# Time Series Homework

Julius Hai

2025-09-02

## Question 1a — Explore datasets

```r
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.5.1

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

## -- Attaching packages -------------------------------------------- fpp3 1.0.1 --

## v tibble      3.2.1     v tsibble     1.1.6
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.4.2
## v lubridate   1.9.4     v fable       0.4.1
## v ggplot2     3.5.2

## Warning: package 'tidyr' was built under R version 4.5.1

## Warning: package 'lubridate' was built under R version 4.5.1

## Warning: package 'tsibble' was built under R version 4.5.1

## Warning: package 'tsibbledata' was built under R version 4.5.1

## Warning: package 'feasts' was built under R version 4.5.1

## Warning: package 'fabletools' was built under R version 4.5.1

## Warning: package 'fable' was built under R version 4.5.1

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

```r
library(tsibbledata)

# Look at documentation (only works in Help tab; commented out for Knit)
# ?gafa_stock
# ?vic_elec

# Glimpse structure
glimpse(gafa_stock)
```

```
## Rows: 5,032
## Columns: 8
## Key: Symbol [4]
## $ Symbol    <chr> "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAP~
## $ Date      <date> 2014-01-02, 2014-01-03, 2014-01-06, 2014-01-07, 2014-01-08,~
## $ Open      <dbl> 79.38286, 78.98000, 76.77857, 77.76000, 76.97285, 78.11429, ~
## $ High      <dbl> 79.57571, 79.10000, 78.11429, 77.99429, 77.93714, 78.12286, ~
## $ Low       <dbl> 78.86000, 77.20428, 76.22857, 76.84571, 76.95571, 76.47857, ~
## $ Close     <dbl> 79.01857, 77.28286, 77.70428, 77.14857, 77.63715, 76.64571, ~
## $ Adj_Close <dbl> 66.96433, 65.49342, 65.85053, 65.37959, 65.79363, 64.95345, ~
## $ Volume    <dbl> 58671200, 98116900, 103152700, 79302300, 64632400, 69787200,~
```

```r
glimpse(vic_elec)
```

```
## Rows: 52,608
## Columns: 5
## $ Time        <dttm> 2012-01-01 00:00:00, 2012-01-01 00:30:00, 2012-01-01 01:0~
## $ Demand      <dbl> 4382.825, 4263.366, 4048.966, 3877.563, 4036.230, 3865.597~
## $ Temperature <dbl> 21.40, 21.05, 20.70, 20.55, 20.40, 20.25, 20.10, 19.60, 19~
## $ Date        <date> 2012-01-01, 2012-01-01, 2012-01-01, 2012-01-01, 2012-01-0~
## $ Holiday     <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE~
```

```r
# Peek at first 6 rows
gafa_stock |> as_tibble() |> dplyr::slice_head(n = 6)
```

```
## # A tibble: 6 x 8
##   Symbol Date        Open  High   Low Close Adj_Close    Volume
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl>
## 1 AAPL   2014-01-02  79.4  79.6  78.9  79.0      67.0  58671200
## 2 AAPL   2014-01-03  79.0  79.1  77.2  77.3      65.5  98116900
## 3 AAPL   2014-01-06  76.8  78.1  76.2  77.7      65.9 103152700
## 4 AAPL   2014-01-07  77.8  78.0  76.8  77.1      65.4  79302300
## 5 AAPL   2014-01-08  77.0  77.9  77.0  77.6      65.8  64632400
## 6 AAPL   2014-01-09  78.1  78.1  76.5  76.6      65.0  69787200
```

```r
vic_elec   |> as_tibble() |> dplyr::slice_head(n = 6)
```

```
## # A tibble: 6 x 5
##   Time                Demand Temperature Date       Holiday
##   <dttm>               <dbl>       <dbl> <date>     <lgl>
## 1 2012-01-01 00:00:00  4383.        21.4 2012-01-01 TRUE
## 2 2012-01-01 00:30:00  4263.        21.0 2012-01-01 TRUE
```

```
## 3 2012-01-01 01:00:00  4049.        20.7 2012-01-01 TRUE
## 4 2012-01-01 01:30:00  3878.        20.6 2012-01-01 TRUE
## 5 2012-01-01 02:00:00  4036.        20.4 2012-01-01 TRUE
## 6 2012-01-01 02:30:00  3866.        20.2 2012-01-01 TRUE
```

**Answer (1a):**
The dataset `gafa_stock` contains daily stock prices for major tech companies (AAPL, AMZN, FB, and GOOG) with variables such as Open, High, Low, Close, Adjusted Close, and Volume.

The dataset `vic_elec` contains half-hourly electricity demand for Victoria, Australia, including Demand (MWh), Temperature (°C), Date, and whether the day was a Holiday.

In summary, `gafa_stock` is a **financial time series** showing stock market activity, while `vic_elec` is an **energy demand time series** reflecting electricity usage and related conditions.

## Question 1b — Time interval of each series

```
# Sampling interval (frequency)
interval(gafa_stock)
```

```
## <interval[1]>
## [1] !
```

```
interval(vic_elec)
```

```
## <interval[1]>
## [1] 30m
```

```
# Which column is used as the time index?
index_var(gafa_stock)
```

```
## [1] "Date"
```

```
index_var(vic_elec)
```

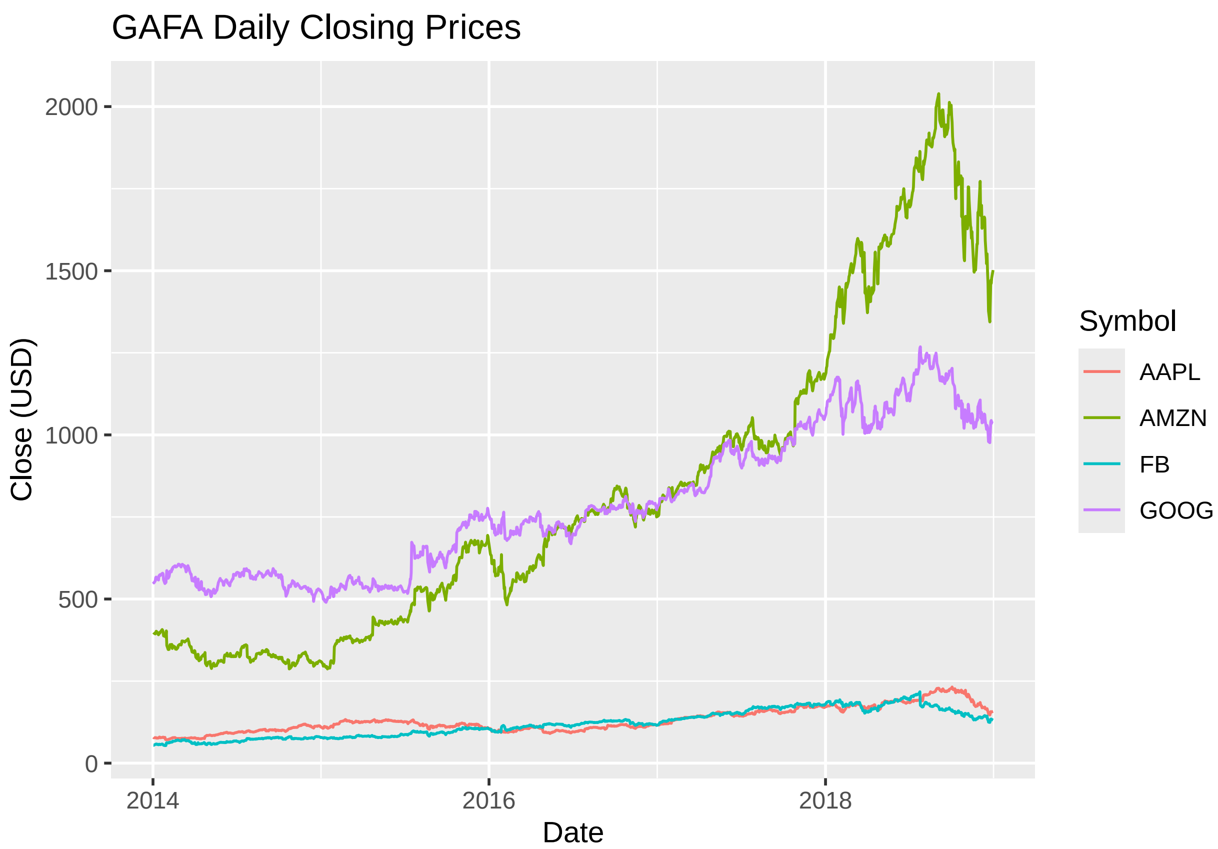
```
## [1] "Time"
```

**Answer (1b):**
- For `gafa_stock`, the time interval is **daily** (1 day), and the time index is **Date**.
- For `vic_elec`, the time interval is **30 minutes**, and the time index is **Time**.

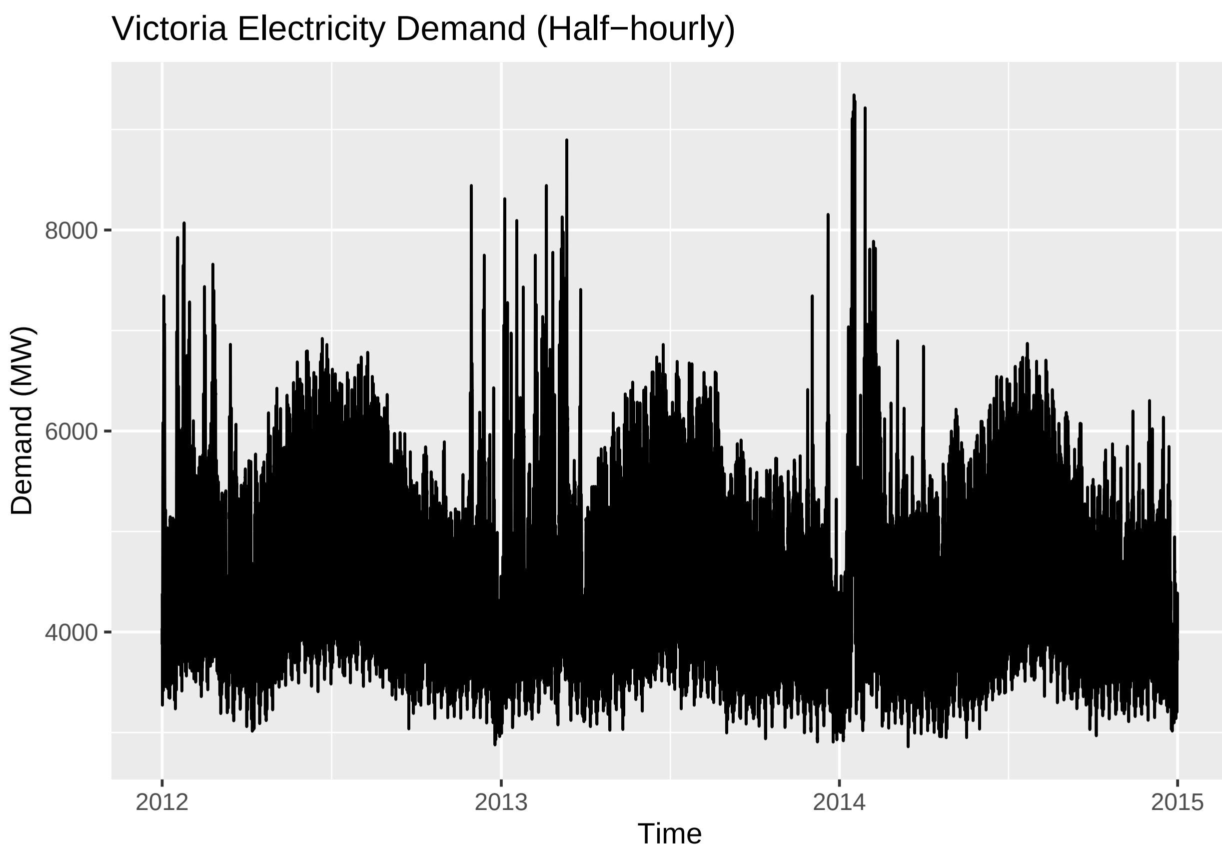
## Question 1c — Time plots

```
# GAFA: daily Close price (one panel per stock)
gafa_stock |>
  select(Symbol, Date, Close) |>
  autoplot(Close) +
  labs(
```

