Predicting Total Residential Building Permits in British Columbia

2024-12-02

```r
library(fpp3)

library(readxl) # For reading Excel files
library(tidyr)
library(purrr)
library(quantmod)

library(fable.prophet)

library(prophet)

library(kableExtra)

library(webshot2)



# Step 1: Read the data
data <- read_excel("DATA/BUILDING_PERMITS_CANADA.xlsx", sheet = 1)

# View the first few rows of the dataset
head(data)

## # A tibble: 6 × 87
##   File                 Seasonal adjustment,…¹ Geography `Type of work 6` Var
iables
##   <chr>                <chr>                  <chr>     <chr>            <ch
r>
## 1 BritishColumbiaVa… Unadjusted, current    British … Types of work, … Val
ue of…
## 2 BritishColumbiaVa… Unadjusted, current    British … Types of work, … Val
ue of…
## 3 BritishColumbiaVa… Unadjusted, current    British … Types of work, … Val
ue of…
## 4 BritishColumbiaVa… Unadjusted, current    British … Types of work, … Val
ue of…
## 5 BritishColumbiaVa… Unadjusted, current    British … Types of work, … Val
ue of…
## 6 BritishColumbiaVa… Unadjusted, current    British … Types of work, … Val
ue of…
## # ℹ abbreviated name: ¹`Seasonal adjustment, value type 4 5`
## # ℹ 82 more variables: `Type of building` <chr>, `Jan-18` <chr>,
## #   `Feb-18` <chr>, `Mar-18` <chr>, `Apr-18` <chr>, `May-18` <chr>,
## #   `Jun-18` <chr>, `Jul-18` <chr>, `Aug-18` <chr>, `Sep-18` <chr>,
```

```
## #   `Oct-18` <chr>, `Nov-18` <chr>, `Dec-18` <chr>, `Jan-19` <chr>,
## #   `Feb-19` <chr>, `Mar-19` <chr>, `Apr-19` <chr>, `May-19` <chr>,
## #   `Jun-19` <chr>, `Jul-19` <chr>, `Aug-19` <chr>, `Sep-19` <chr>, …

# Reshape the data from wide to long format
tidy_data <- data %>%
  pivot_longer(
    cols = starts_with("Jan-"), # Adjust to match the actual column patterns
    names_to = "DATE",
    values_to = "Value"
  )

# Separate the columns into specific data types if necessary
tidy_data <- tidy_data %>%
  rename(
    `Types of work, total` = `Type of work 6`,
    `Seasonal adjustment, value type` = `Seasonal adjustment, value type 4 5`
,
    `Geography` = Geography
  )

# Finalize the format
tidy_data <- tidy_data %>%
  select(
    `Types of work, total`,
    `Seasonal adjustment, value type`,
    Geography,
    DATE,
    everything()
  )

# View the reshaped data
print(tidy_data)

## # A tibble: 672 × 82
##    `Types of work, total` Seasonal adjustment,…¹ Geography DATE  File  Var
iables
##    <chr>                  <chr>                  <chr>     <chr> <chr> <ch
r>
##  1 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
##  2 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
##  3 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
##  4 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
##  5 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
##  6 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
```

```
ue of…
##  7 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
##  8 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
##  9 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
## 10 Types of work, total   Unadjusted, current    British … Jan-… Brit… Val
ue of…
## # ℹ 662 more rows
## # ℹ abbreviated name: ¹`Seasonal adjustment, value type`
## # ℹ 76 more variables: `Type of building` <chr>, `Feb-18` <chr>,
## #   `Mar-18` <chr>, `Apr-18` <chr>, `May-18` <chr>, `Jun-18` <chr>,
## #   `Jul-18` <chr>, `Aug-18` <chr>, `Sep-18` <chr>, `Oct-18` <chr>,
## #   `Nov-18` <chr>, `Dec-18` <chr>, `Feb-19` <chr>, `Mar-19` <chr>,
## #   `Apr-19` <chr>, `May-19` <chr>, `Jun-19` <chr>, `Jul-19` <chr>, …

# Step 2: Identify date columns (adjust based on your data)
# Assuming all date columns start with a month abbreviation (e.g., "Jan-", "F
eb-", etc.)
date_columns <- grep("^(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)-", n
ames(data), value = TRUE)

# Step 3: Reshape data from wide to long format
tidy_data <- data %>%
  pivot_longer(
    cols = all_of(date_columns),    # Only reshape the date columns
    names_to = "DATE",              # New column for dates
    values_to = "Value"             # New column for values
  )

# Step 4: Clean and reformat DATE column
# Assuming dates are formatted as "MMM-YY" (e.g., "Jan-23"), convert them to
proper date format
tidy_data <- tidy_data %>%
  mutate(
    Value = as.numeric(Value),
    DATE = as.Date(paste0("01-", DATE), format = "%d-%b-%y")  # Adding "01-"
assumes day 1 for month-year
  )

# Step 5: Reorder columns to match the desired output
tidy_data <- tidy_data %>%
  select(
    #`Type of work 6`,
    #`Seasonal adjustment, value type 4 5`,
    #File,
    Geography,
    DATE,
    Value,
```

```
    Variables,
    `Type of building`
  )

# Step 6: Save the transformed data to a new CSV file
write.csv(tidy_data, "Transformed_Building_Permits.csv", row.names = FALSE)

# Preview of table properties
print(glimpse(tidy_data))
```

```
## Rows: 7,776
## Columns: 5
## $ Geography          <chr> "British Columbia", "British Columbia", "Britis
h Co…
## $ DATE               <date> 2018-01-01, 2018-02-01, 2018-03-01, 2018-04-01
, 20…
## $ Value              <dbl> 1214273, 1342322, 1565004, 1228669, 1716029, 17
7399…
## $ Variables          <chr> "Value of permits", "Value of permits", "Value
of p…
## $ `Type of building` <chr> "Total residential and non-residential", "Total
res…
## # A tibble: 7,776 × 5
##    Geography        DATE          Value Variables        `Type of building`
##    <chr>            <date>        <dbl> <chr>            <chr>
##  1 British Columbia 2018-01-01 1214273 Value of permits Total residential
and n…
##  2 British Columbia 2018-02-01 1342322 Value of permits Total residential
and n…
##  3 British Columbia 2018-03-01 1565004 Value of permits Total residential
and n…
##  4 British Columbia 2018-04-01 1228669 Value of permits Total residential
and n…
##  5 British Columbia 2018-05-01 1716029 Value of permits Total residential
and n…
##  6 British Columbia 2018-06-01 1773993 Value of permits Total residential
and n…
##  7 British Columbia 2018-07-01 1565949 Value of permits Total residential
and n…
##  8 British Columbia 2018-08-01 2234310 Value of permits Total residential
and n…
##  9 British Columbia 2018-09-01 1384002 Value of permits Total residential
and n…
## 10 British Columbia 2018-10-01 1643830 Value of permits Total residential
and n…
## # ℹ 7,766 more rows
```

```
# Preview the transformed data
print(knitr::kable(head(tidy_data)))
```

```
##
##
## |Geography        |DATE       |   Value|Variables        |Type of building
|
## |:----------------|:----------|-------:|:----------------|:----------------
----------------------|
## |British Columbia |2018-01-01 | 1214273|Value of permits |Total residentia
l and non-residential |
## |British Columbia |2018-02-01 | 1342322|Value of permits |Total residentia
l and non-residential |
## |British Columbia |2018-03-01 | 1565004|Value of permits |Total residentia
l and non-residential |
## |British Columbia |2018-04-01 | 1228669|Value of permits |Total residentia
l and non-residential |
## |British Columbia |2018-05-01 | 1716029|Value of permits |Total residentia
l and non-residential |
## |British Columbia |2018-06-01 | 1773993|Value of permits |Total residentia
l and non-residential |
```

```r
print(knitr::kable(
  tidy_data |>
    filter(`Type of building` %in% c('Total non-residential','Total residenti
al','Total residential and non-residential')) |>
    group_by(Geography,`Type of building`,Variables) |>
    summarise(Value = sum(Value), Count = n()) |>
    mutate(Value = format(Value,big.mark = ',')) |>
    select(Geography,`Type of building`,Variables, Value, Count)
))
```

```
## `summarise()` has grouped output by 'Geography', 'Type of building'. You c
an
## override using the `.groups` argument.
```

```
##
##
## |Geography                    |Type of building                      |Variab
les         |Value       | Count|
## |:----------------------------|:-------------------------------------|:-----
------------|:-----------|-----:|
## |Alberta                      |Total non-residential                 |Number
of permits |60,961      |    81|
## |Alberta                      |Total non-residential                 |Value
of permits  |34,143,983  |    81|
## |Alberta                      |Total residential                     |Number
of permits |295,759     |    81|
## |Alberta                      |Total residential                     |Value
of permits  |61,561,561  |    81|
## |Alberta                      |Total residential and non-residential |Number
of permits |356,720     |    81|
## |Alberta                      |Total residential and non-residential |Value
```

```
of permits  |95,705,539  |    81|
## |British Columbia           |Total non-residential                 |Number
of permits |60,561      |    81|
## |British Columbia           |Total non-residential                 |Value
of permits  |44,106,405  |    81|
## |British Columbia           |Total residential                     |Number
of permits |177,109     |    81|
## |British Columbia           |Total residential                     |Value
of permits  |98,651,472  |    81|
## |British Columbia           |Total residential and non-residential |Number
of permits |237,670     |    81|
## |British Columbia           |Total residential and non-residential |Value
of permits  |142,757,877 |    81|
## |Kamloops, British Columbia |Total non-residential                 |Number
of permits |965         |    81|
## |Kamloops, British Columbia |Total non-residential                 |Value
of permits  |821,575     |    81|
## |Kamloops, British Columbia |Total residential                     |Number
of permits |3,469       |    81|
## |Kamloops, British Columbia |Total residential                     |Value
of permits  |1,280,120   |    81|
## |Kamloops, British Columbia |Total residential and non-residential |Number
of permits |4,434       |    81|
## |Kamloops, British Columbia |Total residential and non-residential |Value
of permits  |2,101,696   |    81|
## |Manitoba                   |Total non-residential                 |Number
of permits |15,589      |    81|
## |Manitoba                   |Total non-residential                 |Value
of permits  |10,683,730  |    81|
## |Manitoba                   |Total residential                     |Number
of permits |77,297      |    81|
## |Manitoba                   |Total residential                     |Value
of permits  |14,282,642  |    81|
## |Manitoba                   |Total residential and non-residential |Number
of permits |92,886      |    81|
## |Manitoba                   |Total residential and non-residential |Value
of permits  |24,966,374  |    81|
## |Ontario                    |Total non-residential                 |Number
of permits |181,742     |    81|
## |Ontario                    |Total non-residential                 |Value
of permits  |119,013,927 |    81|
## |Ontario                    |Total residential                     |Number
of permits |721,348     |    81|
## |Ontario                    |Total residential                     |Value
of permits  |217,133,163 |    81|
## |Ontario                    |Total residential and non-residential |Number
of permits |903,090     |    81|
## |Ontario                    |Total residential and non-residential |Value
of permits  |336,147,089 |    81|
## |Saskatchewan               |Total non-residential                 |Number
```

