Traffic Code

Erin Xu

2025-11-14

Initial Setup

```r
library(tidyverse)
library(knitr)
library(ggplot2)
library(openxlsx)
library(dplyr)
library(ggfortify)
library(plotly)
library(corrplot)
library(patchwork)
library(pheatmap)
set.seed(25)

file <- "traffic.xlsx"
sheet_names <- getSheetNames(file)
print(sheet_names)
```

```
##  [1] "Loc1"  "Loc2"  "Loc3"  "Loc4"  "Loc5"  "Loc6"  "Loc7"  "Loc8"  "Loc9"
## [10] "Loc10" "Loc11" "Loc12" "Loc13" "Loc14" "Loc15" "Loc16" "Loc17" "Loc18"
## [19] "Loc19" "Loc20" "Loc21" "Loc22" "Loc23" "Loc24" "Loc25" "Loc26"
```

```r
num_sheets <- length(sheet_names)
```

Cleaning, summary stats, graphs by location

```r
cat("Loading all location data:\n\n")
```

```
## Loading all location data:
```

```r
# Store in list because dealing with 26 locations
data_list <- list()  # Original: time points × days
data_transposed <- list()  # Transposed: days × time points

expected_rows <- 288  # time points (5-min intervals over 24 hours)
expected_cols <- 384  # days

for (sheet in sheet_names){
  df <- read.xlsx(file, sheet = sheet)
  df_numeric <- df %>% select(where(is.numeric))
  # Check dimensions BEFORE storing
  actual_rows <- nrow(df_numeric)
  actual_cols <- ncol(df_numeric)
  cat("Location:", sheet, "\n")
  cat("  Dimensions:", actual_rows, "×", actual_cols)
  # Check if dimensions match expectations
  if (actual_rows != expected_rows || actual_cols != expected_cols) {
    cat("WARNING: Expected", expected_rows, "×", expected_cols, "\n")
    stop()
  } else {
    cat(" it's fine \n")
    }
  # Check for missing values
  n_missing <- sum(is.na(df_numeric))
  if (n_missing > 0) {
    cat("Missing values:", n_missing,)
    stop()
    }
  # Store data
  data_list[[sheet]] <- df_numeric
  data_transposed[[sheet]] <- as.data.frame(t(df_numeric))
  # Summary stats
  cat("Summary stats for ", sheet, "\n")
  print(summary(as.vector(as.matrix(data_transposed[[sheet]]))))
  cat("\n")
    }
```

```
## Location: Loc1
##   Dimensions: 288 × 384 it's fine
## Summary stats for  Loc1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      41     197     408     343     469     666
##
## Location: Loc2
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc2
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    31.0   185.0   447.0   360.8   506.0   709.0
##
## Location: Loc3
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc3
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    25.0   130.0   268.0   229.1   315.0   525.0
##
## Location: Loc4
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc4
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    33.0   187.0   415.0   335.9   464.0   613.0
##
## Location: Loc5
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc5
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    40.0   167.0   355.0   293.6   403.0   534.0
##
## Location: Loc6
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc6
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      37     160     380     310     437     601
##
## Location: Loc7
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc7
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0   105.0   238.0   209.9   296.0   514.0
##
## Location: Loc8
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc8
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    47.0   172.0   361.0   304.9   422.0   577.0
##
## Location: Loc9
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc9
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    38.0   169.0   374.0   314.4   437.0   650.0
##
## Location: Loc10
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc10
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    28.0   105.0   229.0   198.5   275.0   449.0
##
## Location: Loc11
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc11
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      30      94     163     150     192     380
##
## Location: Loc12
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc12
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0   135.0   309.0   262.2   378.0   526.0
##
## Location: Loc13
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc13
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    11.0    96.0   221.0   182.8   256.0   383.0
##
## Location: Loc14
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc14
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    24.0   157.0   355.0   293.8   414.0   560.0
##
## Location: Loc15
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc15
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    18.0   164.0   421.0   332.9   480.0   632.0
##
## Location: Loc16
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc16
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    20.0   144.0   365.0   288.3   414.0   543.0
##
## Location: Loc17
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc17
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    19.0   170.0   434.0   339.3   486.0   636.0
##
## Location: Loc18
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc18
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    15.0   154.0   340.0   277.2   386.0   525.0
##
## Location: Loc19
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc19
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    19.0    97.0   198.0   168.9   236.0   346.0
##
## Location: Loc20
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc20
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      21     167     432     341     482     683
##
## Location: Loc21
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc21
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      23     167     364     298     408     652
##
## Location: Loc22
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc22
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    24.0   181.0   445.0   354.7   491.0   751.0
##
## Location: Loc23
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc23
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    21.0   142.0   351.0   283.4   390.0   644.0
##
## Location: Loc24
##   Dimensions: 288 × 384 it’s fine
## Summary stats for  Loc24
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0   102.0   218.0   187.3   261.0   428.0
##
## Location: Loc25
##   Dimensions: 288 × 384 it's fine
## Summary stats for  Loc25
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    37.0   164.0   353.0   295.3   410.0   582.0
##
## Location: Loc26
##   Dimensions: 288 × 384 it's fine
## Summary stats for  Loc26
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0   117.0   271.0   233.5   337.0   540.0
```

```
cat("\n All data loaded successfully!\n")
```

```
##
##  All data loaded successfully!
```

```
cat("Total locations:", length(data_list), "\n\n")
```

```
## Total locations: 26
```

List Name	What it stores	How it's added
`data_list[[sheet]]`	Original data (rows = time points, columns = days)	`data_list[[sheet]] <- df_numeric`
`data_transposed[[sheet]]`	Transposed data (rows = days, columns = time points)	`data_transposed[[sheet]] <- as.data.frame(t(df_numeric))`

Stored as **named list elements** (key = sheet name)
Access like: `data_list$Loc01`, `data_transposed$Loc15`
Each one is a **data frame**

Paranoid

```
# Validation check for all sheets
mismatch_sheets <- c()

for (sheet in names(data_list)) {
  original <- as.matrix(data_list[[sheet]])
  recon <- t(as.matrix(data_transposed[[sheet]]))

  if (!isTRUE(all.equal(original, recon, check.attributes = FALSE))) {
    mismatch_sheets <- c(mismatch_sheets, sheet)
  }
}

if (length(mismatch_sheets) == 0) {
  message("All sheets match when transposed back.")
} else {
  message("Mismatch found in sheets: ", paste(mismatch_sheets, collapse = ", "))
}
```

```
## All sheets match when transposed back.
```

```
sheet <- names(data_list)[1]
identical(as.matrix(data_list[[sheet]]), t(as.matrix(data_transposed[[sheet]])))
```

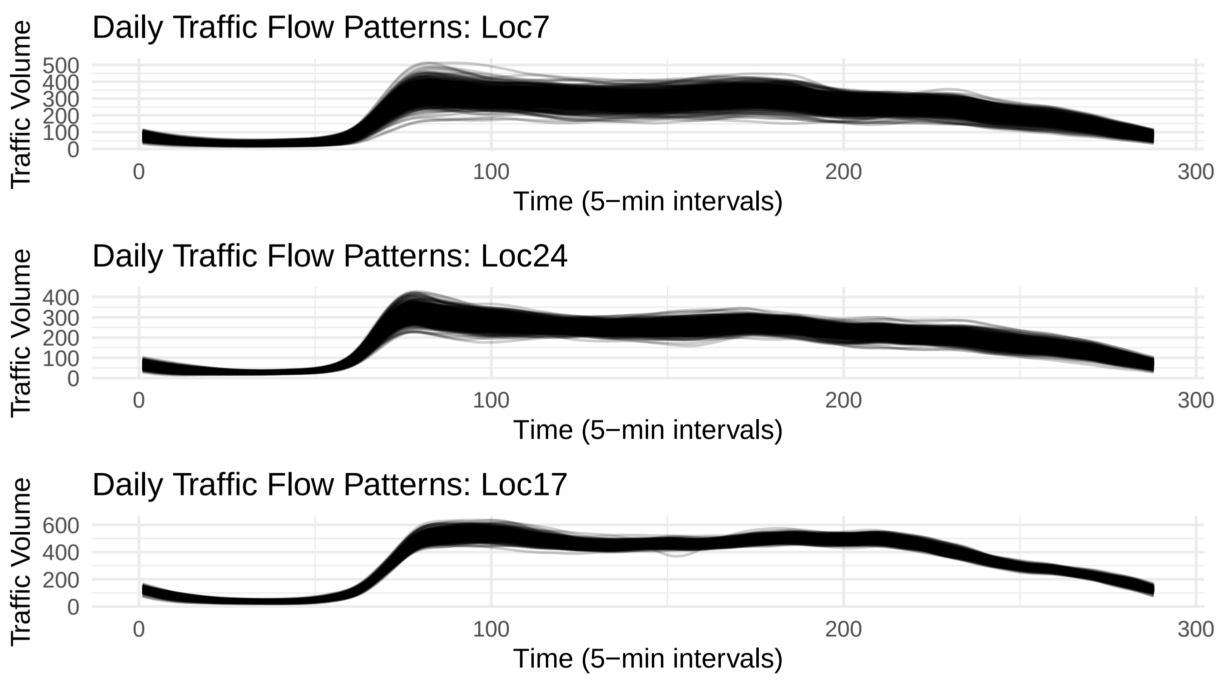
```
## [1] TRUE
```

Spaghetti Plots by Location

Show daily time series patterns by location. Clearly we do not ened to care about normality

```
sample_sheets <- sample(names(data_list), 3)
spaghettis_locations <- list()
for (sheet in sample_sheets) {

  #Spaghetti
  df_t <- data_transposed[[sheet]] %>%
    mutate(Day = 1:n()) %>%
    pivot_longer(cols = -Day, names_to = "Time", values_to = "Value") %>%
    mutate(Time = as.numeric(Time))

  p2 <- ggplot(df_t, aes(x = Time, y = Value, group = Day)) +
    geom_line(alpha = 0.2) +
    labs(title = paste("Daily Traffic Flow Patterns:", sheet),
         x = "Time (5-min intervals)", y = "Traffic Volume") +
    theme_minimal()

  spaghettis_locations[[sheet]] <- p2
```

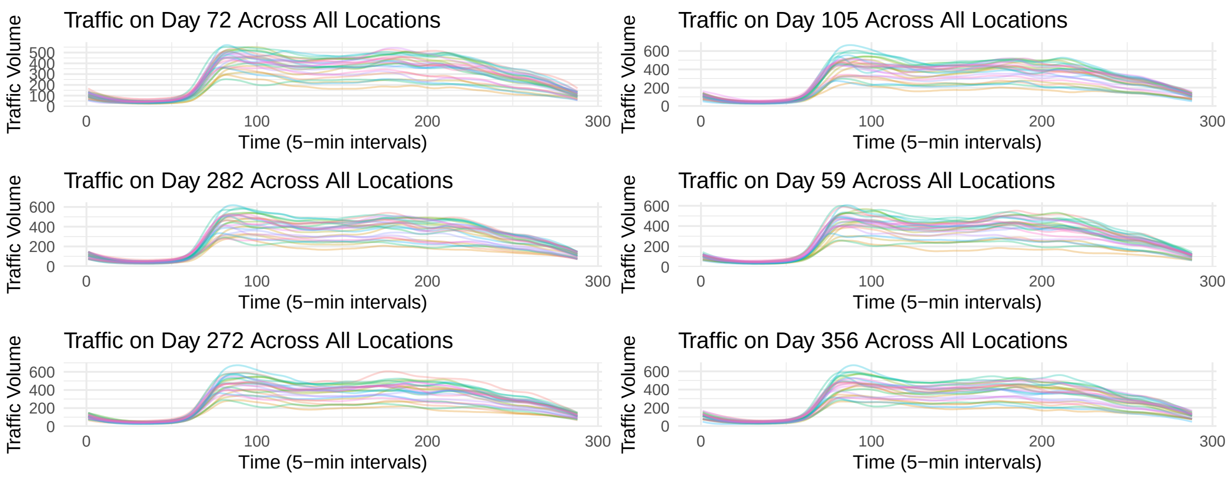
```r
}
print(wrap_plots(spaghettis_locations, ncol = 1))
```



Spaghetti Plots by Time

Show daily time series by day. I think this is more informative.

```r
random_days <- sample(1:384, 6)
spag_by_day <- list()

for (d in random_days) {
  # Collect data for this day from ALL locations
  df_day <- data.frame()

  for (sheet in names(data_list)) {
    df <- data_list[[sheet]]

    # Extract traffic values for day d across 288 timepoints
    temp <- data.frame(
      Time = 1:nrow(df),
      Value = df[, d],
      Location = sheet
    )
    df_day <- rbind(df_day, temp)
  }

  # Build spaghetti plot for this day (across all locations)
  p <- ggplot(df_day, aes(x = Time, y = Value, group = Location, color = Location)) +
    geom_line(alpha = 0.3) +
    labs(title = paste("Traffic on Day", d, "Across All Locations"),
         x = "Time (5-min intervals)", y = "Traffic Volume") +
    theme_minimal() +
    theme(legend.position = "none")   # Remove cluttered legend

  spag_by_day[[paste0("Day_", d)]] <- p
}

# Display all 3 spaghetti plots (patchwork)
wrap_plots(spag_by_day, ncol = 2)
```

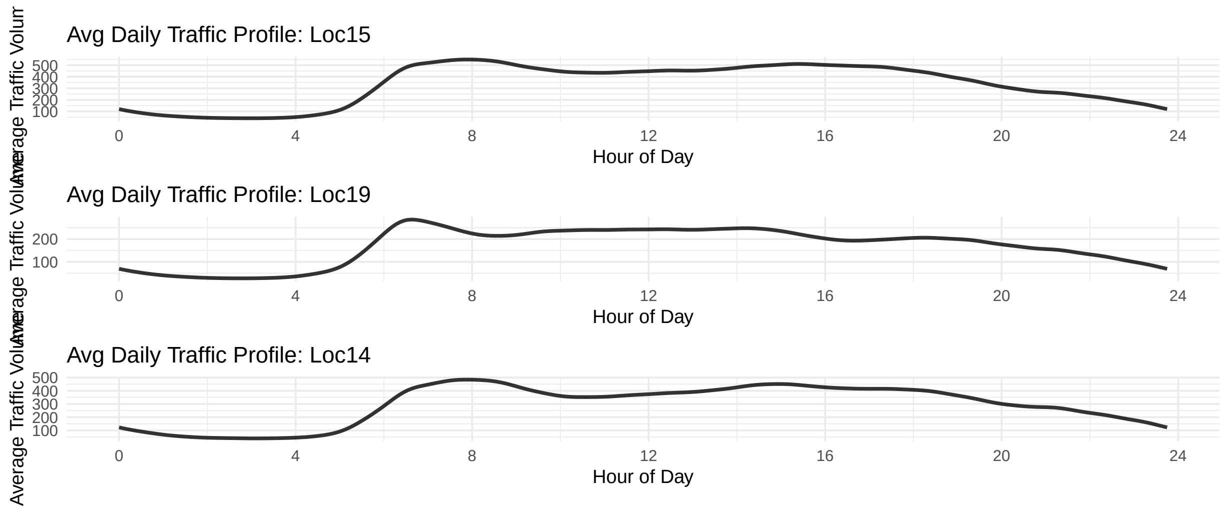


## Average daily profile, by location

```
set.seed(123)  # for reproducibility
sample_locs <- sample(names(data_list), 3)
profile_plots <- list()
time_labels <- seq(0, 23.75, length.out = 288)


for (sheet in sample_locs) {
  df <- data_transposed[[sheet]]
  avg_day <- colMeans(df, na.rm = TRUE)

  df_avg <- data.frame(Time = time_labels, AvgVolume = avg_day)
  p <- ggplot(df_avg, aes(x = Time, y = AvgVolume)) +
    geom_line(size = 1, alpha = 0.8) +
    labs(title = paste("Avg Daily Traffic Profile:", sheet),
         x = "Hour of Day", y = "Average Traffic Volume") +
    scale_x_continuous(breaks = seq(0, 24, by = 4)) +
    theme_minimal()

  profile_plots[[sheet]] <- p
}
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
print(wrap_plots(profile_plots, ncol = 1))
```



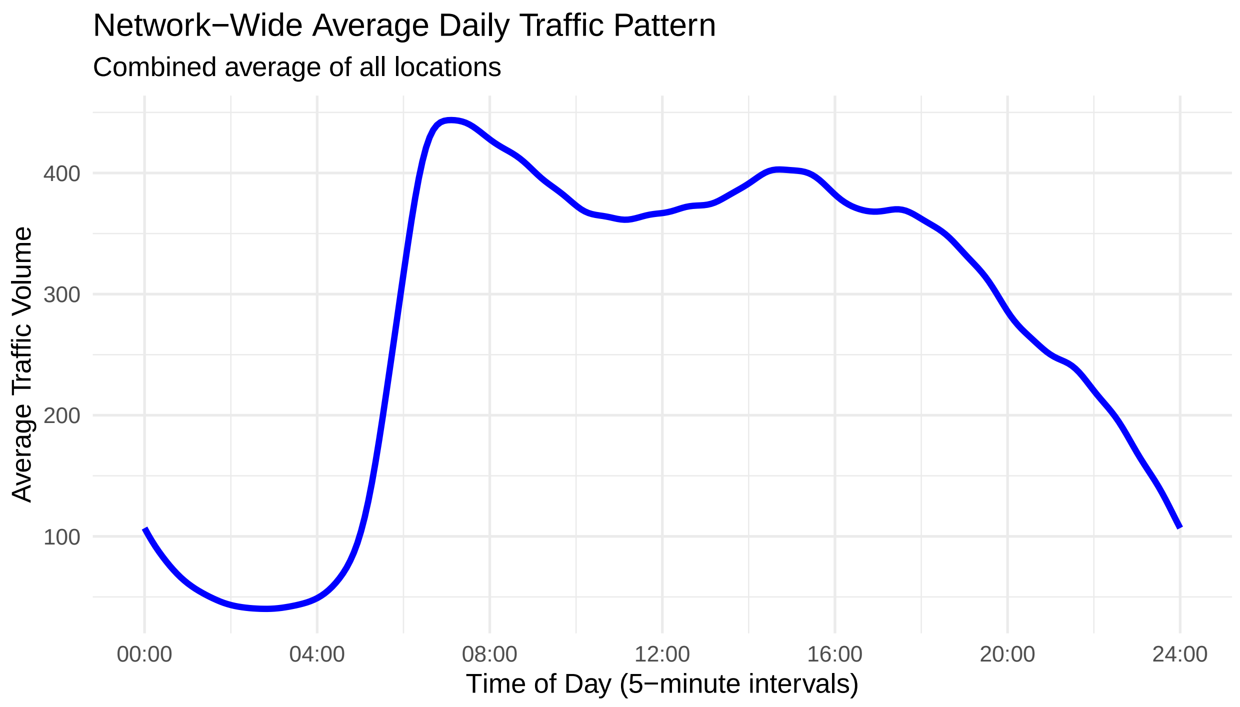
```
for (sheet in names(data_transposed)) {
  df <- data_transposed[[sheet]]
  cat(sheet, ": ", ncol(df), " time points\n")
}
```

```
## Loc1 :  288  time points
## Loc2 :  288  time points
## Loc3 :  288  time points
## Loc4 :  288  time points
## Loc5 :  288  time points
## Loc6 :  288  time points
## Loc7 :  288  time points
## Loc8 :  288  time points
## Loc9 :  288  time points
## Loc10 :  288  time points
## Loc11 :  288  time points
## Loc12 :  288  time points
## Loc13 :  288  time points
## Loc14 :  288  time points
## Loc15 :  288  time points
## Loc16 :  288  time points
## Loc17 :  288  time points
## Loc18 :  288  time points
## Loc19 :  288  time points
## Loc20 :  288  time points
## Loc21 :  288  time points
## Loc22 :  288  time points
## Loc23 :  288  time points
## Loc24 :  288  time points
## Loc25 :  288  time points
## Loc26 :  288  time points
```

