10    1         8     23     0       0    12     0
##       6       37         0    1         0      0     0      18     0     0
##       7        1         3    1         1     35     3       3     3    49
##       8        0         2    7         0      2     0       0     0     0
##       9        1         6    0        48      0     0       0     7     1
##       10       0         0   10         0      0    23       4     0     0
##       11       0        23    0         6     10     0       0     5     0
##       12       0        23    0        24      0     0       0     0     0
##       13       2         1    9         0     12     1       4     1    16
```

Hierarchical with K=13

```r
library(cluster)

# Use same distance + linkage as before
urban_hc_data <- urban_scaled[, -1]
dist_mat      <- dist(urban_hc_data, method = "euclidean")
hc_ward       <- hclust(dist_mat, method = "ward.D2")

# Cut tree into 12 clusters (k = 12 from K-means silhouette)
clusters_hc12 <- cutree(hc_ward, k = 13)

# Attach to data
urban_scaled$cluster_hc12 <- factor(clusters_hc12)

# Compare hierarchical clusters to true classes
table(
  Cluster = urban_scaled$cluster_hc12,
  Class   = urban_scaled$class
)
```

```
##         Class
## Cluster asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##       1        0         2    7         0      2     0       0     0     0
##       2        1         6    0        48      0     0       0     7     1
##       3        0        23    0        24      0     0       0     0     0
```

```
##      4         2         0    0        23      7     0       0     3     0
##      5         2         1    9         0     12     1       4     1    16
##      6         0         0   10         0      0    23       4     0     0
##      7         0        23    0         6     10     0       0     5     0
##      8         1         3    1         1     35     3       3     3    49
##      9         0        50    0         3      1     2       0     1     0
##      10       37         0    1         0      0     0      18     0     0
##      11        0        10    1         8     23     0       0    12     0
##      12       12         4    0         0      0     0      31     0     3
##      13        4         0    7         3     22     0       1     2    37
```

Compare agreement between k=9 and k=13 directly

- See which points overlap

```
# K-Means cluster assignments for k=9 and k=13
urban_scaled$cluster_km9
```

```
##   [1] 6 3 8 3 7 5 5 6 2 4 2 9 8 6 1 2 4 2 2 1 6 1 7 4 4 3 8 2 2 8 9 1 8 4 5 3 9
##  [38] 3 4 3 3 4 8 2 2 2 8 1 8 8 1 4 6 6 1 6 2 4 4 4 9 5 6 6 2 4 2 8 2 8 3 2 6 6
##  [75] 6 9 2 8 5 1 6 6 2 2 2 1 8 8 9 2 7 8 8 8 3 4 6 5 1 5 8 9 5 6 5 9 8 2 7 5 9
## [112] 4 4 6 3 5 8 8 3 5 5 5 1 6 4 1 5 2 2 8 7 9 9 8 9 7 9 6 6 6 5 1 7 1 7 1 2 9
## [149] 1 4 5 6 5 1 8 8 3 5 9 7 8 5 4 6 8 4 4 3 5 1 9 5 4 2 1 8 4 5 4 5 1 7 4 9 5
## [186] 5 5 8 4 5 4 2 7 9 5 4 2 6 3 3 2 3 9 4 5 2 5 6 1 8 2 4 5 5 3 2 2 4 4 6 5 5
## [223] 6 2 5 1 8 4 6 4 4 5 2 2 6 3 2 2 6 3 3 8 5 5 5 9 1 4 7 2 8 4 4 2 2 4 2 4 3
## [260] 5 5 9 4 8 5 5 5 2 4 6 4 9 5 6 5 8 5 8 4 1 5 4 5 9 9 5 4 8 5 1 5 1 1 8 3 3
## [297] 2 3 5 5 4 6 5 4 1 5 4 3 1 5 3 3 9 1 4 4 9 5 8 8 4 4 5 2 1 4 3 4 2 2 1 4 8
## [334] 1 2 4 5 5 3 5 3 8 8 5 2 5 6 2 2 2 2 5 4 8 5 5 4 2 1 9 8 1 3 3 7 2 5 2 5 5
## [371] 8 4 4 2 6 2 2 5 6 2 3 8 7 3 6 2 5 4 6 4 3 5 7 5 4 4 8 7 7 8 8 5 3 2 8 8 5
## [408] 2 8 3 7 1 7 3 2 3 2 4 2 3 9 5 1 2 8 8 4 5 5 5 5 8 8 2 8 8 8 2 8 6 5 5 4 2
## [445] 8 7 1 3 8 1 4 4 6 9 9 9 3 2 2 8 2 2 7 8 5 2 4 1 8 8 2 8 2 9 5 4 5 7 1 4 2
## [482] 7 4 9 8 8 1 7 7 5 2 9 8 5 8 9 3 3 4 2 2 5 5 3 6 3 2 7 5 5 8 4 2 4 9 3 2 3
## [519] 8 5 5 3 3 4 2 5 7 4 1 9 5 4 5 8 5 9 5 9 4 5 1 5 3 6 4 2 4 4 8 3 5 9 9 8 9
## [556] 7 7 3 5 5 2 4 2 6 9 9 8 3 9 7 8 2 2 8 1 6 5 8 5 9 4 5 8 4 2 7 6 8 9 3 1 9
## [593] 8 1 9 8 2 1 4 6 4 4 4 5 8 5 3 3 8 8 5 5 3 7 4 8 7 2 8 3 3 4 8 5 8 6 2 8 4
## [630] 7 7 4 5 8 8 9 2 5 1 5 5 9 6 9 9 5 9 9 4 4 8 2 8 2 4 5 4 2 2 3 3 3 9 2 5 8
## [667] 5 5 8 1 2 4 2 8 8
## Levels: 1 2 3 4 5 6 7 8 9
```

```
urban_scaled$cluster_km13
```

```
##   [1] 8  9  12 9  1  13 13 10 11 7  4  6  5  8  3  4  7  11 4  3  10 3  1  7  7
##  [26] 9  5  4  11 5  6  3  12 7  2  9  6  9  7  9  9  7  12 4  4  11 5  3  5  5
##  [51] 3  7  8  8  3  10 4  7  7  7  6  2  10 8  11 7  4  12 11 5  9  4  10 10 10
##  [76] 6  4  5  13 3  10 10 11 11 4  3  12 5  6  11 1  5  12 5  9  7  10 2  3  13
## [101] 5  6  2  10 2  6  5  11 1  2  6  7  7  10 9  2  5  5  9  2  13 13 3  10 7
## [126] 3  2  4  11 5  1  6  6  5  6  1  6  10 10 10 13 3  1  3  1  3  4  6  3  7
## [151] 13 10 2  3  12 12 9  2  6  1  5  2  7  10 5  7  7  9  2  3  6  13 7  11 3
## [176] 5  7  13 7  2  3  1  7  6  2  2  2  12 7  2  7  11 1  6  2  7  4  10 9  9
## [201] 4  9  6  7  2  11 13 10 3  12 11 7  2  2  9  4  4  7  7  10 2  2  10 4  2
## [226] 3  12 7  10 7  7  2  11 4  8  9  4  4  10 9  9  5  2  13 2  6  3  7  1  4
## [251] 12 7  7  4  11 7  11 7  9  2  13 6  7  5  13 2  2  4  7  8  7  6  2  10 2
```

```
## [276] 5  2  5  7  3  2  7  13 6  6  13 7  12 13 3  2  3  3  12 9  9  4  9  2  13
## [301] 7  8  13 7  3  2  7  9  3  13 9  9  6  3  7  7  6  13 12 5  7  7  2  4  3
## [326] 7  9  7  4  11 3  7  5  3  4  7  2  2  9  2  9  12 12 2  4  13 8  4  11 4
## [351] 4  2  7  5  2  2  7  4  3  6  12 3  9  9  1  11 13 11 2  13 5  7  7  4  10
## [376] 11 11 2  10 11 9  12 1  9  10 11 13 7  10 7  9  13 1  2  7  7  5  1  1  5
## [401] 5  13 9  4  5  12 2  11 5  9  1  3  1  9  4  9  11 7  11 9  6  2  3  11 5
## [426] 5  7  13 13 13 13 12 12 11 5  5  5  11 12 10 2  2  7  11 12 1  3  9  12 3
## [451] 7  7  10 6  6  6  9  4  11 12 4  4  1  5  13 11 7  3  5  12 4  12 4  6  2
## [476] 7  2  1  3  7  4  1  7  6  12 12 3  1  1  2  11 6  12 2  5  6  9  9  7  4
## [501] 4  2  2  9  10 9  11 1  2  2  12 7  4  7  6  9  4  9  12 13 2  9  9  7  4
## [526] 2  1  7  3  6  2  7  2  12 13 6  13 6  7  2  3  2  9  10 7  4  7  7  5  9
## [551] 13 6  6  12 6  1  1  9  2  13 11 7  4  8  6  6  12 9  6  1  5  4  11 5  3
## [576] 10 13 12 13 6  7  2  12 7  4  1  10 5  6  9  3  6  12 3  6  5  4  3  7  8
## [601] 7  7  7  2  12 2  9  9  5  5  13 2  9  1  7  5  1  4  12 9  9  7  5  2  12
## [626] 10 11 12 7  1  1  7  2  12 12 6  4  13 3  13 13 6  10 6  6  13 6  6  7  7
## [651] 5  11 5  11 7  2  7  4  11 9  9  9  6  4  13 12 2  13 5  3  4  7  11 5  12
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13
```

```
#Hierarchical versions
urban_hc_data <- urban_scaled[, -1]
dist_mat      <- dist(urban_hc_data, method = "euclidean")
hc_ward       <- hclust(dist_mat, method = "ward.D2")

# Hierarchical at k = 9
urban_scaled$cluster_hc9  <- factor(cutree(hc_ward, k = 9))

# Hierarchical at k = 13
urban_scaled$cluster_hc13 <- factor(cutree(hc_ward, k = 13))


### Contingency tables for comparison
### --------------------------

cat("\n==============================\n")
```

```
##
## ==============================
```

```
cat("K-MEANS (k = 9)  vs  HIERARCHICAL (k = 9)\n")
```

```
## K-MEANS (k = 9)  vs  HIERARCHICAL (k = 9)
```

```
cat("==============================\n")
```

```
## ==============================
```

```
print(table(
  k_means_9  = urban_scaled$cluster_km9,
  Hierarchical_9  = urban_scaled$cluster_hc9
))
```

```
##           Hierarchical_9
## k_means_9  1  2  3  4  5  6  7  8  9
##         1  0  0  0  0  0  0  0 50  0
##         2  0  0  0 44  0 57  0  0  0
##         3  0 63  0  0  0  0  0  0  0
##         4  0  0  0  0 99  0  0  0  0
##         5  0  0  0 46  0  0  0  0 76
##         6 48  0  0  0  0  0  0  0  0
##         7  0  0 35  0  0  0  0  0  0
##         8 47  0  0  0  0  0 54  0  0
##         9  0  0  0  0  0  0 56  0  0
```