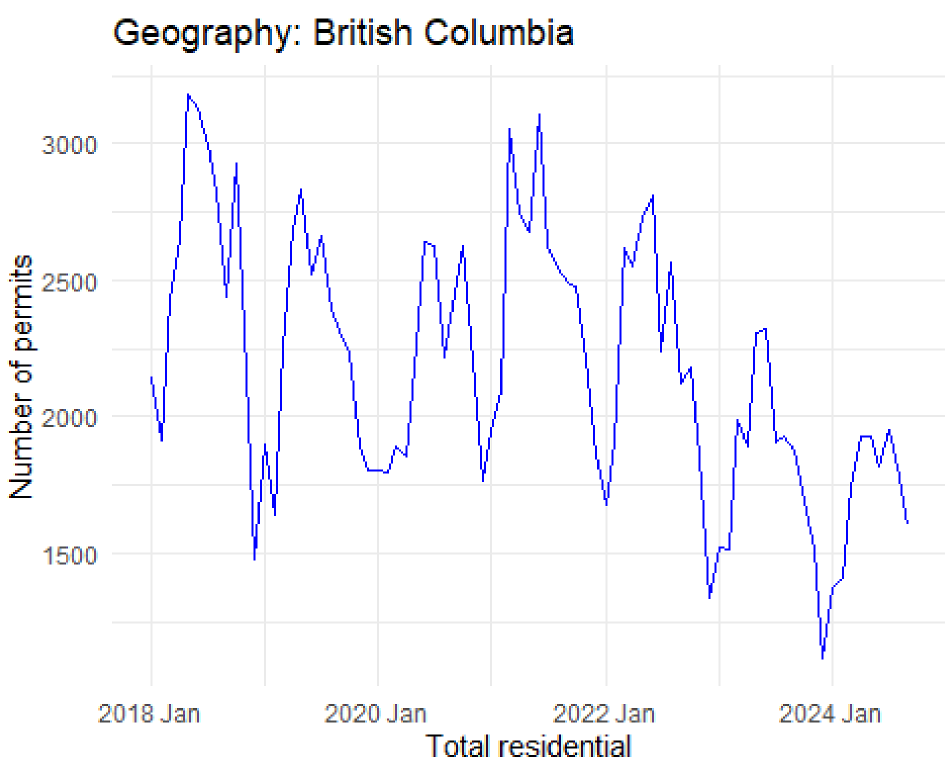
```
of permits |11,057       |     81|
## |Saskatchewan                |Total non-residential                     |Value
of permits  |6,759,061    |     81|
## |Saskatchewan                |Total residential                         |Number
of permits |48,173       |     81|
## |Saskatchewan                |Total residential                         |Value
of permits  |6,638,812    |     81|
## |Saskatchewan                |Total residential and non-residential |Number
of permits |59,230       |     81|
## |Saskatchewan                |Total residential and non-residential |Value
of permits  |13,397,876   |     81|
```

```r
# Choose just the data we need to plot.
filtered_data <- tidy_data |>
  filter(`Type of building` %in% c('Total non-residential', 'Total residentia
l', 'Total residential and non-residential')) |>
  filter(`Type of building` %in% c('Total residential')) |>
  #filter(`Geography` %in% c('Alberta','British Columbia','Ontario')) |>
  filter(`Variables` %in% c('Number of permits')) |>
  filter(`Geography` %in% c('British Columbia')) |>
  mutate(DATE = yearmonth(DATE)) |>
  select(Geography,`Type of building`,Variables, Value, DATE) |>
  #arrange(`Type of building`,Geography,Variables) |>
  as_tsibble(index = DATE, key = c(Geography,`Type of building`,Variables))

head(filtered_data)
```

```
## # A tsibble: 6 x 5 [1M]
## # Key:       Geography, Type of building, Variables [1]
##   Geography        `Type of building` Variables         Value     DATE
##   <chr>            <chr>              <chr>             <dbl>    <mth>
## 1 British Columbia Total residential  Number of permits  2145 2018 Jan
## 2 British Columbia Total residential  Number of permits  1917 2018 Feb
## 3 British Columbia Total residential  Number of permits  2436 2018 Mar
## 4 British Columbia Total residential  Number of permits  2648 2018 Apr
## 5 British Columbia Total residential  Number of permits  3181 2018 May
## 6 British Columbia Total residential  Number of permits  3129 2018 Jun
```

```r
# Loop to generate and display plots for each category
for (var in unique(filtered_data$Variables)) {
  for (type in unique(filtered_data$`Type of building`)) {
    for (geo in unique(filtered_data$Geography)) {
      # Subset the data for the current combination
      subset_data <- filtered_data |>
        filter(Geography == geo, `Type of building` == type, Variables == var
)

      # Generate the plot

      p <- autoplot(subset_data, Value) +
```