Average daily profile by entire network

```r
time_labels <- format(
  seq(
    from = as.POSIXct("00:00", format="%H:%M"),
    by = "5 min",
    length.out = 288
  ),
  "%H:%M"
)

# Initialize dataframe
network_avg <- data.frame(Time = time_labels)

# Compute average profile for each location
for (sheet in names(data_transposed)) {
  df <- data_transposed[[sheet]]
  avg_day <- colMeans(df, na.rm = TRUE)     # 288-length vector
  network_avg[[sheet]] <- avg_day
}

# Compute overall network-level average across all locations
```

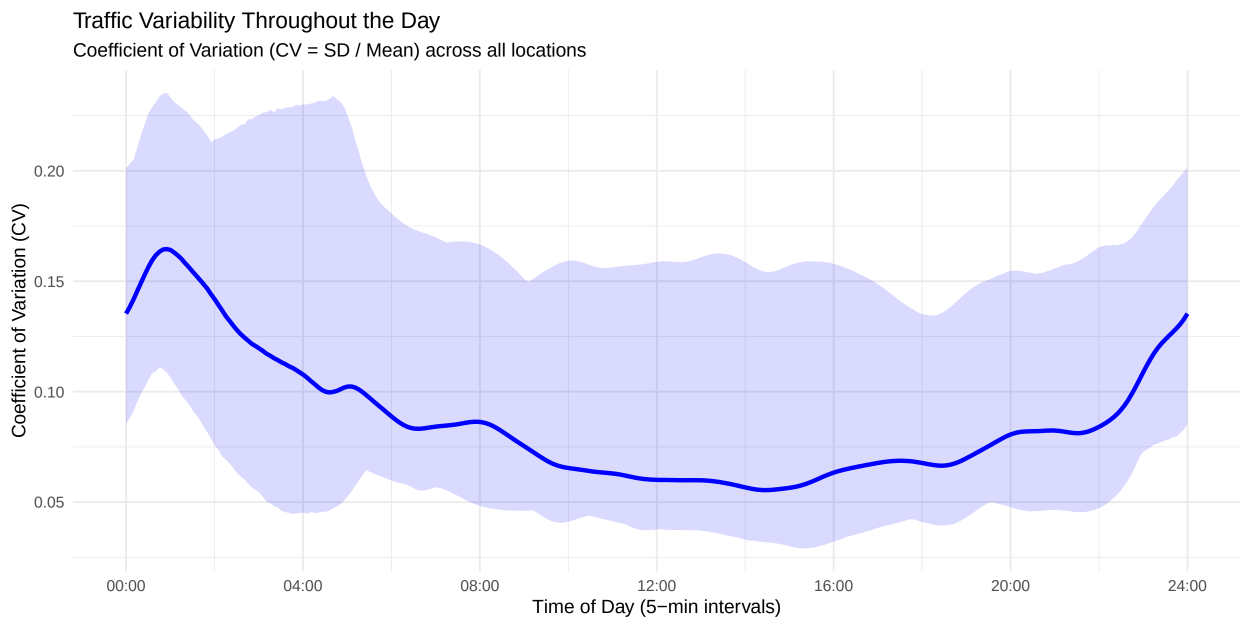
```r
network_avg$AvgVolume <- rowMeans(network_avg[, -1], na.rm = TRUE)

# Plot network-wide average traffic pattern
ggplot(network_avg, aes(x = 1:288, y = AvgVolume)) +
  geom_line(size = 1.2, color = "blue") +
  labs(title = "Network-Wide Average Daily Traffic Pattern",
       subtitle = "Combined average of all locations",
       x = "Time of Day (5-minute intervals)",
       y = "Average Traffic Volume") +
  scale_x_continuous(breaks = seq(1, 288, length.out = 7),
                     labels = c("00:00", "04:00", "08:00", "12:00", "16:00", "20:00", "2
  theme_minimal()
```



## Day-to-Day Variability Analysis using the Coefficient of Variation (CV) |CV Value | Interpretation| |———|————| Low CV (< 0.1)| Very stable — traffic almost the same every day Moderate CV (0.1–0.3)| Normal variation — common for daytime High CV (> 0.3)| Unstable — spikes, holidays, events, irregular travel

Blue line represents the average Coefficient of Variation (CV) across all locations, indicating the predictability of traffic at each 5-minute interval.
Shaded ribbon represents the range of CV values (min–max) across locations, showing how much variability differs between locations at each time of day.

```r
# Make sure time labels are properly defined (length 288)
time_labels <- format(
  seq(from = as.POSIXct("00:00", format="%H:%M"),
      by = "5 min", length.out = 288),
  "%H:%M"
)

# Initialize a dataframe to hold CV values for each location
cv_by_time <- data.frame(Time = time_labels)

for (sheet in names(data_transposed)) {
  df <- data_transposed[[sheet]]   # df = 384 days × 288 timepoints

  # Calculate CV for each timepoint (across all days)
  cv <- apply(df, 2, function(x) sd(x, na.rm = TRUE) / mean(x, na.rm = TRUE))

  cv_by_time[[sheet]] <- cv
}

# Compute summary CV across locations at each timepoint
avg_cv <- data.frame(
  Time = time_labels,
  AvgCV = rowMeans(cv_by_time[, -1], na.rm = TRUE),
  MinCV = apply(cv_by_time[, -1], 1, min, na.rm = TRUE),
  MaxCV = apply(cv_by_time[, -1], 1, max, na.rm = TRUE)
)

# Plot variability across time
ggplot(avg_cv, aes(x = 1:288, y = AvgCV)) +
  geom_ribbon(aes(ymin = MinCV, ymax = MaxCV), alpha = 0.15, fill = "blue") +
  geom_line(color = "blue", size = 1.2) +
  labs(
    title = "Traffic Variability Throughout the Day",
    subtitle = "Coefficient of Variation (CV = SD / Mean) across all locations",
    x = "Time of Day (5-min intervals)",
    y = "Coefficient of Variation (CV)"
  ) +
  scale_x_continuous(
    breaks = seq(1, 288, length.out = 7),
    labels = c("00:00", "04:00", "08:00", "12:00", "16:00", "20:00", "24:00")
  ) +
  theme_minimal()
```



Peak times

```r
peak_times <- data.frame(
  Location = character(),
  MorningPeak = character(),
  EveningPeak = character(),
  QuietHour = character()
)

for (sheet in names(data_transposed)) {
  df <- data_transposed[[sheet]]
  avg_day <- colMeans(df, na.rm = TRUE)

  morning_peak_idx <- which.max(avg_day[1:144])
  evening_peak_idx <- which.max(avg_day[145:288]) + 144
  quiet_idx <- which.min(avg_day)

  peak_times <- rbind(peak_times, data.frame(
    Location = sheet,
    MorningPeak = time_labels[morning_peak_idx],
    EveningPeak = time_labels[evening_peak_idx],
    QuietHour = time_labels[quiet_idx]
  ))
}
```

```
# Display just first few rows, not full table
head(peak_times, 5)
```

```
##   Location MorningPeak EveningPeak QuietHour
## 1     Loc1       07:20       14:50     02:50
## 2     Loc2       06:50       14:35     02:45
## 3     Loc3       06:30       14:15     02:35
## 4     Loc4       07:20       14:40     02:45
## 5     Loc5       07:20       14:40     02:40
```

Temporal Autocorrelation, ACF

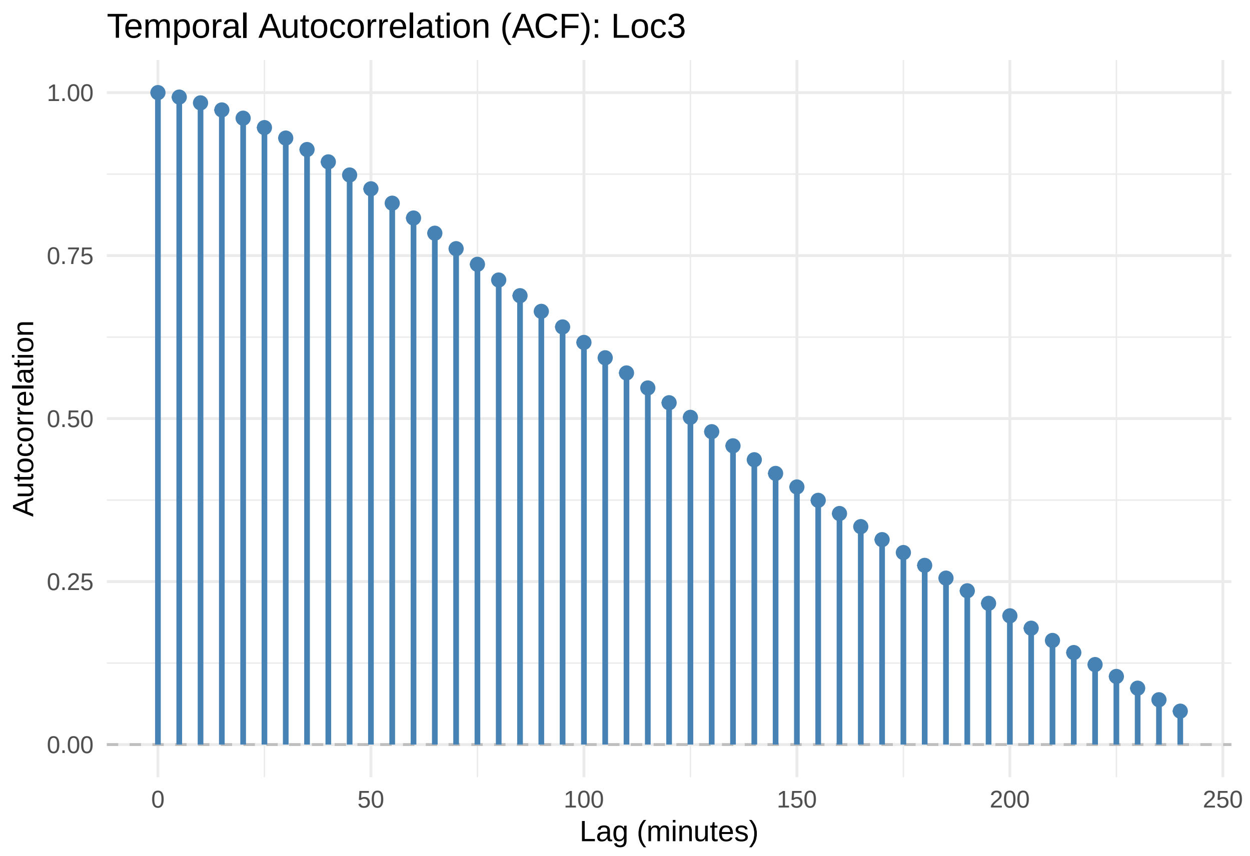
Picking median CV

```
# Compute mean CV for each location
cv_location_summary <- data.frame(
  Location = names(cv_by_time)[-1],
  MeanCV = colMeans(cv_by_time[, -1], na.rm = TRUE)
)

# Pick median CV location (representative)
loc_acf <- cv_location_summary$Location[
  which.min(abs(cv_location_summary$MeanCV - median(cv_location_summary$MeanCV)))
]

cat("Selected location for ACF:", loc_acf, "\n")
```

```
## Selected location for ACF: Loc5
```

```
loc_acf <- sample(names(data_transposed), 1)
df_loc <- data_transposed[[loc_acf]]

# Use median day to reduce noise
typical_day <- apply(df_loc, 2, median)

# Compute ACF (up to 4 hours = 48 lags of 5-min intervals)
acf_result <- acf(typical_day, lag.max = 48, plot = FALSE)

acf_df <- data.frame(
  Lag = acf_result$lag * 5,  # Convert to minutes
  ACF = as.numeric(acf_result$acf)
)
```

```r
ggplot(acf_df, aes(x = Lag, y = ACF)) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_segment(aes(xend = Lag, yend = 0), color = "steelblue", linewidth = 1) +
  geom_point(color = "steelblue", size = 2) +
  labs(title = paste("Temporal Autocorrelation (ACF):", loc_acf),
       x = "Lag (minutes)", y = "Autocorrelation") +
  theme_minimal()
```

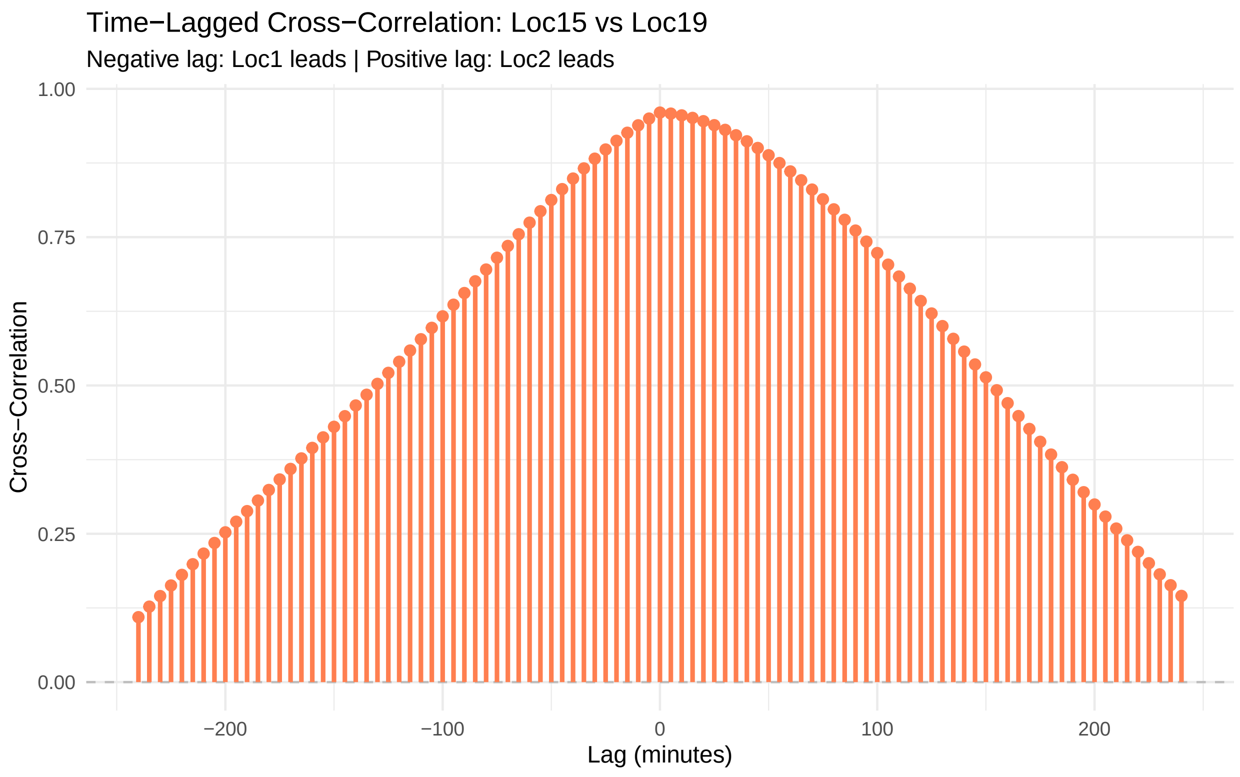


Time-Lagged Cross-Correlation (Lead–Lag Detection)

```r
set.seed(123)

# Always define this BEFORE using it
selected_locs <- sample(names(data_transposed), 2)

loc1 <- selected_locs[1]
loc2 <- selected_locs[2]
cat("Analyzing lagged relationship between:", loc1, "and", loc2, "\n\n")
```

```
## Analyzing lagged relationship between: Loc15 and Loc19
```

```
# Extract median daily profiles
day1 <- apply(data_transposed[[loc1]], 2, median)
day2 <- apply(data_transposed[[loc2]], 2, median)

# Cross-correlation
ccf_result <- ccf(day1, day2, lag.max = 48, plot = FALSE)

ccf_df <- data.frame(
  Lag = ccf_result$lag * 5,
  CCF = as.numeric(ccf_result$acf)
)

# Plot CCF
ggplot(ccf_df, aes(x = Lag, y = CCF)) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_segment(aes(xend = Lag, yend = 0), color = "coral", linewidth = 1) +
  geom_point(color = "coral", size = 2) +
  labs(title = paste("Time-Lagged Cross-Correlation:", loc1, "vs", loc2),
       subtitle = "Negative lag: Loc1 leads | Positive lag: Loc2 leads",
       x = "Lag (minutes)", y = "Cross-Correlation") +
  theme_minimal()
```