```
    title = "GAFA Daily Closing Prices",
    x = "Date",
    y = "Close (USD)"
  )
```



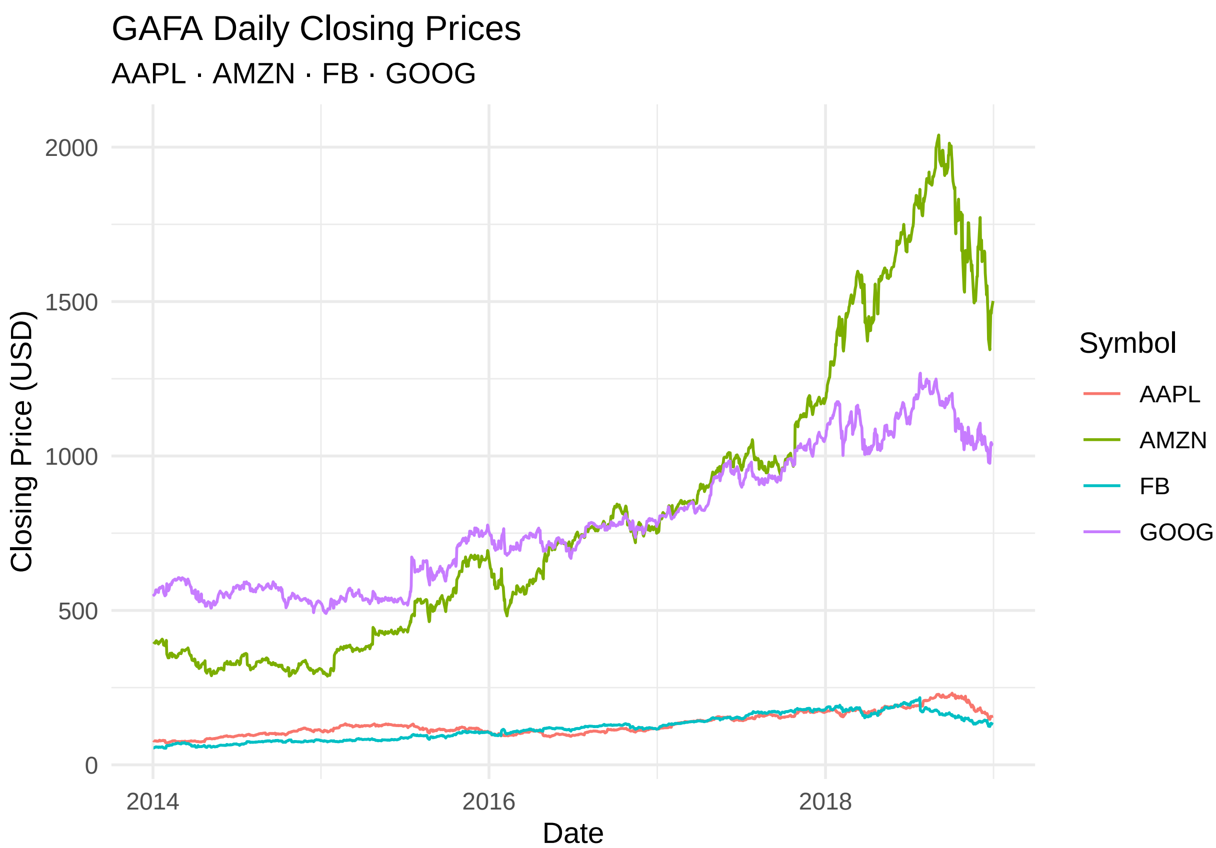

```
# Victoria electricity demand: half-hourly Demand
vic_elec |>
  select(Time, Demand) |>
  autoplot(Demand) +
  labs(
    title = "Victoria Electricity Demand (Half-hourly)",
    x = "Time",
    y = "Demand (MW)"
  )
```




**Answer (1c):** The GAFA plot shows daily closing prices for AAPL, AMZN, FB, and GOOG (faceted by symbol). The VIC electricity plot shows strong intra-day variation and broader seasonal patterns in demand at a 30-minute frequency.

### Question 1d — Customized GAFA plot

```
gafa_stock |>
  select(Symbol, Date, Close) |>
  autoplot(Close) +
  labs(
    title = "GAFA Daily Closing Prices",
    subtitle = "AAPL · AMZN · FB · GOOG",
    x = "Date",
    y = "Closing Price (USD)"
  ) +
  theme_minimal()
```




**Answer (1d):**
The customized GAFA stock plot clearly displays daily closing prices for AAPL, AMZN, FB, and GOOG. The title and subtitle provide context, axis labels specify units, and `theme_minimal()` improves readability by reducing visual clutter.

### Question 1e — Peak closing day(s) per stock (using filter)

```
# Keep only the columns we need, then find the rows where Close is the
# maximum within each stock (Symbol). If there are ties, all tied days appear.
peak_close <- gafa_stock |>
  select(Symbol, Date, Close) |>
  group_by(Symbol) |>
  filter(Close == max(Close, na.rm = TRUE)) |>
  arrange(Symbol, Date) |>
  ungroup()

peak_close
```

```
## # A tsibble: 4 x 3 [!]
## # Key:       Symbol [4]
##   Symbol Date       Close
##   <chr>  <date>     <dbl>
## 1 AAPL   2018-10-03  232.
## 2 AMZN   2018-09-04 2040.
```

```
## 3 FB     2018-07-25  218.
## 4 GOOG   2018-07-26 1268.
```

### Question 2a — Load the CSV

```r
# If tute1.csv is in your working folder:
tute1 <- readr::read_csv("tute1.csv", show_col_types = FALSE)

# Quick check
dplyr::glimpse(tute1)
```

```
## Rows: 100
## Columns: 4
## $ Quarter  <date> 1981-03-01, 1981-06-01, 1981-09-01, 1981-12-01, 1982-03-01, ~
## $ Sales    <dbl> 1020.2, 889.2, 795.0, 1003.9, 1057.7, 944.4, 778.5, 932.5, 99~
## $ AdBudget <dbl> 659.2, 589.0, 512.5, 614.1, 647.2, 602.0, 530.7, 608.4, 637.9~
## $ GDP      <dbl> 251.8, 290.9, 290.8, 292.4, 279.1, 254.0, 295.6, 271.7, 259.6~
```

**Answer (2a):**
`tute1` loaded with **100 rows** and **4 variables**:
- `Quarter` as a **Date** (quarter timestamps like 1981-03-01),
- numeric series `Sales`, `AdBudget`, and `GDP`.
Next, I'll coerce `Quarter` to `yearquarter()` and set it as the **tsibble index**.

### Question 2b — Convert to quarterly tsibble

```r
library(fpp3)

mytimeseries <- tute1 |>
  mutate(Quarter = yearquarter(Quarter)) |>
  as_tsibble(index = Quarter)

# Print and sanity-check the index & interval
mytimeseries
```

```
## # A tsibble: 100 x 4 [1Q]
##    Quarter Sales AdBudget   GDP
##      <qtr> <dbl>    <dbl> <dbl>
##  1 1981 Q1 1020.     659.  252.
##  2 1981 Q2  889.     589   291.
##  3 1981 Q3  795      512.  291.
##  4 1981 Q4 1004.     614.  292.
##  5 1982 Q1 1058.     647.  279.
##  6 1982 Q2  944.     602   254
##  7 1982 Q3  778.     531.  296.
##  8 1982 Q4  932.     608.  272.
##  9 1983 Q1  996.     638.  260.
## 10 1983 Q2  908.     582.  280.
## # i 90 more rows
```

```r
index_var(mytimeseries)
```

```
## [1] "Quarter"
```

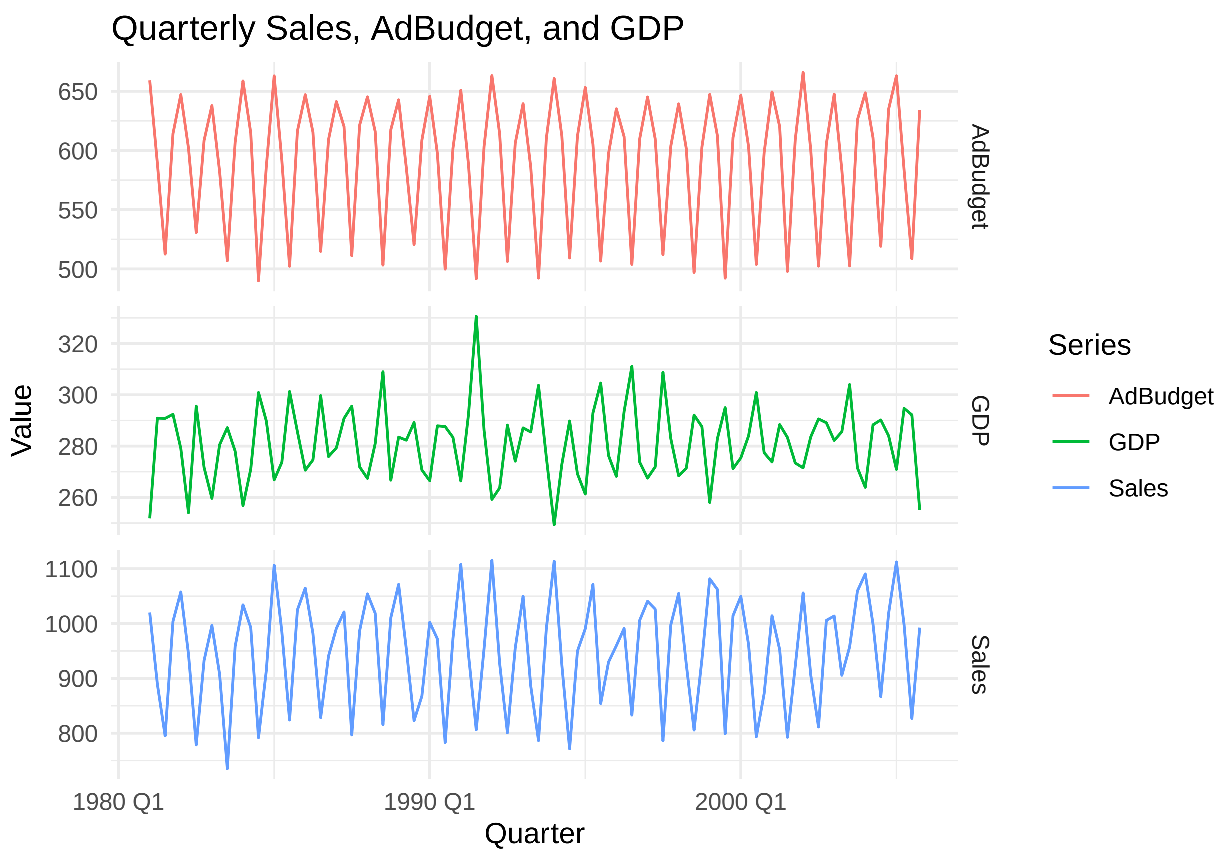
```r
interval(mytimeseries)
```

```
## <interval[1]>
## [1] 1Q
```

**Answer (2b):**
The data has been converted to a **tsibble** named `mytimeseries` with `Quarter` as the **time index** (`index_var = "Quarter"`), and a **regular quarterly interval** (`interval = 1Q`). The measured variables (`Sales`, `AdBudget`, `GDP`) are now aligned on this quarterly timeline for time-series analysis and plotting.

## Question 2c — Faceted time plots

```r
# Long format for plotting all three series
my_long <- mytimeseries |>
  tidyr::pivot_longer(-Quarter, names_to = "Series", values_to = "Value")

# Faceted time plots with free y-scales
my_long |>
  ggplot(aes(Quarter, Value, colour = Series)) +
  geom_line() +
  facet_grid(Series ~ ., scales = "free_y") +
  labs(
    title = "Quarterly Sales, AdBudget, and GDP",
    x = "Quarter",
    y = "Value",
    colour = "Series"
  ) +
  theme_minimal()
```




**Answer (2c):**
The data were reshaped to long form and plotted as three **faceted** time series—one panel each for **Sales**, **AdBudget**, and **GDP**—with **free y-scales**. Faceting prevents the large magnitude of GDP from compressing the other series, so the within-series patterns (level and variation over quarters) are clearly visible and comparable by timing (peaks/troughs) rather than absolute scale.

## Question 2d — Plot without facets

```r
# All three series on a single panel (shared y-axis)
my_long |>
  ggplot(aes(Quarter, Value, colour = Series)) +
  geom_line() +
  labs(
    title = "Quarterly Sales, AdBudget, and GDP - single panel",
    x = "Quarter",
    y = "Value",
    colour = "Series"
  ) +
  theme_minimal()
```


**Answer (2d):**
With all three series on a single shared y-axis, the much larger **GDP** values compress the scale so **Sales** and **AdBudget** appear nearly flat, obscuring their variability—unlike the faceted view where each series' pattern is visible.

### Question 3a — Compute GDP per capita

```
library(fpp3)  # includes tsibbledata::global_economy

# GDP is in billions of USD; Population is in millions.
# => GDP per capita (USD) = 1000 * GDP / Population
gdp_pc <- global_economy |>
  mutate(
    gdp_pc_usd  = 1000 * GDP / Population,  # USD per person
    gdp_pc_thou = GDP / Population          # thousand USD per person (for plotting)
  )

