TaxiDataR_GroupProj

Albert_Hakobyan2

Temporal Heatmap of Taxi Activity Patterns

```r
# Load necessary libraries
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
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```r
library(ggplot2)

# Read the data
taxi_data <- read.csv("TaxiData-Sample.csv", header = FALSE)
names(taxi_data) <- c("VehicleNum", "Time", "Lng", "Lat", "OpenStatus", "Speed")

# Processing time data with lubridate
taxi_data <- taxi_data %>%
  mutate(
    DateTime = as.POSIXct(Time, format = "%H:%M:%S"),
    Hour = hour(DateTime),
    Minute = minute(DateTime)
  )

# Create hourly activity aggregation
```

```r
hourly_activity <- taxi_data %>%
  group_by(Hour, OpenStatus) %>%
  summarise(
    TaxiCount = length(unique(VehicleNum)),
    PointCount = n(),
    AvgSpeed = mean(Speed, na.rm = TRUE),
    .groups = "drop"
  )

# Plot heatmap of taxi activity by hour and passenger status
ggplot(hourly_activity, aes(x = Hour, y = factor(OpenStatus), fill = PointCount)) +
  geom_tile() +
  scale_fill_gradient2(
    low = "blue", mid = "turquoise", high = "yellow",
    midpoint = median(hourly_activity$PointCount),
    name = "GPS Points"
  ) +
  scale_x_continuous(breaks = 0:23) +
  scale_y_discrete(labels = c("Empty", "With Passenger")) +
  labs(
    x = "Hour of Day",
    y = "Passenger Status"
  ) +
  theme_bw() +
  theme(
    plot.title = element_text(face = "bold"),
    legend.position = "right",
    panel.grid.major = element_line(color = "gray90"),
    panel.grid.minor = element_blank()
  )
```

