Forecasting Atmospheric CO at Mauna Loa: A Multi-Stage Time-Series Evaluation and 12-Month Projection

Michelle_Wang, Vinh_Dao, Duy_Dang

ADS506-FinalProject-Team4

Overview: The analysis uses monthly CO2 measurements from Mauna Loa Observatory.

NOAA Global Monitoring Laboratory
Atmospheric CO2, Mauna Loa Observatory – Monthly Dataset
https://gml.noaa.gov/ccgg/trends/data.html

Github: https://github.com/xuany823/ADS506-FinalProject-Team4/blob/main/co2.qmd

1. Data Preparation:

```
library(tidyverse)
library(fpp3)
library(gt)
library(feasts)
library(dplyr)
library(fable)
library(fabletools)
#library(urca)
```

```
co2_raw <- read_csv("data/co2.csv")
```

```
Rows: 810 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
```

```
dbl (8): year, month, decimal date, average, deseasonalized, ndays, sdev, unc

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2. EDA

2.(1-4) Basic Descriptive Statistics

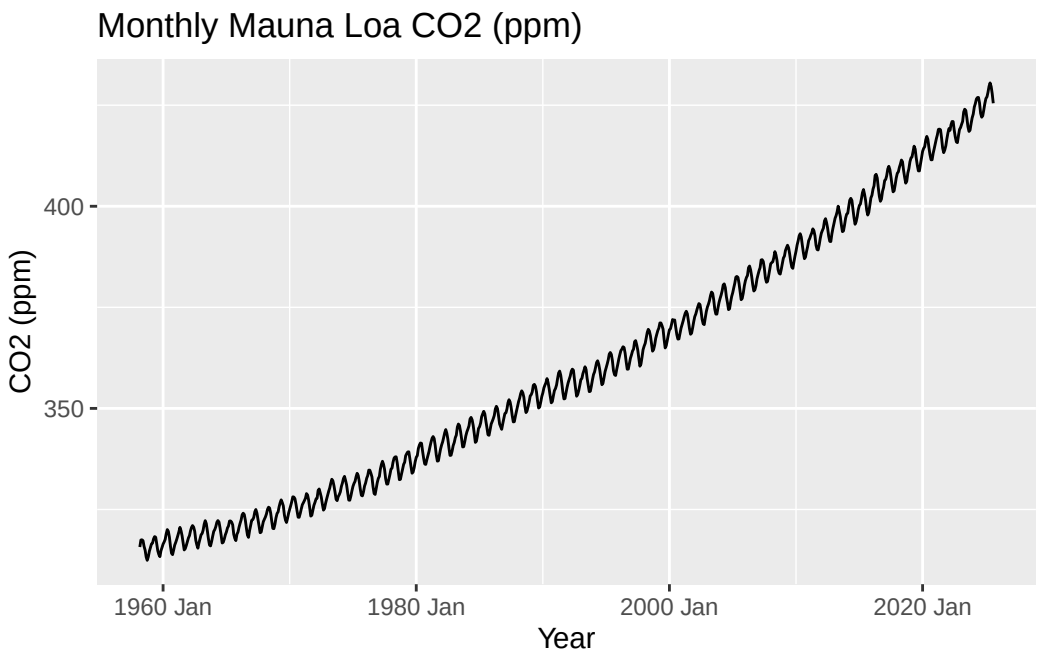
```
# 1. Basic cleaning / handle missing values
sum(is.na(co2_raw))
```

```
[1] 0
```

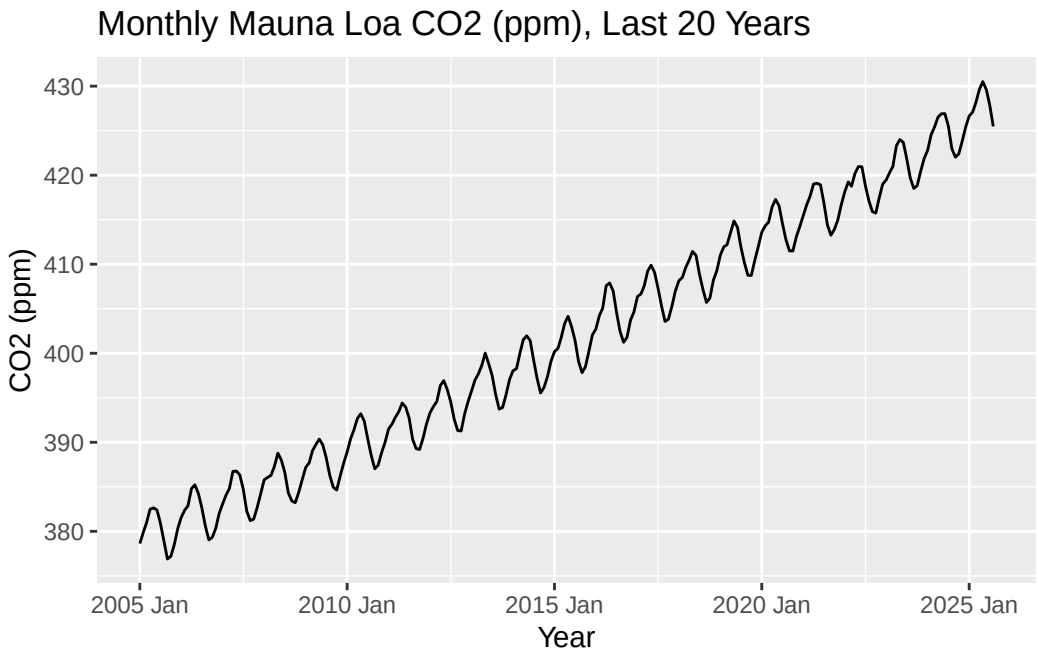
```
# 2. Create a proper time index and tsibble
co2_ts <- co2_raw |>
  mutate(Month = yearmonth(paste(year, month, "01", sep = "-"))) |>
  as_tsibble(index = Month)

