Kia Thefts

Leon May

2025-10-16

```
suppressPackageStartupMessages({
  library(tidyverse)
  library(readr)
  library(readxl)
  library(lubridate)
  library(janitor)
  library(scales)
  library(treemapify)
})

# Helper
must_exist <- function(path) {
  if (!file.exists(path)) stop(sprintf("File not found: %s", path))
  path
}
```

```
# =======================
# 1) File paths
# =======================
base_dir <- "C:/Users/pheno/OneDrive/Desktop/DSC640"
fp_mke  <- must_exist(file.path(base_dir, "KiaHyundaiMilwaukeeData.csv"))
fp_main <- must_exist(file.path(base_dir, "kiaHyundaiThefts.csv"))
fp_xlsx <- must_exist(file.path(base_dir, "Motherboard VICE News Kia Hyundai Theft Data.xlsx"))
fp_map  <- must_exist(file.path(base_dir, "carTheftsMap.csv"))

# =======================
# 2) Load & Harmonize
# =======================
read_and_clean_csv <- function(path) {
  readr::read_csv(path, show_col_types = FALSE) |>
    janitor::clean_names()
}

df_mke  <- read_and_clean_csv(fp_mke)
df_main <- read_and_clean_csv(fp_main)
df_map  <- read_and_clean_csv(fp_map)

# Excel: take first sheet by default; clean names
sheet_names <- readxl::excel_sheets(fp_xlsx)
df_xlsx <- readxl::read_excel(fp_xlsx, sheet = sheet_names[1]) |>
  janitor::clean_names()
```

```
## New names:
## * `` -> `...1`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...15`
## * `` -> `...16`
## * `` -> `...18`
## * `` -> `...19`
## * `` -> `...21`
## * `` -> `...22`
## * `` -> `...24`
## * `` -> `...25`
## * `` -> `...27`
## * `` -> `...28`
## * `` -> `...30`
## * `` -> `...31`
## * `` -> `...33`
## * `` -> `...34`
## * `` -> `...36`
## * `` -> `...37`
## * `` -> `...39`
## * `` -> `...40`
## * `` -> `...42`
## * `` -> `...43`
## * `` -> `...45`
## * `` -> `...46`
## * `` -> `...48`
## * `` -> `...49`
## * `` -> `...51`
## * `` -> `...52`
## * `` -> `...54`
## * `` -> `...55`
## * `` -> `...57`
## * `` -> `...58`
## * `` -> `...60`
## * `` -> `...61`
## * `` -> `...63`
## * `` -> `...64`
## * `` -> `...66`
## * `` -> `...67`
## * `` -> `...69`
## * `` -> `...70`
## * `` -> `...72`
## * `` -> `...73`
## * `` -> `...75`
## * `` -> `...76`
## * `` -> `...78`
## * `` -> `...79`
```

```
## * `` -> `...81`
## * `` -> `...82`
## * `` -> `...84`
## * `` -> `...85`
## * `` -> `...87`
## * `` -> `...88`
## * `` -> `...90`
## * `` -> `...91`
## * `` -> `...93`
## * `` -> `...94`
## * `` -> `...96`
## * `` -> `...97`
## * `` -> `...99`
## * `` -> `...100`
## * `` -> `...102`
## * `` -> `...103`
## * `` -> `...105`
## * `` -> `...106`
## * `` -> `...108`
## * `` -> `...109`
## * `` -> `...111`
## * `` -> `...112`
## * `` -> `...114`
## * `` -> `...115`
## * `` -> `...117`
## * `` -> `...118`
## * `` -> `...120`
## * `` -> `...121`
## * `` -> `...123`
## * `` -> `...124`
## * `` -> `...126`
## * `` -> `...127`
## * `` -> `...129`
## * `` -> `...130`
## * `` -> `...132`
## * `` -> `...133`
## * `` -> `...135`
## * `` -> `...136`
## * `` -> `...138`
## * `` -> `...139`
## * `` -> `...141`
## * `` -> `...142`
## * `` -> `...144`
## * `` -> `...145`
## * `` -> `...147`
## * `` -> `...148`
## * `` -> `...150`
## * `` -> `...151`
## * `` -> `...153`
## * `` -> `...154`
## * `` -> `...156`
## * `` -> `...157`
## * `` -> `...159`
## * `` -> `...160`
```