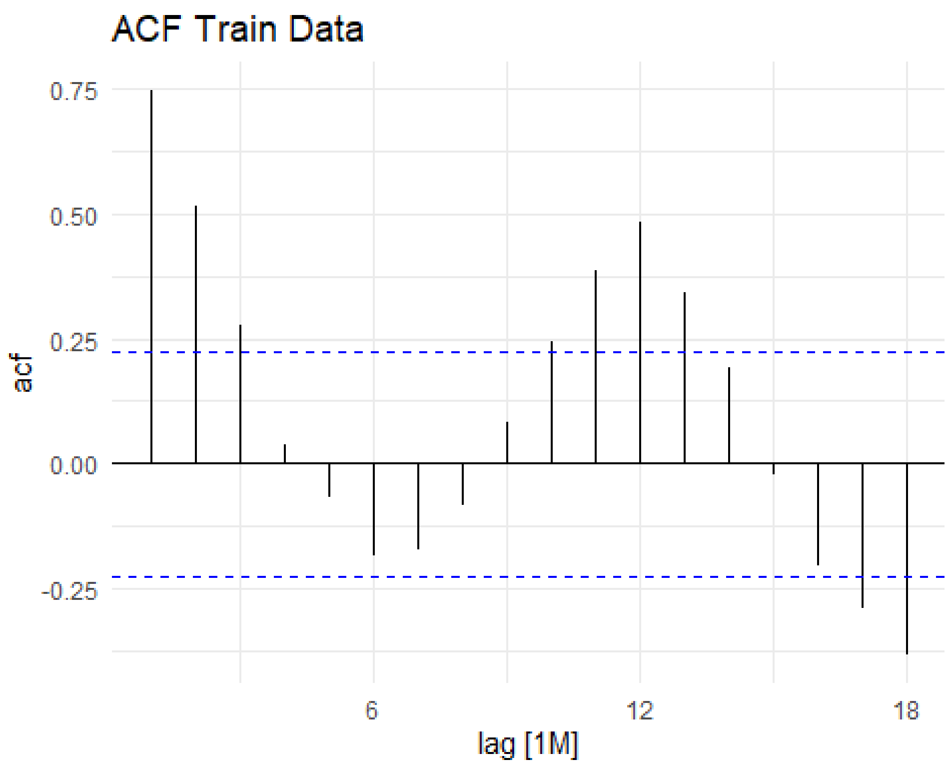
```
         geom_line(color = "blue") +
         labs(title = paste("Geography:", geo),y = var, x = type) +
         theme_minimal()

       #par(mfrow = c(1, 2))
       # Print the plot
       print(p)
       # par(mfrow = c(1,1))

       ggsave(paste0("images/","01_", geo, "_", type, "_", var, "_plot.jpg"),
plot = p, width = 8, height = 6,create.dir = TRUE)

       data <- subset_data

       #2. Split Data
       #Split the data into training and testing sets for evaluation:
       train_data <- data %>% filter(DATE < yearmonth("2024 May"))
       test_data <- data %>% filter(DATE >= yearmonth("2024 May"))
     }
   }
}
```



