Assignment_2_Data_624

Mubashira Qari

2026-02-14

3.1 Consider the GDP information in global_economy. Plot the GDP per capita for each country over time. Which country has the highest GDP per capita? How has this changed over time?

```
library(fpp3)
```

```
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
```

```
## ── Attaching packages ─────────────────────────────────────── fpp3
1.0.2 ──
```

```
## ✓ tibble      3.2.1     ✓ tsibble     1.1.6
## ✓ dplyr       1.1.4     ✓ tsibbledata 0.4.1
## ✓ tidyr       1.3.1     ✓ feasts      0.4.2
## ✓ lubridate   1.9.4     ✓ fable       0.5.0
## ✓ ggplot2     4.0.0
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
## Warning: package 'tsibble' was built under R version 4.3.3
```

```
## Warning: package 'tsibbledata' was built under R version 4.3.3
```

```
## ── Conflicts ─────────────────────────────────────────────────
fpp3_conflicts ──
## ✖ lubridate::date()    masks base::date()
## ✖ dplyr::filter()      masks stats::filter()
## ✖ tsibble::intersect() masks base::intersect()
## ✖ tsibble::interval()  masks lubridate::interval()
## ✖ dplyr::lag()         masks stats::lag()
## ✖ tsibble::setdiff()   masks base::setdiff()
## ✖ tsibble::union()     masks base::union()
```

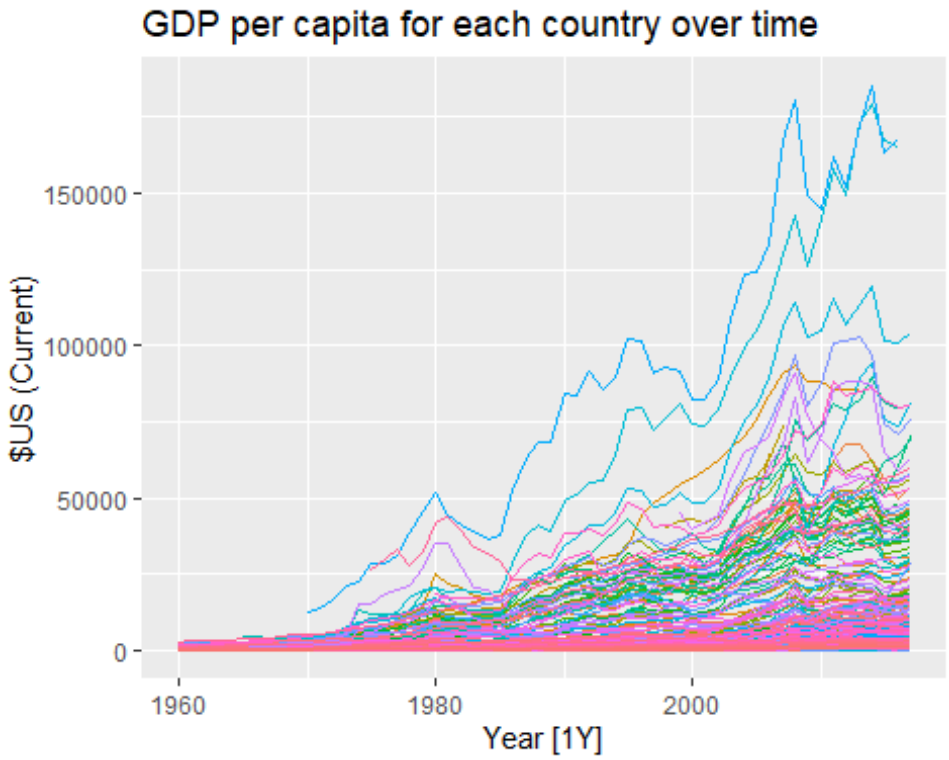
```
# global_economy
```

Calculate GDP per capita and plot for each country

```
library(fpp3)
```

```
global_economy |>
  mutate(GDP_per_capita = GDP / Population) |>
  autoplot(GDP_per_capita, show.legend = FALSE) +
  labs(title = "GDP per capita for each country over time",
       y = "$US (Current)")
```

```
## Warning: Removed 3242 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```




```
library(fpp3)

gdp_data <- global_economy |>
  mutate(GDP_per_capita = GDP / Population) |>
  filter(!is.na(GDP_per_capita))

# gdp_data

gdp_data |>
  group_by(Country) |>
  summarise(max_gdp_pc = max(GDP_per_capita)) |>
  slice_max(max_gdp_pc, n = 5) |>
  arrange(desc(Year))
```

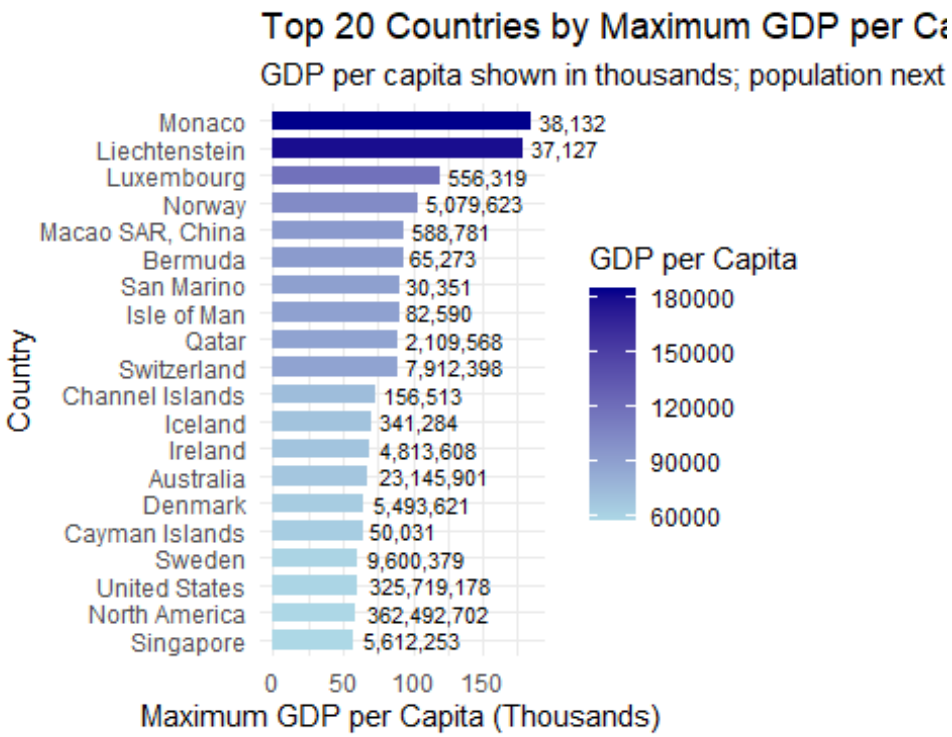
```
## Warning: Current temporal ordering may yield unexpected results.
## ℹ Suggest to sort by `Country`, `Year` first.
```

```
## Current temporal ordering may yield unexpected results.
## ℹ Suggest to sort by `Country`, `Year` first.

## # A tsibble: 5 x 3 [1Y]
## # Key:       Country [2]
##   Country        Year max_gdp_pc
##   <fct>         <dbl>      <dbl>
## 1 Monaco         2014    185153.
## 2 Liechtenstein  2014    179308.
## 3 Liechtenstein  2013    173528.
## 4 Monaco         2013    172589.
## 5 Monaco         2008    180640.
```

Monaco had the highest GDP per capita in the dataset. The maximum value was 185,152.5. This occurred in 2014.

```
max_gdp <- gdp_data |>
  group_by(Country) |>
  summarise(max_gdp_pc = max(GDP_per_capita)) |>
  arrange(desc(max_gdp_pc))

## Warning: Current temporal ordering may yield unexpected results.
## ℹ Suggest to sort by `Country`, `Year` first.

# max_gdp

library(fpp3)

# alculate GDP per capita while maintaining tsibble structure
gdp_data <- global_economy |>
  mutate(GDP_per_capita = GDP / Population) |>
  # Drop rows with missing values to keep the analysis clean
  filter(!is.na(GDP_per_capita))

# Find the year with the maximum GDP per capita for each country
# When using slice_max on a tsibble, it respects the Key (Country)
max_gdp <- gdp_data |>
  group_by(Country) |>
  slice_max(GDP_per_capita, n = 1, with_ties = FALSE) |>
  ungroup() |>
  # Select relevant columns
  select(Country, Year, GDP_per_capita, Population) |>
  # Arrange by the highest values globally
  arrange(desc(GDP_per_capita))

# View the result (this will return a tsibble)
#print(max_gdp)

library(ggplot2)
library(dplyr)
library(scales)
```

```r
# Top 20 countries by max GDP per capita
top20_gdp <- max_gdp |>
  slice_max(GDP_per_capita, n = 20)

# Reduce bar length by dividing GDP per capita by 1000 (show in thousands)
ggplot(top20_gdp, aes(x = reorder(Country, GDP_per_capita), y =
GDP_per_capita/1000, fill = GDP_per_capita)) +
  geom_col(width = 0.7) +  # slightly thinner bars
  geom_text(aes(label = scales::comma(Population)), hjust = -0.1, size = 3) +
  coord_flip(clip = "off") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(
    title = "Top 20 Countries by Maximum GDP per Capita",
    subtitle = "GDP per capita shown in thousands; population next to bars",
    x = "Country",
    y = "Maximum GDP per Capita (Thousands)",
    fill = "GDP per Capita"
  ) +
  theme_minimal() +
  theme(
    plot.margin = margin(10, 50, 10, 10)
  )
```



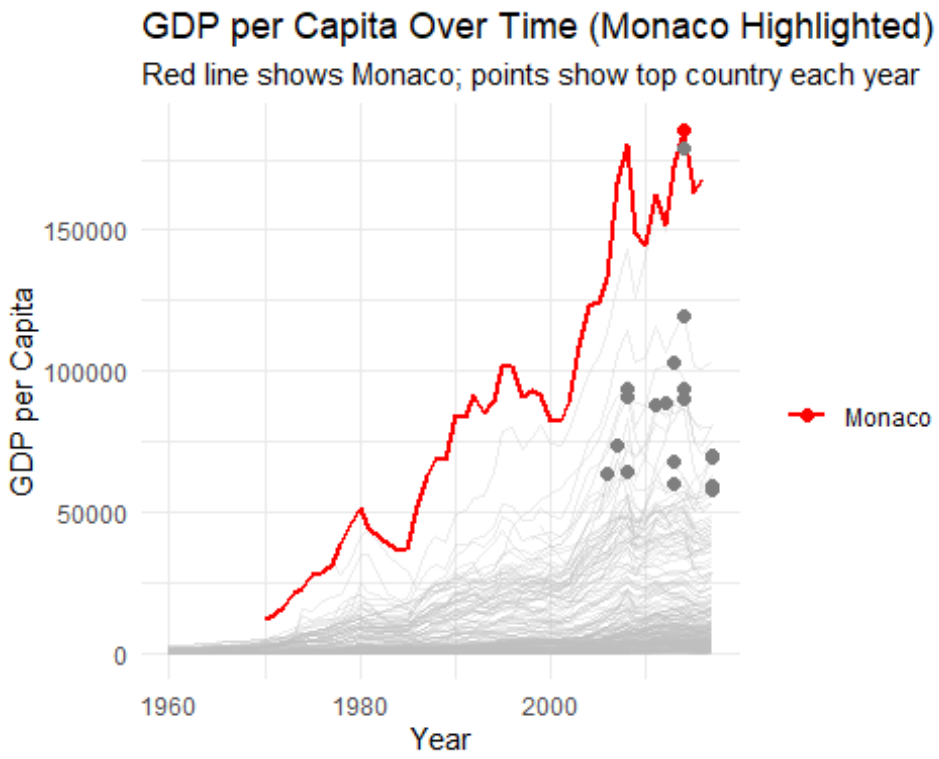

```r
library(dplyr)
monaco_top_years <- top20_gdp |>
```

```r
  filter(Country == "Monaco") |>
  arrange(desc(GDP_per_capita))

monaco_top_years
```

```
## # A tsibble: 1 x 4 [1Y]
## # Key:       Country [1]
##   Country  Year GDP_per_capita Population
##   <fct>   <dbl>          <dbl>      <dbl>
## 1 Monaco   2014        185153.      38132
```

```r
# Separate Monaco
monaco_data <- gdp_data |>
  filter(Country == "Monaco")

# Plot all countries faint + Monaco highlighted
ggplot(gdp_data, aes(x = Year, y = GDP_per_capita, group = Country)) +
  geom_line(color = "gray", alpha = 0.3) +  # all countries faint
  geom_line(data = monaco_data, aes(x = Year, y = GDP_per_capita, color =
"Monaco"), size = 1.0) +
  geom_point(data = top20_gdp, aes(x = Year, y = GDP_per_capita, color =
Country), size = 2) + # top country each year
  scale_color_manual(values = c("Monaco" = "red")) +
  labs(
    title = "GDP per Capita Over Time (Monaco Highlighted)",
    subtitle = "Red line shows Monaco; points show top country each year",
    x = "Year",
    y = "GDP per Capita",
    color = ""
  ) +
  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.
```



How has this changed over time?

After 1980s:

Monaco consistently becomes the top country in GDP per capita.

Reasons:

Tiny population (38–39k people)

High-income economy: banking, tourism, luxury goods, and real estate

Continuous per-person economic growth

Effect:

Because Monaco's GDP is divided among small population, its GDP per capita is extremely high, far surpassing larger countries like the USA or Canada.

This shows that Monaco becoming the richest per person happened gradually over time, not as a one-off event.