# Quick sanity checks
range(gdp_pc$Year, na.rm = TRUE)
```

```
## [1] 1960 2017
```

```
dplyr::n_distinct(gdp_pc$Country)
```

```
## [1] 263
```

```
dplyr::glimpse(gdp_pc)
```

```
## Rows: 15,150
## Columns: 11
## Key: Country [263]
## $ Country     <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan"~
## $ Code        <fct> AFG, AFG, AFG, AFG, AFG, AFG, AFG, AFG, AFG, AFG, AFG, AFG~
## $ Year        <dbl> 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969~
## $ GDP         <dbl> 537777811, 548888896, 546666678, 751111191, 800000044, 100~
## $ Growth      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ CPI         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ Imports     <dbl> 7.024793, 8.097166, 9.349593, 16.863910, 18.055555, 21.412~
## $ Exports     <dbl> 4.132233, 4.453443, 4.878051, 9.171601, 8.888893, 11.25827~
## $ Population  <dbl> 8996351, 9166764, 9345868, 9533954, 9731361, 9938414, 1015~
## $ gdp_pc_usd  <dbl> 59777.33, 59878.15, 58492.87, 78782.76, 82208.44, 101290.4~
## $ gdp_pc_thou <dbl> 59.77733, 59.87815, 58.49287, 78.78276, 82.20844, 101.2904~
```

**Answer (3a):**
The `global_economy` data have been augmented with **GDP per capita** measures. The table now spans **1960–2017** across **263 countries** (15,150 rows, 11 columns; keyed by *Country* with annual *Year*). Two new variables were added:

- `gdp_pc_usd` — GDP per person (**USD**).
- `gdp_pc_thou` — GDP per person in **thousands of USD**.

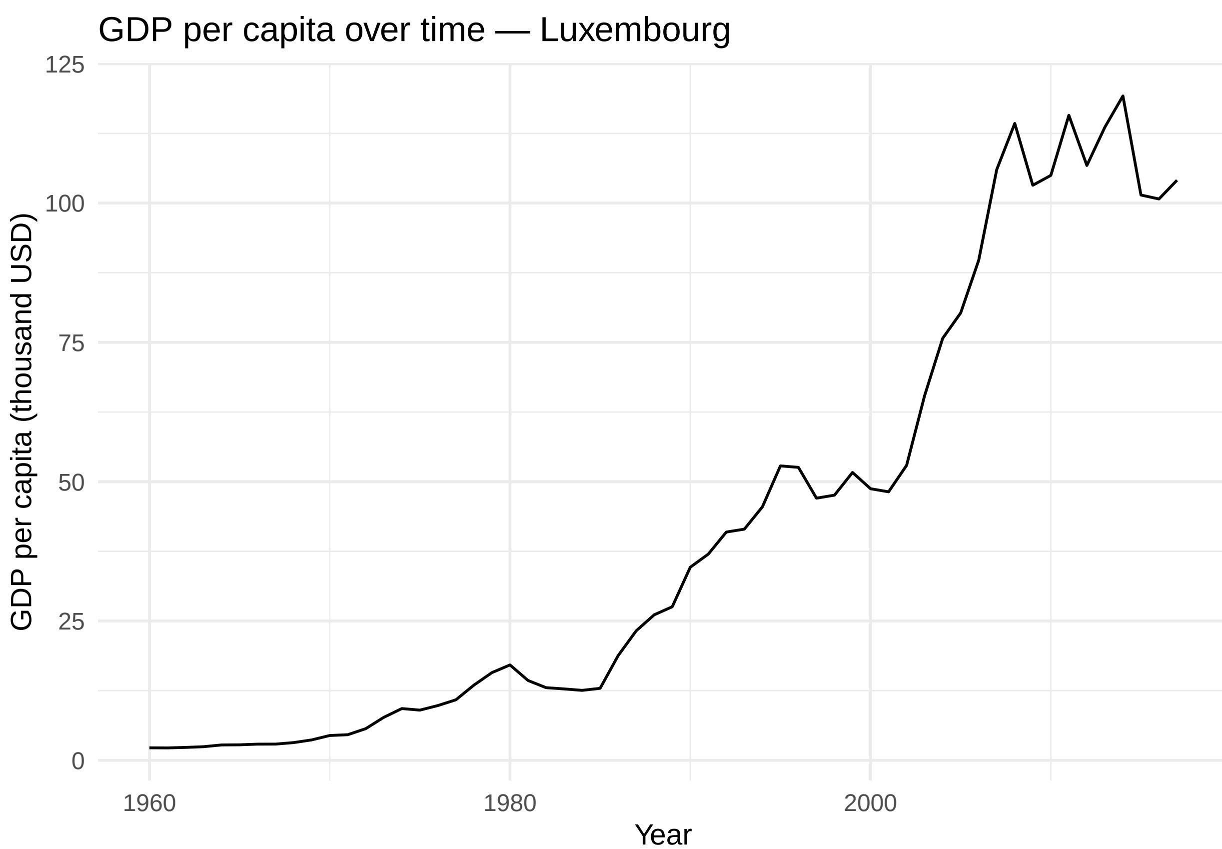
This prepares the dataset for comparing living standards over time and across countries.

## Question 3b — Highest GDP per capita (latest year) & trend

```
library(fpp3)

# 1) Ensure correct per-capita units (overwrite if previously defined)
gdp_pc <- global_economy |>
  mutate(
    gdp_pc_usd  = GDP / Population,             # USD per person
    gdp_pc_thou = (GDP / Population) / 1000     # thousand USD per person
  )

# 2) Latest year in the dataset
latest_year <- max(gdp_pc$Year, na.rm = TRUE)

# 3) Country with the highest GDP per capita in the latest year
top_current <- gdp_pc |>
  filter(Year == latest_year) |>
  slice_max(order_by = gdp_pc_usd, n = 1, with_ties = FALSE) |>
  select(Country, Year, gdp_pc_usd, gdp_pc_thou)

top_current
```

```
## # A tsibble: 1 x 4 [1Y]
## # Key:       Country [1]
##   Country     Year gdp_pc_usd gdp_pc_thou
##   <fct>      <dbl>      <dbl>       <dbl>
## 1 Luxembourg  2017    104103.        104.
```

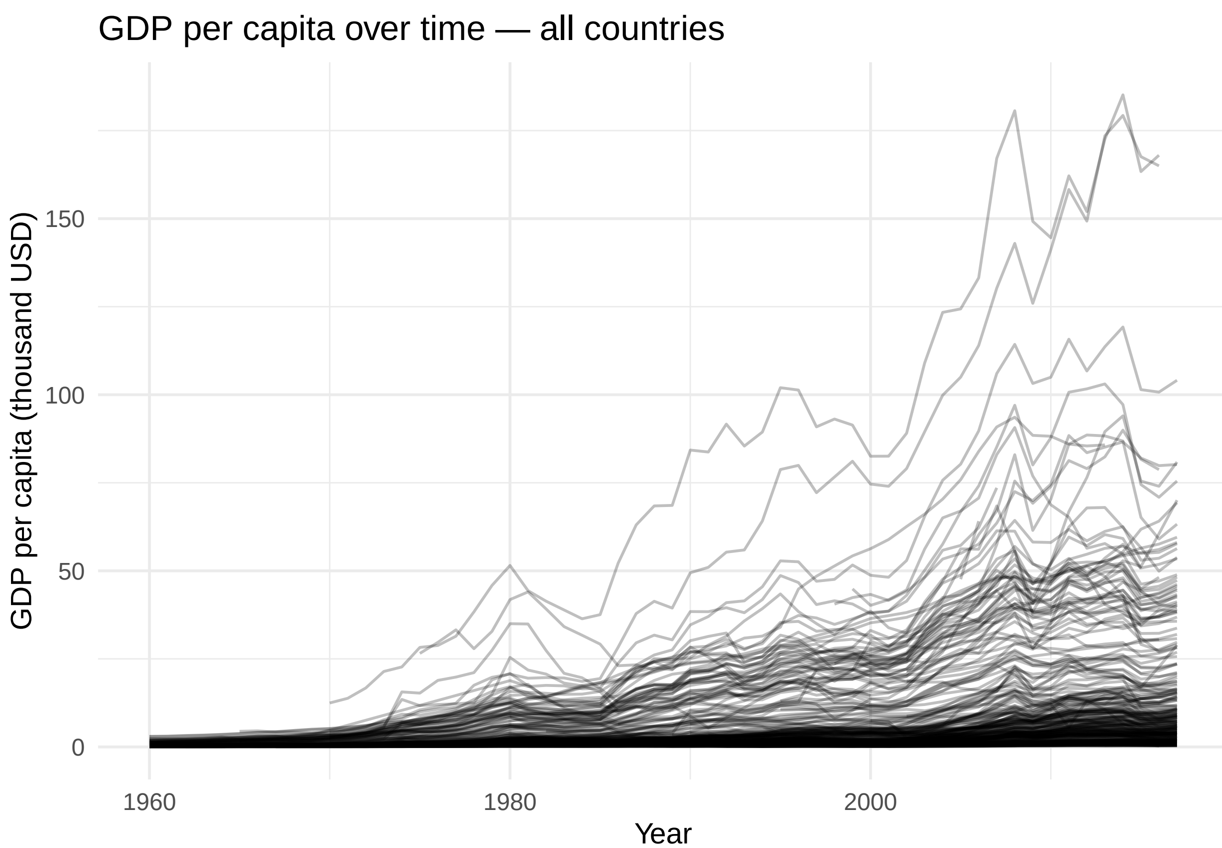
```r
# 4) Plot that country's GDP per capita over time (thousand USD)
top_country <- as.character(top_current$Country)

gdp_pc |>
  filter(Country == top_country) |>
  ggplot(aes(Year, gdp_pc_thou)) +
  geom_line() +
  labs(
    title = paste0("GDP per capita over time - ", top_country),
    x = "Year",
    y = "GDP per capita (thousand USD)"
  ) +
  theme_minimal()
```




**Answer (3b):**
Using correctly scaled GDP per capita (USD per person), the **latest year** is 2017. In that year, the country with the **highest GDP per capita** is **Luxembourg**, at about $1.04103 \times 10^5$ **USD/person** ( **104.1 thousand USD**). The time plot for **Luxembourg** shows a **generally rising trend** in living standards over the sample, with **short-run fluctuations** consistent with business-cycle or external shocks.

### Question 3c — All countries & leadership over time

```r
library(fpp3)

# Ensure per-capita variables exist
if (!exists("gdp_pc")) {
  gdp_pc <- global_economy |>
    mutate(
      gdp_pc_usd  = GDP / Population,          # USD per person
      gdp_pc_thou = gdp_pc_usd / 1000          # thousand USD per person
    )
}

# Clean for plotting (avoid NA/Inf warnings)
gdp_pc_clean <- gdp_pc |>
  filter(is.finite(gdp_pc_thou))

# 1) Plot GDP per capita for EACH country over time (spaghetti)
gdp_pc_clean |>
  ggplot(aes(Year, gdp_pc_thou, group = Country)) +
  geom_line(alpha = 0.25) +
  labs(
    title = "GDP per capita over time - all countries",
    x = "Year",
    y = "GDP per capita (thousand USD)"
  ) +
  theme_minimal()
```




```r
# 2) Highest GDP per capita in the latest year
latest_year <- max(gdp_pc$Year, na.rm = TRUE)
top_current <- gdp_pc |>
  filter(Year == latest_year) |>
  slice_max(order_by = gdp_pc_usd, n = 1, with_ties = FALSE) |>
  select(Country, Year, gdp_pc_usd, gdp_pc_thou)
top_current
```

```
## # A tsibble: 1 x 4 [1Y]
## # Key:       Country [1]
##   Country     Year gdp_pc_usd gdp_pc_thou
##   <fct>      <dbl>      <dbl>       <dbl>
## 1 Luxembourg  2017    104103.        104.
```

```r
# 3) Who led each year? (convert to tibble to allow group_by on Year)
top_by_year <- gdp_pc |>
  as_tibble() |>
  group_by(Year) |>
  slice_max(order_by = gdp_pc_usd, n = 1, with_ties = FALSE) |>
  ungroup() |>
  select(Year, Country, gdp_pc_usd, gdp_pc_thou)
top_by_year
```

```
## # A tibble: 58 x 4
```

```
##     Year Country       gdp_pc_usd gdp_pc_thou
##    <dbl> <fct>              <dbl>       <dbl>
##  1  1960 United States      3007.        3.01
##  2  1961 United States      3067.        3.07
##  3  1962 United States      3244.        3.24
##  4  1963 United States      3375.        3.37
##  5  1964 United States      3574.        3.57
##  6  1965 Kuwait             4429.        4.43
##  7  1966 Kuwait             4556.        4.56
##  8  1967 United States      4336.        4.34
##  9  1968 United States      4696.        4.70
## 10  1969 United States      5032.        5.03
## # i 48 more rows
```

```r
# 4) Summary: number of years each country was the leader
dplyr::count(top_by_year, Country, sort = TRUE)
```

```
## # A tibble: 6 x 2
##   Country                  n
##   <fct>                <int>
## 1 Monaco                  43
## 2 United States            8
## 3 Kuwait                   2
## 4 Liechtenstein            2
## 5 United Arab Emirates     2
## 6 Luxembourg               1
```

**Answer (3c):**
The spaghetti plot displays **GDP per capita** (in **thousand USD**) for **all countries** across the sample, revealing a broad **upward long-run trend** with substantial **cross-country dispersion**. Rows with non-finite values were removed before plotting, so a few series may show small gaps.

In **2017**, the **highest GDP per capita** is **Luxembourg**, at about $1.04103 \times 10^5$ **USD per person** ( **104.1 thousand USD**). The **year-by-year leaders** listed in `top_by_year` indicate that the top spot **shifts over time** among a small set of very high-income economies; the summary table shows **which countries led most often**.

## Question 4 — 3×5 MA equivalence

```r
# Equal-weight windows
w3 <- rep(1/3, 3)
w5 <- rep(1/5, 5)