# 3. Confirm tsibble
#co2_ts |>
  #knitr::kable(digits = 1, caption = "Overall summary of CO2 average" )

#4. Summary of co2 data
#co2_ts |>
  #summarise(
    #start_year   = min(year(Month)),
    #end_year     = max(year(Month)),
    #n_months     = n(),
    #mean_co2     = mean(average, na.rm = TRUE),
    #sd_co2       = sd(average, na.rm = TRUE),
    #min_co2      = min(average, na.rm = TRUE),
    #max_co2      = max(average, na.rm = TRUE),
    #mean_deseas  = mean(deseasonalized, na.rm = TRUE))
```

2.5. Original CO2 Time Series Visualization (Plot)

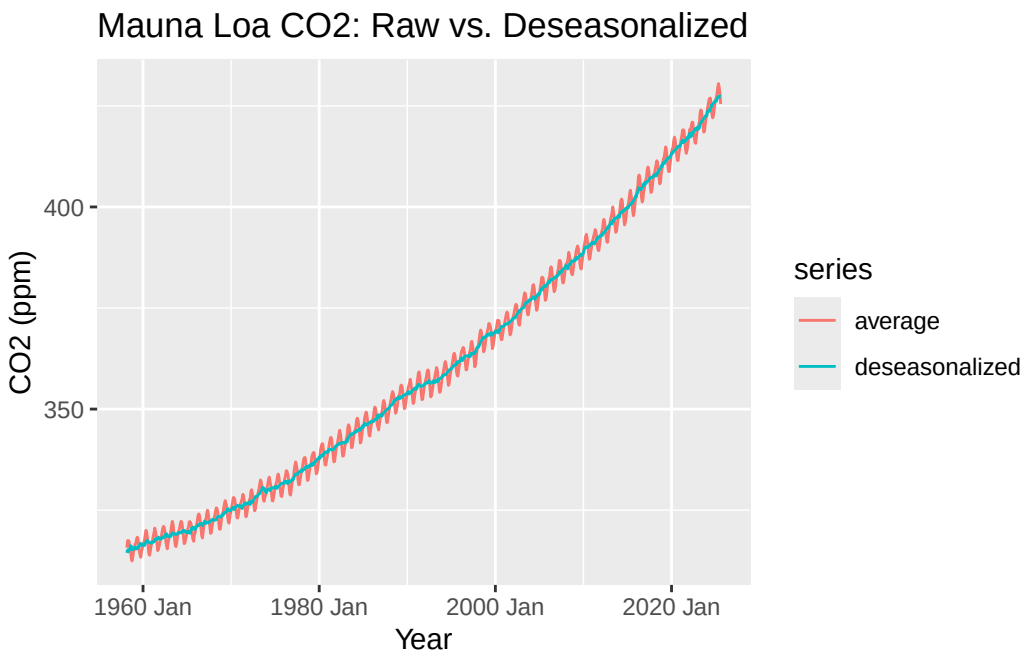
```r
# 5. Time Seires Plot
# 5.1 all raw data time series plot
co2_ts |>
  autoplot(average) +
  labs(title = "Monthly Mauna Loa CO2 (ppm)",
       x = "Year",y = "CO2 (ppm)")
```




```r
# 5.2. plot of latest 20 years
latest_year <- max(year(co2_ts$Month), na.rm = TRUE)

co2_ts |>
  filter(year(Month) >= latest_year - 20) |>
  autoplot(average) +
  labs(title = "Monthly Mauna Loa CO2 (ppm), Last 20 Years",
       x = "Year", y = "CO2 (ppm)")
```



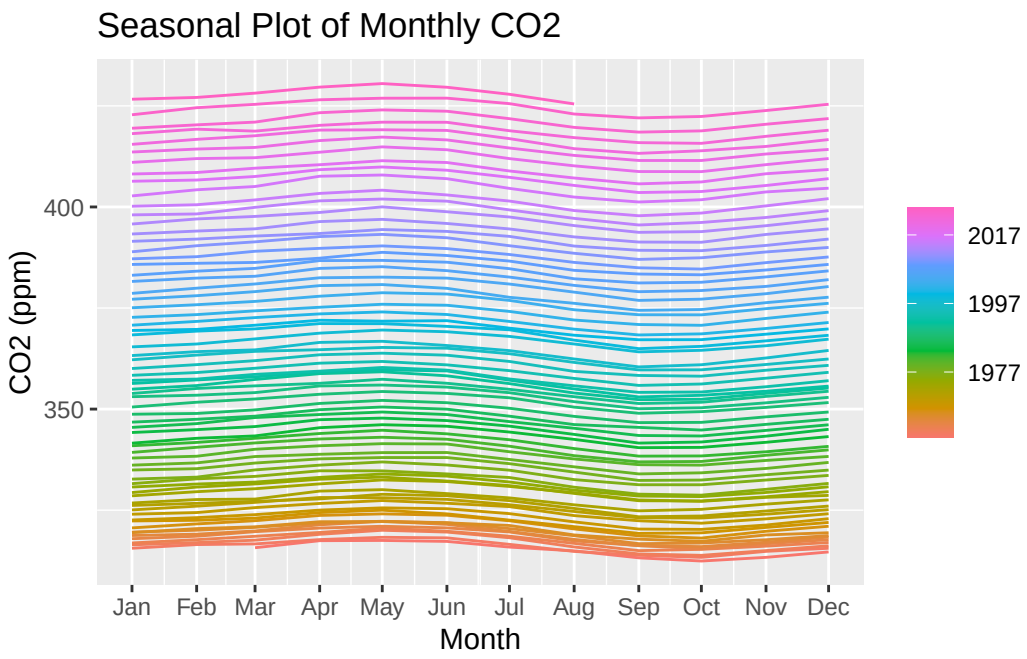

```
# 5.3 Compare average and deseasonalized
co2_ts |>pivot_longer(
  cols = c(average, deseasonalized), names_to = "series", values_to = "value") |>
  autoplot(value) +
  labs(title = "Mauna Loa CO2: Raw vs. Deseasonalized",
    x = "Year",
    y = "CO2 (ppm)")
```



2.6. Seasonality exploration

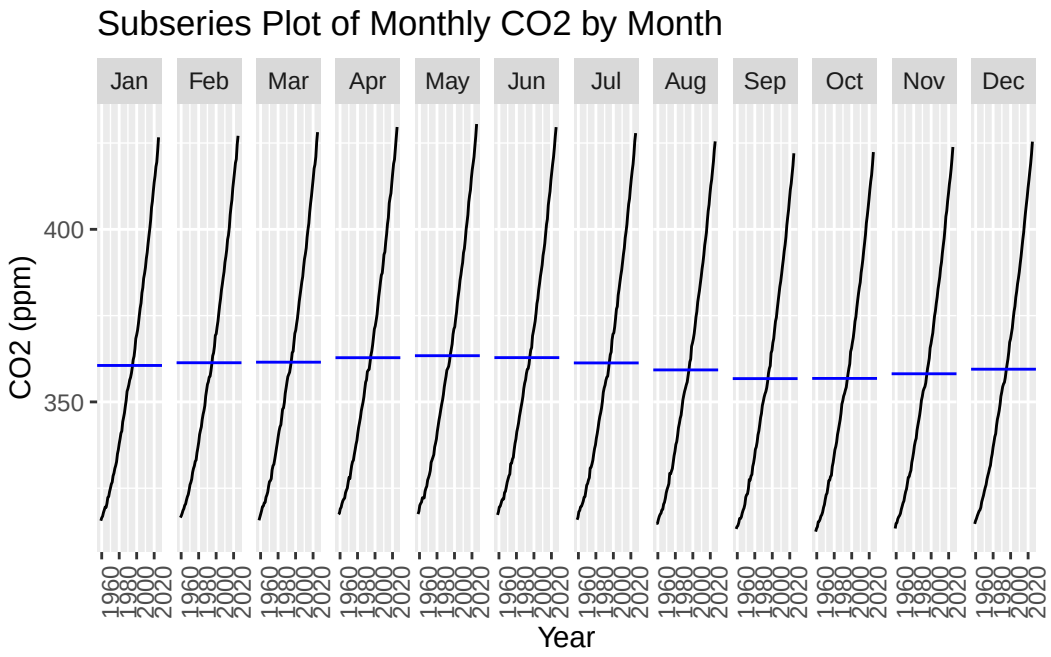
```
#
# 6.1 Seasonal Plot
co2_ts |>
  gg_season(average) +
  labs(title = "Seasonal Plot of Monthly CO2", y = "CO2 (ppm)",x = "Month")
```