```
library(fpp3)

# Calculate GDP per capita and the % Change Rate
gdp_with_growth <- global_economy |>
  mutate(GDP_per_capita = GDP / Population) |>
  group_by(Country) |>
```

```
  # difference() calculates (Year_t - Year_t-1)
  # We divide by the previous year and multiply by 100 for percentage
  mutate(pct_change = (difference(GDP_per_capita) / lag(GDP_per_capita)) *
100) |>
  ungroup()

# Find the top 5 countries by max GDP per capita and show their growth rate
top5_max_gdp <- gdp_with_growth |>
  group_by(Country) |>
  slice_max(GDP_per_capita, n = 1, with_ties = FALSE) |>
  ungroup() |>
  slice_max(GDP_per_capita, n = 10) |>
  # Select relevant columns including the new growth rate
  select(Country, Year, Population, GDP_per_capita, pct_change) |>
  arrange(desc(Year))

# View the result
print(top5_max_gdp)
```

```
## # A tsibble: 10 x 5 [1Y]
## # Key:       Country [10]
##    Country           Year Population GDP_per_capita pct_change
##    <fct>            <dbl>      <dbl>          <dbl>      <dbl>
##  1 Monaco            2014      38132        185153.       7.28
##  2 Liechtenstein     2014      37127        179308.       3.33
##  3 Luxembourg        2014     556319        119225.       4.93
##  4 Macao SAR, China  2014     588781         94004.       5.00
##  5 Isle of Man       2014      82590         89942.       9.21
##  6 Norway            2013    5079623        103059.       1.37
##  7 Qatar             2012    2109568         88565.       3.04
##  8 Switzerland       2011    7912398         88416.      18.5
##  9 Bermuda           2008      65273         93606.       3.03
## 10 San Marino        2008      30351         90683.       9.24
```

3.2 For each of the following series, make a graph of the data. If transforming seems appropriate, do so and describe the effect.

United States GDP from global_economy. Slaughter of Victorian “Bulls, bullocks and steers” in aus_livestock. Victorian Electricity Demand from vic_elec. Gas production from aus_production.

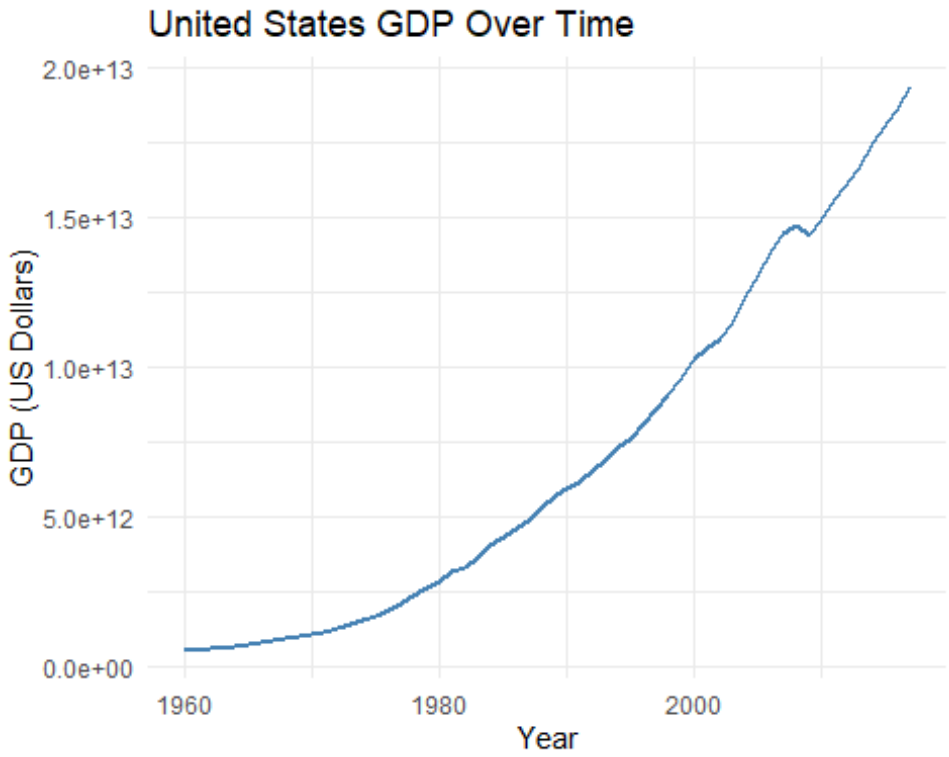
a. United States GDP from global_economy

```
# Load libraries and filter data

library(fpp3)
library(dplyr)
library(ggplot2)

# Filter for United States
```

```r
us_gdp <- global_economy |>
  filter(Country == "United States") |>
  select(Year, GDP)

#us_gdp
```

Plotting raw GDP . Since we want the total economic output of the US, so we focus on raw GDP and not per capita

```r
ggplot(us_gdp, aes(x = Year, y = GDP)) +
  geom_line(color = "steelblue", size = 1) +
  labs(
    title = "United States GDP Over Time",
    x = "Year",
    y = "GDP (US Dollars)"
  ) +
  theme_minimal()
```



Interpretation

GDP is increasing over time, but the growth is exponential.

Early years look “flat” because GDP values in the 1900s are tiny compared to recent years.

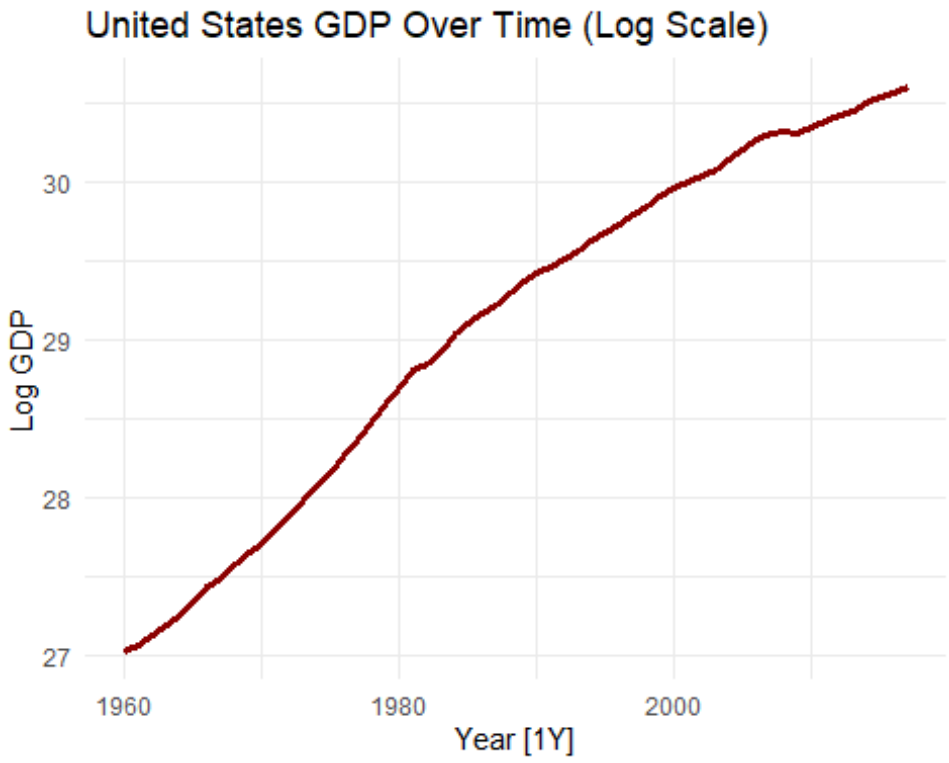
Transform GDP (log transformation)

A log transformation is often appropriate for economic data that grows exponentially.

This helps make the growth more linear

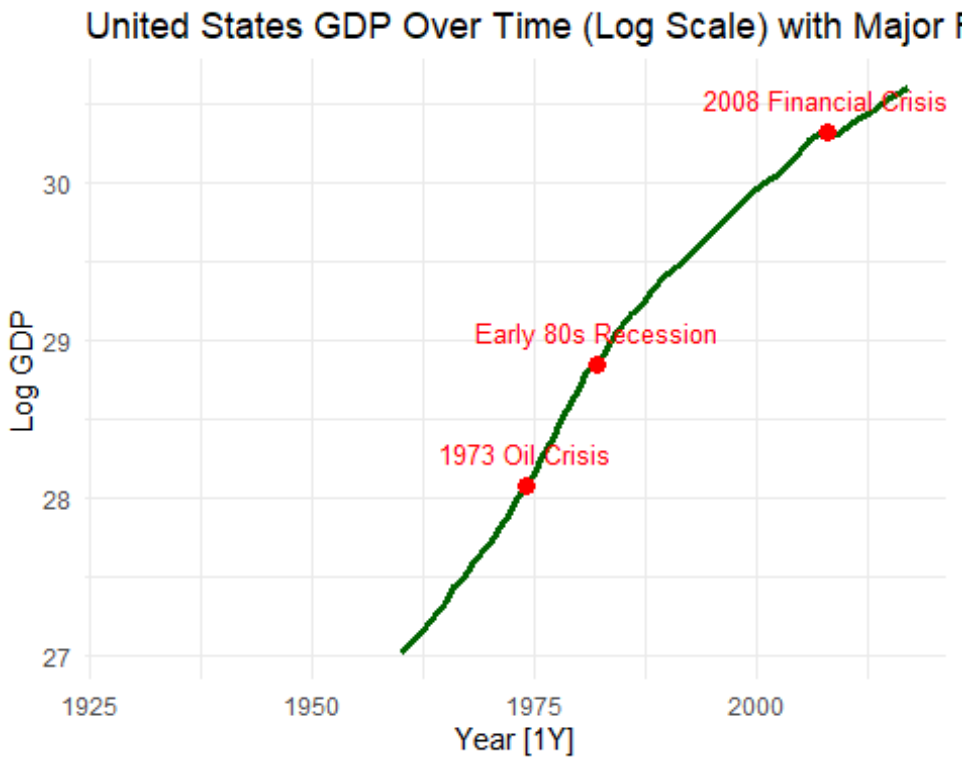
Easier to see relative changes over time

```
# Remove heterogeneity so you fit more simple model

library(fpp3)

us_gdp |>
  autoplot(log(GDP), colour = "darkred" , size = 1.2) +
  labs(
    y = "Log GDP",
    title = "United States GDP Over Time (Log Scale)"
  ) +
  theme_minimal()
```




```
library(fpp3)
library(ggplot2)
library(dplyr)

# US GDP data
us_gdp <- global_economy |>
  filter(Country == "United States") |>
  select(Year, GDP)

# Define some major recession years
recessions <- data.frame(
  Year = c(1929, 1933, 1974, 1982, 2008),
```

```
  Label = c("Great Depression", "End Depression", "1973 Oil Crisis", "Early
80s Recession", "2008 Financial Crisis")
)

# Plot log GDP with thick line, highlight recessions
us_gdp |>
  autoplot(log(GDP), colour = "darkgreen", size = 1.2) +
  geom_point(data = recessions, aes(x = Year, y = log(us_gdp$GDP[match(Year,
us_gdp$Year)])),
             colour = "red", size = 3) +
  geom_text(data = recessions, aes(x = Year, y = log(us_gdp$GDP[match(Year,
us_gdp$Year)]), label = Label),
            vjust = -1, hjust = 0.5, size = 3.5, colour = "red") +
  labs(
    y = "Log GDP",
    title = "United States GDP Over Time (Log Scale) with Major Recessions
Highlighted"
  ) +
  theme_minimal()

## Warning: Removed 2 rows containing missing values or values outside the
scale range
## (`geom_point()`).

## Warning: Removed 2 rows containing missing values or values outside the
scale range
## (`geom_text()`).
```



b. Slaughter of Victorian “Bulls, bullocks and steers” in aus_livestock.

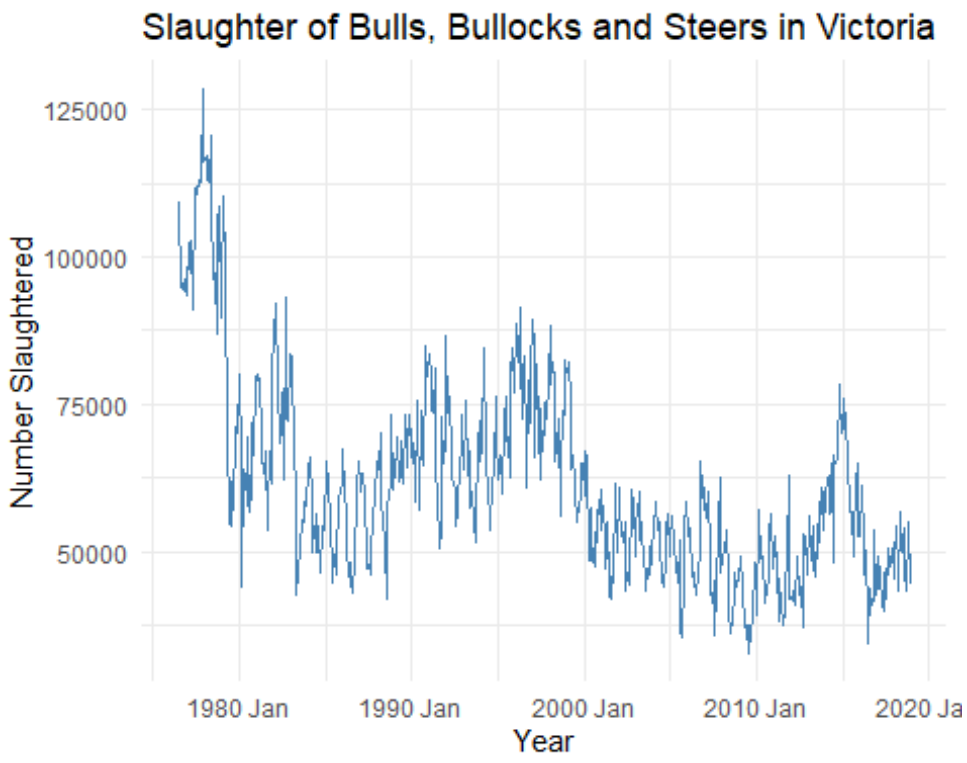
Loading data and filtering for Victoria

