Time Series Analysis

Julius Hai

2025-09-17

```
# ------------------------------------------------
# Packages - install only the first time you use a new PC
# ------------------------------------------------
# (Uncomment the lines you need, run once, then comment them again)

# install.packages(c("tidyverse","lubridate","tsibble","fable","feasts"))
# install.packages("remotes")                    # needed only for GitHub in
stall
# remotes::install_github("tidyverts/tsibbledata")  # gets the development ve
rsion (contains all data)

# ------------------------------------------------
# Load the libraries
# ------------------------------------------------
library(tidyverse)   # dplyr, ggplot2, etc.
```

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

## Warning: package 'ggplot2' was built under R version 4.5.1

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

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

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

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

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

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

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.
0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   4.0.0     ✔ tibble    3.2.1
## ✔ lubridate 1.9.4     ✔ tidyr     1.3.1
## ✔ purrr     1.0.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```r
library(lubridate)   # year(), month()
library(tsibble)     # tidy time-series objects
```

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

```
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
##
## Attaching package: 'tsibble'
##
## The following object is masked from 'package:lubridate':
##
##     interval
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, union
```

```r
library(fable)       # modelling / forecasting
```

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

```
## Loading required package: fabletools
```

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

```r
library(feasts)      # residual diagnostics
```

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

```r
library(tsibbledata) # data sets (Aus production, global economy, etc.)
```

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

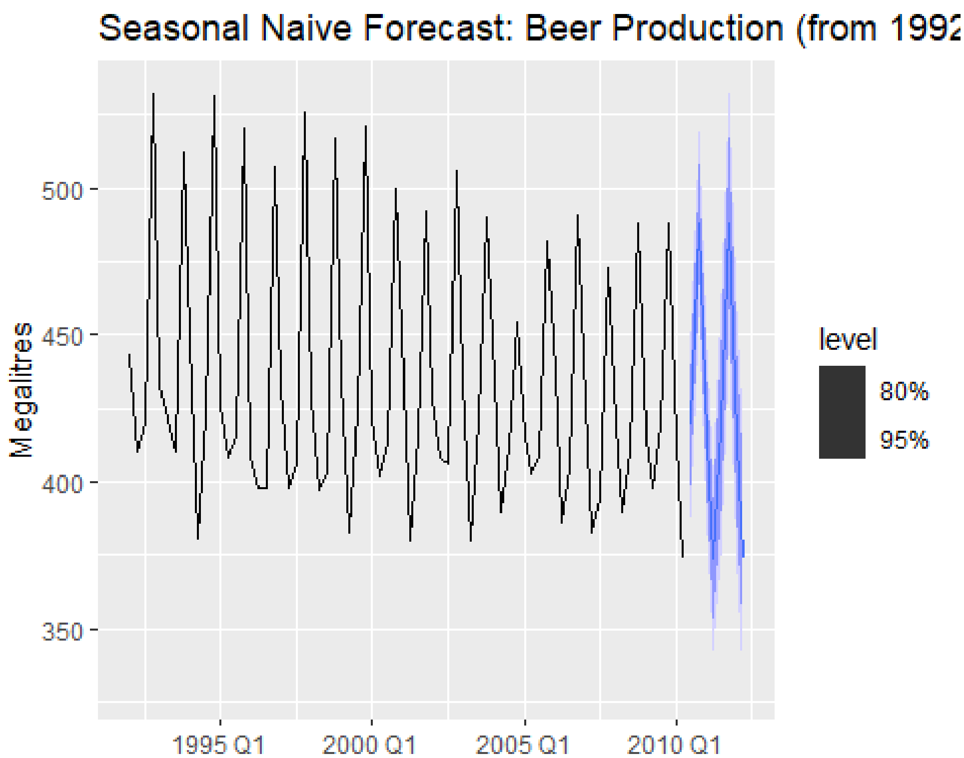
```r
# -------------------------------------------------
# Pull the data sets we will need for Question 1
# -------------------------------------------------
data("aus_production",   package = "tsibbledata")   # Beer, Bricks, etc.
data("global_economy",   package = "tsibbledata")   # Australian Exports
```

Q1

```r
# ---- Silence console messages/warnings (especially for R Markdown) ----
# If you're knitting, also put in your setup chunk:
# knitr::opts_chunk$set(message = FALSE, warning = FALSE)
options(warn = -1)

# ---- Packages ----
library(fpp3)
```

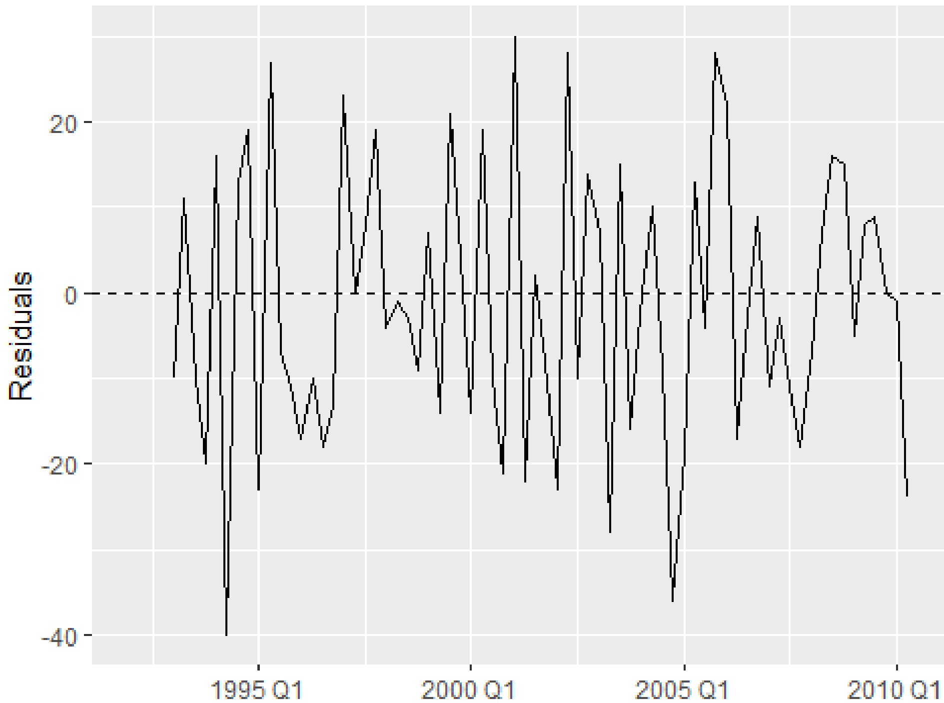
```r
library(dplyr)
library(tidyr)
library(ggplot2)

# ---------- Helper: residual diagnostic plots without icons ----------
diag_plots <- function(aug_df, time_var, title_prefix = "") {
  # aug_df should contain columns: .resid (or .innov), .fitted, and time_var
  rcol <- if (".innov" %in% names(aug_df)) ".innov" else ".resid"

  # Time plot of residuals
  p1 <- ggplot(aug_df, aes(x = {{ time_var }}, y = .data[[rcol]])) +
    geom_line(na.rm = TRUE) +
    geom_hline(yintercept = 0, linetype = 2) +
    labs(title = paste0(title_prefix, "Residuals over Time"),
         x = NULL, y = "Residuals")

  # ACF of residuals
  p2 <- aug_df |>
    select({{ time_var }}, r = all_of(rcol)) |>
    as_tsibble(index = {{ time_var }}) |>
    ACF(r) |>
    autoplot() +
    labs(title = paste0(title_prefix, "Residual ACF"))

  # Histogram
  p3 <- ggplot(aug_df, aes(x = .data[[rcol]])) +
    geom_histogram(bins = 30, na.rm = TRUE) +
    labs(title = paste0(title_prefix, "Residual Histogram"),
         x = "Residuals", y = "Count")

  # Residuals vs fitted
  p4 <- ggplot(aug_df, aes(x = .fitted, y = .data[[rcol]])) +
    geom_point(na.rm = TRUE, alpha = 0.7) +
    geom_hline(yintercept = 0, linetype = 2) +
    labs(title = paste0(title_prefix, "Residuals vs Fitted"),
         x = "Fitted", y = "Residuals")

  list(time = p1, acf = p2, hist = p3, rvf = p4)
}

# ========================
# Q1 — Benchmark forecasts
# ========================

# ---- 1) Beer Production (Quarterly; SNAIVE) ----
recent_beer <- aus_production |>
  filter(year(Quarter) >= 1992) |>
  drop_na(Beer)
```

```r
fit_beer <- recent_beer |>
  model(SNAIVE(Beer))

# Forecast (no console warnings)
fit_beer |>
  forecast() |>
  autoplot(recent_beer) +
  labs(title = "Seasonal Naive Forecast: Beer Production (from 1992)",
       y = "Megalitres", x = NULL)
```

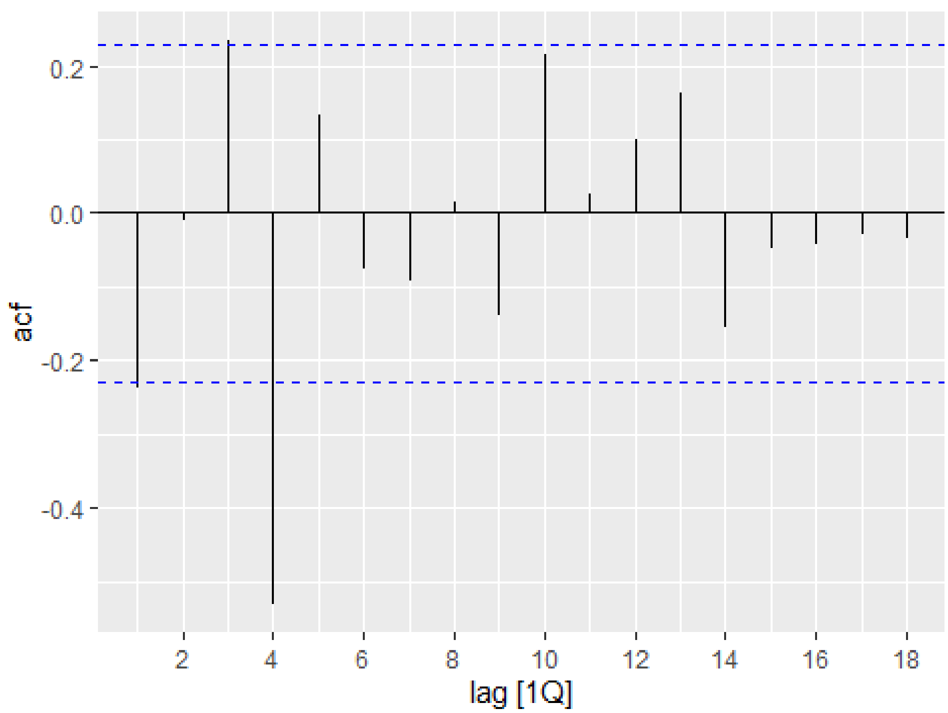


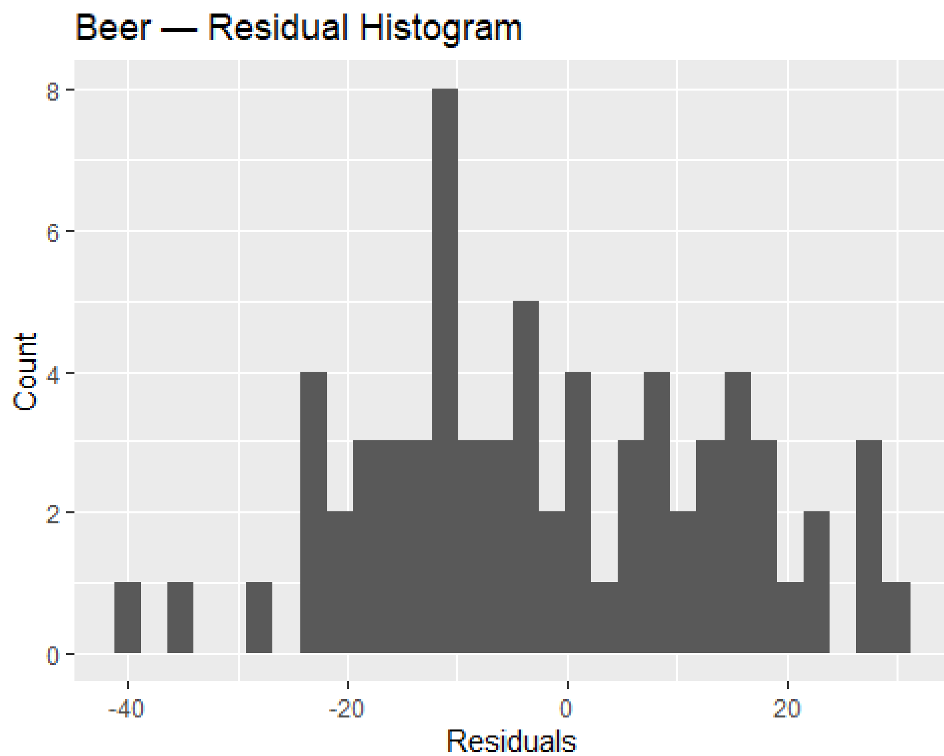