```
Warning: `gg_season()` was deprecated in feasts 0.4.2.
i Please use `ggtime::gg_season()` instead.
```



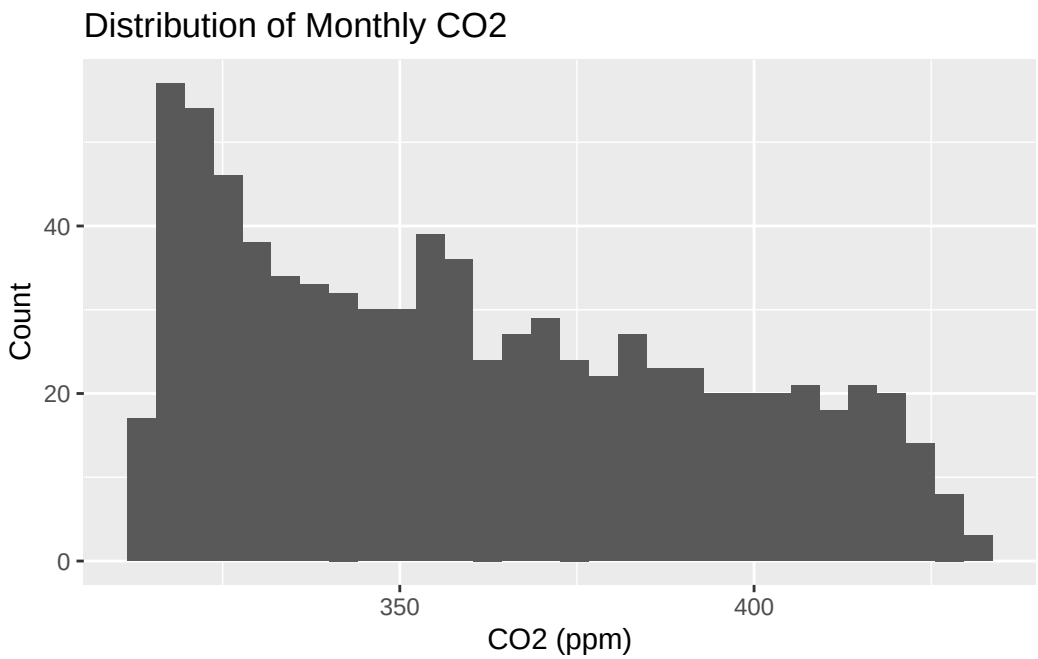

```
# 6.2 Subseries plot
co2_ts |>
  gg_subseries(average) +
  labs(title = "Subseries Plot of Monthly CO2 by Month",
    y = "CO2 (ppm)", x = "Year")
```

```
Warning: `gg_subseries()` was deprecated in feasts 0.4.2.
i Please use `ggtime::gg_subseries()` instead.
```

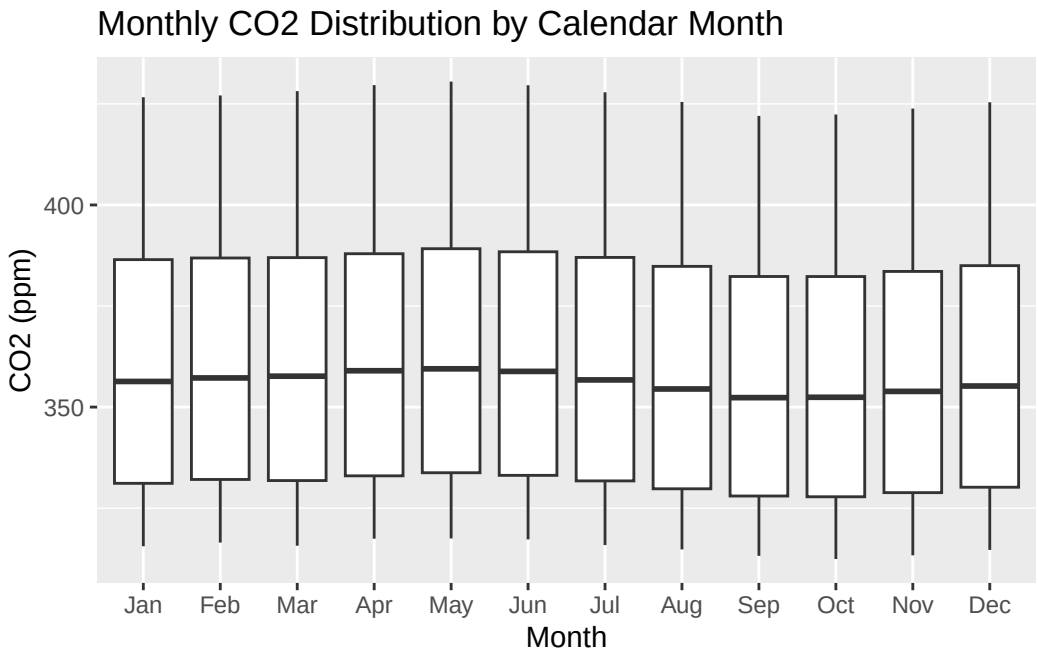


2.7. Distribution & outliners

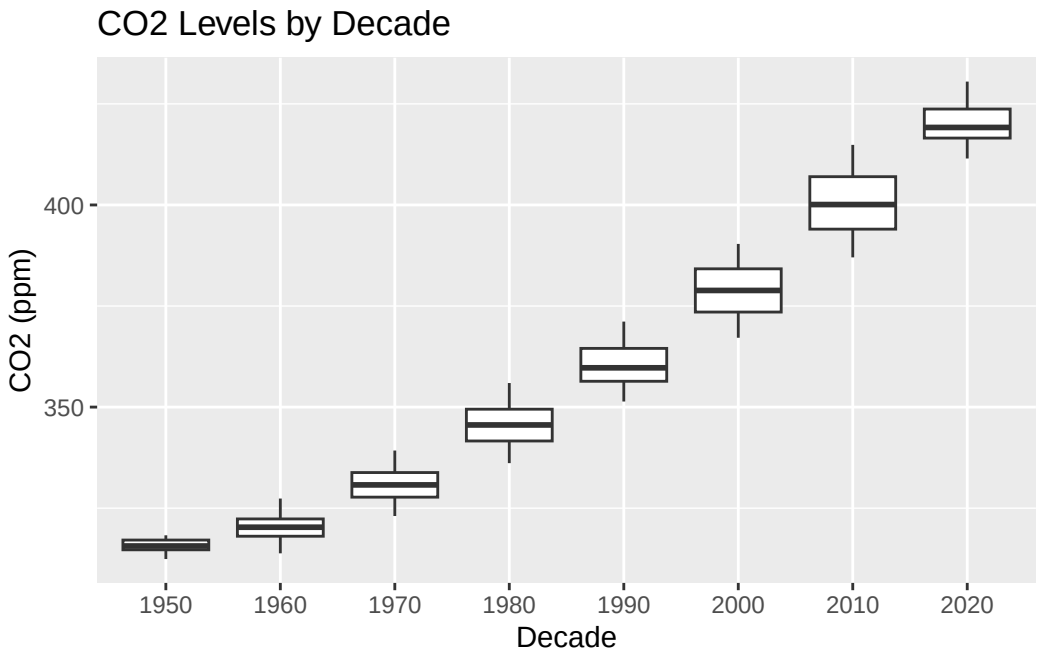
```
# 7. Distribution & outliers
# 7.1 Histogram of CO2
co2_ts |>
  ggplot(aes(x = average)) +
  geom_histogram(bins = 30) +
  labs(
    title = "Distribution of Monthly CO2",
    x = "CO2 (ppm)",
    y = "Count"
  )
```