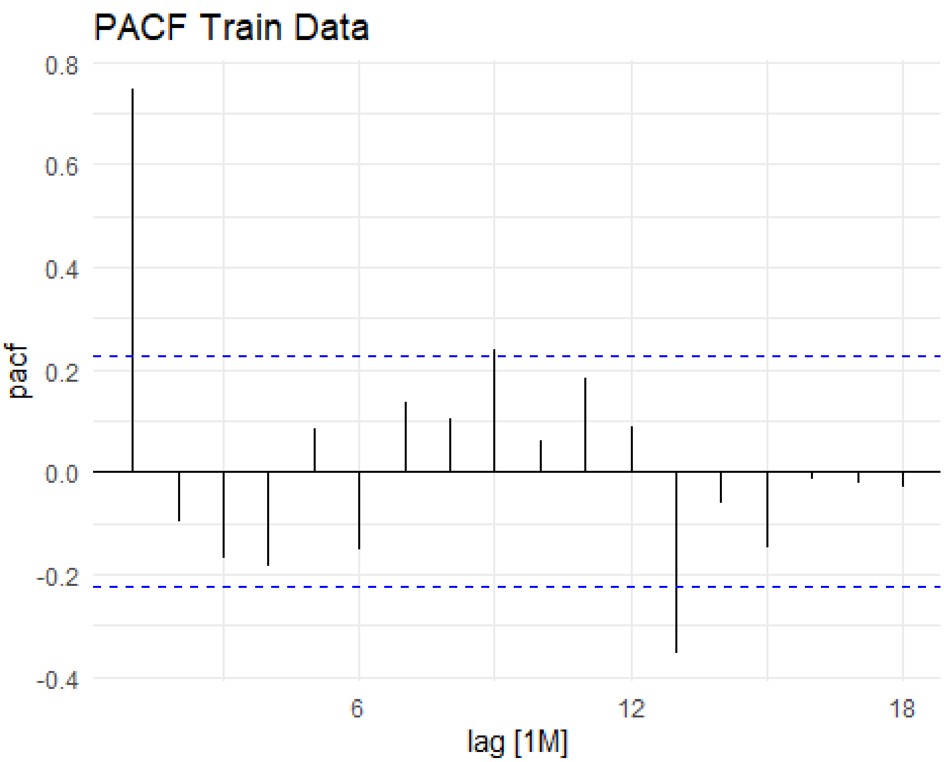
ARIMA

Check for Stationarity

```
# Plot ACF and PACF
acf_plot <- ACF(train_data, Value) %>% autoplot() +
  labs(title = "ACF Train Data") +
  theme_minimal()

pacf_plot <- PACF(train_data, Value) %>% autoplot() +
  labs(title = "PACF Train Data") +
  theme_minimal()

ggsave(paste0("images/","02_acf_plot.jpg"), plot = acf_plot,create.dir = TRUE
)

## Saving 5 x 4 in image

ggsave(paste0("images/","02_pacf_plot.jpg"), plot = pacf_plot,create.dir = TR
UE)

## Saving 5 x 4 in image

print(acf_plot)
```




```
print(pacf_plot)
```




```
# Perform stationarity test
library(urca)

## Warning: package 'urca' was built under R version 4.3.3

adf_test <- ur.df(train_data$Value, type = "drift")
summary(adf_test)

##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -627.77 -212.25   -9.24  222.23  911.14
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 606.93972  190.04076   3.194  0.00209 **
## z.lag.1      -0.27366    0.08397  -3.259  0.00172 **
```

```
## z.diff.lag    0.08936    0.11853   0.754  0.45342
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 323.8 on 71 degrees of freedom
## Multiple R-squared:  0.1324, Adjusted R-squared:  0.1079
## F-statistic: 5.417 on 2 and 71 DF,  p-value: 0.006467
##
##
## Value of test-statistic is: -3.2589 5.3102
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.51 -2.89 -2.58
## phi1  6.70  4.71  3.86
```

Fit ARIMA and Benchmark Models

```
# Fit ARIMA model
arima_model <- train_data %>% model(ARIMA(Value))

# Fit benchmark models
benchmark_models <- train_data %>% model(
  Mean = MEAN(Value),
  Naive = NAIVE(Value),
  Drift = RW(Value ~ drift())
)

mean_Model <- train_data %>% model(
  Mean = MEAN(Value)
)

naive_Model <- train_data %>% model(
  Naive = NAIVE(Value)
)

drift_Model <- train_data %>% model(
  Drift = RW(Value ~ drift())
)
```

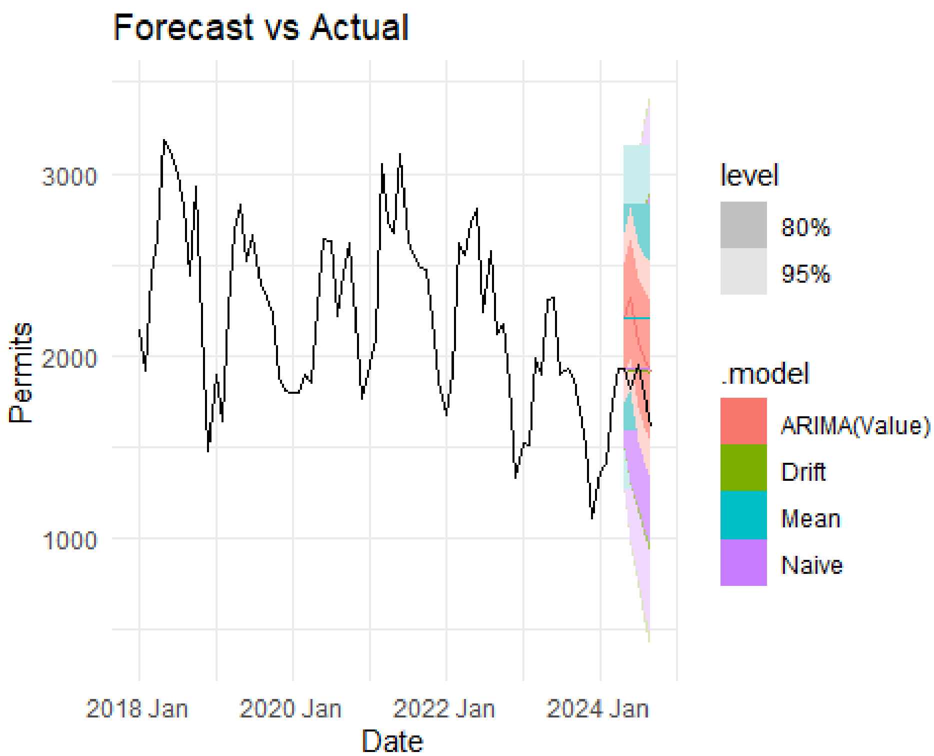
Evaluate Models

```
# Forecast using all models
forecasts <- bind_rows(
  arima_model %>% forecast(h = nrow(test_data)),
  benchmark_models %>% forecast(h = nrow(test_data))
)

# Evaluate accuracy
arima_accuracy_metrics <- forecasts %>%
  accuracy(test_data)
```

```
# Display evaluation
print(arima_accuracy_metrics)

## # A tibble: 4 × 13
##   .model    Geography `Type of building` Variables .type    ME  RMSE   MAE
MPE
##   <chr>     <chr>     <chr>              <chr>     <chr> <dbl> <dbl> <dbl>
<dbl>
## 1 ARIMA(V… British … Total residential  Number o… Test  -281.  308.  281.
-15.7
## 2 Drift     British … Total residential  Number o… Test  -103.  158.  116.
-6.17
## 3 Mean      British … Total residential  Number o… Test  -394.  412.  394.
-22.2
## 4 Naive     British … Total residential  Number o… Test  -112.  166.  120.
-6.67
## # ℹ 4 more variables: MAPE <dbl>, MASE <dbl>, RMSSE <dbl>, ACF1 <dbl>
```