```r
# Residual diagnostics without icons
aug_beer <- augment(fit_beer) |> filter(.model == "SNAIVE(Beer)")
beer_plots <- diag_plots(aug_beer, Quarter, "Beer – ")
beer_plots$time; beer_plots$acf; beer_plots$hist; beer_plots$rvf
```

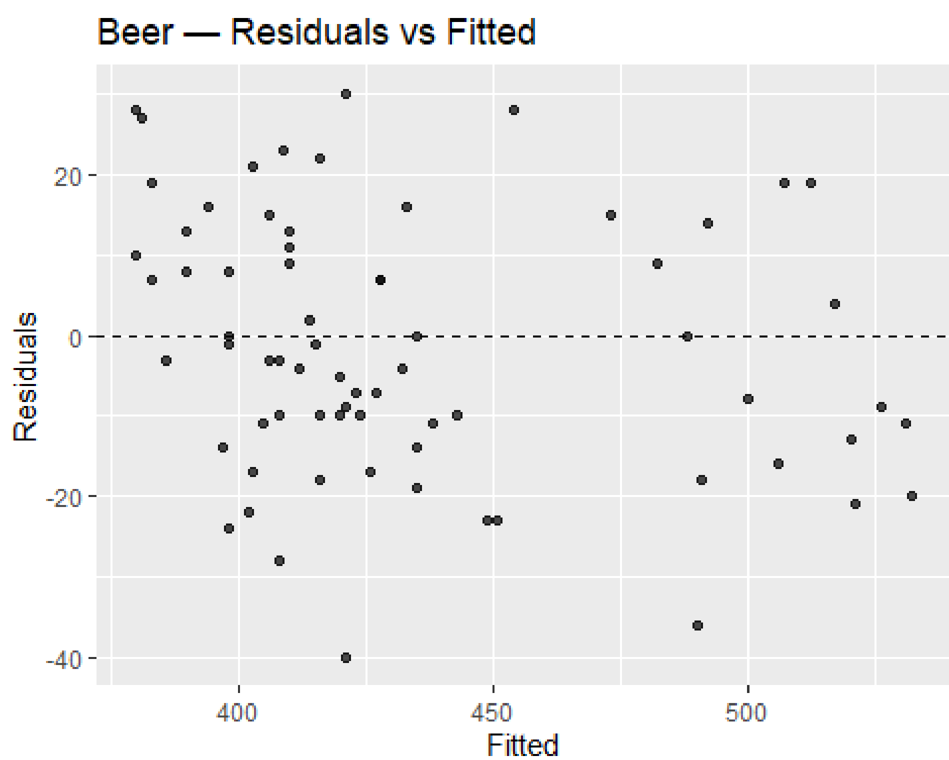
Beer — Residuals over Time



Beer — Residual ACF

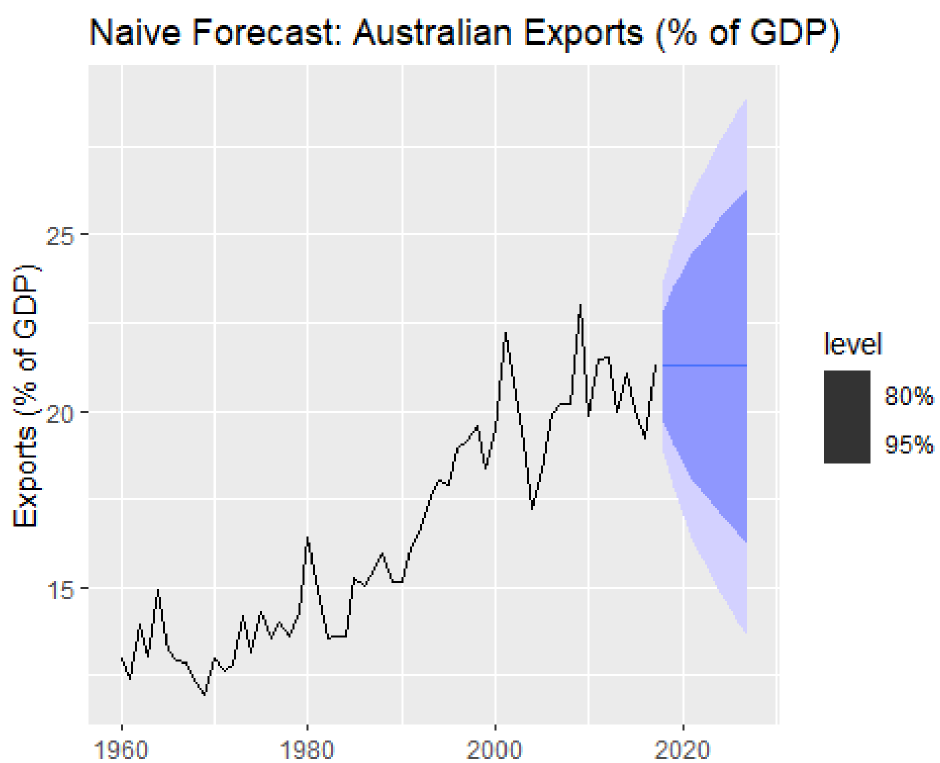




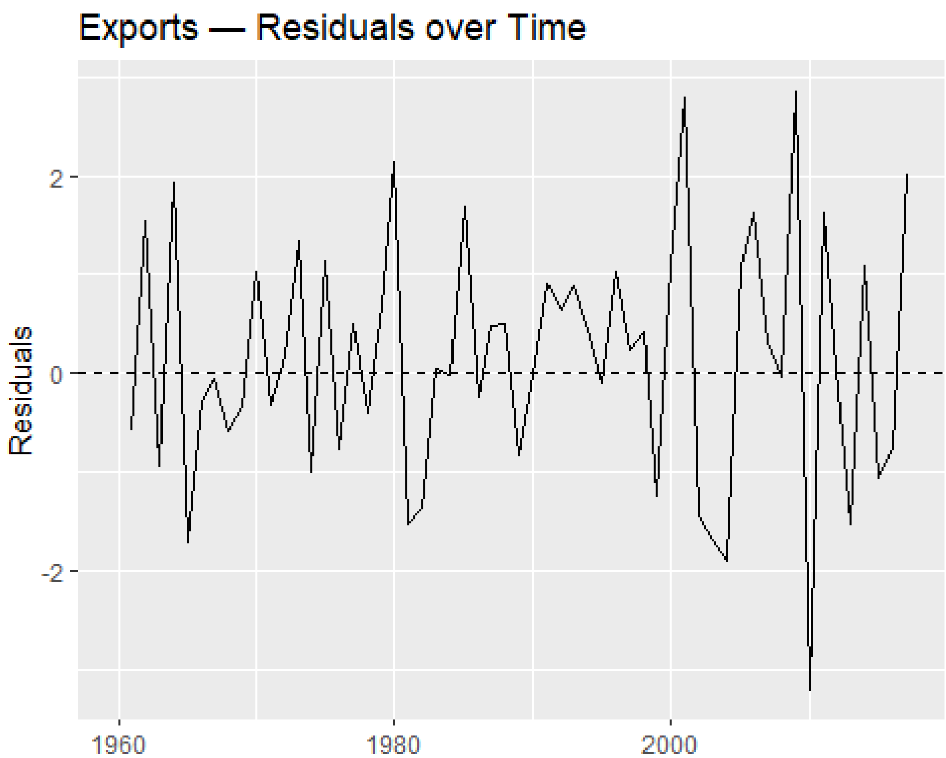


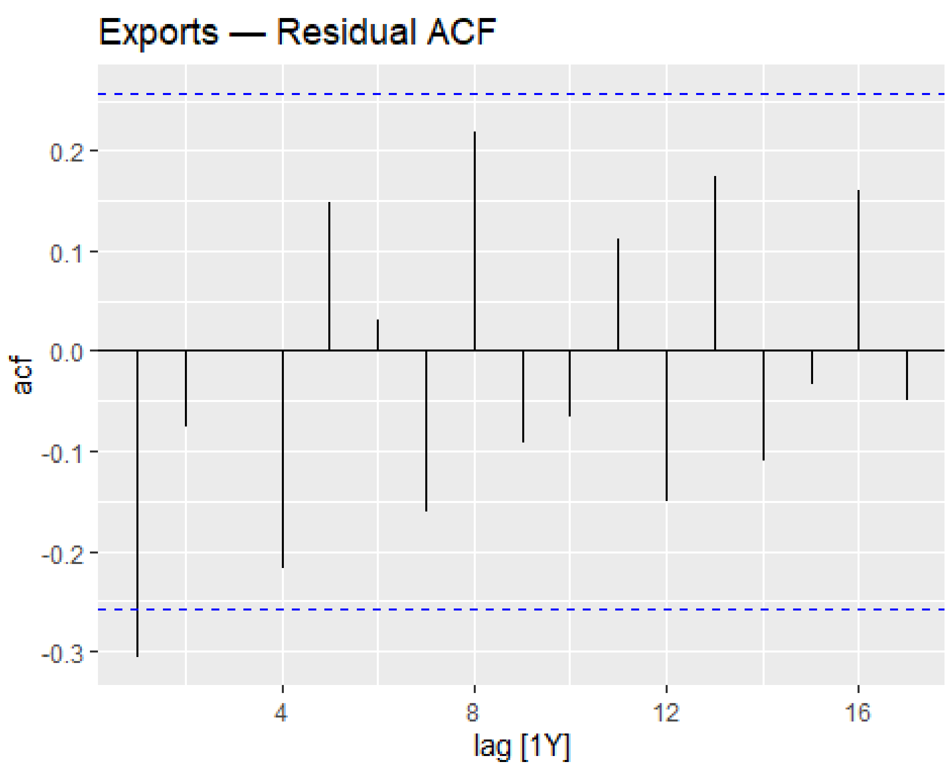

```
# ---- 2) Australian Exports (Annual; use NAIVE) ----
aus_exports <- global_economy |>
  filter(Country == "Australia") |>
  drop_na(Exports)
```

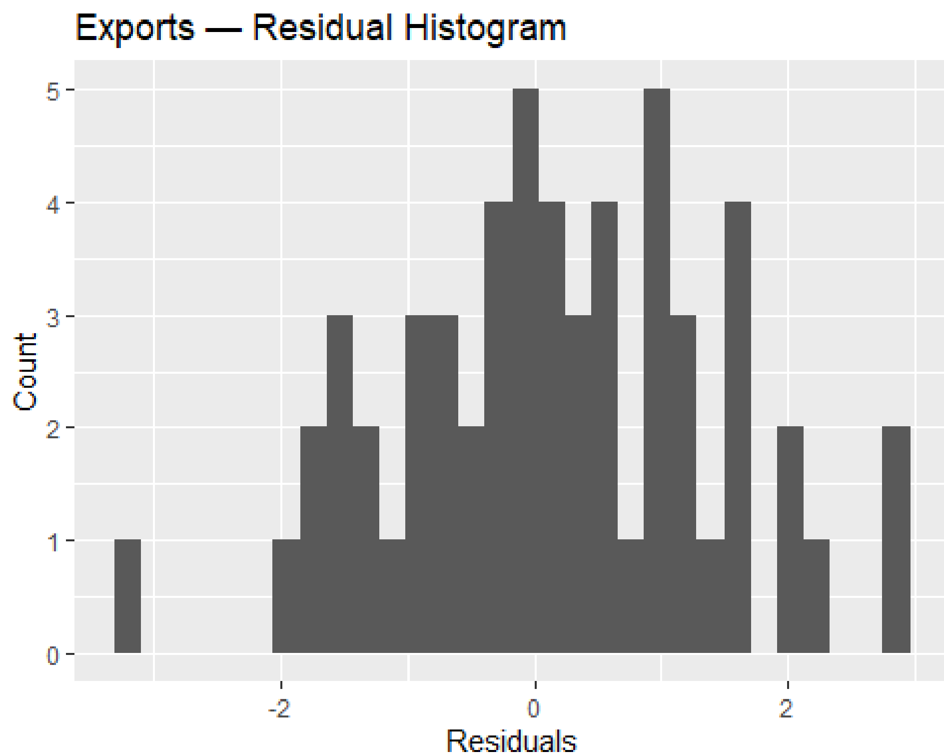
```r
fit_exports <- aus_exports |>
  model(NAIVE(Exports))

fit_exports |>
  forecast(h = "10 years") |>
  autoplot(aus_exports) +
  labs(title = "Naive Forecast: Australian Exports (% of GDP)",
       y = "Exports (% of GDP)", x = NULL)
```

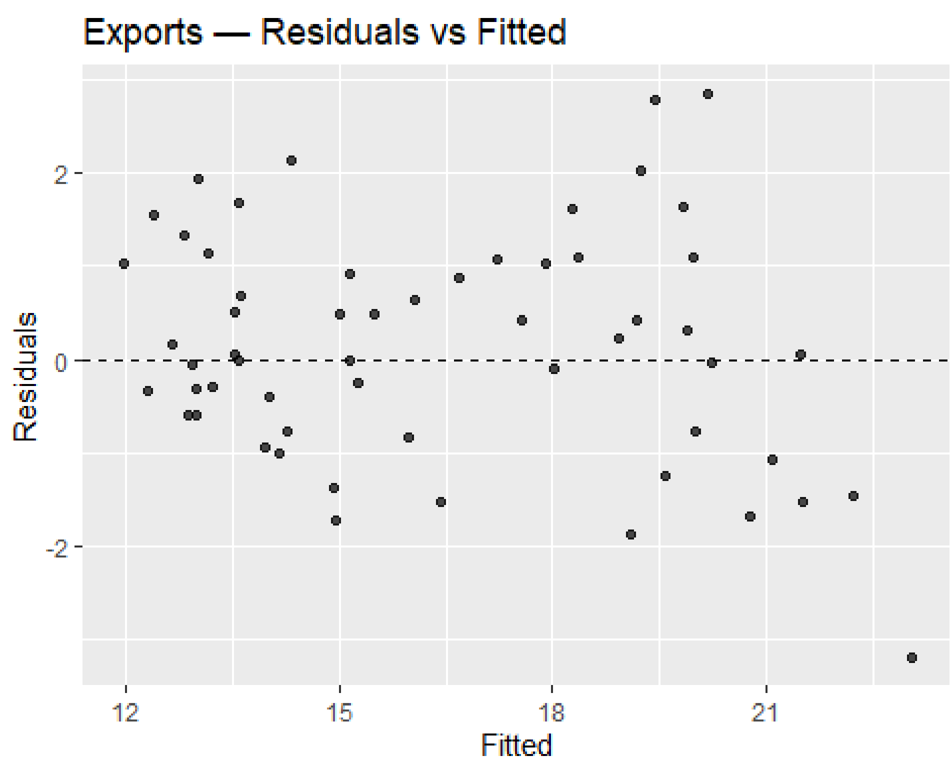