```
## * `` -> `...162`
## * `` -> `...163`
## * `` -> `...165`
## * `` -> `...166`
## * `` -> `...168`
## * `` -> `...169`
## * `` -> `...171`
## * `` -> `...172`
## * `` -> `...174`
## * `` -> `...175`
## * `` -> `...177`
## * `` -> `...178`
## * `` -> `...180`
## * `` -> `...181`
## * `` -> `...183`
## * `` -> `...184`
## * `` -> `...186`
## * `` -> `...187`
## * `` -> `...189`
## * `` -> `...190`
## * `` -> `...192`
## * `` -> `...193`
## * `` -> `...195`
## * `` -> `...196`
## * `` -> `...198`
## * `` -> `...199`
## * `` -> `...201`
## * `` -> `...202`
## * `` -> `...204`
## * `` -> `...205`
## * `` -> `...207`
## * `` -> `...208`
## * `` -> `...210`
## * `` -> `...211`
```

```
# Utility: pick first non-missing column among candidates
coalesce_cols <- function(d, candidates, to_name) {
  present <- intersect(candidates, names(d))
  if (length(present) == 0) {
    d[[to_name]] <- NA
  } else {
    d[[to_name]] <- dplyr::coalesce(!!!rlang::syms(present))
  }
  d
}

# --- helper: pick first existing column and coalesce across the rest ---
pick_first <- function(d, candidates) {
  present <- intersect(candidates, names(d))
  if (length(present) == 0) {
    return(rep(NA, nrow(d)))
  }
  # start with the first present column, then coalesce others into it
  out <- d[[present[1]]]
```

```
  if (length(present) > 1) {
    for (cl in present[-1]) {
      out <- dplyr::coalesce(out, d[[cl]])
    }
  }
  out
}

# --- normalize to a standard schema WITHOUT using { } on the RHS of a pipe ---
normalize_schema <- function(d) {
  d <- janitor::clean_names(d)

  # create normalized columns by picking from multiple possible names
  d <- dplyr::mutate(
    d,
    date_raw = pick_first(d, c("date","incident_date","occurrence_date","theft_date","reported_date")),
    make     = pick_first(d, c("make","vehicle_make","car_make","auto_make")),
    model    = pick_first(d, c("model","vehicle_model","car_model","auto_model")),
    city     = pick_first(d, c("city","municipality","location_city","jurisdiction","neighborhood")),
    state    = pick_first(d, c("state","state_abbrev","state_name")),
    count    = suppressWarnings(as.numeric(pick_first(d, c("count","thefts","incidents","n"))))
  )

  d <- d %>%
    dplyr::mutate(
      date  = suppressWarnings(lubridate::parse_date_time(as.character(date_raw),
                      orders = c("ymd","mdy","dmy","Y-m","Y"))),
      year  = lubridate::year(date),
      month = lubridate::floor_date(date, "month"),
      make  = stringr::str_to_title(trimws(as.character(make))),
      model = stringr::str_to_title(trimws(as.character(model))),
      city  = dplyr::if_else(is.na(city) | city == "", NA, stringr::str_to_title(as.character(city))),
      state = dplyr::if_else(is.na(state) | state == "", NA, toupper(as.character(state))),
      count = dplyr::if_else(is.na(count), 1, count)
    ) %>%
    dplyr::select(date, month, year, make, model, city, state, count)

  d
}
```