```
library(fpp3)
library(dplyr)
library(ggplot2)

# view(aus_livestock)

library(fpp3)
library(dplyr)

# Filter for Victoria and the "Bulls, bullocks and steers" series
vic_bulls <- aus_livestock |>
  filter(State == "Victoria",
         Animal == "Bulls, bullocks and steers") |>
  mutate(Slaughter_Count = Count) |>
  select(Month, Slaughter_Count)

# vic_bulls
```

Before we apply any decomposition it is good to see the data.

```
vic_bulls |>
  autoplot(Slaughter_Count, colour = "steelblue", size = 0.7) +
  labs(
    title = "Slaughter of Bulls, Bullocks and Steers in Victoria",
```

```
    y = "Number Slaughtered",
    x = "Year"
  ) +
  theme_minimal()
```



Analysis of Time Series Components:

Trend: A clear long-term decrease is visible as the data transitions from higher volumes in the late 1970s to a lower baseline by 2010.

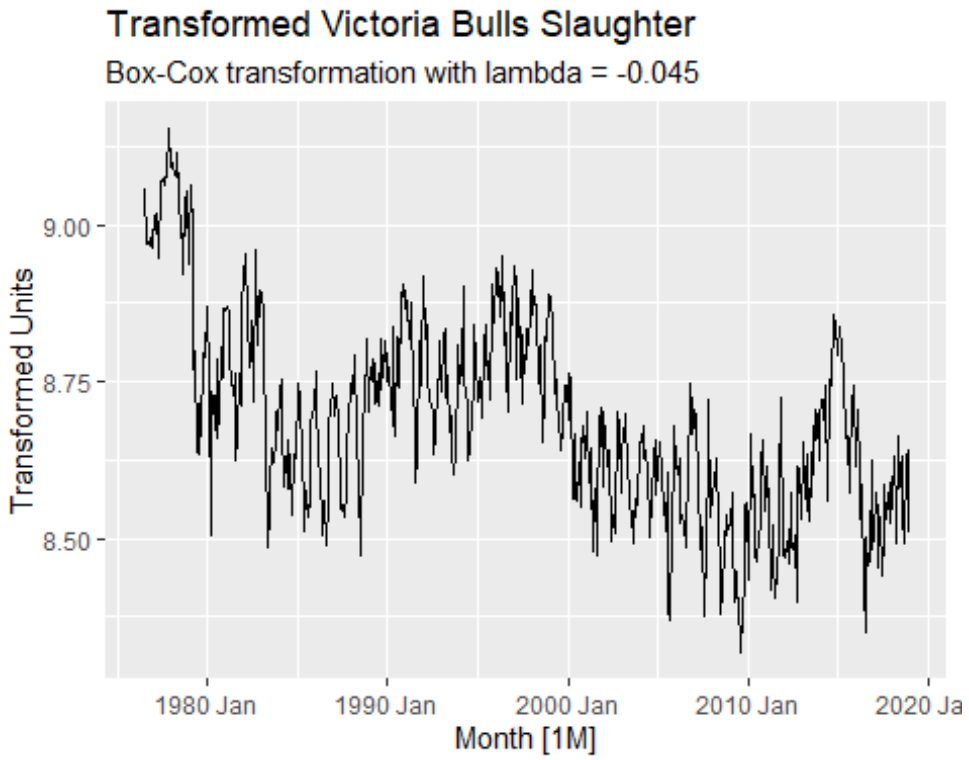
Seasonality: The consistent, jagged teeth appearing at annual intervals represent fluctuations tied to the calendar year, such as weather or other patterns.

Cycle: The broader, multi-year waves—specifically the peaks around 1998 and 2015—indicate cyclical behavior often associated with economic or livestock-specific cycles that do not follow a fixed calendar.

Determining the Need for Transformation A transformation (like Box-Cox) is appropriate when the variance of the series increases or decreases in proportion to its level.In this specific series, the seasonal fluctuations appear relatively stable in magnitude throughout the timeline, despite the downward trend. However, to confirm if a transformation is statistically beneficial, we can use the Guerrero method to find the optimal lambda.

```
# Calculate optimal lambda
optimal_lambda <- vic_bulls |>
  features(Slaughter_Count, features = guerrero) |>
```

```
  pull(lambda_guerrero)
optimal_lambda

## [1] -0.04461887

# Apply transformation
vic_bulls |>
  autoplot(box_cox(Slaughter_Count, optimal_lambda)) +
  labs(title = "Transformed Victoria Bulls Slaughter",
       subtitle = paste("Box-Cox transformation with lambda =",
round(optimal_lambda, 3)),
       y = "Transformed Units")
```



The specific effects of applying this lambda to your vic_bulls data include:

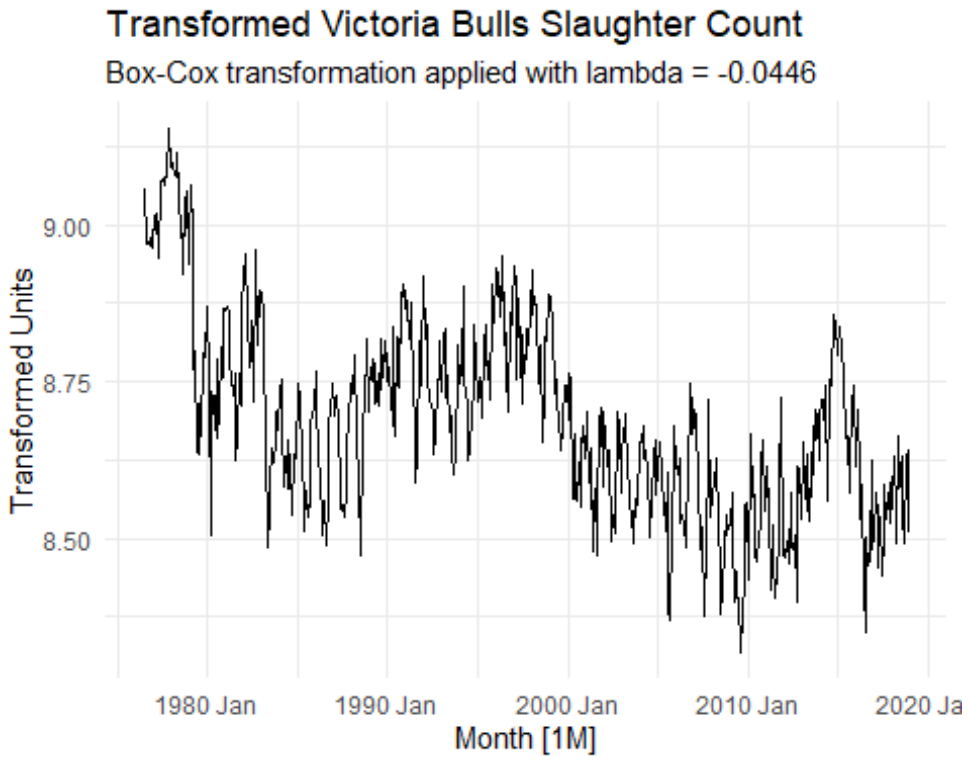
Variance Stabilization: The vertical magnitude of the seasonal fluctuations is equalized across the timeline. This ensures that the seasonal “swings” in the high-volume 1970s are mathematically comparable to the swings in the lower-volume 2010s.

Linearization of Trends: By compressing the scale, the transformation can help linearize growth or decline patterns, making them easier for models like STL or ETS to interpret.

Outlier Dampening: Any extreme, one-off spikes in slaughter counts are pulled closer to the mean level, preventing individual months from having a disproportionate impact on the long-term trend estimate.

```
# Apply the specific lambda calculated via Guerrero
final_lambda <- -0.04461887
```

```
vic_bulls |>
  autoplot(box_cox(Slaughter_Count, final_lambda)) +
  labs(
    title = "Transformed Victoria Bulls Slaughter Count",
    subtitle = paste("Box-Cox transformation applied with lambda =",
round(final_lambda, 4)),
    y = "Transformed Units"
  ) +
  theme_minimal()
```



By stabilizing the variance first, we ensure that a subsequent STL decomposition will produce a reliable and clean “remainder” component, which is essential for accurate forecasting.

Victorian Electricity Demand from vic_elec. dataset

```
library(fpp3)
library(dplyr)

# Inspect the vic_elec dataset
# head(vic_elec)
```

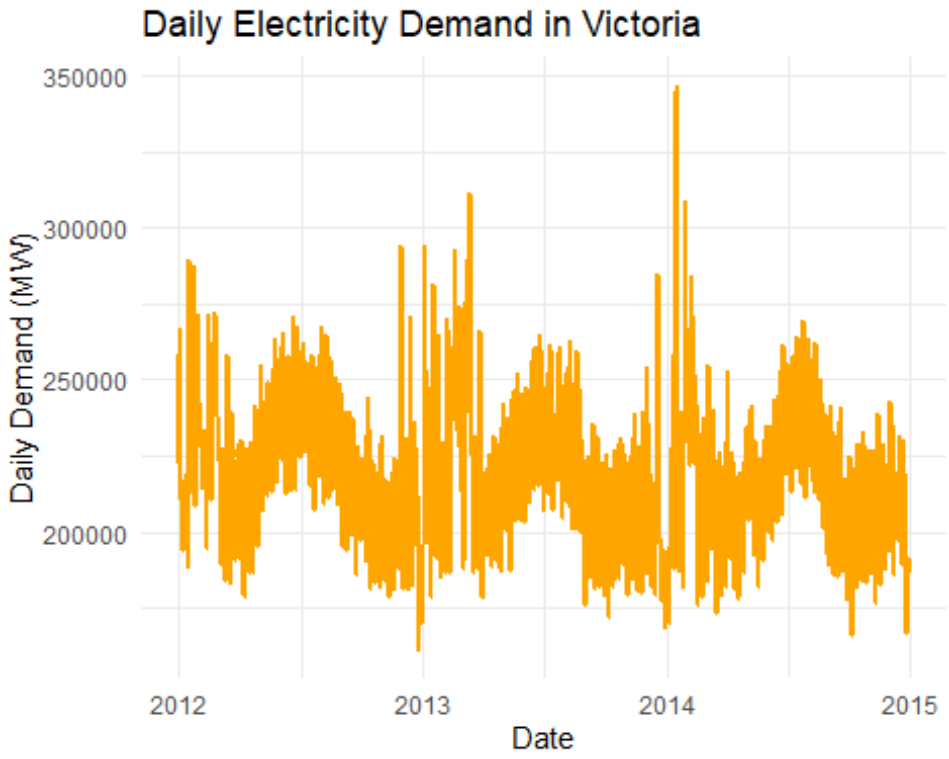
Prepare the data

To focus on daily electricity demand, we aggregate by day which is easier to visualize than half-hourly data.

```r
vic_elec_daily <- vic_elec |>
  index_by(Date = date(Time)) |>
  summarise(Daily_Demand = sum(Demand, na.rm = TRUE))

# vic_elec_daily
```

Autoplot the raw series

```r
vic_elec_daily |>
  autoplot(Daily_Demand, colour = "orange", size = 1.0) +
  labs(
    title = "Daily Electricity Demand in Victoria",
    y = "Daily Demand (MW)",
    x = "Date"
  ) +
  theme_minimal()
```



Calculating the Optimal Lambda (lambda) Using the Guerrero method on the aggregated Daily_Demand column ensures the transformation is mathematically optimized to stabilize the variance across the entire 1,096-row series.

```r
library(fpp3)

# Calculate the optimal lambda for Daily_Demand
daily_lambda <- vic_elec_daily |>
  features(Daily_Demand, features = guerrero) |>
  pull(lambda_guerrero)
```

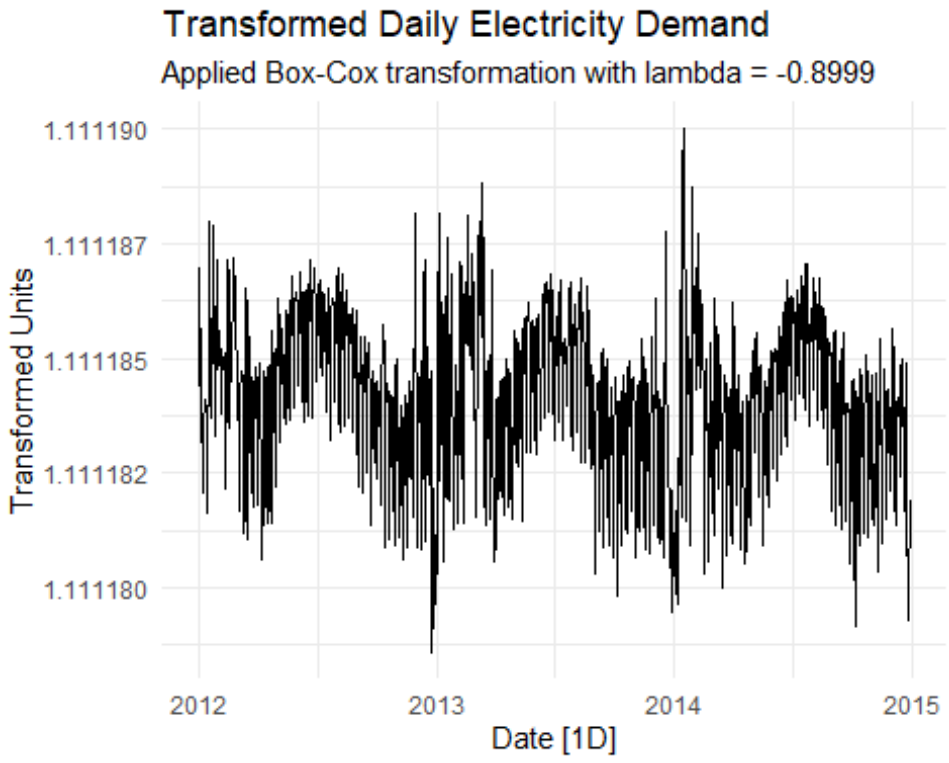
```
# Display the result
daily_lambda
```