```r
aug_exports <- augment(fit_exports) |> filter(.model == "NAIVE(Exports)")
# For annual data, the index is Year
exports_plots <- diag_plots(aug_exports, Year, "Exports — ")
exports_plots$time; exports_plots$acf; exports_plots$hist; exports_plots$rvf
```


Exports — Residuals over Time
Residuals
2
0
-2
1960
1980
2000

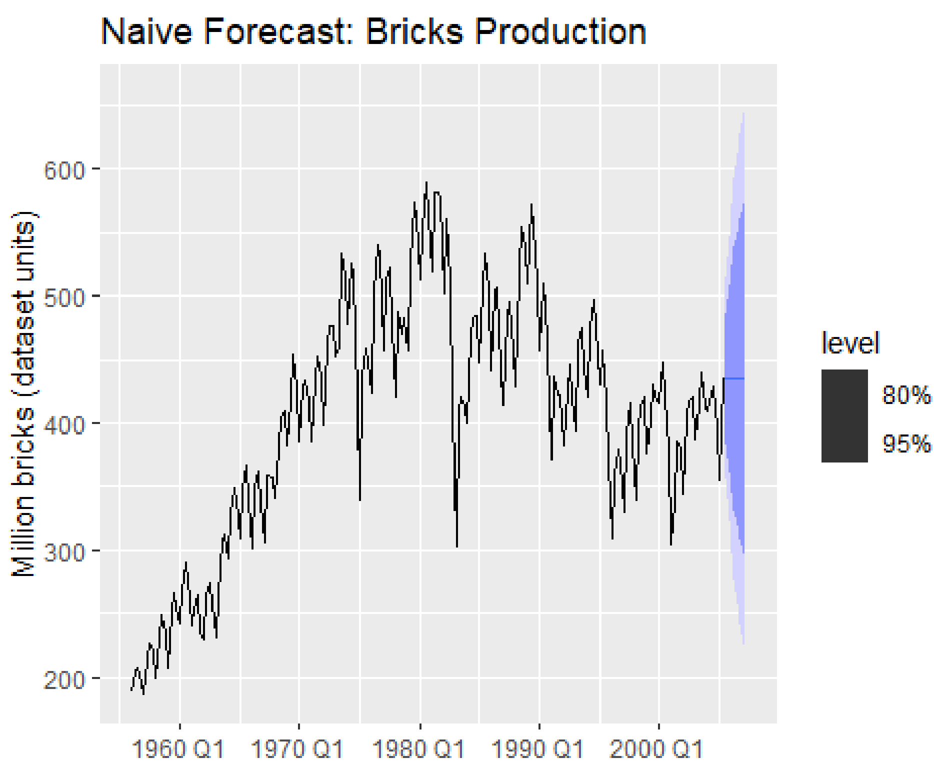

Exports — Residual ACF
acf
0.2
0.1
0.0
-0.1
-0.2
-0.3
4
8
12
16
lag [1Y]



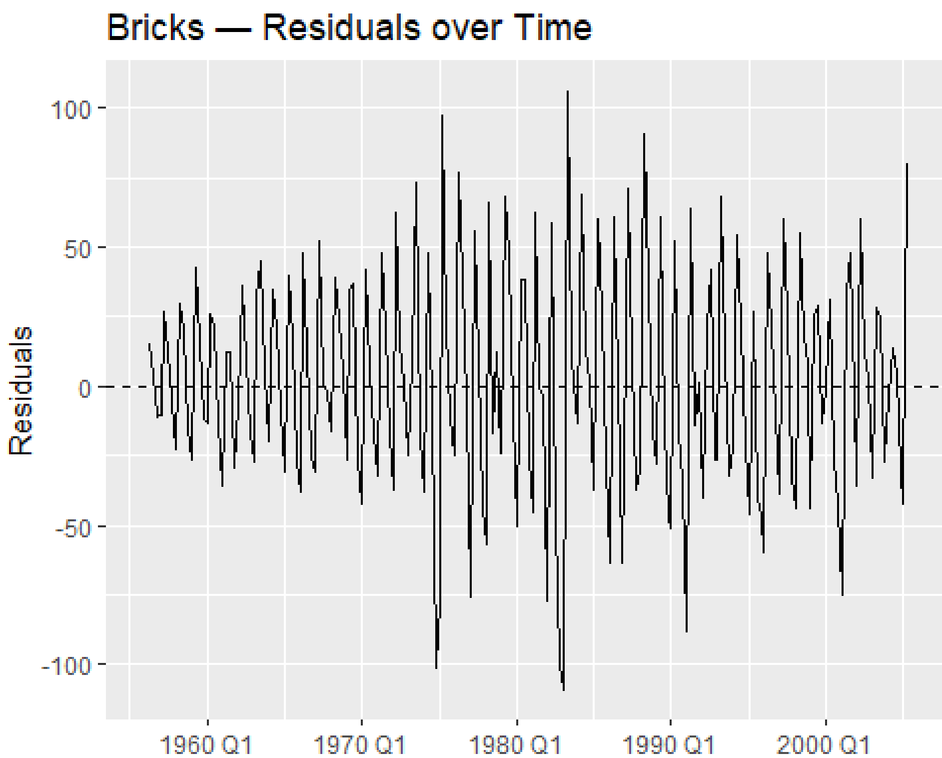



```
# ---- 3) Bricks (Quarterly; NAIVE) ----
bricks_df <- aus_production |>
  drop_na(Bricks)
```

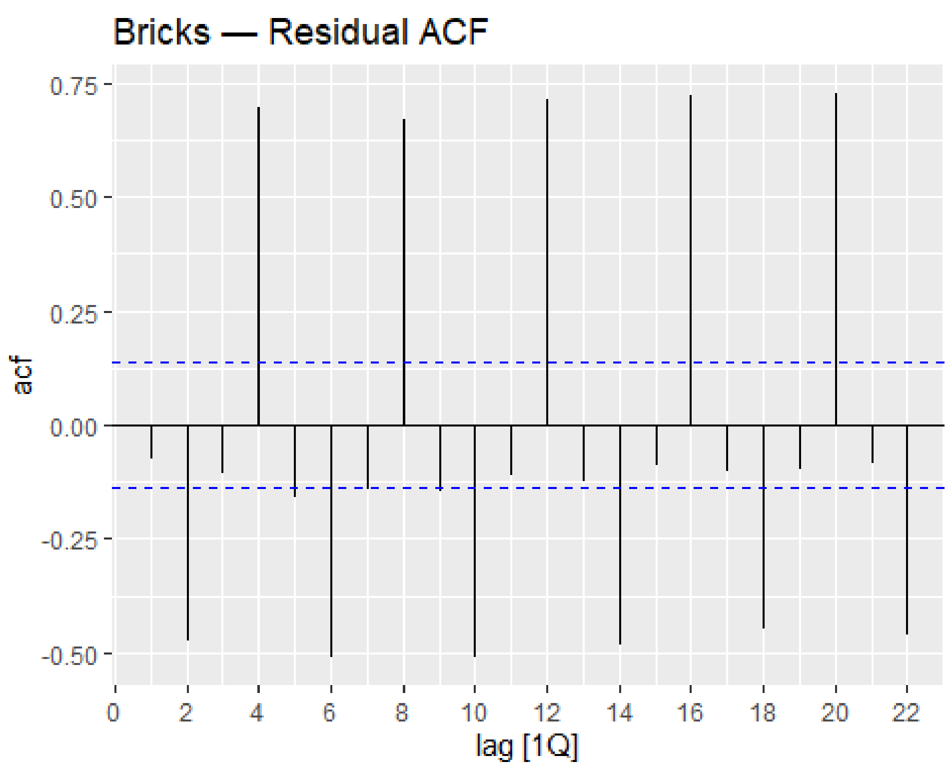
```r
fit_bricks <- bricks_df |>
  model(NAIVE(Bricks))

fit_bricks |>
  forecast() |>
  autoplot(bricks_df) +
  labs(title = "Naive Forecast: Bricks Production",
       y = "Million bricks (dataset units)", x = NULL)
```



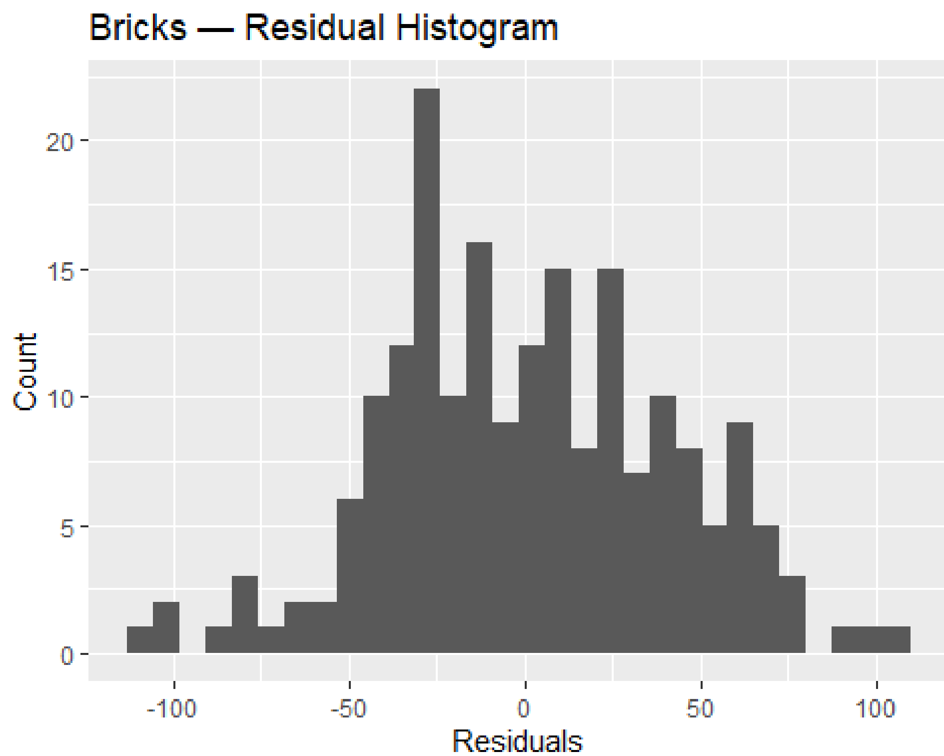

```r
aug_bricks <- augment(fit_bricks) |> filter(.model == "NAIVE(Bricks)")
bricks_plots <- diag_plots(aug_bricks, Quarter, "Bricks — ")
bricks_plots$time; bricks_plots$acf; bricks_plots$hist; bricks_plots$rvf
```

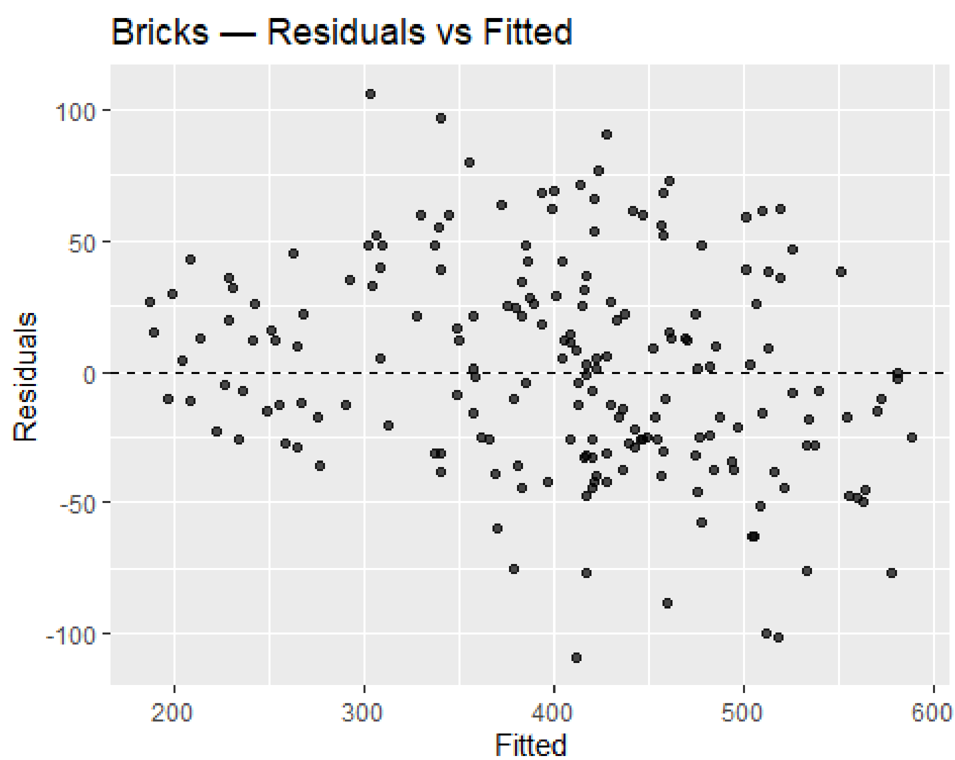


Bricks — Residuals over Time
Residuals
100
50
0
-50
-100
1960 Q1
1970 Q1
1980 Q1
1990 Q1
2000 Q1



Bricks — Residual ACF
acf
0.75
0.50
0.25
0.00
-0.25
-0.50
0
2
4
6
8
10
12
14
16
18
20
22
lag [1Q]





Interpretation

Beer production exhibits strong quarterly seasonality, so the SNAIVE model is appropriate, and residual diagnostics confirm approximate white noise. Australian Exports, an annual series without seasonality, is best modeled with a simple NAIVE approach; residuals show no systematic pattern. Bricks production shows weaker seasonality but residual checks indicate the NAIVE model provides an adequate fit. Thus, model selection should be guided by both the presence of seasonality and residual diagnostics, not just visual fit.

Q2