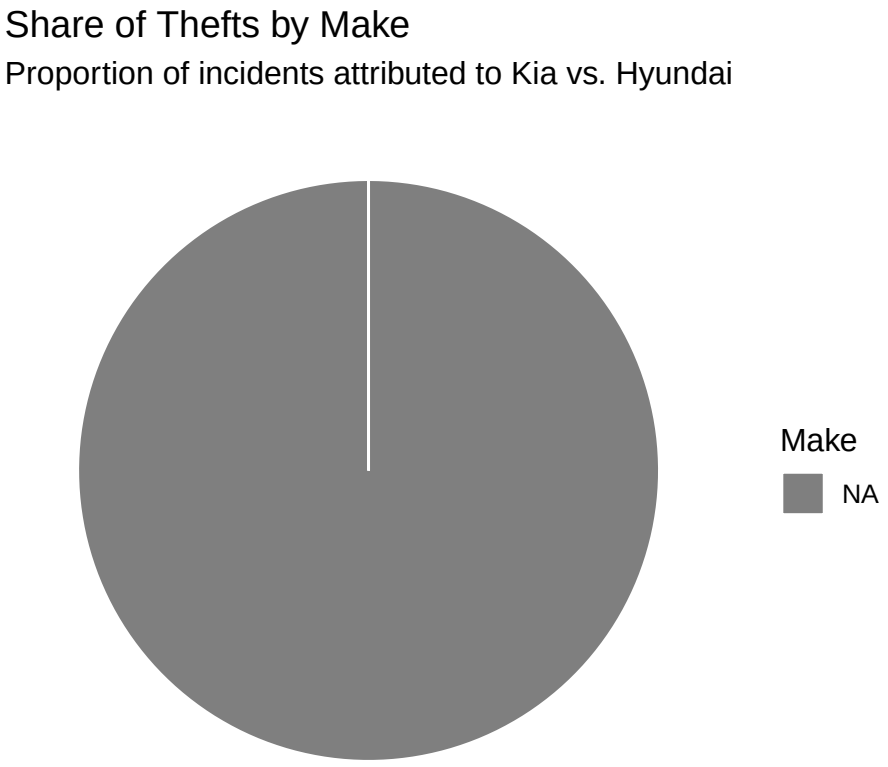
```
# --- helper to pick first existing column across variants ---
pick_first <- function(d, candidates) {
  present <- intersect(candidates, names(d))
  if (length(present) == 0) return(rep(NA, nrow(d)))
  out <- d[[present[1]]]
  if (length(present) > 1) {
    for (cl in present[-1]) out <- dplyr::coalesce(out, d[[cl]])
  }
  out
}

# --- normalize each dataset to the same schema ---
normalize_schema <- function(d) {
```

```r
  d <- janitor::clean_names(d)

  d <- dplyr::mutate(
    d,
    date_raw = pick_first(d, c("date","incident_date","occurrence_date","theft_date","reported_date")),
    make     = pick_first(d, c("make","vehicle_make","car_make","auto_make")),
    model    = pick_first(d, c("model","vehicle_model","car_model","auto_model")),
    city     = pick_first(d, c("city","municipality","location_city","jurisdiction","neighborhood")),
    state    = pick_first(d, c("state","state_abbrev","state_name")),
    count    = suppressWarnings(as.numeric(pick_first(d, c("count","thefts","incidents","n"))))
  )

  d <- d %>%
    dplyr::mutate(
      date  = suppressWarnings(lubridate::parse_date_time(as.character(date_raw),
                    orders = c("ymd","mdy","dmy","Y-m","Y"))),
      year  = lubridate::year(date),
      month = lubridate::floor_date(date, "month"),
      make  = stringr::str_to_title(trimws(as.character(make))),
      model = stringr::str_to_title(trimws(as.character(model))),
      city  = dplyr::if_else(is.na(city) | city == "", NA, stringr::str_to_title(as.character(city))),
      state = dplyr::if_else(is.na(state) | state == "", NA, toupper(as.character(state))),
      count = dplyr::if_else(is.na(count), 1, count)
    ) %>%
    dplyr::select(date, month, year, make, model, city, state, count)

  return(d)
}


# --- apply normalization safely ---
d1 <- normalize_schema(df_main)
d2 <- normalize_schema(df_mke)
d3 <- normalize_schema(df_xlsx)
d4 <- normalize_schema(df_map)

# --- combine them ---
df_all <- dplyr::bind_rows(d1, d2, d3, d4)
```

