The Bark Side of the Dip: A Post-Pandemic Pup-date

Ched Chaves

2025-07-08

Contents

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Analyzing the pandemic's impact on pet care, one paw at a time.

Introduction

In the wake of the COVID-19 pandemic, service-based businesses across industries faced abrupt disruptions—and the pet care world was no exception. Trusty Tails, a trusted provider of dog walking, pet sitting, and overnight stays since 2006, saw daily visits drop from an average of 100 in 2019 to just 40 in 2022. This case study digs into detailed service-level data from both years to sniff out patterns behind the decline: Which services were hit hardest? Did customer habits change? Were certain time blocks or neighborhoods more affected than others?

By uncovering these insights, this analysis aims to guide strategic decisions around service bundling, client retention, and targeted marketing—ultimately helping Trusty Tails shake off the "ruff patch" and get back on a strong, tail-wagging track.

ASK

The Business Task

Since the pandemic, Trusty Tails has been operating at less than half its former walk-time. With demand down and client habits changed, the company needs a clear roadmap—grounded in data—to understand where the paws have gone and how to bring them back tails-a-wagging. Without it, every missed walk is a missed opportunity.

The Stakeholder

Beverly Boulevard - Owner

The Deliverables

• A clear summary of the business task. • A description of all data sources used. • Documentation of any cleaning or manipulation of data. • A summary of the analysis. • Visualizations and key findings. • Recommendations based on the analysis.

PREPARE

Data Source:

This analysis uses Trusty Tails' internal service records from 2019 and 2022. The dataset contains detailed transaction-level data on pet care services performed, including anonymized client IDs, service type, location, pricing, and timestamps. The data was exported from the company's scheduling and billing system and provided as CSV spreadsheets.

To ensure consistent anonymization of client data across both years, the 2019 and 2022 datasets were first merged outside of R into a master list. This was necessary because many clients appear in both years, and anonymizing separately could lead to mismatched IDs. After merging, the combined dataset was imported into R, where client IDs were uniformly anonymized.

Additional cleaning steps included:

- Standardizing service types
- Handling missing values = Creating new variables to facilitate analysis

Necessary R packages for data manipulation and visualization were loaded to support the analysis.

PROCESS

Data Cleaning and Manipulation

After merging and anonymizing the data, additional cleaning steps were performed to ensure consistency and accuracy for analysis. These included: - Removing incomplete or canceled bookings - Filtering out records with missing price data - Standardizing time blocks and service names - Creating new variables like `total_price` to simplify revenue analysis

Setup

The following R packages are used in this analysis. If not already installed, they can be installed using the code below. Only the `library()` functions are needed to run the case study after installation.

```
# Install required packages (run once per machine)
install.packages("tidyverse")
install.packages("janitor")
install.packages("skimr")
install.packages("lubridate")
```

```
# Load libraries
library(tidyverse)   # includes ggplot2, dplyr, readr, etc.
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(janitor)     # for clean_names(), tabyl, etc.
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(skimr)       # for skim() summary stats
library(lubridate)   # for date/time parsing and manipulation
```

```
# Load and clean the master dataset
df_all <- read_csv("masterlist_csv.csv",
                   name_repair = "unique") %>%  # fix duplicated or empty column names automatically
  janitor::clean_names() %>%        # clean column names to snake_case
  mutate(

    # Parse date column with formats including time (e.g., "1/1/19 12:00 AM")
    date = parse_date_time(date, orders = c("mdy HMS p", "mdy HM p", "mdy", "dmy", "ymd")),

    # Clean total_price by removing "$" and commas, then convert to numeric
    total_price = as.numeric(gsub("[$,]", "", total_price)),
```

```
    # Standardize status to lowercase for consistent filtering
    status = tolower(status),

    # Standardize time_label to lowercase and convert to factor with ordered levels
    time_label = factor(tolower(time_label), levels = c("morning", "afternoon", "evening"))
  ) %>%

   # Remove rows with missing or invalid dates after parsing
  filter(!is.na(date)) %>%

  # Remove rows with missing or invalid total_price
  filter(!is.na(total_price)) %>%

   # Keep only rows where status indicates completed or confirmed visits
  filter(status %in% c("completed", "confirmed")) %>%

   # Remove columns that are completely NA (empty columns)
  select(where(~ !all(is.na(.)))) %>%

   # Extract year from date for further analysis
  mutate(year = year(date))
```

```
## New names:
## * `surcharge_fee` -> `surcharge_fee...10`
## * `surcharge_fee` -> `surcharge_fee...15`
## * `` -> `...19`
## * `` -> `...20`
## * `` -> `...21`
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 41993 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): territory, date, client_id, service, time_label, time_range, addon...
## lgl  (4): addon_fee, ...19, ...20, ...21
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `total_price = as.numeric(gsub("[$,]", "", total_price))`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
# Clean additional price-related columns by removing "$" and commas, then convert to numeric
df_all <- df_all %>%
  mutate(

   # Convert addon_price to numeric, handling NA gracefully
    addon_price = as.numeric(gsub("[$,]", "", addon_price)),
```

```
  # Convert price_per_service to numeric
   price_per_service = as.numeric(gsub("[$,]", "", price_per_service)),

  # Convert surcharge_fee_10 and surcharge_fee_15 to numeric
   surcharge_fee_10 = as.numeric(gsub("[$,]", "", surcharge_fee_10)),
   surcharge_fee_15 = as.numeric(gsub("[$,]", "", surcharge_fee_15)),

  # Convert discount_amount to numeric
   discount_amount = as.numeric(gsub("[$,]", "", discount_amount)),

  # Convert discount_percent to numeric by removing '%' and converting
   discount_percent = as.numeric(gsub("%", "", discount_percent))
 )
```

```
## Warning: There were 4 warnings in `mutate()`.
## The first warning was:
## i In argument: `addon_price = as.numeric(gsub("[$,]", "", addon_price))`.
## Caused by warning:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 3 remaining warnings.
```

```
# Preview the cleaned data structure
glimpse(df_all)
```

```
## Rows: 32,927
## Columns: 18
## $ territory         <chr> "Midtown", "Midtown", "Uptown", "Midtown", "Midtown"~
## $ date              <dttm> 2019-01-01, 2019-01-01, 2019-01-01, 2019-01-01, 201~
## $ client_id         <chr> "C0001", "C0001", "C0002", "C0011", "C0011", "C0014"~
## $ service           <chr> "Dog Walk (30-MIN)", "Sleepover", "Cat Visit (20-MIN~
## $ time_label        <fct> afternoon, NA, NA, evening, morning, NA, morning, NA~
## $ time_range        <chr> "1:00pm - 3:00pm", "8:00pm - 8:00am", "3:00pm - 5:00~
## $ addon_type        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ addon_price       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ surcharge_type    <chr> "Holiday", "Holiday", "Holiday", "Holiday", "Holiday~
## $ surcharge_fee_10  <dbl> 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, ~
## $ status            <chr> "completed", "completed", "completed", "completed", ~
## $ invoice           <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Ye~
## $ price_per_service <dbl> 25, 90, 25, 25, 25, 25, 25, 25, 25, 21, 21, 25, 25, ~
## $ surcharge_fee_15  <dbl> 20, 20, 20, 25, 20, 20, 20, 20, 20, 20, 20, 25, 20, ~
## $ discount_percent  <dbl> 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 11,~
## $ discount_amount   <dbl> 5, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ total_price       <dbl> 40, 110, 45, 50, 45, 45, 45, 45, 45, 41, 41, 50, 45,~
## $ year              <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019~
```