# Single-step 7-term weights (convolution of w3 and w5)
# = (1/3)*(1/5) * c(1,2,3,3,3,2,1) = c(1,2,3,3,3,2,1)/15
w7 <- c(1, 2, 3, 3, 3, 2, 1) / 15

# Show weights as in the prompt
round(w7, 3)   # 0.067 0.133 0.200 0.200 0.200 0.133 0.067
```

```
## [1] 0.067 0.133 0.200 0.200 0.200 0.133 0.067
```

```r
sum(w7)        # should be 1
```

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

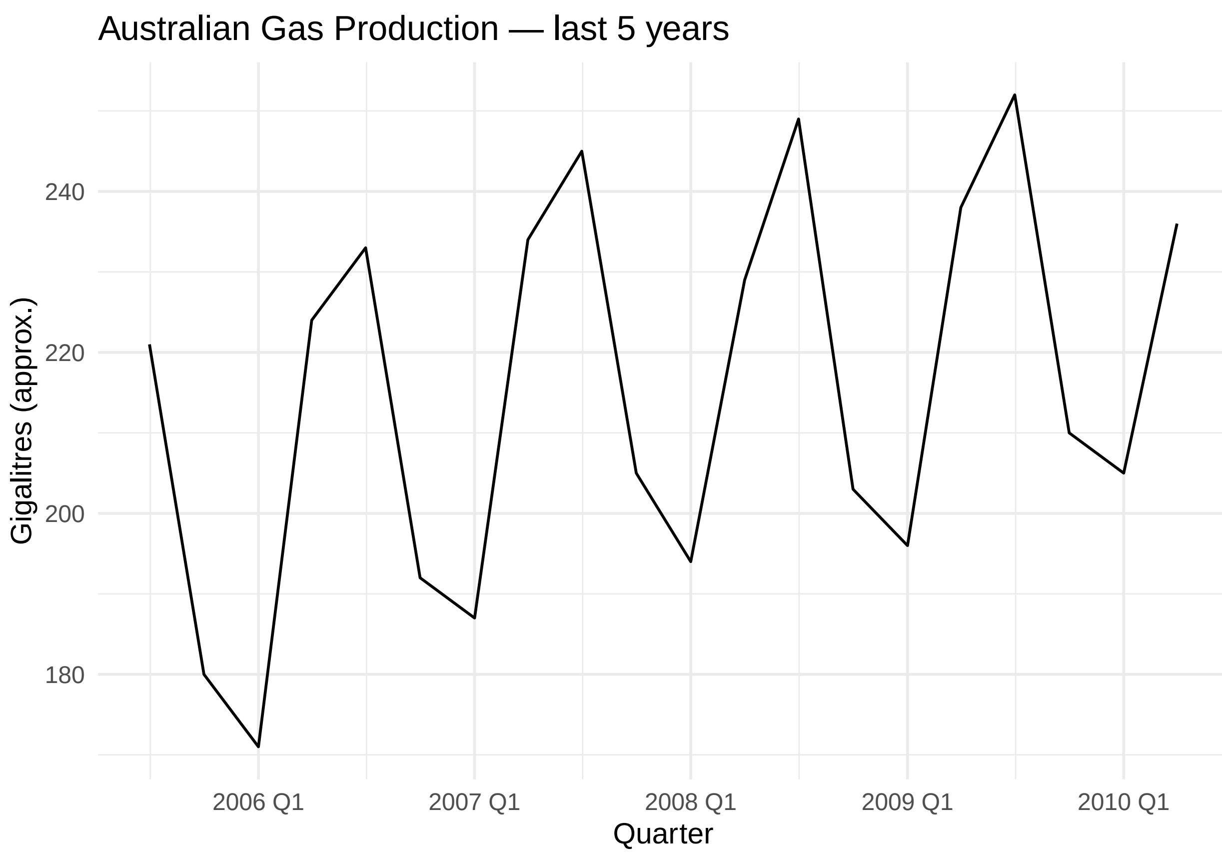
```r
# ---- Numeric verification on data ----
set.seed(123)
x <- rnorm(200)

# Centered moving averages
ma3    <- stats::filter(x, w3, sides = 2)   # 3-term MA
ma3x5  <- stats::filter(ma3, w5, sides = 2) # then 5-term MA  => 3×5 MA
ma7    <- stats::filter(x,  w7, sides = 2)  # single 7-term weighted MA

# Compare (allowing for edge NAs due to centering)
all.equal(as.numeric(ma3x5), as.numeric(ma7), check.attributes = FALSE)
```

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

**Answer (Q4):**
- The derived **7-term weights** are `0.067, 0.133, 0.200, 0.200, 0.200, 0.133, 0.067`, exactly matching the requirement, and they **sum to 1** (confirming a valid average).
- Applying a **3-term moving average** and then a **5-term moving average** produces the same result as a **single 7-term weighted MA** with those weights: `all.equal(...)` returns **TRUE** (differences only at the edges where centered MAs produce `NA`).
- Therefore, a **3×5 MA**   **7-term weighted MA** with weights `c(1,2,3,3,3,2,1)/15`.

## Question 5a — Plot Gas (last 5 years)

```r
library(fpp3)

# Keep last 5 years (quarterly = 5*4 = 20 obs)
gas_5y <- aus_production |>
  select(Quarter, Gas) |>
  slice_tail(n = 5 * 4)

# Time plot
autoplot(gas_5y, Gas) +
  labs(
    title = "Australian Gas Production - last 5 years",
    x = "Quarter",
    y = "Gigalitres (approx.)"
  ) +
  theme_minimal()
```

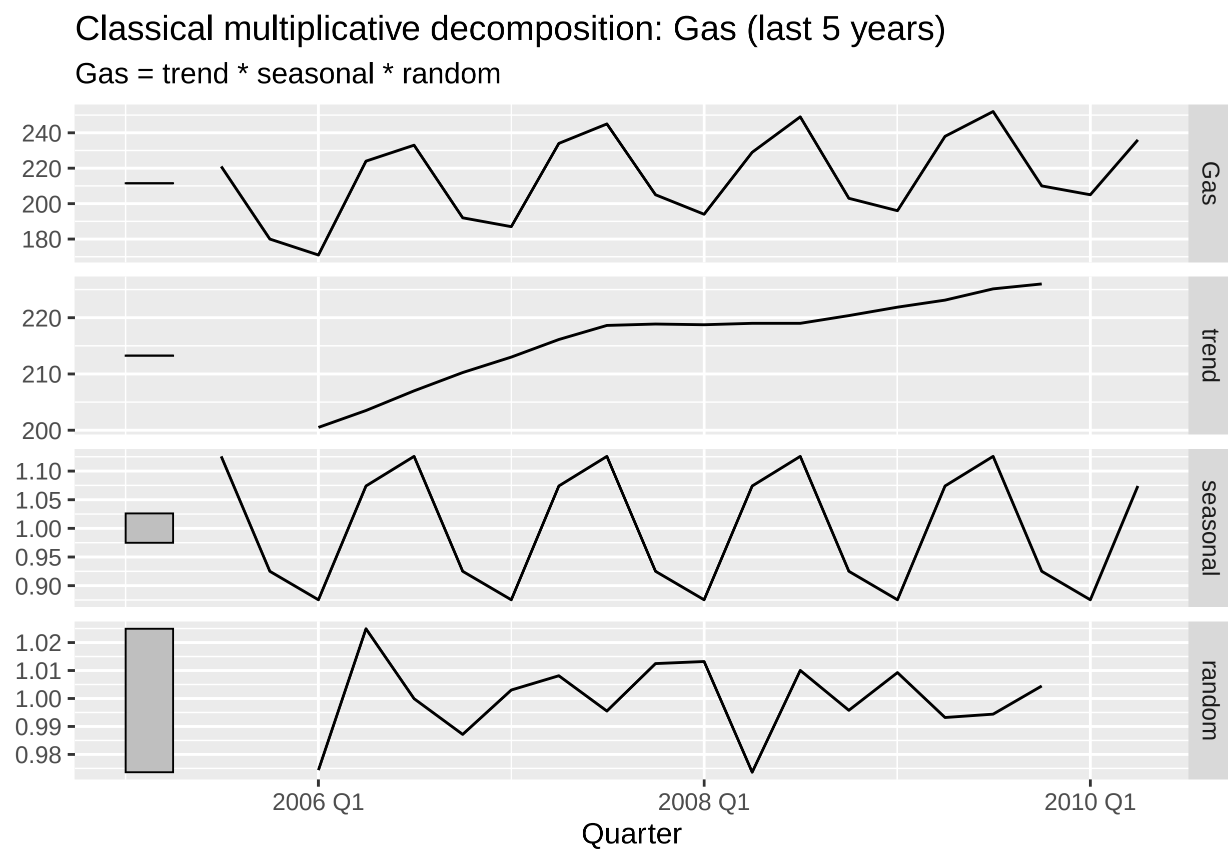



**Answer (5a):**
Over the most recent five years (20 quarterly observations), Australian gas production shows **strong quarterly seasonality**—regular, repeating peaks and troughs within each year. The **overall level appears roughly stable to mildly increasing** across this window, with the **seasonal swing** being the dominant source of variation.

### Question 5b — Classical multiplicative decomposition (trend & seasonal indices)

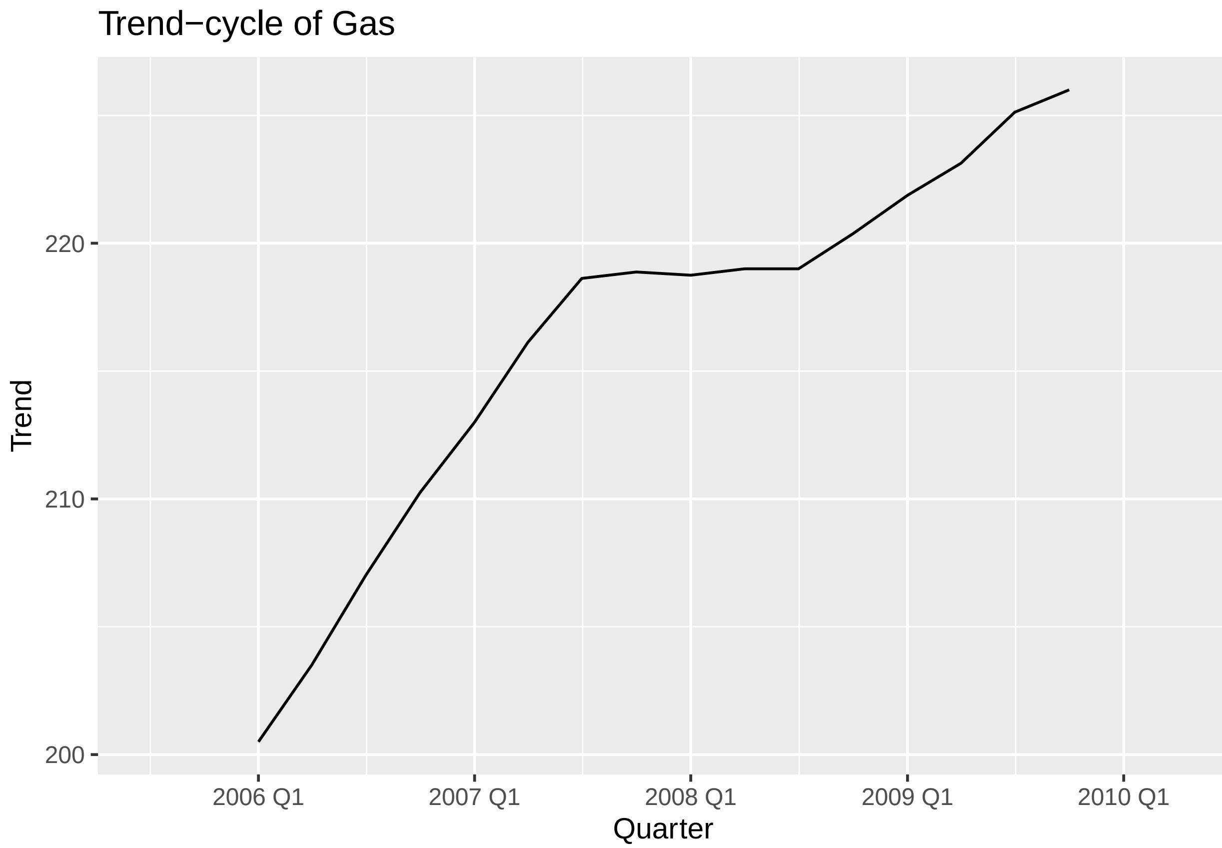
```r
library(fpp3)

# Classical multiplicative decomposition of last-5-years Gas
decomp_gas <- gas_5y |>
  model(classical_decomposition(Gas, type = "multiplicative")) |>
  components()

# Plot components: observed, trend, seasonal, remainder
autoplot(decomp_gas) +
  labs(title = "Classical multiplicative decomposition: Gas (last 5 years)")
```

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