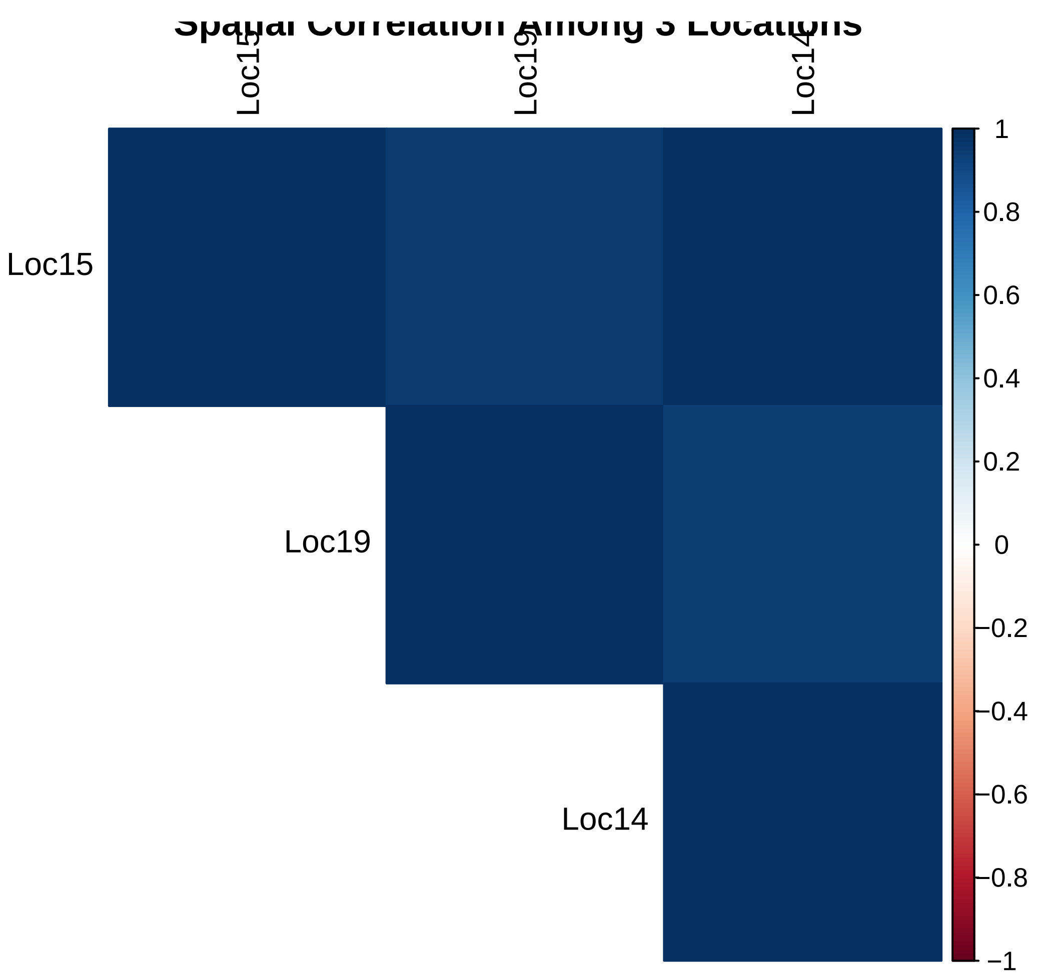


```r
# Highest lag
max_idx <- which.max(abs(ccf_df$CCF))
cat("Max correlation:", round(ccf_df$CCF[max_idx], 3),
    "at", ccf_df$Lag[max_idx], "minutes\n")
```

```
## Max correlation: 0.96 at 0 minutes
```

Spatial correlation (correlgram)

```r
set.seed(123)
selected_locs <- sample(names(data_transposed), 3)

# Compute average daily profiles for selected locations
daily_profile_subset <- data.frame(Time = 1:288)

for (loc in selected_locs) {
  df <- data_transposed[[loc]]
  daily_profile_subset[[loc]] <- colMeans(df, na.rm = TRUE)
}

# Correlation matrix
cor_subset <- cor(daily_profile_subset[, -1])

corrplot(cor_subset, method = "color",
         type = "upper", tl.col = "black", tl.cex = 1,
         title = "Spatial Correlation Among 3 Locations")
```



##

Correlogram from all 26 locations

```r
cat("\n## Spatial Correlation Structure (CORRECTED)\n\n")
```

```
##
## ## Spatial Correlation Structure (CORRECTED)
```

```r
# METHOD 1: Correlate average profiles (what we did - TOO HIGH)
cat("Method 1: Correlating average daily profiles\n")
```

```
## Method 1: Correlating average daily profiles
```

```r
profile_matrix <- matrix(0, nrow = 288, ncol = length(sheet_names))
colnames(profile_matrix) <- sheet_names

for (i in 1:length(sheet_names)) {
  sheet <- sheet_names[i]
  df <- data_transposed[[sheet]]
  avg_profile <- colMeans(df)
  profile_matrix[, i] <- avg_profile
}

cor_profiles <- cor(profile_matrix)
cat("Mean correlation:", round(mean(cor_profiles[upper.tri(cor_profiles)]), 3), "\n\n")
```

```
## Mean correlation: 0.98
```

```r
# METHOD 2: Correlate day-by-day values (BETTER)
cat("Method 2: Correlating daily total traffic volumes\n")
```

```
## Method 2: Correlating daily total traffic volumes
```

```r
# Calculate total daily traffic for each location
daily_totals <- matrix(0, nrow = 384, ncol = length(sheet_names))
colnames(daily_totals) <- sheet_names

for (i in 1:length(sheet_names)) {
  sheet <- sheet_names[i]
  df <- data_transposed[[sheet]]
  # Sum across time points for each day
  daily_totals[, i] <- rowSums(df)
}

# Correlation between daily totals
cor_daily <- cor(daily_totals)

cat("Mean correlation:", round(mean(cor_daily[upper.tri(cor_daily)]), 3), "\n")
```

```
## Mean correlation: 0.235
```

```r
cat("Min correlation:", round(min(cor_daily[upper.tri(cor_daily)]), 3), "\n")
```

```
## Min correlation: -0.437
```

```r
cat("Max correlation:", round(max(cor_daily[upper.tri(cor_daily)]), 3), "\n\n")
```

```
## Max correlation: 0.814
```

```r
# METHOD 3: Correlate full time series (MOST COMPREHENSIVE)
cat("Method 3: Correlating complete time series (384 days × 288 times)\n")
```

```
## Method 3: Correlating complete time series (384 days × 288 times)
```

```r
# Flatten each location to a single vector
full_series <- matrix(0, nrow = 384 * 288, ncol = length(sheet_names))
colnames(full_series) <- sheet_names

for (i in 1:length(sheet_names)) {
  sheet <- sheet_names[i]
  df <- data_transposed[[sheet]]
  # Flatten to single vector
  full_series[, i] <- as.vector(t(df))
}

# Correlation between full time series
cor_full <- cor(full_series)

cat("Mean correlation:", round(mean(cor_full[upper.tri(cor_full)]), 3), "\n")
```

```
## Mean correlation: 0.954
```

```r
cat("Min correlation:", round(min(cor_full[upper.tri(cor_full)]), 3), "\n")
```

```
## Min correlation: 0.86
```

```r
cat("Max correlation:", round(max(cor_full[upper.tri(cor_full)]), 3), "\n\n")
```

```
## Max correlation: 0.994
```

```r
# Compare all three methods
cat("## Comparison of Methods:\n")
```

```
## ## Comparison of Methods:
```

```r
cat("Average profiles correlation:", round(mean(cor_profiles[upper.tri(cor_profiles)]),
```

```
## Average profiles correlation: 0.98
```

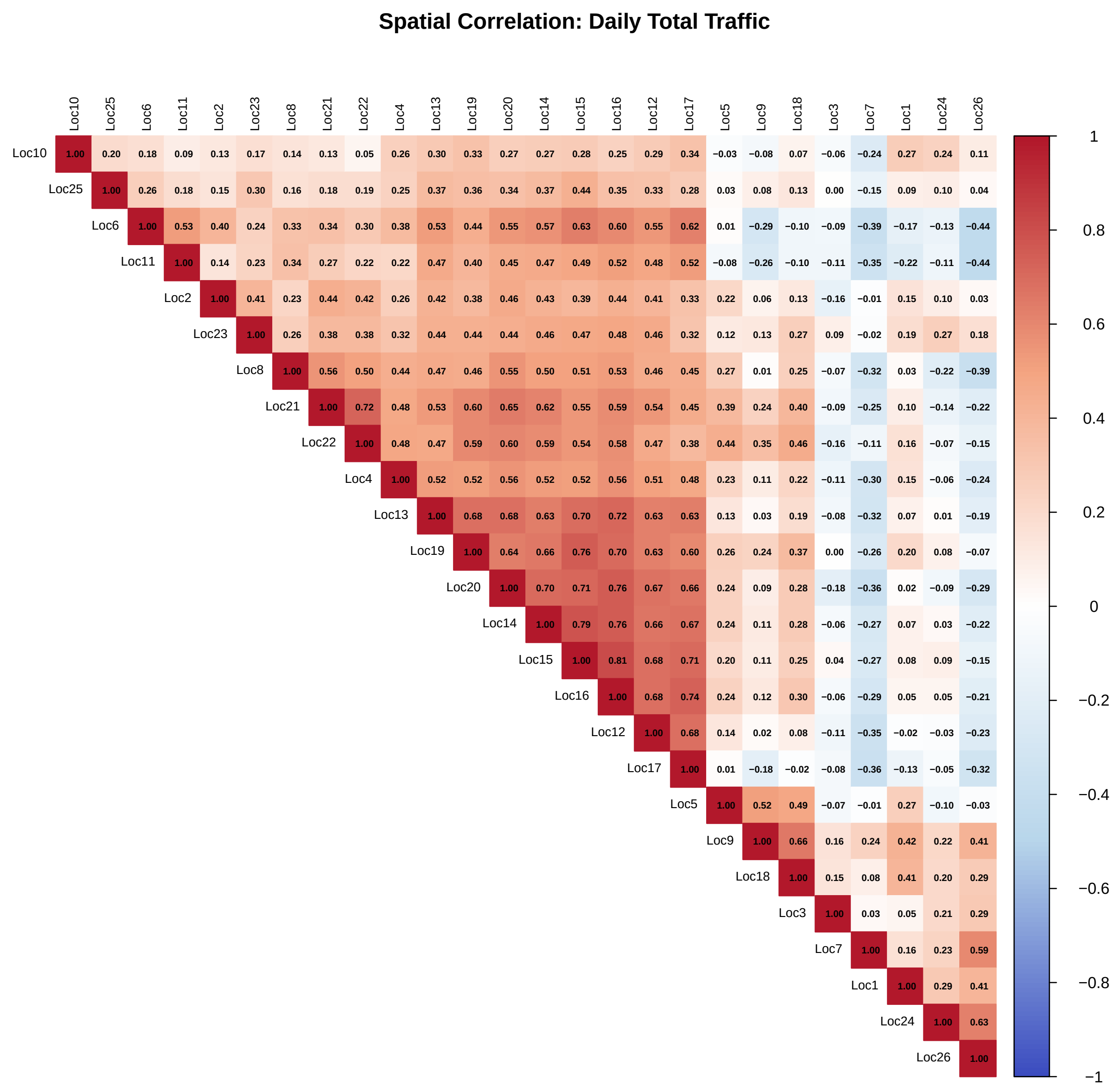
```r
cat("Daily totals correlation:", round(mean(cor_daily[upper.tri(cor_daily)]), 3), "\n")
```

```
## Daily totals correlation: 0.235
```

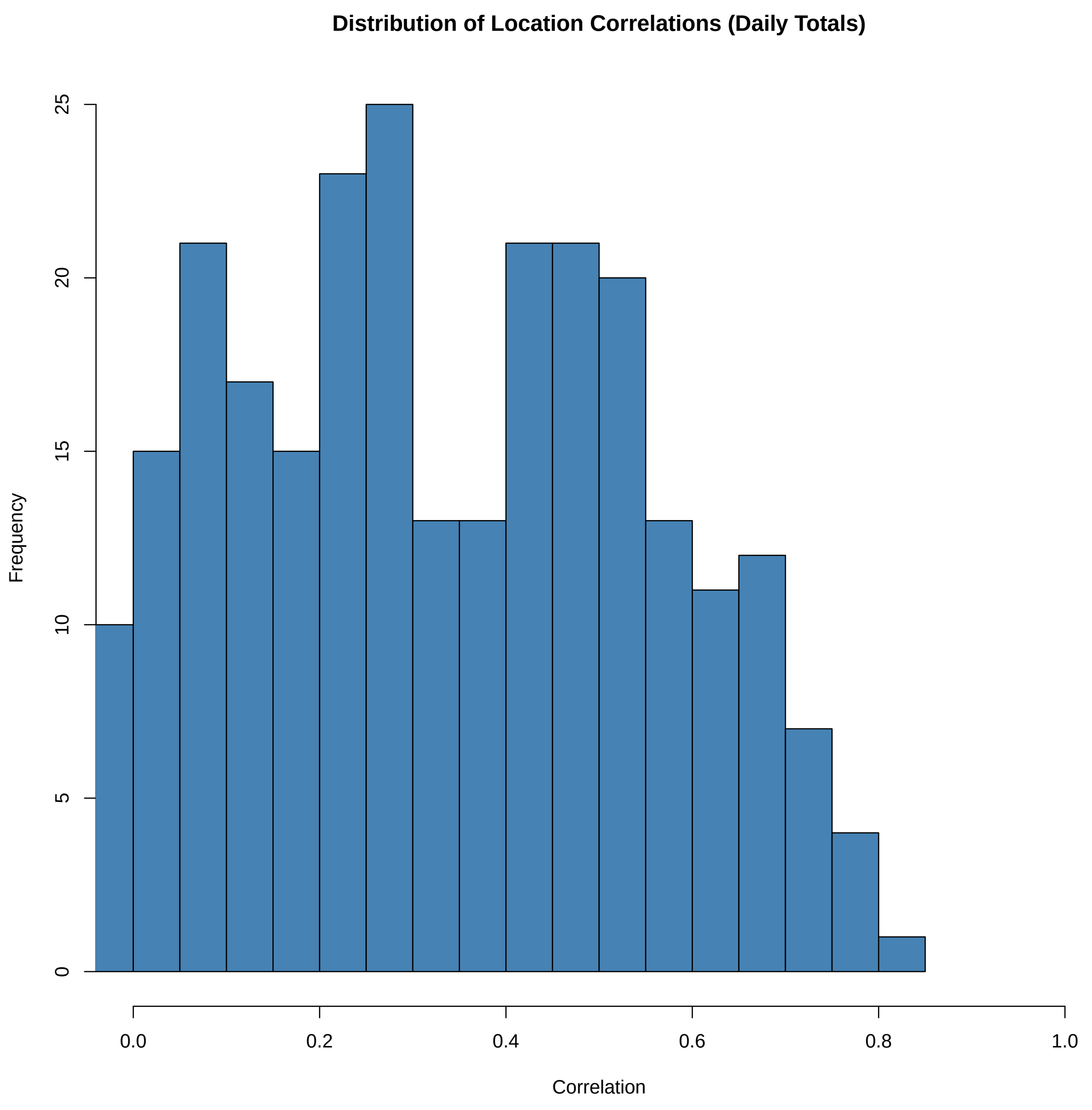
```r
cat("Full series correlation:", round(mean(cor_full[upper.tri(cor_full)]), 3), "\n\n")
```

```
## Full series correlation: 0.954
```

```r
# Plot the DAILY TOTALS correlation (most interpretable)
library(corrplot)

corrplot(cor_daily,
         method = "color",
         type = "upper",
         order = "hclust",
         tl.col = "black",
         tl.cex = 0.7,
         col = colorRampPalette(c("#3B4CC0", "#B8D6EB", "white",
                                   "#F4A582", "#B2182B"))(200),
         title = "Spatial Correlation: Daily Total Traffic",
         mar = c(0, 0, 2, 0),
         addCoef.col = "black",
         number.cex = 0.5)
```



```r
# Histogram of correlations
hist(cor_daily[upper.tri(cor_daily)],
     breaks = 30,
     main = "Distribution of Location Correlations (Daily Totals)",
     xlab = "Correlation",
     col = "steelblue",
     xlim = c(0, 1))
```



```r
cat("\n## Which method to use?\n")
```

```
##
## ## Which method to use?
```

```r
cat("- Average profiles: Shows similarity of daily SHAPE (too high)\n")
```

```
## - Average profiles: Shows similarity of daily SHAPE (too high)
```

```r
cat("- Daily totals: Shows if locations have busy/quiet days together (RECOMMENDED)\n")
```

```
## - Daily totals: Shows if locations have busy/quiet days together (RECOMMENDED)
```

```r
cat("- Full series: Most complete but computationally intensive\n")
```

```
## - Full series: Most complete but computationally intensive
```

```r
cat("\n## Spatial Correlation Structure Across Locations\n\n")
```

```
##
## ## Spatial Correlation Structure Across Locations
```

```r
# Calculate total daily traffic for each location
# (sum of 288 five-minute intervals per day)
daily_totals <- matrix(0, nrow = 384, ncol = length(sheet_names))
colnames(daily_totals) <- sheet_names

for (i in 1:length(sheet_names)) {
  sheet <- sheet_names[i]
  df <- data_transposed[[sheet]]
  daily_totals[, i] <- rowSums(df)
}

# Compute correlation matrix
location_cor <- cor(daily_totals)

# Summary statistics
cor_values <- location_cor[upper.tri(location_cor)]

cat("Correlation Statistics:\n")
```

```
## Correlation Statistics:
```

```r
cat("  Mean:", round(mean(cor_values), 3), "\n")
```

```
##   Mean: 0.235
```

```r
cat("  Median:", round(median(cor_values), 3), "\n")
```

```
##   Median: 0.249
```

```r
cat("  Min:", round(min(cor_values), 3), "\n")
```

```
##   Min: -0.437
```