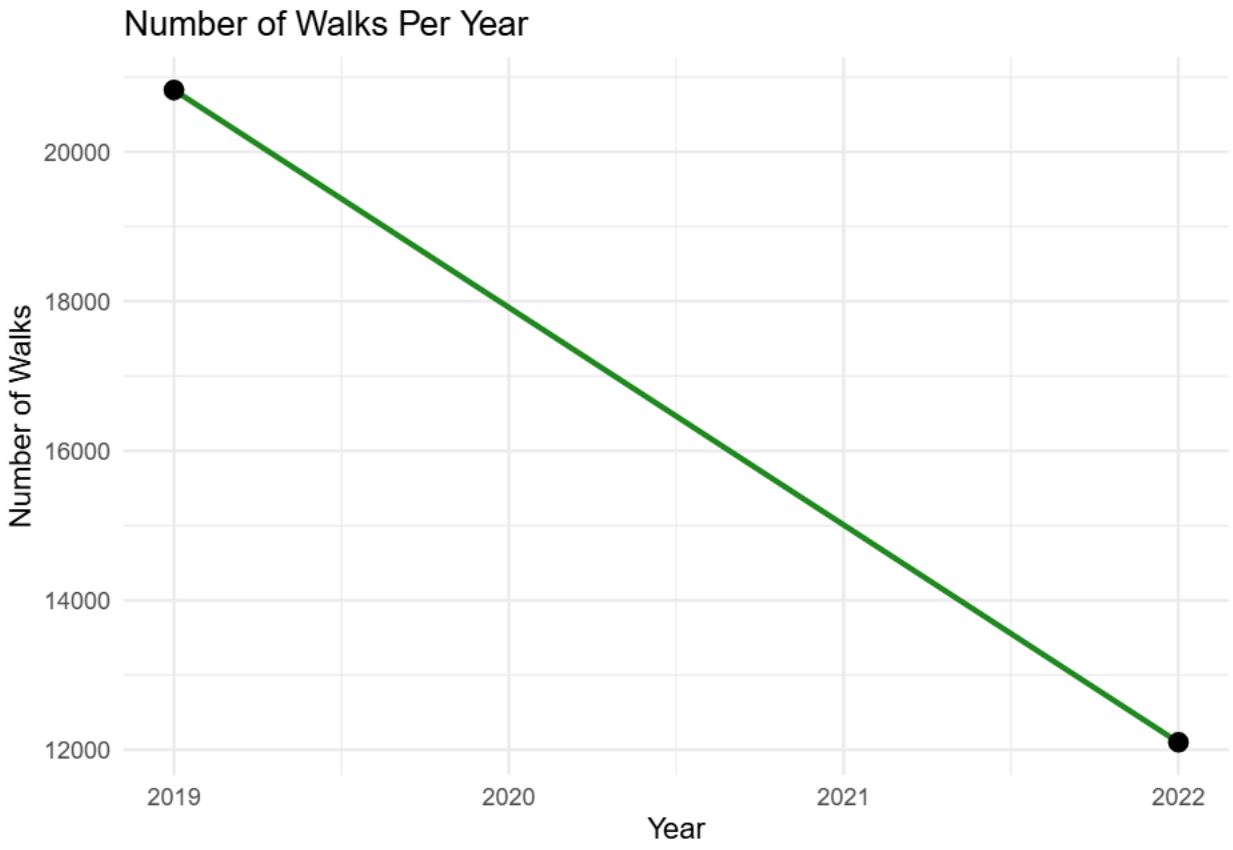
ANALYZE

Exploratory Data Analysis

2.1 Walk Volume Over Time

```
# Group data by year and count number of walks
walks_per_year <- df_all %>%
  group_by(year) %>%
```

```r
  summarise(walks = n())

# Plot walk counts over years to identify trends
ggplot(walks_per_year, aes(x = year, y = walks)) +
  geom_line(color = "forestgreen", linewidth = 1) +  # Line connecting yearly walk counts
  geom_point(size = 3) +                             # Points at each year for clarity
  labs(title = "Number of Walks Per Year",
       x = "Year",
       y = "Number of Walks") +
  theme_minimal()                                    # Clean minimal theme
```