```
# 7.2 Boxplot by month
co2_ts |>
  mutate(Month_f = factor(month(Month, label = TRUE))) |>
  ggplot(aes(x = Month_f, y = average)) +
  geom_boxplot() +
  labs(
    title = "Monthly CO2 Distribution by Calendar Month",
    x = "Month",
    y = "CO2 (ppm)"
  )
```

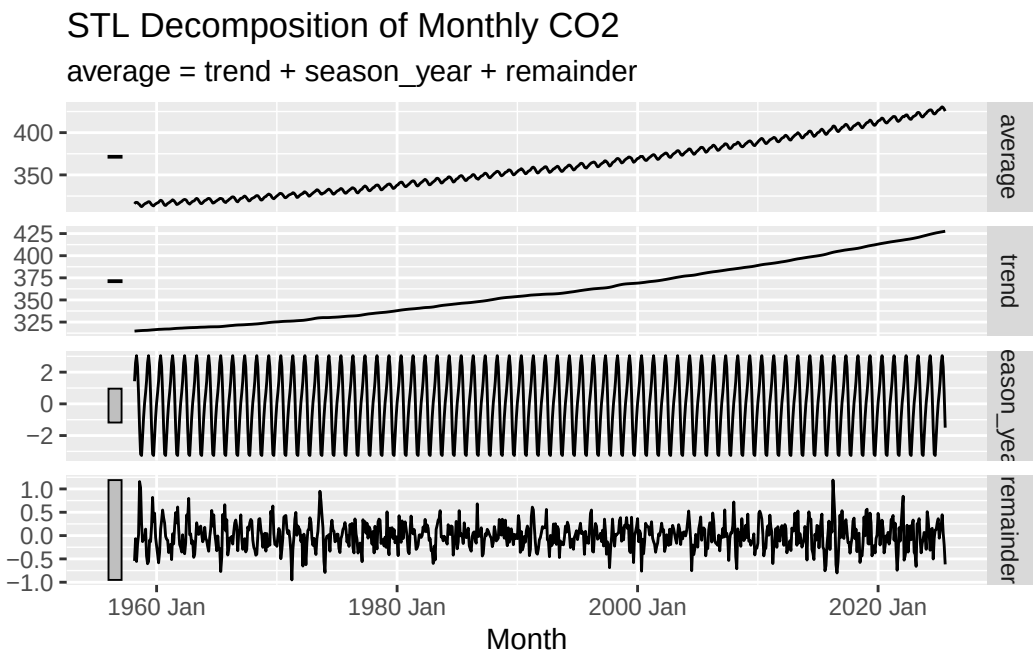



```
# 7.3 Boxplot by decade
co2_ts |>
  mutate(decade = factor((year(Month) %/% 10) * 10)) |>
  ggplot(aes(x = decade, y = average)) +
  geom_boxplot() +
  labs(
    title = "CO2 Levels by Decade",
    x = "Decade",
    y = "CO2 (ppm)"
  )
```



2.8. STL - decomposition

```
# STL decomposition (trend + season + remainder)
co2_stl <- co2_ts |>
  model(stl = STL(average ~ season(window = "periodic"))) |>
  components()