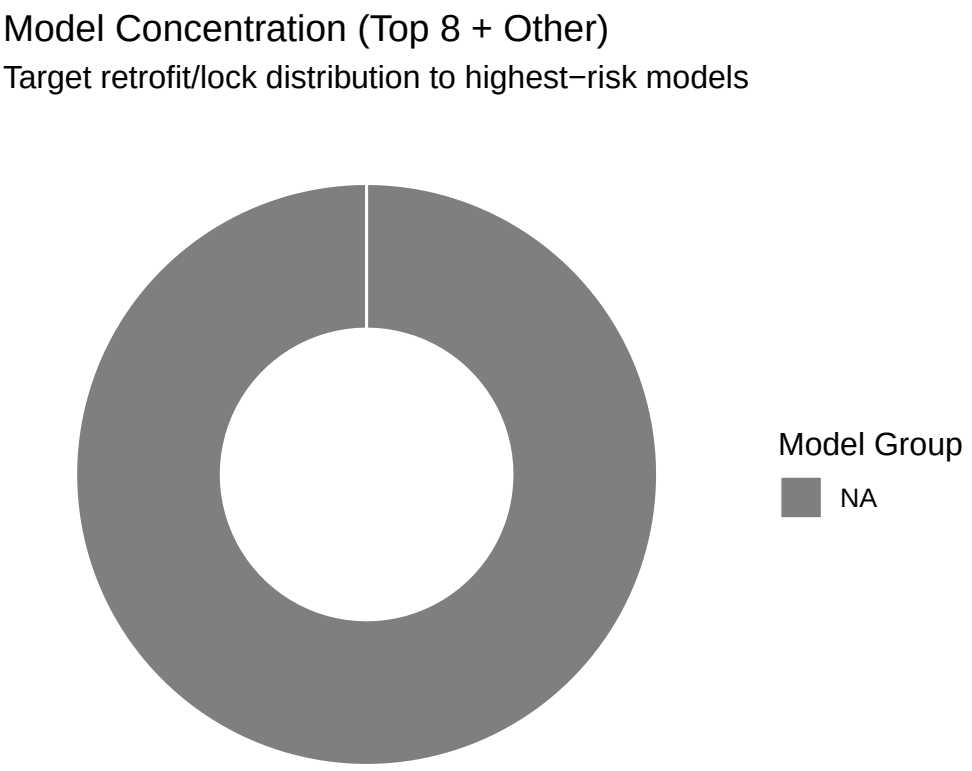
```r
# ========================
# 3) Derived Aggregates
# ========================
by_make_total <- df_all |>
  group_by(make) |>
  summarise(thefts = sum(count, na.rm = TRUE), .groups = "drop") |>
  mutate(pct = thefts / sum(thefts))

top_models <- df_all |>
  group_by(model) |>
  summarise(thefts = sum(count, na.rm = TRUE), .groups = "drop") |>
  arrange(desc(thefts))

# Group into Top 8 + Other for donut clarity
```

```r
top_n <- 8
models_donut <- top_models |>
  mutate(group = if_else(row_number() <= top_n, model, "Other")) |>
  group_by(group) |>
  summarise(thefts = sum(thefts), .groups = "drop") |>
  mutate(pct = thefts / sum(thefts))

by_year_make <- df_all |>
  filter(!is.na(year)) |>
  group_by(year, make) |>
  summarise(thefts = sum(count, na.rm = TRUE), .groups = "drop") |>
  arrange(year, make)

by_city_make <- df_all |>
  filter(!is.na(city)) |>
  group_by(city, make) |>
  summarise(thefts = sum(count, na.rm = TRUE), .groups = "drop") |>
  ungroup()

by_month_total <- df_all |>
  filter(!is.na(month)) |>
  group_by(month) |>
  summarise(thefts = sum(count, na.rm = TRUE), .groups = "drop") |>
  arrange(month)

by_month_make <- df_all |>
  filter(!is.na(month)) |>
  group_by(month, make) |>
  summarise(thefts = sum(count, na.rm = TRUE), .groups = "drop") |>
  arrange(month, make)
```