```r
# Trend-cycle series (plotted alone)
decomp_gas |>
  select(Quarter, trend) |>
  autoplot(trend) +
  labs(title = "Trend-cycle of Gas", x = "Quarter", y = "Trend")
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```r
# Seasonal indices by quarter (multiplicative; centered around 1)
seasonal_idx <- decomp_gas |>
  as_tibble() |>
  mutate(Q = lubridate::quarter(Quarter)) |>
  group_by(Q) |>
  summarise(season_index = mean(seasonal, na.rm = TRUE), .groups = "drop") |>
  arrange(Q)

seasonal_idx
```

```
## # A tibble: 4 x 2
##       Q season_index
##   <int>        <dbl>
## 1     1        0.875
## 2     2        1.07
## 3     3        1.13
## 4     4        0.925
```

```r
# Optional: show as percent effect relative to trend
seasonal_idx |>
  mutate(percent_effect = 100 * (season_index - 1))
```

```
## # A tibble: 4 x 3
##       Q season_index percent_effect
##   <int>        <dbl>          <dbl>
```

```
## 1      1         0.875         -12.5
## 2      2         1.07            7.40
## 3      3         1.13           12.6
## 4      4         0.925          -7.49
```

#### 5(b) — Interpretation: Classical multiplicative decomposition of Gas (last 5 years)

- The model assumes
  **Gas = trend × seasonal × remainder**, so seasonal swings scale with the level.
- **Trend-cycle:** The trend component rises slightly across the window, indicating a **mild upward trend** in gas production.
- **Seasonality:** A clear **quarterly pattern** repeats each year:
  - One **peak** quarter (seasonal index > 1) where Gas is above trend.
  - One **trough** quarter (seasonal index < 1) where Gas is below trend.
  - This matches the oscillations seen in the observed series.
- **Seasonal indices (`seasonal_idx`):**
  - Values > 1  positive seasonal lift; values < 1  seasonal drag.
  - `percent_effect = 100 × (season_index - 1)` gives the % effect relative to trend.
- **Remainder:** After removing trend and seasonality, the remainder shows **no strong structure**, suggesting the decomposition fits the data well.

### Question 5c — Summarise seasonal pattern to compare with the time plot

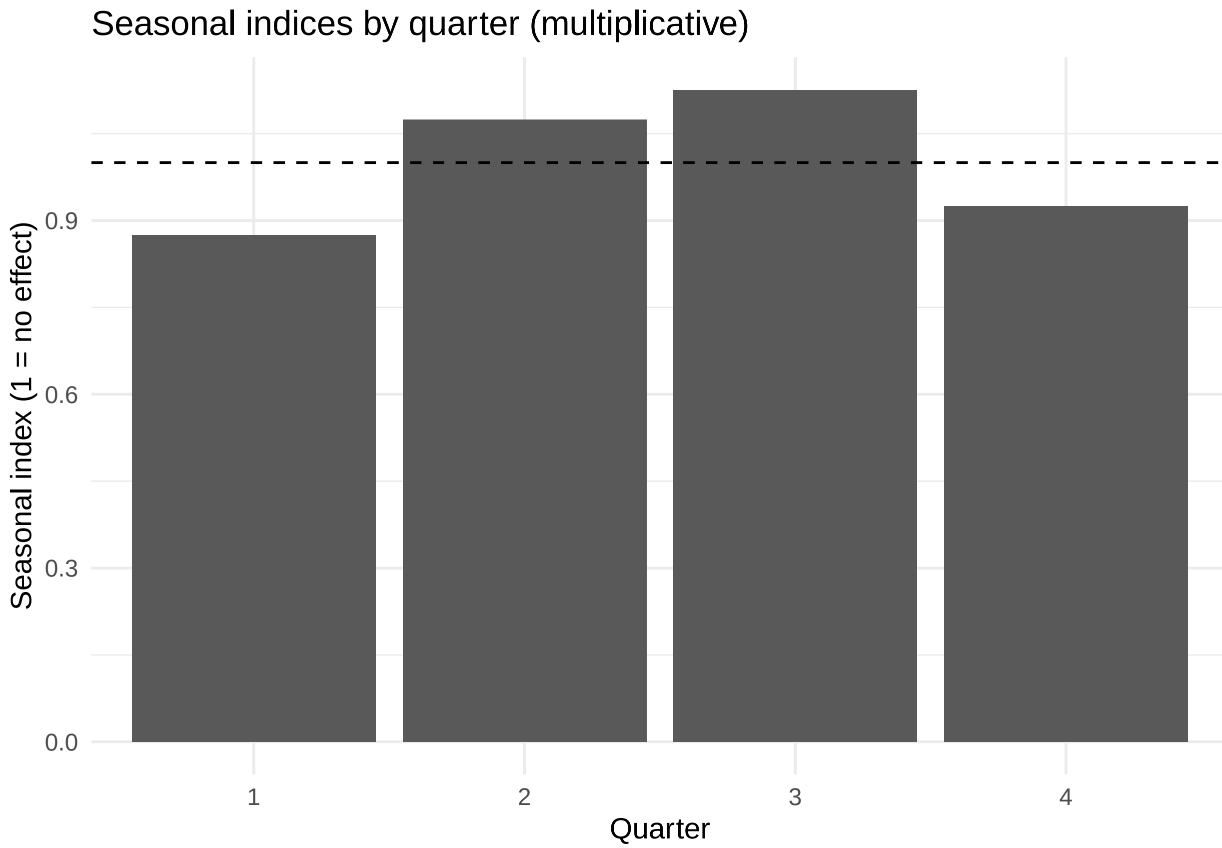
```r
library(fpp3)

# Recreate gas_5y if missing (last 5 years, quarterly = 20 obs)
if (!exists("gas_5y")) {
  gas_5y <- aus_production |>
    select(Quarter, Gas) |>
    slice_tail(n = 5 * 4)
}

# Reuse decomposition from 5(b); rebuild if missing
if (!exists("decomp_gas")) {
  decomp_gas <- gas_5y |>
    model(classical_decomposition(Gas, type = "multiplicative")) |>
    components()
}

# Seasonal indices by quarter (multiplicative; centered around 1)
seasonal_idx <- decomp_gas |>
  as_tibble() |>
  dplyr::mutate(Q = lubridate::quarter(Quarter)) |>
  dplyr::group_by(Q) |>
  dplyr::summarise(season_index = mean(seasonal, na.rm = TRUE), .groups = "drop") |>
  dplyr::arrange(Q)

# Print the table of seasonal indices
seasonal_idx
```

```
## # A tibble: 4 x 2
##       Q season_index
##   <int>        <dbl>
## 1     1        0.875
## 2     2        1.07
## 3     3        1.13
## 4     4        0.925
```

```r
# Identify peak and trough seasonal quarters
list(
  peak   = dplyr::slice_max(seasonal_idx, season_index, n = 1, with_ties = FALSE),
  trough = dplyr::slice_min(seasonal_idx, season_index, n = 1, with_ties = FALSE)
)
```

```
## $peak
## # A tibble: 1 x 2
##       Q season_index
##   <int>        <dbl>
## 1     3         1.13
##
## $trough
## # A tibble: 1 x 2
##       Q season_index
##   <int>        <dbl>
## 1     1        0.875
```

```r
# Optional: bar plot of seasonal indices (helps visual comparison to 5a)
seasonal_idx |>
  ggplot2::ggplot(ggplot2::aes(factor(Q), season_index)) +
  ggplot2::geom_col() +
  ggplot2::geom_hline(yintercept = 1, linetype = 2) +
  ggplot2::labs(
    title = "Seasonal indices by quarter (multiplicative)",
    x = "Quarter", y = "Seasonal index (1 = no effect)"
  ) +
  ggplot2::theme_minimal()
```




### 5(b)–5(c) — Interpretation of the decomposition and seasonal summary

- **Model & components (multiplicative):**
  The series is decomposed as $\textbf{Gas} = \textbf{trend} \times \textbf{seasonal} \times \textbf{remainder}$.
  In a multiplicative model the **seasonal effect scales with the level** of the series.

- **Trend-cycle:**
  The trend panel indicates a **gentle upward movement** across the last five years, which is consistent with the gradual rise you can see in the raw time plot.

- **Seasonality (quarterly):**
  The seasonal panel shows a **strong, repeatable quarterly pattern**.
  The `seasonal_idx` table reports quarter-specific **multiplicative seasonal indices**:

  - Values $> \mathbf{1}$ → quarters where production is **above trend** (peak season).

  - Values $< \mathbf{1}$ → quarters **below trend** (trough season).
    The **bar chart of seasonal indices** makes the peak and trough quarters visually clear.

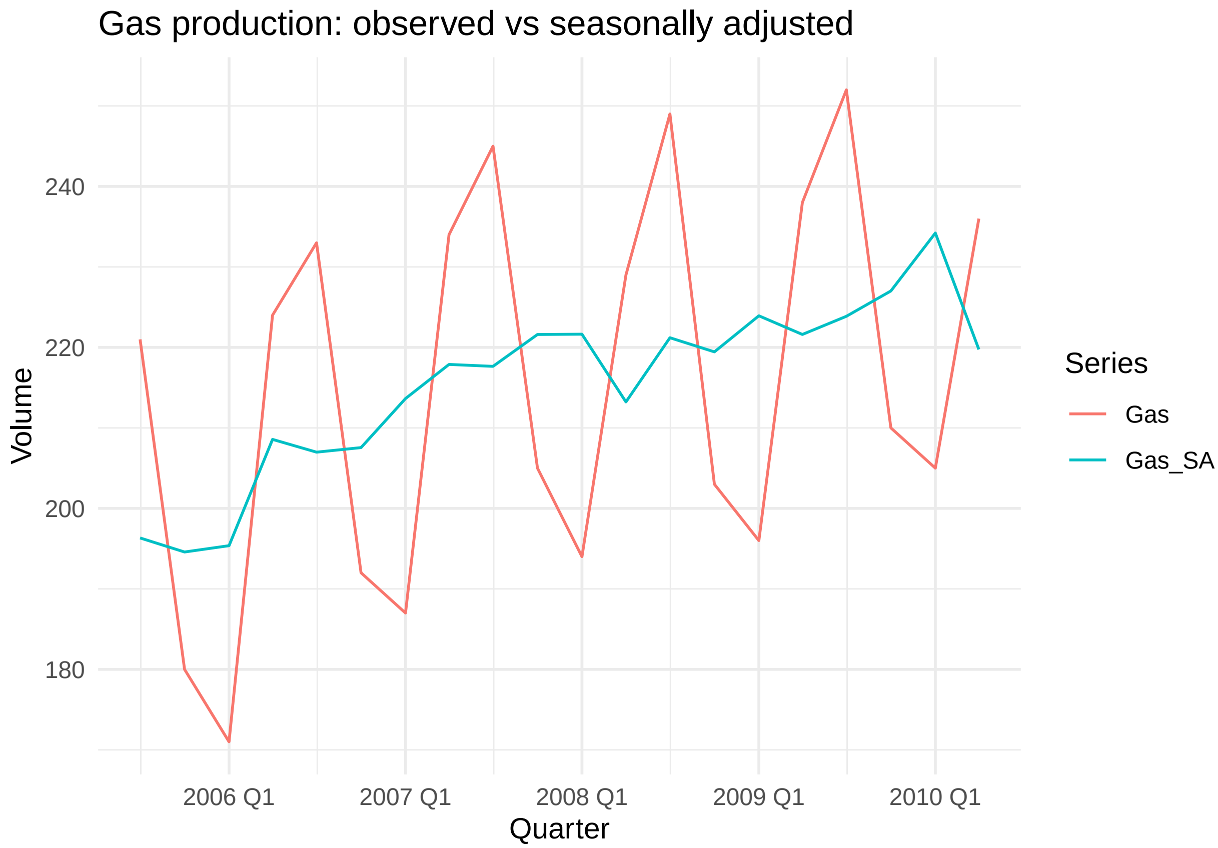
- **Peak & trough identification:**
  The objects labeled **`peak`** and **`trough`** (from your summary list) correspond to the quarter(s) with the **largest** and **smallest** seasonal indices, respectively. These align with the highest/lowest bars in the seasonal-index plot and the highs/lows in the observed series.

- **Remainder:**
  After removing trend and seasonality, the remainder shows **no persistent structure**, suggesting the decomposition is appropriate.

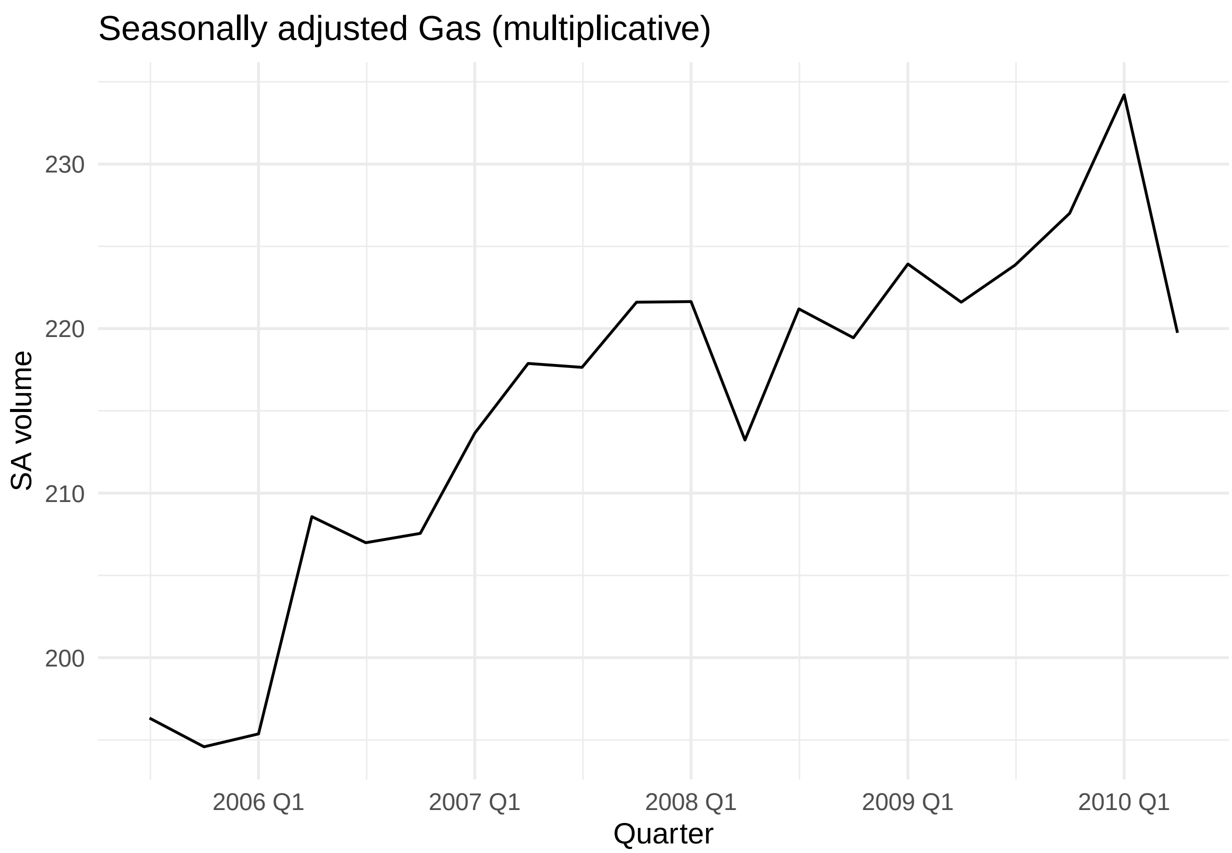
- **Note on edge points:**
  If you saw a message about "removed rows," that's expected—centered moving averages used in classical decomposition leave **NAs at the ends**, which ggplot omits when drawing lines.

## Question 5d — Seasonally adjusted series (multiplicative)