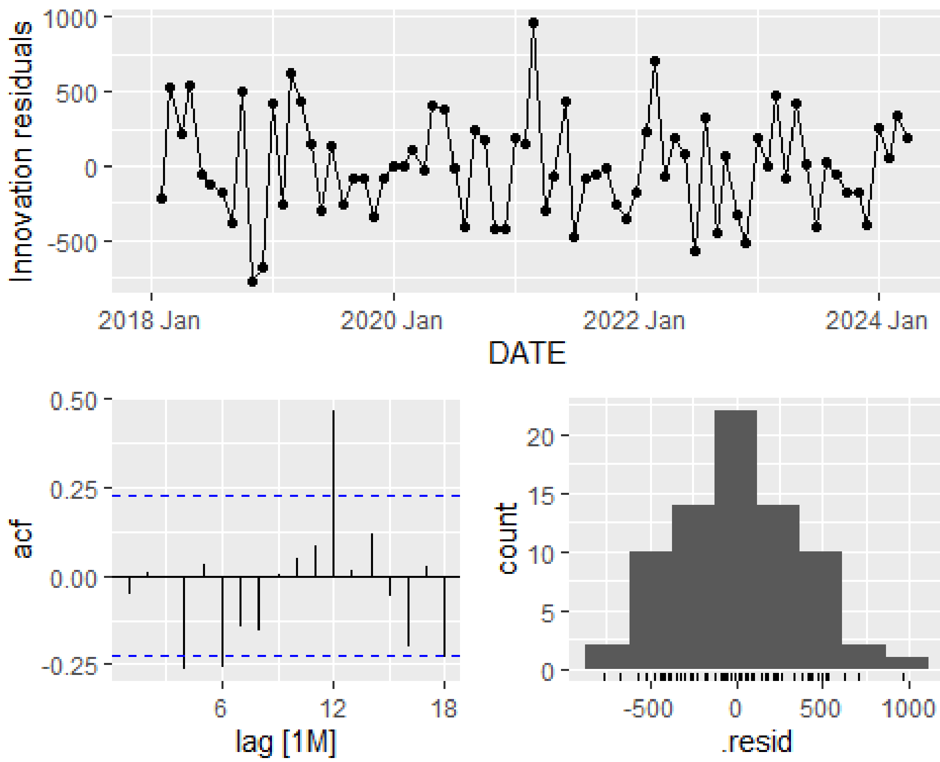
Visualize forecasts and actual values

```
arima_forecast_plot <-
forecasts %>%
  autoplot(data) +
  labs(title = "Forecast vs Actual",
       y = "Permits", x = "Date") +
  theme_minimal()

ggsave(paste0("images/","03_arima_forecast_plot.jpg"), plot = arima_forecast_
plot,create.dir = TRUE)

## Saving 5 x 4 in image

print(arima_forecast_plot)
```



Residual Diagnostics for ARIMA and Benchmark Models

```
plot <- gg_tsresiduals(drift_Model)
ggsave("images/04_drift_residuals_plot.png", plot, width = 8, height = 6)

## Warning: Removed 1 row containing missing values or values outside the sca
le range
## (`geom_line()`).

## Warning: Removed 1 row containing missing values or values outside the sca
le range
## (`geom_point()`).

## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_bin()`).

print(plot)

## Warning: Removed 1 row containing missing values or values outside the sca
le range
## (`geom_line()`).

## Warning: Removed 1 row containing missing values or values outside the sca
le range
## (`geom_point()`).

## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_bin()`).
```




```r
# png("images/04_drift_residuals_plot.png", width = 800, height = 600)
# gg_tsresiduals(drift_Model)
# dev.off()
```

Statistical Tests: Box-Pierce and Ljung-Box

```r
# Augment model to extract residuals
drift_aug <- drift_Model |> augment()

# Perform Box-Pierce Test
box_pierce_results <- drift_aug |>
  features(.innov, box_pierce, lag = 10) |>
  rename(Box_Pierce_pvalue = bp_pvalue)

# Perform Ljung-Box Test
ljung_box_results <- drift_aug |>
  features(.innov, ljung_box, lag = 10) |>
  rename(Ljung_Box_pvalue = lb_pvalue)

# Combine the Results
residual_tests <- bind_cols(
  box_pierce_results |> select(Box_Pierce_pvalue),
  ljung_box_results |> select(Ljung_Box_pvalue)
)

# Display the Results
```

```
print(knitr::kable(residual_tests, caption = "Residual Diagnostic Tests (Box-
Pierce & Ljung-Box)"))
```

```
##
##
## Table: Residual Diagnostic Tests (Box-Pierce & Ljung-Box)
##
## | Box_Pierce_pvalue| Ljung_Box_pvalue|
## |-----------------:|----------------:|
## |         0.1722768|        0.1129104|
```

EXPONENTIAL SMOOTHING

Fit Models

```
#Fit various exponential smoothing models to the training data.
#Simple Exponential Smoothing (SES):
ses_model <- train_data %>%
model(SES = ETS(Value ~ error("A") + trend("N") + season("N")))

#Holt's Linear Trend Method:
holt_model <- train_data %>%
model(Holt = ETS(Value ~ error("A") + trend("A") + season("N")))

#Holt-Winters Seasonal Methods:
hw_additive_model <- train_data %>%
model(HoltWinters_Additive = ETS(Value ~ error("A") + trend("A") + season("A"
)))
hw_multiplicative_model <- train_data %>%
model(HoltWinters_Multiplicative = ETS(Value ~ error("M") + trend("A") + seas
on("M")))
```

Forecast Future Values

```
#Generate forecasts for the test period:
forecasts <- bind_rows(
  ses_model %>% forecast(h = nrow(test_data)),
  holt_model %>% forecast(h = nrow(test_data)),
  hw_additive_model %>% forecast(h = nrow(test_data)),
  hw_multiplicative_model %>% forecast(h = nrow(test_data))
)
```

Evaluate Models

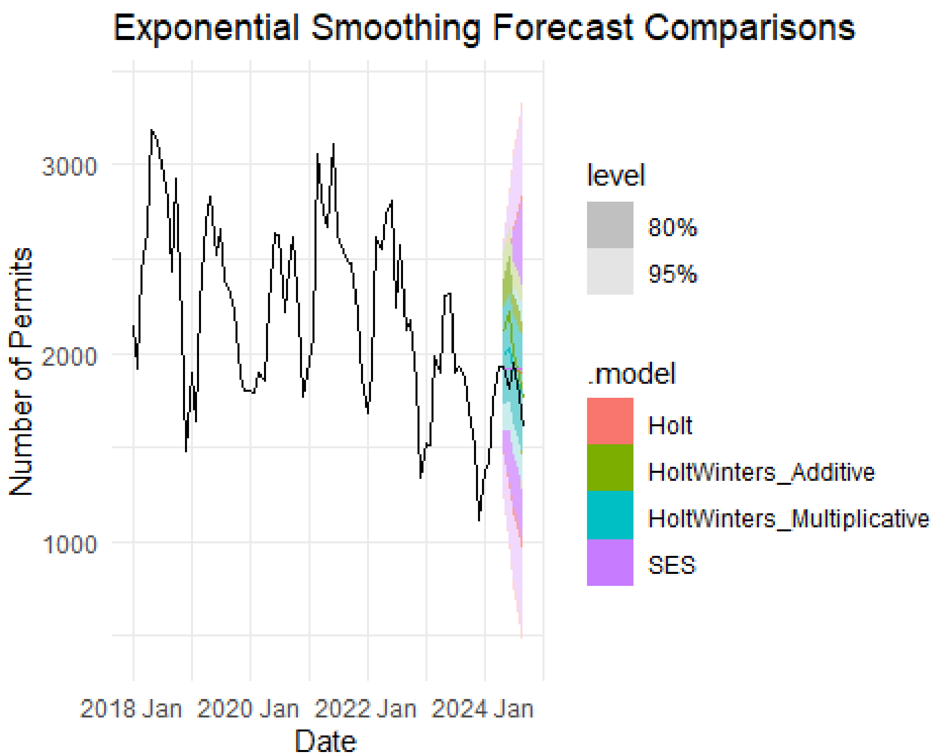
```
#Calculate forecast accuracy using metrics like RMSE, MAE, and MAPE:
exponential_smoothing_accuracy_metrics <- forecasts %>%
  accuracy(test_data)

print(exponential_smoothing_accuracy_metrics)
```

```
## # A tibble: 4 × 13
##   .model   Geography `Type of building` Variables .type     ME  RMSE   MAE
```

```
MPE
##   <chr>    <chr>     <chr>              <chr>     <chr>   <dbl> <dbl> <dbl> 
<dbl>
## 1 Holt     British … Total residential  Number o… Test    -94.3  152.  113. 
-5.68
## 2 HoltWin… British … Total residential  Number o… Test   -177.   216.  177. 
-9.85
## 3 HoltWin… British … Total residential  Number o… Test    -89.9  128.  103. 
-5.20
## 4 SES      British … Total residential  Number o… Test   -102.   160.  118. 
-6.11
## # ℹ 4 more variables: MAPE <dbl>, MASE <dbl>, RMSSE <dbl>, ACF1 <dbl>
```