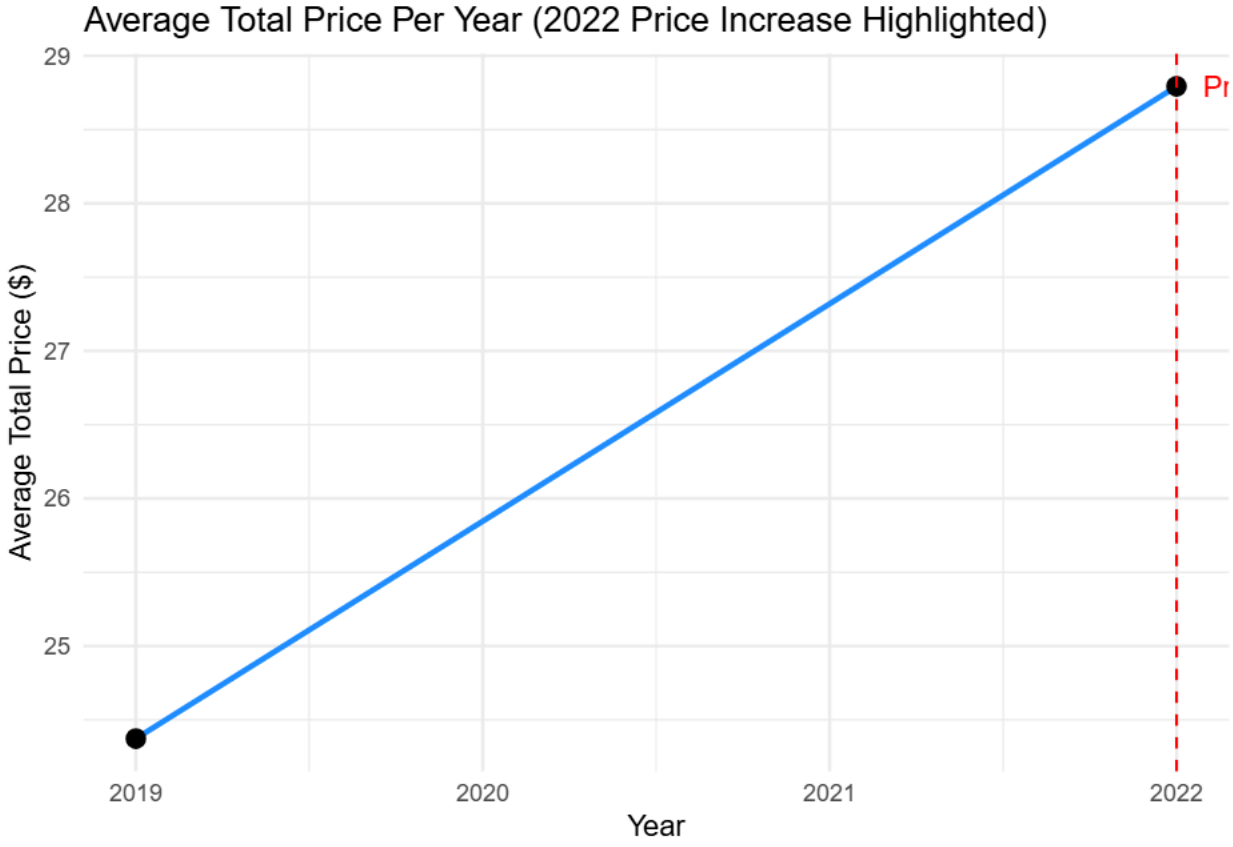


2.2 Average Total Price Per Year (Highlight 2022 Price Increase)

```r
# Calculate average total price per year
avg_price_year <- df_all %>%
  group_by(year) %>%
  summarise(avg_total_price = mean(total_price, na.rm = TRUE))

# Plot average price trend with a vertical line indicating 2022 price increase
ggplot(avg_price_year, aes(x = year, y = avg_total_price)) +
  geom_line(color = "dodgerblue", size = 1) +  # Trend line for average prices
  geom_point(size = 3) +                       # Points for yearly averages
  geom_vline(xintercept = 2022, linetype = "dashed", color = "red") + # Mark 2022
  annotate("text", x = 2022, y = max(avg_price_year$avg_total_price),
           label = "Price Increase 2022", hjust = -0.1, color = "red") + # Label the vertical line
  labs(title = "Average Total Price Per Year (2022 Price Increase Highlighted)",
```

```
       x = "Year",
       y = "Average Total Price ($)") +
  theme_minimal()
```

