Random Forest (Ranger) Pedestrian Model

Below is the code we used to tune the model. Once we ran this to get ballpark numbers, we fine tuned the 'mtry' and 'min.node.size' values. The size of our data set greatly influenced our tuning parameter values. Increasing trees used did not improve model fit.

```
df <- read_csv("daily_counts_aggregated.csv") %>%
  filter(qaqc_volume >= 0, user_type == "Pedestrian", pass_fail == 1) %>%
  select(-user_type, -daily_volume, -maxhour, -maxday, -gap, -zero, -pass_fail, -weather_sta
  clean_names() %>%
  mutate(
    date = as.Date(date, format = "%m/%d/%Y"),
    is_holiday = if_else(date %in% us_holidays, 1, 0),
    is_weekend = if_else(weekdays(date) %in% c("Saturday", "Sunday"), 1, 0),
    counter_id = as.factor(counter_id),
    month = as.factor(format(date, "%m")),
    week = week(date),
    year = as.factor(lubridate::year(date)),
    day_of_year = yday(date),
    row_id = row_number()
  )

df_model <- df %>% drop_na()

# Stratified sampling (20% per month)
set.seed(42)
test_rows <- df_model %>%
  mutate(month_group = format(date, "%Y-%m")) %>%
  group_by(month_group) %>%
  sample_frac(0.2) %>%
  ungroup() %>%
  pull(row_id)

# Split data
```

```r
test_df <- df_model %>% filter(row_id %in% test_rows)
train_df <- df_model %>% filter(!row_id %in% test_rows)

train_df <- train_df %>% select(-date)
test_df  <- test_df %>% select(-date)
full_df  <- bind_rows(train_df, test_df)



#combine them for later complete test
full_df <- bind_rows(train_df, test_df)

# Step 6: TrainControl for caret
ctrl <- trainControl(
  method = "cv",        # cross-validation
  number = 5,           # 5-fold CV
  verboseIter = TRUE
)

# Step 7: Define hyperparameter grid
rf_grid <- expand.grid(
  mtry = c(40, 70, 86),         # Adjust depending on total # of predictors
  splitrule = "variance",     # for regression
  min.node.size = c(5, 20, 40, 80) #adjust depending on #obs
)

# Step 8: Train ranger model with caret
set.seed(123)
rf_tuned <- train(
  qaqc_volume ~ . - row_id,  # exclude row_id
  data = train_df,
  method = "ranger",
  trControl = ctrl,
  tuneGrid = rf_grid,
  num.trees = 100,
  importance = "impurity"
)

# Predict on the training set
train_preds <- predict(rf_tuned, newdata = train_df)

# Compute RMSE on training set
```

```r
train_rmse <- sqrt(mean((train_preds - train_df$qaqc_volume)^2))
cat("Training RMSE:", train_rmse, "\n")

# Step 9: Evaluate on Test
preds <- predict(rf_tuned, newdata = test_df)
rmse <- sqrt(mean((preds - test_df$qaqc_volume)^2))
cat("Test RMSE:", rmse, "\n")

#Test on entire dataset
preds <- predict(rf_tuned, newdata = full_df)
rmse <- sqrt(mean((preds - full_df$qaqc_volume)^2))
cat("Actual RMSE:", rmse, "\n")

#view mean on daily volume for context of RMSE
mean(full_df$qaqc_volume)
sd(full_df$qaqc_volume)

# Compute R^2 for full dataset
ss_total <- sum((full_df$qaqc_volume - mean(full_df$qaqc_volume))^2)
ss_res   <- sum((full_df$qaqc_volume - preds)^2)
r_squared <- 1 - (ss_res / ss_total)
cat("R-squared on full dataset:", r_squared, "\n")

# Get variable importance from caret::train model
vip_df <- varImp(rf_tuned)$importance %>%
  rownames_to_column("Variable") %>%
  arrange(desc(Overall)) %>%
  slice(1:20)  # top 20 variables

ggplot(vip_df, aes(x = reorder(Variable, Overall), y = Overall)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 20 Variable Importances", x = "", y = "Importance")
```