```
cat("\n==============================\n")
```

```
##
## ==============================
```

```
cat("K-MEANS (k = 13)  vs  HIERARCHICAL (k = 13)\n")
```

```
## K-MEANS (k = 13)  vs  HIERARCHICAL (k = 13)
```

```
cat("==============================\n")
```

```
## ==============================
```

```
print(table(
  k_means_13 = urban_scaled$cluster_km13,
  Hierarchical_13 = urban_scaled$cluster_hc13
))
```

```
##            Hierarchical_13
## k_means_13  1  2  3  4  5  6  7  8  9 10 11 12 13
##         1   0  0  0 35  0  0  0  0  0  0  0  0  0
##         2   0  0  0  0  0  0  0  0  0  0  0  0 76
##         3   0  0  0  0  0  0  0  0  0  0  0 50  0
##         4   0  0  0  0  0  0  0  0 57  0  0  0  0
##         5   0  0  0  0  0  0  0  0  0  0 54  0  0
##         6   0  0  0  0  0  0  0  0  0 56  0  0  0
##         7   0  0  0  0  0  0  0 99  0  0  0  0  0
##         8  11  0  0  0  0  0  0  0  0  0  0  0  0
##         9   0 63  0  0  0  0  0  0  0  0  0  0  0
##         10  0  0  0  0  0 37  0  0  0  0  0  0  0
##         11  0  0  0  0  0  0 44  0  0  0  0  0  0
##         12  0  0 47  0  0  0  0  0  0  0  0  0  0
##         13  0  0  0  0 46  0  0  0  0  0  0  0  0
```

```
### ============================
### K-MEANS (k = 9) vs TRUE CLASS
### ============================
cat("\n==============================\n")
```

```
##
## ============================
```

```
cat("K-MEANS (k = 9)  vs  TRUE CLASS\n")
```

```
## K-MEANS (k = 9)  vs  TRUE CLASS
```

```
cat("============================\n")
```

```
## ============================
```

```
print(table(
  KM9   = urban_scaled$cluster_km9,
  Class = urban_scaled$class
))
```

```
##    Class
## KM9 asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##   1       12         4    0         0      0     0      31     0     3
##   2        0        73    0         9     11     2       0     6     0
##   3        1         6    0        48      0     0       0     7     1
##   4        1         3    1         1     35     3       3     3    49
##   5        6         1   16         3     34     1       5     3    53
##   6        0         2   17         0      2    23       4     0     0
##   7        2         0    0        23      7     0       0     3     0
##   8        0        33    1        32     23     0       0    12     0
##   9       37         0    1         0      0     0      18     0     0
```

```
### ============================
### HIERARCHICAL (k = 9) vs TRUE CLASS
### ============================
cat("\n============================\n")
```

```
##
## ============================
```

```
cat("HIERARCHICAL (k = 9)  vs  TRUE CLASS\n")
```

```
## HIERARCHICAL (k = 9)  vs  TRUE CLASS
```

```
cat("============================\n")
```

```
## ============================
```

```
print(table(
  HC9   = urban_scaled$cluster_hc9,
  Class = urban_scaled$class
))
```

```
##    Class
## HC9 asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##   1        0        25   17        24      2    23       4     0     0
##   2        1         6    0        48      0     0       0     7     1
##   3        2         0    0        23      7     0       0     3     0
##   4        2        24    9         6     22     1       4     6    16
##   5        1         3    1         1     35     3       3     3    49
##   6        0        50    0         3      1     2       0     1     0
##   7       37        10    2         8     23     0      18    12     0
##   8       12         4    0         0      0     0      31     0     3
##   9        4         0    7         3     22     0       1     2    37
```

```
### ============================
### K-MEANS (k = 13) vs TRUE CLASS
### ============================
cat("\n==============================\n")
```

```
##
## ==============================
```

```
cat("K-MEANS (k = 13)  vs  TRUE CLASS\n")
```

```
## K-MEANS (k = 13)  vs  TRUE CLASS
```

```
cat("==============================\n")
```

```
## ==============================
```

```
print(table(
  KM13  = urban_scaled$cluster_km13,
  Class = urban_scaled$class
))
```

```
##     Class
## KM13 asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##   1        2         0    0        23      7     0       0     3     0
##   2        4         0    7         3     22     0       1     2    37
##   3       12         4    0         0      0     0      31     0     3
##   4        0        50    0         3      1     2       0     1     0
##   5        0        10    1         8     23     0       0    12     0
##   6       37         0    1         0      0     0      18     0     0
##   7        1         3    1         1     35     3       3     3    49
##   8        0         2    7         0      2     0       0     0     0
##   9        1         6    0        48      0     0       0     7     1
##   10       0         0   10         0      0    23       4     0     0
##   11       0        23    0         6     10     0       0     5     0
##   12       0        23    0        24      0     0       0     0     0
##   13       2         1    9         0     12     1       4     1    16
```

```r
### ============================
### HIERARCHICAL (k = 13) vs TRUE CLASS
### ============================
cat("\n==============================\n")
```

```
##
## ==============================
```

```r
cat("HIERARCHICAL (k = 13)  vs  TRUE CLASS\n")
```

```
## HIERARCHICAL (k = 13)  vs  TRUE CLASS
```

```r
cat("==============================\n")
```

```
## ==============================
```

```r
print(table(
  HC13  = urban_scaled$cluster_hc13,
  Class = urban_scaled$class
))
```

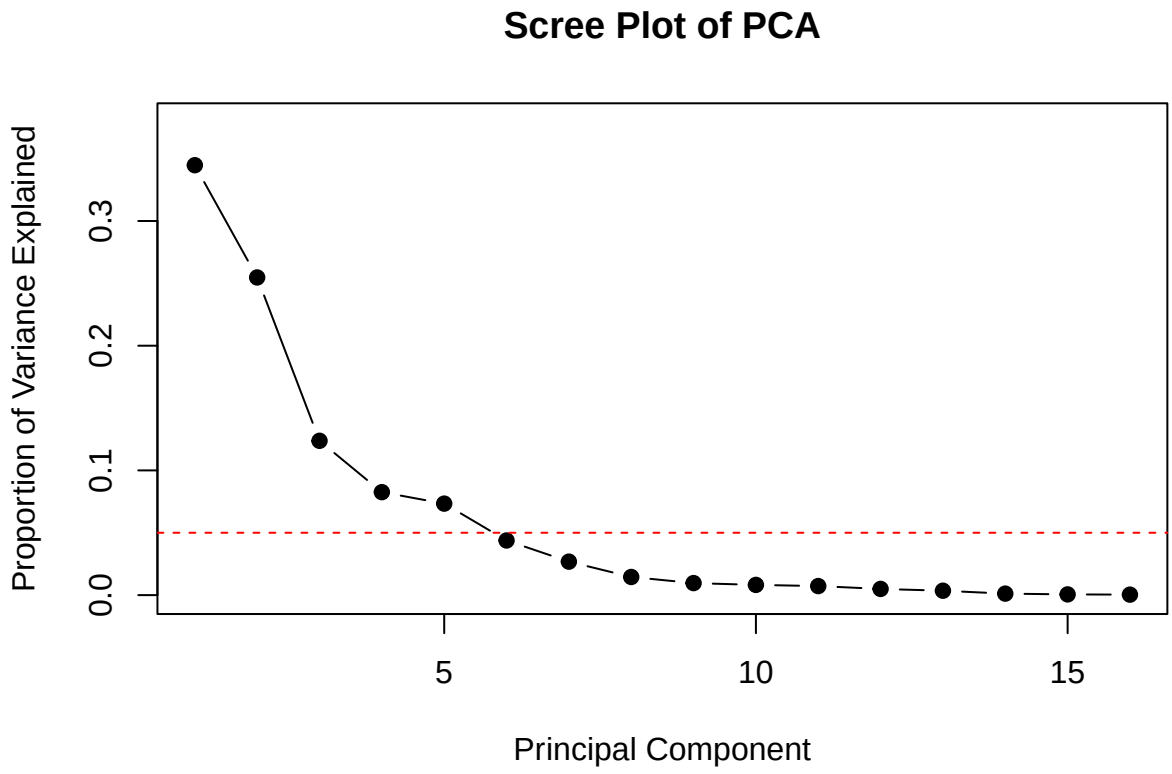
```
##     Class
## HC13 asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##   1         0         2    7         0      2     0       0     0     0
##   2         1         6    0        48      0     0       0     7     1
##   3         0        23    0        24      0     0       0     0     0
##   4         2         0    0        23      7     0       0     3     0
##   5         2         1    9         0     12     1       4     1    16
##   6         0         0   10         0      0    23       4     0     0
##   7         0        23    0         6     10     0       0     5     0
##   8         1         3    1         1     35     3       3     3    49
##   9         0        50    0         3      1     2       0     1     0
##   10       37         0    1         0      0     0      18     0     0
##   11        0        10    1         8     23     0       0    12     0
##   12       12         4    0         0      0     0      31     0     3
##   13        4         0    7         3     22     0       1     2    37
```

PCA for cluster visualization

##Set up the PCA

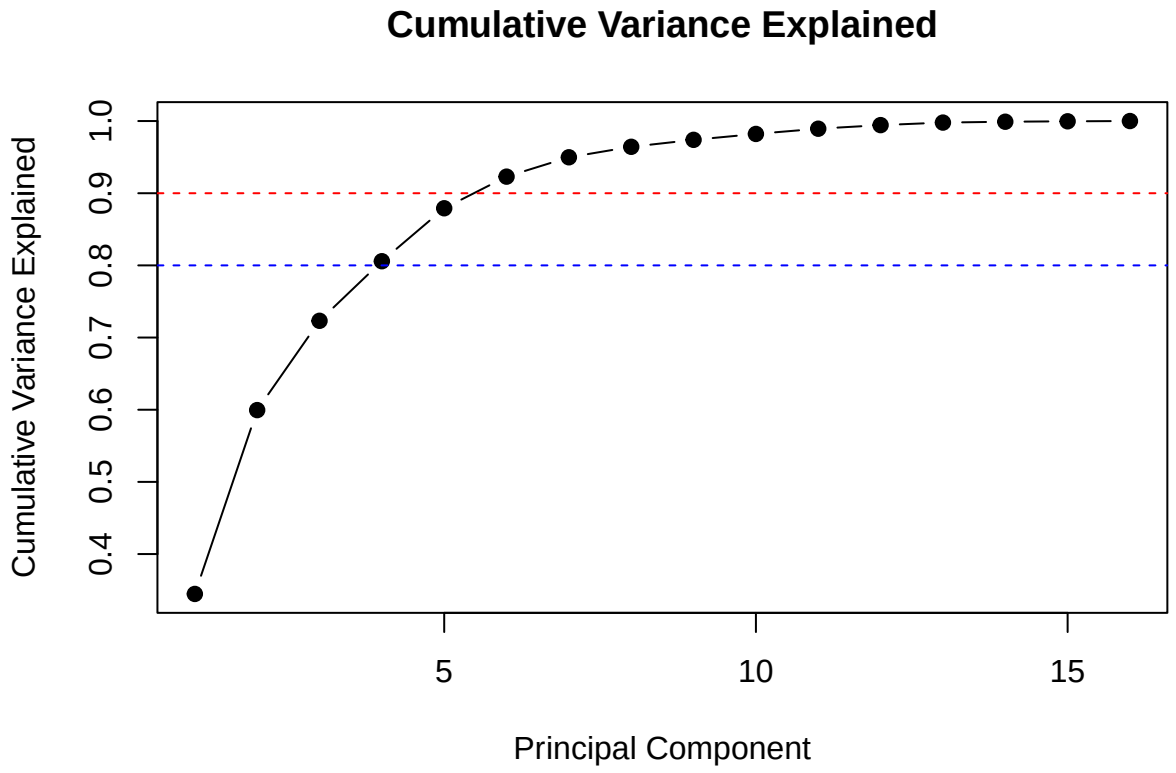
```r
library(dplyr)
library(ggplot2)