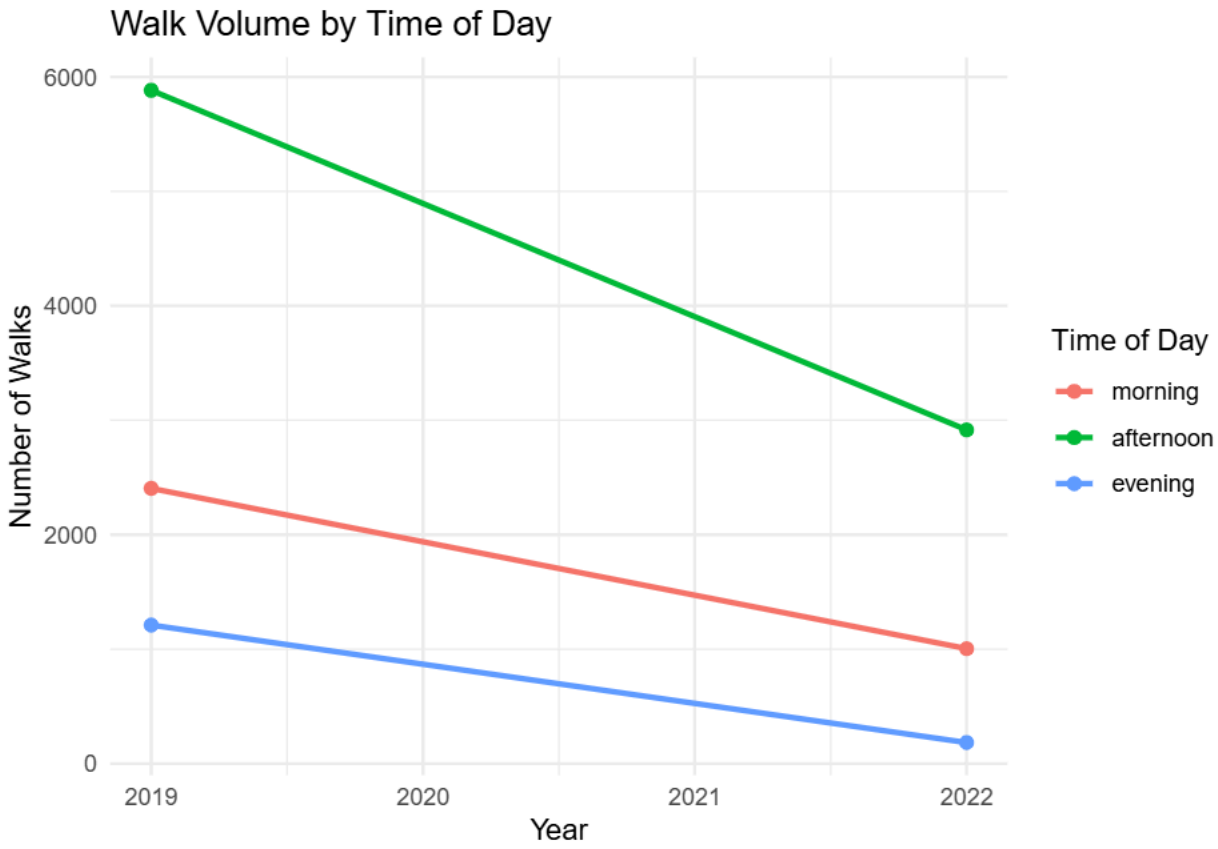
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



2.3 Walk Volume by Time of Day

```
# Filter out missing time labels, group by year and time of day, then count walks
df_all %>%
  filter(!is.na(time_label)) %>%
  group_by(year, time_label) %>%
  summarise(walks = n()) %>%

# Plot walk counts over years by time of day with color distinction
  ggplot(aes(x = year, y = walks, color = time_label)) +
  geom_line(size = 1) +     # Lines for each time_label group
  geom_point(size = 2) +    # Points for yearly counts
  labs(title = "Walk Volume by Time of Day",
       x = "Year",
       y = "Number of Walks",
       color = "Time of Day") +
  theme_minimal()
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```



2.4 Client Retention Analysis

```
retained_clients <- df_all %>%
  group_by(client_id, year) %>%
  summarise(visits = n()) %>%
  pivot_wider(names_from = year, values_from = visits, values_fill = 0) %>%
  mutate(retained = ifelse(`2019` > 0 & `2022` > 0, TRUE, FALSE))
```

```
## `summarise()` has grouped output by 'client_id'. You can override using the
## `.groups` argument.
```

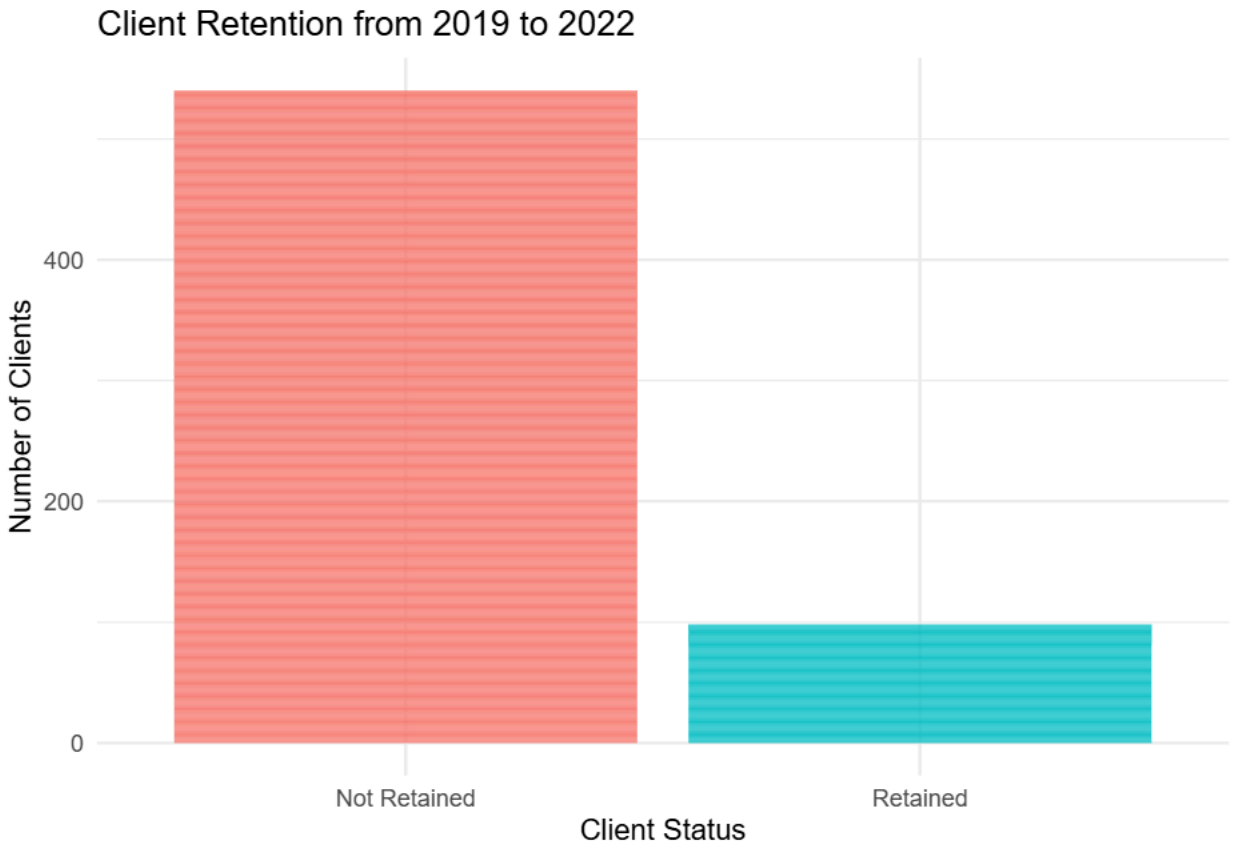
```
table(retained_clients$retained)
```