Data cleaning and model

Once the best tune has been determined, insert those parameters into the code below and the document will render much faster.

```r
# Load libraries
library(tidyverse)
```

```r
library(caret)
library(ranger)
library(ggplot2)
library(lubridate)
library(janitor)
library(timeDate)
library(reshape2)
library(knitr)

# Load and preprocess data
years <- unique(year(as.Date(read_csv("daily_counts_aggregated.csv")$date, format = "%m/%d/%Y
us_holidays <- as.Date(holidayNYSE(year = years))

df <- read_csv("daily_counts_aggregated.csv") %>%
  filter(qaqc_volume >= 0, user_type == "Pedestrian", pass_fail == 1) %>%
  select(-user_type, -daily_volume, -maxhour, -maxday, -gap, -zero, -pass_fail, -weather_sta
  clean_names() %>%
  mutate(
    date = as.Date(date, format = "%m/%d/%Y"),
    is_holiday = if_else(date %in% us_holidays, 1, 0),
    is_weekend = if_else(weekdays(date) %in% c("Saturday", "Sunday"), 1, 0),
    counter_id = as.factor(counter_id),
    month = as.factor(format(date, "%m")),
    week = week(date),
    year = as.factor(lubridate::year(date)),
    day_of_year = yday(date),
    row_id = row_number()
  )

#print how many obs we have
nrow(df)
```

```
[1] 20034
```

```r
df_model <- df %>% drop_na()

#Print how many obs we have
nrow(df_model)
```

```
[1] 19029
```

```r
# Stratified sampling (20% per month)
set.seed(32)
test_rows <- df_model %>%
  mutate(month_group = format(date, "%Y-%m")) %>%
  group_by(month_group) %>%
  sample_frac(0.2) %>%
  ungroup() %>%
  pull(row_id)

# Split data
test_df <- df_model %>% filter(row_id %in% test_rows)
train_df <- df_model %>% filter(!row_id %in% test_rows)

train_df <- train_df %>% select(-date)
test_df  <- test_df %>% select(-date)
full_df  <- bind_rows(train_df, test_df)
nrow(train_df)
```

```
[1] 15223
```

```r
nrow(test_df)
```

```
[1] 3806
```

```r
# Train Random Forest model
set.seed(123)
rf_tuned <- ranger(
  formula = qaqc_volume ~ . - row_id,
  data = train_df,
  num.trees = 100,
  mtry = 70,
  min.node.size = 4,
  importance = "impurity"
)
```

RMSE

```r
# Evaluate performance
train_preds <- predict(rf_tuned, data = train_df)$predictions
test_preds  <- predict(rf_tuned, data = test_df)$predictions
full_preds <- predict(rf_tuned, data = full_df)$predictions
train_rmse <- sqrt(mean((train_preds - train_df$qaqc_volume)^2))
cat("Training RMSE:", train_rmse, "\n")
```

```
Training RMSE: 171.0273
```

```r
test_rmse <- sqrt(mean((test_preds - test_df$qaqc_volume)^2))
cat("Test RMSE:", test_rmse, "\n")
```

```
Test RMSE: 276.3278
```

```r
full_rmse <- sqrt(mean((full_preds - full_df$qaqc_volume)^2))
cat("Full Dataset RMSE:", full_rmse, "\n")
```

```
Full Dataset RMSE: 196.6524
```

```r
# Contextual stats
cat("Mean Volume:", mean(full_df$qaqc_volume), "\n")
```

```
Mean Volume: 259.8743
```

```r
cat("SD Volume:", sd(full_df$qaqc_volume), "\n")
```

```
SD Volume: 715.8316
```

```r
# Variable importance
vip_df <- as.data.frame(rf_tuned$variable.importance) %>%
  rownames_to_column("Variable") %>%
  setNames(c("Variable", "Importance")) %>%
  arrange(desc(Importance)) %>%
  slice(1:20)

#combined data r^2
ss_total <- sum((full_df$qaqc_volume - mean(full_df$qaqc_volume))^2)
ss_res   <- sum((full_df$qaqc_volume - full_preds)^2)
```

```r
r_squared <- 1 - ss_res / ss_total
full_mean <- mean(full_df$qaqc_volume)
full_sd <- sd(full_df$qaqc_volume)

#test R^2
ss_total_test <- sum((test_df$qaqc_volume - mean(test_df$qaqc_volume))^2)
ss_res_test   <- sum((test_df$qaqc_volume - test_preds)^2)
r_squared_test <- 1 - ss_res_test / ss_total_test

#mean and sd
test_sd <- sd(test_df$qaqc_volume)
test_mean <- mean(test_df$qaqc_volume)
train_mean <- mean(train_df$qaqc_volume)
train_sd <- sd(train_df$qaqc_volume)

# Create long format tibble
summary_tbl <- tibble(
  Metric = c(
    "R Squared", "R Squared", "R Squared",
    "RMSE", "RMSE", "RMSE",
    "SD", "SD", "SD",
    "Mean", "Mean", "Mean"
  ),
  Set = c(
    "From Test Data", "From Both", "From Train Data",
    "From Test Data", "From Both", "From Train Data",
    "From Test Data", "From Both", "From Train Data",
    "From Test Data", "From Both", "From Train Data"
  ),
  Value = c(
    r_squared_test, r_squared, "High",
    test_rmse, full_rmse, train_rmse,
    test_sd, full_sd, train_sd,
    test_mean, full_mean, train_mean
  )
)

# Pivot to wide format
summary_tbl_wide <- summary_tbl %>%
  pivot_wider(
    names_from = Set,
    values_from = Value
```