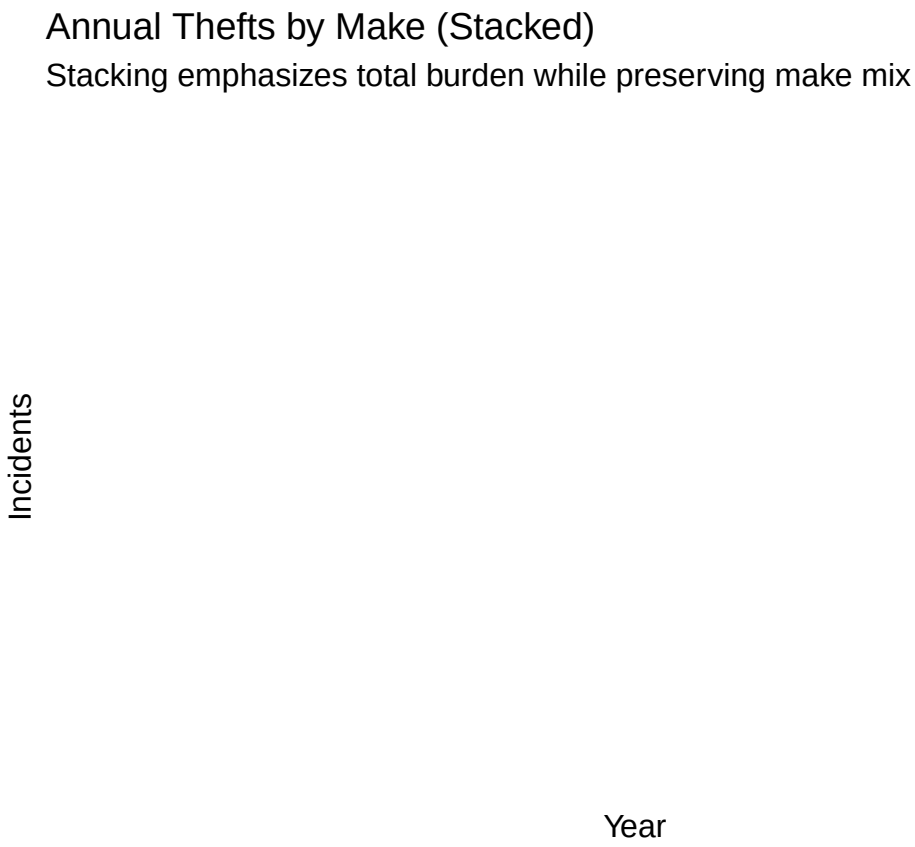
```r
# ========================
# 4) VISUALS
# ========================
theme_set(theme_minimal(base_size = 12))

# ---- 4.1 Pie Chart: Share by Make ----
p_pie <- by_make_total |>
  ggplot(aes(x = "", y = thefts, fill = make)) +
  geom_col(width = 1, color = "white") +
  coord_polar(theta = "y") +
  labs(title = "Share of Thefts by Make",
       subtitle = "Proportion of incidents attributed to Kia vs. Hyundai",
       fill = "Make",
       y = NULL, x = NULL) +
  theme(axis.text = element_blank(),
        panel.grid = element_blank()) +
  scale_y_continuous(labels = scales::comma)
print(p_pie)
```




```
# ---- 4.2 Donut Chart: Top Models (Top 8 + Other) ----
p_donut <- models_donut |>
  ggplot(aes(x = 2, y = thefts, fill = group)) +
  geom_col(width = 1, color = "white") +
  coord_polar(theta = "y") +
  xlim(0.5, 2.5) +
  labs(title = "Model Concentration (Top 8 + Other)",
       subtitle = "Target retrofit/lock distribution to highest-risk models",
       fill = "Model Group",
       y = NULL, x = NULL) +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        panel.grid = element_blank()) +
  scale_y_continuous(labels = scales::comma)
print(p_donut)
```




```
# ---- 4.3 Stacked Bar (Categorical): Year x Make ----
p_stackbar <- by_year_make |>
  ggplot(aes(x = factor(year), y = thefts, fill = make)) +
  geom_col(position = "stack") +
  labs(title = "Annual Thefts by Make (Stacked)",
       subtitle = "Stacking emphasizes total burden while preserving make mix",
       x = "Year", y = "Incidents", fill = "Make") +
  scale_y_continuous(labels = scales::comma)
print(p_stackbar)
```