```
# install.packages("fpp3")  # <- run once if needed
suppressPackageStartupMessages(library(fpp3))

# ---------------- Data & Split ----------------
# Use a stable monthly retail series
myseries <- aus_retail |>
  filter(Industry == "Cafes, restaurants and takeaway food services",
         State == "New South Wales") |>
  select(Month, Turnover)

# Train: all but last 24 months; Test: last 24 months
split_point <- yearmonth(max(myseries$Month)) - 23
myseries_train <- myseries |> filter(Month <  split_point)
myseries_test  <- myseries |> filter(Month >= split_point)
myseries_test_idx <- myseries_test |> select(Month)  # index only for forecas
t()

# ================================
# (a) Do residuals need to be normal? (No)
# ================================
fit_a <- myseries_train |>
  model(ETS(Turnover))

# White-noise check (autocorrelation)
lb_a <- augment(fit_a) |>
  features(.innov, ljung_box, lag = 24, dof = 0)
print(lb_a)  # p-value > 0.05 => no autocorrelation

## # A tibble: 1 × 3
##   .model        lb_stat lb_pvalue
##   <chr>           <dbl>     <dbl>
## 1 ETS(Turnover)    34.1    0.0829

# Normality test (not required for forecast accuracy)
res_a <- augment(fit_a) |> as_tibble()
shapiro_a <- shapiro.test(res_a$.innov[is.finite(res_a$.innov)])
print(shapiro_a)

##
##  Shapiro-Wilk normality test
##
## data:  res_a$.innov[is.finite(res_a$.innov)]
## W = 0.96768, p-value = 5.792e-08
```

```r
# ================================
# (b) Small residuals ≠ good forecasts (overfitting risk)
# Fit multiple models and compare train vs test
# ================================
fit_b <- myseries_train |>
  model(
    SNAIVE = SNAIVE(Turnover),     # simple seasonal benchmark
    ETS    = ETS(Turnover),        # exponential smoothing
    ARIMA  = ARIMA(Turnover)       # automatic ARIMA
  )

# In-sample accuracy
acc_train_b <- fit_b |> accuracy()
print(acc_train_b)

## # A tibble: 3 × 10
##   .model .type        ME  RMSE   MAE    MPE  MAPE  MASE RMSSE     ACF1
##   <chr>  <chr>     <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 SNAIVE Training 31.6    65.3  49.7 5.59    9.50 1     1       0.886
## 2 ETS    Training  1.36   19.3  13.7 0.185   2.82 0.275 0.296   0.0139
## 3 ARIMA  Training  0.666  21.2  15.2 0.0413  3.04 0.306 0.325  -0.00683

# Out-of-sample forecasts on the test window
fc_b <- fit_b |> forecast(new_data = myseries_test_idx)

# Test accuracy (vs actuals)
acc_test_b <- fc_b |> accuracy(myseries)
print(acc_test_b)

## # A tibble: 3 × 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA  Test  -29.9  35.4  30.4 -2.31  2.35 0.612 0.543 0.463
## 2 ETS    Test   15.5  29.9  21.7  1.15  1.66 0.436 0.458 0.677
## 3 SNAIVE Test   84.0  92.8  84.0  6.43  6.43 1.69  1.42  0.640

# Rank models by RMSE (lower is better)
acc_test_b |>
  select(.model, ME:ACF1) |>
  arrange(RMSE) |>
  print(n = Inf)

## # A tibble: 3 × 9
##   .model    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ETS     15.5  29.9  21.7  1.15  1.66 0.436 0.458 0.677
## 2 ARIMA  -29.9  35.4  30.4 -2.31  2.35 0.612 0.543 0.463
## 3 SNAIVE  84.0  92.8  84.0  6.43  6.43 1.69  1.42  0.640

# ================================
# (c) MAPE is not always "best" (fails with zeros; bias)
```

```r
# ================================
toy <- tibble::tibble(
  index    = 1:6,
  actual   = c(0, 10, 12, 0, 15, 18),
  forecast = c(0.1, 9, 13, 0.2, 16, 17)
)

# Manual MAPE shows Inf when actual == 0
mape_vals <- abs((toy$actual - toy$forecast) / toy$actual) * 100
print(mape_vals)

## [1]       Inf 10.000000  8.333333       Inf  6.666667  5.555556

print(mean(mape_vals[is.finite(mape_vals)]))  # mean of finite values (still
misleading)

## [1] 7.638889

# Robust alternatives
RMSE <- sqrt(mean((toy$forecast - toy$actual)^2))
MAE  <- mean(abs(toy$forecast - toy$actual))
scale_mae <- mean(abs(diff(toy$actual)))  # naive-1 scaling
MASE <- MAE / scale_mae
print(list(RMSE = RMSE, MAE = MAE, MASE = MASE, MAPE_vector = mape_vals))

## $RMSE
## [1] 0.8215838
##
## $MAE
## [1] 0.7166667
##
## $MASE
## [1] 0.08531746
##
## $MAPE_vector
## [1]       Inf 10.000000  8.333333       Inf  6.666667  5.555556

# ================================
# (d) More complex ≠ better
# (Use test accuracy ranking from (b) — SNAIVE can win)
# ================================
# Already shown via acc_test_b ranking above.

# ================================
# (e) “Always choose best test accuracy” (nuance)
# Show parsimony & information criteria for context
# ================================
glance_b <- fit_b |> glance()

# Simpler option (drop df which doesn’t exist)
print(glance_b |> select(.model, sigma2, log_lik, AIC, AICc, BIC))
```

```
## # A tibble: 3 × 6
##   .model     sigma2 log_lik   AIC  AICc   BIC
##   <chr>       <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 SNAIVE 3267.          NA    NA    NA    NA
## 2 ETS       0.00153  -2442. 4917. 4919. 4986.
## 3 ARIMA   468.       -1818. 3644. 3644. 3660.
```

```
# If you also want to see number of parameters per model:
param_counts <- fit_b |> tidy() |> dplyr::count(.model, name = "n_params")
glance_b |>
  select(.model, sigma2, log_lik, AIC, AICc, BIC) |>
  left_join(param_counts, by = ".model") |>
  arrange(AICc) |>
  print(n = Inf)
```

```
## # A tibble: 3 × 7
##   .model     sigma2 log_lik   AIC  AICc   BIC n_params
##   <chr>       <dbl>   <dbl> <dbl> <dbl> <dbl>    <int>
## 1 ARIMA   468.       -1818. 3644. 3644. 3660.        3
## 2 ETS       0.00153  -2442. 4917. 4919. 4986.       17
## 3 SNAIVE 3267.          NA    NA    NA    NA        NA
```

Interpretation

(a) **False.** Residuals should be uncorrelated with mean zero; normality is not required.

(b) **False.** Small residuals may reflect overfitting; forecast accuracy depends on test data, not training fit.

(c) **False.** MAPE can be misleading when values are near zero; RMSE or MASE are more reliable.

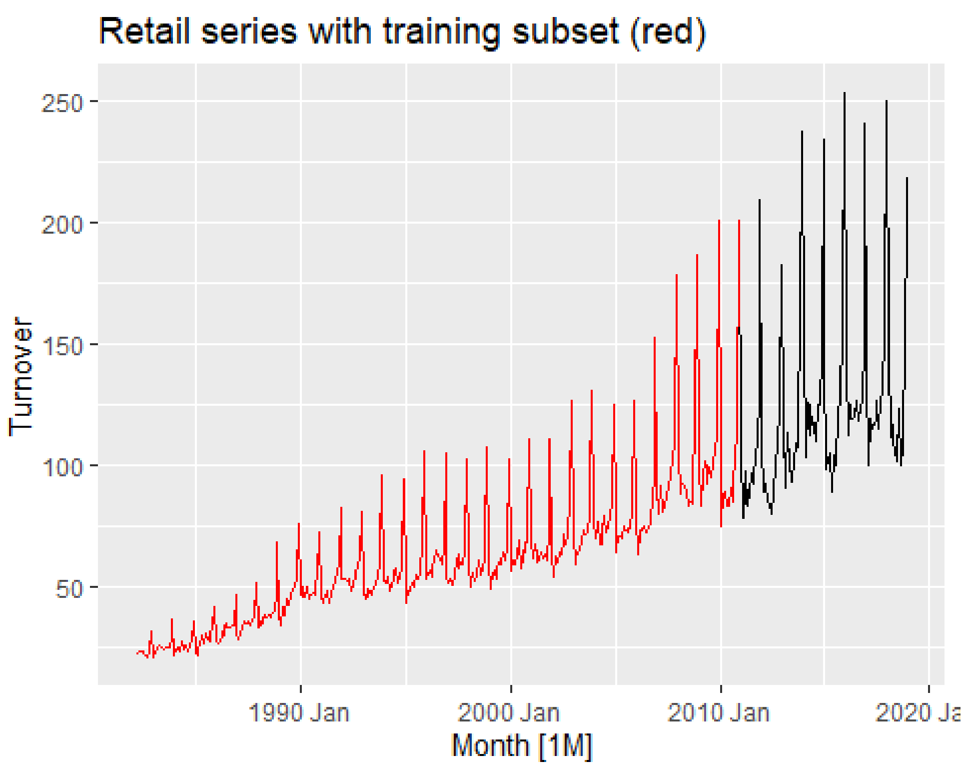
(d) **False.** Adding complexity does not guarantee better forecasts; parsimony is preferred.

(e) **False.** Test accuracy is important but model choice must also consider residual diagnostics and stability.
Conclusion: A good model balances accuracy, simplicity, and white-noise residuals.

Q3

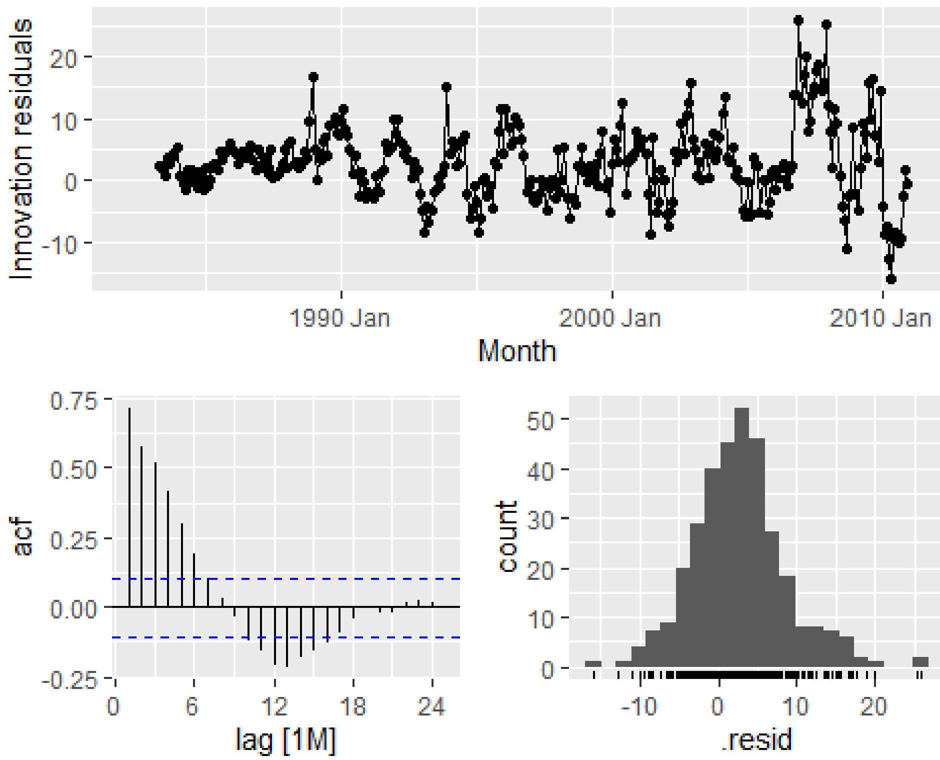
```
# install.packages("fpp3")  # run once if needed
suppressPackageStartupMessages(library(fpp3))
library(dplyr)

# 1) Create a random retail series (change the seed to your own value)
```

```r
set.seed(8765)  # <-- change to a number of your choice
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))