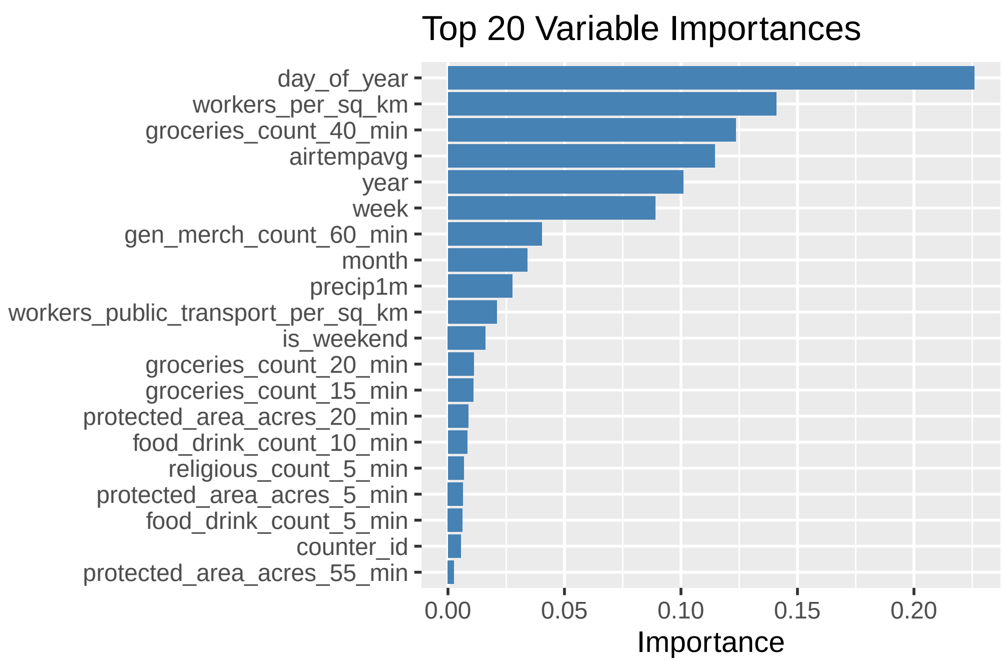
```
  )

summary_tbl_wide
```

```
# A tibble: 4 x 4
  Metric    `From Test Data` `From Both`       `From Train Data`
  <chr>     <chr>            <chr>             <chr>
1 R Squared 0.84437361759867 0.924525612527368 High
2 RMSE      276.32777336036  196.65240125815   171.027279074632
3 SD        700.551371782959 715.831601260194  719.623830761058
4 Mean      259.518129269574 259.874349676809  259.963410628654
```

Plotting

This bar plot shows the top 20 variables ranked by importance from the trained random forest model. The most influential predictor is airtempavg (average air temperature), followed by non_workday, and workers_public_transport_per_sq_km. These indicate that weather and day type (e.g., weekend or holiday) strongly influence Pedestrian volume, as does the built environment (e.g., public transit density).

```
vip_df <- vip_df %>%
  mutate(Importance = Importance / sum(Importance),
         label = scales::percent(Importance, accuracy = 0.1))
ggplot(vip_df, aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 20 Variable Importances", x = "", y = "Importance")
```