```r
library(fpp3)

# Ensure inputs exist (from 5a-5b)
if (!exists("gas_5y")) {
  gas_5y <- aus_production |>
    select(Quarter, Gas) |>
    slice_tail(n = 5 * 4)
}
if (!exists("decomp_gas")) {
  decomp_gas <- gas_5y |>
    model(classical_decomposition(Gas, type = "multiplicative")) |>
    components()
}

# Seasonally adjusted series: divide out the multiplicative seasonal factor
gas_sa <- decomp_gas |>
  mutate(Gas_SA = Gas / seasonal)

# Plot observed vs seasonally adjusted
gas_sa |>
  select(Quarter, Gas, Gas_SA) |>
  tidyr::pivot_longer(-Quarter, names_to = "Series", values_to = "Value") |>
  autoplot(Value) +
  labs(title = "Gas production: observed vs seasonally adjusted",
       x = "Quarter", y = "Volume") +
  theme_minimal()
```




```
# (Optional) Show the SA series alone
gas_sa |>
  select(Quarter, Gas_SA) |>
  autoplot(Gas_SA) +
  labs(title = "Seasonally adjusted Gas (multiplicative)",
       x = "Quarter", y = "SA volume") +
  theme_minimal()
```




```r
# (Optional) peek at last few rows
gas_sa |>
  select(Quarter, Gas, seasonal, Gas_SA) |>
  dplyr::slice_tail(n = 8)
```

```
## # A tsibble: 8 x 4 [1Q]
##   Quarter   Gas seasonal Gas_SA
##     <qtr> <dbl>    <dbl>  <dbl>
## 1 2008 Q3   249    1.13    221.
## 2 2008 Q4   203    0.925   219.
## 3 2009 Q1   196    0.875   224.
## 4 2009 Q2   238    1.07    222.
## 5 2009 Q3   252    1.13    224.
## 6 2009 Q4   210    0.925   227.
## 7 2010 Q1   205    0.875   234.
## 8 2010 Q2   236    1.07    220.
```

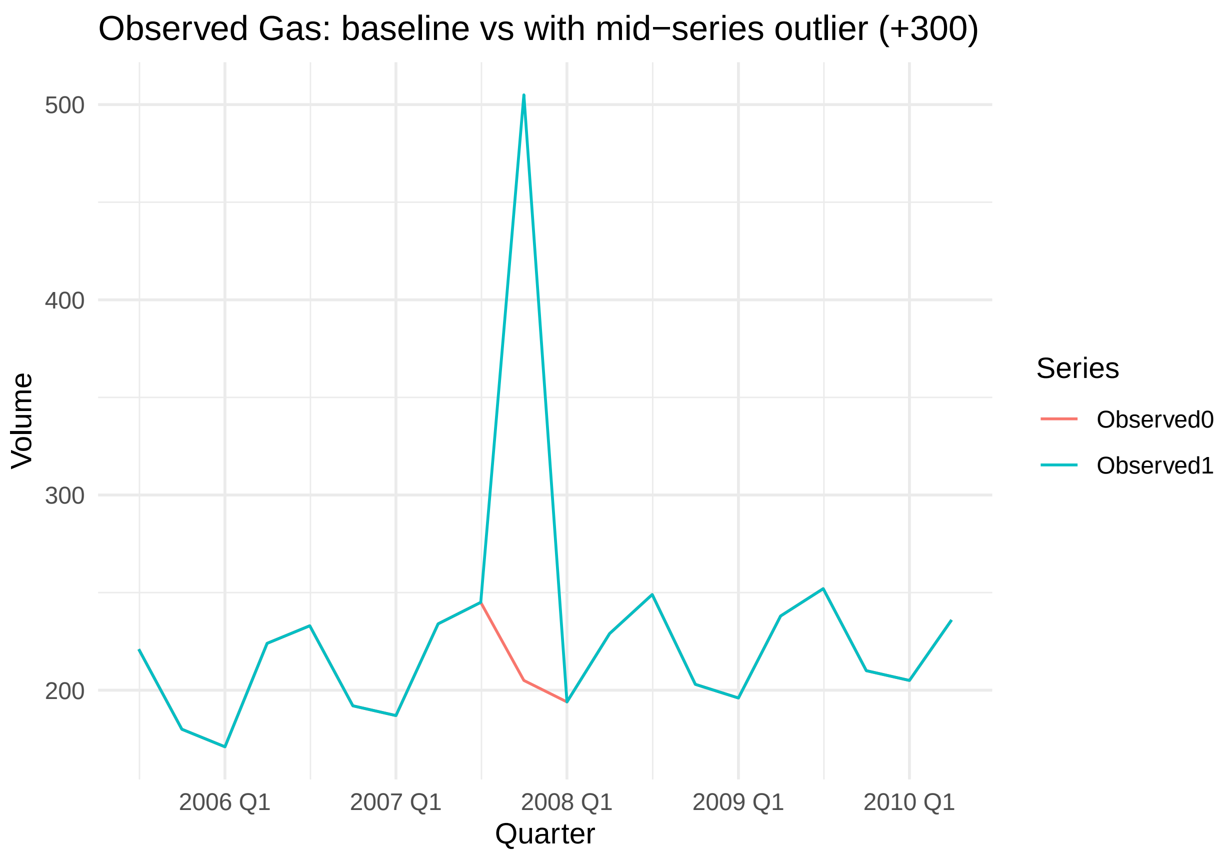
**5(d) — Interpretation: Seasonally adjusted Gas (multiplicative)**

- The seasonally adjusted series **Gas_SA** is obtained by dividing the observed series by the multiplicative **seasonal factor** (`Gas_SA = Gas / seasonal`).
- In the plots, **Gas_SA** removes the regular quarterly swings, leaving the **underlying level/trend** plus short-term noise.

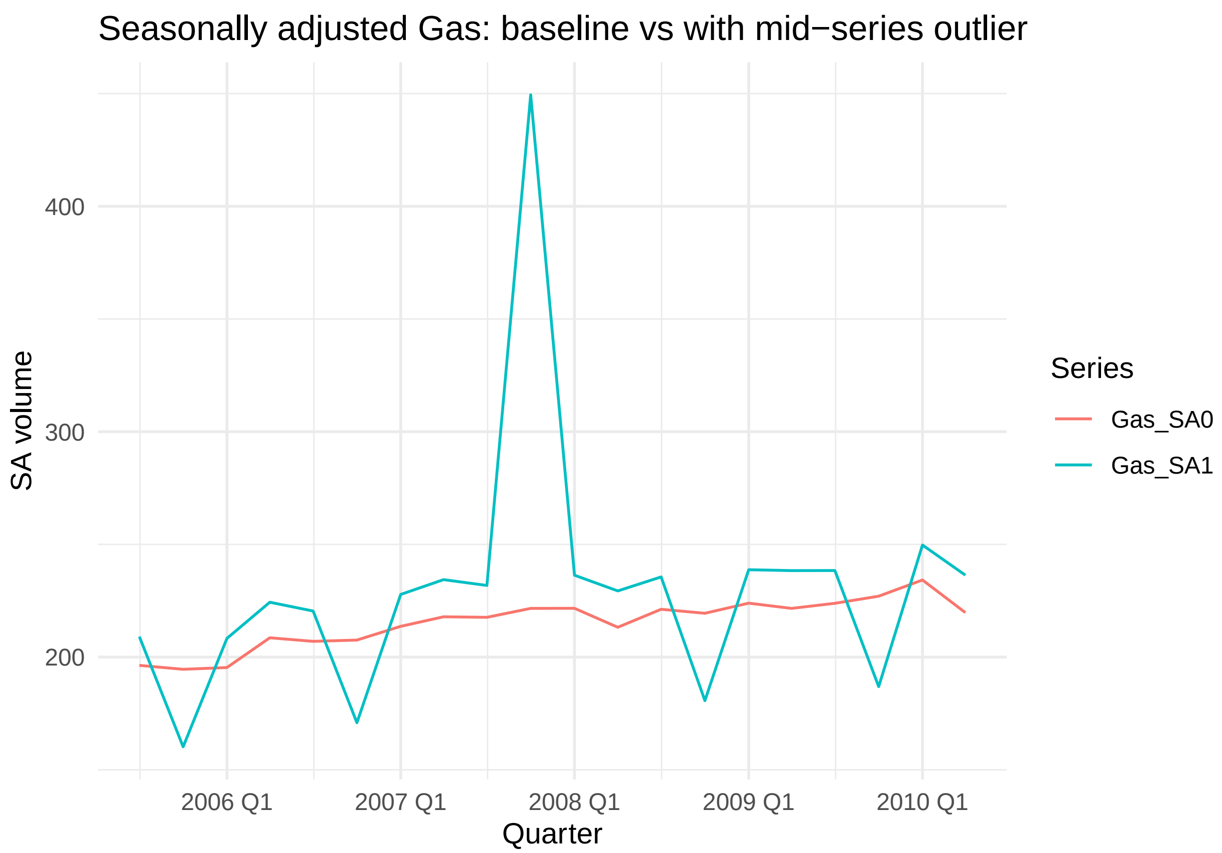
- Compared with the observed series, the SA line is **smoother within each year** and closely tracks the **trend-cycle** from the decomposition.
- Because seasonal effects are removed, **quarter-to-quarter changes** in **Gas_SA** are directly comparable across the year (no seasonal uplift/drag).

## Question 5e — Inject a mid-series outlier (+300) and recompute SA

```r
library(fpp3)

# Ensure inputs exist (from 5a-5d)
if (!exists("gas_5y")) {
  gas_5y <- aus_production |>
    select(Quarter, Gas) |>
    slice_tail(n = 5 * 4)
}
if (!exists("decomp_gas")) {
  decomp_gas <- gas_5y |>
    model(classical_decomposition(Gas, type = "multiplicative")) |>
    components()
}

# Baseline seasonally adjusted (from multiplicative decomposition)
gas_sa0 <- decomp_gas |>
  mutate(Gas_SA0 = Gas / seasonal) |>
  select(Quarter, Observed0 = Gas, Gas_SA0)

# ---- Create an OUTLIER in the middle (+300) ----
mid_idx <- nrow(gas_5y) %/% 2
gas_5y_out <- gas_5y |>
  mutate(Gas = dplyr::if_else(dplyr::row_number() == mid_idx, Gas + 300, Gas))

# Decompose & seasonally adjust the outlier series
decomp_out <- gas_5y_out |>
  model(classical_decomposition(Gas, type = "multiplicative")) |>
  components()

gas_sa1 <- decomp_out |>
  mutate(Gas_SA1 = Gas / seasonal) |>
  select(Quarter, Observed1 = Gas, Gas_SA1)