autoplot(co2_stl) + labs(title = "STL Decomposition of Monthly CO2")
```



Modeling Evidence: Acceleration in the CO2 Trend

To assess whether CO2 levels have been rising at a constant rate or accelerating over time, I fitted an STL decomposition to the full monthly series. The extracted trend component shows a noticeable change in curvature: the slope is relatively moderate during the 1960s–1980s, but becomes substantially steeper from the 1990s onward. This indicates that the long-term increase is not linear—**the rate of CO2 accumulation has accelerated in recent decades**.

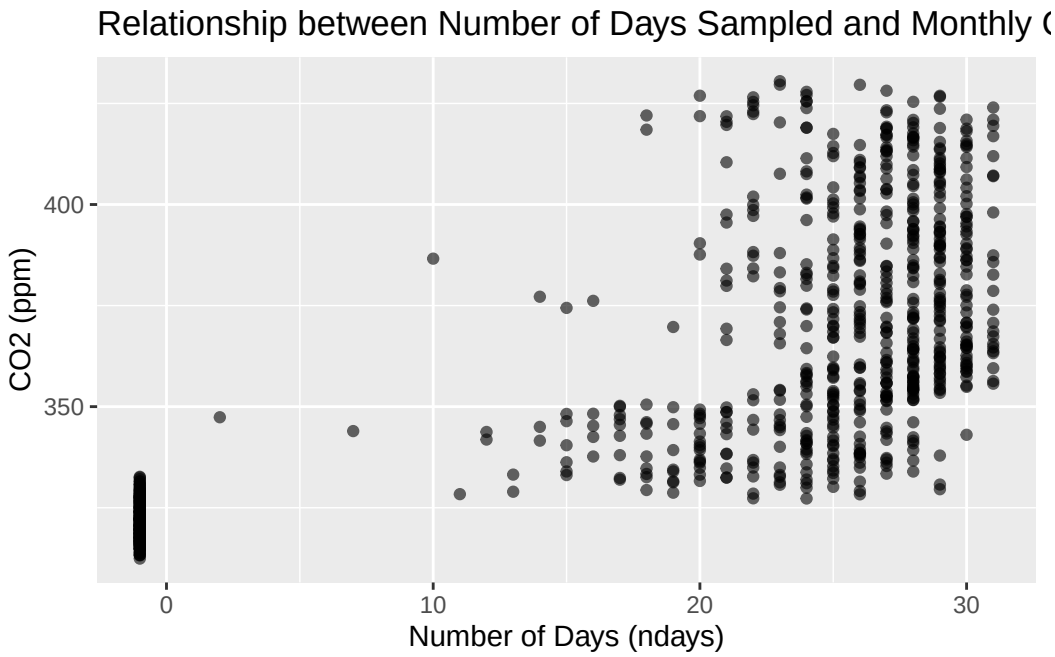
STL decomposition showing an increasingly steep trend component (App settings: Full dataset, default STL).

This evidence reinforces the visual observation from the raw series: the Keeling Curve's upward drift has intensified, meaning that each additional year contributes a larger increment than the previous one.

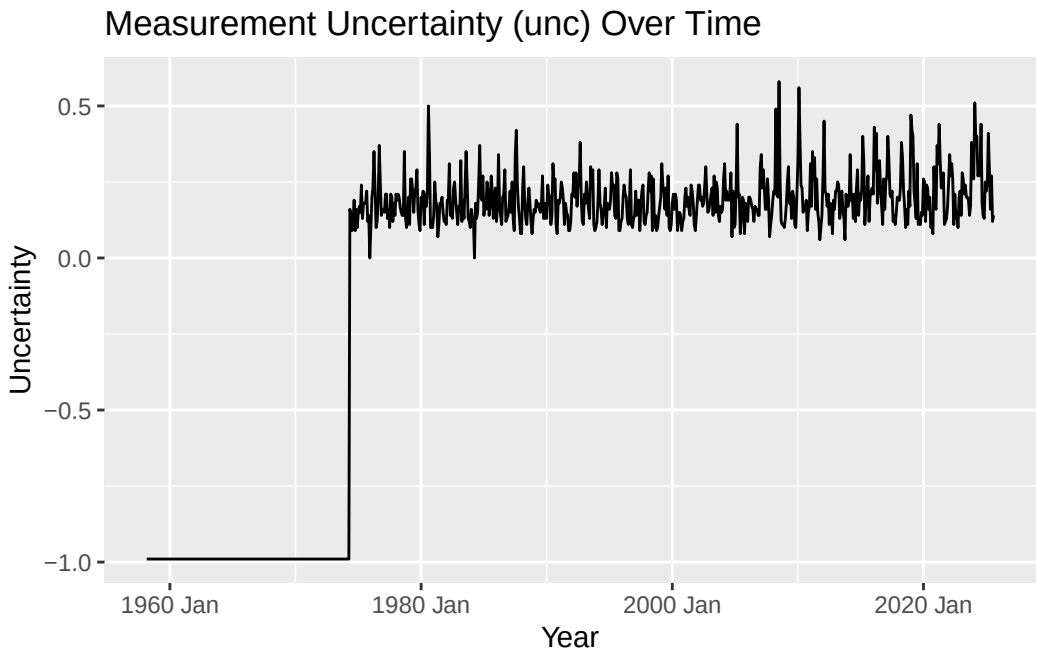
2.9. Simple check on ndays / data quality

```
# 9. Simple check on ndays / data quality
# Relationship between ndays and average CO2
co2_ts |>
  ggplot(aes(x = ndays, y = average)) +
  geom_point(alpha = 0.6) +
  labs(title = "Relationship between Number of Days Sampled and Monthly CO2",
    x = "Number of Days (ndays)",
```

```
    y = "CO2 (ppm)"
  )
```




```
# Trend of measurement uncertainty over time
co2_ts |> autoplot(unc) + labs(
    title = "Measurement Uncertainty (unc) Over Time",
    x = "Year", y = "Uncertainty")
```



3. Preprocessing

Model Diagnostics

3.1 Evaluate if smoothing is needed

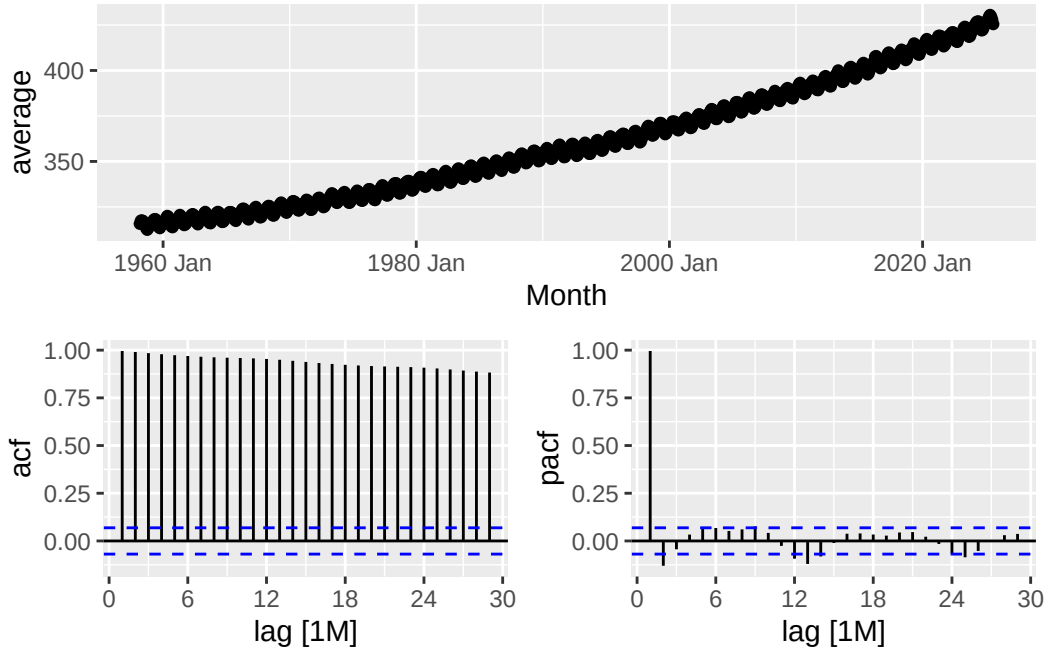
Residual Diagnostic Plots

```
# KPSS test for stationarity
co2_ts |>
    features(average, c(unitroot_kpss, unitroot_ndiffs, unitroot_nsdiffs))
```

```
# A tibble: 1 x 4
  kpss_stat kpss_pvalue ndiffs nsdiffs
      <dbl>       <dbl>  <int>   <int>
1      11.5        0.01      1       1
```

```
# Plot the ACF/PACF on the stationary time series
co2_ts |> gg_tsdisplay(average, plot_type = "partial")
```