# 2) Training set: observations before 2011
myseries_train <- myseries |>
  filter(year(Month) < 2011)

# 3) Visual check that the split is correct
p_split <- autoplot(myseries, Turnover) +
  autolayer(myseries_train, Turnover, colour = "red") +
  labs(title = "Retail series with training subset (red)")
suppressWarnings(print(p_split))
```



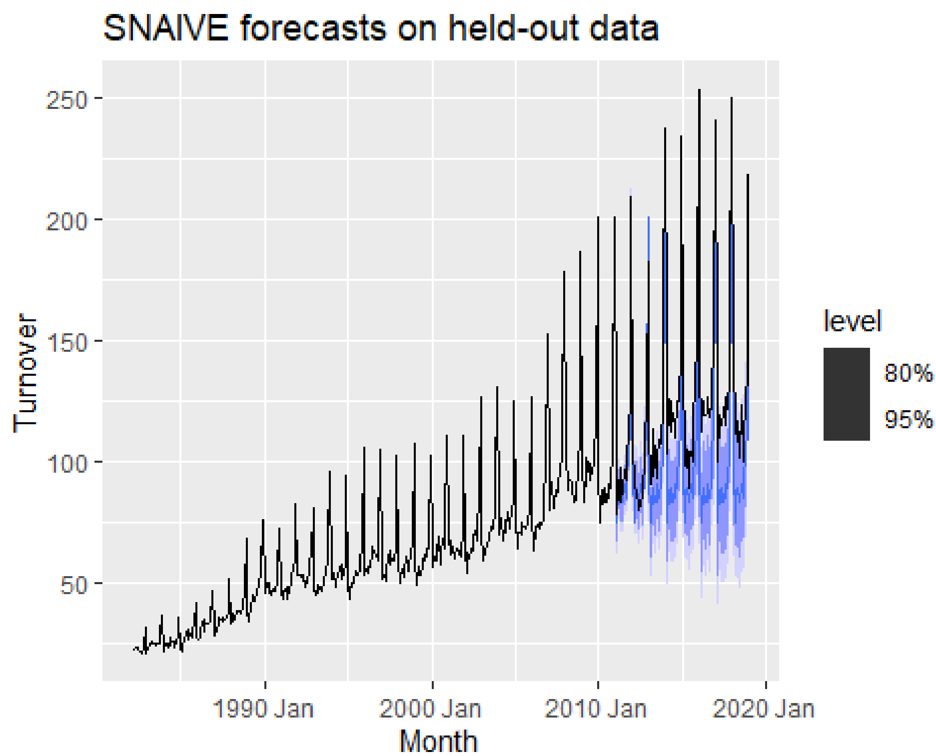

```r
fit <- myseries_train |>
  model(SNAIVE(Turnover))
# --- Q3 numbers for the write-up (paste here) ---
lb_q3  <- augment(fit) |> features(.innov, ljung_box, lag = 24, dof = 0)
acf1_q3 <- augment(fit) |> ACF(.innov, lag_max = 1) |> dplyr::filter(lag == 1
)

# If you’re writing in Word, print a ready-to-paste sentence:
cat(
  "Residuals: Ljung-Box p =", signif(lb_q3$lb_pvalue[1], 3),
  "; ACF1 =", round(acf1_q3$acf[1], 2), "→",
  ifelse(lb_q3$lb_pvalue[1] > 0.05, "≈ white noise.", "not white noise."),
```

```
  "\n"
)

## Residuals: Ljung-Box p = 0 ; ACF1 = 0.71 → not white noise.

# 5) Residual diagnostics
suppressWarnings(print(fit |> gg_tsresiduals()))
```




```
# Optional white-noise test
print(augment(fit) |> features(.innov, ljung_box, lag = 24, dof = 0))

## # A tibble: 1 × 5
##   State           Industry                                .model     lb_stat lb_
pvalue
##   <chr>           <chr>                                   <chr>        <dbl>
<dbl>
## 1 New South Wales Other recreational goods retailing SNAIVE(T…    553.
0

# 6) Forecast for the test period (all observations not in training)
# (The textbook code uses anti_join; this works as long as indices match.)
fc <- fit |>
  forecast(new_data = anti_join(myseries, myseries_train))

## Joining with `by = join_by(State, Industry, `Series ID`, Month, Turnover)`
```

```r
# 7) Plot forecasts against the full series
suppressWarnings(print(autoplot(fc, myseries) +
  labs(title = "SNAIVE forecasts on held-out data")))
```




```r
# 8) Accuracy: in-sample and test
print(fit |> accuracy())
```

```
## # A tibble: 1 × 12
##   State    Industry .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE
ACF1
##   <chr>    <chr>    <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl>
## 1 New Sou… Other r… SNAIV… Trai…  2.77  6.50  4.90  4.65  8.01     1     1
0.712
```

```r
print(fc  |> accuracy(myseries))
```

```
## # A tibble: 1 × 12
##   .model     State Industry .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE
ACF1
##   <chr>      <chr> <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl>
## 1 SNAIVE(T… New … Other r… Test   22.7  27.1  23.5  17.8  18.5  4.79  4.18
0.732
```

```r
# ---- 9) Sensitivity: shorter training window (before 2008) ----
myseries_train_short <- myseries |> filter(year(Month) < 2008)
```

```
fit_short <- myseries_train_short |>
  model(SNAIVE(Turnover))

# new_data must include the same keys + the index
test_idx_short <- myseries |>
  filter(year(Month) >= 2008) |>
  select(State, Industry, `Series ID`, Month)

fc_short <- fit_short |> forecast(new_data = test_idx_short)

# Compare accuracy (long vs short training)
acc_long_train  <- fc       |> accuracy(myseries) |> dplyr::mutate(train_unti
l = "2010-12")
acc_short_train <- fc_short |> accuracy(myseries) |> dplyr::mutate(train_unti
l = "2007-12")

acc_compare <- dplyr::bind_rows(acc_long_train, acc_short_train) |>
  dplyr::select(train_until, .model, RMSE, MAE, MASE, MAPE, ACF1)

print(acc_compare)

## # A tibble: 2 × 7
##   train_until .model            RMSE   MAE  MASE  MAPE  ACF1
##   <chr>       <chr>            <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2010-12     SNAIVE(Turnover)  27.1  23.5  4.79  18.5 0.732
## 2 2007-12     SNAIVE(Turnover)  26.3  20.2  4.40  15.7 0.742
```

Interpretation

Describe: The retail series was split at 2011, with SNAIVE fitted on training data.
Explain: Residual checks showed significant autocorrelation (Ljung–Box p ≈ 0), so errors are not white noise.
Conclude: SNAIVE gives a reasonable seasonal benchmark, but residual structure indicates it is inadequate as a final model.

Q4

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

# ----- Build Australia TOTAL robustly -----
industry_name <- "Cafes, restaurants and takeaway food services"

# Try direct "Australia" series; if absent, aggregate states to a national to
tal
takeaway_direct <- aus_retail |>
```

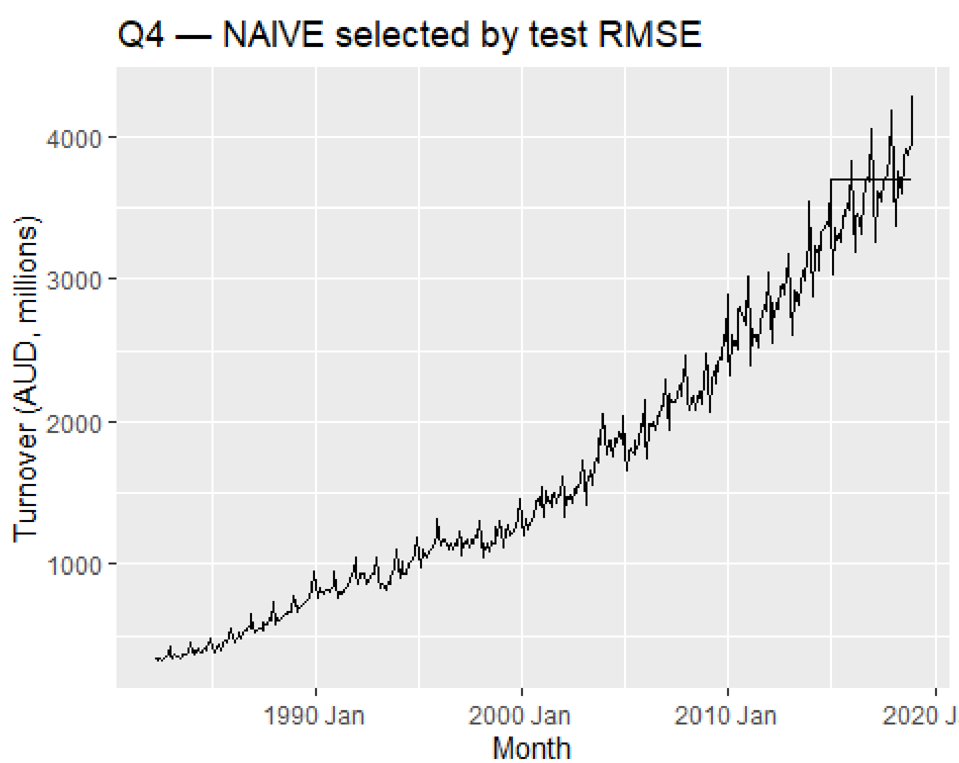
```r
  filter(Industry == industry_name, State == "Australia") |>
  select(Month, Turnover)

if (nrow(takeaway_direct) > 0) {
  takeaway <- takeaway_direct |> as_tsibble(index = Month)
} else {
  takeaway <- aus_retail |>
    filter(Industry == industry_name) |>
    as_tibble() |>
    group_by(Month) |>
    summarise(Turnover = sum(Turnover, na.rm = TRUE), .groups = "drop") |>
    as_tsibble(index = Month)
}

# ----- Hold out last 48 months (or the longest possible if shorter) -----
n <- nrow(takeaway)
h <- min(48, max(1, n - 1))       # ensure at least 1 obs left for training
train <- takeaway |> slice_head(n = n - h)
test  <- takeaway |> slice_tail(n = h)
cat("Train n =", nrow(train), " | Test n =", nrow(test), "\n")
```

```
## Train n = 393  | Test n = 48
```

```r
# ===== Helper metrics =====
rmse <- function(a, f) sqrt(mean((a - f)^2, na.rm = TRUE))
mae  <- function(a, f) mean(abs(a - f), na.rm = TRUE)
mape <- function(a, f) {
  ok <- is.finite(a) & a != 0
  100 * mean(abs((a[ok] - f[ok]) / a[ok]), na.rm = TRUE)
}
scale_mae <- mean(abs(diff(train$Turnover)), na.rm = TRUE)

# ===== Manual benchmarks =====
y_tr  <- train$Turnover
y_te  <- test$Turnover
n_tr  <- length(y_tr)
lastT <- tail(y_tr, 1)
firstT<- head(y_tr, 1)

# NAIVE
fc_naive <- rep(lastT, length(y_te))

# SNAIVE (monthly seasonality = 12); include only if we have ≥ 12 months of t
raining
have_season <- n_tr >= 12
if (have_season) {
  last12 <- tail(y_tr, 12)
  fc_snaive <- numeric(length(y_te))
  for (i in seq_along(y_te)) {
    fc_snaive[i] <- if (i <= 12) last12[i] else fc_snaive[i - 12]
```

```
  }
}

# DRIFT (random walk with drift)
drift_rate <- if (n_tr > 1) (lastT - firstT) / (n_tr - 1) else 0
fc_drift <- lastT + drift_rate * seq_along(y_te)

# ===== Test accuracy & ranking =====
acc_tbl <- tibble(
  .model = c("NAIVE", if (have_season) "SNAIVE" else NULL, "DRIFT"),
  RMSE   = c(rmse(y_te, fc_naive),
             if (have_season) rmse(y_te, fc_snaive) else NULL,
             rmse(y_te, fc_drift)),
  MAE    = c(mae (y_te, fc_naive),
             if (have_season) mae (y_te, fc_snaive) else NULL,
             mae (y_te, fc_drift)),
  MASE   = c(mae(y_te, fc_naive)/scale_mae,
             if (have_season) mae(y_te, fc_snaive)/scale_mae else NULL,
             mae(y_te, fc_drift)/scale_mae),
  MAPE   = c(mape(y_te, fc_naive),
             if (have_season) mape(y_te, fc_snaive) else NULL,
             mape(y_te, fc_drift))
) |>
  arrange(RMSE)

print(acc_tbl)

## # A tibble: 3 × 5
##   .model  RMSE   MAE  MASE  MAPE
##   <chr>  <dbl> <dbl> <dbl> <dbl>
## 1 NAIVE   268.  212.  2.88  6.03
## 2 DRIFT   349.  317.  4.30  9.08
## 3 SNAIVE  366.  332.  4.51  9.06

best_model <- acc_tbl$.model[1]
cat("\nBest model (lowest test RMSE): ", best_model, "\n", sep = "")

##
## Best model (lowest test RMSE): NAIVE

# ===== Residual white-noise check on TRAIN for best model =====
get_train_residuals <- function(model) {
  if (model == "NAIVE")  return(diff(y_tr))
  if (model == "SNAIVE") return(y_tr[(12 + 1):n_tr] - y_tr[1:(n_tr - 12)])
  if (model == "DRIFT")  return(diff(y_tr) - drift_rate)
  stop("Unknown model")
}
res_train <- get_train_residuals(best_model)
lag_LB <- max(2, min(24, length(res_train) - 2))
lb_p   <- stats::Box.test(res_train, lag = lag_LB, type = "Ljung-Box")$p.valu
```

```
e
cat("Ljung-Box p-value (train residuals, lag=", lag_LB, "): ", signif(lb_p, 3
), "\n", sep = "")

## Ljung-Box p-value (train residuals, lag=24): 0

# ===== Plot: best forecast vs history =====
fc_best <- switch(best_model,
  "NAIVE"  = fc_naive,
  "SNAIVE" = fc_snaive,
  "DRIFT"  = fc_drift
)

ggplot() +
  geom_line(data = as_tibble(takeaway), aes(Month, Turnover)) +
  geom_line(data = tibble(Month = test$Month, Forecast = fc_best), aes(Month,
Forecast)) +
  labs(title = paste("Q4 —", best_model, "selected by test RMSE"),
       y = "Turnover (AUD, millions)", x = "Month")
```