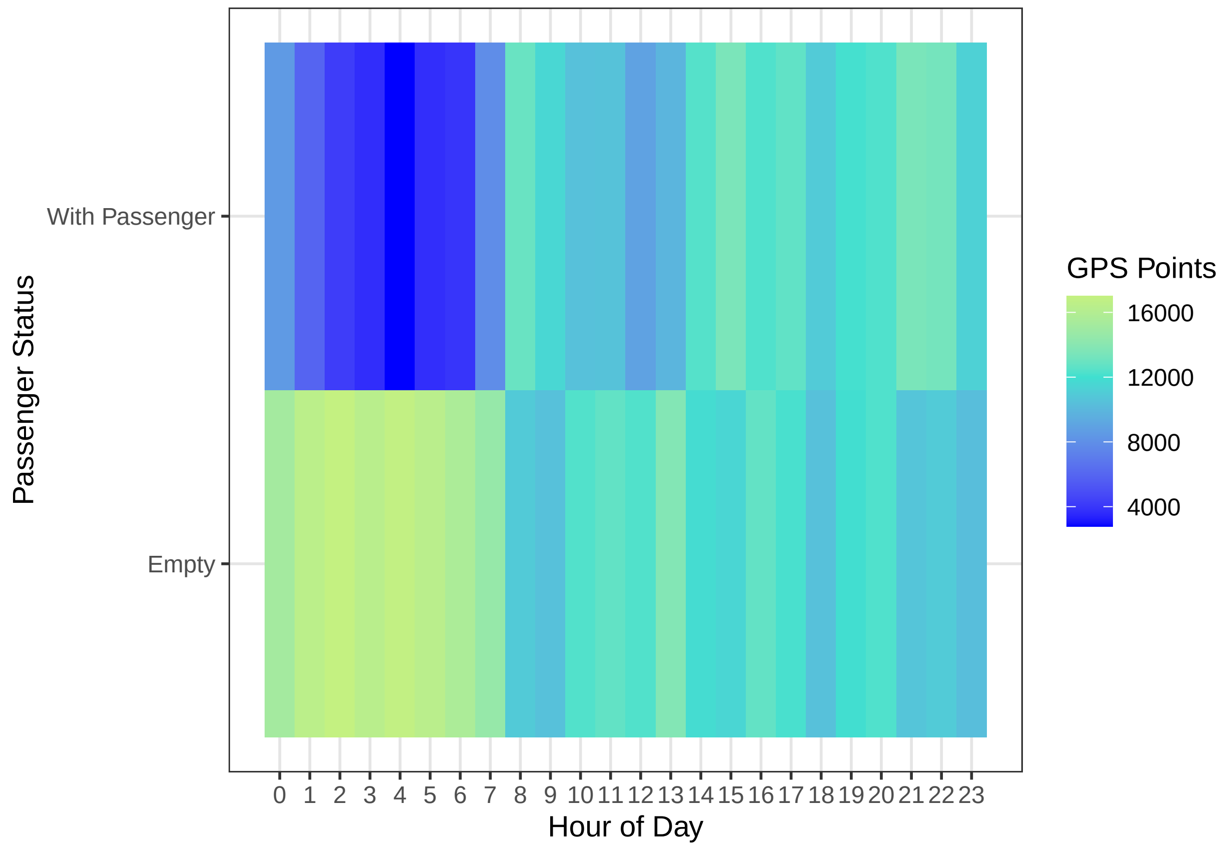


Figure 6 shows a temporal heatmap of taxi activity based on GPS points frequency by hour and passenger status. The x-axis shows hours (0-23), and the y-axis indicates whether taxis are empty or have passengers. Blue areas indicate low activity while green-yellow areas show high activity.

Spatial Analysis of Pickup and Dropoff Transitions

```r
taxi_data <- read.csv("TaxiData-Sample.csv", header = FALSE)
names(taxi_data) <- c("VehicleNum", "Time", "Lng", "Lat", "OpenStatus", "Speed")

# Sort data for status change detection
taxi_data <- taxi_data %>%
  arrange(VehicleNum, Time)

# Identify status changes with dplyr
status_changes <- taxi_data %>%
  group_by(VehicleNum) %>%
  mutate(
    PrevStatus = lag(OpenStatus),
    StatusChange = case_when(
      is.na(PrevStatus) ~ "None",
      OpenStatus == 1 & PrevStatus == 0 ~ "Pickup",
      OpenStatus == 0 & PrevStatus == 1 ~ "Dropoff",
      TRUE ~ "None")) %>%
  filter(StatusChange %in% c("Pickup", "Dropoff")) %>%