Visualize Results

```
#Plot the actual values and forecasts for each method:
exponential_smooth_forecast_plot <- forecasts %>%
autoplot(data) +
labs(title = "Exponential Smoothing Forecast Comparisons", y = "Number of Per
mits", x = "Date") +
theme_minimal()

ggsave(paste0("images/","05_exponential_smooth_forecast_plot.jpg"), plot = ex
ponential_smooth_forecast_plot,create.dir = TRUE)

## Saving 5 x 4 in image

print(exponential_smooth_forecast_plot)
```

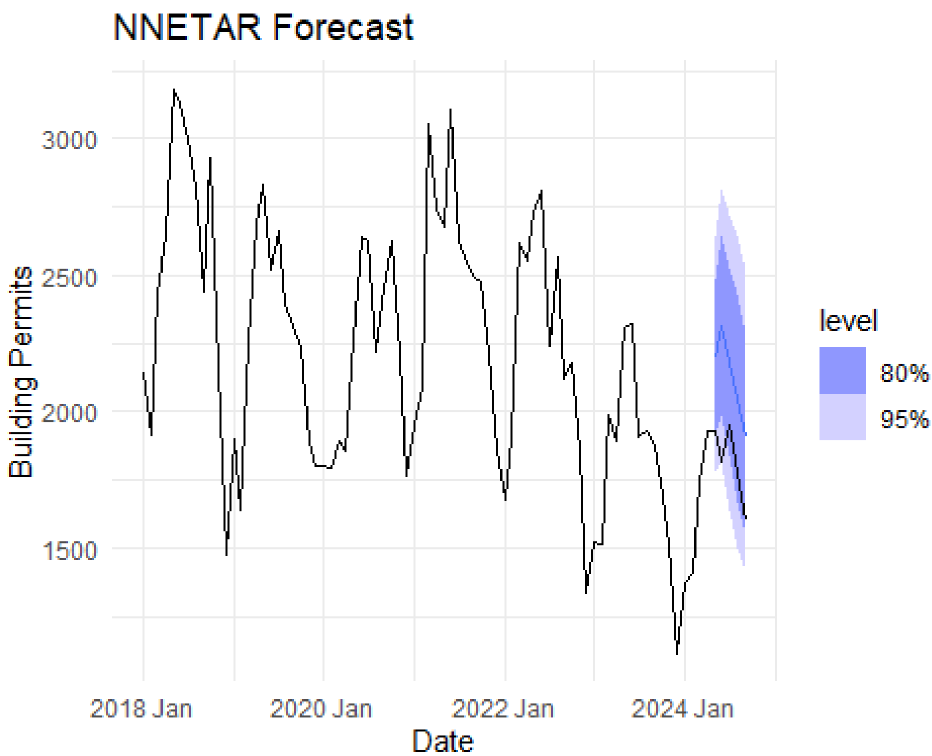



```
#print(forecasts)
```

NEURAL NETWORK AUTOREGRESSION (NNETAR) AND PROPHET

NNETAR

```
nnetar_model <- train_data %>%
  model(NNETAR = NNETAR(sqrt(Value)))

# Forecast with NNETAR Model
nnetar_forecast <- nnetar_model %>%
  forecast(h = nrow(test_data))

# Plot NNETAR Forecast
nnetar_forecast_plot <- autoplot(nnetar_forecast, data) +
  labs(title = "NNETAR Forecast", y = "Building Permits", x = "Date") +
  theme_minimal()

ggsave(paste0("images/","06_nnetar_forecast_plot.jpg"), plot = nnetar_forecas
t_plot,create.dir = TRUE)

## Saving 5 x 4 in image

print(nnetar_forecast_plot)
```




```
nnetar_accuracy <- nnetar_forecast %>%
  accuracy(test_data)

# Combine Accuracy Metrics
nnetar_accuracy <- bind_rows(
  mutate(nnetar_accuracy, .model = "NNETAR")
)

# Display Accuracy Metrics
print(knitr::kable(nnetar_accuracy, caption = "NNETAR Model Accuracy"))

##
##
## Table: NNETAR Model Accuracy
##
## |.model |Geography        |Type of building  |Variables         |.type |
ME|     RMSE|       MAE|        MPE|      MAPE| MASE| RMSSE|        ACF1|
## |:------|:----------------|:-----------------|:-----------------|:-----|--
-------:|-------:|--------:|---------:|--------:|----:|-----:|----------:|
## |NNETAR |British Columbia |Total residential |Number of permits |Test  | -
311.6945| 325.974| 311.6945| -17.29955| 17.29955|   NaN|   NaN| -0.4358787|
```