# 1. Use only numeric predictors for PCA (scaled already)
urban_num <- urban_scaled %>%
  select(where(is.numeric))      # drops class & cluster factors automatically

# 2. PCA on scaled features + Scree Plot
```

```r
pca_res <- prcomp(urban_num, center = FALSE, scale. = FALSE)
# (already standardized, so no need to re-center/scale)

# Variance explained
var_explained <- pca_res$sdev^2
prop_var_explained <- var_explained / sum(var_explained)

#-------------------------------------------------------------------------------
#-----Scree plot------
#-------------------------------------------------------------------------------
plot(
  prop_var_explained,
  type = "b",
  pch = 19,
  xlab = "Principal Component",
  ylab = "Proportion of Variance Explained",
  main = "Scree Plot of PCA",
  ylim = c(0, max(prop_var_explained) * 1.1)
)
abline(h = 0.05, lty = 2, col = "red")  # optional: threshold line
```




```r
#-------------------------------------------------------------------------------
# -----Cumulative plot-----
#-------------------------------------------------------------------------------
plot(
```

```
  cumsum(prop_var_explained),
  type = "b",
  pch = 19,
  xlab = "Principal Component",
  ylab = "Cumulative Variance Explained",
  main = "Cumulative Variance Explained"
)

abline(h = 0.8, lty = 2, col = "blue")   # 80% threshold
abline(h = 0.9, lty = 2, col = "red")    # 90% threshold
```




```
# 3. Take first two PCs
pca_df <- as.data.frame(pca_res$x[, 1:2])
colnames(pca_df) <- c("PC1", "PC2")

# 4. Add cluster labels and true class to the PCA dataframe
pca_df$KM9   <- urban_scaled$cluster_km9
pca_df$KM13  <- urban_scaled$cluster_km13
pca_df$HC9   <- urban_scaled$cluster_hc9
pca_df$HC13  <- urban_scaled$cluster_hc13
pca_df$Class <- urban_scaled$class

#-------------------------------------------------------------------------------
# Variance in first 2 PCs
#-------------------------------------------------------------------------------
```

```r
# Variance of each component
var_explained <- pca_res$sdev^2

# Proportion of variance explained by each PC
prop_var_explained <- var_explained / sum(var_explained)

# PC1 variance
pc1_var <- prop_var_explained[1]

# PC2 variance
pc2_var <- prop_var_explained[2]

# Combined variance (PC1 + PC2)
pc12_var <- pc1_var + pc2_var

# Print results
pc1_var
```

```
## [1] 0.3447412
```

```r
pc2_var
```

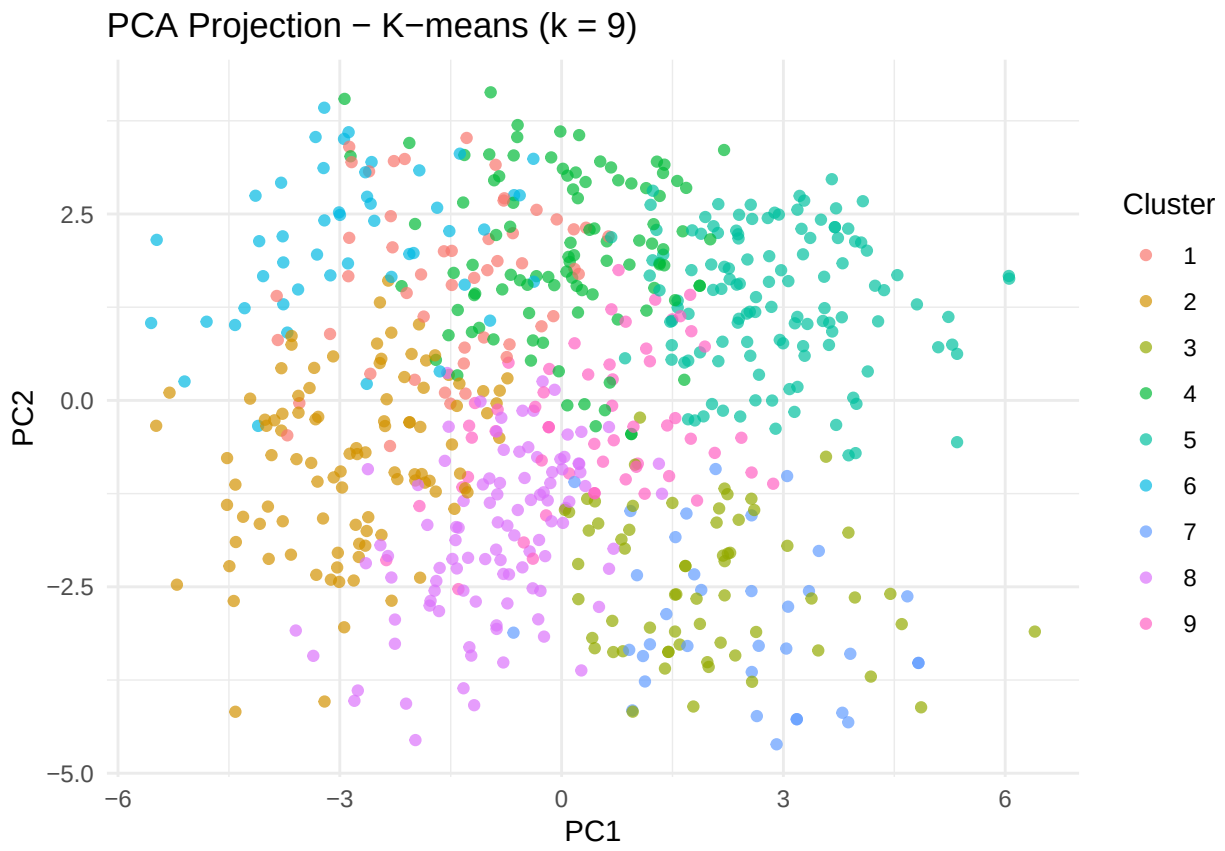
```
## [1] 0.2547235
```

```r
pc12_var
```

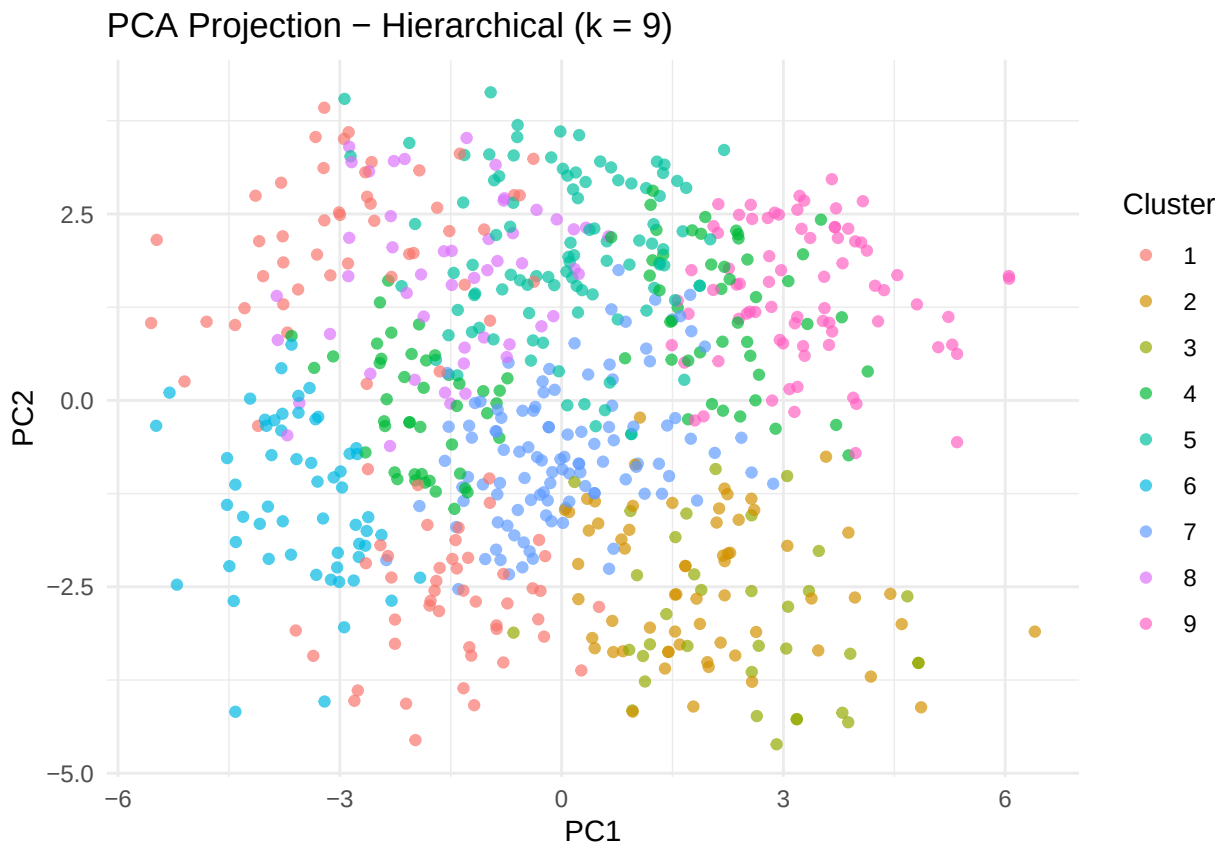
```
## [1] 0.5994647
```

Plot PCA by clusterings

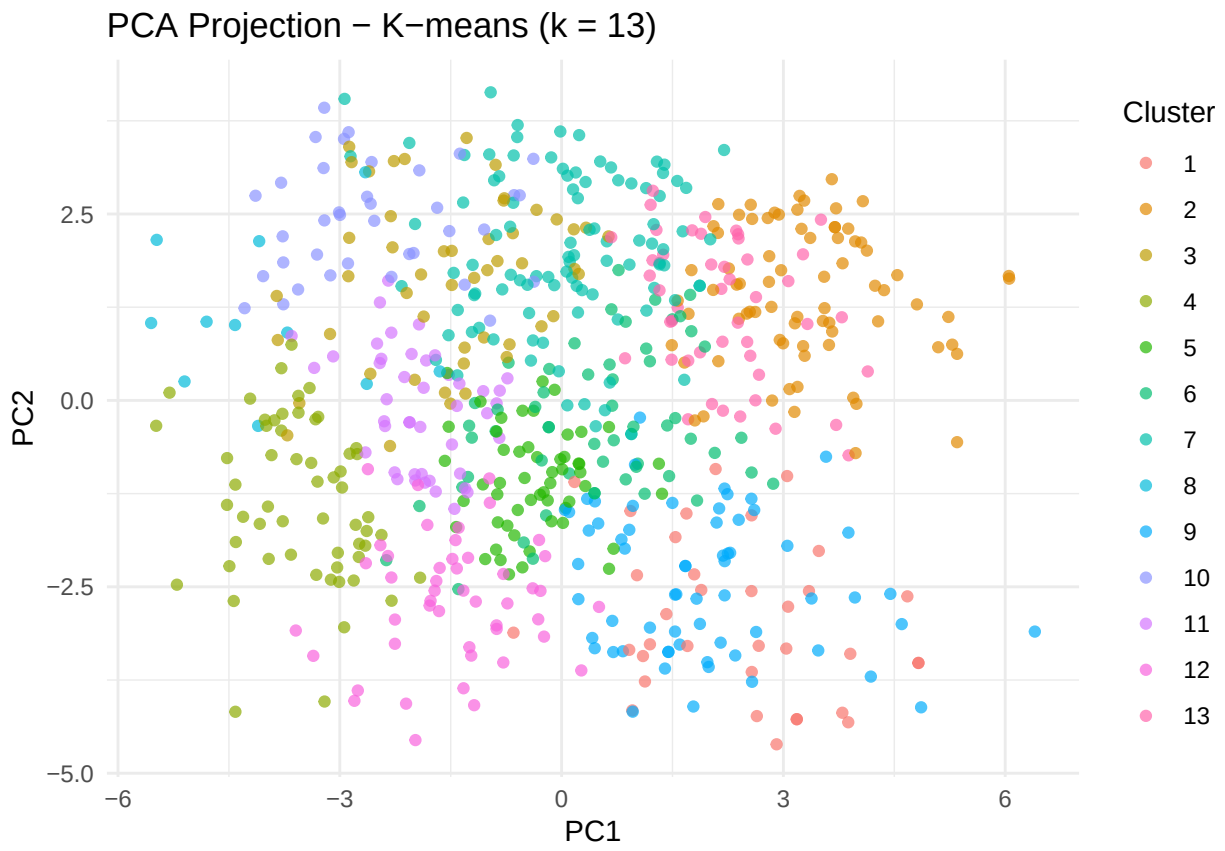
```r
# Function to plot PCA colored by any clustering
plot_pca_clusters <- function(df, cluster_col, title) {
  ggplot(df, aes(x = PC1, y = PC2, color = .data[[cluster_col]])) +
    geom_point(alpha = 0.7, size = 1.5) +
    labs(
      title = title,
      color = "Cluster"
    ) +
    theme_minimal()
}