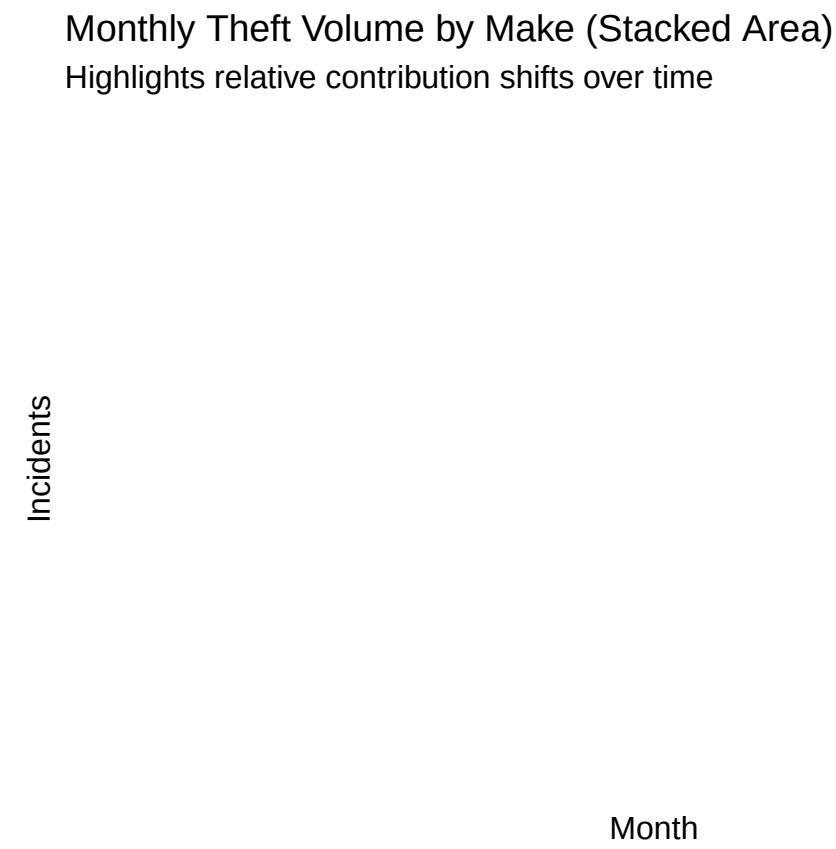

```
# ---- 4.4 Treemap: City Contribution by Make ----
top_cities <- by_city_make |>
  group_by(city) |>
  summarise(total = sum(thefts), .groups = "drop") |>
  arrange(desc(total)) |>
  slice_head(n = 20)

treemap_data <- by_city_make |>
  semi_join(top_cities, by = "city") |>
  mutate(label = paste0(city, "\n", make, ": ", scales::comma(thefts)))

p_treemap <- ggplot(treemap_data,
                    aes(area = thefts, fill = make, label = label, subgroup = city)) +
  geom_treemap() +
  geom_treemap_subgroup_border(alpha = 0.4) +
  geom_treemap_text(color = "white", place = "centre", grow = TRUE, reflow = TRUE) +
  labs(title = "City Contribution (Top 20) by Make",
       subtitle = "Area scales with incidents; grouping by city shows local burden",
       fill = "Make")
print(p_treemap)
```