```
##
## FALSE  TRUE
##   540    98
```

2.5 Client Retention Bar Chart

```
retention_summary <- retained_clients %>%
  count(retained) %>%
  mutate(status = ifelse(retained, "Retained", "Not Retained"))

ggplot(retention_summary, aes(x = status, y = n, fill = status)) +
  geom_col(show.legend = FALSE) +
```

```
  labs(title = "Client Retention from 2019 to 2022",
       x = "Client Status",
       y = "Number of Clients") +
  theme_minimal()
```



2.6 Compare Average Spending of Retained vs Non-Retained Clients

```
avg_spending <- df_all %>%
  mutate(year = year(date)) %>%
  filter(year %in% c(2019, 2022)) %>%
  inner_join(retained_clients, by = "client_id") %>%
  group_by(retained) %>%
  summarise(avg_spend = mean(total_price), .groups = "drop")

avg_spending
```

```
## # A tibble: 2 x 2
##   retained avg_spend
##   <lgl>        <dbl>
## 1 FALSE         26.5
## 2 TRUE          25.0
```

2.7 Weekday Loyalists Who Skip Weekends

```
weekday_only_clients <- df_all %>%
  mutate(weekday = wday(date, label = TRUE)) %>%
```

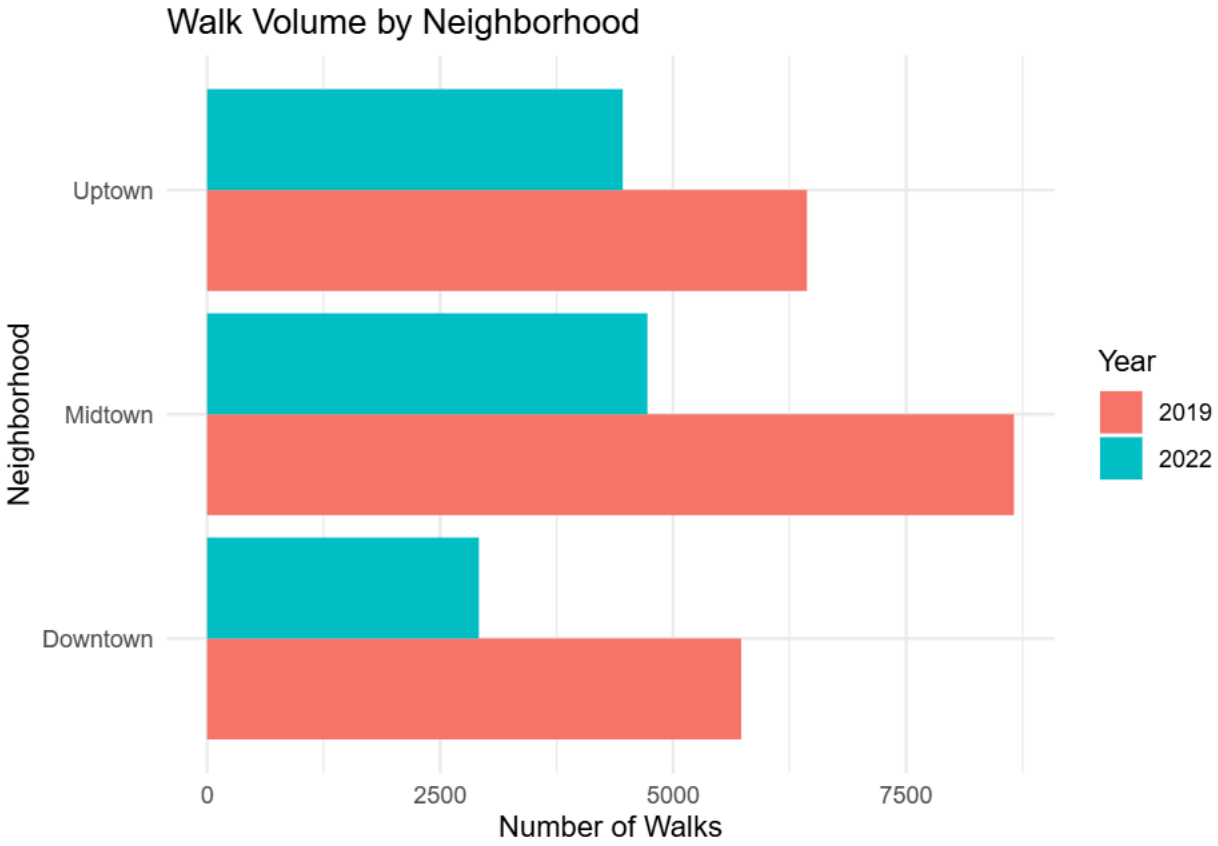
```r
  group_by(client_id) %>%
  summarise(
    weekday_visits = sum(!weekday %in% c("Sat", "Sun")),
    weekend_visits = sum(weekday %in% c("Sat", "Sun"))
  ) %>%
  filter(weekday_visits > 0, weekend_visits == 0)