  ungroup()

# Create density map with hexbins
ggplot(status_changes, aes(x = Lng, y = Lat)) +
  stat_bin_hex(aes(fill = ..count..), bins = 50) +
  facet_wrap(~StatusChange, ncol = 2) +
  scale_fill_gradient(low = "darkred", high = "yellow", trans = "log10", name = "Count") +
  labs(x = "Longitude", y = "Latitude") +
  theme_bw() +
  theme(plot.title = element_text(face = "bold"),
   legend.position = "right",
        panel.grid = element_line(color = "gray"),
        strip.text = element_text(face = "bold"))
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

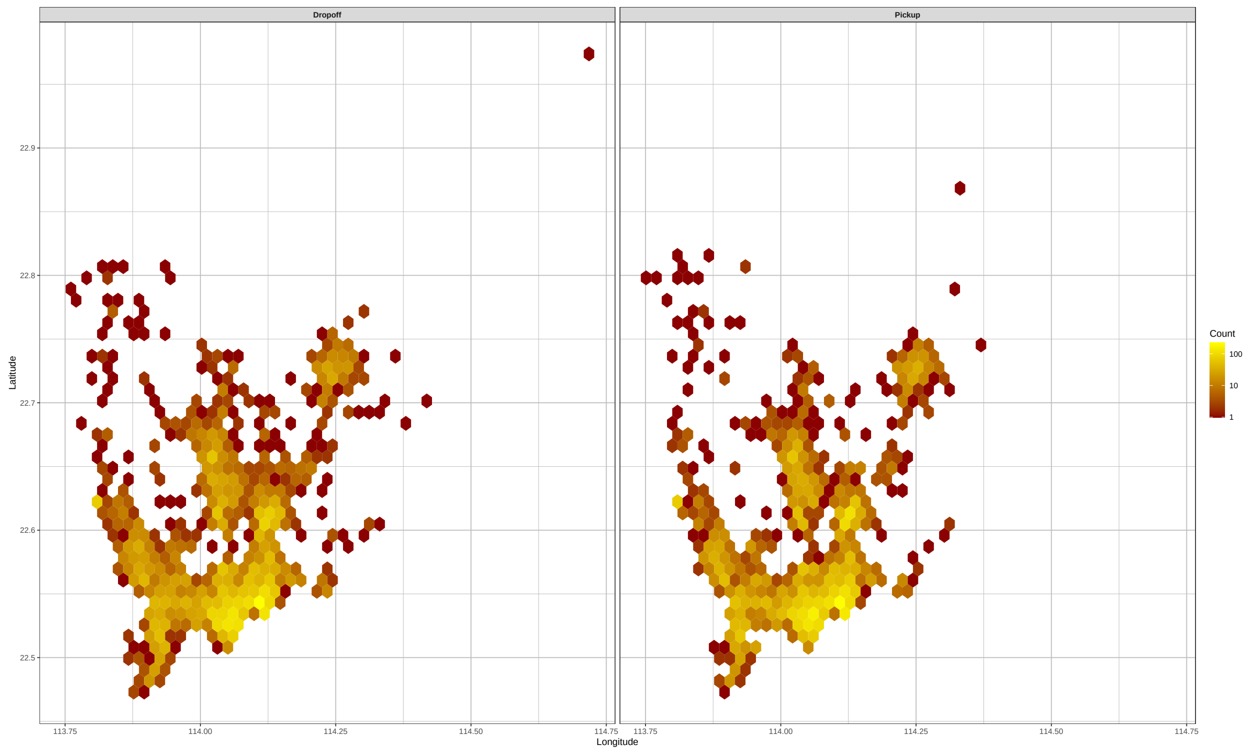


Figure 7 visualizes the spatial distribution of taxi pickup and dropoff transitions using hexagonal bins. The color intensity represents density, with yellow showing lower concentration and red showing higher concentrations. This visualization identifies high-traffic areas for taxi pickup and dropoffs across Shanghai.

Comparative Analysis of Pickup and Dropoff Patterns