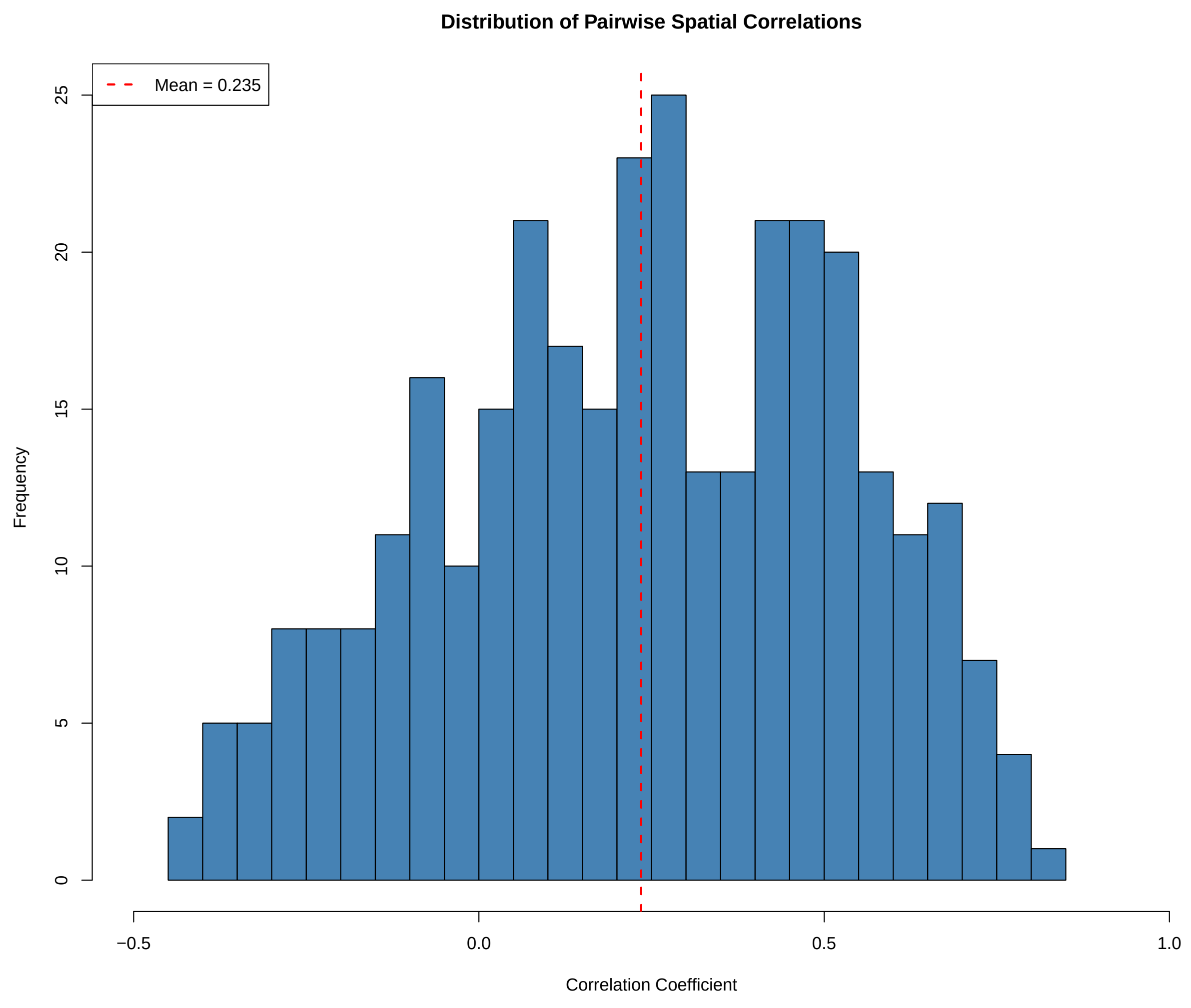
```r
cat("  Max:", round(max(cor_values), 3), "\n")
```

```
##   Max: 0.814
```

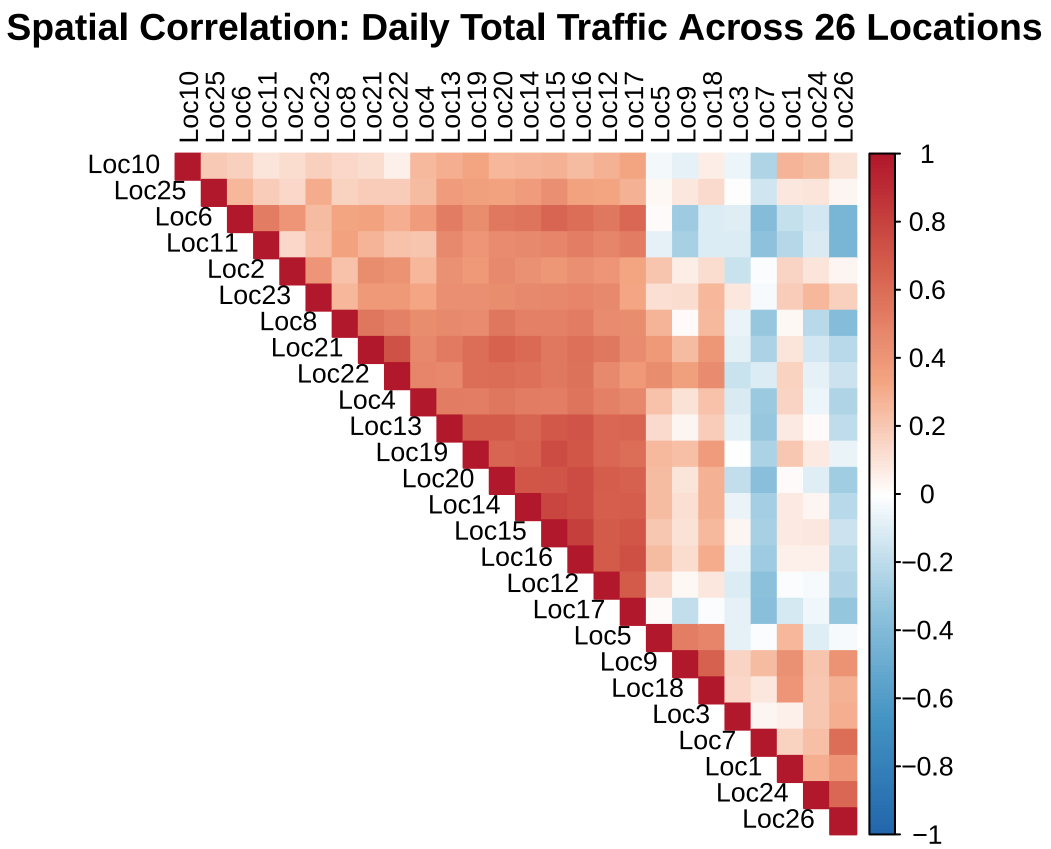
```r
cat("  SD:", round(sd(cor_values), 3), "\n\n")
```

```
##   SD: 0.289
```

```r
# Histogram of correlations
hist(cor_values,
     breaks = 30,
     main = "Distribution of Pairwise Spatial Correlations",
     xlab = "Correlation Coefficient",
     col = "steelblue",
     xlim = c(-0.5, 1))
abline(v = mean(cor_values), col = "red", lwd = 2, lty = 2)
legend("topleft", legend = paste("Mean =", round(mean(cor_values), 3)),
       col = "red", lty = 2, lwd = 2)
```



```r
# Visualization 1: corrplot with clustering

corrplot(location_cor,
         method = "color",
         type = "upper",
         order = "hclust",
         tl.col = "black",
         tl.cex = 0.8,
         col = colorRampPalette(c("#2166AC", "#4393C3", "#92C5DE",
                                  "white",
                                  "#F4A582", "#D6604D", "#B2182B"))(200),
         title = "Spatial Correlation: Daily Total Traffic Across 26 Locations",
         mar = c(0, 0, 2, 0),
         cl.cex = 0.8)
```



```r
cat("\nInterpretation:\n")
```

```
##
## Interpretation:
```

```r
cat("- Red = positive correlation (locations co-vary)\n")
```

```
## - Red = positive correlation (locations co-vary)
```

```r
cat("- Blue = negative correlation (inverse relationship)\n")
```

```
## - Blue = negative correlation (inverse relationship)
```

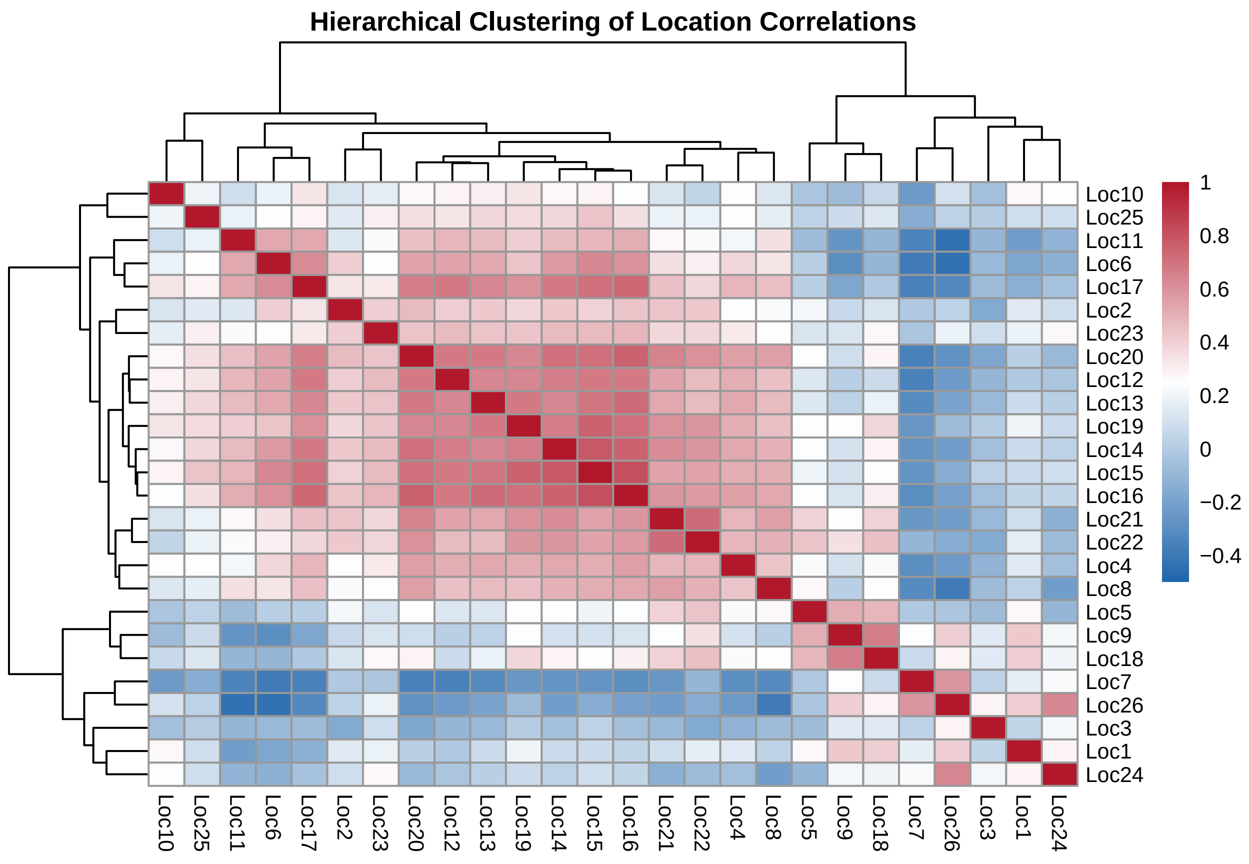
```r
cat("- White = no correlation (independent)\n")
```

```
## - White = no correlation (independent)
```

```r
cat("- Clustering reveals groups with similar day-to-day dynamics\n\n")
```

```
## - Clustering reveals groups with similar day-to-day dynamics
```

```r
# Visualization 2: pheatmap with dendrograms
library(pheatmap)

pheatmap(location_cor,
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         color = colorRampPalette(c("#2166AC", "white", "#B2182B"))(100),
         display_numbers = FALSE,
         fontsize = 8,
         main = "Hierarchical Clustering of Location Correlations",
         breaks = seq(-0.5, 1, length.out = 101))
```



```r
cat("\nDendrograms show spatial clustering:\n")
```

```
##
## Dendrograms show spatial clustering:
```

```r
cat("- Nearby branches = locations with similar traffic patterns\n")
```

```
## - Nearby branches = locations with similar traffic patterns
```

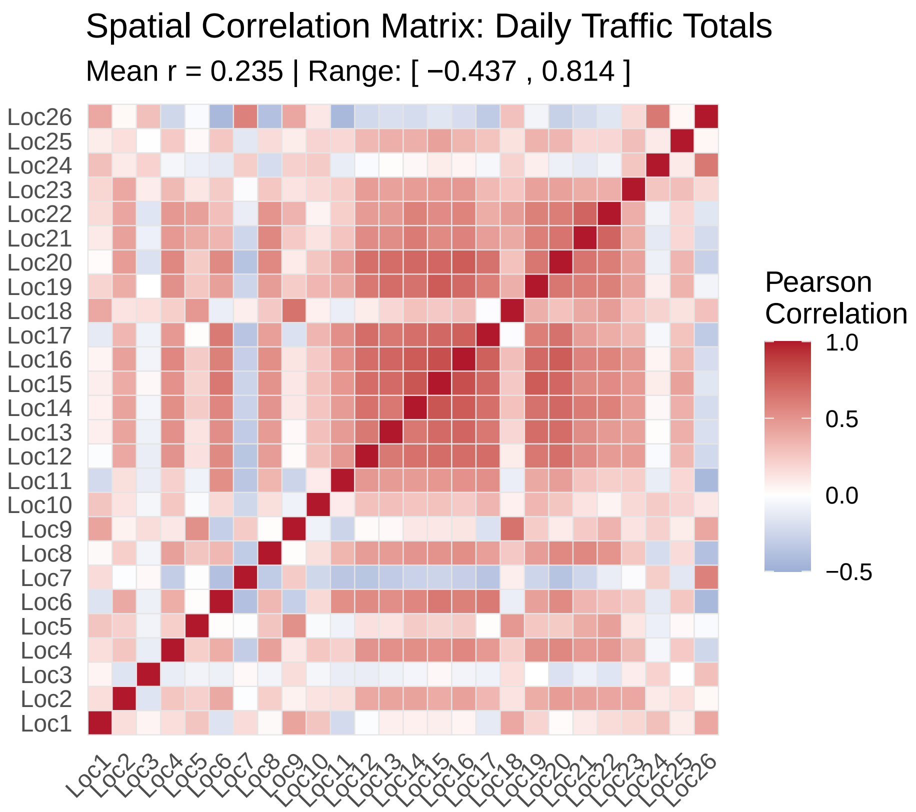
```r
cat("- Branch height = degree of dissimilarity\n\n")
```

```
## - Branch height = degree of dissimilarity
```

```r
# Visualization 3: ggplot heatmap
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
cor_melted <- melt(location_cor)
colnames(cor_melted) <- c("Location1", "Location2", "Correlation")

p_cor <- ggplot(cor_melted, aes(x = Location1, y = Location2, fill = Correlation)) +
  geom_tile(color = "gray90", linewidth = 0.2) +
  scale_fill_gradient2(low = "#2166AC", mid = "white", high = "#B2182B",
                       midpoint = 0,
                       limits = c(-0.5, 1),
                       name = "Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 9),
        axis.text.y = element_text(size = 9),
        panel.grid = element_blank()) +
  labs(title = "Spatial Correlation Matrix: Daily Traffic Totals",
       subtitle = paste("Mean r =", round(mean(cor_values), 3),
                        "| Range: [", round(min(cor_values), 3), ",",
                        round(max(cor_values), 3), "]"),
       x = "", y = "") +
  coord_fixed()

print(p_cor)
```



```r
# Find most and least correlated pairs
cor_df <- melt(location_cor)
cor_df <- cor_df %>%
  filter(Var1 != Var2) %>%
  filter(as.character(Var1) < as.character(Var2))

top_cor <- cor_df %>% arrange(desc(value)) %>% head(10)
bottom_cor <- cor_df %>% arrange(value) %>% head(10)

cat("\n## Most Correlated Location Pairs (Top 10):\n")
```

```
##
## ## Most Correlated Location Pairs (Top 10):
```

```r
print(kable(top_cor, digits = 3,
            col.names = c("Location 1", "Location 2", "Correlation")))
```

```
##
##
## |Location 1 |Location 2 | Correlation|
## |:----------|:----------|-----------:|
## |Loc15      |Loc16      |       0.814|
## |Loc14      |Loc15      |       0.786|
## |Loc15      |Loc19      |       0.759|
## |Loc14      |Loc16      |       0.756|
## |Loc16      |Loc20      |       0.756|
## |Loc16      |Loc17      |       0.737|
## |Loc21      |Loc22      |       0.720|
## |Loc13      |Loc16      |       0.716|
## |Loc15      |Loc20      |       0.711|
## |Loc15      |Loc17      |       0.706|
```

```
cat("\n## Least Correlated Location Pairs (Bottom 10):\n")
```

```
##
## ## Least Correlated Location Pairs (Bottom 10):
```

```
print(kable(bottom_cor, digits = 3,
            col.names = c("Location 1", "Location 2", "Correlation")))
```

```
##
##
## |Location 1 |Location 2 | Correlation|
## |:----------|:----------|-----------:|
## |Loc26      |Loc6       |      -0.437|
## |Loc11      |Loc26      |      -0.436|
## |Loc26      |Loc8       |      -0.388|
## |Loc6       |Loc7       |      -0.388|
## |Loc20      |Loc7       |      -0.365|
## |Loc17      |Loc7       |      -0.362|
## |Loc12      |Loc7       |      -0.353|
## |Loc11      |Loc7       |      -0.347|
## |Loc17      |Loc26      |      -0.324|
## |Loc7       |Loc8       |      -0.318|
```

```
cat("\n## Key Findings:\n")
```

```
##
## ## Key Findings:
```

```r
cat("1. Moderate spatial correlation (mean r =", round(mean(cor_values), 3), ")\n")
```

```
## 1. Moderate spatial correlation (mean r = 0.235 )
```

```r
cat("   indicates locations are NOT perfectly synchronized\n\n")
```

```
##    indicates locations are NOT perfectly synchronized
```

```r
cat("2. Substantial variation in correlations suggests:\n")
```

```
## 2. Substantial variation in correlations suggests:
```

```r
cat("   - Some location pairs strongly connected\n")
```

```
##    - Some location pairs strongly connected
```

```r
cat("   - Others operate relatively independently\n")
```

```
##    - Others operate relatively independently
```

```r
cat("   - Network structure exists and can be analyzed\n\n")
```

```
##    - Network structure exists and can be analyzed
```

```r
cat("3. This justifies multi-level approach:\n")
```

```
## 3. This justifies multi-level approach:
```

```r
cat("   - Location-specific analysis (local patterns)\n")
```

```
##    - Location-specific analysis (local patterns)
```

```r
cat("   - Network-level analysis (coordinated disruptions)\n")
```

```
##    - Network-level analysis (coordinated disruptions)
```