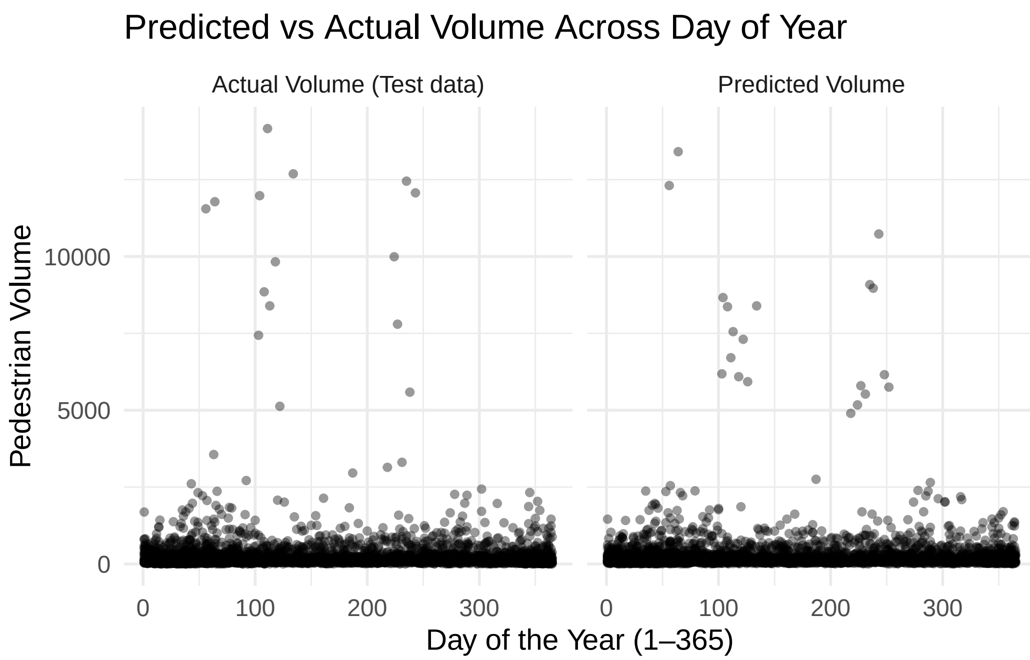



```
# Add predictions and residuals
plot_df <- test_df %>%
  mutate(
    predicted = test_preds,
    actual = qaqc_volume,
    residual = actual - predicted
  )
```

This plot shows that our model captures season variation (spring (60-151), summer (152-243), and fall (244-334)).

```
# Prepare long format for faceting
plot_df_long <- plot_df %>%
  select(day_of_year, predicted, actual) %>%
  pivot_longer(cols = c(predicted, actual), names_to = "type", values_to = "volume")

# Facet plot
ggplot(plot_df_long, aes(x = day_of_year, y = volume)) +
  geom_point(alpha = 0.4, color = "black", size = 1) +
  facet_wrap(~type, ncol = 2, labeller = labeller(type = c(
    predicted = "Predicted Volume",
    actual = "Actual Volume (Test data)"
  ))) +
```

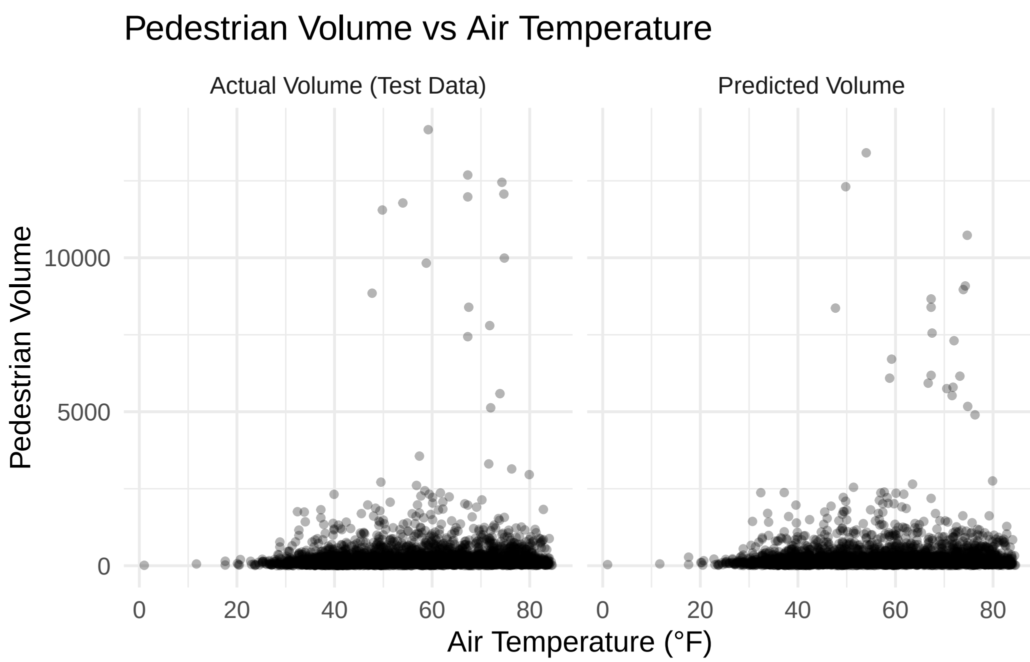
```
  labs(
    title = "Predicted vs Actual Volume Across Day of Year",
    x = "Day of the Year (1-365)",
    y = "Pedestrian Volume"
  ) +
  theme_minimal()
```



Now lets do the same side by side lot for air temperature, which shows us a similar trend.