```r
# Create spatial bins
status_changes$lng_bin <- cut(status_changes$Lng,
                              breaks = seq(min(status_changes$Lng), max(status_changes$Lng), length.out =
                              include.lowest = TRUE)
status_changes$lat_bin <- cut(status_changes$Lat,
                              breaks = seq(min(status_changes$Lat), max(status_changes$Lat), length.out =
                              include.lowest = TRUE)

# Count events by type in each bin
bin_counts <- status_changes %>%
  group_by(lng_bin, lat_bin, StatusChange) %>%
  summarise(count = n(), .groups = "drop") %>%
  tidyr::pivot_wider(
    names_from = StatusChange,
    values_from = count,
    values_fill = list(count = 0)
  ) %>%
  mutate(
    total = Pickup + Dropoff,
    ratio = (Pickup - Dropoff) / (Pickup + Dropoff),
    # Calculate bin centers for plotting
    lng_center = as.numeric(substr(as.character(lng_bin), 2,
                                   regexpr(",", as.character(lng_bin)) - 1)),
    lat_center = as.numeric(substr(as.character(lat_bin), 2,
                                   regexpr(",", as.character(lat_bin)) - 1))) %>%
  filter(total > 5)  # Filter out low-count bins for clarity


ggplot(bin_counts, aes(x = lng_center, y = lat_center, fill = ratio, size = total)) +
  geom_point(shape = 21, color = "white", alpha = 0.8) +
  scale_fill_gradient2(
    low = "black",
    mid = "turquoise",
    high = "darkred",
    midpoint = 0,
    name = "Pickup vs. Dropoff\nRatio",
    limits = c(-1, 1)
  ) +
  scale_size_continuous(
    range = c(1, 8),
    name = "Total Events"
  ) +
  labs(
    x = "Longitude",
    y = "Latitude"
  ) +
  theme_bw() +
  theme(
    plot.title = element_text(face = "bold"),
    legend.position = "right",
    panel.grid.minor = element_blank()
)
```

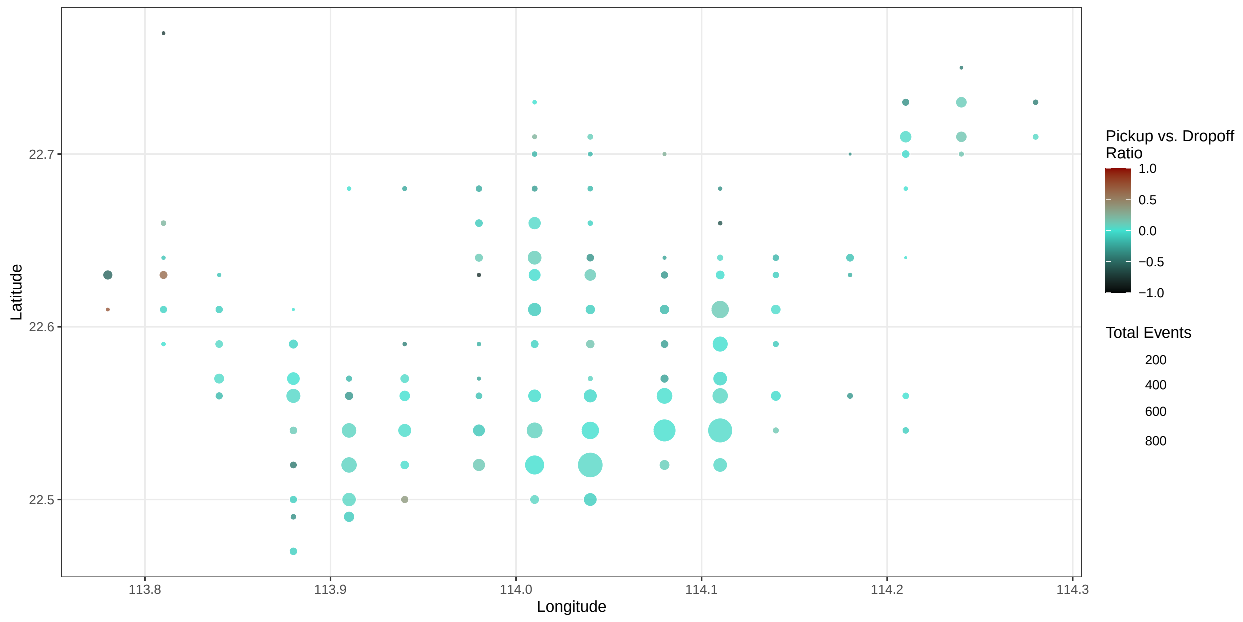


Figure 8 enhances the analysis by binning data into longitude and latitude intervals to visualize the ratio of pickups to dropoffs. The color scale shows blue areas with more dropoffs, red areas with more pickups, and turquoise areas with balanced activity. Circle size indicates total activity volume in each location.

Speed Distribution Analysis by Time Period

```r
taxi_data <- read.csv("TaxiData-Sample.csv", header = FALSE)
names(taxi_data) <- c("VehicleNum", "Time", "Lng", "Lat", "OpenStatus", "Speed")

# Processing time with lubridate
taxi_data <- taxi_data %>%
  mutate(
    DateTime = as.POSIXct(Time, format = "%H:%M:%S"),
    Hour = hour(DateTime),
    # Create time periods
    TimePeriod = case_when(
      Hour >= 6 & Hour < 10 ~ "Morning Rush (6-10)",
      Hour >= 10 & Hour < 16 ~ "Midday (10-16)",
      Hour >= 16 & Hour < 20 ~ "Evening Rush (16-20)",
      TRUE ~ "Night (20-6)"
    ),
    # Creating passenger status label
    PassengerStatus = ifelse(OpenStatus == 1, "With Passenger", "Empty")
  ) %>%
  # Only including moving vehicles for speed analysis
  filter(Speed > 0)
```