PCA Analysis for All Locations

```r
cat("Hello from cat()\n")
```

```
## Hello from cat()
```

```r
print("Hello from print()")
```

```
## [1] "Hello from print()"
```

```r
message("Hello from message()")
```

```
## Hello from message()
```

```r
for (i in 1:5) {
  message("Checking iteration ", i)
}
```

```
## Checking iteration 1
```

```
## Checking iteration 2
```

```
## Checking iteration 3
```

```
## Checking iteration 4
```

```
## Checking iteration 5
```

```r
cat("\n## PCA Analysis: All 26 Locations\n\n")
```

```
##
## ## PCA Analysis: All 26 Locations
```

```r
# Initialize storage
pca_results <- list()
loadings_list <- list()
scores_list <- list()
variance_explained <- list()

# Set variance threshold
variance_threshold <- 0.90  # Retain components explaining 90% variance
```

```r
for (sheet in sheet_names) {
  cat("Location:", sheet, "\n")

  # Use already-loaded data
  df_transposed <- data_transposed[[sheet]]  # 384 days × 288 time points

  # Perform PCA
  pca <- prcomp(df_transposed, center = TRUE, scale. = TRUE)

  # Determine number of components to retain
  cumvar <- cumsum(pca$sdev^2 / sum(pca$sdev^2))
  n_comp <- which(cumvar >= variance_threshold)[1]

  cat("  Retaining", n_comp, "components (",
      round(cumvar[n_comp] * 100, 1), "% variance)\n")

  # Store results (ONLY retained components)
  pca_results[[sheet]] <- pca
  loadings_list[[sheet]] <- pca$rotation[, 1:n_comp]
  scores_list[[sheet]] <- pca$x[, 1:n_comp]
  variance_explained[[sheet]] <- cumvar[n_comp]
}
```

```
## Location: Loc1
##   Retaining 6 components ( 90.8 % variance)
## Location: Loc2
##   Retaining 7 components ( 90 % variance)
## Location: Loc3
##   Retaining 6 components ( 91.1 % variance)
## Location: Loc4
##   Retaining 8 components ( 90.7 % variance)
## Location: Loc5
##   Retaining 8 components ( 90.6 % variance)
## Location: Loc6
##   Retaining 7 components ( 90.5 % variance)
## Location: Loc7
##   Retaining 2 components ( 90.3 % variance)
## Location: Loc8
##   Retaining 10 components ( 92 % variance)
## Location: Loc9
##   Retaining 6 components ( 91.1 % variance)
## Location: Loc10
##   Retaining 6 components ( 91.1 % variance)
## Location: Loc11
##   Retaining 10 components ( 91 % variance)
## Location: Loc12
##   Retaining 8 components ( 91.6 % variance)
## Location: Loc13
##   Retaining 9 components ( 90.7 % variance)
## Location: Loc14
##   Retaining 8 components ( 90.8 % variance)
## Location: Loc15
##   Retaining 8 components ( 91.3 % variance)
## Location: Loc16
##   Retaining 9 components ( 91.3 % variance)
## Location: Loc17
##   Retaining 8 components ( 90.7 % variance)
## Location: Loc18
##   Retaining 6 components ( 91.8 % variance)
## Location: Loc19
##   Retaining 10 components ( 90.5 % variance)
## Location: Loc20
##   Retaining 7 components ( 91.3 % variance)
## Location: Loc21
##   Retaining 6 components ( 90.8 % variance)
## Location: Loc22
##   Retaining 7 components ( 91.3 % variance)
## Location: Loc23
##   Retaining 8 components ( 91.6 % variance)
## Location: Loc24
##   Retaining 4 components ( 92.2 % variance)
## Location: Loc25
##   Retaining 8 components ( 91.7 % variance)
## Location: Loc26
##   Retaining 3 components ( 92.6 % variance)
```

```r
cat("\nPCA completed for all", length(pca_results), "locations\n")
```

```
##
## PCA completed for all 26 locations
```

```r
cat("Average components retained:",
    round(mean(sapply(loadings_list, ncol)), 1), "\n\n")
```

```
## Average components retained: 7.1
```

```r
cat("\n## Summary: Components Retained Per Location\n\n")
```

```
##
## ## Summary: Components Retained Per Location
```

```r
# Create summary table
summary_df <- data.frame(
  Location = sheet_names,
  Components = sapply(loadings_list, ncol),
  Variance = round(unlist(variance_explained) * 100, 1)
)

print(kable(summary_df,
            col.names = c("Location", "Components Retained", "% Variance")))
```

```
##
##
## |      |Location | Components Retained| % Variance|
## |:-----|:--------|-------------------:|----------:|
## |Loc1  |Loc1     |                   6|       90.8|
## |Loc2  |Loc2     |                   7|       90.0|
## |Loc3  |Loc3     |                   6|       91.1|
## |Loc4  |Loc4     |                   8|       90.7|
## |Loc5  |Loc5     |                   8|       90.6|
## |Loc6  |Loc6     |                   7|       90.5|
## |Loc7  |Loc7     |                   2|       90.3|
## |Loc8  |Loc8     |                  10|       92.0|
## |Loc9  |Loc9     |                   6|       91.1|
## |Loc10 |Loc10    |                   6|       91.1|
## |Loc11 |Loc11    |                  10|       91.0|
## |Loc12 |Loc12    |                   8|       91.6|
## |Loc13 |Loc13    |                   9|       90.7|
## |Loc14 |Loc14    |                   8|       90.8|
## |Loc15 |Loc15    |                   8|       91.3|
## |Loc16 |Loc16    |                   9|       91.3|
## |Loc17 |Loc17    |                   8|       90.7|
## |Loc18 |Loc18    |                   6|       91.8|
## |Loc19 |Loc19    |                  10|       90.5|
## |Loc20 |Loc20    |                   7|       91.3|
## |Loc21 |Loc21    |                   6|       90.8|
## |Loc22 |Loc22    |                   7|       91.3|
## |Loc23 |Loc23    |                   8|       91.6|
## |Loc24 |Loc24    |                   4|       92.2|
## |Loc25 |Loc25    |                     8|       91.7|
## |Loc26 |Loc26    |                     3|       92.6|
```

Detect anomalies from PCA scores using boxplots

```
cat("\n## Anomaly Detection: PCA Score Outliers\n\n")
```

```
##
## ## Anomaly Detection: PCA Score Outliers
```

```
anomalies <- list()  # Store anomalies per location

for (sheet in sheet_names) {
  scores <- as.data.frame(scores_list[[sheet]])
  colnames(scores) <- paste0("PC", 1:ncol(scores))

  sheet_anomalies <- data.frame(Day = integer(), PC = character(), Score = numeric())

  for (pc in colnames(scores)) {
    q1 <- quantile(scores[[pc]], 0.25)
    q3 <- quantile(scores[[pc]], 0.75)
    iqr <- q3 - q1
    lower <- q1 - 1.5 * iqr
    upper <- q3 + 1.5 * iqr

    flagged <- which(scores[[pc]] < lower | scores[[pc]] > upper)

    if (length(flagged) > 0) {
      sheet_anomalies <- rbind(sheet_anomalies,
                               data.frame(Day = flagged,
                                          PC = pc,
                                          Score = scores[[pc]][flagged]))
    }
  }

  anomalies[[sheet]] <- sheet_anomalies
}

cat("Anomaly detection completed.\n\n")
```

```
## Anomaly detection completed.
```

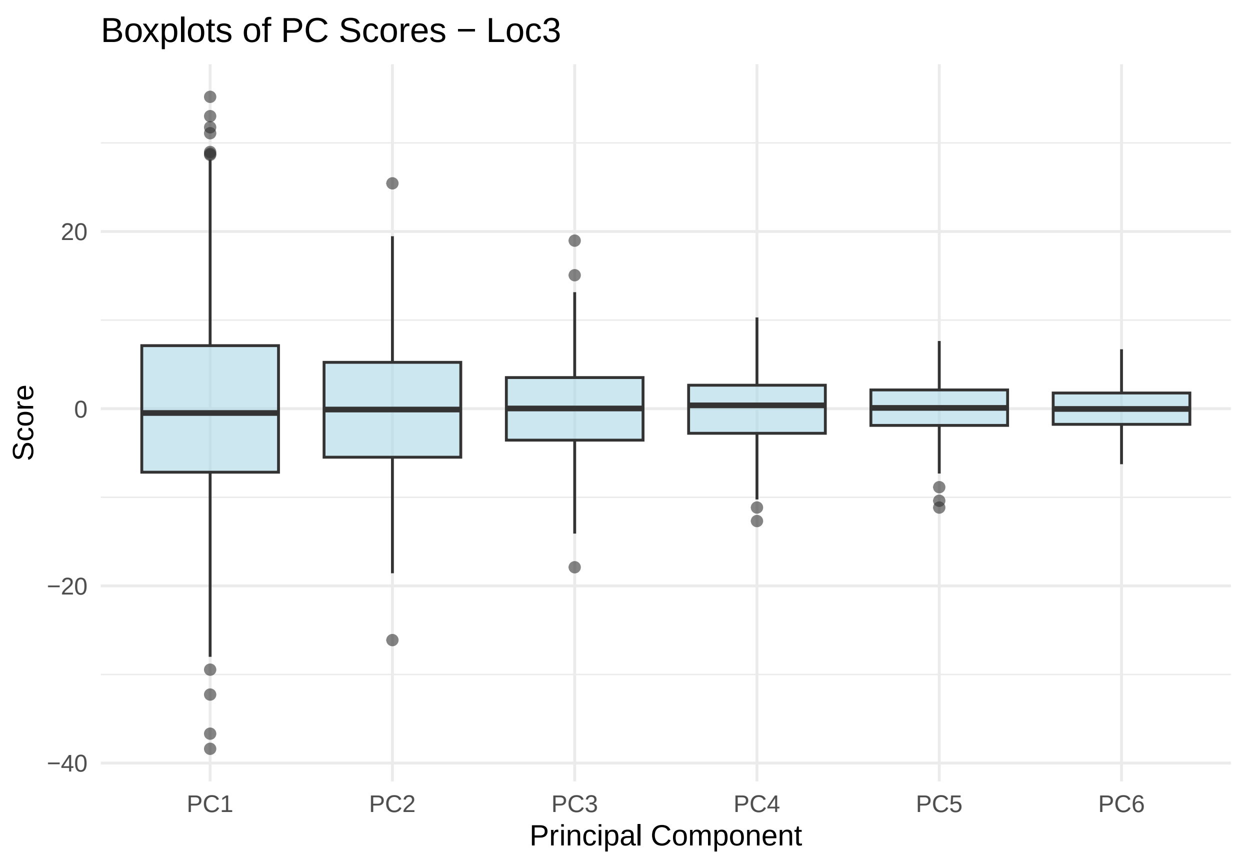
```r
cat("Example anomalies:\n")
```

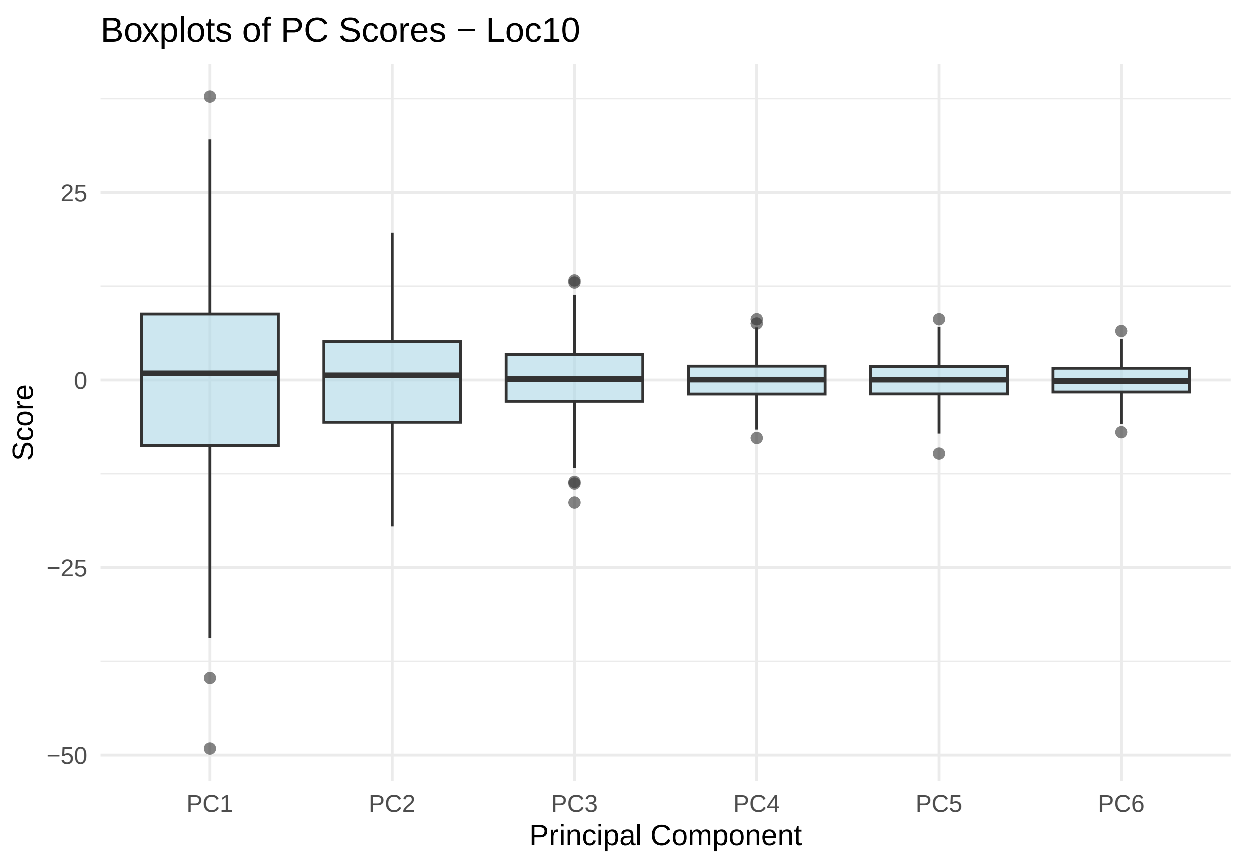
```
## Example anomalies:
```

```r
print(head(anomalies[[sheet_names[1]]], 10))
```

```
##    Day  PC     Score
## 1  357 PC1 -37.05090
## 2  358 PC1 -35.44125
## 3  359 PC1 -43.51325
## 4  382 PC1 -36.11765
## 5   26 PC2  17.50476
## 6   75 PC2 -19.24042
## 7  259 PC2 -19.86931
## 8  294 PC2  22.80139
## 9  295 PC2  17.60088
## 10 331 PC2  21.00681
```

```r
## Plot boxplots of PC scores for sample locations
sample_locs <- sample(sheet_names, 2)

for (sheet in sample_locs) {
  scores <- as.data.frame(scores_list[[sheet]])
  colnames(scores) <- paste0("PC", 1:ncol(scores))

  scores_long <- scores %>%
    mutate(Day = 1:n()) %>%
    pivot_longer(cols = starts_with("PC"), names_to = "Component", values_to = "Score")

  p <- ggplot(scores_long, aes(x = Component, y = Score)) +
    geom_boxplot(fill = "lightblue", alpha = 0.6) +
    labs(title = paste("Boxplots of PC Scores -", sheet),
         x = "Principal Component", y = "Score") +
    theme_minimal()

  print(p)
}
```


Boxplots of PC Scores − Loc3
Score
20
0
−20
−40
PC1
PC2
PC3
PC4
PC5
PC6
Principal Component



```
## Classify anomalies using PC loading time-of-day regimes

classifications <- list()

for (sheet in sheet_names) {
  cat("\nProcessing:", sheet, "\n")

  loadings <- loadings_list[[sheet]]  # Timepoints × components
  scores <- scores_list[[sheet]]      # Days × components
  anomalies_df <- anomalies[[sheet]]  # Day, PC, Score

  if (nrow(anomalies_df) == 0) {
    classifications[[sheet]] <- data.frame()
    next
  }

  anomaly_results <- anomalies_df %>% mutate(Regime = NA)

  for (i in 1:nrow(anomaly_results)) {
    pc_name <- anomaly_results$PC[i]
    pc_index <- as.numeric(gsub("PC", "", pc_name))
    pc_loading <- loadings[, pc_index]
    peak_hour <- which.max(abs(pc_loading)) * 5 / 60

    anomaly_results$Regime[i] <- case_when(
      peak_hour >= 6 & peak_hour < 10 ~ "Morning anomaly",
      peak_hour >= 10 & peak_hour < 16 ~ "Afternoon anomaly",
      peak_hour >= 16 & peak_hour < 21 ~ "Evening anomaly",
      TRUE ~ "All-day / Off-peak anomaly"
    )
  }

  classifications[[sheet]] <- anomaly_results
}
```

```
##
## Processing: Loc1
##
## Processing: Loc2
##
## Processing: Loc3
##
## Processing: Loc4
##
## Processing: Loc5
##
## Processing: Loc6
##
## Processing: Loc7
##
## Processing: Loc8
##
## Processing: Loc9
##
## Processing: Loc10
##
## Processing: Loc11
##
## Processing: Loc12
##
## Processing: Loc13
##
## Processing: Loc14
##
## Processing: Loc15
```

```
##
## Processing: Loc16
##
## Processing: Loc17
##
## Processing: Loc18
##
## Processing: Loc19
##
## Processing: Loc20
##
## Processing: Loc21
##
## Processing: Loc22
##
## Processing: Loc23
##
## Processing: Loc24
##
## Processing: Loc25
##
## Processing: Loc26
```

```
cat("\nClassification COMPLETE.\n")
```

```
##
## Classification COMPLETE.
```

```
print(head(classifications[[sheet_names[1]]], 10))
```