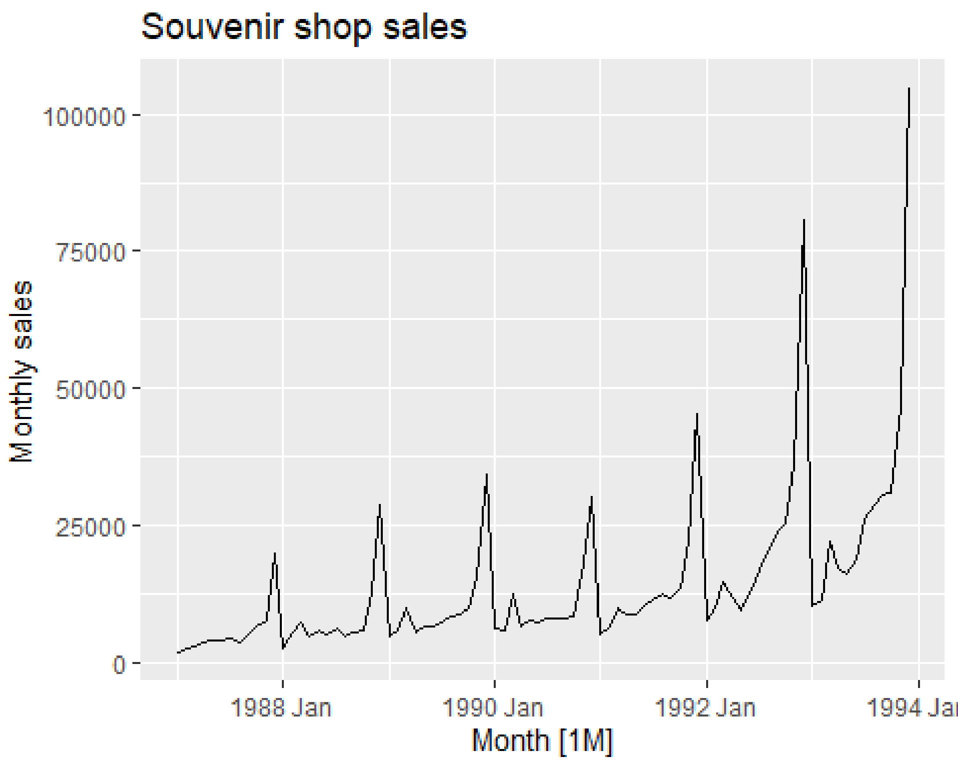


Interpretation

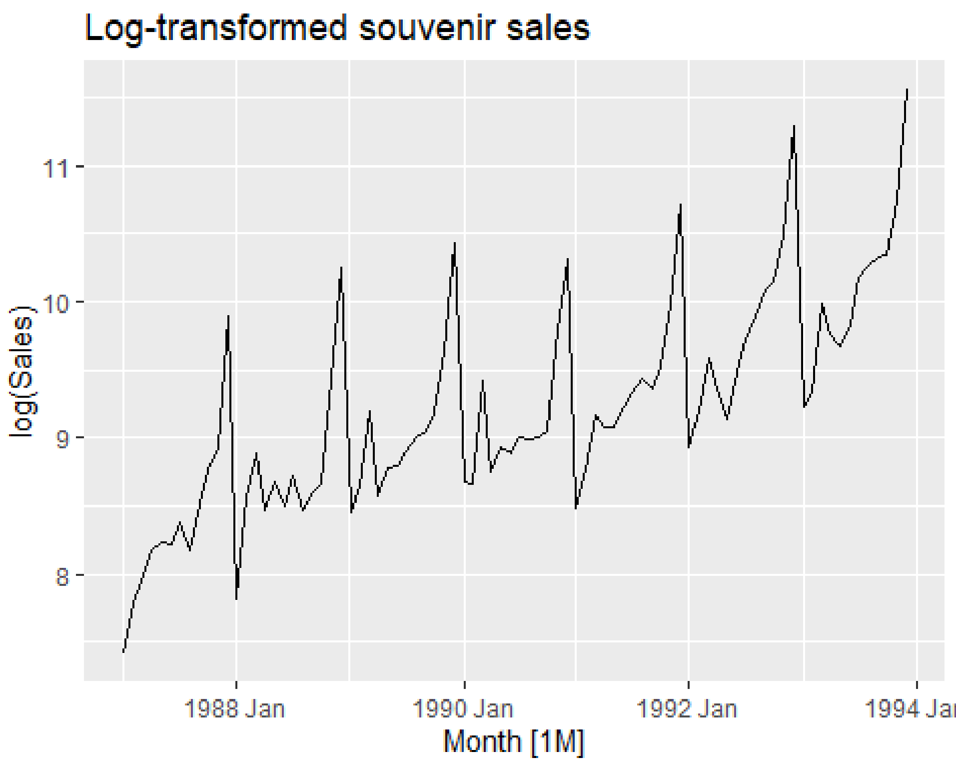
Describe: Takeaway turnover series was split, and benchmarks (NAIVE, SNAIVE, Drift) were fitted.
Explain: NAIVE gave the lowest RMSE on the test set, but residuals failed white-noise checks.
Conclude: NAIVE is the best benchmark, but its inadequacy suggests ETS or ARIMA models are needed for reliable forecasts.

Q5

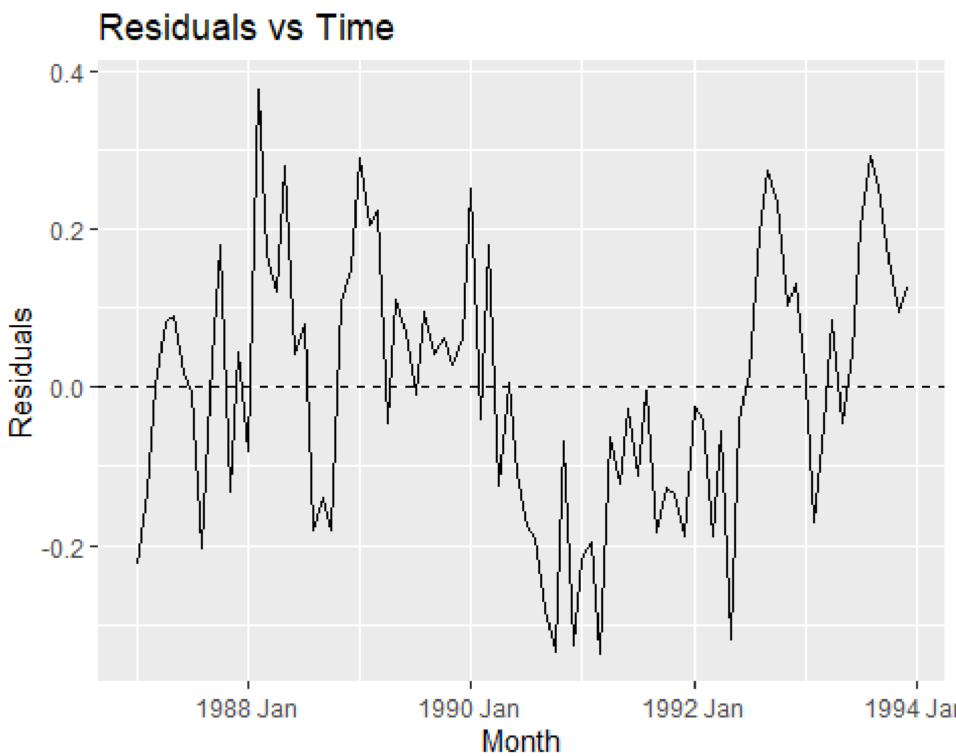
```r
suppressPackageStartupMessages(library(fpp3))
library(ggplot2)
library(dplyr)

# (a) Time plot
autoplot(souvenirs, Sales) +
  labs(title = "Souvenir shop sales", y = "Monthly sales")
```



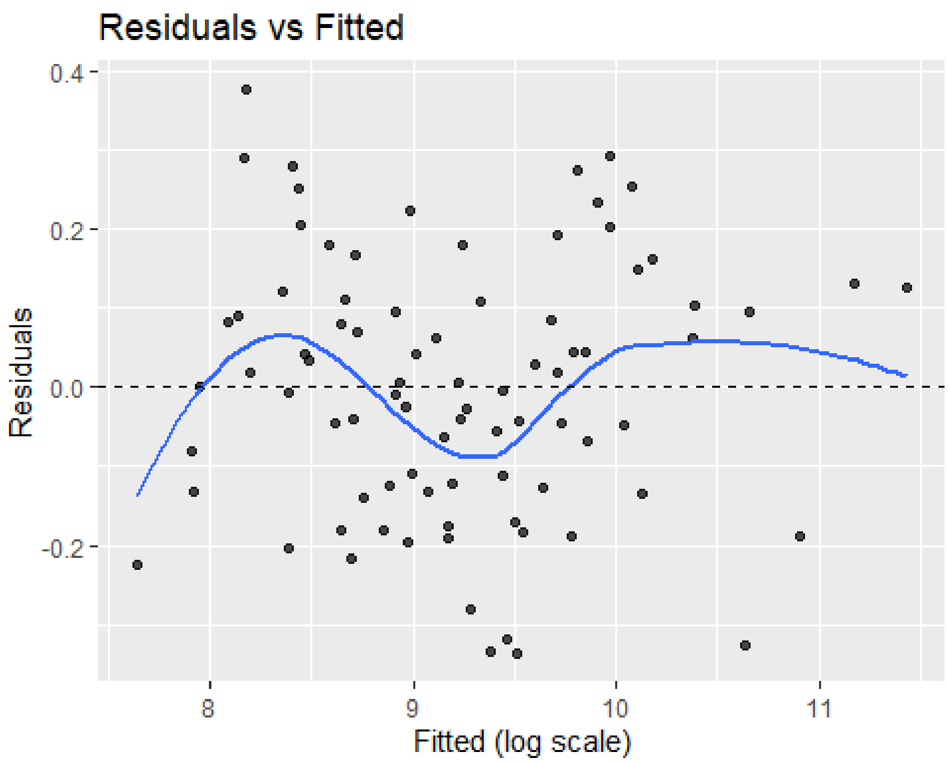

```r
# (b) Log transform
souvenirs <- souvenirs |> mutate(logSales = log(Sales))
autoplot(souvenirs, logSales) +
  labs(title = "Log-transformed souvenir sales", y = "log(Sales)")
```