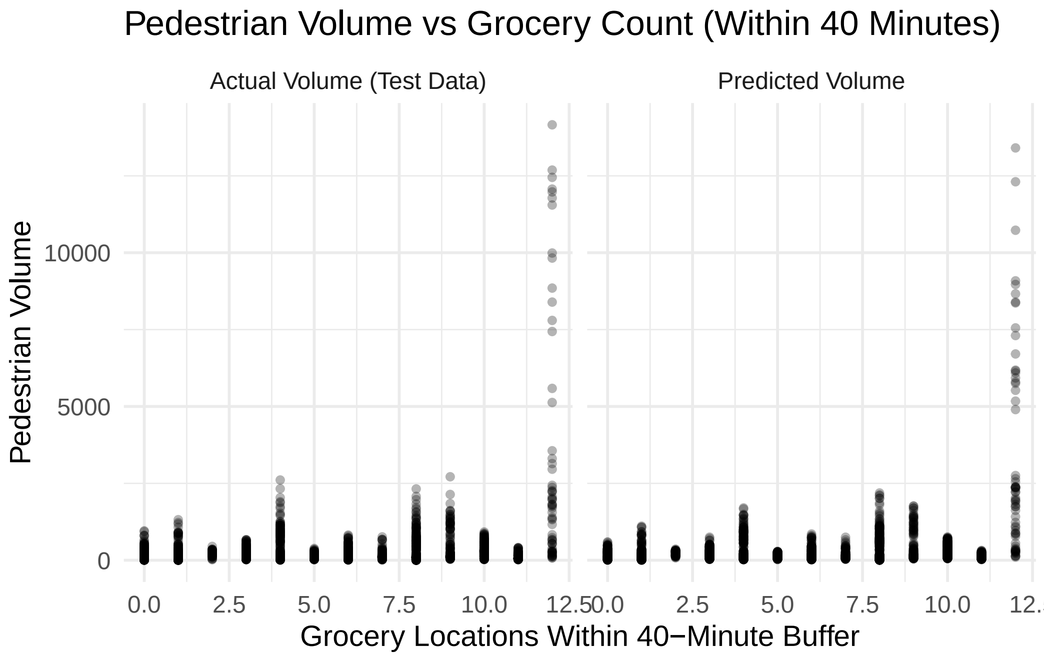
```
plot_df_long <- plot_df %>%
  select(airtempavg, predicted, actual) %>%
  pivot_longer(cols = c(predicted, actual), names_to = "type", values_to = "volume")
# Plot side-by-side (faceted)
ggplot(plot_df_long, aes(x = airtempavg, y = volume)) +
  geom_point(alpha = 0.3, color = "black", size = 1) +
  facet_wrap(~type, ncol = 2, labeller = labeller(type = c(
    predicted = "Predicted Volume",
    actual = "Actual Volume (Test Data)"
  ))) +
  labs(
    title = "Pedestrian Volume vs Air Temperature",
    x = "Air Temperature (°F)",
    y = "Pedestrian Volume"
```

```
  ) +
  theme_minimal()
```



Same thing, but with grocery stores within 40minutes.