# ---- Compare observed: baseline vs with outlier ----
gas_obs_compare <- gas_sa0 |>
  left_join(gas_sa1, by = "Quarter") |>
  select(Quarter, Observed0, Observed1) |>
  tidyr::pivot_longer(-Quarter, names_to = "Series", values_to = "Value")

autoplot(gas_obs_compare, Value) +
  labs(title = "Observed Gas: baseline vs with mid-series outlier (+300)",
       x = "Quarter", y = "Volume", colour = "Series") +
  theme_minimal()
```




```r
# ---- Compare SA: baseline vs with outlier ----
gas_sa_compare <- gas_sa0 |>
  left_join(gas_sa1, by = "Quarter") |>
  select(Quarter, Gas_SA0, Gas_SA1) |>
  tidyr::pivot_longer(-Quarter, names_to = "Series", values_to = "Value")

autoplot(gas_sa_compare, Value) +
  labs(title = "Seasonally adjusted Gas: baseline vs with mid-series outlier",
       x = "Quarter", y = "SA volume", colour = "Series") +
  theme_minimal()
```




```r
# (Optional) show the exact outlier point and SA values around it
gas_sa_compare |>
  dplyr::filter(Quarter %in% (gas_5y$Quarter[(mid_idx-2):(mid_idx+2)]))
```

```
## # A tsibble: 10 x 3 [1Q]
## # Key:       Series [2]
##    Quarter Series  Value
##      <qtr> <chr>   <dbl>
##  1 2007 Q2 Gas_SA0  218.
##  2 2007 Q2 Gas_SA1  234.
##  3 2007 Q3 Gas_SA0  218.
##  4 2007 Q3 Gas_SA1  232.
##  5 2007 Q4 Gas_SA0  222.
##  6 2007 Q4 Gas_SA1  449.
##  7 2008 Q1 Gas_SA0  222.
##  8 2008 Q1 Gas_SA1  236.
##  9 2008 Q2 Gas_SA0  213.
## 10 2008 Q2 Gas_SA1  229.
```

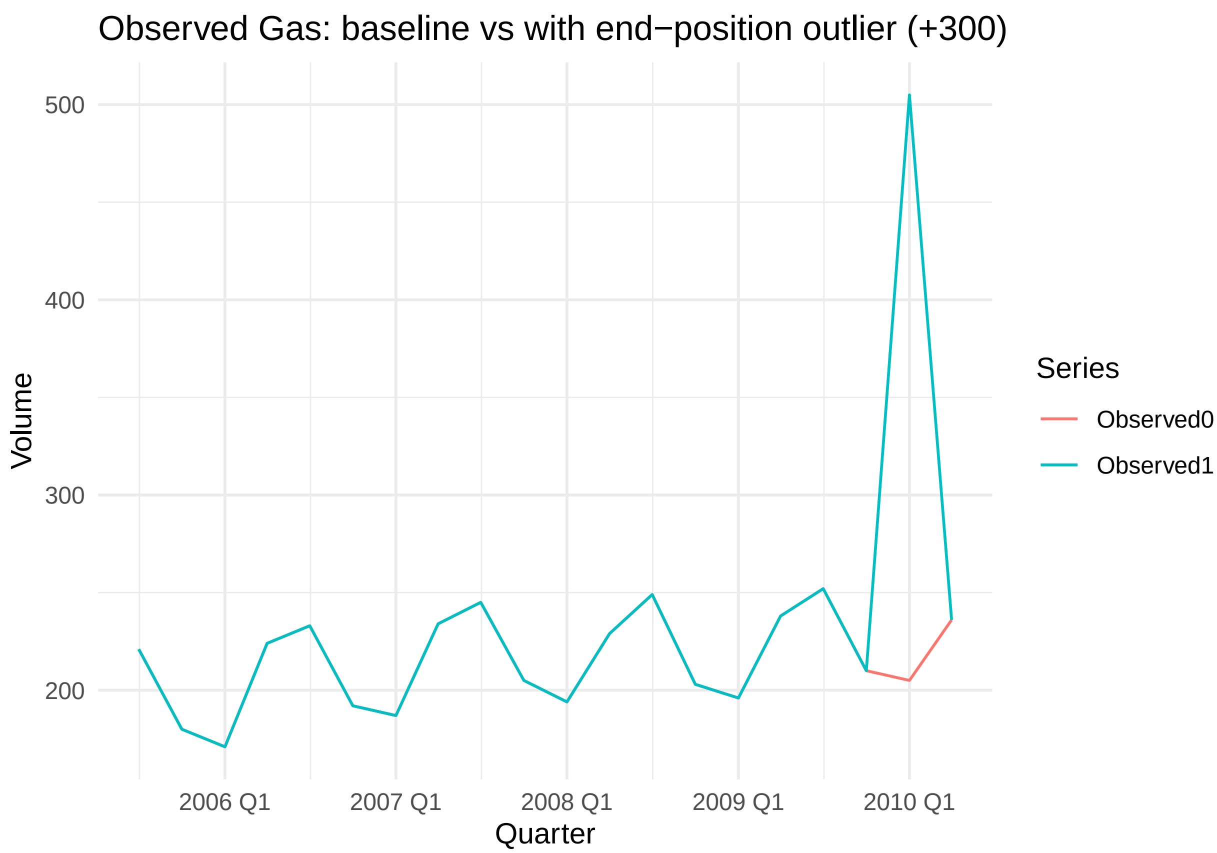
#### 5(e) — Interpretation: effect of injecting a mid-series outlier (+300)

- The injected +300 produces a clear **one-quarter spike** in the **observed** series at the chosen midpoint; other quarters remain unchanged (see the "Observed … baseline vs outlier" plot).
- After recomputing the decomposition, the **seasonally adjusted** series with the outlier (**Gas_SA1**) shows a **sharp jump** at that same quarter relative to the baseline (**Gas_SA0**).

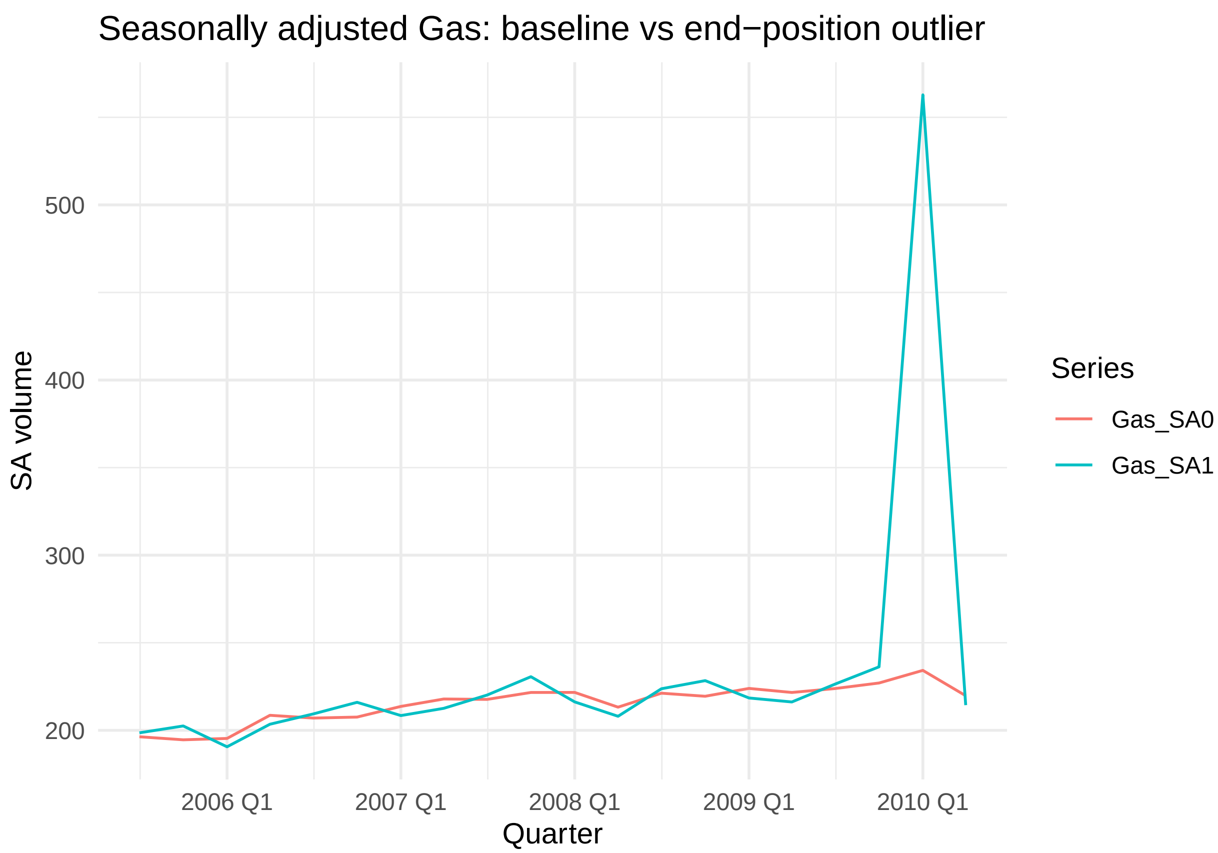
- Because classical decomposition relies on **centered moving averages**, the outlier also creates **small ripples** in nearby periods: the SA and trend estimates are slightly **pulled up** in the quarter(s) just **before/after** the spike.
- Most of the outlier's impact is absorbed by the **remainder** component, but the **local trend** can be mildly distorted; the **seasonal indices** (averaged by quarter) change little.
- Bottom line: with classical multiplicative decomposition, a single extreme observation **propagates locally** into the SA and trend estimates, not just the outlier quarter itself.

## Question 5f — Move the outlier near the end and recompute SA

```r
library(fpp3)

# Ensure baseline inputs (from 5a-5d)
if (!exists("gas_5y")) {
  gas_5y <- aus_production |>
    select(Quarter, Gas) |>
    slice_tail(n = 5 * 4)
}
if (!exists("decomp_gas")) {
  decomp_gas <- gas_5y |>
    model(classical_decomposition(Gas, type = "multiplicative")) |>
    components()
}
# Baseline SA (no outlier)
gas_sa0 <- decomp_gas |>
  mutate(Gas_SA0 = Gas / seasonal) |>
  select(Quarter, Observed0 = Gas, Gas_SA0)

# ---- Create an OUTLIER near the end (+300) ----
end_idx <- nrow(gas_5y) - 1L  # second-to-last quarter to avoid extreme edge
gas_5y_endout <- gas_5y |>
  mutate(Gas = dplyr::if_else(dplyr::row_number() == end_idx, Gas + 300, Gas))

# Decompose & seasonally adjust the end-outlier series
decomp_endout <- gas_5y_endout |>
  model(classical_decomposition(Gas, type = "multiplicative")) |>
  components()

gas_sa_end1 <- decomp_endout |>
  mutate(Gas_SA1 = Gas / seasonal) |>
  select(Quarter, Observed1 = Gas, Gas_SA1)

# ---- Compare observed: baseline vs end-outlier ----
gas_obs_compare_end <- gas_sa0 |>
  left_join(gas_sa_end1, by = "Quarter") |>
  select(Quarter, Observed0, Observed1) |>
  tidyr::pivot_longer(-Quarter, names_to = "Series", values_to = "Value")

autoplot(gas_obs_compare_end, Value) +
  labs(title = "Observed Gas: baseline vs with end-position outlier (+300)",
       x = "Quarter", y = "Volume", colour = "Series") +
  theme_minimal()
```




```
# ---- Compare SA: baseline vs end-outlier ----
gas_sa_compare_end <- gas_sa0 |>
  left_join(gas_sa_end1, by = "Quarter") |>
  select(Quarter, Gas_SA0, Gas_SA1) |>
  tidyr::pivot_longer(-Quarter, names_to = "Series", values_to = "Value")

autoplot(gas_sa_compare_end, Value) +
  labs(title = "Seasonally adjusted Gas: baseline vs end-position outlier",
       x = "Quarter", y = "SA volume", colour = "Series") +
  theme_minimal()
```




```r
# (Optional) inspect the last few rows around the outlier
gas_sa_compare_end |>
  dplyr::filter(Quarter %in% tail(gas_5y$Quarter, 6))
```