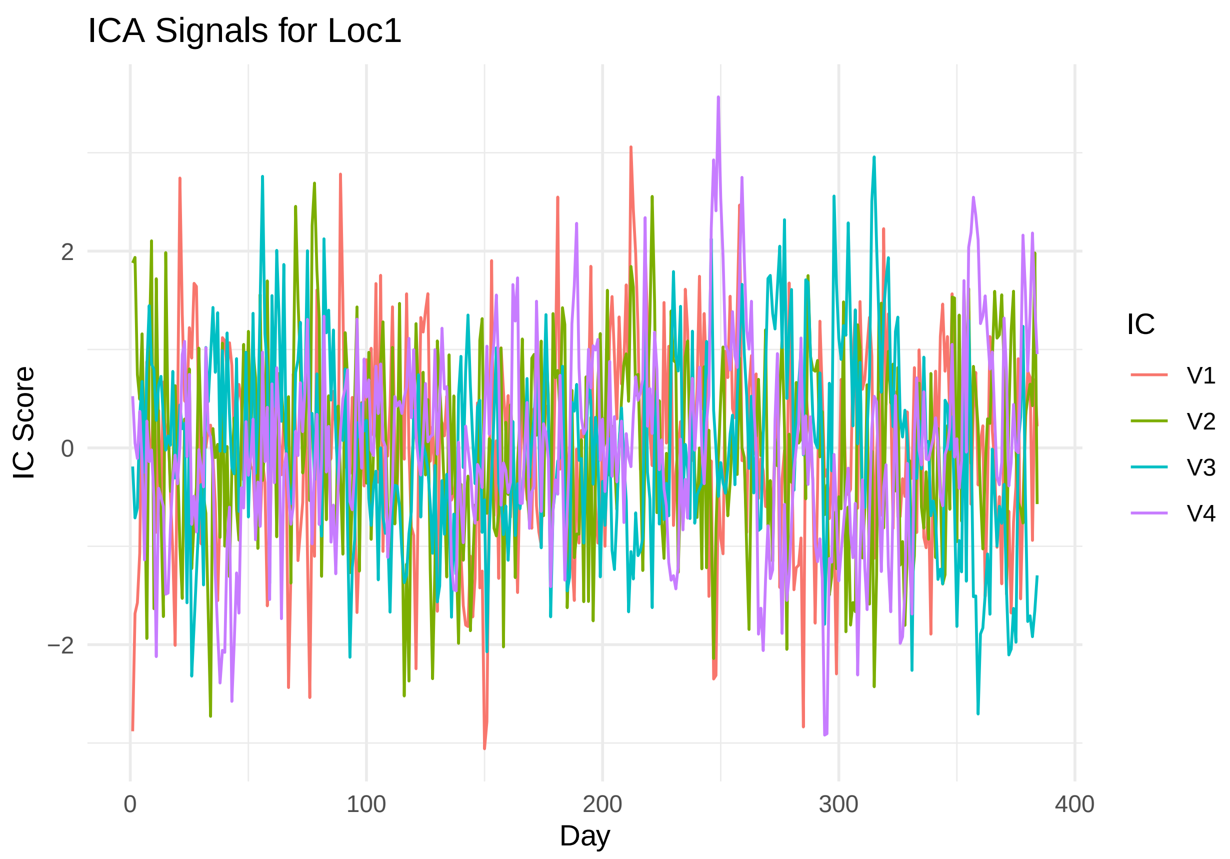
```
##    Day  PC     Score                     Regime
## 1  357 PC1 -37.05090           Morning anomaly
## 2  358 PC1 -35.44125           Morning anomaly
## 3  359 PC1 -43.51325           Morning anomaly
## 4  382 PC1 -36.11765           Morning anomaly
## 5   26 PC2  17.50476 All-day / Off-peak anomaly
## 6   75 PC2 -19.24042 All-day / Off-peak anomaly
## 7  259 PC2 -19.86931 All-day / Off-peak anomaly
## 8  294 PC2  22.80139 All-day / Off-peak anomaly
## 9  295 PC2  17.60088 All-day / Off-peak anomaly
## 10 331 PC2  21.00681 All-day / Off-peak anomaly
```

```r
library(fastICA)

ica_results <- list()

for (sheet in sheet_names) {
  scores <- as.data.frame(scores_list[[sheet]])  # use PCA scores

  if (ncol(scores) >= 3) {
    ica <- fastICA(scores, n.comp = min(4, ncol(scores)), method = "C")

    ica_results[[sheet]] <- list(
      Sources = ica$S,
      MixingMatrix = ica$A
    )

    cat("\nICA successful for:", sheet,
        "| Components extracted:", ncol(ica$S), "\n")
  } else {
    cat("Skipping", sheet, "- not enough components\n")
  }
}
```

```
##
## ICA successful for: Loc1 | Components extracted: 4
##
## ICA successful for: Loc2 | Components extracted: 4
##
## ICA successful for: Loc3 | Components extracted: 4
##
## ICA successful for: Loc4 | Components extracted: 4
##
## ICA successful for: Loc5 | Components extracted: 4
##
## ICA successful for: Loc6 | Components extracted: 4
## Skipping Loc7 - not enough components
##
## ICA successful for: Loc8 | Components extracted: 4
##
## ICA successful for: Loc9 | Components extracted: 4
##
## ICA successful for: Loc10 | Components extracted: 4
##
## ICA successful for: Loc11 | Components extracted: 4
##
```

```
## ICA successful for: Loc12 | Components extracted: 4
##
## ICA successful for: Loc13 | Components extracted: 4
##
## ICA successful for: Loc14 | Components extracted: 4
##
## ICA successful for: Loc15 | Components extracted: 4
##
## ICA successful for: Loc16 | Components extracted: 4
##
## ICA successful for: Loc17 | Components extracted: 4
##
## ICA successful for: Loc18 | Components extracted: 4
##
## ICA successful for: Loc19 | Components extracted: 4
##
## ICA successful for: Loc20 | Components extracted: 4
##
## ICA successful for: Loc21 | Components extracted: 4
##
## ICA successful for: Loc22 | Components extracted: 4
##
## ICA successful for: Loc23 | Components extracted: 4
##
## ICA successful for: Loc24 | Components extracted: 4
##
## ICA successful for: Loc25 | Components extracted: 4
##
## ICA successful for: Loc26 | Components extracted: 3
```

```r
sheet <- sheet_names[1]  # Example
sources <- as.data.frame(ica_results[[sheet]]$Sources)
sources$Day <- 1:nrow(sources)

sources_long <- melt(sources, id.vars = "Day",
                     variable.name = "IC", value.name = "Score")

ggplot(sources_long, aes(x = Day, y = Score, color = IC)) +
  geom_line() +
  labs(title = paste("ICA Signals for", sheet),
       x = "Day", y = "IC Score") +
  theme_minimal()
```



```
ica_anomalies <- list()

for (sheet in sheet_names) {

  if (!is.null(ica_results[[sheet]])) {
    ics <- as.data.frame(ica_results[[sheet]]$Sources)

    flagged_days <- unique(unlist(
      lapply(ics, function(ic) {
        q1 <- quantile(ic, 0.25)
        q3 <- quantile(ic, 0.75)
        iqr <- q3 - q1
        lower <- q1 - 1.5 * iqr
        upper <- q3 + 1.5 * iqr
        which(ic < lower | ic > upper)
      })
    ))

    ica_anomalies[[sheet]] <- if (length(flagged_days) > 0) {
      data.frame(Day = flagged_days, Method = "ICA")
    } else {
```

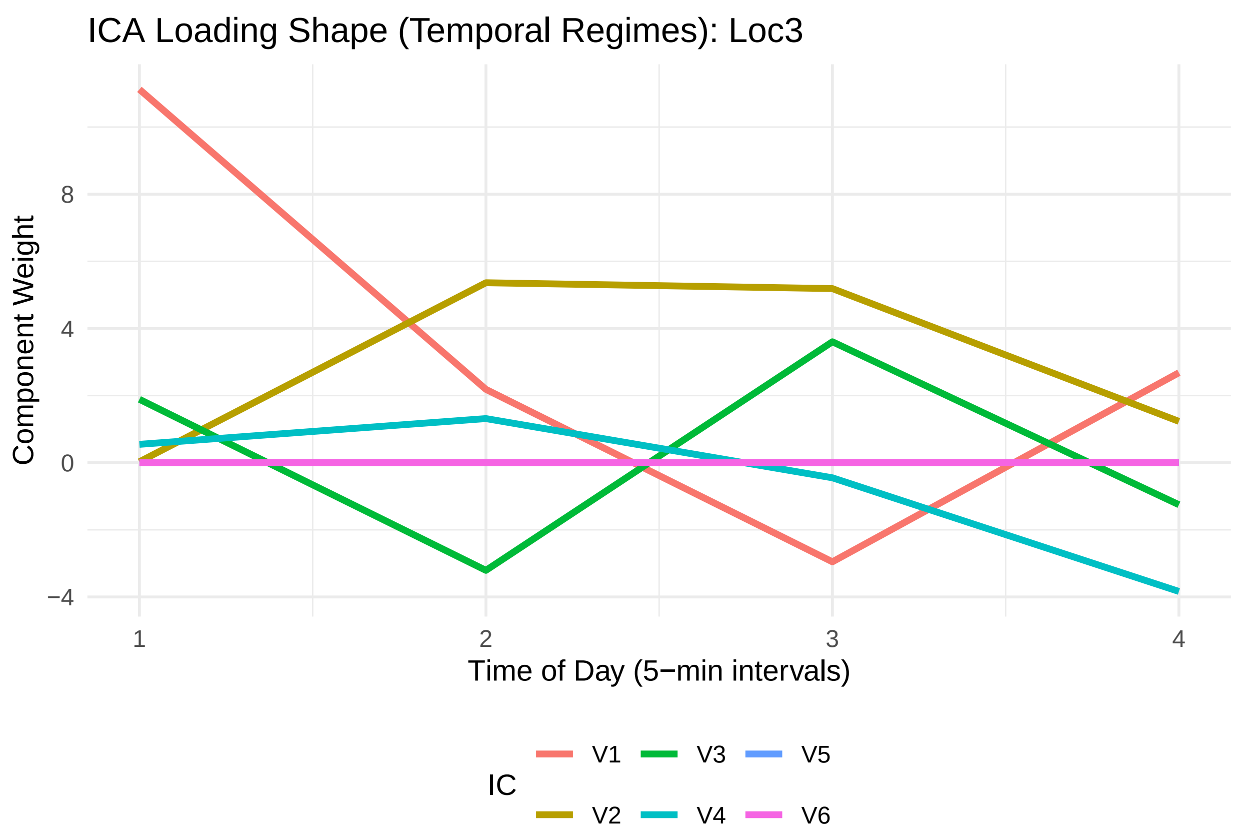
```r
      data.frame(Day = numeric(0), Method = character(0))
    }
  } else {
    ica_anomalies[[sheet]] <- data.frame(Day = numeric(0), Method = character(0))
  }
}

cat("\nICA anomaly detection complete.\n")
```

```
##
## ICA anomaly detection complete.
```

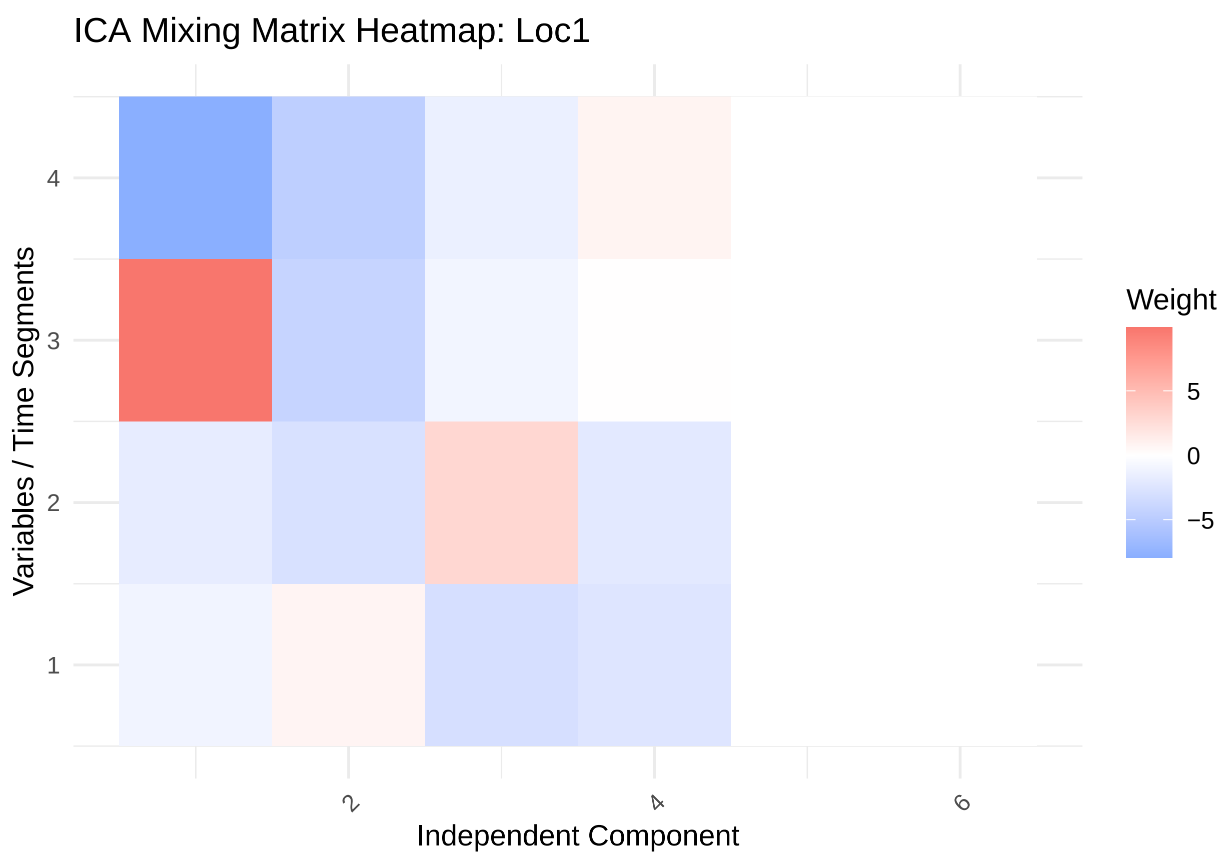
```r
print(head(ica_anomalies[[sheet_names[1]]]))
```

```
##   Day Method
## 1   1    ICA
## 2  21    ICA
## 3  67    ICA
## 4  76    ICA
## 5  89    ICA
## 6 150    ICA
```