```r
# Ordering time periods correctly for visualization
taxi_data$TimePeriod <- factor(taxi_data$TimePeriod,
                              levels = c("Morning Rush (6-10)",
                                         "Midday (10-16)",
                                         "Evening Rush (16-20)",
                                         "Night (20-6)"))

# Create violin plot with embedded boxplot for detailed distribution analysis
ggplot(taxi_data, aes(x = TimePeriod, y = Speed, fill = PassengerStatus)) +
  geom_violin(position = position_dodge(width = 0.8), alpha = 0.7, trim = TRUE, scale = "width") +
  geom_boxplot(position = position_dodge(width = 0.8), width = 0.2, alpha = 0.4, outlier.shape = NA) +
  coord_flip() +
  scale_fill_manual(values = c("With Passenger" = "#1B9E77", "Empty" = "#D95F02")) +
  scale_y_continuous(limits = c(0, 80), breaks = seq(0, 80, by = 10)) +
  labs(
    title = "Taxi Speed Distributions by Time of Day",
    subtitle = "Comparing speeds when empty vs. with passengers",
    y = "Speed (km/h)",
    x = "Time Period",
    fill = "Status"
  ) +
  theme_bw() +
  theme(
    plot.title = element_text(face = "bold"),
    legend.position = "top",
    panel.grid.minor = element_blank(),
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_line(color = "gray"),
    axis.text.y = element_text(face = "bold")
  )
```

```
## Warning: Removed 8038 rows containing non-finite outside the scale range
## (‘stat_ydensity()‘).