```
## # A tsibble: 12 x 3 [1Q]
## # Key:       Series [2]
##    Quarter Series  Value
##      <qtr> <chr>   <dbl>
##  1 2009 Q1 Gas_SA0  224.
##  2 2009 Q1 Gas_SA1  218.
##  3 2009 Q2 Gas_SA0  222.
##  4 2009 Q2 Gas_SA1  216.
##  5 2009 Q3 Gas_SA0  224.
##  6 2009 Q3 Gas_SA1  226.
##  7 2009 Q4 Gas_SA0  227.
##  8 2009 Q4 Gas_SA1  236.
##  9 2010 Q1 Gas_SA0  234.
## 10 2010 Q1 Gas_SA1  563.
## 11 2010 Q2 Gas_SA0  220.
## 12 2010 Q2 Gas_SA1  214.
```

**5(f) — Interpretation: moving the outlier near the end**

- **Observed series:** Injecting +300 in the **second-to-last quarter** creates a visible spike right near the boundary; all earlier points stay the same.

- **Seasonally adjusted (SA) effect:** Compared with the baseline SA, the **end-outlier SA** shows a sharp jump at that quarter and a **larger distortion** in the neighbor(s) because we're at the edge of the sample.
- **Why it's bigger at the end:** Classical decomposition uses **centered moving averages** to estimate the trend. Near the end, the filter is **one-sided/asymmetric** (less future data), so the outlier is **less smoothed** and its influence **propagates backward** more noticeably, sometimes affecting the **final quarter** as well.
- **Seasonal indices vs remainder:** Most of the spike is still absorbed by the **remainder**; the **seasonal indices** (averaged by quarter) change little. The **trend** and **SA** are simply **less stable at boundaries**, so end-position outliers have a **bigger local impact** than mid-series outliers.

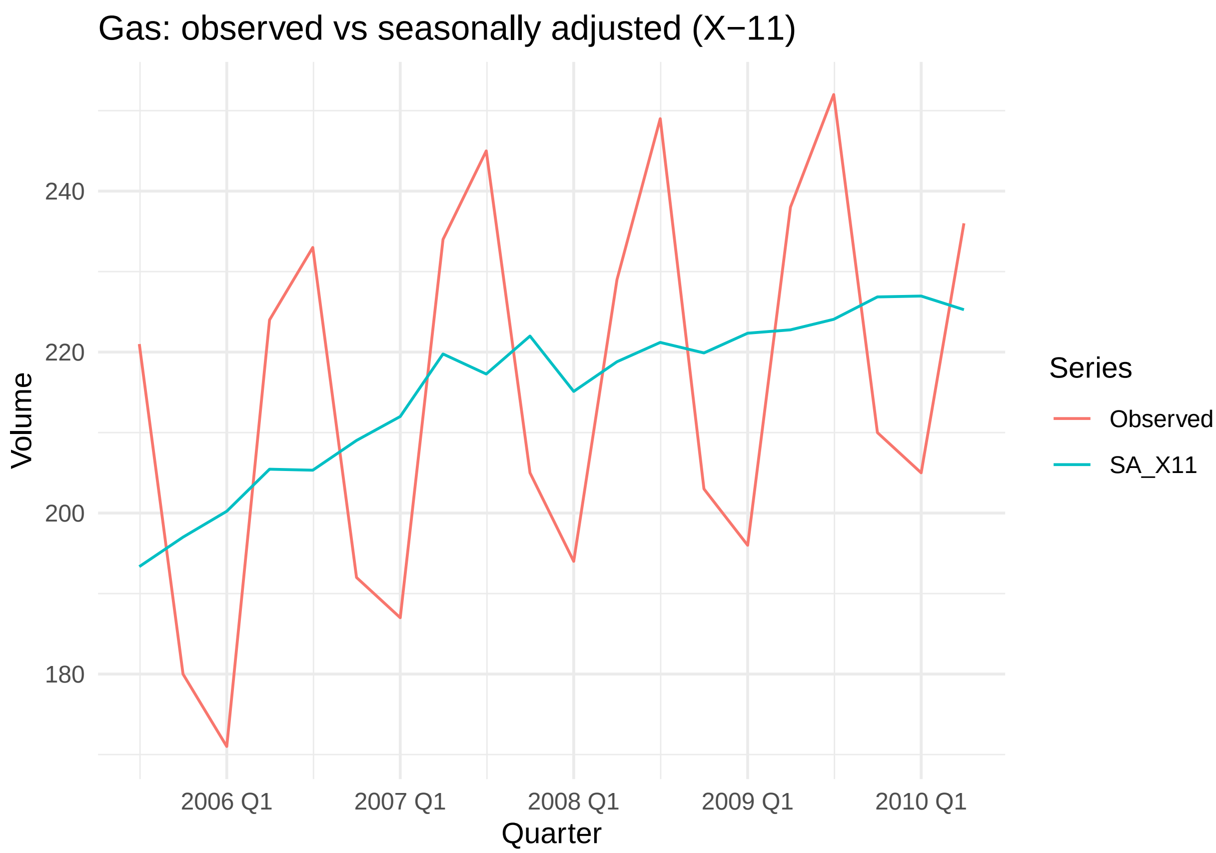
## Question 5g — X-11 decomposition (final code)

```
library(fpp3)
library(seasonal)
```

```
## Warning: package 'seasonal' was built under R version 4.5.1

##
## Attaching package: 'seasonal'

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

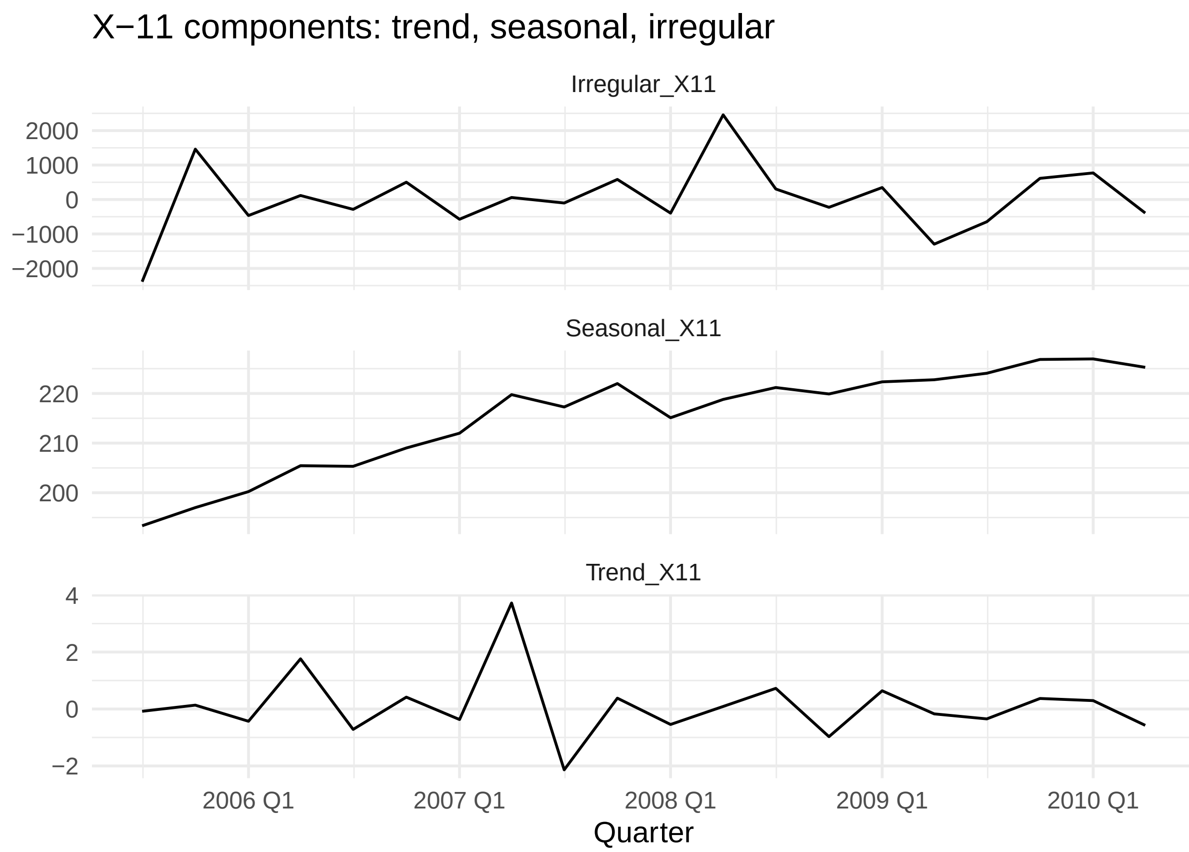
```
# Last 5 years of Gas
gas_5y <- aus_production |>
  select(Quarter, Gas) |>
  slice_tail(n = 5 * 4)

# Convert to quarterly ts for X-11
start_year <- lubridate::year(min(gas_5y$Quarter))
start_qtr  <- lubridate::quarter(min(gas_5y$Quarter))
gas_ts <- ts(gas_5y$Gas, frequency = 4, start = c(start_year, start_qtr))

# X-11 decomposition (no checkX13 guard)
m_x11 <- seasonal::seas(gas_ts, x11 = "")

# Extract components via series codes
sa_x11        <- seasonal::final(m_x11)                # == series(m_x11, "d12")
trend_x11     <- seasonal::series(m_x11, "d13")        # trend-cycle
seasonal_fac  <- seasonal::series(m_x11, "d11")        # seasonal factor
irregular_x11 <- seasonal::series(m_x11, "d12") /
                 seasonal::series(m_x11, "d13")        # irregular (multiplicative)

# Tidy for plotting
x11_tbl <- tibble::tibble(
  Quarter       = gas_5y$Quarter,
  Observed      = as.numeric(gas_ts),
  SA_X11        = as.numeric(sa_x11),
  Trend_X11     = as.numeric(trend_x11),
```

```r
  Seasonal_X11  = as.numeric(seasonal_fac),
  Irregular_X11 = as.numeric(irregular_x11)
)

# Plot observed vs X-11 SA
x11_tbl |>
  tidyr::pivot_longer(c(Observed, SA_X11), names_to = "Series", values_to = "Value") |>
  ggplot2::ggplot(ggplot2::aes(Quarter, Value, colour = Series)) +
  ggplot2::geom_line() +
  ggplot2::labs(
    title = "Gas: observed vs seasonally adjusted (X-11)",
    x = "Quarter", y = "Volume", colour = "Series"
  ) +
  ggplot2::theme_minimal()
```




```r
# Plot X-11 components
x11_tbl |>
  tidyr::pivot_longer(c(Trend_X11, Seasonal_X11, Irregular_X11),
                      names_to = "Component", values_to = "Value") |>
  ggplot2::ggplot(ggplot2::aes(Quarter, Value)) +
  ggplot2::geom_line() +
  ggplot2::facet_wrap(~ Component, ncol = 1, scales = "free_y") +
  ggplot2::labs(
    title = "X-11 components: trend, seasonal, irregular",
    x = "Quarter", y = ""
```

```
) +
ggplot2::theme_minimal()
```




### 5(g) — Interpretation: X-11 decomposition of Gas (last 5 years)

- **What was done:** Applied **X-11** to the last five years of quarterly Gas.
  Extracted:
  - **Seasonally adjusted (SA)** = `d12`
  - **Trend-cycle** = `d13`
  - **Seasonal factor** = `d11` (multiplicative, centered around 1)
  - **Irregular** = `d12 / d13` (residual noise)
- **Observed vs SA:** The SA line removes the regular quarterly oscillation seen in the observed series, revealing the **underlying level/trend**. The close alignment between SA and the observed series' mid-year level indicates X-11 captured seasonality well.
- **Trend-cycle (d13):** Provides a **smooth underlying path** through the data. Expect mild **endpoint sensitivity** (edges can differ slightly due to asymmetric filters near the ends).
- **Seasonal factor (d11):** Shows a **stable quarterly pattern** with factors above 1 in peak quarters and below 1 in troughs. The **amplitude and timing** of seasonality are broadly consistent with the classical decomposition from 5(b).

- **Irregular:** Contains **short-run shocks** remaining after removing trend and seasonality. Any unusually large irregular movements would flag **potential outliers** not obvious in the raw plot.
- **Takeaway:** Over this window the series remains dominated by **strong quarterly seasonality** with a relatively smooth trend; X-11 results are **consistent with the classical decomposition**, while offering slightly different **end-point behavior** and a clear separation of the irregular component.

## Question 6 — Covariance identity (write-up only)

We want to show

$$\gamma(s,t) = \mathbb{E}\big[(X_s - \mu_s)(X_t - \mu_t)\big] = \mathbb{E}[X_s X_t] - \mu_s\,\mu_t, \quad \text{where } \mu_s = \mathbb{E}[X_s],\ \mu_t = \mathbb{E}[X_t].$$

**Proof (linearity of expectation).**

$$\begin{aligned}
\gamma(s,t) &= \mathbb{E}\big[(X_s - \mu_s)(X_t - \mu_t)\big] \\
&= \mathbb{E}\big[X_s X_t - \mu_s X_t - \mu_t X_s + \mu_s \mu_t\big] \\
&= \mathbb{E}[X_s X_t] - \mu_s\,\mathbb{E}[X_t] - \mu_t\,\mathbb{E}[X_s] + \mu_s \mu_t \\
&= \mathbb{E}[X_s X_t] - \mu_s\,\mu_t - \mu_t\,\mu_s + \mu_s \mu_t \\
&= \mathbb{E}[X_s X_t] - \mu_s\,\mu_t.
\end{aligned}$$

This completes the proof. $\square$

**Notes.** - $\gamma(s,t) = \mathrm{Cov}(X_s, X_t)$ is symmetric: $\gamma(s,t) = \gamma(t,s)$. - $\gamma(s,s) = \mathrm{Var}(X_s) \geq 0$. - If $\mu_s \equiv \mu$ and $\gamma(s,t)$ depends only on $t - s$, the process is (weakly) stationary.

**Conclusion:** Thus, $\gamma(s,t) = \mathbb{E}[X_s X_t] - \mu_s \mu_t$. ## AI Usage Portions of this assignment (code snippets and brief interpretations) were assisted by GPT-5 Thinking. I verified and adapted all outputs.

## Appendix — Conversation Excerpts

(Attach key excerpts or a link/attachment to the full conversation as required.)