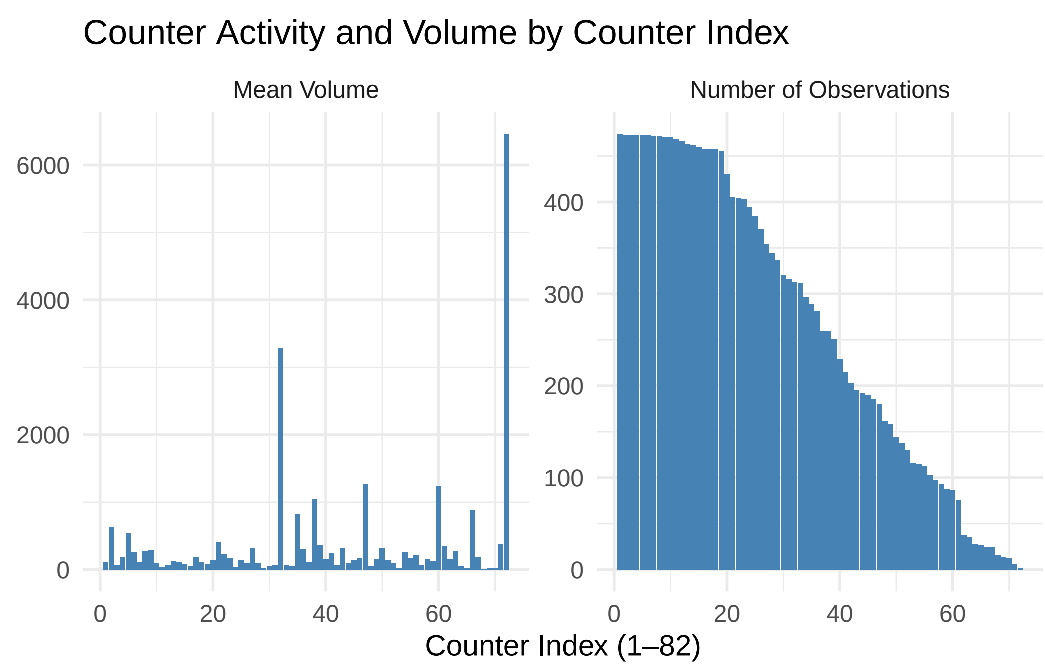
```
# Convert to long format for facet plotting
plot_df_long <- plot_df %>%
  select(groceries_count_40_min, predicted, actual) %>%
  pivot_longer(cols = c(predicted, actual), names_to = "type", values_to = "volume")

# Create side-by-side facet plot
ggplot(plot_df_long, aes(x = groceries_count_40_min, y = volume)) +
  geom_point(alpha = 0.3, color = "black", size = 1) +
  facet_wrap(~type, ncol = 2, labeller = labeller(type = c(
    predicted = "Predicted Volume",
    actual = "Actual Volume (Test Data)"
  ))) +
  labs(
    title = "Pedestrian Volume vs Grocery Count (Within 40 Minutes)",
    x = "Grocery Locations Within 40-Minute Buffer",
    y = "Pedestrian Volume"
  ) +
  theme_minimal()
```



Now let's see how many observations each counter is contributing to our dataset. We have 1 counter with an extremely high mean, but very few observations. This could be a mistake, but considering the data was hand checked, this is unlikely. Even so, it appears to only be a couple of observations, so it would have little influence on the model. Counter (approximately) 34 should be looked into due to it's relatively high mean and high observation count.

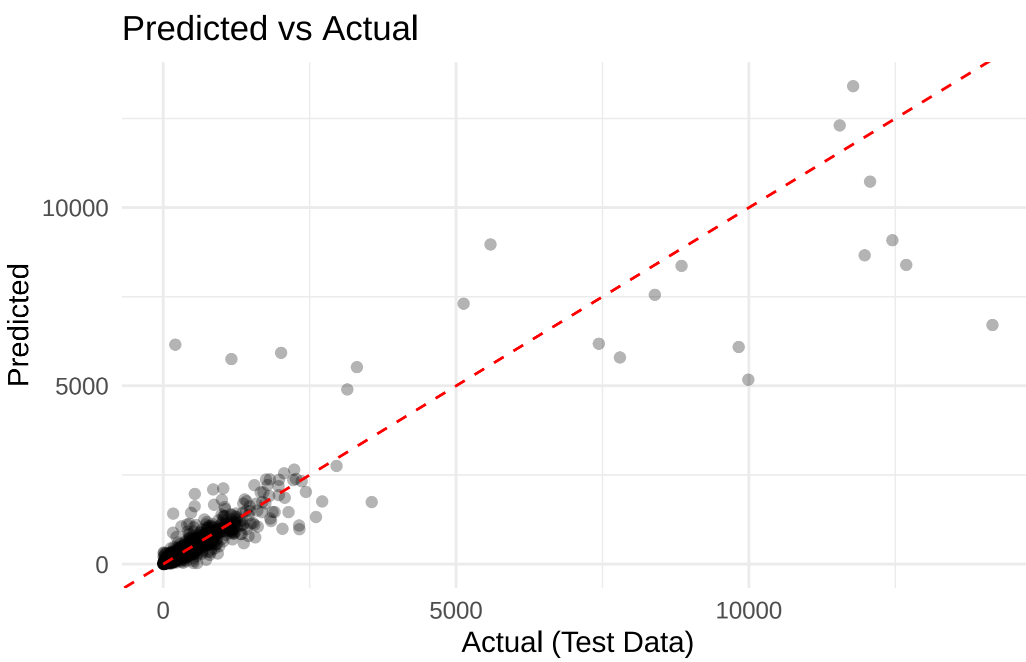
```
# Summarize data by counter_id
summary_df <- df_model %>%
  group_by(counter_id) %>%
  summarise(
    n_obs = n(),
    total_volume = sum(qaqc_volume, na.rm = TRUE),
    mean_volume = mean(qaqc_volume, na.rm = TRUE)
  ) %>%
  arrange(desc(n_obs)) %>%
  mutate(counter_index = row_number())  # 1 to 82

# Reshape for faceting
plot_df_long <- summary_df %>%
  select(counter_index, n_obs, mean_volume) %>%
  pivot_longer(cols = c(n_obs, mean_volume),
               names_to = "metric",
               values_to = "value")
```

```r
# Faceted bar plot
ggplot(plot_df_long, aes(x = counter_index, y = value)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  facet_wrap(~metric, scales = "free_y", labeller = labeller(
    metric = c(n_obs = "Number of Observations", mean_volume = "Mean Volume")
  )) +
  labs(
    title = "Counter Activity and Volume by Counter Index",
    x = "Counter Index (1-82)",
    y = NULL
  ) +
  theme_minimal()
```