## Warning: Removed 8038 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```

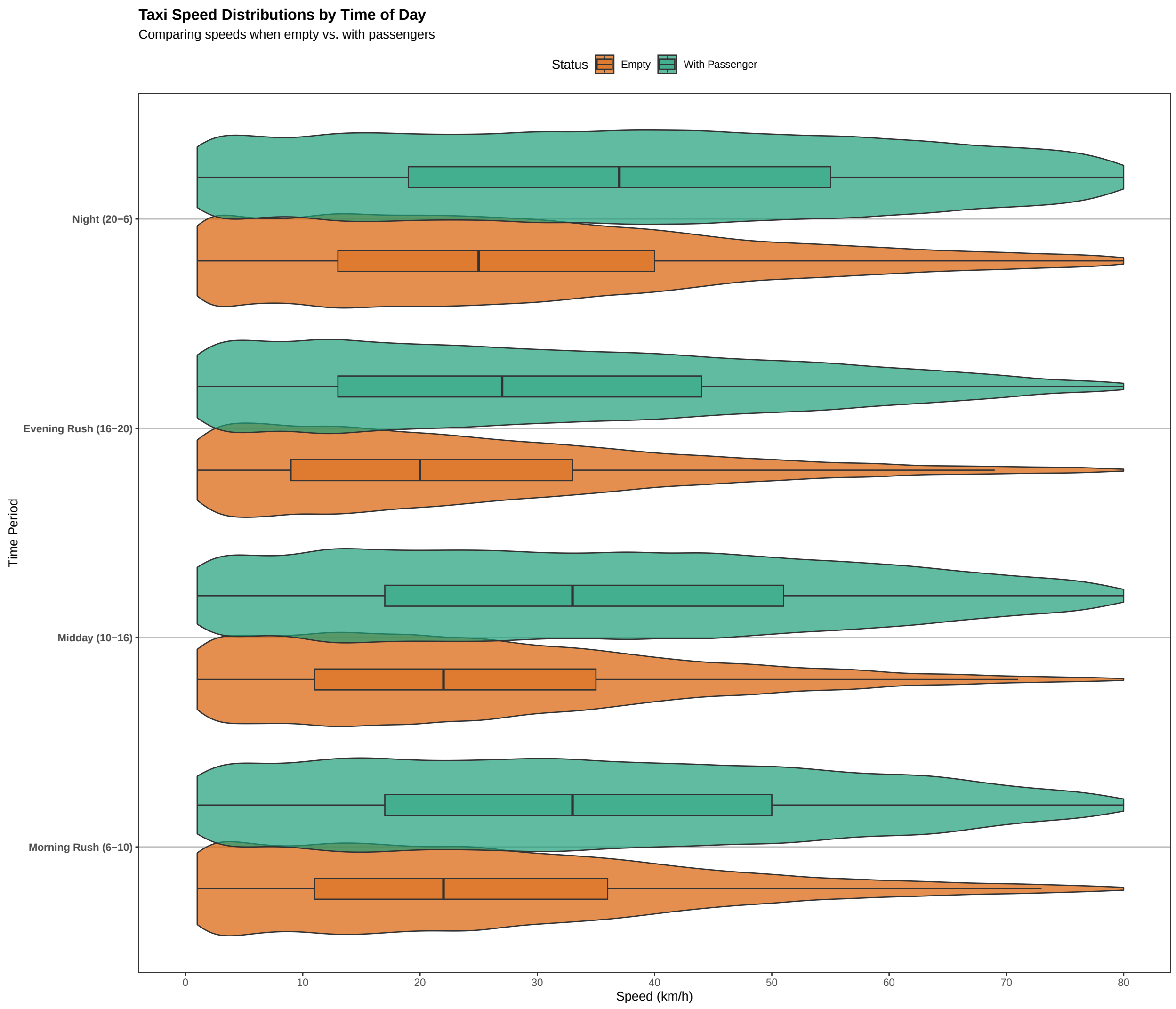


Figure 9 analyzes taxi speed distribution by time period, with the dataset divided into four time groups: Morning Rush (6-10), Midday (10-16), Evening Rush (16-20), and Night (20-6). The violin plot with embedded boxplots shows speed variations between empty taxis and those with passengers. This reveals how factors like traffic conditions and passenger status influence speeds throughout the day.

Urban Regions Speed Analysis

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

# Read taxi data
taxi_data <- read.csv("TaxiData-Sample.csv", header = FALSE)
names(taxi_data) <- c("VehicleNum", "Time", "Lng", "Lat", "OpenStatus", "Speed")

# Filter moving vehicles
moving_taxis <- taxi_data %>% filter(Speed > 0)
```

```r
# Define regions by quantiles
lng_breaks <- quantile(moving_taxis$Lng, probs = c(0, 0.33, 0.67, 1))
lat_breaks <- quantile(moving_taxis$Lat, probs = c(0, 0.33, 0.67, 1))

# Assign regions
moving_taxis$region_name <- NA
for (i in 1:3) {
  for (j in 1:3) {
    lng_min <- lng_breaks[j]
    lng_max <- lng_breaks[j+1]
    lat_min <- lat_breaks[i]
    lat_max <- lat_breaks[i+1]

    lat_label <- c("South", "Central", "North")[4-i]
    lng_label <- c("West", "Central", "East")[j]
    region_label <- paste(lat_label, lng_label)

    idx <- which(moving_taxis$Lng >= lng_min & moving_taxis$Lng <= lng_max &
                 moving_taxis$Lat >= lat_min & moving_taxis$Lat <= lat_max)

    if (length(idx) > 0) {
      moving_taxis$region_name[idx] <- region_label
    }
  }
}

# Plot
p1 <- ggplot(moving_taxis %>% filter(!is.na(region_name)),
       aes(x = reorder(region_name, Speed, median), y = Speed)) +
  geom_boxplot(aes(fill = reorder(region_name, Speed, median)),
               alpha = 0.7, outlier.shape = NA) +
  coord_flip() +
  coord_cartesian(ylim = c(0, quantile(moving_taxis$Speed, 0.95))) +
  scale_fill_viridis_d(option = "plasma", guide = "none") +
  labs(