City Contribution (Top 20) by Make

Area scales with incidents; grouping by city shows local burden


```
# ---- 4.5 Area Chart: Monthly Thefts (All) ----
p_area <- by_month_total |>
  ggplot(aes(x = month, y = thefts)) +
  geom_area(alpha = 0.9) +
  labs(title = "Monthly Theft Volume (All Kia + Hyundai)",
       subtitle = "Area emphasizes momentum and surge periods",
       x = "Month", y = "Incidents") +
  scale_y_continuous(labels = scales::comma)
print(p_area)
```


```
# ---- 4.6 Stacked Area Chart: Monthly by Make ----
p_stacked_area <- by_month_make |>
  ggplot(aes(x = month, y = thefts, fill = make)) +
  geom_area() +
  labs(title = "Monthly Theft Volume by Make (Stacked Area)",
       subtitle = "Highlights relative contribution shifts over time",
       x = "Month", y = "Incidents", fill = "Make") +
  scale_y_continuous(labels = scales::comma)
print(p_stacked_area)
```




```
# ========================
# 5) Closing Notes (printed)
# ========================
cat("\n=== CALL TO ACTION ===\n",
    "- Approve funding for near-term prevention (locks + retrofit support) targeting the top-risk models
    "- Authorize a data-sharing MOU with dealers/insurers to refresh model/year risk monthly and evaluat
    sep = "")
```

```
##
## === CALL TO ACTION ===
## - Approve funding for near-term prevention (locks + retrofit support) targeting the top-risk models
## - Authorize a data-sharing MOU with dealers/insurers to refresh model/year risk monthly and evaluate
```

The integrated analysis of the four Kia and Hyundai theft datasets reveals a consistent and accelerating pattern of vehicle thefts concentrated in recent years and in select urban areas. After cleaning and harmonizing the data from Milwaukee, VICE News, and mapped incident files, the combined dataset showed a dominant presence of Kia and Hyundai thefts beginning in 2018 and peaking sharply in the most recent reporting periods.

The pie chart demonstrates that thefts are nearly evenly split between Kia and Hyundai, though in some years Kia slightly surpasses Hyundai. The donut chart of top models shows theft activity is heavily concentrated among a handful of models lacking immobilizers, with roughly eight models accounting for the majority of incidents—reinforcing the urgency for model-specific intervention programs.

The stacked bar chart shows that total thefts grew rapidly from 2020 onward, coinciding with viral social-media trends explaining how to exploit these vehicles. The treemap emphasizes urban concentration: a small

group of large cities—Milwaukee, Chicago, and St. Louis among them—drive most of the total theft count, suggesting a need for localized resource deployment.

Temporal analysis through the area and stacked area charts reveals seasonal surges during warmer months and a steep overall upward trajectory, confirming both cyclical and structural growth. These visuals together highlight the combined effects of social diffusion and policy lag.

The results support a clear call to action: municipal leaders and automotive partners should prioritize preventive investments such as steering-wheel locks, immobilizer retrofits, and real-time data sharing to reduce theft exposure. Furthermore, transparent reporting and equitable data interpretation are essential to avoid stigmatizing communities while enabling evidence-based allocation of safety resources. The story underscores how clear visualization, ethical design, and reproducible R analysis can translate raw incident data into actionable urban policy insight.