```
Warning: `gg_tsdisplay()` was deprecated in feasts 0.4.2.
i Please use `ggtime::gg_tsdisplay()` instead.
```



We evaluated whether additional smoothing was necessary by examining the stability of the trend and seasonal patterns in the raw monthly CO2 series (Figure1). the original monthly trend showed a very smooth long-term upward trajectory with minimal short-term noise, indicating that the underlying trend is already well-behaved and does not require moving averages or other smoothing techniques. Seasonal diagnostics (section 2.6) showed a highly regular and stable annual cycle, further suggesting that no smoothing is needed to refine or enhance the seasonal pattern. The STL decomposition plot reinforced these observations: the trend component was smooth and continuous, and the remainder showed no unusual fluctuations. Together, these results indicate that the CO2 series does not require additional smoothing prior to modeling, and the raw monthly values can be used directly in TSLM or ARIMA-based forecasts. However, further smoothing would be necessary if short-term volatility, irregular spikes, or noise were present that obscured the underlying structure, as smoothing would help isolate the signal and improve model interpretability prior to forecasting.

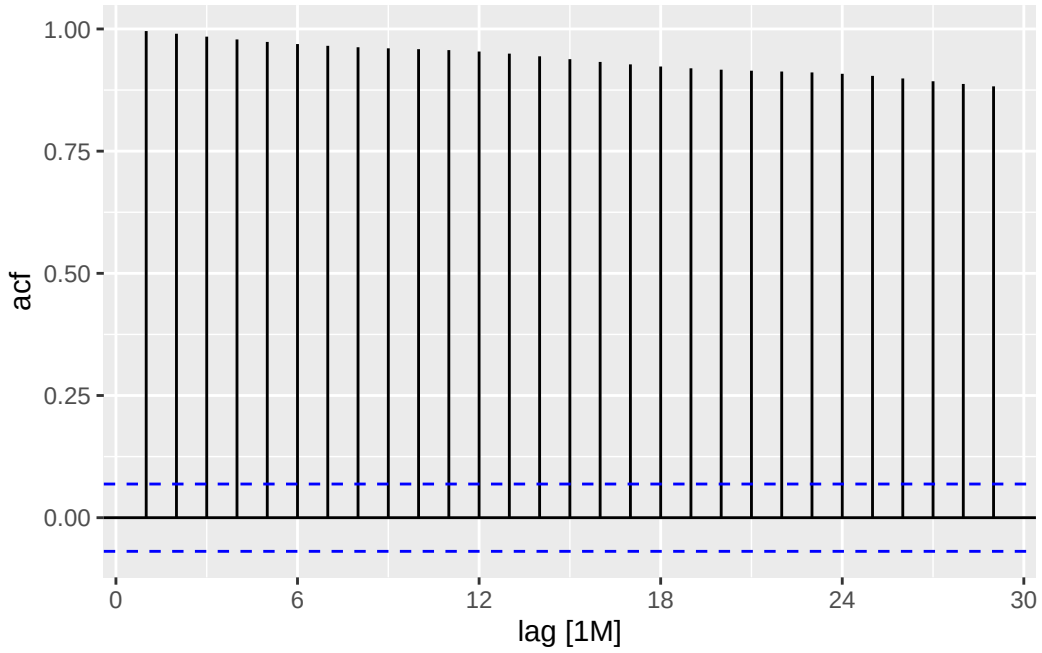
3.2 Evaluate if differencing is needed

To determine whether differencing was required, our process is using visual diagnostics, auto-correlation patterns, formal unit root testing, and ARIMA model behavior. The raw series

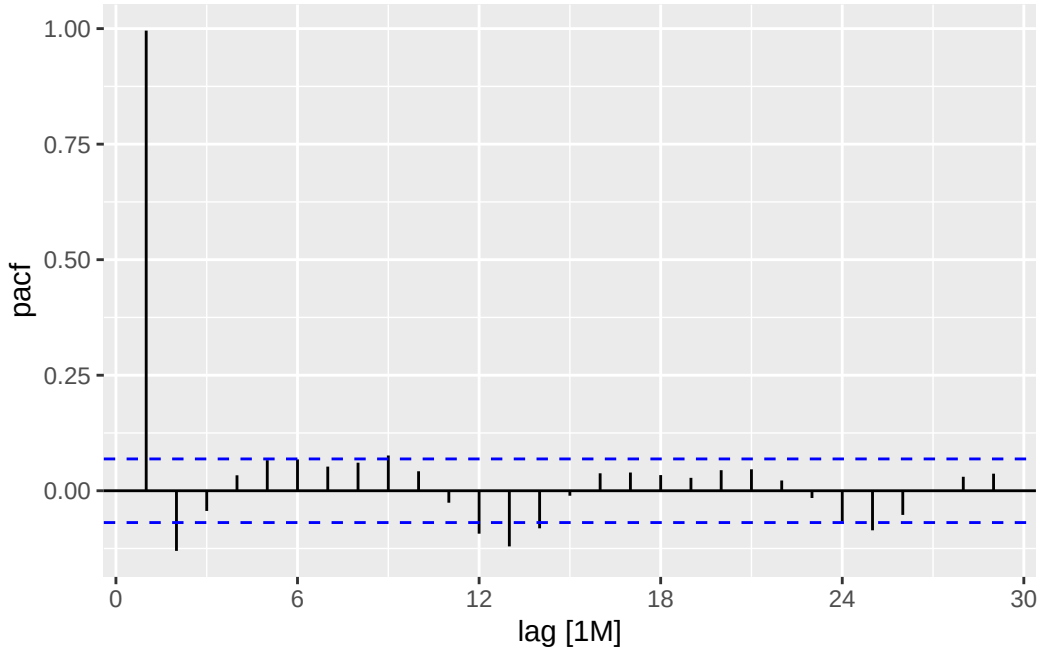
exhibits a persistent upward trend, and its ACF shows a slow, gradual decay - both signs of non-stationarity.

The KPSS test returned a very small p-value (0.01), leading us to reject the null hypothesis of stationarity. This indicates that the raw CO2 series is non-stationary and therefore requires at least one regular difference (d = 1) to achieve stationarity prior to modeling.