# K-means with k = 9
plot_pca_clusters(pca_df, "KM9",  "PCA Projection - K-means (k = 9)")
```



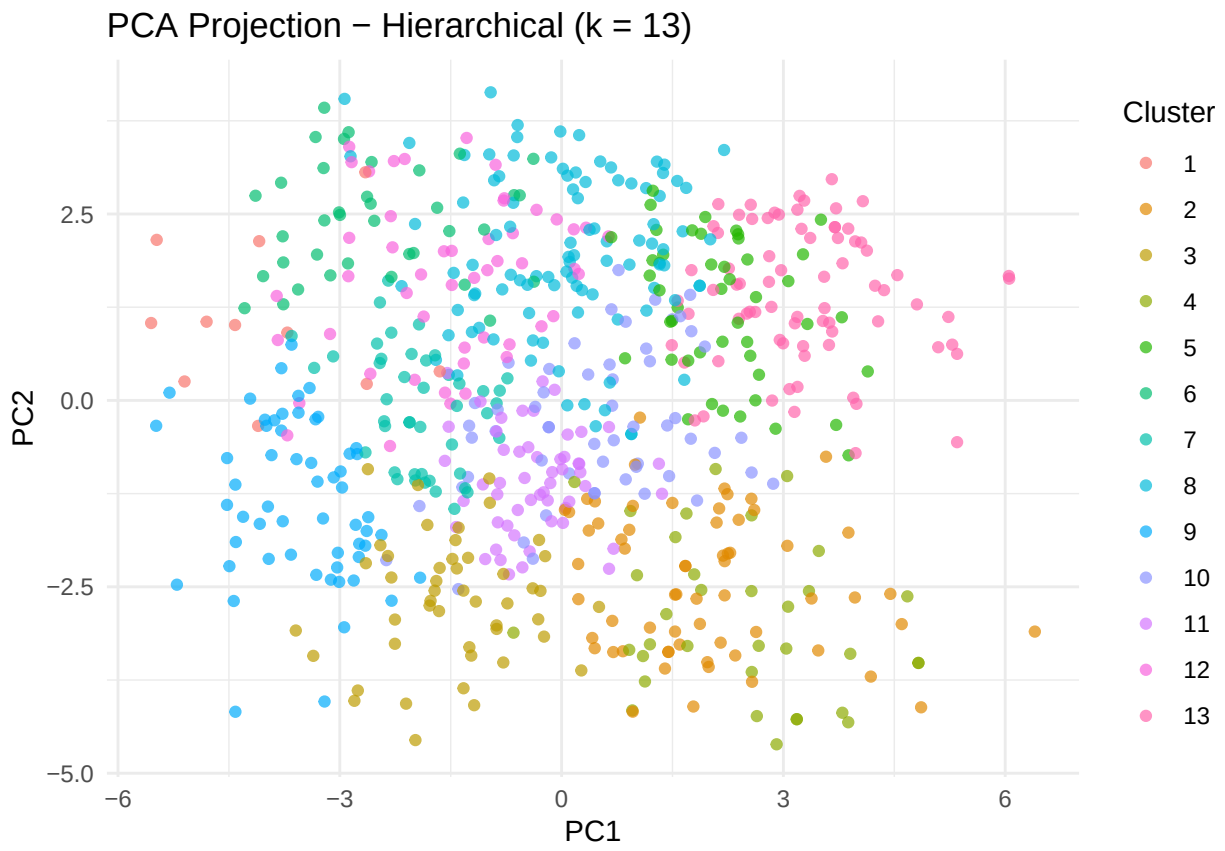

```r
# Hierarchical with k = 9
plot_pca_clusters(pca_df, "HC9",  "PCA Projection - Hierarchical (k = 9)")
```



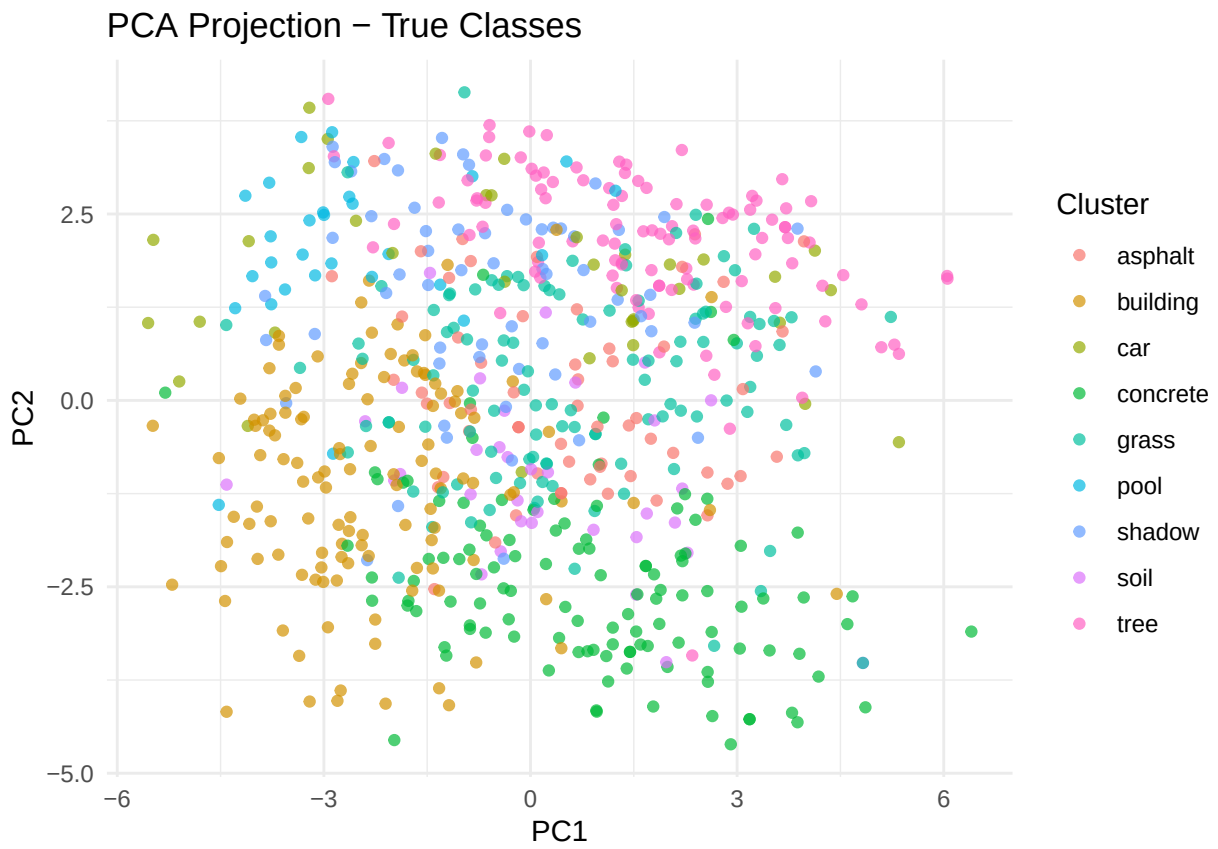

```r
# K-means with k = 13 (optimal)
plot_pca_clusters(pca_df, "KM13", "PCA Projection - K-means (k = 13)")
```