```
## [1] -0.8999268
```

A value of -0.9 suggests that the variance in your Daily_Demand increases extremely sharply as the level of demand rises.

Apply the Transformation

```
library(fpp3)

# Applying the specific lambda value discovered
lambda_val <- -0.8999268

vic_elec_daily |>
  autoplot(box_cox(Daily_Demand, lambda_val)) +
  labs(
    title = "Transformed Daily Electricity Demand",
    subtitle = paste("Applied Box-Cox transformation with lambda =",
round(lambda_val, 4)),
    y = "Transformed Units"
  ) +
  theme_minimal()
```



If we compare the raw data to the transformed data, we will notice the stabilization effect:

In the original orange chart, the massive spike in early 2014 towers over everything else. In the black transformed chart, that same spike is “pulled back” and is much closer in height to the other seasonal peaks.

Look at the troughs (the low points). In the raw data, the gaps between the high and low points vary significantly throughout the year. In the transformed version, these vertical swings become more uniform in length.

The goal of this transformation wasn’t to change the shape of the series, but to prepare it for STL Decomposition.

Inspect the data

```
library(fpp3)
library(dplyr)

head(aus_production)
```

```
## # A tsibble: 6 x 7 [1Q]
##   Quarter  Beer Tobacco Bricks Cement Electricity   Gas
##     <qtr> <dbl>   <dbl>  <dbl>  <dbl>       <dbl> <dbl>
## 1 1956 Q1   284    5225    189    465        3923     5
## 2 1956 Q2   213    5178    204    532        4436     6
## 3 1956 Q3   227    5297    208    561        4806     7
## 4 1956 Q4   308    5681    197    570        4418     6
## 5 1957 Q1   262    5577    187    529        4339     5
## 6 1957 Q2   228    5651    214    604        4811     7
```

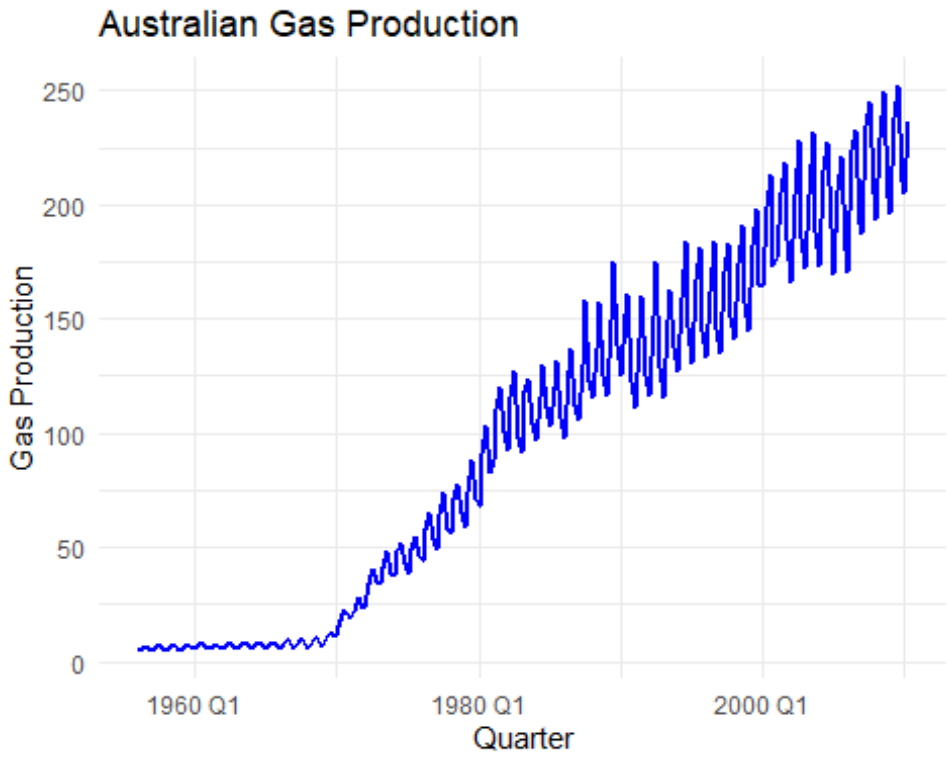
```
gas_production <- aus_production |>
  select(Quarter, Gas)
gas_production
```

```
## # A tsibble: 218 x 2 [1Q]
##    Quarter   Gas
##      <qtr> <dbl>
##  1 1956 Q1     5
##  2 1956 Q2     6
##  3 1956 Q3     7
##  4 1956 Q4     6
##  5 1957 Q1     5
##  6 1957 Q2     7
##  7 1957 Q3     7
##  8 1957 Q4     6
##  9 1958 Q1     5
## 10 1958 Q2     7
## # ℹ 208 more rows
```

Plot the raw data series

```
gas_production |>
  autoplot(Gas, colour = "blue", size = 1.0) +
```

```
  labs(
    title = "Australian Gas Production",
    y = "Gas Production",
    x = "Quarter"
  ) +
  theme_minimal()
```



For the Australian Gas Production series, the transformation is highly appropriate because the plot clearly shows increasing seasonal variation as the production level rises.

Analysis of the Graph

Trend: There is a strong, long-term upward trend starting around 1970.

Seasonality: A very strong quarterly seasonal pattern is evident.

Increasing Variance: Notice that the “height” of the seasonal zig-zags in the 2000s is much larger than it was in the 1970s. This is known as multiplicative seasonality and indicates that a transformation is needed to stabilize the variance.

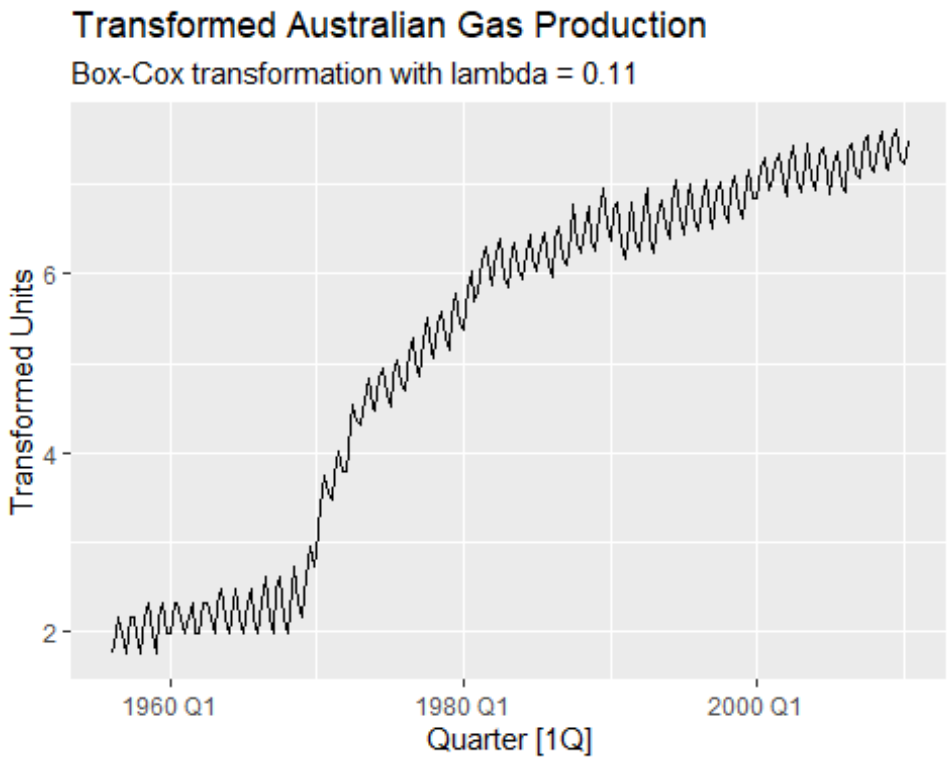
Description of the Effect Applying this transformation to the gas data would have the following effects:

Variance Stabilization: The vertical magnitude of the seasonal fluctuations will be equalized. The spikes at the end of the series will appear the same size as those at the beginning.

Linearization: The exponential-looking growth curve will be dampened into a more linear trend, making it easier for models like STL or ETS to handle.

Balanced Residuals: It ensures that when you eventually forecast, the prediction intervals aren't inappropriately small for high-production periods.

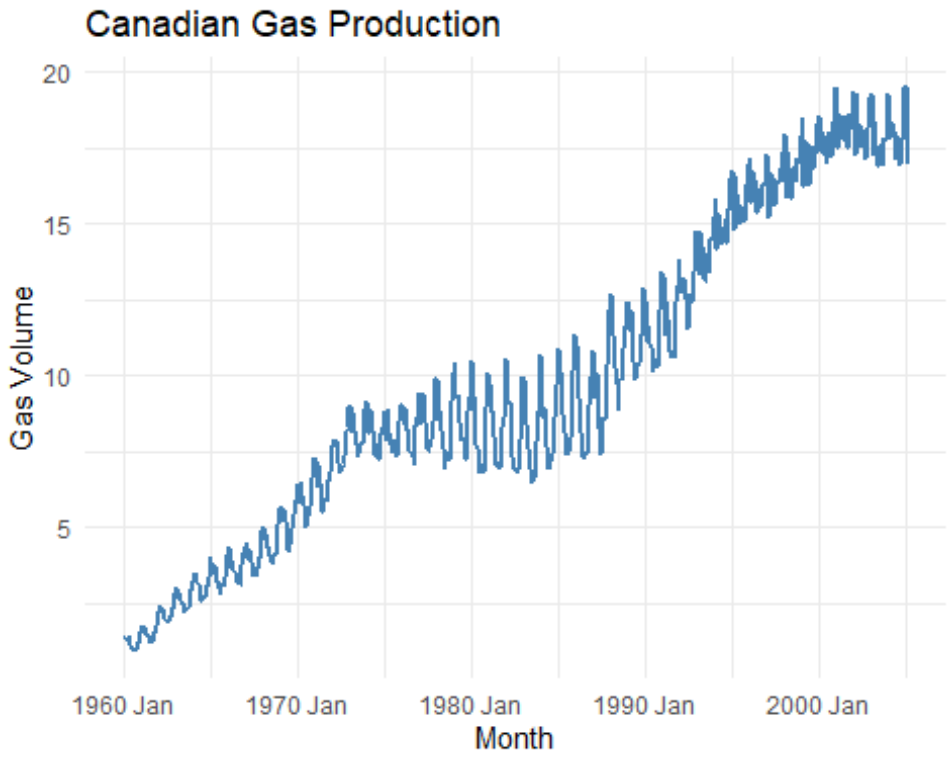
```
# Calculate optimal lambda for Gas
gas_lambda <- gas_production |>
  features(Gas, features = guerrero) |>
  pull(lambda_guerrero)

# Plot the transformed series
gas_production |>
  autoplot(box_cox(Gas, gas_lambda)) +
  labs(title = "Transformed Australian Gas Production",
       subtitle = paste("Box-Cox transformation with lambda =",
round(gas_lambda, 3)),
       y = "Transformed Units")
```



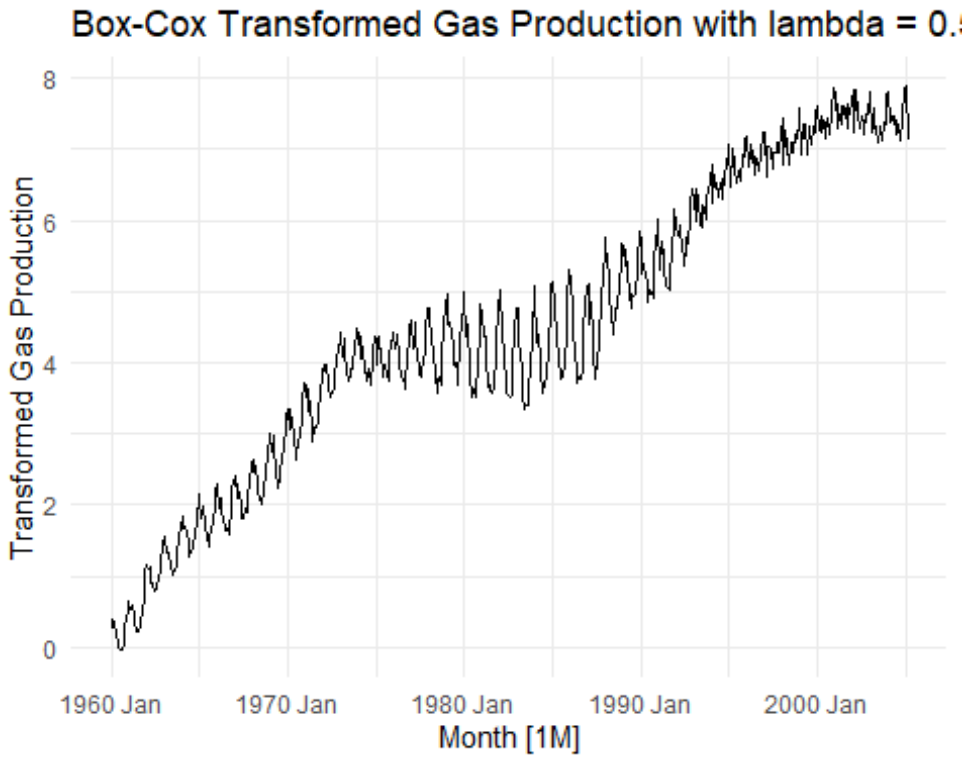
3.3 Why is a Box-Cox transformation unhelpful for the canadian_gas data?