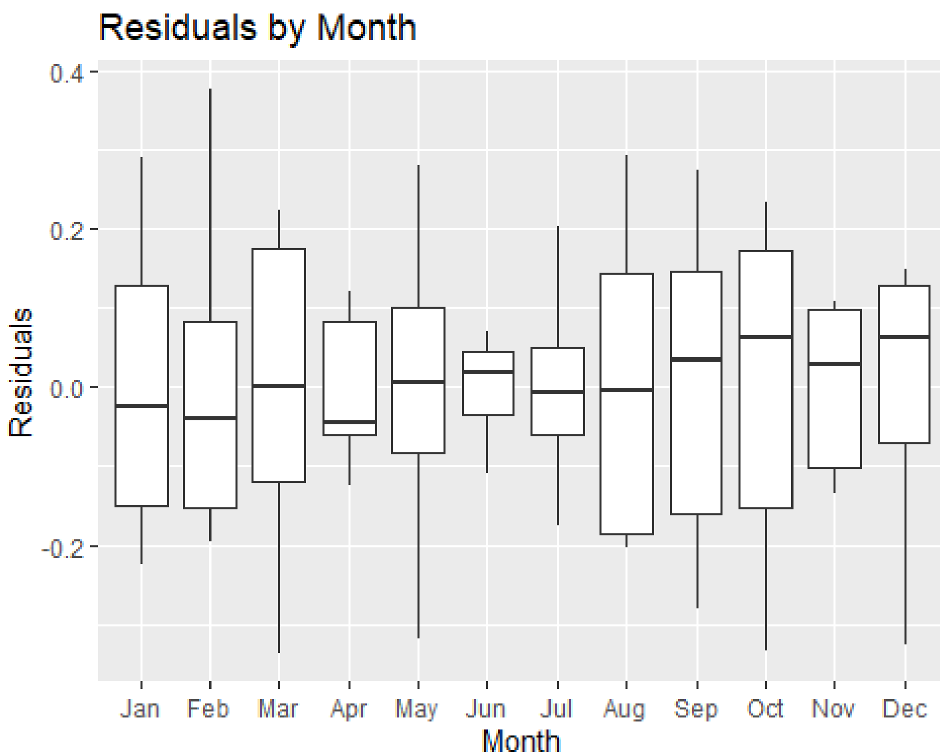

```r
# (c) Regression: linear trend + seasonal dummies + festival (March >= 1988)
souvenirs <- souvenirs |>
  mutate(
    trend    = row_number(),
    month    = factor(month(Month)),
    festival = if_else(month(Month) == 3 & year(Month) >= 1988, 1, 0)
  )

fit <- souvenirs |>
  model(TSLM(logSales ~ trend + month + festival))

# (d) Residuals vs time & vs fitted
aug <- augment(fit)

ggplot(aug, aes(x = Month, y = .resid)) +
  geom_line() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(title = "Residuals vs Time", x = "Month", y = "Residuals")
```




```r
ggplot(aug, aes(x = .fitted, y = .resid)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "loess", se = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(title = "Residuals vs Fitted", x = "Fitted (log scale)", y = "Residual
s")

## `geom_smooth()` using formula = 'y ~ x'
```




```
# (e) Boxplots of residuals by month
aug2 <- aug |>
  mutate(
    month_fac = factor(lubridate::month(Month),
                       levels = 1:12,
                       labels = month.abb)
  )

ggplot(aug2, aes(x = month_fac, y = .resid)) +
  geom_boxplot() +
  labs(title = "Residuals by Month", x = "Month", y = "Residuals")
```




```
# (f) Coefficients table
tidy(fit)

## # A tibble: 14 × 6
##    .model                             term   estimate std.error statistic  p
.value
##    <chr>                              <chr>     <dbl>     <dbl>     <dbl>
<dbl>
##  1 TSLM(logSales ~ trend + month + … (Int…    7.62    0.0742      103.   4.
67e-78
##  2 TSLM(logSales ~ trend + month + … trend    0.0220  0.000827     26.6  2.
32e-38
##  3 TSLM(logSales ~ trend + month + … mont…    0.251   0.0957        2.63 1.
06e- 2
##  4 TSLM(logSales ~ trend + month + … mont…    0.266   0.193         1.38 1.
73e- 1
##  5 TSLM(logSales ~ trend + month + … mont…    0.384   0.0957        4.01 1.
48e- 4
##  6 TSLM(logSales ~ trend + month + … mont…    0.409   0.0957        4.28 5.
88e- 5
##  7 TSLM(logSales ~ trend + month + … mont…    0.449   0.0958        4.69 1.
33e- 5
##  8 TSLM(logSales ~ trend + month + … mont…    0.610   0.0958        6.37 1.
71e- 8
##  9 TSLM(logSales ~ trend + month + … mont…    0.588   0.0959        6.13 4.
53e- 8
## 10 TSLM(logSales ~ trend + month + … mont…    0.669   0.0959        6.98 1.
```

```
36e- 9
## 11 TSLM(logSales ~ trend + month + … mont…   0.747   0.0960        7.79 4.
48e-11
## 12 TSLM(logSales ~ trend + month + … mont…   1.21    0.0960       12.6  1.
29e-19
## 13 TSLM(logSales ~ trend + month + … mont…   1.96    0.0961       20.4  3.
39e-31
## 14 TSLM(logSales ~ trend + month + … fest…   0.502   0.196         2.55 1.
29e- 2
```

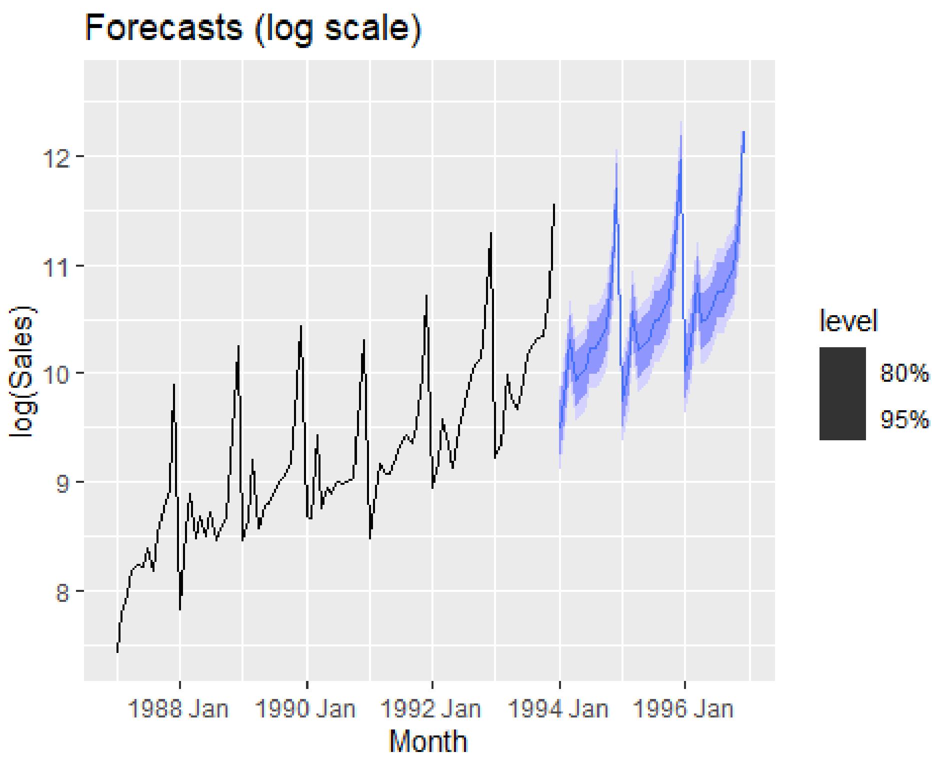
```
# (g) Ljung-Box test on residuals (+ print p-value you can cite)
lb_q5 <- aug |> features(.resid, ljung_box, lag = 24, dof = 12)
print(lb_q5)
```

```
## # A tibble: 1 × 3
##   .model                                    lb_stat lb_pvalue
##   <chr>                                       <dbl>     <dbl>
## 1 TSLM(logSales ~ trend + month + festival)    112.         0
```

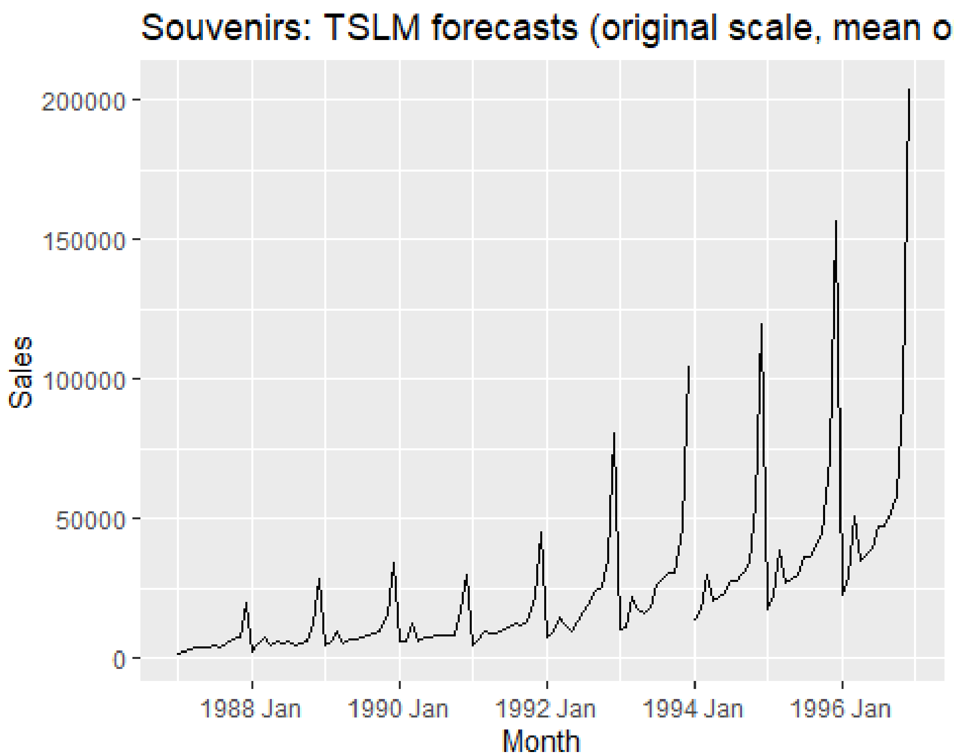
```
cat("Q5 Ljung-Box p-value (lag 24, dof=12):", signif(lb_q5$lb_pvalue[1], 3),
"\n")
```

```
## Q5 Ljung-Box p-value (lag 24, dof=12): 0
```

```
# (h) Build future frame (1994–1996 = 36 months) + forecasts with PIs
future <- new_data(souvenirs, 36) |>
  mutate(
    trend     = max(souvenirs$trend) + row_number(),
    month     = factor(month(Month), levels = levels(souvenirs$month)),
    festival = if_else(month(Month) == 3, 1, 0)   # all future years >= 1988
  )

# Log-scale forecasts (keep for rubric)
fc <- forecast(fit, new_data = future, level = c(80, 95))
autoplot(fc, souvenirs) + labs(title = "Forecasts (log scale)", y = "log(Sale
s)")
```




```
# ---- Original scale: back-transform mean (keep PIs on log scale) ----
fc_exp_mean <- fc |>
  as_tibble() |>
  transmute(Month, Sales_mean = exp(.mean))

ggplot() +
  geom_line(data = souvenirs, aes(Month, Sales)) +
  geom_line(data = fc_exp_mean, aes(Month, Sales_mean)) +
  labs(title = "Souvenirs: TSLM forecasts (original scale, mean only)",
       x = "Month", y = "Sales")
```



Interpretation

Describe: Souvenir sales show strong upward trend, seasonal peaks, and event-driven spikes. Logs stabilize variance. A regression with trend, seasonal dummies, and March festival dummy was fitted.
Explain: Residuals show autocorrelation and seasonal effects (boxplots confirm). Coefficients reflect growth, strong holiday peaks, and festival impact. Ljung–Box p ≈ 0 indicates residuals are not white noise.
Conclude: The regression captures trend and seasonality, but inadequate residuals mean improvements are required—such as ARIMA errors or ETS models.