```
# Hierarchical with k = 13 (optimal)
plot_pca_clusters(pca_df, "HC13", "PCA Projection - Hierarchical (k = 13)")
```



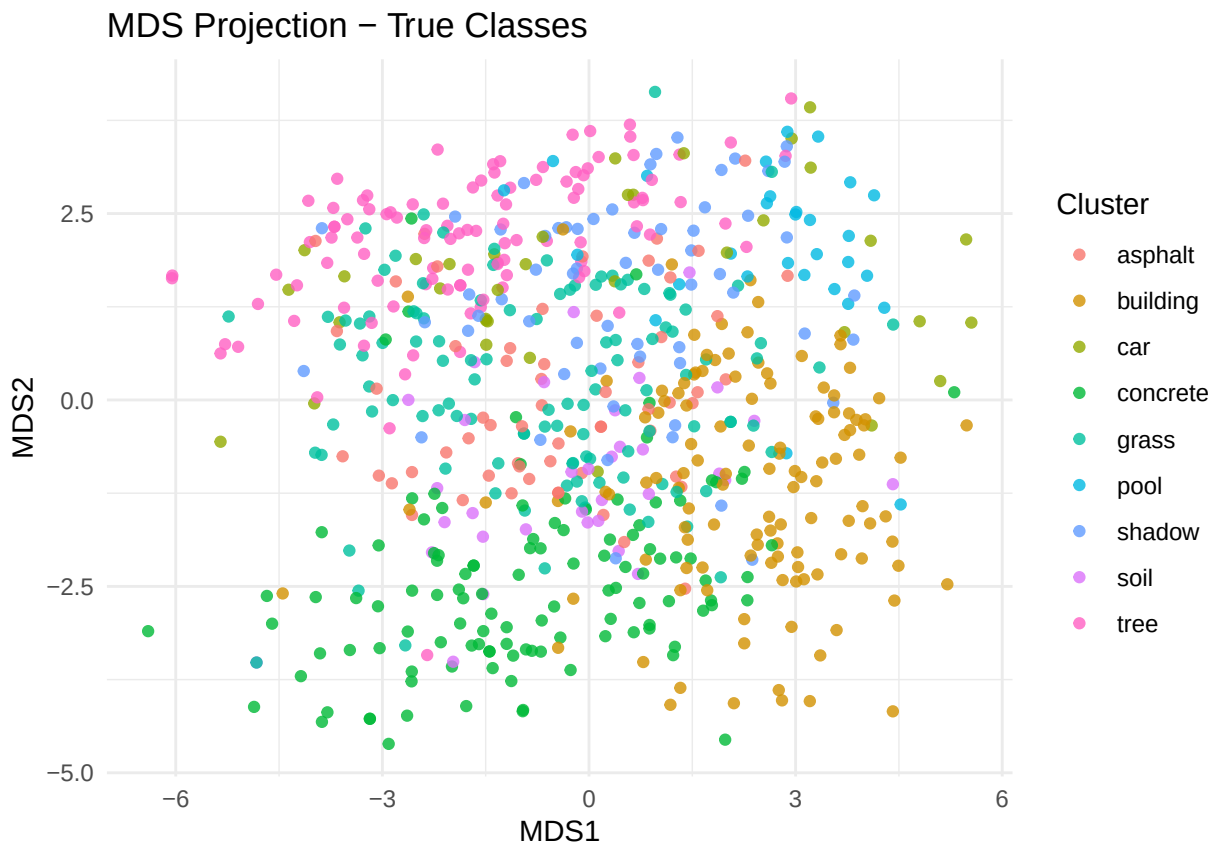

```
# Optional: color by TRUE class labels
plot_pca_clusters(pca_df, "Class", "PCA Projection - True Classes")
```



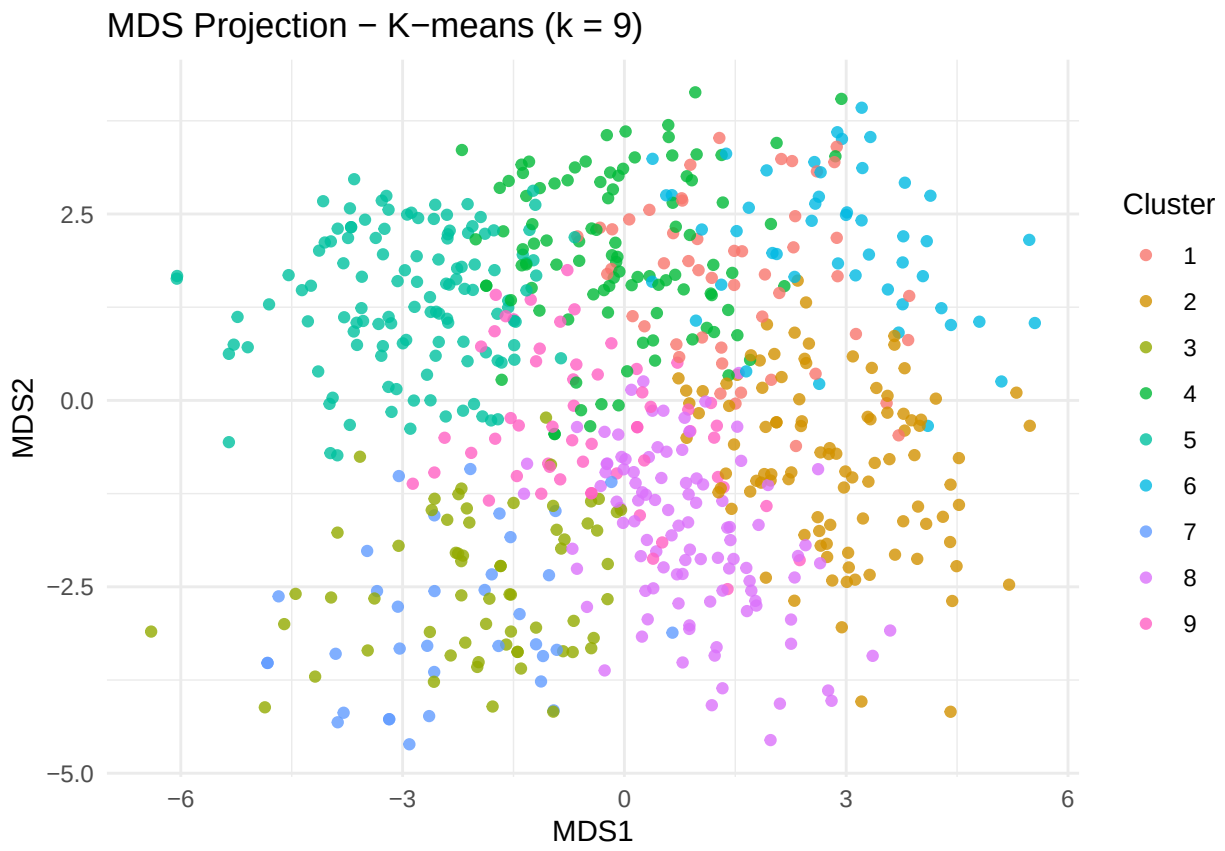
MDS visualization - See if we can get better look at clusters

```
library(dplyr)
library(ggplot2)

# 1. Use only scaled numeric variables
urban_num <- urban_scaled %>%
  dplyr::select(where(is.numeric))

# 2. Euclidean distance
dist_mat <- dist(urban_num, method = "euclidean")

# 3. Classical MDS (2D)
mds_res <- cmdscale(dist_mat, k = 2, eig = TRUE)

# 4. Build MDS dataframe with all labelings
mds_df <- data.frame(
  MDS1   = mds_res$points[, 1],
  MDS2   = mds_res$points[, 2],
  class  = urban_scaled$class,
  km13   = urban_scaled$cluster_km13,
  km9    = urban_scaled$cluster_km9,
  hc13   = urban_scaled$cluster_hc13,
  hc9    = urban_scaled$cluster_hc9
)

# 5. Helper function to plot MDS colored by any label column
```

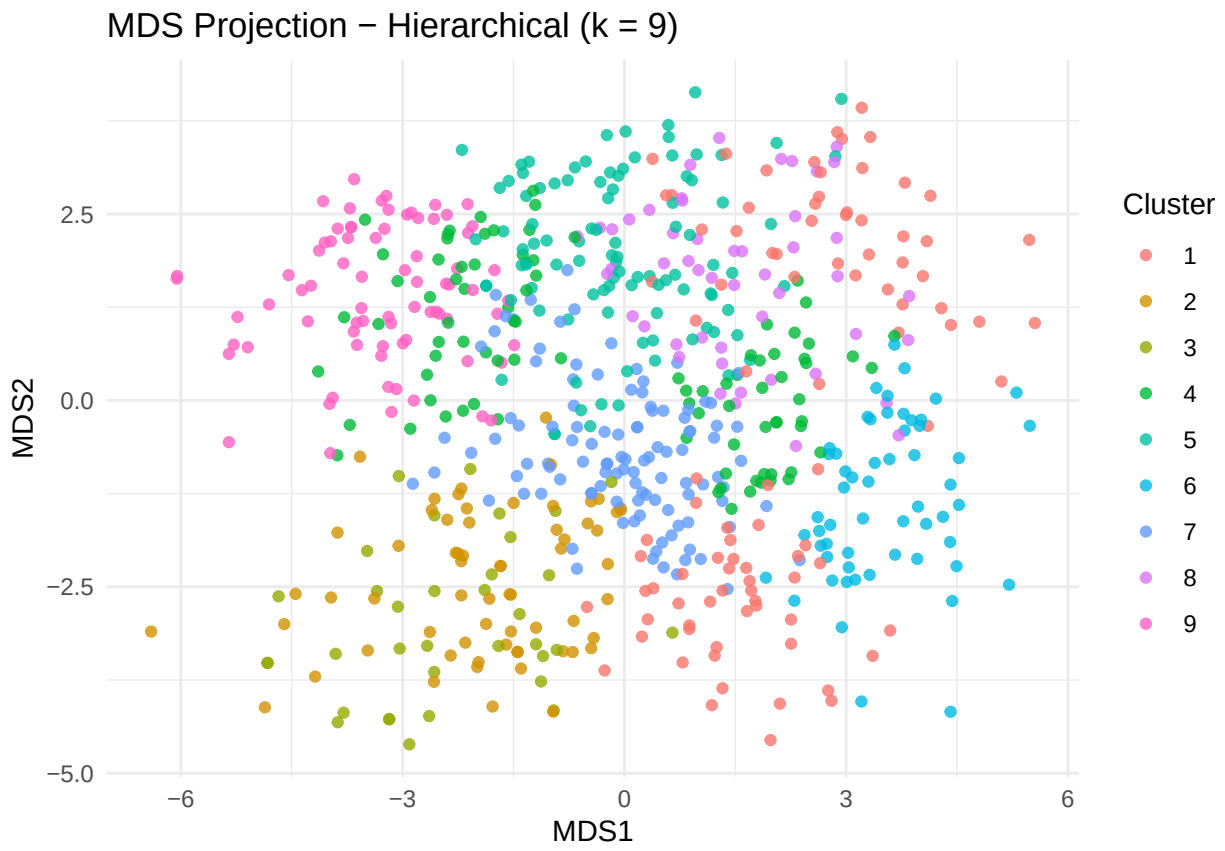
```r
plot_mds_clusters <- function(df, label_col, title) {
  ggplot(df, aes(x = MDS1, y = MDS2, color = .data[[label_col]])) +
    geom_point(alpha = 0.8, size = 1.5) +
    labs(title = title, color = "Cluster") +
    theme_minimal()
}