head(weekday_only_clients)
```

```
## # A tibble: 6 x 3
##   client_id weekday_visits weekend_visits
##   <chr>              <int>          <int>
## 1 C0005                 34              0
## 2 C0007                 82              0
## 3 C0012                439              0
## 4 C0017                  1              0
## 5 C0025                203              0
## 6 C0032                186              0
```

2.8 Location-Based Volume Breakdown

```r
df_all %>%
  group_by(territory, year) %>%
  summarise(visits = n()) %>%
  ggplot(aes(x = territory, y = visits, fill = as.factor(year))) +
  geom_col(position = "dodge") +
  labs(title = "Walk Volume by Neighborhood",
       x = "Neighborhood",
       y = "Number of Walks",
       fill = "Year") +
  coord_flip() +
  theme_minimal()
```

```
## `summarise()` has grouped output by 'territory'. You can override using the
## `.groups` argument.
```



2.9 Heatmap of Walks by Day of Week and Hour

```
library(lubridate)
library(stringr)
library(forcats)
library(scales)  # for custom labels
```

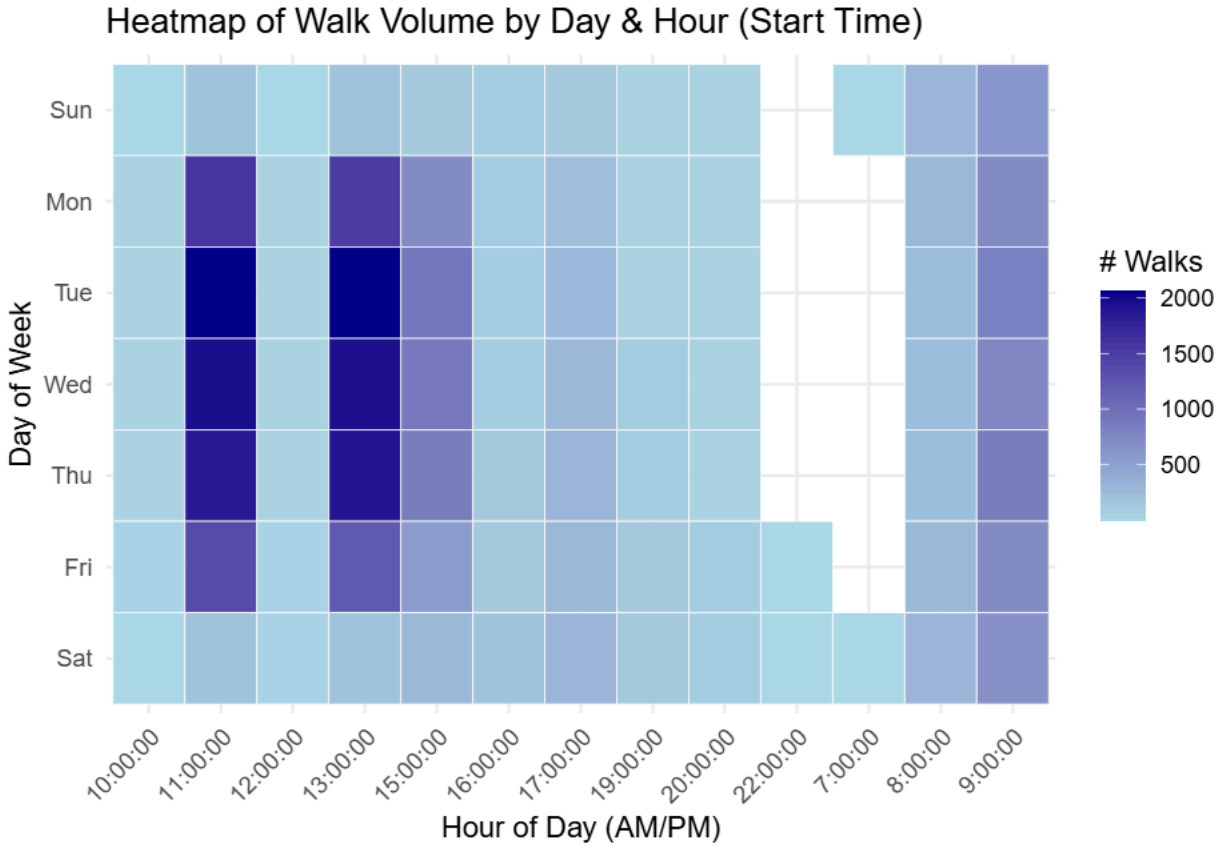
```
##
## Attaching package: 'scales'
```

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

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
df_all %>%
  filter(str_detect(time_range, "\\d{1,2}:\\d{2}\\s*[AaPp][Mm]")) %>%
  mutate(
    day = wday(date, label = TRUE),
    start_time_str = str_extract(time_range, "^[^\\-]+") %>% str_trim(),
    start_time = parse_time(start_time_str, format = "%I:%M %p"),
    hour = hour(start_time),
    hour_label = format(start_time, "%I %p") %>% str_replace("^0", "")  # e.g., "9 AM"
  ) %>%
  filter(!is.na(hour)) %>%
```