    x = "Region",
    y = "Speed (km/h)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold"),
    axis.text.x = element_text(angle = 45, vjust = 0.5, hjust = 1, size = 7)
  )
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```r
# Save with fixed size
ggsave("urban_regions_boxplot.png", p1, width = 8, height = 6, dpi = 300)
```

Figure 10(a) analyzes taxi speed distribution in different urban regions of Shanghai by dividing the city into nine areas (three latitudinal × three longitudinal). The boxplots show speed variations by region, with the 95th percentile cap excluding extreme outliers. This visualization helps identify which areas experience higher or lower speeds, indicating varying levels of congestion.

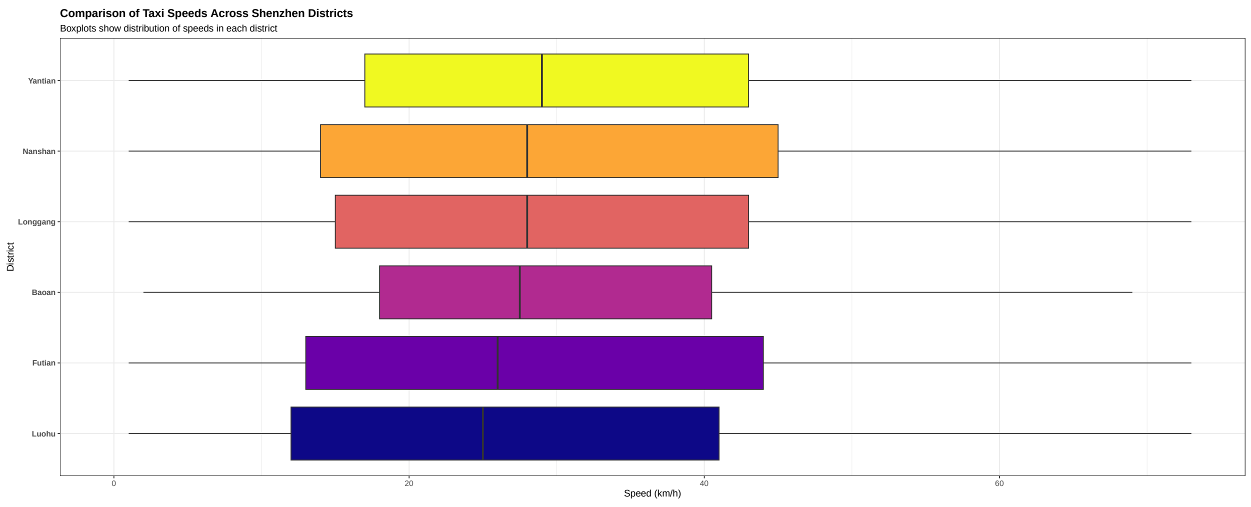
Shenzhen Districts Speed Analysis

```r
library(dplyr)
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```r
library(ggplot2)

# Read taxi data
taxi_data <- read.csv("TaxiData-Sample.csv", header = FALSE)
names(taxi_data) <- c("VehicleNum", "Time", "Lng", "Lat", "OpenStatus", "Speed")

# Filter for moving vehicles only
moving_taxis <- taxi_data %>%
  filter(Speed > 0)

# Convert taxi data to sf object
taxi_sf <- st_as_sf(moving_taxis, coords = c("Lng", "Lat"), crs = 4326)

# This would be where you load actual district boundaries
# For example:
# districts_sf <- st_read("shenzhen_districts.geojson")
# or:
# districts_sf <- st_read("shenzhen_districts.shp")

# Instead, creating a dummy example:
# (In a real implementation, you would use actual district shapefiles)
districts_sf <- data.frame(
  district_name = c("Futian", "Luohu", "Nanshan", "Yantian", "Baoan", "Longgang"),
  geometry = c(
    "POLYGON((114.0 22.5, 114.1 22.5, 114.1 22.6, 114.0 22.6, 114.0 22.5))",
    "POLYGON((114.1 22.5, 114.2 22.5, 114.2 22.6, 114.1 22.6, 114.1 22.5))",
    "POLYGON((113.8 22.5, 114.0 22.5, 114.0 22.6, 113.8 22.6, 113.8 22.5))",
    "POLYGON((114.2 22.5, 114.4 22.5, 114.4 22.7, 114.2 22.7, 114.2 22.5))",
    "POLYGON((113.7 22.5, 113.8 22.5, 113.8 22.7, 113.7 22.7, 113.7 22.5))",
    "POLYGON((114.1 22.6, 114.3 22.6, 114.3 22.8, 114.1 22.8, 114.1 22.6))"))