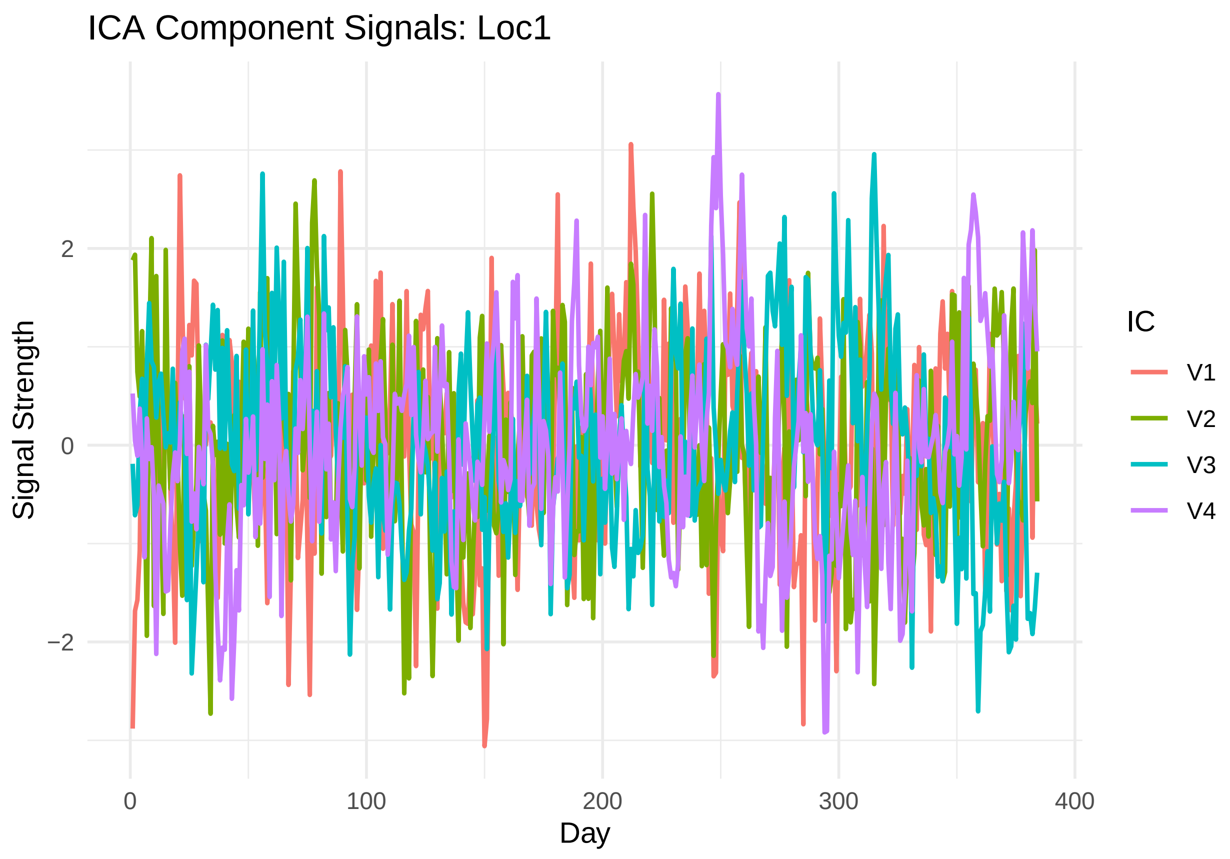
```r
plot_IC_loading_profiles <- function(sheet) {
  mixing_matrix <- as.data.frame(ica_results[[sheet]]$MixingMatrix)
  mixing_matrix$Time <- 1:nrow(mixing_matrix)

  mixing_long <- reshape2::melt(mixing_matrix, id.vars = "Time",
                                variable.name = "IC", value.name = "Loading")

  ggplot(mixing_long, aes(x = Time, y = Loading, color = IC)) +
    geom_line(linewidth = 1.2) +
    labs(title = paste("ICA Loading Shape (Temporal Regimes):", sheet),
         x = "Time of Day (5-min intervals)", y = "Component Weight") +
    theme_minimal() +
    theme(legend.position = "bottom")
}

plot_IC_loading_profiles(sheet_names[3])
```



```r
sheet <- sheet_names[1]
A <- ica_results[[sheet]]$MixingMatrix

A_df <- melt(A)
colnames(A_df) <- c("Variable", "IC", "Weight")

ggplot(A_df, aes(x = IC, y = Variable, fill = Weight)) +
  geom_tile() +
  scale_fill_gradient2(low = "#619CFF", mid = "white", high = "#F8766D") +
  labs(title = paste("ICA Mixing Matrix Heatmap:", sheet),
       x = "Independent Component",
       y = "Variables / Time Segments") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

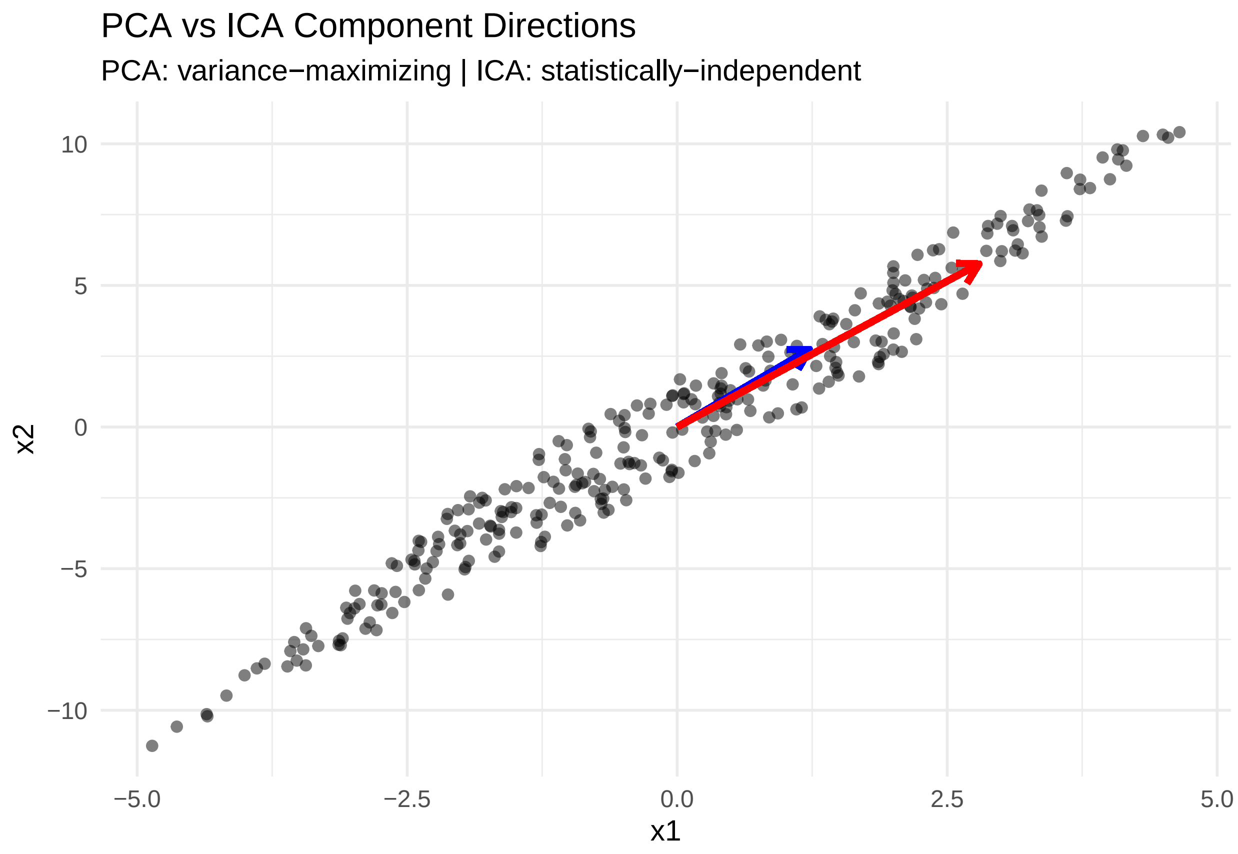


```r
valid_sheets <- names(ica_results)[sapply(ica_results, function(x) !is.null(x$Sources))]

if (length(valid_sheets) > 0) {

  sheet <- valid_sheets[1]
  ics <- as.data.frame(ica_results[[sheet]]$Sources)
  ics$Day <- 1:nrow(ics)

  # Reshape safely using pivot_longer - avoids melt errors
  ics_long <- ics %>%
    select(Day, starts_with("V")) %>%   # or rename IC1, IC2, etc.
    mutate(across(starts_with("V"), as.numeric)) %>%
    pivot_longer(cols = -Day,
                 names_to = "IC",
                 values_to = "Score")

  # Plot
  ggplot(ics_long, aes(x = Day, y = Score, color = IC)) +
    geom_line(size = 0.8) +
    labs(title = paste("ICA Component Signals:", sheet),
         x = "Day", y = "Signal Strength") +
    theme_minimal()
}
```



```r
n <- 300
Z <- matrix(runif(2*n, -sqrt(3), sqrt(3)), n, 2)
L <- matrix(c(1,2,3,4), 2, 2)
X <- scale(Z %*% L, center = TRUE, scale = FALSE)

pca <- prcomp(X)
ica <- fastICA(X, n.comp = 2)

df <- as.data.frame(X); colnames(df) <- c("x1","x2")

ggplot(df, aes(x1,x2)) +
  geom_point(alpha = 0.5) +
  geom_segment(aes(x=0, y=0, xend=pca$rotation[1,1]*3, yend=pca$rotation[2,1]*3),
               arrow=arrow(length=unit(0.3,"cm")), color="blue", size=1.2) +
  geom_segment(aes(x=0, y=0, xend=ica$A[1,1]*3, yend=ica$A[2,1]*3),
               arrow=arrow(length=unit(0.3,"cm")), color="red", size=1.2) +
  labs(title="PCA vs ICA Component Directions",
       subtitle="PCA: variance-maximizing | ICA: statistically-independent",
      x="x1", y="x2") +
  theme_minimal()
```

```
## Warning in geom_segment(aes(x = 0, y = 0, xend = pca$rotation[1, 1] * 3, : All aesthe
## i Please consider using `annotate()` or provide this layer with data containing
##   a single row.

## Warning in geom_segment(aes(x = 0, y = 0, xend = ica$A[1, 1] * 3, yend = ica$A[2, : A
## i Please consider using `annotate()` or provide this layer with data containing
##   a single row.
```



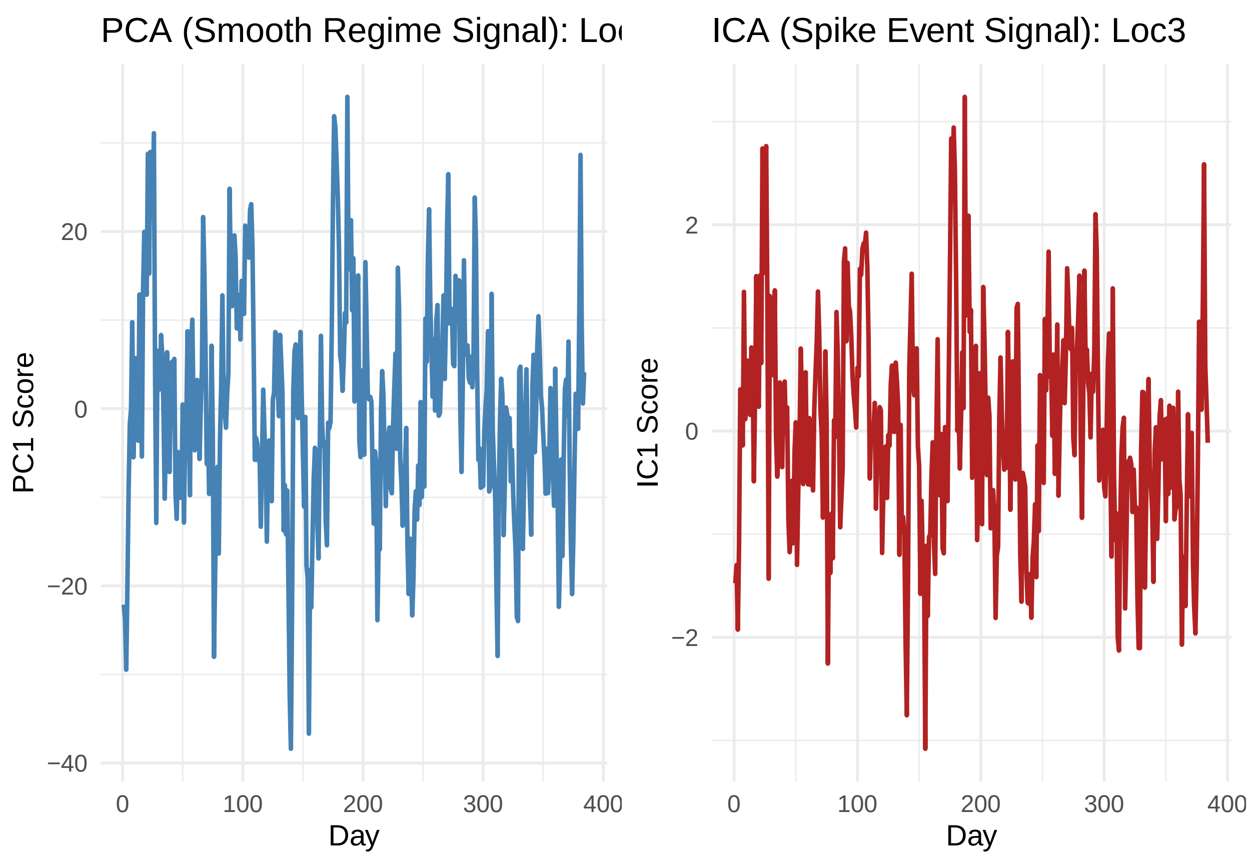
```r
# Merge PCA and ICA anomalies into a unified table
library(dplyr)

combined_anomalies <- list()

for (sheet in sheet_names) {

  # PCA anomalies (may include PC and Score)
  pca_df <- anomalies[[sheet]]
  # ICA anomalies (only Day + Method)
  ica_df <- ica_anomalies[[sheet]]

  # Standardize PCA columns
  if (!is.null(pca_df) && nrow(pca_df) > 0) {
    pca_df <- pca_df %>%
      mutate(Method = paste("PCA", PC)) %>%  # e.g., "PCA PC1"
      select(Day, Method, PC, Score)         # Keep consistent order
  }

  # Standardize ICA columns - add missing ones
  if (!is.null(ica_df) && nrow(ica_df) > 0) {
    ica_df <- ica_df %>%
      mutate(PC = NA, Score = NA) %>%        # Add missing columns
      select(Day, Method, PC, Score)         # Same column order
  }

  # Combine safely (works even if one is empty)
  combined <- bind_rows(pca_df, ica_df)

  # Sort by day
  combined <- combined[order(combined$Day), ]

  combined_anomalies[[sheet]] <- combined
}

# Show example
kable(head(combined_anomalies[[sheet_names[3]]], 10))
```

	Day	Method	PC	Score
1	3	PCA PC1	PC1	-29.45735
2	21	PCA PC1	PC1	28.77384
3	23	PCA PC1	PC1	28.97749
22	23	ICA	NA	NA
4	26	PCA PC1	PC1	31.09326
23	26	ICA	NA	NA
14	27	PCA PC3	PC3	15.06657
12	28	PCA PC2	PC2	25.43781
31	28	ICA	NA	NA
43	67	ICA	NA	NA

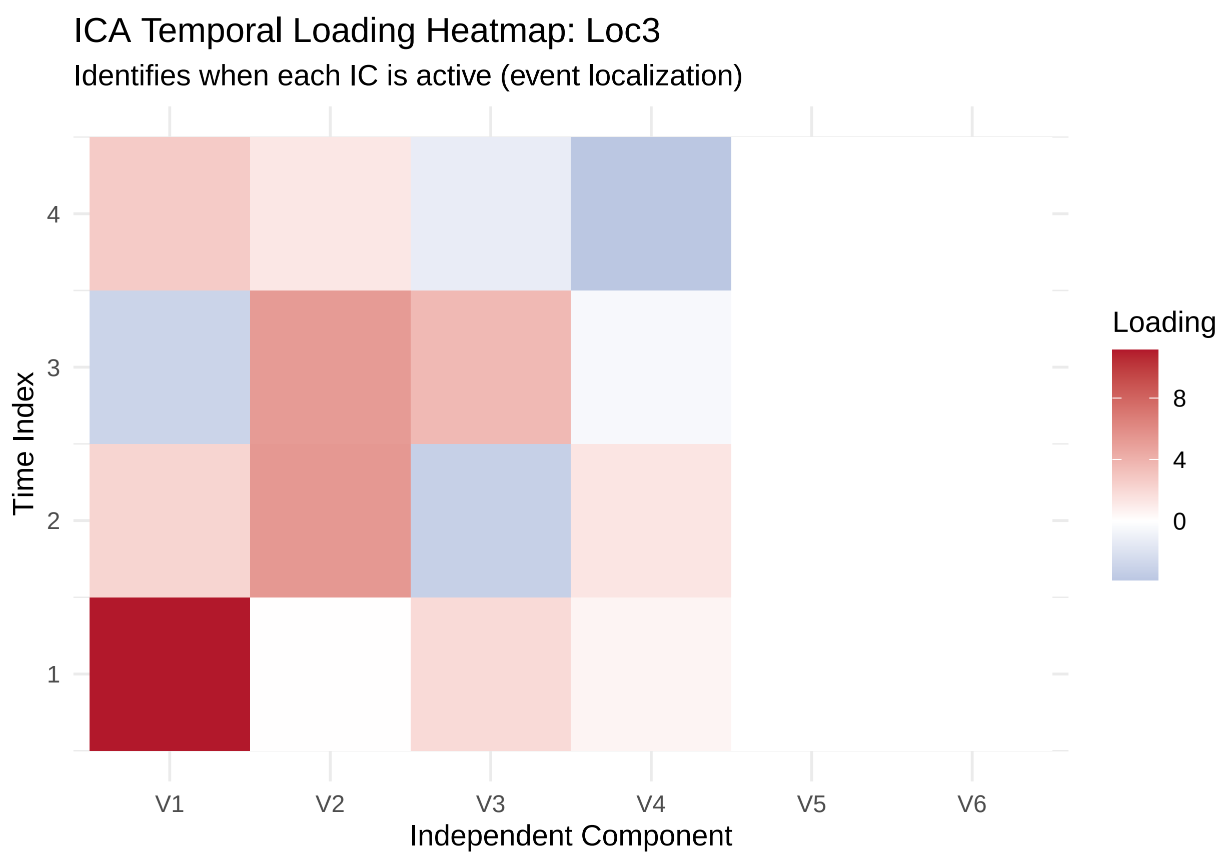
```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

```r
plot_PCA_vs_ICA <- function(sheet) {

  # PCA smoothed component (first PC)
  pca_scores <- as.data.frame(scores_list[[sheet]])
  pca_scores$Day <- 1:nrow(pca_scores)

  p_pca <- ggplot(pca_scores, aes(x = Day, y = PC1)) +
    geom_line(color = "steelblue", size = 0.8) +
    labs(title = paste("PCA (Smooth Regime Signal):", sheet),
         y = "PC1 Score", x = "Day") +
    theme_minimal()

  # ICA spike component (IC1)
  ics <- as.data.frame(ica_results[[sheet]]$Sources)
  ics$Day <- 1:nrow(ics)

  p_ica <- ggplot(ics, aes(x = Day, y = V1)) +
    geom_line(color = "firebrick", size = 0.8) +
    labs(title = paste("ICA (Spike Event Signal):", sheet),
         y = "IC1 Score", x = "Day") +
    theme_minimal()

  grid.arrange(p_pca, p_ica, ncol = 2)
}

plot_PCA_vs_ICA(sheet_names[3])  # pick one location
```



```r
sheet <- sheet_names[3]  # Example location

A <- as.data.frame(ica_results[[sheet]]$MixingMatrix)
A$Time <- 1:nrow(A)

A_long <- tidyr::pivot_longer(A,
                              cols = -Time,
                              names_to = "IC",
                              values_to = "Loading")

ggplot(A_long, aes(x = IC, y = Time, fill = Loading)) +
  geom_tile() +
  scale_fill_gradient2(low = "#2166AC", mid = "white", high = "#B2182B") +
  labs(title = paste("ICA Temporal Loading Heatmap:", sheet),
       subtitle = "Identifies when each IC is active (event localization)",
       x = "Independent Component", y = "Time Index") +
  theme_minimal()
```