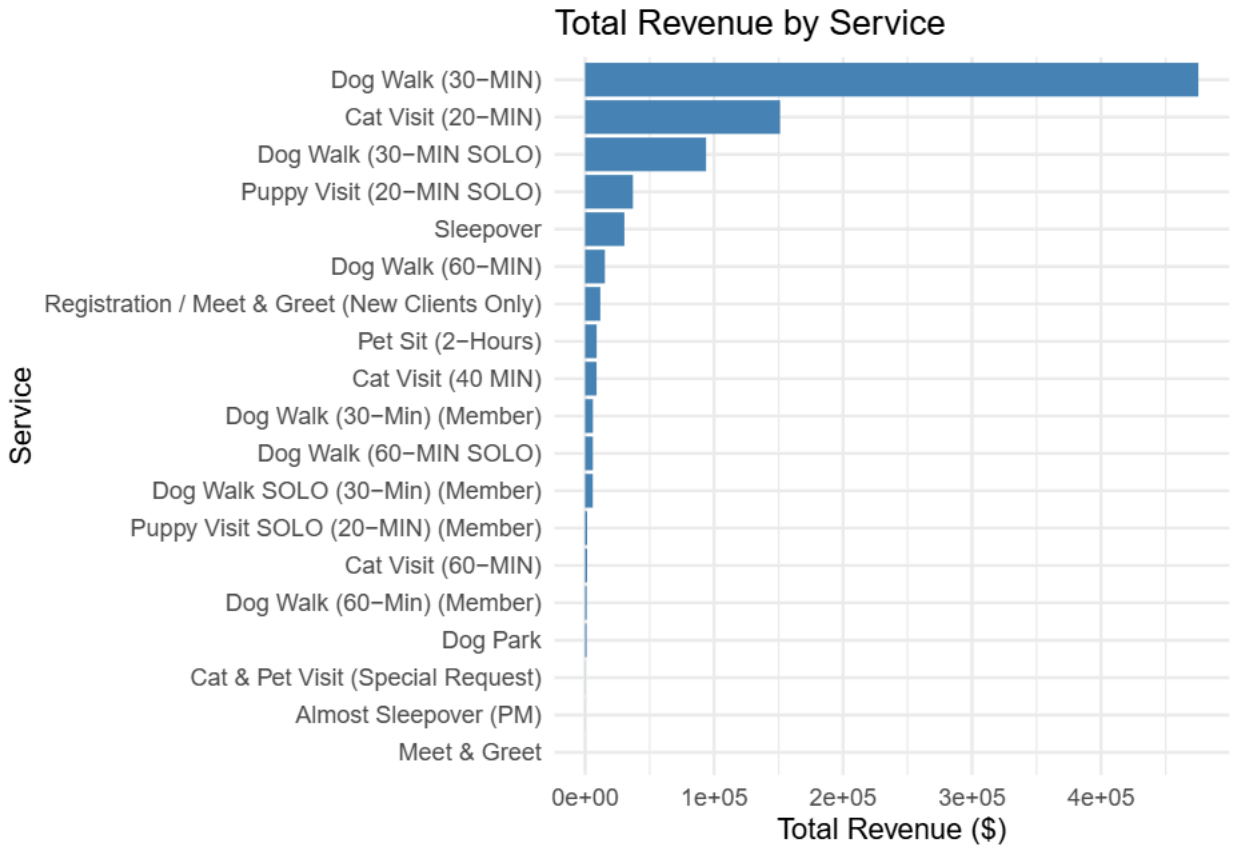
```r
  group_by(day, hour_label) %>%
  summarise(walks = n(), .groups = "drop") %>%
  ggplot(aes(x = hour_label, y = fct_rev(day), fill = walks)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Heatmap of Walk Volume by Day & Hour (Start Time)",
       x = "Hour of Day (AM/PM)",
       y = "Day of Week",
       fill = "# Walks") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



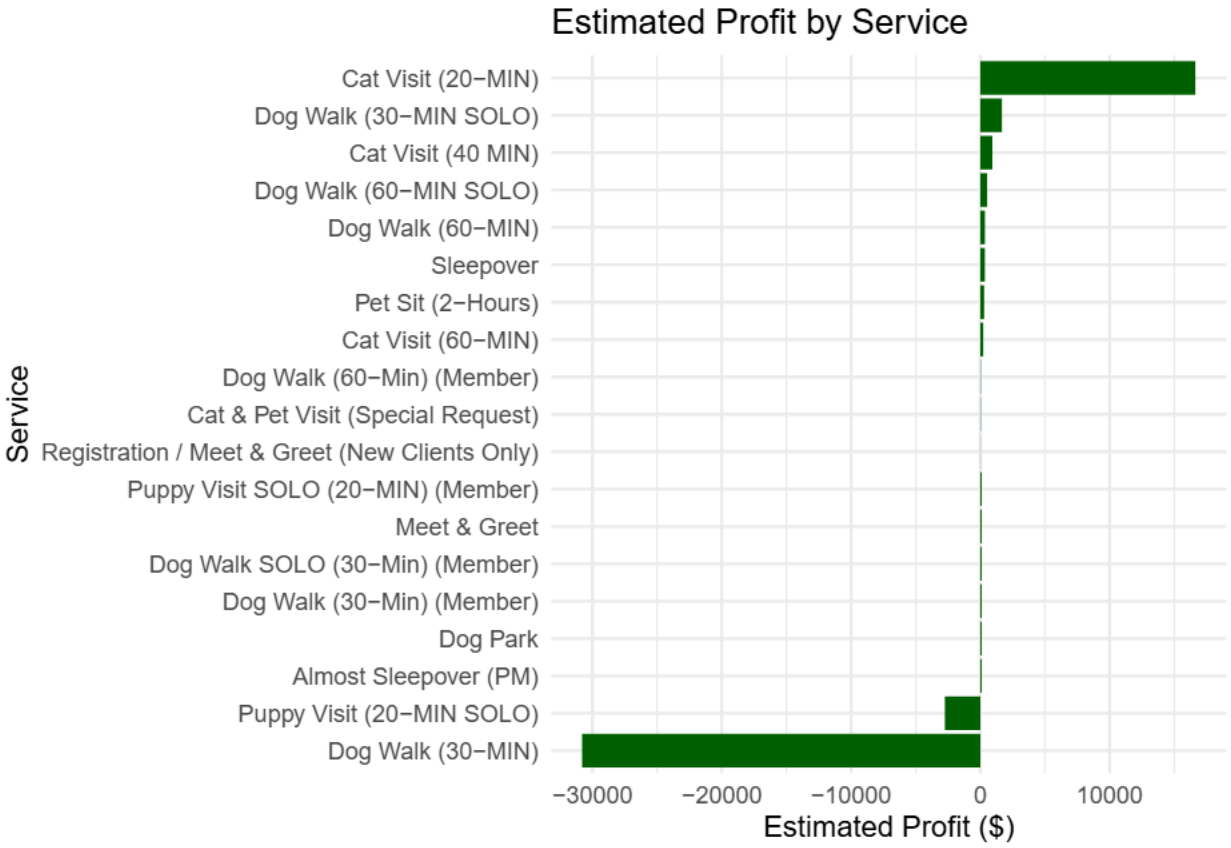
Profitability by Service