Prophet model

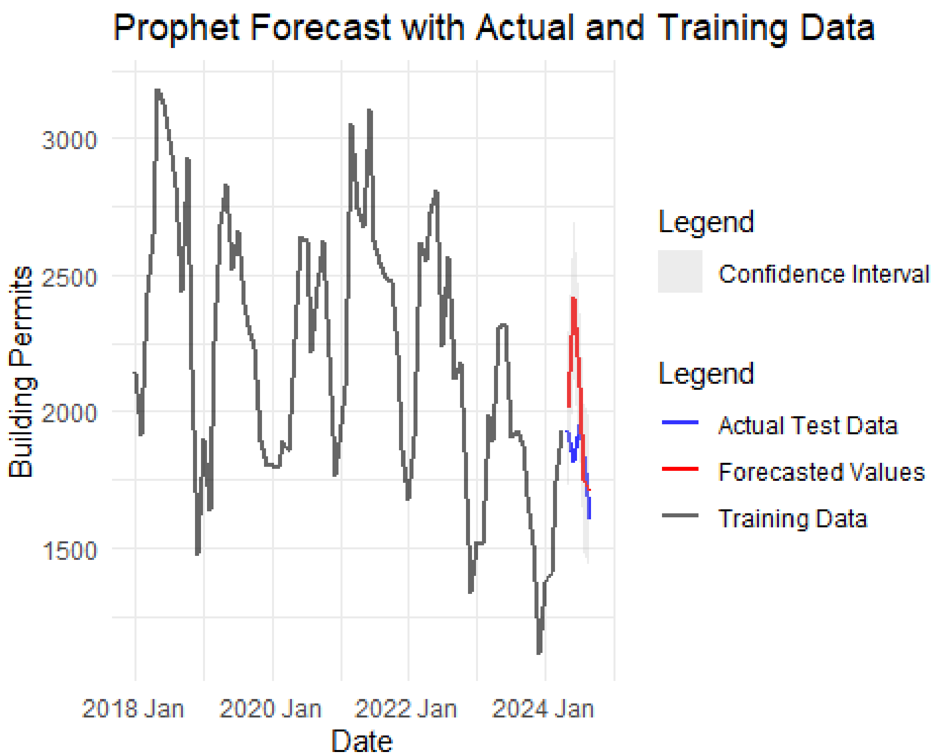
```
# Define the training dataset with proper dates
N <- nrow(data)
n <- 5 # Number of periods to forecast
```

```r
# Create the training data
train_dates <- data$DATE[1:(N - n)] # Use actual dates
train_values <- data$Value[1:(N - n)]

# Create a data frame for Prophet
df_prophet <- data.frame(ds = train_dates, y = train_values) # ds = dates, y
= values
head(df_prophet) # Verify the structure

##         ds    y
## 1 2018 Jan 2145
## 2 2018 Feb 1917
## 3 2018 Mar 2436
## 4 2018 Apr 2648
## 5 2018 May 3181
## 6 2018 Jun 3129

# Fit the Prophet model
m <- prophet(df_prophet)

## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to
override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to ov
erride this.

# Generate future dates starting after the last date in the training data
future_prophet <- make_future_dataframe(m, periods = n, freq = "month")
tail(future_prophet) # Verify the future dates

##            ds
## 76 2024-04-01
## 77 2024-05-01
## 78 2024-06-01
## 79 2024-07-01
## 80 2024-08-01
## 81 2024-09-01

# Forecast the future values
forecast_prophet <- predict(m, future_prophet)

forecast_data <- forecast_prophet %>%
  inner_join(test_data, by = c("ds" = "DATE")) %>%  # Match forecast dates wi
th test_data dates
  select(ds, yhat, yhat_lower, yhat_upper)  # Select necessary columns from t
he forecast

# Create the plot
prophet_forecast_plot <- ggplot() +
  # Plot training data
```

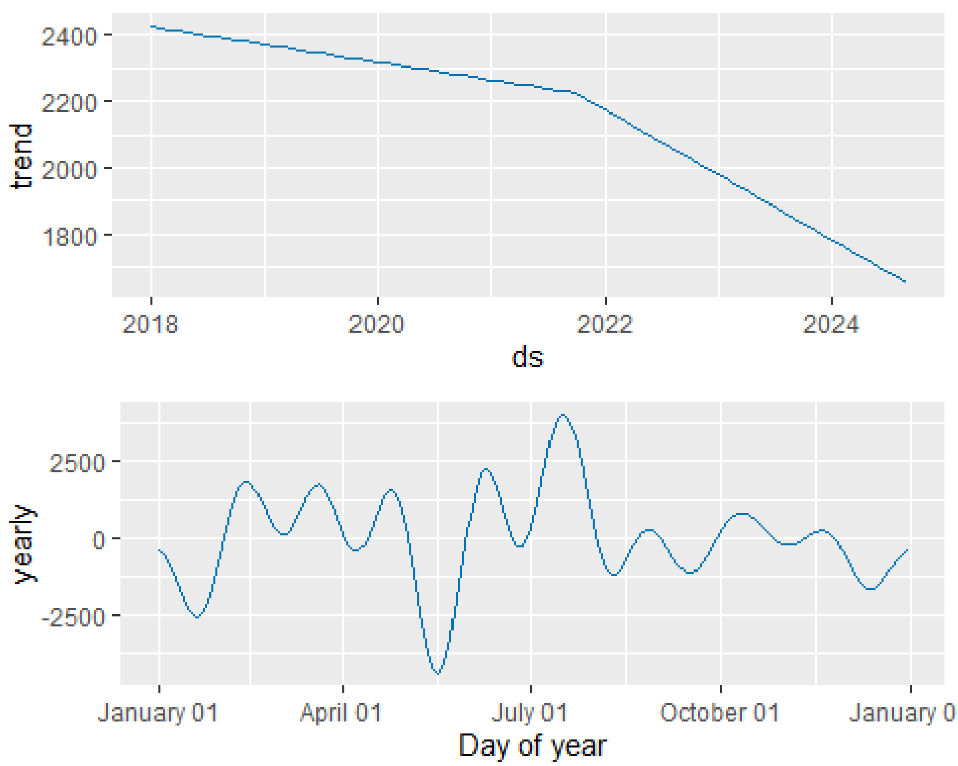
```r
  geom_line(data = df_prophet, aes(x = ds, y = y, color = "Training Data"), s
ize = 1, alpha = 0.6) +
  # Plot actual test data
  geom_line(data = test_data, aes(x = DATE, y = Value, color = "Actual Test D
ata"), size = 1, alpha = 0.8) +
  # Plot forecasted values
  geom_line(data = forecast_data, aes(x = ds, y = yhat, color = "Forecasted V
alues"), size = 1) +
  # Add confidence interval
  geom_ribbon(data = forecast_data, aes(x = ds, ymin = yhat_lower, ymax = yha
t_upper, fill = "Confidence Interval"), alpha = 0.3) +
  # Customize colors and labels for the legend
  scale_color_manual(
    name = "Legend",
    values = c("Training Data" = "black", "Actual Test Data" = "blue", "Forec
asted Values" = "red")
  ) +
  scale_fill_manual(
    name = "Legend",
    values = c("Confidence Interval" = "grey")
  ) +
  # Add title and labels
  ggtitle("Prophet Forecast with Actual and Training Data") +
  labs(
    x = "Date",
    y = "Building Permits"
  ) +
  theme_minimal() +
  theme(legend.position = "right")  # Place legend at the right
```

```
## 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.
```

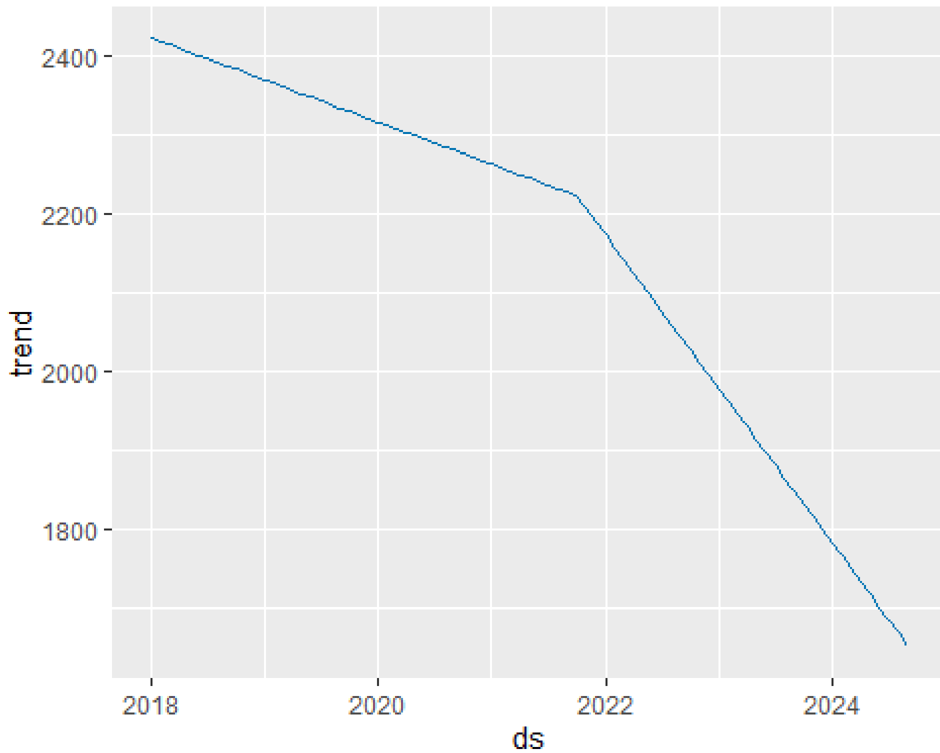
```r
ggsave("images/07_forecast_prophet_plot.jpg", plot = prophet_forecast_plot, w
idth = 8, height = 6)

print(prophet_forecast_plot)
```