# 6. Make the 5 plots

# True classes
plot_mds_clusters(mds_df, "class", "MDS Projection - True Classes")
```



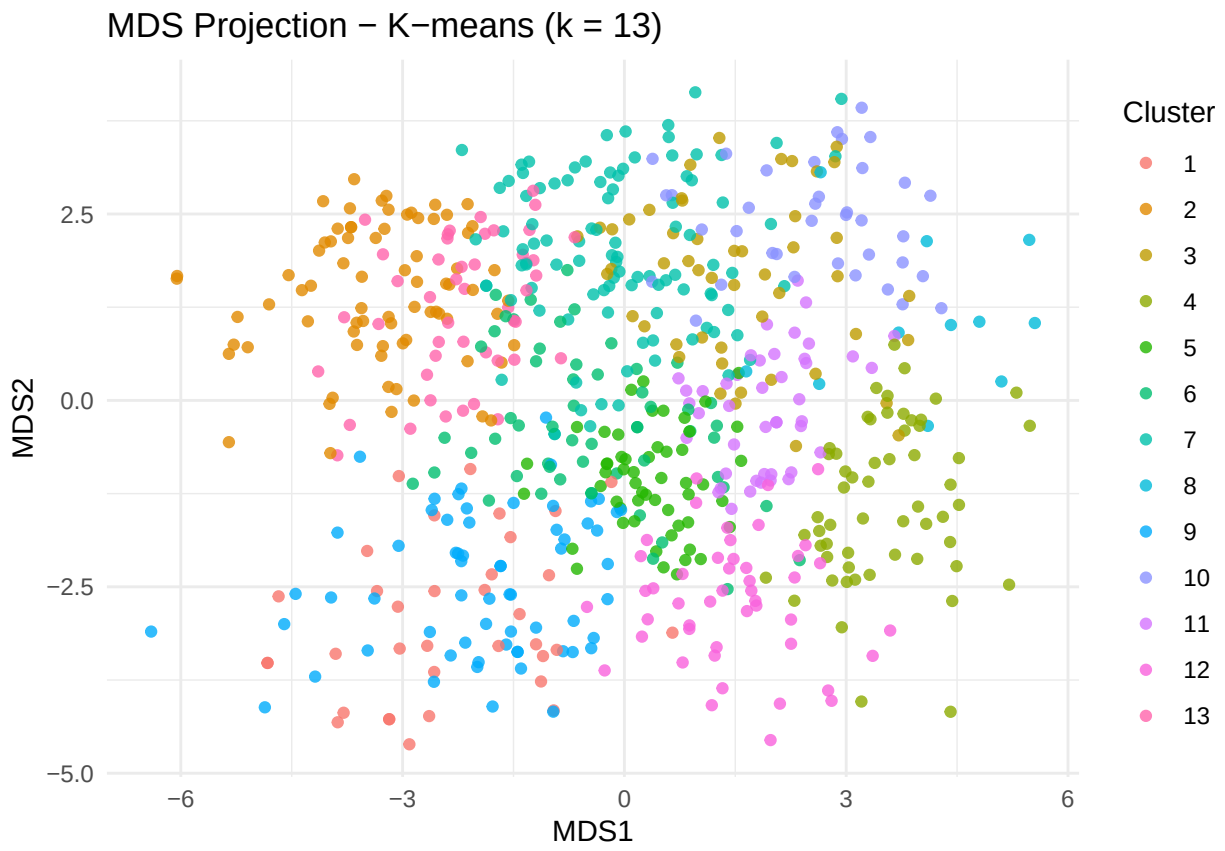

```r
# K-means k = 9
plot_mds_clusters(mds_df, "km9",   "MDS Projection - K-means (k = 9)")
```



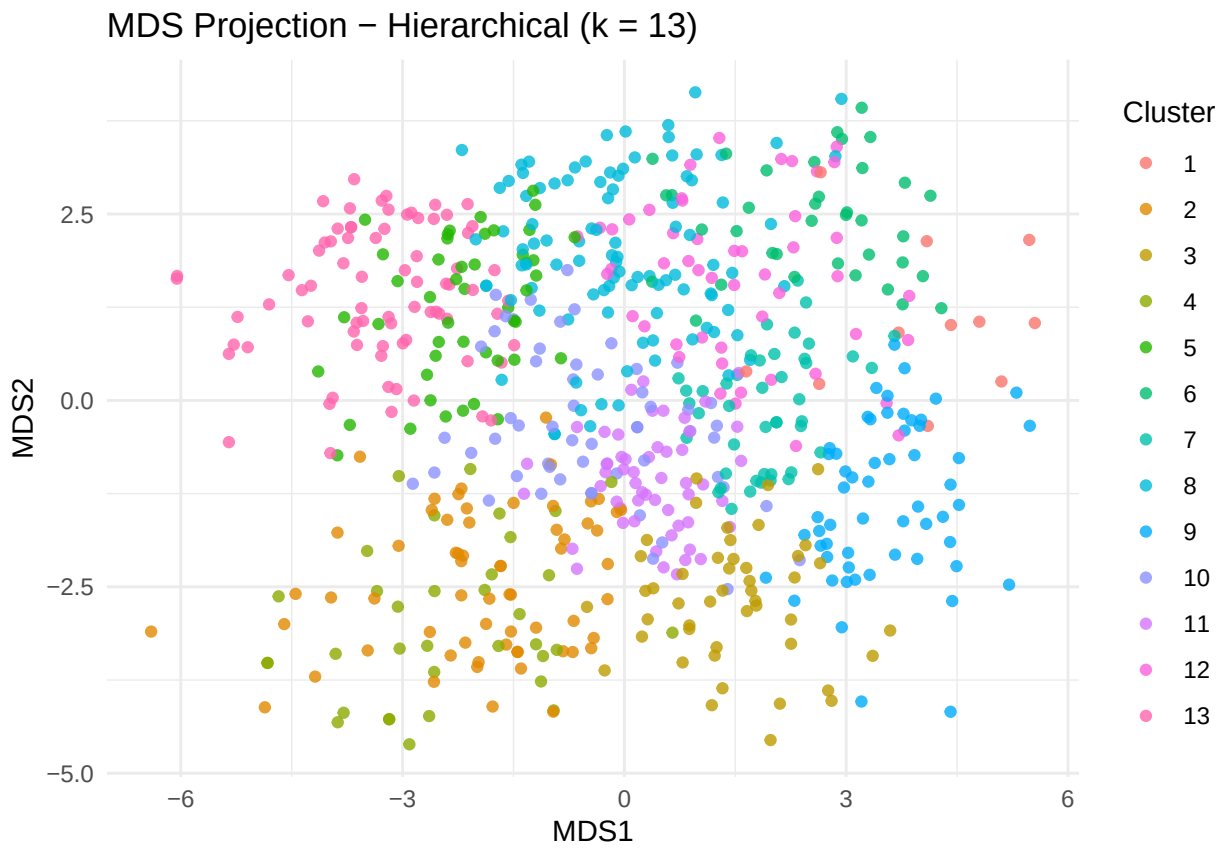

```
# Hierarchical k = 9
plot_mds_clusters(mds_df, "hc9",   "MDS Projection - Hierarchical (k = 9)")
```




```r
# K-means k = 13
plot_mds_clusters(mds_df, "km13",  "MDS Projection - K-means (k = 13)")
```




```
# Hierarchical k = 13
plot_mds_clusters(mds_df, "hc13",  "MDS Projection - Hierarchical (k = 13)")
```



6. Classification

k-NN

```
# Install if needed:
# install.packages("caret")
# install.packages("e1071")  # caret dependency for some models

library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

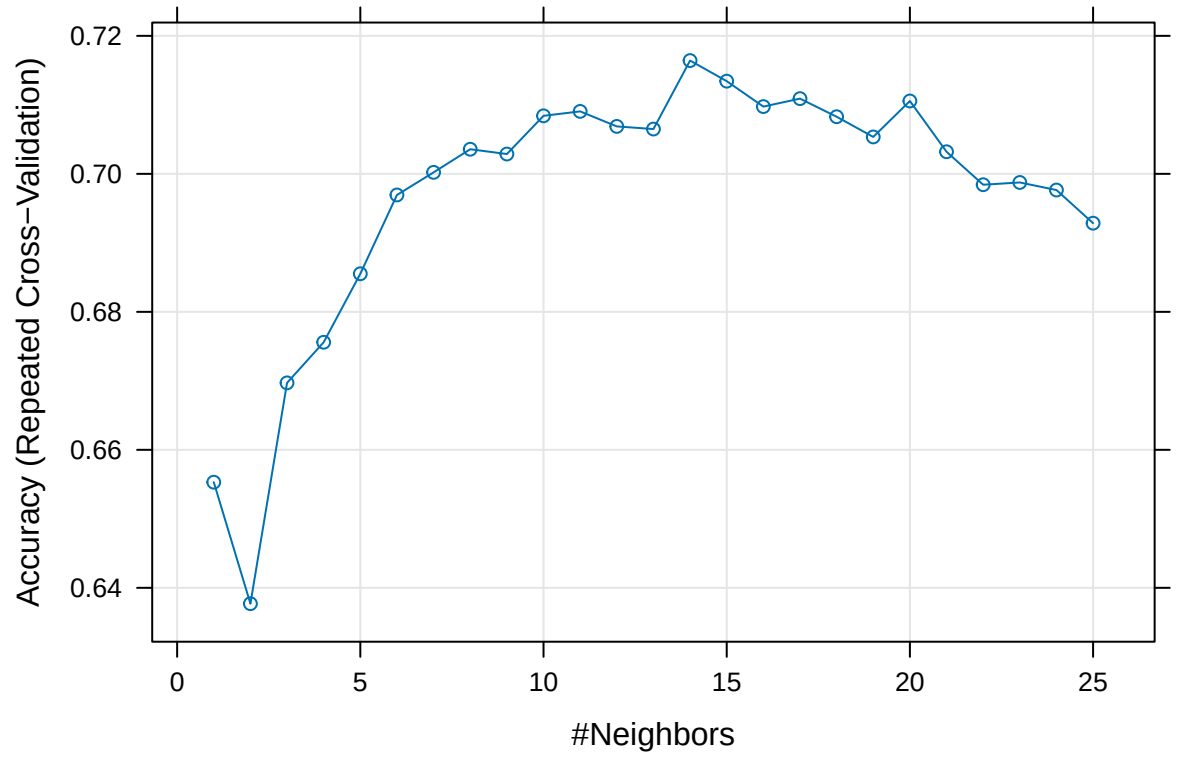
```
#Remove the cluster columns we added for clustering analysis
urban_scaled <- urban_scaled %>%
  select(-starts_with("cluster_"))
```

```r
set.seed(7378294)  # for reproducibility

## 1. Train/Test Split ----
# 80% train, 20% test, stratified by class
train_index <- createDataPartition(urban_scaled$class, p = 0.8, list = FALSE)

train_data <- urban_scaled[train_index, ]
test_data  <- urban_scaled[-train_index, ]

## 2. Set up cross-validation ----
ctrl <- trainControl(
  method  = "repeatedcv",  # repeated cross-validation
  number  = 5,             # 5-fold CV
  repeats = 5              # repeat 5 times
)

## 3. Define k grid to search over ----
k_grid <- expand.grid(k = 1:25)   # you can narrow later if you want

## 4. Train kNN classifier ----
knn_model <- train(
  class ~ .,            # predict class using all other columns
  data      = train_data,
  method    = "knn",
  tuneGrid  = k_grid,
  trControl = ctrl
)

## 5. Inspect best k and CV performance ----
knn_model           # prints accuracy for each k and best k
```

```
## k-Nearest Neighbors
##
## 544 samples
##  16 predictor
##   9 classes: 'asphalt ', 'building ', 'car ', 'concrete ', 'grass ', 'pool ', 'shadow ', 'soil ', 't
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 434, 434, 436, 435, 437, 435, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.6553071  0.6010072
##    2  0.6377006  0.5801186
##    3  0.6697044  0.6162042
##    4  0.6755908  0.6228856
##    5  0.6855162  0.6338546
##    6  0.6969372  0.6467601
##    7  0.7002170  0.6504296
##    8  0.7035649  0.6541741
##    9  0.7028703  0.6532009
##   10  0.7084095  0.6594188
```

```
##    11  0.7090526  0.6599715
##    12  0.7068746  0.6570456
##    13  0.7064910  0.6565358
##    14  0.7164195  0.6681328
##    15  0.7134335  0.6645881
##    16  0.7097463  0.6601357
##    17  0.7109017  0.6614547
##    18  0.7082787  0.6582052
##    19  0.7053497  0.6547181
##    20  0.7105579  0.6607083
##    21  0.7032048  0.6520353
##    22  0.6984232  0.6462578
##    23  0.6987528  0.6466018
##    24  0.6976525  0.6453969
##    25  0.6928509  0.6396830
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 14.
```

```r
plot(knn_model)      # accuracy vs k
```




```r
## 6. Predict on test set ----
knn_pred <- predict(knn_model, newdata = test_data)

## 7. Evaluate test performance ----
```

```r
# Ensure predicted outputs have the same levels as the reference
# Make sure reference is a factor
test_data$class <- factor(test_data$class)

# Force predictions to use the same levels as the reference
knn_pred <- factor(knn_pred, levels = levels(test_data$class))

# Now compute confusion matrix
confusionMatrix(knn_pred, test_data$class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  asphalt  building  car  concrete  grass  pool  shadow  soil  tree
##   asphalt          8         0    0         0      0     0       1     0     0
##   building         0        21    0         1      0     0       0     1     0
##   car              0         1    5         1      0     0       0     0     0
##   concrete         1         2    0        21      1     0       0     0     0
##   grass            0         0    0         0     16     0       0     1     2
##   pool             0         0    1         0      0     5       1     0     0
##   shadow           1         0    0         0      0     0       9     0     2
##   soil             0         0    0         0      1     0       0     4     0
##   tree             1         0    1         0      4     0       1     0    17
##
## Overall Statistics
##
##                Accuracy : 0.8092
##                  95% CI : (0.7313, 0.8725)
##     No Information Rate : 0.1832
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.778
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: asphalt  Class: building  Class: car
## Sensitivity                 0.72727           0.8750     0.71429
## Specificity                 0.99167           0.9813     0.98387
## Pos Pred Value              0.88889           0.9130     0.71429
## Neg Pred Value              0.97541           0.9722     0.98387
## Prevalence                  0.08397           0.1832     0.05344
## Detection Rate              0.06107           0.1603     0.03817
## Detection Prevalence        0.06870           0.1756     0.05344
## Balanced Accuracy           0.85947           0.9282     0.84908
##                      Class: concrete  Class: grass  Class: pool  Class: shadow
## Sensitivity                   0.9130        0.7273      1.00000         0.7500
## Specificity                   0.9630        0.9725      0.98413         0.9748
## Pos Pred Value                0.8400        0.8421      0.71429         0.7500
## Neg Pred Value                0.9811        0.9464      1.00000         0.9748
## Prevalence                    0.1756        0.1679      0.03817         0.0916
## Detection Rate                0.1603        0.1221      0.03817         0.0687
```

```
## Detection Prevalence           0.1908        0.1450      0.05344         0.0916
## Balanced Accuracy              0.9380        0.8499      0.99206         0.8624
##                       Class: soil  Class: tree
## Sensitivity               0.66667       0.8095
## Specificity               0.99200       0.9364
## Pos Pred Value            0.80000       0.7083
## Neg Pred Value            0.98413       0.9626
## Prevalence                0.04580       0.1603
## Detection Rate            0.03053       0.1298
## Detection Prevalence      0.03817       0.1832
## Balanced Accuracy         0.82933       0.8729
```