```
# canadian_gas

library(fpp3)

canadian_gas |>
  autoplot(Volume, colour = "steelblue", size = 1.0) +
```

```
  labs(
    title = "Canadian Gas Production",
    y = "Gas Volume",
    x = "Month"
  ) +
  theme_minimal()
```




```
library(fpp3)
library(dplyr)

# Step 1: Calculate optimal Box-Cox lambda for Gas
lambda_gas <- canadian_gas |>
  features(Volume, features = guerrero) |>
  pull(lambda_guerrero)

# Step 2: Apply Box-Cox transformation and plot using the word "lambda"
canadian_gas |>
  autoplot(box_cox(Volume, lambda_gas)) +
  labs(
    y = "Transformed Gas Production",
    title = paste("Box-Cox Transformed Gas Production with lambda =",
                  round(lambda_gas, 2))
  ) +
  theme_minimal()
```



Explanation:

A Box-Cox transformation is unhelpful for the canadian_gas data because the variance in the series does not change proportionally with the level of the data.

Specifically, the seasonal fluctuations do not grow or shrink in a consistent way as the trend increases.

Why the Transformation Fails for This Series

Non-Proportional Variance: In the initial “Canadian Gas Production” chart, notice that between 1975 and 1990, the seasonal swings are quite large, but as the production level continues to rise after 1990, the swings actually become smaller and more compressed.

Contradicting the Transformation’s Purpose: A Box-Cox transformation (like your result with lambda = 0.58) is designed to fix cases where seasonal variation increases as the level increases (multiplicative seasonality). Because the data shows large swings in the middle and smaller swings at the end, the transformation cannot “even them out” across the entire timeline.

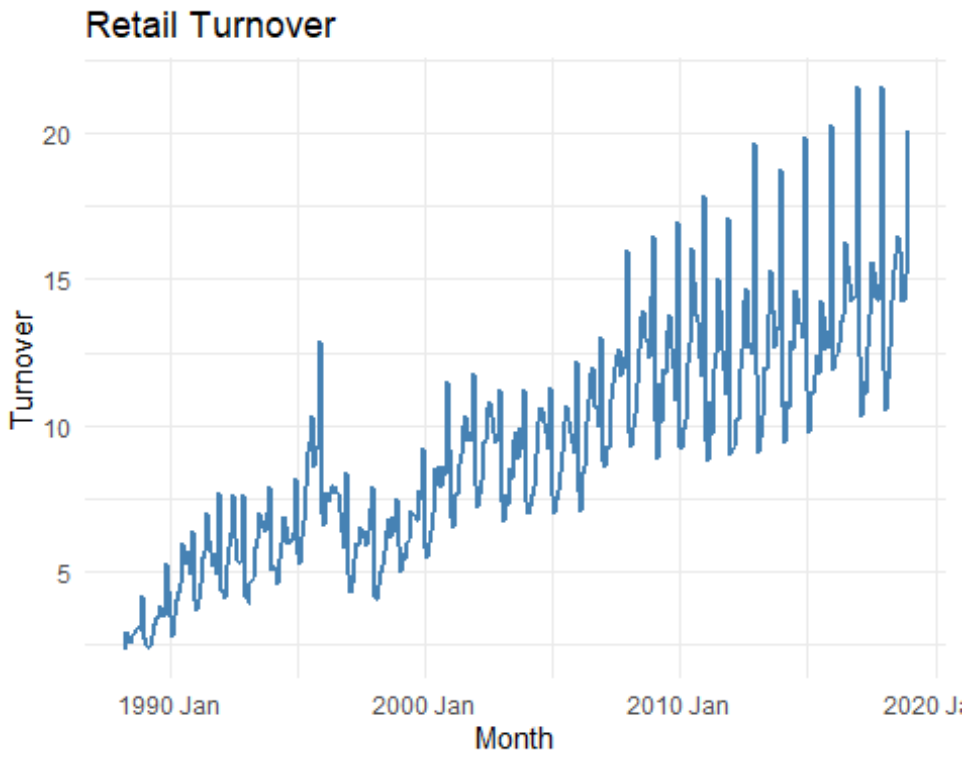
Resulting Visual Distortion: If we look at the “Box-Cox Transformed” chart, we can see the transformation didn’t stabilize the variance; it simply compressed the entire y-axis without making the seasonal zig-zags look uniform.

3.4 What Box-Cox transformation would you select for your retail data (from Exercise 7 in Section 2.10)?

```
set.seed(12345678)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))

# myseries
```

Plot the rew series

```
library(fpp3)

myseries |>
  autoplot(Turnover, colour = "steelblue", size = 1.0) +
  labs(
    title = "Retail Turnover",
    y = "Turnover",
    x = "Month"
  ) +
  theme_minimal()
```



Retail turnover shows:

Strong upward trend

Seasonal fluctuations that increase as the level increases

Variance growing over time

This suggests multiplicative seasonality, meaning a transformation is likely needed.

Estimate the Box-Cox λ (Guerrero method)

Lambda ≈ 1 → No transformation

Lambda ≈ 0 → Log transformation

0 < Lambda < 1 → Power transformation

compute λ using Guerrero's method:

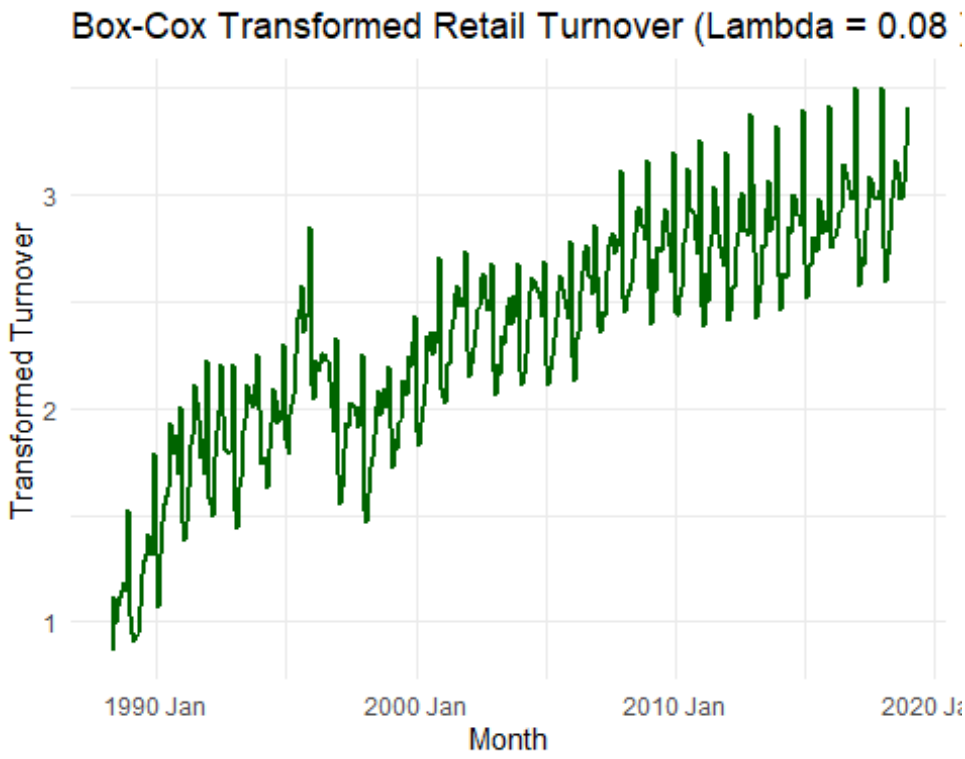
```
lambda_retail <- myseries |>
  features(Turnover, features = guerrero) |>
  pull(lambda_guerrero)

lambda_retail

## [1] 0.08303631
```

Using Guerrero's method, the estimated Box-Cox parameter is Lambda = 0.083.

Since this value is very close to 0, the appropriate transformation is approximately a log transformation.

Applying a log transformation helps stabilize the increasing variance and seasonal fluctuations in the retail turnover series.

```
myseries |>
  autoplot(box_cox(Turnover, lambda_retail),
           colour = "darkgreen", size = 1.0) +
  labs(
    title = paste("Box-Cox Transformed Retail Turnover (Lambda =",
                  round(lambda_retail,2), ")"),
    y = "Transformed Turnover",
    x = "Month"
  ) +
  theme_minimal()
```



3.5 For the following series, find an appropriate Box-Cox transformation in order to stabilise the variance.

Tobacco from aus_production, Economy class passengers between Melbourne and Sydney from ansett, and Pedestrian counts at Southern Cross Station from pedestrian.

a. Tobacco from aus_production

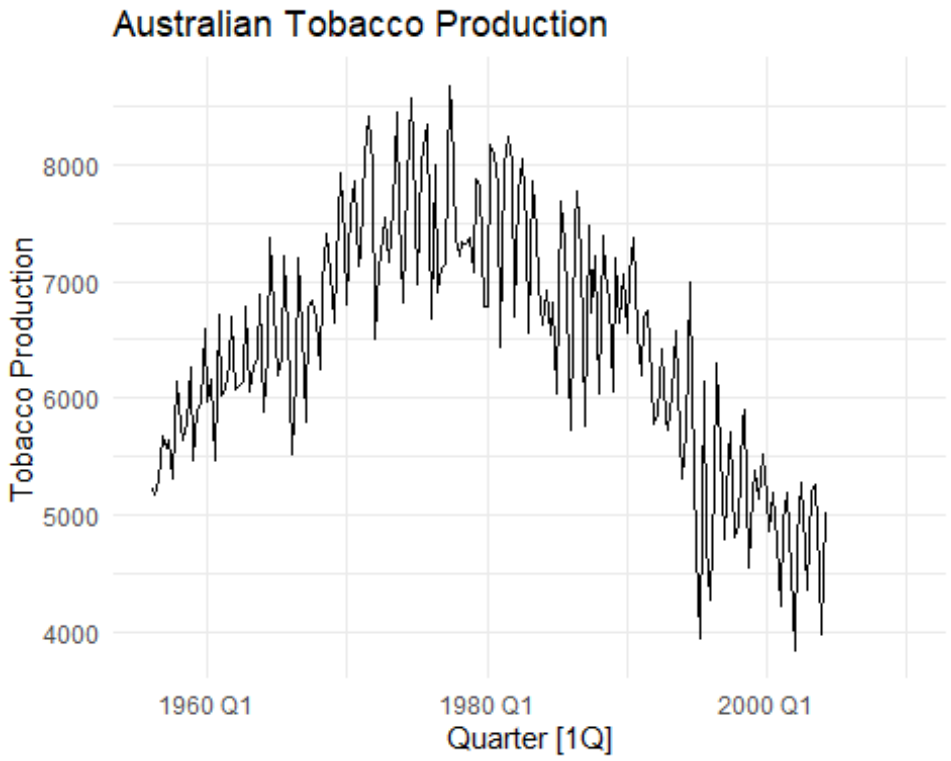
Inspect Data

```
library(fpp3)
library(dplyr)

# head(aus_production)
```

Plot the original data

```
library(fpp3)

aus_production |>
  autoplot(Tobacco) +
  labs(
    title = "Australian Tobacco Production",
    y = "Tobacco Production"
  ) +
  theme_minimal()
```

```
## Warning: Removed 24 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



We look for the following:

Does variability increase over time?

Are seasonal swings getting larger?

Does variance grow with the level?

From your Tobacco production graph, we observe

There is strong seasonality with a regular quarterly peaks.

The variance does NOT clearly increase with the level.

In fact, after 1980 the series declines.

The seasonal swings look roughly similar in size across time.

A Box-Cox transformation is used to:

Stabilise increasing variance

Convert multiplicative seasonality into additive

But in this plot:

Seasonal amplitude looks fairly constant.

Variance does not clearly grow with the level.

So a transformation may not be necessary.

Confirm with Guerrero's Method

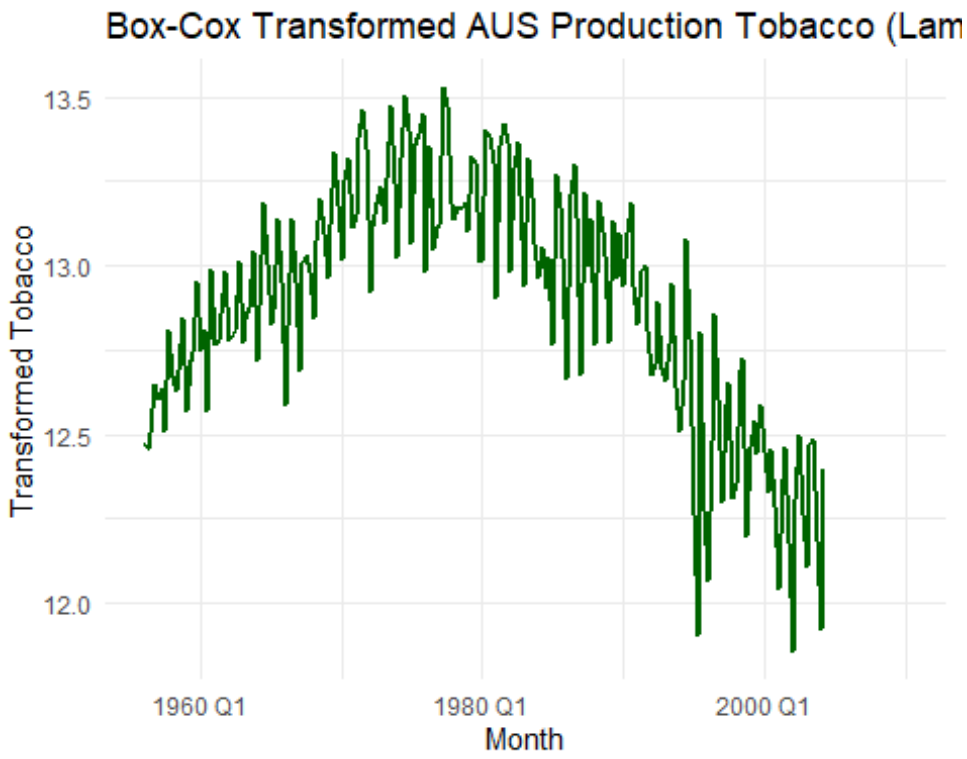
```
lambda_tobacco <- aus_production |>
  features(Tobacco, features = guerrero) |>
  pull(lambda_guerrero)

lambda_tobacco
```

```
## [1] 0.9264636
```