# Convert to sf
districts_sf <- sf::st_as_sf(districts_sf, wkt = "geometry", crs = 4326)

# Perform spatial join to assign districts to points
taxi_districts <- st_join(taxi_sf, districts_sf)

# Calculate statistics by district
district_stats <- taxi_districts %>%
  st_drop_geometry() %>%
  filter(!is.na(district_name)) %>%
  group_by(district_name) %>%
  summarise(
    avg_speed = mean(Speed, na.rm = TRUE),
    median_speed = median(Speed, na.rm = TRUE),
    count = n(),
    .groups = "drop")

# Create visualization
ggplot(taxi_districts %>% filter(!is.na(district_name)),
       aes(x = reorder(district_name, Speed, FUN = median), y = Speed)) +
  geom_boxplot(aes(fill = reorder(district_name, Speed, FUN = median)),
               outlier.shape = NA) +
  coord_flip() +
  ylim(0, quantile(taxi_districts$Speed, 0.95)) +
  scale_fill_viridis_d(option = "plasma", guide = "none") +
  labs(
    title = "Comparison of Taxi Speeds Across Shenzhen Districts",
    subtitle = "Boxplots show distribution of speeds in each district",
    x = "District",
    y = "Speed (km/h)"
  ) +
  theme_bw() +
  theme(
    plot.title = element_text(face = "bold"),
    axis.text.y = element_text(face = "bold"))
```

```
## Warning: Removed 10720 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```



Shenzhen Districts Speed Analysis (Alternative Method)

```r
# Read taxi data
taxi_data <- read.csv("TaxiData-Sample.csv", header = FALSE)
names(taxi_data) <- c("VehicleNum", "Time", "Lng", "Lat", "OpenStatus", "Speed")
```

```r
# Filter moving vehicles
moving_taxis <- taxi_data %>% filter(Speed > 0)

# Create districts
districts <- data.frame(
  district_name = c("Futian", "Luohu", "Nanshan", "Yantian", "Baoan", "Longgang"),
  lat_min = c(22.5, 22.53, 22.5, 22.55, 22.55, 22.6),
  lat_max = c(22.6, 22.62, 22.6, 22.7, 22.75, 22.8),
  lng_min = c(113.9, 114.05, 113.8, 114.2, 113.75, 114.15),
  lng_max = c(114.05, 114.15, 113.95, 114.3, 113.9, 114.3)
)

# Assign districts
assign_district <- function(point_lng, point_lat, districts) {
  for(i in 1:nrow(districts)) {
    if(point_lng >= districts$lng_min[i] &&
       point_lng <= districts$lng_max[i] &&
       point_lat >= districts$lat_min[i] &&
       point_lat <= districts$lat_max[i]) {
      return(districts$district_name[i])
    }
  }
  return(NA)
}
moving_taxis$district <- sapply(1:nrow(moving_taxis), function(i) {
  assign_district(moving_taxis$Lng[i], moving_taxis$Lat[i], districts)
})

# Plot
p2 <- ggplot(moving_taxis %>% filter(!is.na(district)),
       aes(x = reorder(district, Speed, median), y = Speed)) +
  geom_boxplot(aes(fill = reorder(district, Speed, median)),
               alpha = 0.7, outlier.shape = NA) +
  coord_flip() +
  coord_cartesian(ylim = c(0, quantile(moving_taxis$Speed, 0.95))) +
  scale_fill_viridis_d(option = "plasma", guide = "none") +
  labs(
    x = "District",
    y = "Speed (km/h)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold"),
    axis.text.y = element_text(face = "bold")
  )
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```r
# Save with same size
ggsave("shenzhen_districts_boxplot.png", p2, width = 8, height = 6, dpi = 300)
```

Figure 10(b) provides a similar analysis for Shenzhen districts. The report mentions that this is included to allow for comparison between Shanghai and Shenzhen taxi operations. The analysis again helps identify districts with faster or slower taxi speeds, informing urban planning and transportation management decisions.