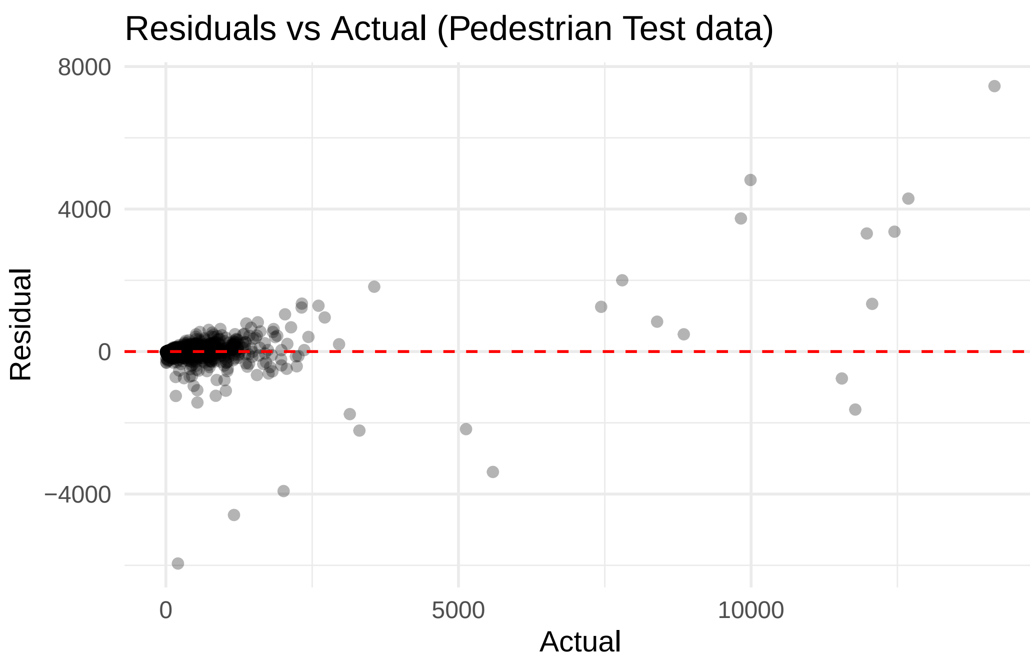
This scatterplot compares predicted values to actual observations. Ideally, points should fall along the red 45-degree line. We see that while the model performs reasonably well.

```r
# Basic diagnostic plots
ggplot(plot_df, aes(x = actual, y = predicted)) +
  geom_point(alpha = 0.3) +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Predicted vs Actual", x = "Actual (Test Data)", y = "Predicted") +
  theme_minimal()
```



This diagnostic plot shows the residuals (predicted minus actual) against actual values. A well-fitted model should show residuals randomly distributed around zero. Here, we observe that residual spread increases with actual volume, again indicating the model struggles to predict large volumes.