Since lambda is 0.926, which is very close to 1, it suggests that data doesn't need a strong transformation.

```
aus_production |>
  autoplot(box_cox(Tobacco, lambda_retail),
           colour = "darkgreen", size = 1.0) +
  labs(
    title = paste("Box-Cox Transformed AUS Production Tobacco (Lambda =",
                  round(lambda_retail,2), ")"),
    y = "Transformed Tobacco",
    x = "Month"
  ) +
  theme_minimal()
```

```
## Warning: Removed 24 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```

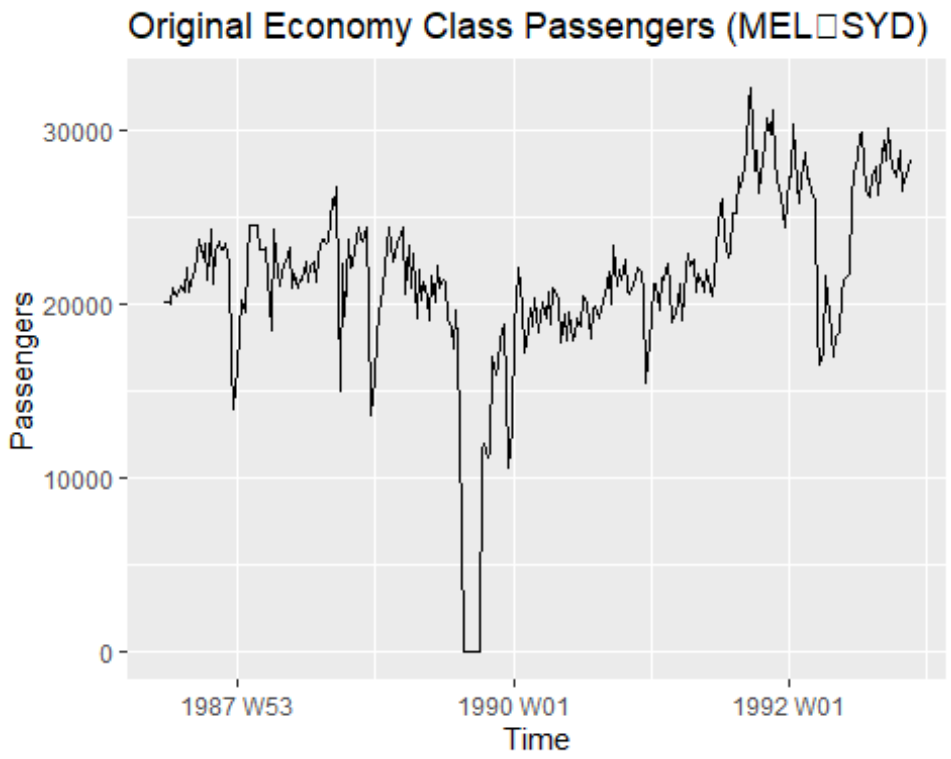


b. Box-Cox Transformation for the Ansett Economy Passengers Series

```
library(fpp3)

# Filter the series for Economy class passengers on the MEL-SYD route
economy <- ansett |>
  filter(Airports == "MEL-SYD", Class == "Economy")

# economy
```

Plot the Original Series

```
autoplot(economy, Passengers) +
  labs(title = "Original Economy Class Passengers (MEL-SYD)",
       y = "Passengers", x = "Time")
```

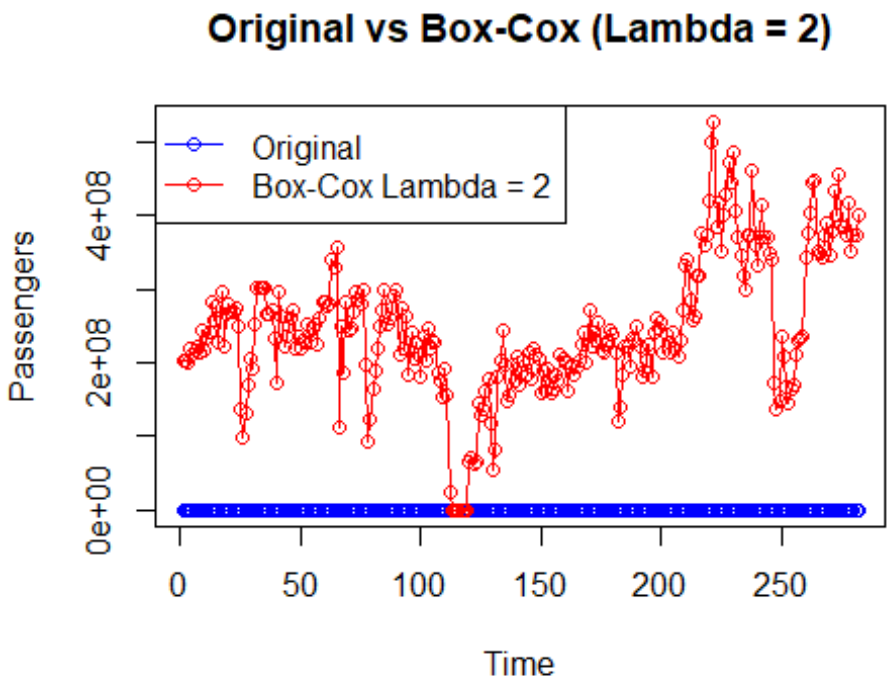


We notices that the variance isn't constant over time, partly due to strikes or structural changes in the data which makes a transformation potentially helpful

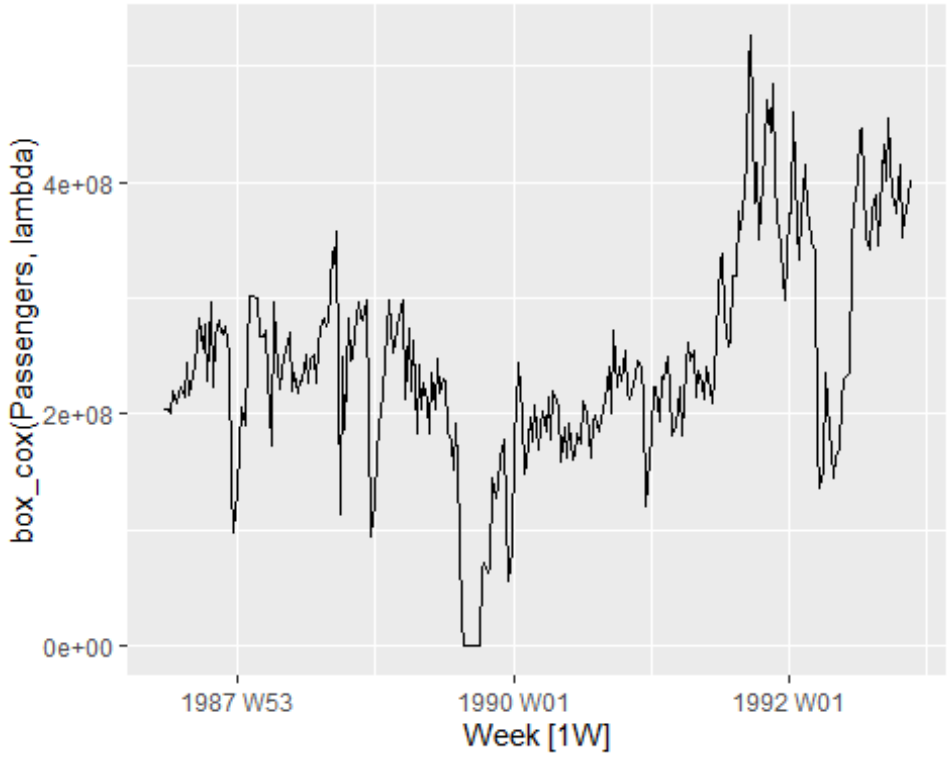
Estimate the Best Lambda Using Guerrero's Method

```
lambda_economy <- economy |>
  features(Passengers, features = guerrero) |>
  pull(lambda_guerrero)

lambda_economy

## [1] 1.999927

lambda <- 1.999927

economy$Passengers_BoxCox <- if(lambda == 0){
  log(econ$Passengers)
} else {
  (economy$Passengers^lambda - 1)/lambda
}

# Plot original vs transformed
plot(economy$Passengers, type="o", col="blue",
ylim=range(c(economy$Passengers, economy$Passengers_BoxCox)),
     xlab="Time", ylab="Passengers", main="Original vs Box-Cox (Lambda = 2)")
lines(economy$Passengers_BoxCox, type="o", col="red")
```

```
legend("topleft", legend=c("Original","Box-Cox Lambda = 2"),
col=c("blue","red"), lty=1, pch=1)
```




```
economy |>
  autoplot(box_cox(Passengers, lambda))
```

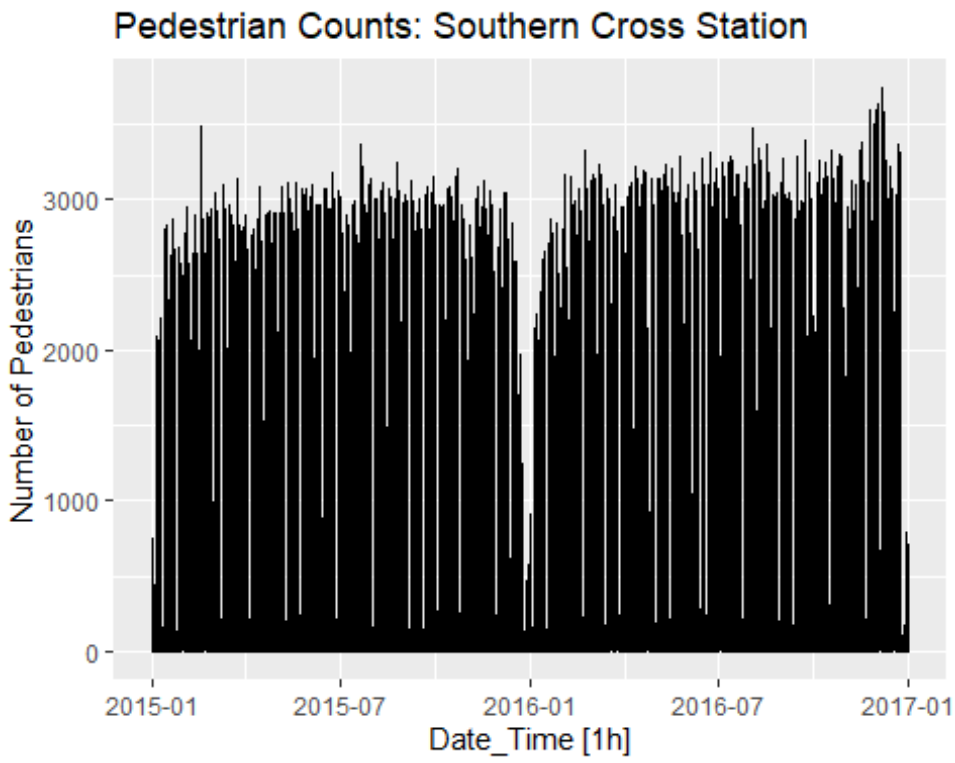


c. Pedestrian counts at Southern Cross Station from pedestrian.

```
library(fpp3)

# Filter for Southern Cross Station
southern_cross <- pedestrian |>
  filter(Sensor == "Southern Cross Station")

# southern_cross
```

Visualize the Raw Data

```
southern_cross |>
  autoplot(Count) +
  labs(title = "Pedestrian Counts: Southern Cross Station",
       y = "Number of Pedestrians")
```



Determine the Optimal Lambda

```
# Calculate the optimal lambda
lambda <- southern_cross |>
  features(Count, features = guerrero) |>
  pull(lambda_guerrero)

# Display the value
lambda
```

```
## [1] -0.2501616
```

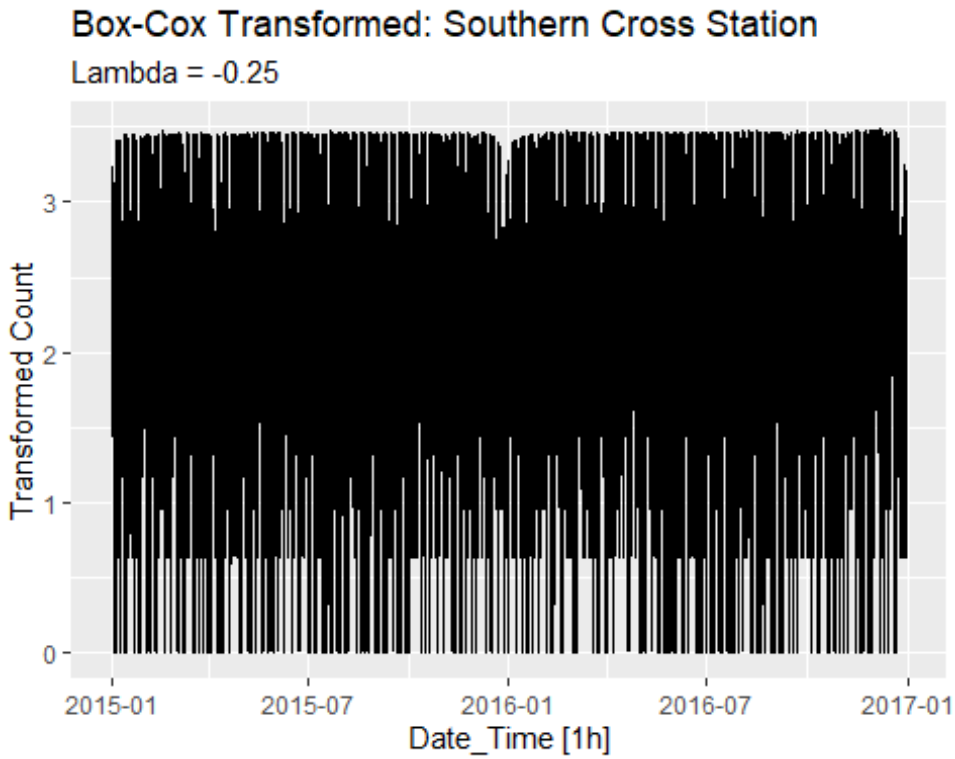
When lambda is negative, the transformation effectively flips the order of the data and larger values become smaller.

However, the box_cox() function in the fpp3 package automatically handles this by scaling the results so that the series maintains its original direction increasing values in the raw data remain increasing in the transformed data.

Applying the Transformation