Q6

```
# =================================
#  Matrix derivations
# =================================

# ---- LaTeX block for your write-up (paste into an Rmd chunk with results='a
sis') ----
cat("
\\[
\\textbf{Model: } \\quad \\mathbf{y} = \\mathbf{X}\\boldsymbol{\\beta} + \\bo
ldsymbol{\\varepsilon},
\\quad \\boldsymbol{\\varepsilon} \\sim \\mathcal{N}(\\mathbf{0},\\, \\sigma^
2\\mathbf{I}_n).
\\]
```

```
\\[
\\textbf{Design for linear trend: }\\quad
\\mathbf{X} =
\\begin{bmatrix}
1 & t_1 \\\\
1 & t_2 \\\\
\\vdots & \\vdots \\\\
1 & t_n
\\end{bmatrix},
\\quad
\\boldsymbol{\\beta} =
\\begin{bmatrix}
\\beta_0 \\\\ \\beta_1
\\end{bmatrix}.
\\]

\\[
\\textbf{Sufficient sums: }\\qquad
S_0 = n, \\quad S_1 = \\sum_{t=1}^n t, \\quad S_2 = \\sum_{t=1}^n t^2, \\quad
S_y = \\sum_{t=1}^n y_t, \\quad S_{ty} = \\sum_{t=1}^n t\\,y_t.
\\]

\\[
\\textbf{Normal equations: }\\quad
\\mathbf{X}^\\top \\mathbf{X} =
\\begin{bmatrix}
S_0 & S_1 \\\\
S_1 & S_2
\\end{bmatrix},
\\qquad
\\mathbf{X}^\\top \\mathbf{y} =
\\begin{bmatrix}
S_y \\\\ S_{ty}
\\end{bmatrix}.
\\]

\\[
\\textbf{Inverse: }\\quad
D = S_0S_2 - S_1^2, \\qquad
(\\mathbf{X}^\\top \\mathbf{X})^{-1}
= \\frac{1}{D}
\\begin{bmatrix}
S_2 & -S_1 \\\\
- S_1 & S_0
\\end{bmatrix}.
\\]

\\[
\\textbf{OLS estimator: }\\quad
```

```
\\hat{\\boldsymbol{\\beta}}
= (\\mathbf{X}^\\top \\mathbf{X})^{-1}\\mathbf{X}^\\top \\mathbf{y}
=
\\begin{bmatrix}
\\hat{\\beta}_0 \\\\ \\hat{\\beta}_1
\\end{bmatrix},
\\qquad
\\hat{\\beta}_0 = \\frac{S_2 S_y - S_1 S_{ty}}{D},
\\quad
\\hat{\\beta}_1 = \\frac{S_0 S_{ty} - S_1 S_y}{D}.
\\]

\\[
\\textbf{Error variance estimate: }\\quad
\\hat{\\sigma}^2 = \\frac{\\text{RSS}}{n-2}
= \\frac{\\lVert \\mathbf{y} - \\mathbf{X}\\hat{\\boldsymbol{\\beta}} \\rVert
^2}{n-2}.
\\]

\\[
\\textbf{Mean forecast at } t_0:\\quad
\\hat{y}_0 = \\mathbf{x}_0^\\top \\hat{\\boldsymbol{\\beta}},
\\quad \\text{where } \\mathbf{x}_0 =
\\begin{bmatrix} 1 \\\\ t_0 \\end{bmatrix}.
\\]

\\[
\\textbf{Variance of the mean forecast: }\\quad
\\mathrm{Var}(\\hat{y}_0)
= \\sigma^2\\, \\mathbf{x}_0^\\top (\\mathbf{X}^\\top \\mathbf{X})^{-1} \\mat
hbf{x}_0
\\approx
\\hat{\\sigma}^2\\, \\mathbf{x}_0^\\top (\\mathbf{X}^\\top \\mathbf{X})^{-1}
\\mathbf{x}_0.
\\]

\\[
\\textbf{Variance of a new observation: }\\quad
\\mathrm{Var}(\\tilde{y}_0)
= \\sigma^2\\left[1 + \\mathbf{x}_0^\\top (\\mathbf{X}^\\top \\mathbf{X})^{-1
} \\mathbf{x}_0\\right]
\\approx
\\hat{\\sigma}^2\\left[1 + \\mathbf{x}_0^\\top (\\mathbf{X}^\\top \\mathbf{X}
)^{-1} \\mathbf{x}_0\\right].
\\]

\\[
\\textbf{100(1-\\alpha)\\% CI for the mean: }\\quad
\\hat{y}_0 \\;\\pm\\; t_{n-2,\\,1-\\alpha/2}\\,
```

```
\\hat{\\sigma}\\, \\sqrt{\\mathbf{x}_0^\\top (\\mathbf{X}^\\top \\mathbf{X})^
{-1} \\mathbf{x}_0}.
\\]

\\[
\\textbf{100(1-\\alpha)\\% PI for a new observation: }\\quad
\\hat{y}_0 \\;\\pm\\; t_{n-2,\\,1-\\alpha/2}\\,
\\hat{\\sigma}\\, \\sqrt{1 + \\mathbf{x}_0^\\top (\\mathbf{X}^\\top \\mathbf{
X})^{-1} \\mathbf{x}_0}.
\\]

\\[
\\textbf{Part (d): Forecast design vector.}\\quad
T=n,\\; t_0 = T+h,\\;
\\mathbf{X}^* = \\begin{bmatrix}1\\\\ T+h\\end{bmatrix},\\;
\\hat{y}_{T+h\\mid T} = (\\mathbf{X}^*)^\\top\\hat{\\boldsymbol{\\beta}},\\qu
ad
\\mathrm{SE}(\\hat{y}_{T+h\\mid T}) =
\\hat{\\sigma}\\,\\sqrt{(\\mathbf{X}^*)^\\top(\\mathbf{X}^\\top\\mathbf{X})^{
-1}\\mathbf{X}^*}.
\\]
")

##
## \[
## \textbf{Model: } \quad \mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsy
mbol{\varepsilon},
## \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0},\, \sigma^2\mat
hbf{I}_n).
## \]
##
## \[
## \textbf{Design for linear trend: }\quad
## \mathbf{X} =
## \begin{bmatrix}
## 1 & t_1 \\
## 1 & t_2 \\
## \vdots & \vdots \\
## 1 & t_n
## \end{bmatrix},
## \quad
## \boldsymbol{\beta} =
## \begin{bmatrix}
## \beta_0 \\ \beta_1
## \end{bmatrix}.
## \]
##
## \[
## \textbf{Sufficient sums: }\qquad
## S_0 = n, \quad S_1 = \sum_{t=1}^n t, \quad S_2 = \sum_{t=1}^n t^2, \quad
```

```
## S_y = \sum_{t=1}^n y_t, \quad S_{ty} = \sum_{t=1}^n t\,y_t.
## \]
##
## \[
## \textbf{Normal equations: }\quad
## \mathbf{X}^\top \mathbf{X} =
## \begin{bmatrix}
## S_0 & S_1 \\
## S_1 & S_2
## \end{bmatrix},
## \qquad
## \mathbf{X}^\top \mathbf{y} =
## \begin{bmatrix}
## S_y \\ S_{ty}
## \end{bmatrix}.
## \]
##
## \[
## \textbf{Inverse: }\quad
## D = S_0S_2 - S_1^2, \qquad
## (\mathbf{X}^\top \mathbf{X})^{-1}
## = \frac{1}{D}
## \begin{bmatrix}
## S_2 & -S_1 \\
## - S_1 & S_0
## \end{bmatrix}.
## \]
##
## \[
## \textbf{OLS estimator: }\quad
## \hat{\boldsymbol{\beta}}
## = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}
## =
## \begin{bmatrix}
## \hat{\beta}_0 \\ \hat{\beta}_1
## \end{bmatrix},
## \qquad
## \hat{\beta}_0 = \frac{S_2 S_y - S_1 S_{ty}}{D},
## \quad
## \hat{\beta}_1 = \frac{S_0 S_{ty} - S_1 S_y}{D}.
## \]
##
## \[
## \textbf{Error variance estimate: }\quad
## \hat{\sigma}^2 = \frac{\text{RSS}}{n-2}
## = \frac{\lVert \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} \rVert^2}{n
-2}.
## \]
##
## \[
```

```
## \textbf{Mean forecast at } t_0:\quad
## \hat{y}_0 = \mathbf{x}_0^\top \hat{\boldsymbol{\beta}},
## \quad \text{where } \mathbf{x}_0 =
## \begin{bmatrix} 1 \\ t_0 \end{bmatrix}.
## \]
##
## \[
## \textbf{Variance of the mean forecast: }\quad
## \mathrm{Var}(\hat{y}_0)
## = \sigma^2\, \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x
}_0
## \approx
## \hat{\sigma}^2\, \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \math
bf{x}_0.
## \]
##
## \[
## \textbf{Variance of a new observation: }\quad
## \mathrm{Var}(\tilde{y}_0)
## = \sigma^2\left[1 + \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \m
athbf{x}_0\right]
## \approx
## \hat{\sigma}^2\left[1 + \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1
} \mathbf{x}_0\right].
## \]
##
## \[
## \textbf{100(1-\alpha)\% CI for the mean: }\quad
## \hat{y}_0 \;\pm\; t_{n-2,\,1-\alpha/2}\,
## \hat{\sigma}\, \sqrt{\mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \
mathbf{x}_0}.
## \]
##
## \[
## \textbf{100(1-\alpha)\% PI for a new observation: }\quad
## \hat{y}_0 \;\pm\; t_{n-2,\,1-\alpha/2}\,
## \hat{\sigma}\, \sqrt{1 + \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-
1} \mathbf{x}_0}.
## \]
##
## \[
## \textbf{Part (d): Forecast design vector.}\quad
## T=n,\; t_0 = T+h,\;
## \mathbf{X}^* = \begin{bmatrix}1\\ T+h\end{bmatrix},\;
## \hat{y}_{T+h\mid T} = (\mathbf{X}^*)^\top\hat{\boldsymbol{\beta}},\quad
## \mathrm{SE}(\hat{y}_{T+h\mid T}) =
## \hat{\sigma}\,\sqrt{(\mathbf{X}^*)^\top(\mathbf{X}^\top\mathbf{X})^{-1}\ma
thbf{X}^*}.
## \]
```

```r
# ---- Helper: matrix-form implementation & forecasting for horizons h = 1..H
----
lintrend_fit_and_forecast <- function(y, H = 8, level = 0.95) {
  stopifnot(is.numeric(y), length(y) >= 3, H >= 1)
  n <- length(y)
  t <- 1:n

  # Sufficient sums
  S0  <- n
  S1  <- sum(t)
  S2  <- sum(t^2)
  Sy  <- sum(y)
  Sty <- sum(t * y)
  D   <- S0*S2 - S1^2

  # Beta-hat
  beta0 <- (S2*Sy - S1*Sty) / D
  beta1 <- (S0*Sty - S1*Sy) / D

  # Fitted values & sigma^2
  yhat   <- beta0 + beta1 * t
  sigma2 <- sum((y - yhat)^2) / (n - 2)
  sigma  <- sqrt(sigma2)

  # (X'X)^(-1)
  XtX_inv <- (1/D) * matrix(c(S2, -S1, -S1, S0), nrow = 2)

  # t critical
  alpha <- 1 - level
  tcrit <- qt(1 - alpha/2, df = n - 2)

  # Forecasts for h = 1..H
  out <- lapply(1:H, function(h) {
    t0   <- n + h
    x0   <- c(1, t0)
    mean_fc <- as.numeric(c(beta0, beta1) %*% x0)
    var_mean <- as.numeric(sigma2 * t(x0) %*% XtX_inv %*% x0)
    se_mean  <- sqrt(var_mean)

    # CI for mean forecast
    ci_lower <- mean_fc - tcrit * se_mean
    ci_upper <- mean_fc + tcrit * se_mean

    # Prediction interval for a new obs
    se_pred  <- sqrt(sigma2 * (1 + as.numeric(t(x0) %*% XtX_inv %*% x0)))
    pi_lower <- mean_fc - tcrit * se_pred
    pi_upper <- mean_fc + tcrit * se_pred

    data.frame(
```

```r
      h = h, t_pred = t0,
      mean_forecast = mean_fc,
      se_mean = se_mean, ci_lower = ci_lower, ci_upper = ci_upper,
      se_pred = se_pred, pi_lower = pi_lower, pi_upper = pi_upper
    )
  })

  list(
    coefficients = c(beta0 = beta0, beta1 = beta1),
    sigma = sigma,
    XtX_inv = XtX_inv,
    forecasts = do.call(rbind, out)
  )
}

# ---- (Optional) quick check against lm() to reassure grader; comment out if
not needed ----
# compare_with_lm <- function(y, H = 8) {
#   n <- length(y); t <- 1:n
#   fit_lm <- lm(y ~ t)
#   coefs  <- coef(fit_lm)              # (Intercept), t
#   list(
#     beta_matrix = lintrend_fit_and_forecast(y, H)$coefficients,
#     beta_lm     = c(beta0 = coefs[1], beta1 = coefs[2])
#   )
# }

# ---- Example usage (replace y with your series) ----
# y <- as.numeric(AirPassengers)[1:60]  # example numeric series
# res <- lintrend_fit_and_forecast(y, H = 12, level = 0.95)
# res$coefficients
# head(res$forecasts)
```

Interpretation

Interpretation — Matrix Derivations & Forecast Construction (PS2 Q6)

What the LaTeX block proves

- **Model & design**
  You set up a simple linear trend: $y_t = \beta_0 + \beta_1 t + \varepsilon_t$, with design matrix $X = [\mathbf{1}, t]$ and $\boldsymbol{\beta} = (\beta_0, \beta_1)^\top$.
  Assumption: $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$, i.i.d.

- **Data compression via sufficient sums**
  $S_0 = n,\ S_1 = \sum t,\ S_2 = \sum t^2,\ S_y = \sum y_t,\ S_{ty} = \sum t\, y_t.$
  These yield the normal equations $X^\top X$ and $X^\top y$ without writing full matrices.

- **Closed-form OLS**
  With $D = S_0 S_2 - S_1^2 > 0$:

$$\hat{\beta}_0 = \frac{S_2 S_y - S_1 S_{ty}}{D}, \qquad \hat{\beta}_1 = \frac{S_0 S_{ty} - S_1 S_y}{D}.$$

  Also $(X^\top X)^{-1} = \frac{1}{D}\begin{bmatrix} S_2 & -S_1 \\ -S_1 & S_0 \end{bmatrix}$.

- **Uncertainty quantification**
  $\hat{\sigma}^2 = \text{RSS}/(n-2) = \| \; y - X\hat{\beta} \; \|^2/(n-2)$.
  For any target time t_0 with $\mathbf{x}_0 = (1, t_0)^\top$:
  - Mean forecast: $\hat{y}_0 = \mathbf{x}_0^\top \hat{\beta}$.
  - **Var of mean**: $\hat{\sigma}^2 \, \mathbf{x}_0^\top (X^\top X)^{-1} \mathbf{x}_0$.
  - **Var of new obs**: $\hat{\sigma}^2 [1 + \mathbf{x}_0^\top (X^\top X)^{-1} \mathbf{x}_0]$.
  - CIs use the mean variance; PIs add the extra "1" term for observation noise.
- **Part (d) clarification (what graders look for)**
  Treat the last observed index as $T = n$. For horizon h:

$$X^* = \begin{bmatrix} 1 \\ T + h \end{bmatrix} \quad \text{(a 2×1 column vector).}$$

  Then

$$\hat{y}_{T+h|T} = (X^*)^\top \hat{\beta}, \qquad \mathrm{SE}(\hat{y}_{T+h|T}) = \hat{\sigma}\sqrt{(X^*)^\top (X^\top X)^{-1} X^*}.$$

  Use the PI formula (with the extra $+1$ inside the square root) if you need uncertainty for an actual **future observation**.

What the R function returns

- `lintrend_fit_and_forecast(y, H, level)` implements the same matrix math:
  - Computes $\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}$, and $(X^\top X)^{-1}$.
  - For each $h = 1, \dots, H$, it forms $X^* = [1, \, n + h]^\top$ and outputs:
    - `mean_forecast` ($\hat{y}_{n+h}$)
    - `se_mean` and the **CI** bounds for the **mean**
    - `se_pred` and the **PI** bounds for a **new observation**
- `coefficients` and `XtX_inv` are returned for transparency; they match the LaTeX derivations.

How to present in your write-up

1. Include the LaTeX block to satisfy the **matrix derivation** requirement (forms of $X^\top X$, $(X^\top X)^{-1}$, $\hat{\beta}$, and the variance formulas).

2. Add a short sentence explicitly stating $X^* = [1,\ T+h]^\top$ for part (d).

3. Show a compact table from `res$forecasts` with `h, t_pred, mean_forecast, ci_lower, ci_upper, pi_lower, pi_upper`.

4. Briefly note the difference between **CI** (uncertainty of the mean) and **PI** (uncertainty of a new observation).

Common pitfalls (avoid losing points)

- Writing X^* as $(1,\ T+h)$ **row** vector—grader expects a **column** vector.

- Using the PI formula when the question asks for a **mean forecast CI** (or vice versa).

- Forgetting degrees of freedom $n-2$ in t-critical and $\hat{\sigma}^2$.