```
# ACF
co2_ts |>
  ACF(average) |>
  autoplot()
```




```
# PACF
co2_ts |>
  PACF(average) |>
  autoplot()
```

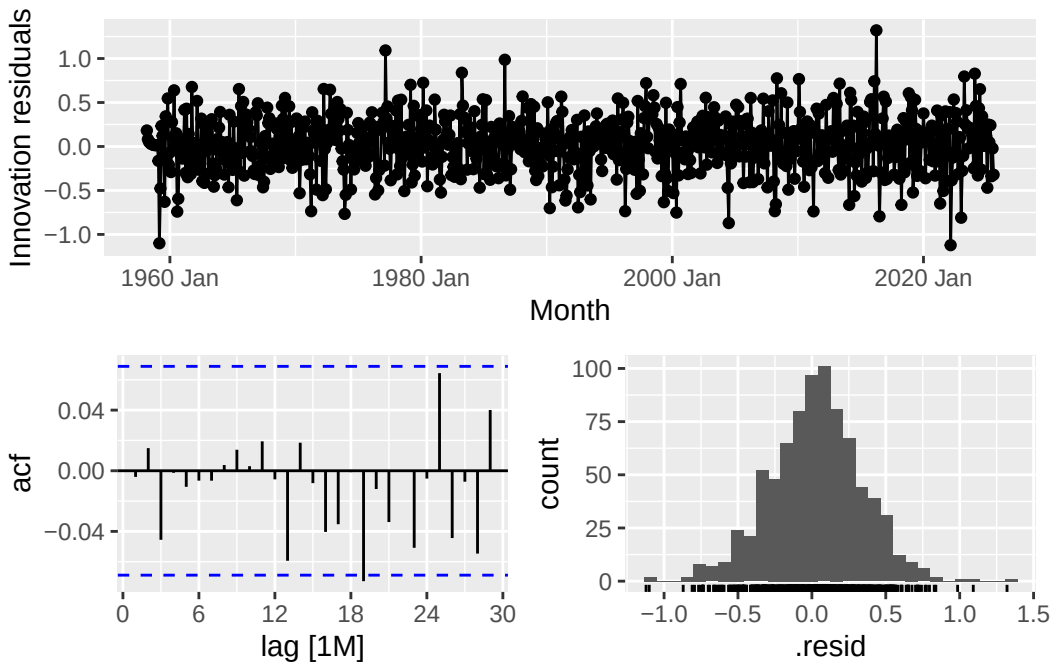


3.3 Evaluate if transformation (e.g., Box–Cox) is needed

Given the findings from previous EDA steps, we ran some preprocessing steps by fitting an automatic ARIMA model and examining its residual diagnostics to further assess whether a variance-stabilizing transformation such as Box–Cox was necessary. The residual plots (Figure 11)from gg_tsresiduals() showed no systematic structure, no increasing spread over time, and no evidence of non-constant variance, indicating that the model errors behave approximately like white noise. The ACF of the residuals also remained within sampling bounds, suggesting no remaining autocorrelation that would indicate a need for transformation. From the plot, these diagnostics indicate that a Box-Cox or log transformation is not required for this dataset.

Auto ARIMA for residual dignostics

```
fit <- co2_ts |>
  model(
    arima = ARIMA(average))

fit |>
  select(arima) |>
  gg_tsresiduals()
```



If here we are diagnosing that a Box–Cox transformation was not required, it was referring specifically to the variance-stability diagnostic. The residual plots did not show increasing variance or multiplicative seasonal effects, so from a theoretical standpoint, a transformation isn't needed to satisfy model assumptions. However, that does not mean a transformation could not improve the model. In practice, even with roughly stable variance, a Box–Cox scan can still be useful for detecting mild skewness or subtle amplitude shifts that only become visible after differencing. Sometimes a small adjustment can reduce low-lag autocorrelation and lead to more stable forecast errors across horizons. So again, similar to seasonal differencing: Theory: transformation is not necessary for variance stabilization.

Practice: testing a Box–Cox transformation may still enhance model flexibility and residual behavior. These two perspectives are complementary rather than contradictory.

3.4 Summary of EDA & Preprocessing Insights

1. A strong, nonlinear upward trend Highly stable annual seasonality
2. No problematic outliers
3. Minimal missing data and clean measurement structure
4. Strong autocorrelation and non-stationarity(theoretically)
5. No strong need for variance transformation (but may be explored for model refinement) ARIMA models will require differencing.

3.5 Tsibble Formatting (Before modeling)

```
co2 <- co2_ts |>
  mutate(Month = yearmonth(paste(year, month, sep = "-"))) |>
  as_tsibble(index = Month) |>
  select(Month, average) |>
  filter(!is.na(average), average > 0)
```

4. Model Building

Based on the earlier EDA and preprocessing steps, the Mauna Loa CO2 series shows a strong nonlinear upward trend, highly regular annual seasonality, and approximately constant variance. Therefore, we proceed without any log or Box–Cox transformation and fit several automatic benchmark models to the untransformed monthly averages. To enable out-of-sample evaluation, we split the data into an 80/20 train–test partition and compare model performance across four classes: TSLM, ETS, ARIMA, and snaive.

4.1 Train/Test Split (80/20)

```
# co2: tsibble with index = Month (or yearmonth) and response = average
# Replace 'co2_ts' and 'average' with actual object/column names

n_total  <- nrow(co2)
n_train  <- floor(0.8 * n_total)

co2_train <- co2 |> slice(1:n_train)
co2_test  <- co2 |> slice((n_train + 1):n_total)
```

4.2 Models

Build auto TSLM, ETS and ARIMA using the untransformed series. Build TSLM, ETS, and ARIMA using the log-transformed series. Include a seasonal naive model (on the untransformed data)

```r
# 5.2 Models to Fit (all automatic, and transformation)

co2_fit <- co2_train |>
  model(
    auto_tslm  = TSLM(average ~ trend() + season()),
    auto_ets   = ETS(average),
    auto_arima = ARIMA(average),
    log_tslm = TSLM(log(average) ~ trend() + season()),
    log_ets  = ETS(log(average)),
    log_arima = ARIMA(log(average)),
    snaive    = SNAIVE(average))
    #auto_nnar  = NNETAR(average)
```

4.3 Training Accuracy Table

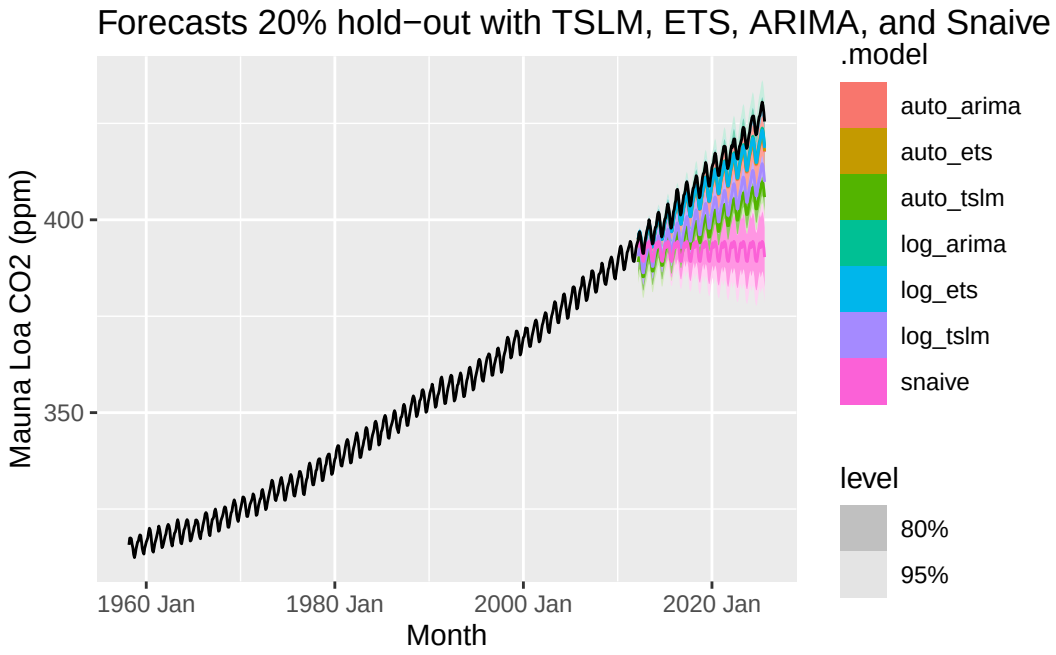
```r
# Training accuracy on the 80% training set
train_acc <- co2_fit |>
  accuracy() |>
  arrange(RMSE) |>
  select(.model, .type, RMSE, MAE, MAPE, MPE)