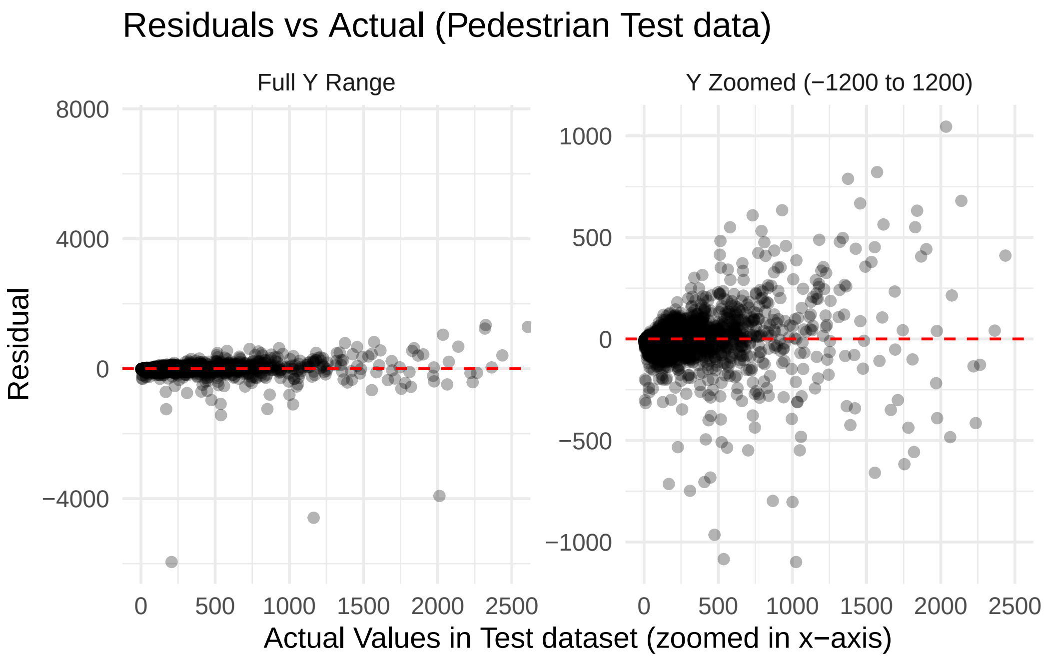
```
ggplot(plot_df, aes(x = actual, y = residual)) +
  geom_point(alpha = 0.3) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Residuals vs Actual (Pedestrian Test data)", x = "Actual", y = "Residual") +
  theme_minimal()
```



Since our residual plot is very zoomed out due to a few outliers, let's create a plot that is zoomed in. Now we will be able to see what the residuals look like for most of our data points.

```
# RESIDUALS One version: zoomed-in y
plot_df_zoomed <- plot_df %>%
  filter(residual >= -1200, residual <= 1200) %>%
  mutate(view = "Y Zoomed (-1200 to 1200)")

# One version: full y-range
plot_df_full <- plot_df %>%
  mutate(view = "Full Y Range")

# Combine for faceting
plot_df_facet <- bind_rows(plot_df_zoomed, plot_df_full)

# Plot with facet
ggplot(plot_df_facet, aes(x = actual, y = residual)) +
  geom_point(alpha = 0.3) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Residuals vs Actual (Pedestrian Test data)",
    x = "Actual Values in Test dataset (zoomed in x-axis)",
```

```
    y = "Residual"
  ) +
  coord_cartesian(xlim = c(0, 2500)) +
  facet_wrap(~view, scales = "free_y") +  # this allows separate y-axes
  theme_minimal()
```



In general, the model performs well.

```
summary_stats <- summary(plot_df[c("actual", "predicted")])
kable(summary_stats, caption = "Summary Statistics for Selected Variables")
```

Table 1: Summary Statistics for Selected Variables

actual	predicted
Min. : 1.0	Min. : 4.417
1st Qu.: 69.0	1st Qu.: 74.895
Median : 126.0	Median : 127.917
Mean : 259.5	Mean : 259.944
3rd Qu.: 246.0	3rd Qu.: 251.572
Max. :14159.0	Max. :13407.712

This scatterplot with a GAM-smoothed trend line shows a nonlinear relationship between predicted pedestrian volume and public transit worker density. Predicted volume remains modest at both low and moderate levels of transit density, with a slight dip in the midrange. However, in areas with very high transit worker density (above 150 per km^2), predicted pedestrian volumes rise sharply. This may reflect highly walkable urban cores where transit use and pedestrian activity co-locate, or it could indicate transit hubs where foot traffic is naturally higher. The pattern suggests that while transit density alone is not always a strong predictor, extreme values are associated with higher pedestrian activity, likely due to underlying urban form factors.

```r
ggplot(plot_df, aes(x = workers_public_transport_per_sq_km, y = predicted)) +
  geom_point(alpha = 0.3) +
  geom_smooth(se = FALSE, color = "darkgreen") +
  labs(title = "Predicted Volume vs Public Transit Density",
       x = "Transit Density", y = "Predicted Volume") +
  theme_minimal()
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```