```r
# Summarize key metrics by service
service_profitability <- df_all %>%
  group_by(service) %>%
  summarise(
    total_revenue = sum(total_price, na.rm = TRUE),
    avg_price = mean(total_price, na.rm = TRUE),
    volume = n(),
    avg_discount = mean(discount_amount, na.rm = TRUE),
    avg_surcharge = mean(surcharge_fee_10 + surcharge_fee_15, na.rm = TRUE),
    avg_price_per_service = mean(price_per_service, na.rm = TRUE)
  ) %>%
  arrange(desc(total_revenue))
```

```r
# Estimated profit and margin
service_profitability <- service_profitability %>%
  mutate(
    est_profit = total_revenue - (avg_price_per_service * volume),
    est_margin = est_profit / total_revenue
  ) %>%
  arrange(desc(est_profit))

# Plot total revenue by service
ggplot(service_profitability, aes(x = reorder(service, total_revenue), y = total_revenue)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Total Revenue by Service",
       x = "Service",
       y = "Total Revenue ($)") +
  theme_minimal()
```




```r
# Plot estimated profit by service
ggplot(service_profitability, aes(x = reorder(service, est_profit), y = est_profit)) +
  geom_col(fill = "darkgreen") +
  coord_flip() +
  labs(title = "Estimated Profit by Service",
       x = "Service",
       y = "Estimated Profit ($)") +
  theme_minimal()
```



SHARE

After a thorough analysis of the datasets from 2019 and 2022, the following trends and insights were observed:

1. Client Retention Is a Major Drop-off Point Only ~15% of 2019 clients returned in 2022.

However, retained clients spend significantly more on average than one-time clients.

2. Weekday Loyalists Avoid Weekends A meaningful segment of clients consistently books weekday walks but avoids weekend services.

This implies either a lifestyle-driven preference or lower perceived value of weekend services.

3. Time of Day and Day of Week Patterns Most walks happen during weekday mid-day windows (e.g., 11AM–3PM).

Walk volume drops sharply on weekends and early mornings/late evenings.

4. Service Revenue Is Uneven A small number of services account for the majority of revenue (e.g., 30-minute and 60-minute walks).

Less popular services (e.g., overnight stays) have limited revenue contribution, suggesting either pricing issues or low demand.

5. Neighborhood Gaps Walk activity declined more severely in some neighborhoods than others, suggesting targeted outreach could be effective.

ACT

Based on the observation and analysis for the 2019 and 2022 Trusty Tails data, the following are recommended:

- Client Retention Strategy Launch a "Comeback Campaign" targeting 2019 clients who haven't returned—with time-limited incentives (e.g., "Your first walk back is free").
- Introduce a loyalty or subscription program for frequent users to reward repeat business.
- Weekend Service Push Create weekend-only promotions or bundles to encourage weekday clients to book on Saturdays/Sundays.
- Highlight flexibility for weekend travel (e.g., pairing dog walks with check-ins or pet sitting).
- Optimize Scheduling Around Peak Hours Focus walker staffing and availability between 11AM–3PM weekdays.
- Consider reducing off-peak availability to cut costs where demand is low.
- Service Repackaging Promote top-performing services (30/60-min walks) and bundle underperforming services (e.g., "Overnight + Walks" package).
- Reevaluate pricing or retire rarely used service types.
- Geo-Targeted Marketing Identify high-dropoff neighborhoods and run hyper-local campaigns (flyers, Google Local Ads, direct emails).
- Use heatmap and location breakdowns to expand services in active areas while reconsidering underperforming zones.

Conclusion & Next Steps

Trusty Tail's post-pandemic performance reflects broad behavioral shifts among pet owners. While overall volume has declined, deeper analysis reveals loyal weekday users, location-based opportunities, and signs that pricing power may have softened the blow.

Next Steps: - Conduct client surveys to validate assumptions behind the behavioral trends. - A/B test subscription and bundle offers in targeted neighborhoods. - Refresh your analytics dashboard quarterly to monitor recovery and retention in real time.

By taking data-informed action today, Trusty Tails can re-leash its market position tomorrow.