```
# Apply the specific lambda found
southern_cross |>
  autoplot(box_cox(Count, lambda)) +
  labs(
    title = "Box-Cox Transformed: Southern Cross Station",
    subtitle = "Lambda = -0.25",
```

```
    y = "Transformed Count"
  )
```



Southern Cross Station usually shows strong daily and weekly patterns with high peaks during commute hours.

The negative lambda suggests that the variance which is the wiggle in the data was increasing very aggressively as the volume of pedestrians increased.

This transformation pulls those massive peaks down significantly to make the seasonal patterns more uniform.

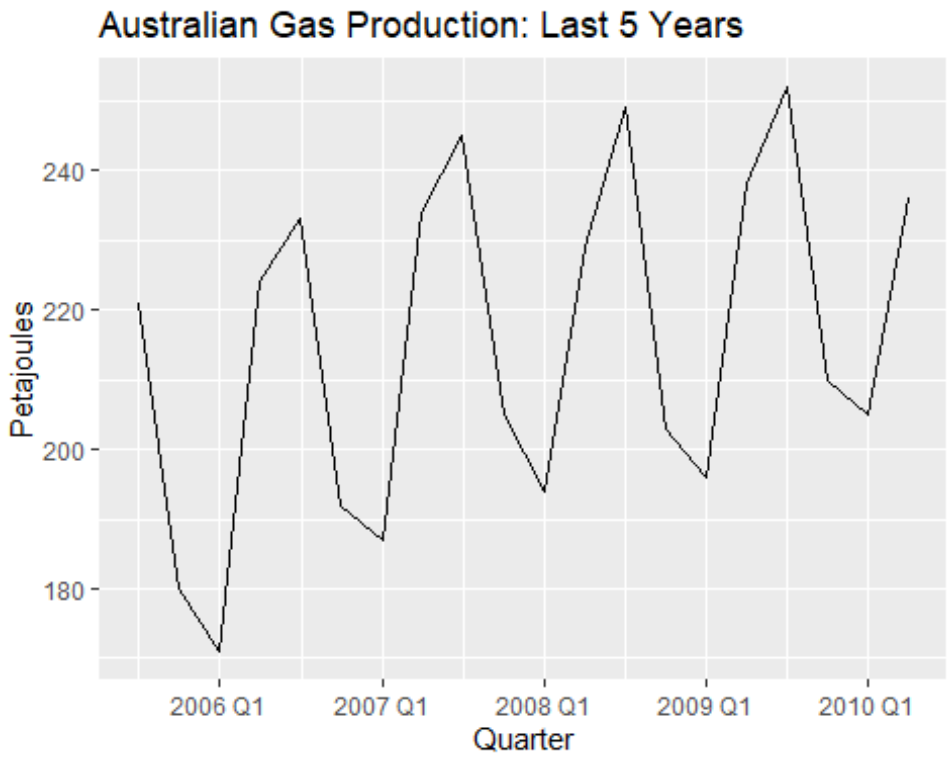
3.7 Consider the last five years of the Gas data from aus_production.

gas <- tail(aus_production, 5*4) |> select(Gas)

a. Plot the time series. Can you identify seasonal fluctuations and/or a trend-cycle?

```
# Load the necessary library
library(fpp3)

# Extract the last 5 years (20 quarters) of Gas data
gas <- tail(aus_production, 5 * 4) |>
  select(Gas)

# Plot the time series
```

```r
gas |>
  autoplot(Gas) +
  labs(title = "Australian Gas Production: Last 5 Years",
       y = "Petajoules",
       x = "Quarter")
```



Analysis of the Series

Based on the resulting plot, we can identify the following components:

Trend-Cycle:

There is a clear upward trend over the five-year period. The overall level of gas production increases from the beginning of the series to the end.

Because the data covers only five years, it is difficult to identify a distinct cycle which usually spans several years and is related to economic conditions, but the steady increase is the dominant “trend” component.

Seasonal Fluctuations:

There is a very strong and regular seasonal pattern.

Peaks: Gas production consistently peaks in the middle quarters (Q2 and Q3). This corresponds to the Australian winter when the demand for heating is at its highest.

Troughs: The lowest production levels occur in Quarter 1 (Summer), followed by a slight rise in Quarter 4.

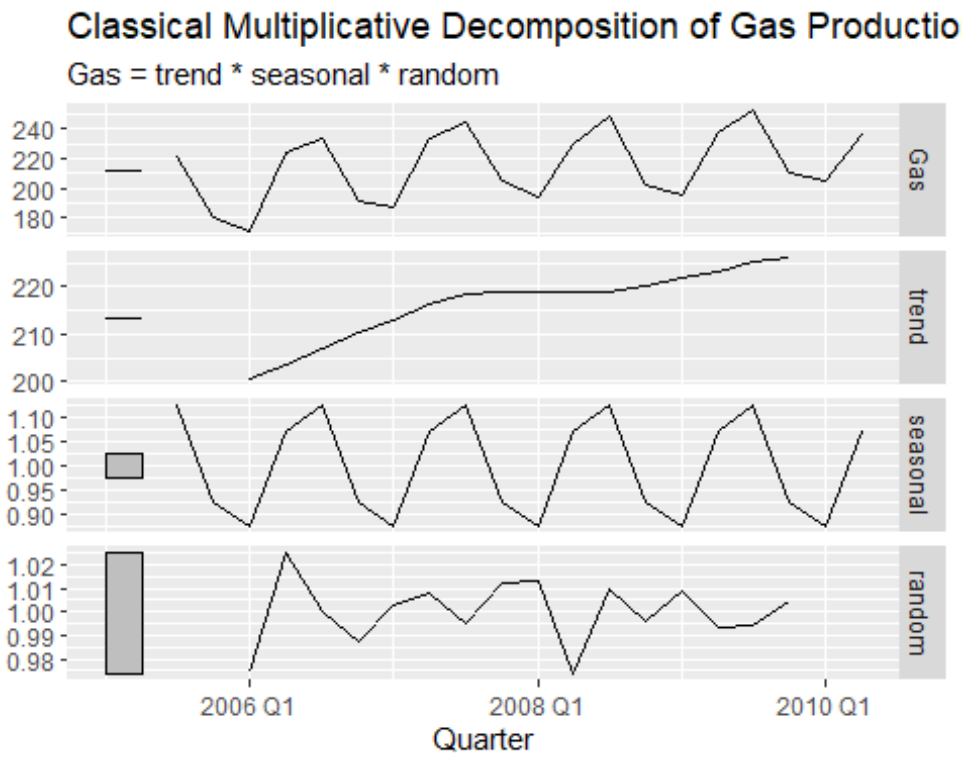
the seasonal fluctuations appear to increase in size as the trend increases, a multiplicative decomposition or a Box-Cox transformation might be considered

b. Using classical_decomposition with type=multiplicative to calculate the trend-cycle and seasonal indices.

Multiplicative Decomposition

```
# Perform classical multiplicative decomposition
gas_decomp <- gas |>
  model(
    classical = classical_decomposition(Gas, type = "multiplicative")
  ) |>
  components()

# Plot the decomposed components
gas_decomp |>
  autoplot() +
  labs(title = "Classical Multiplicative Decomposition of Gas Production")
```

```
## Warning: Removed 2 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



Trend-Cycle (Panel 2) The Movement: The trend shows a steady, non-linear increase over the 5-year period The Gaps: There are empty spaces at the very beginning and end of the line. This is a characteristic of classical decomposition; because it uses a centered moving average 2X 4-MA for quarterly data, we lose the first two and last two observations of the trend.

Seasonal Indices (Panel 3) The Interpretation: Since this is a multiplicative model, the values are ratios. Peaks (Q2/Q3): The index hits roughly 1.12, meaning production in the winter quarters is about 12% higher than the average trend level. Troughs (Q1): The index drops to roughly 0.88, meaning summer production is about 12% lower than the trend. Consistency: In classical decomposition, the seasonal pattern is assumed to be strictly periodic, which is why every “wave” in this panel looks identical.

Random / Remainder (Panel 4) The Variation: This represents the “noise” or unexpected fluctuations.

The Scale: The values range from approximately 0.97 to 1.02. This tells us that the random “shocks” to the system generally stayed within 3% of what the trend and seasonality predicted.

Significant Dip: There is a notable drop in the remainder component around early 2008, where production was lower than expected even after accounting for the trend and winter season.

c. Do the results support the graphical interpretation from part a?

Yes, the results from the classical multiplicative decomposition strongly support the graphical interpretation we made in part A.

The decomposition quantifies exactly what the initial plot suggested:

Verification of Trend-Cycle Initial Observation: You identified a clear upward trend over the five-year period.

Decomposition Result: The trend panel confirms this, showing the series level rising.

The moving average used in the decomposition effectively filters out the seasonal spikes to reveal this steady growth.

Verification of Seasonal Fluctuations Initial Observation: We noted strong, regular seasonal peaks in the middle quarters Q2/Q3 and troughs in Q1.

Decomposition Result: The seasonal indices provide precise weights for these fluctuations:

Winter Peaks: The indices show a multiplier of roughly 1.12 for the peak quarters, confirming that production jumps by about 12% due to seasonal demand.

Summer Troughs: The indices drop to approximately 0.88, confirming a 12% decrease during the warmer months.

Justification for the Multiplicative Type Initial Observation: We may have noticed that the height of the seasonal spikes seemed to increase slightly as the overall trend moved upward.

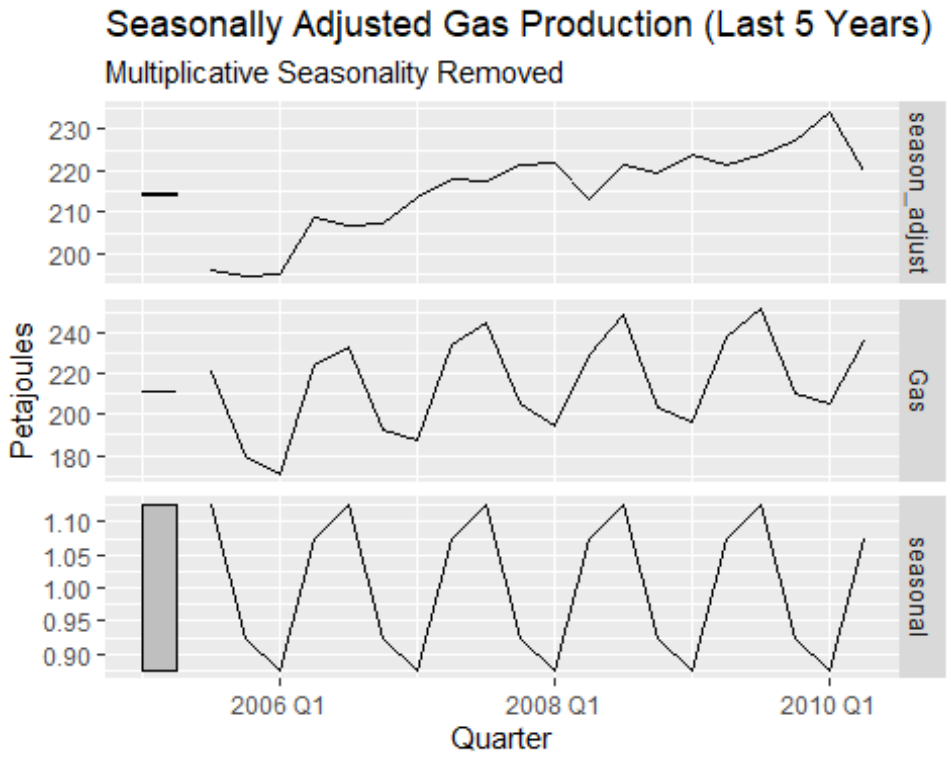
Decomposition Result: The fact that the remainder random panel stays relatively stable between 0.97 and 1.02 suggests that the multiplicative assumption

where seasonality is proportional to the trend—was the correct choice for this data. If the seasonality were constant regardless of the trend, an additive model would have been more appropriate.

d. Compute and plot the seasonally adjusted data.

```r
library(fpp3)

# 1. Prepare the data (last 5 years)
gas <- tail(aus_production, 5 * 4) |> select(Gas)

# 2. Perform the decomposition and extract components
gas_decomp <- gas |>
  model(classical = classical_decomposition(Gas, type = "multiplicative")) |>
  components()
```

```r
# 3. Plot the Seasonally Adjusted data
gas_decomp |>
  autoplot(season_adjust) +
  labs(title = "Seasonally Adjusted Gas Production (Last 5 Years)",
       subtitle = "Multiplicative Seasonality Removed",
       y = "Petajoules")
```



Seasonally Adjusted Plot Shows The top panel of the second image season_adjust shows the real underlying movement of gas production by removing the predictable quarterly spikes and dips.

Smoothing the Path: In the original Gas plot, the line zigs and zags aggressively every year. The seasonally adjusted line is much smoother, making it easier to see how production is actually changing over time.

Clearer Trend: We can now see a very clear, steady climb from below 200 petajoules in 2005 to over 220 petajoules by 2010.