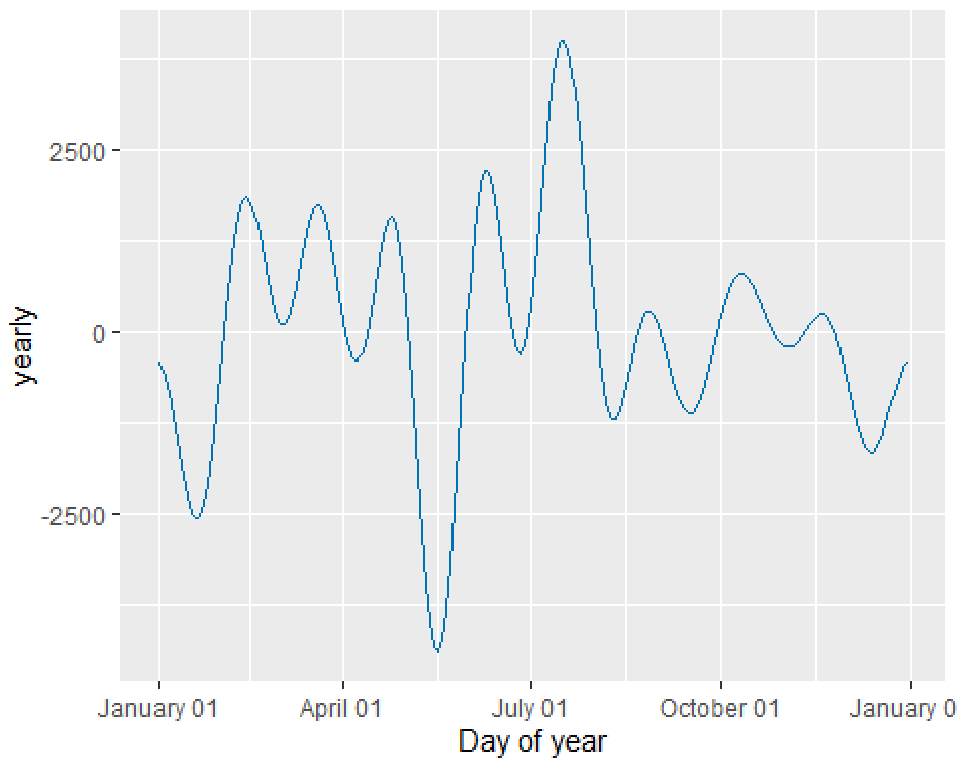

```
#Plot forecast components (trend, seasonality)
print(prophet_plot_components(m, forecast_prophet))  # Generate the base plot
```



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




```
##
## [[2]]
```




```
# Plot forecast components (trend, seasonality)
png("images/07_prophet_components_plot.png", width = 800, height = 600)
prophet_plot_components(m, forecast_prophet)
dev.off()

## png
##   2

# Extract the test data
test_dates <- test_data$DATE # Dates of the test data
test_values <- test_data$Value # Actual values of the test data

# Subset the forecast to match the test dates
forecast_subset <- forecast_prophet %>%
  filter(as.Date(ds) %in% as.Date(test_dates)) %>%
  select(ds, yhat)

# Combine actual and forecasted values
performance_data <- data.frame(
  ds = test_dates,
  actual = test_values,
  predicted = forecast_subset$yhat
)

# Calculate accuracy metrics
rmse <- sqrt(mean((performance_data$actual - performance_data$predicted)^2))
mae <- mean(abs(performance_data$actual - performance_data$predicted))
```

```r
mape <- mean(abs((performance_data$actual - performance_data$predicted) / per
formance_data$actual)) * 100

# Display the results
prophet_accuracy <- data.frame(
  .model = "PROPHET",
  RMSE = rmse,
  MAE = mae,
  MAPE = mape
)

print(knitr::kable(prophet_accuracy, caption = "Prophet Model Accuracy"))

##
##
## Table: Prophet Model Accuracy
##
## |.model  |     RMSE|      MAE|     MAPE|
## |:-------|--------:|--------:|--------:|
## |PROPHET | 283.7677| 193.6933| 10.65383|
```

ACCURACY COMPARISION MODELS PERFORMANCE

Merge and print the accuracy metrics

```r
# Merge the accuracy metrics
joined_accuracy_metrics <- bind_rows(
  arima_accuracy_metrics,
  exponential_smoothing_accuracy_metrics,
  prophet_accuracy,
  nnetar_accuracy
)

# Select key metrics and arrange by RMSE
evaluation_summary <- joined_accuracy_metrics %>%
  select(.model, RMSE, MAE, MAPE) %>%
  arrange(RMSE)

# Display the summary in a table
png("images/08_model_evaluation_summary.png", width = 800, height = 600)

# Open a new plot window
grid::grid.newpage()

# Print the table using knitr::kable
grid::grid.draw(
  gridExtra::tableGrob(
    evaluation_summary,
    theme = gridExtra::ttheme_default(core = list(fg_params = list(cex = 0.8)
)),
```

```
    rows = NULL
  )
)

dev.off()

## png
##   2

print(knitr::kable(evaluation_summary, caption = "Model Evaluation Summary"))

##
##
## Table: Model Evaluation Summary
##
## |.model                      |     RMSE|      MAE|      MAPE|
## |:---------------------------|--------:|--------:|---------:|
## |HoltWinters_Multiplicative  | 127.9611| 103.4666|  5.895931|
## |Holt                        | 151.5543| 113.4003|  6.661605|
## |Drift                       | 158.2233| 116.1680|  6.843003|
## |SES                         | 159.7056| 118.1824|  6.954669|
## |Naive                       | 166.3124| 120.2000|  7.096374|
## |HoltWinters_Additive        | 215.6766| 177.2496|  9.850242|
## |PROPHET                     | 283.7677| 193.6933| 10.653825|
## |ARIMA(Value)                | 307.6023| 281.2290| 15.688127|
## |NNETAR                      | 325.9740| 311.6945| 17.299548|
## |Mean                        | 412.4370| 393.6289| 22.249078|
```