ICA on not PC loadings

```
cat("\n## ICA Analysis: All 26 Locations\n\n")
```

```
##
## ## ICA Analysis: All 26 Locations
```

```
# Initialize storage for ICA results
ica_results_direct <- list()

for (sheet in sheet_names) {
  cat("Location:", sheet, "\n")

  # 1. Load the RAW data matrix for the current sheet, mimicking PCA data access
  X_raw <- data_transposed[[sheet]]

  # 2. Get the number of components determined by the PCA's 90% variance threshold
  # This uses the same dimensionality reduction level as your PCA.
  n_comp <- ncol(scores_list[[sheet]])
  P <- ncol(X_raw)

  # Safety check: Ensure the data is valid and has enough dimensions
  if (is.null(X_raw) || nrow(X_raw) == 0 || P < 3) {
      cat("  Skipping", sheet, "- Data is NULL, empty, or has < 3 columns.\n")
      next
  }

  # Safety check: Ensure the determined n_comp is feasible
  if (n_comp < 2 || n_comp > P) {
      n_comp_used <- min(4, P) # Fallback if PCA result is odd
      cat(paste("  Warning: PCA n_comp was unusual. Using fallback:", n_comp_used, "comp
  } else {
      n_comp_used <- n_comp
      cat("  Using PCA-determined dimensionality:", n_comp_used, "components.\n")
  }

  # 3. Center the data
  # fastICA will perform the required whitening/scaling internally.
  X_centered <- scale(X_raw, center = TRUE, scale = FALSE)

  # 4. Run fastICA with the centered raw data
  # The n.comp argument handles the dimensionality reduction (PCA) internally
  ica <- fastICA(
    X_centered,
    n.comp = n_comp_used,
    method = "C" # Symmetric/parallel algorithm
  )

  # 5. Store the results (S: Independent Sources / A: Mixing Matrix)
  ica_results_direct[[sheet]] <- list(
    Sources = ica$S,
    MixingMatrix = ica$A
  )

  cat("  ICA successful. Extracted:", ncol(ica$S), "components.\n")
}
```

```
## Location: Loc1
##   Using PCA-determined dimensionality: 6 components.
##   ICA successful. Extracted: 6 components.
## Location: Loc2
##   Using PCA-determined dimensionality: 7 components.
##   ICA successful. Extracted: 7 components.
## Location: Loc3
##   Using PCA-determined dimensionality: 6 components.
##   ICA successful. Extracted: 6 components.
## Location: Loc4
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc5
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc6
##   Using PCA-determined dimensionality: 7 components.
##   ICA successful. Extracted: 7 components.
## Location: Loc7
##   Using PCA-determined dimensionality: 2 components.
##   ICA successful. Extracted: 2 components.
## Location: Loc8
##   Using PCA-determined dimensionality: 10 components.
##   ICA successful. Extracted: 10 components.
## Location: Loc9
##   Using PCA-determined dimensionality: 6 components.
##   ICA successful. Extracted: 6 components.
## Location: Loc10
##   Using PCA-determined dimensionality: 6 components.
##   ICA successful. Extracted: 6 components.
## Location: Loc11
##   Using PCA-determined dimensionality: 10 components.
##   ICA successful. Extracted: 10 components.
## Location: Loc12
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc13
##   Using PCA-determined dimensionality: 9 components.
##   ICA successful. Extracted: 9 components.
## Location: Loc14
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc15
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc16
##   Using PCA-determined dimensionality: 9 components.
##   ICA successful. Extracted: 9 components.
## Location: Loc17
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc18
##   Using PCA-determined dimensionality: 6 components.
##   ICA successful. Extracted: 6 components.
## Location: Loc19
##   Using PCA-determined dimensionality: 10 components.
##   ICA successful. Extracted: 10 components.
## Location: Loc20
##   Using PCA-determined dimensionality: 7 components.
##   ICA successful. Extracted: 7 components.
## Location: Loc21
##   Using PCA-determined dimensionality: 6 components.
##   ICA successful. Extracted: 6 components.
## Location: Loc22
##   Using PCA-determined dimensionality: 7 components.
##   ICA successful. Extracted: 7 components.
## Location: Loc23
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc24
##   Using PCA-determined dimensionality: 4 components.
##   ICA successful. Extracted: 4 components.
## Location: Loc25
##   Using PCA-determined dimensionality: 8 components.
##   ICA successful. Extracted: 8 components.
## Location: Loc26
##   Using PCA-determined dimensionality: 3 components.
##   ICA successful. Extracted: 3 components.
```

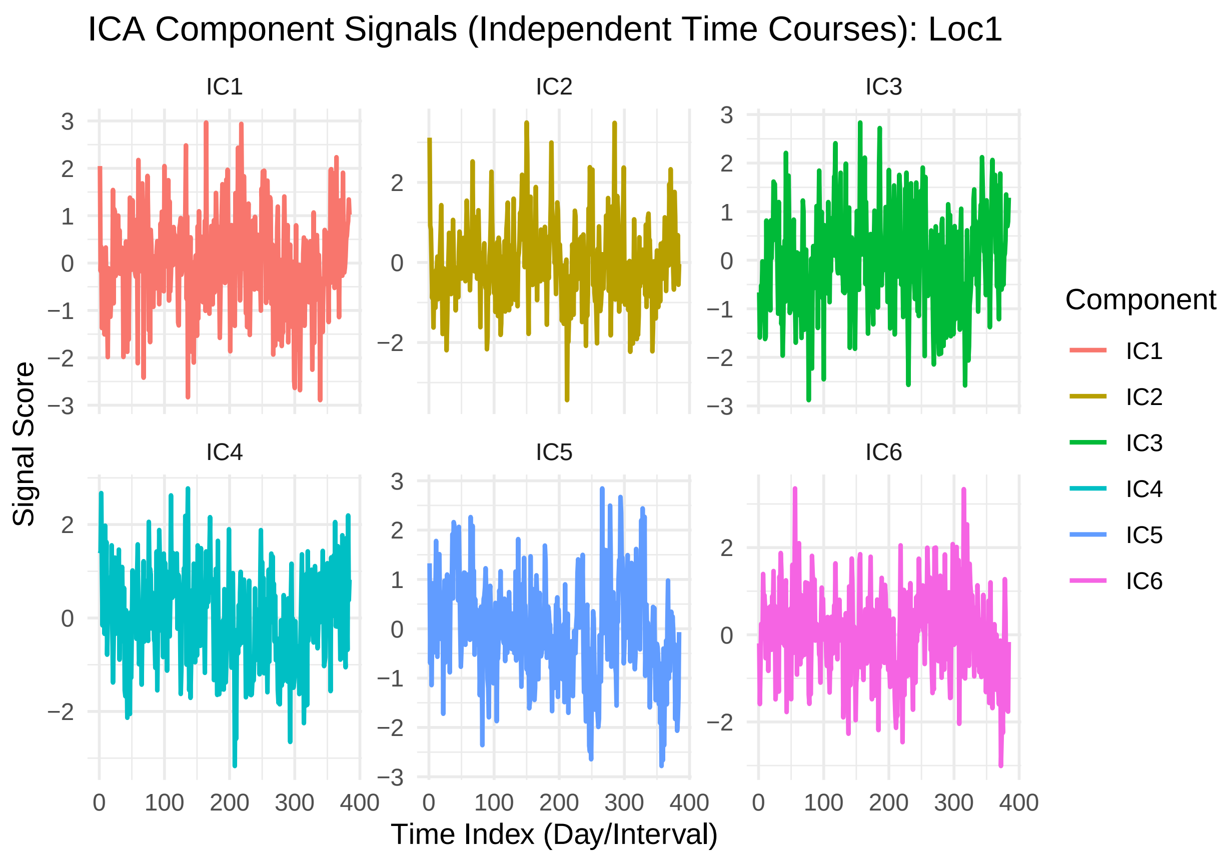
```
cat("\nICA completed for all valid locations.\n")
```

```
##
## ICA completed for all valid locations.
```

```
plot_IC_time_series <- function(ica_results_list, sheet) {

  if (is.null(ica_results_list[[sheet]])) return(NULL)

  # Extract the Sources (S matrix)
  ics <- as.data.frame(ica_results_list[[sheet]]$Sources)
  colnames(ics) <- paste0("IC", 1:ncol(ics))
  ics$Time <- 1:nrow(ics)

  # Reshape to long format for ggplot
  ics_long <- ics %>%
    pivot_longer(cols = starts_with("IC"),
                 names_to = "Component",
                 values_to = "Score")

  # Plot
  ggplot(ics_long, aes(x = Time, y = Score, color = Component)) +
    geom_line(size = 0.8) +
    labs(title = paste("ICA Component Signals (Independent Time Courses):", sheet),
         x = "Time Index (Day/Interval)", y = "Signal Score") +
    theme_minimal() +
    facet_wrap(~Component, scales = "free_y")
}

cat("\n## Example: Independent Component Time Series\n")
```

```
##
## ## Example: Independent Component Time Series
```

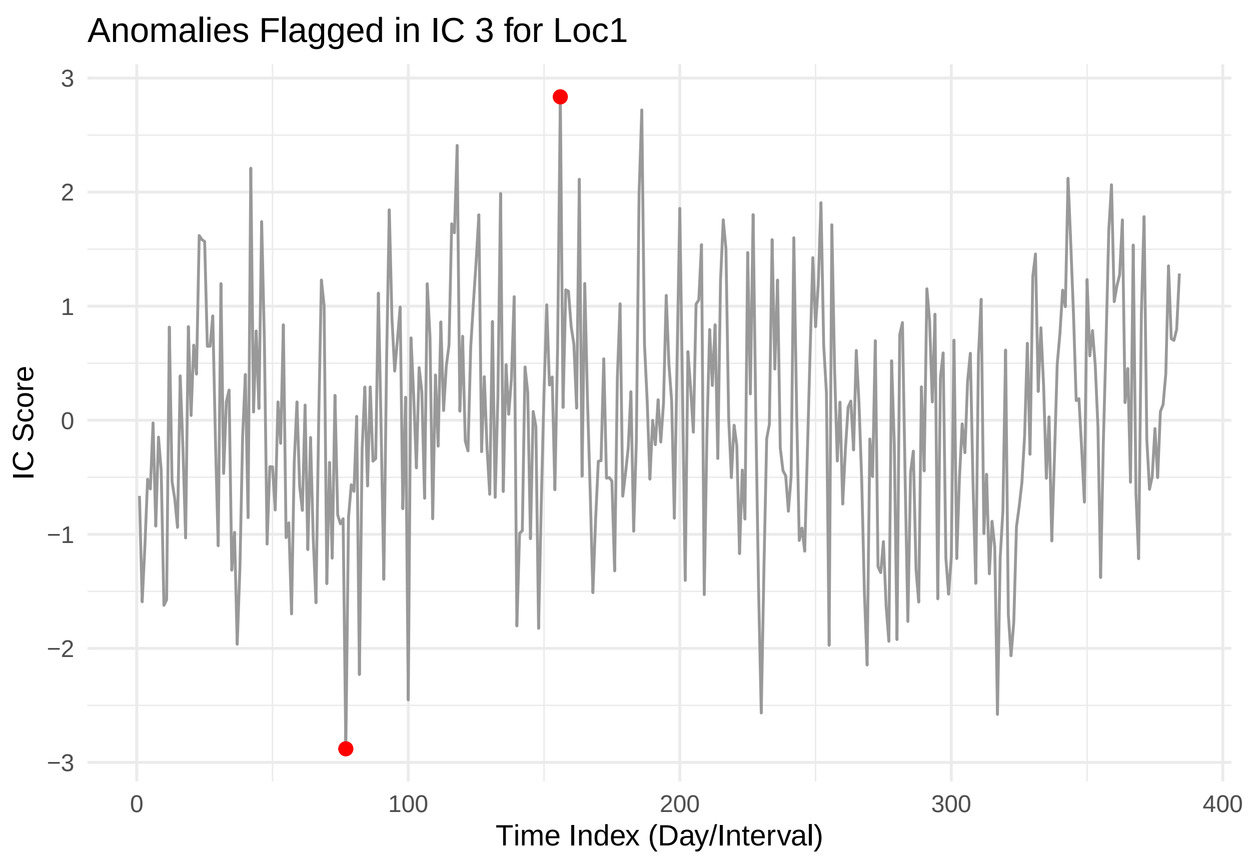
```
plot_IC_time_series(ica_results_direct, sheet_names[1])
```



```r
plot_IC_anomalies <- function(ica_results_list, sheet, ic_to_plot = 1) {

  if (is.null(ica_results_list[[sheet]])) return(NULL)

  # Extract the target IC
  ics <- as.data.frame(ica_results_list[[sheet]]$Sources)
  if (ic_to_plot > ncol(ics)) ic_to_plot <- 1 # Fallback to IC1

  target_ic_name <- paste0("IC", ic_to_plot)
  target_ic <- ics[, ic_to_plot]

  # IQR Anomaly Detection (1.5 * IQR Rule)
  q1 <- quantile(target_ic, 0.25)
  q3 <- quantile(target_ic, 0.75)
  iqr <- q3 - q1
  lower <- q1 - 1.5 * iqr
  upper <- q3 + 1.5 * iqr

  # Create a data frame for plotting
  plot_data <- data.frame(
    Time = 1:length(target_ic),
    Score = target_ic,
    Anomaly = ifelse(target_ic < lower | target_ic > upper, target_ic, NA)
  )

  # Plot
  ggplot(plot_data, aes(x = Time, y = Score)) +
    geom_line(color = "gray60") +
    geom_point(data = dplyr::filter(plot_data, !is.na(Anomaly)),
               aes(y = Anomaly), color = "red", size = 2) +
    labs(title = paste("Anomalies Flagged in IC", ic_to_plot, "for", sheet),
         x = "Time Index (Day/Interval)", y = "IC Score") +
    theme_minimal()
}

cat("\n## Example: IC Anomaly Detection Plot\n")
```

```
##
## ## Example: IC Anomaly Detection Plot
```

```r
plot_IC_anomalies(ica_results_direct, sheet_names[1], ic_to_plot = 3)
```



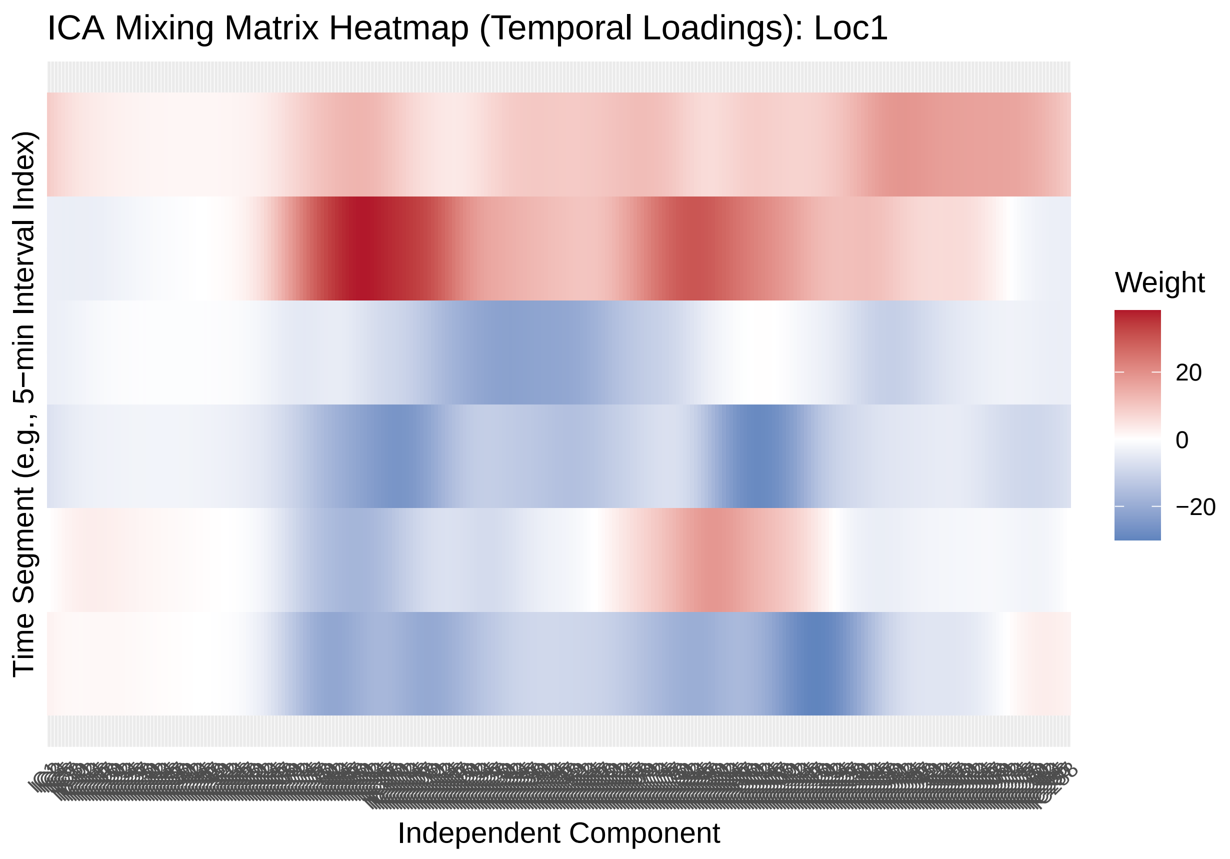
```r
library(ggplot2)
library(reshape2)
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
plot_mixing_matrix_heatmap <- function(ica_results_list, sheet) {

  if (is.null(ica_results_list[[sheet]])) return(NULL)

  A <- as.data.frame(ica_results_list[[sheet]]$MixingMatrix)
  colnames(A) <- paste0("IC", 1:ncol(A))

  # Assuming rows of A are the 288 time segments (0 to 287)
  A$Time_Segment <- 1:nrow(A)

  # Reshape to long format for ggplot
  A_df <- melt(A, id.vars = "Time_Segment",
               variable.name = "IC",
               value.name = "Weight")
  ggplot(A_df, aes(x = IC, y = Time_Segment, fill = Weight)) +
    geom_tile() +
    scale_fill_gradient2(low = "#2166AC", mid = "white", high = "#B2182B", name = "Weigh
    labs(title = paste("ICA Mixing Matrix Heatmap (Temporal Loadings):", sheet),
         x = "Independent Component",
         y = "Time Segment (e.g., 5-min Interval Index)") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
          axis.text.y = element_blank()) # Hide y-axis labels for 288 segments
}

cat("\n## Example: Mixing Matrix Heatmap\n")
```

```
##
## ## Example: Mixing Matrix Heatmap
```

```
plot_mixing_matrix_heatmap(ica_results_direct, sheet_names[1])
```


ICA Mixing Matrix Heatmap (Temporal Loadings): Loc1
Time Segment (e.g., 5−min Interval Index)
Independent Component
Weight