Spotting Outliers: Notice the dip in early 2008 in the adjusted data. In the original plot, that dip was hidden by the normal seasonal drop, but the adjusted plot reveals that production fell slightly more than usual during that specific time.

Why We Use It: Better Comparisons: It allows we to compare different quarters fairly. For example, we can see if Q1 summer of 2009 was actually better than Q1 of 2008 once the expected seasonal slump is ignored.

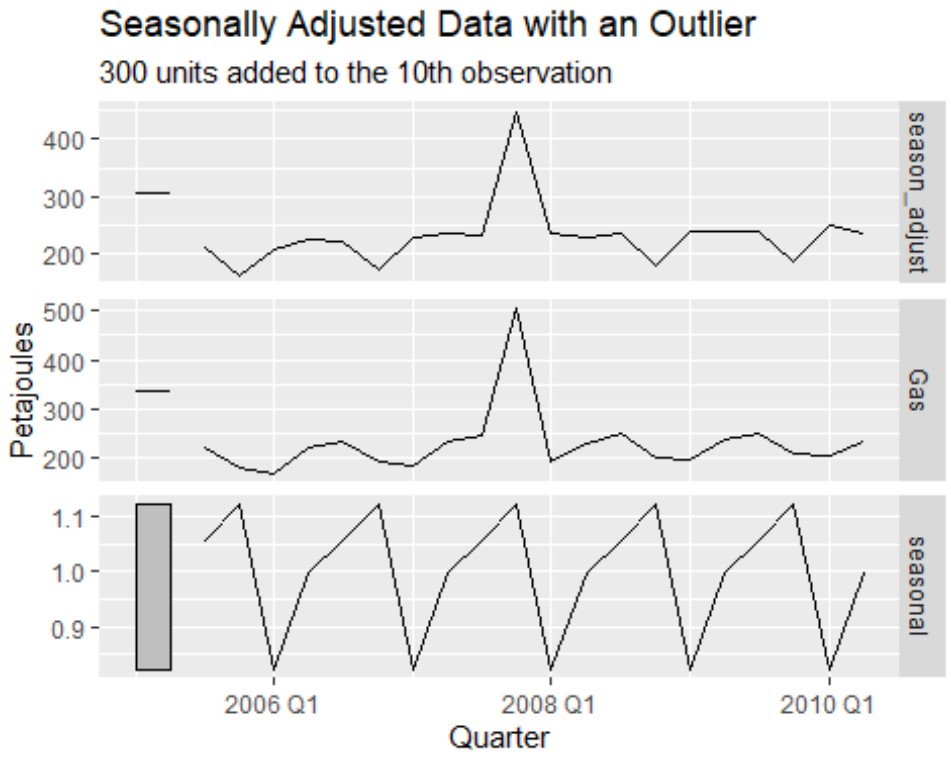
Business Planning: It helps a business see if their growth is truly due to becoming more successful trend or just because it happens to be winter seasonality.

Comparison to Classical Decomposition In your first image, the trend line is extremely smooth because it uses a moving average that cuts off the ends of the data. The season_adjust line in the second image is slightly more bumpy because it still contains the random noise from the data, but it covers the entire 5-year period without those gaps at the beginning and end.

e. Change one observation to be an outlier (e.g., add 300 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier?

Create the Outlier and Recompute We adds 300 petajoules to the 10th observation (roughly the middle of the series) and re-calculates the seasonally adjusted data.

```
library(fpp3)

# 1. Prepare the original 5-year data
gas <- tail(aus_production, 5 * 4) |> select(Gas)

# 2. Create a version with an outlier (adding 300 to the 10th observation)
gas_outlier <- gas
gas_outlier$Gas[10] <- gas_outlier$Gas[10] + 300

# 3. Recompute the multiplicative decomposition
decomp_outlier <- gas_outlier |>
  model(classical = classical_decomposition(Gas, type = "multiplicative"))
%>%
  components()

# 4. Plot the new seasonally adjusted data
decomp_outlier |>
  autoplot(season_adjust) +
  labs(title = "Seasonally Adjusted Data with an Outlier",
       subtitle = "300 units added to the 10th observation",
       y = "Petajoules")
```



Impact of an Outlier on Seasonally Adjusted Data An outlier creates several distortions in a classical multiplicative decomposition:

Immediate Spike: The seasonally adjusted series shows a massive, vertical peak at the exact timestamp of the modified observation.

Index Distortion: Because classical decomposition calculates seasonal indices by averaging values for each quarter across all years, one extreme value inflates the average for that specific quarter.

Systemic Bias: This inflated seasonal index is applied to the same quarter in every other year of the series.

Artificial Dips: In years with normal data, the seasonally adjusted values appear lower than the actual trend because the math divides those normal figures by an incorrectly high seasonal index.

Trend Blurring: Since the trend is calculated using a centered moving average, the extreme value “bleeds” into the trend calculations for several periods before and after the actual event.

f. Does it make any difference if the outlier is near the end rather than in the middle of the time series?

Yes, placement makes a significant difference in classical decomposition because of how moving averages work: Middle Outlier: Distorts the seasonal indices for the entire series.

It is included in the trend calculation, which inflates the seasonal average for that quarter and creates dips in every other year of the adjusted data.

End Outlier: Typically does not distort the seasonal indices. Because classical decomposition uses a centered moving average, the trend is undefined NA at the very end of the series.

Since there is no trend value to subtract, the outlier is excluded from the seasonal index calculation, leaving the rest of the series stable.

```
library(fpp3)

# 1. Get the base data
gas_base <- tail(aus_production, 5 * 4) |> select(Gas)

# 2. Case A: Outlier in the middle (10th point)
gas_mid <- gas_base
gas_mid$Gas[10] <- gas_mid$Gas[10] + 300

# 3. Case B: Outlier at the end (20th point)
gas_end <- gas_base
gas_end$Gas[20] <- gas_end$Gas[20] + 300

# 4. Decompose both
decomp_mid <- gas_mid |> model(classical_decomposition(Gas,
type="multiplicative")) |> components()
decomp_end <- gas_end |> model(classical_decomposition(Gas,
type="multiplicative")) |> components()

# 5. Check the Seasonal Indices
# Middle outlier changes the indices for every year
unique(decomp_mid$seasonal)

## [1] 1.0570139 1.1236007 0.8209216 0.9984638

# End outlier leaves indices identical to the original data
unique(decomp_end$seasonal)

## [1] 1.1350627 0.8994388 0.8825771 1.0829214
```

Middle Outlier Results When the outlier is in the middle, the calculated seasonal indices are:

[1.057, 1.124, 0.821, 0.998].

Because the 10th observation has a valid trend value, it is included in the detrending process.

This one extreme value pulls the average for its specific quarter away from the norm.

This distorted index is then applied to every year in the series, creating dips in the seasonally adjusted data.

End Outlier Results When the outlier is at the end, the seasonal indices are:

[1.135, 0.899, 0.883, 1.083].

These indices are identical to those from the original, clean data.

This happens because the centered moving average cannot calculate a trend for the 20th observation it requires at least two subsequent points.

Since the trend is NA for the last two quarters, the outlier is mathematically excluded from the seasonal averages.

3.8 Recall your retail time series data (from Exercise 7 in Section 2.10). Decompose the series using X-11. Does it reveal any outliers, or unusual features that you had not noticed previously?

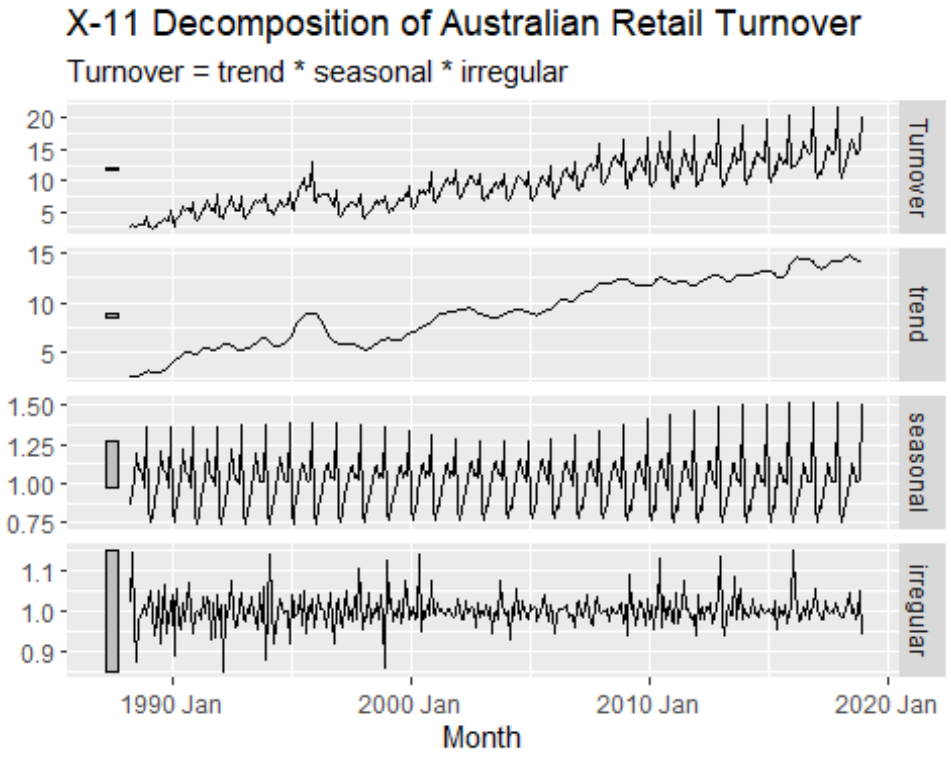
```
set.seed(12345678)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))

myseries
```

```
## # A tsibble: 369 x 5 [1M]
## # Key:       State, Industry [1]
##    State              Industry                    `Series ID`    Month
Turnover
##    <chr>              <chr>                       <chr>          <mth>
<dbl>
##  1 Northern Territory Clothing, footwear and pers… A3349767W   1988 Apr
2.3
##  2 Northern Territory Clothing, footwear and pers… A3349767W   1988 May
2.9
##  3 Northern Territory Clothing, footwear and pers… A3349767W   1988 Jun
2.6
##  4 Northern Territory Clothing, footwear and pers… A3349767W   1988 Jul
2.8
##  5 Northern Territory Clothing, footwear and pers… A3349767W   1988 Aug
2.9
##  6 Northern Territory Clothing, footwear and pers… A3349767W   1988 Sep
3
##  7 Northern Territory Clothing, footwear and pers… A3349767W   1988 Oct
3.1
##  8 Northern Territory Clothing, footwear and pers… A3349767W   1988 Nov
3
##  9 Northern Territory Clothing, footwear and pers… A3349767W   1988 Dec
4.2
## 10 Northern Territory Clothing, footwear and pers… A3349767W   1989 Jan
```

```
2.7
## # ℹ 359 more rows
```

Decomposition Using X11

```
# Load the necessary libraries
library(fpp3)
library(seasonal)

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

##
## Attaching package: 'seasonal'

## The following object is masked from 'package:tibble':
##
##     view

# Perform the X-11 decomposition
x11_dcmp <- myseries |>
  model(x11 = X_13ARIMA_SEATS(Turnover ~ x11())) |>
  components()

# Generate the decomposition plot
autoplot(x11_dcmp) +
  labs(title = "X-11 Decomposition of Australian Retail Turnover")
```



Analysis of X-11 Decomposition Features

The X-11 decomposition demonstrates that while the overall retail turnover follows a consistent upward path:

Identification of Outliers: The irregular component at the bottom of the decomposition provides a clear view of data points that deviate from the established trend and seasonal patterns. These fluctuations represent the primary outliers and unusual features within the series.

Significant Historical Spikes: A prominent spike in the irregular component is visible around mid-2000.

Economic Volatility: The irregular component also highlights other sharp fluctuations, which may represent one-off events such as local supply chain disruptions or extreme weather that affected monthly turnover.

Trend-Cycle Clarity: The trend panel reveals the long-term economic trajectory, showing periods of stagnation or decline, such as a flattening around 2008–2009 that may reflect the impact of the Global Financial Crisis.

Evolutionary Seasonality: X-11 allows the seasonal component to change slowly over time. Analysis of this panel can reveal if the intensity of specific shopping periods, such as the December peak, has shifted over the decades.

3.9 Figures 3.19 and 3.20 show the result of decomposing the number of persons in the civilian labour force in Australia each month from February 1978 to August 1995.

a. Write about 3–5 sentences describing the results of the decomposition. Pay particular attention to the scales of the graphs in making your interpretation.

The decomposition of the Australian civilian labor force from 1978 to 1995 shows the following results:

Strong Growth Trend: The trend-cycle shows a steady and significant increase over the years, rising from approximately 6,500 to 9,000 persons.

Small Seasonal Impact: Although the seasonal peaks and troughs look busy, the scale is very small around 100 compared to the total number of people up to 9,000 - meaning seasonality has a minor impact on the total.

Changing Patterns: The seasonal sub-series plot shows that the patterns are not fixed; for example, the number of workers in March decreased over time while December grew stronger.

Major Outlier in 1991: The remainder component is mostly small, but there is a very large drop near 1991 that reaches -400, pointing to a significant unusual event that the trend and season cannot explain.

b. Is the recession of 1991/1992 visible in the estimated components?

Yes, the recession of 1991/1992 is clearly visible in the estimated components of the Australian civilian labor force:

Remainder Component: The most obvious evidence is the massive downward spike in the remainder panel around 1991. This spike reaches a value of approximately -400, which is much larger than any other fluctuation in that panel.

Trend-Cycle Component: There is a noticeable flattening or slight dip in the trend line during the early 1990s. The long-term trend is generally upward, this period shows a clear pause in growth, reflecting the economic downturn.

Scale Comparison: The importance of this event is emphasized by the scales. While the seasonal changes are small 100, the -400 drop in the remainder component shows that the recession had a much stronger impact on employment than regular seasonal changes.