Random Forest

- Should help learn the complex decision boundaries

```
library(caret)
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# 1. Clean class labels: remove trailing/leading spaces
urban_scaled$class <- trimws(urban_scaled$class)

# 2. Convert class to factor
urban_scaled$class <- factor(urban_scaled$class)

# sanity check
str(urban_scaled$class)
```

```
##  Factor w/ 9 levels "asphalt","building",..: 3 4 4 4 4 9 3 3 2 9 ...
```

```
levels(urban_scaled$class)
```

```
## [1] "asphalt"  "building" "car"      "concrete" "grass"    "pool"     "shadow"
## [8] "soil"     "tree"
```

```r
table(urban_scaled$class)
```

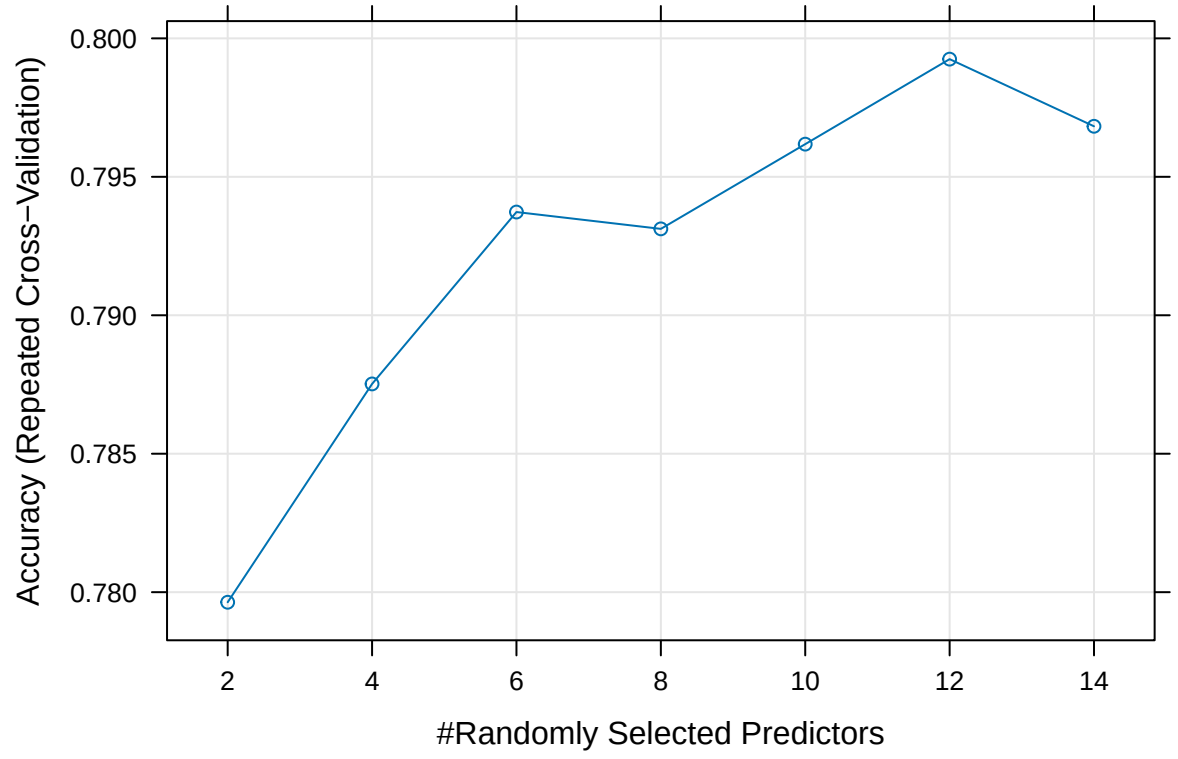
```
##
##  asphalt building      car concrete    grass     pool   shadow     soil
##       59      122       36      116      112       29       61       34
##     tree
##      106
```

```r
set.seed(372415)
#---------------------------------------------------------------------------
# Set Up Train/Test Split
#---------------------------------------------------------------------------
train_index <- createDataPartition(urban_scaled$class, p = 0.8, list = FALSE)

train_data <- urban_scaled[train_index, ]
test_data  <- urban_scaled[-train_index, ]

#---------------------------------------------------------------------------
# CV Training
#---------------------------------------------------------------------------
# Cross-validation setup (5 folds)
ctrl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 3
)

# Create tuning grid for mtry
mtry_grid <- expand.grid(mtry = c(2, 4, 6, 8, 10, 12, 14))

# Train RF
rf_model <- train(
  class ~ .,
  data = train_data,
  method = "rf",
  trControl = ctrl,
  tuneGrid = mtry_grid,
  ntree = 500,          # common choice
  importance = TRUE     # let RF compute feature importance
)

rf_model
```

```
## Random Forest
##
## 544 samples
##  16 predictor
##   9 classes: 'asphalt', 'building', 'car', 'concrete', 'grass', 'pool', 'shadow', 'soil', 'tree'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 433, 437, 435, 436, 435, 434, ...
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa
##    2    0.7796357  0.7431835
##    4    0.7875198  0.7528366
##    6    0.7937271  0.7602452
##    8    0.7931201  0.7596572
##   10    0.7961792  0.7633226
##   12    0.7992499  0.7669198
##   14    0.7968246  0.7641670
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 12.
```

```r
plot(rf_model)
```



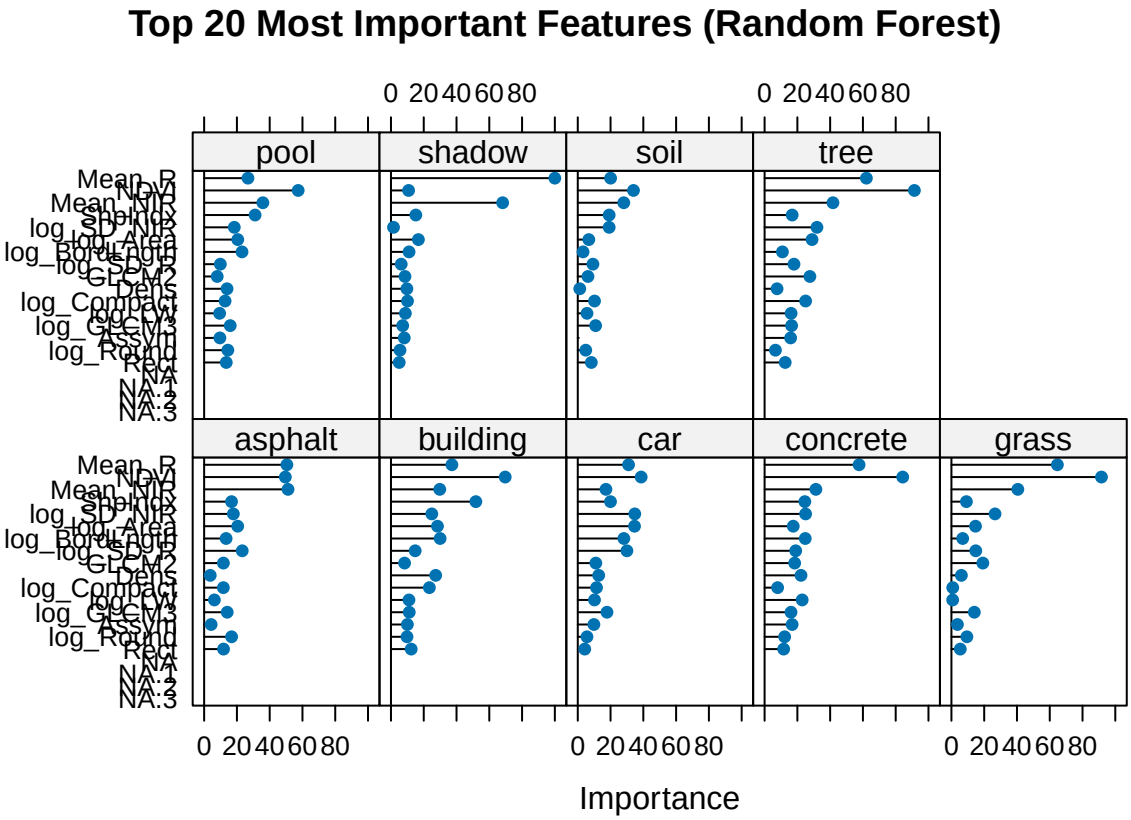

```r
#-------------------------------------------------------------------------------
# Predict on test set
#-------------------------------------------------------------------------------
# 1. Get predictions
rf_pred <- predict(rf_model, newdata = test_data)

# 2. Make sure the reference is a factor with the levels from training
train_levels <- levels(train_data$class)
test_data$class <- factor(test_data$class, levels = train_levels)
```

```r
# 3. Force predictions to use the same levels as well
rf_pred <- factor(rf_pred, levels = train_levels)

# 4. Now this should work
confusionMatrix(rf_pred, test_data$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction asphalt building car concrete grass pool shadow soil tree
##   asphalt       11        0   0        0     0    0      0    0    0
##   building       0       21   0        0     0    0      0    0    0
##   car            0        0   6        0     1    0      0    0    0
##   concrete       0        3   0       23     0    0      0    1    0
##   grass          0        0   1        0    14    0      0    0    3
##   pool           0        0   0        0     0    5      0    0    0
##   shadow         0        0   0        0     0    0     11    0    3
##   soil           0        0   0        0     1    0      0    5    0
##   tree           0        0   0        0     6    0      1    0   15
##
## Overall Statistics
##
##                Accuracy : 0.8473
##                  95% CI : (0.7741, 0.9042)
##     No Information Rate : 0.1832
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8227
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: asphalt Class: building Class: car Class: concrete
## Sensitivity                 1.00000          0.8750    0.85714          1.0000
## Specificity                 1.00000          1.0000    0.99194          0.9630
## Pos Pred Value              1.00000          1.0000    0.85714          0.8519
## Neg Pred Value              1.00000          0.9727    0.99194          1.0000
## Prevalence                  0.08397          0.1832    0.05344          0.1756
## Detection Rate              0.08397          0.1603    0.04580          0.1756
## Detection Prevalence        0.08397          0.1603    0.05344          0.2061
## Balanced Accuracy           1.00000          0.9375    0.92454          0.9815
##                      Class: grass Class: pool Class: shadow Class: soil
## Sensitivity                0.6364     1.00000       0.91667     0.83333
## Specificity                0.9633     1.00000       0.97479     0.99200
## Pos Pred Value             0.7778     1.00000       0.78571     0.83333
## Neg Pred Value             0.9292     1.00000       0.99145     0.99200
## Prevalence                 0.1679     0.03817       0.09160     0.04580
## Detection Rate             0.1069     0.03817       0.08397     0.03817
## Detection Prevalence       0.1374     0.03817       0.10687     0.04580
## Balanced Accuracy          0.7998     1.00000       0.94573     0.91267
##                      Class: tree
## Sensitivity               0.7143
```

```
## Specificity                 0.9364
## Pos Pred Value              0.6818
## Neg Pred Value              0.9450
## Prevalence                  0.1603
## Detection Rate              0.1145
## Detection Prevalence        0.1679
## Balanced Accuracy           0.8253
```

```r
#-------------------------------------------------------------------------------
# Feature Importance
#-------------------------------------------------------------------------------
varImp_rf <- varImp(rf_model)
plot(varImp_rf, top = 20,
     main = "Top 20 Most Important Features (Random Forest)")
```



7. Evaluation

```r
library(dplyr)

urban_binary <- urban_scaled %>%
  mutate(
    veg_binary = case_when(
      class %in% c("tree", "grass", "soil") ~ "Vegetation",
```

```
      class %in% c("concrete", "asphalt", "building", "car", "pool", "shadow") ~ "NonVegetation",
      TRUE ~ NA_character_
    ),
    veg_binary = factor(veg_binary)
  )

table(urban_binary$veg_binary)
```

```
##
## NonVegetation    Vegetation
##           423           252
```

```
#Remove unnecesary cluster columns
urban_binary <- urban_binary %>%
  select(-starts_with("cluster_"))

#-------------------------------------------------------------------------------
# Split data
#-------------------------------------------------------------------------------
set.seed(134902)
train_index <- createDataPartition(urban_binary$veg_binary, p = 0.8, list = FALSE)

train_bin <- urban_binary[train_index, ]
test_bin  <- urban_binary[-train_index, ]


#-------------------------------------------------------------------------------
# RF on Binary Classification
#-------------------------------------------------------------------------------
library(randomForest)

set.seed(123)
rf_bin <- randomForest(
  veg_binary ~ . - class,   # use all predictors except original class
  data = train_bin,
  ntree = 500,
  importance = TRUE
)

rf_bin
```

```
##
## Call:
##  randomForest(formula = veg_binary ~ . - class, data = train_bin,      ntree = 500, importance = TRU
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 4.25%
## Confusion matrix:
##               NonVegetation Vegetation class.error
## NonVegetation           328         11  0.03244838
## Vegetation               12        190  0.05940594
```

```r
#-------------------------------------------------------------------------------
# Confusion Matrix
#-------------------------------------------------------------------------------
bin_pred <- predict(rf_bin, newdata = test_bin)

conf_mat_bin <- table(
  Prediction = bin_pred,
  Reference  = test_bin$veg_binary
)

conf_mat_bin
```

```
##                Reference
## Prediction      NonVegetation Vegetation
##   NonVegetation            82          7
##   Vegetation                2         43
```

```r
#-------------------------------------------------------------------------------
# Manual Precision and Recall Calculations
#-------------------------------------------------------------------------------
TP <- conf_mat_bin["Vegetation","Vegetation"]
FP <- conf_mat_bin["Vegetation","NonVegetation"]
FN <- conf_mat_bin["NonVegetation","Vegetation"]
TN <- conf_mat_bin["NonVegetation","NonVegetation"]

accuracy <- (TP + TN) / sum(conf_mat_bin)
precision <- TP / (TP + FP)
recall    <- TP / (TP + FN)
F1 <- 2 * (precision * recall) / (precision + recall)

accuracy
```

```
## [1] 0.9328358
```

```r
precision
```

```
## [1] 0.9555556
```

```r
recall
```

```
## [1] 0.86
```

```r
F1
```

```
## [1] 0.9052632
```

```r
metrics_df <- data.frame(
  Metric = c("Accuracy", "Precision", "Recall", "F1 Score"),
  Value  = c(accuracy, precision, recall, F1)
)
```

```r
# Print nicely with 3 decimal places
metrics_df$Value <- round(metrics_df$Value, 3)

metrics_df
```

```
##      Metric Value
## 1  Accuracy 0.933
## 2 Precision 0.956
## 3    Recall 0.860
## 4  F1 Score 0.905
```

```r
#-------------------------------------------------------------------------------
# ROC Curve and AUC (1 - Specificity on x-axis)
#-------------------------------------------------------------------------------
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

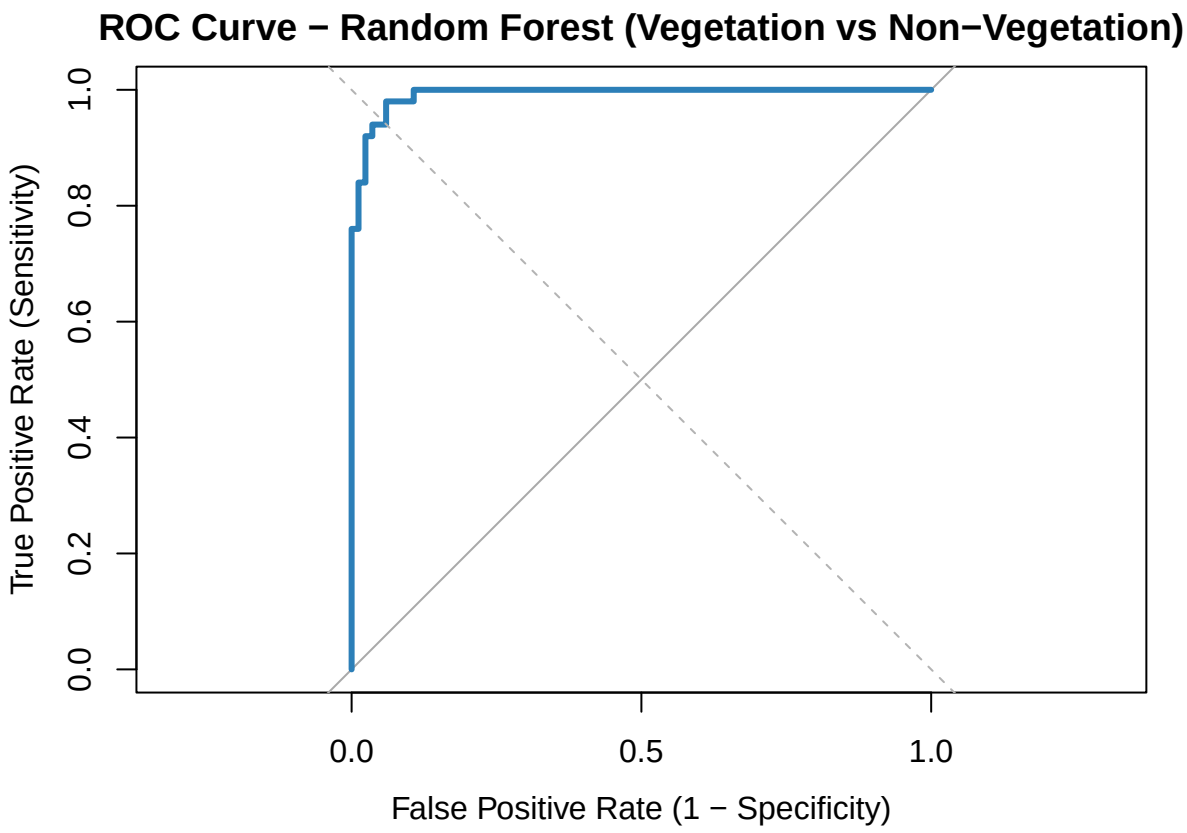
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# get probability for class "Vegetation"
bin_prob <- predict(rf_bin, newdata = test_bin, type = "prob")[, "Vegetation"]

roc_obj <- roc(test_bin$veg_binary, bin_prob)
```

```
## Setting levels: control = NonVegetation, case = Vegetation
```

```
## Setting direction: controls < cases
```

```r
plot(
  roc_obj,
  legacy.axes = TRUE,                        # x-axis = 1 - specificity
  col         = "#2c7fb8",
  lwd         = 3,
  main        = "ROC Curve - Random Forest (Vegetation vs Non-Vegetation)",
  xlab        = "False Positive Rate (1 - Specificity)",
  ylab        = "True Positive Rate (Sensitivity)"
)

abline(a = 0, b = 1, lty = 2, col = "gray70")  # chance line
```




```
# AUC
auc(roc_obj)
```

```
## Area under the curve: 0.9919
```