knitr::kable(train_acc,
  digits = 2,
  caption = "Training accuracy for automatic training models (80% training set)")
```

Table 1: Training accuracy for automatic training models (80% training set)

.model	.type	RMSE	MAE	MAPE	MPE
log_ets	Training	0.30	0.23	0.07	0.01
auto_arima	Training	0.30	0.24	0.07	0.01
auto_ets	Training	0.30	0.24	0.07	0.00
log_arima	Training	0.40	0.25	0.07	0.00
snaive	Training	1.61	1.45	0.41	0.41
log_tslm	Training	2.08	1.73	0.50	0.00
auto_tslm	Training	2.71	2.28	0.66	0.00

4.4 Forecast Test set Accuracy

```
# Forecast horizon = length of test set
co2_fc <- co2_fit |>
  forecast(new_data = co2_test)

#co2_fc

# Plot forecasts against the full series
co2_fc |> autoplot(co2) +
  labs(title = "Forecasts 20% hold-out with TSLM, ETS, ARIMA, and Snaive models",
    x = "Month", y = "Mauna Loa CO2 (ppm)")
```




```
# Forecast (test) accuracy on the 20% hold-out period
test_acc <- co2_fc |>
  accuracy(co2_test) |>
  arrange(RMSE) |>
  select(.model: MAPE)

knitr::kable(test_acc, digits = 2,
             caption = "Forecast accuracy on the 20% hold-out period")
```

Table 2: Forecast accuracy on the 20% hold-out period

.model	.type	ME	RMSE	MAE	MPE	MAPE
log_arima	Test	3.09	3.61	3.10	0.74	0.75
log_ets	Test	3.32	3.92	3.34	0.80	0.80
auto_ets	Test	3.67	4.30	3.67	0.88	0.88
auto_arima	Test	3.78	4.45	3.79	0.91	0.91
log_tslm	Test	9.97	10.50	9.97	2.41	2.41
auto_tslm	Test	12.64	13.30	12.64	3.06	3.06
snaive	Test	17.94	20.46	17.94	4.32	4.32

4.5 Best model parameters

Which is the best performance model on test set and training set?

what are the model parameters?

The best performing model on both the training and test sets is the log_ets model. The model parameters for the log_ets model are as follows. We also gather the parameters for all models for rolling origin cross validation modeling.

Based on the KPSS test (p = 0.01), at least one regular difference was required. Our final Auto ARIMA model implemented $\mathbf{d = 1}$, along with a seasonal difference ($\mathbf{D = 1}$) to handle annual seasonality, resulting in the fitted model ARIMA(1,1,1)(2,1,2)[12]

```
co2_fit |> t()
```

```
           [,1]
auto_tslm  TSLM
auto_ets   ETS(A,A,A)
auto_arima ARIMA(1,1,1)(2,1,2)[12]
log_tslm   TSLM
log_ets    ETS(A,A,A)
log_arima  ARIMA(1,1,1)(2,1,2)[12]
snaive     SNAIVE
```

4.6 Rolling Origin Cross-validation

Step1: Modeling with the first 10 years of the CO2 training data (no CV)

- Use only the first 10 years of the data

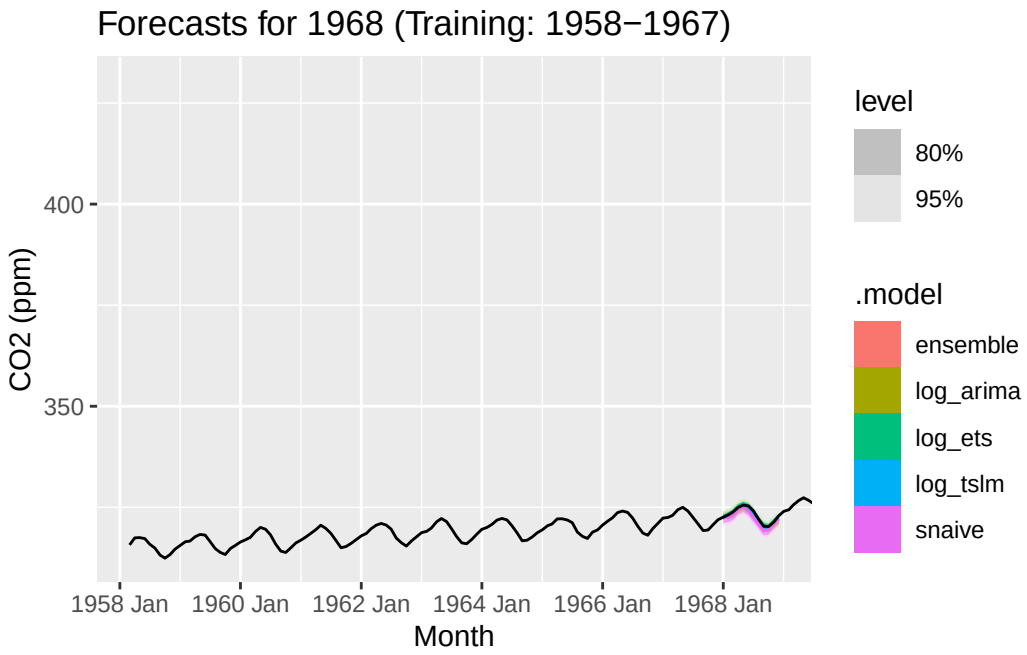
- Create the 3 models above on the log transformed data
- Include a seasonal naive model (on the untransformed data)
- Create an ensemble model of the 3 regression models using a simple average
- Forecast the next year (1968)
- Report the model metrics sorted by RMSE

```
# 1. Training set: the first 10 calendar years
#(1958 Mar-1967 Dec: total 118 months)
co2_train_10 <- co2 |>
  filter(Month >= yearmonth("1958-03"),
         Month <= yearmonth("1967-12"))

# 2. Test set: forecast the next year (1968)
co2_test_1968 <- co2 |>
  filter(year(Month) == 1968)

# 3. Fit models on log(average), plus seasonal naive on original scale
fit_10 <- co2_train_10 |>
  model(
    log_tslm  = TSLM(log(average) ~ trend() + season()),
    log_ets   = ETS(log(average)),
    log_arima = ARIMA(log(average)),
    snaive    = SNAIVE(average)
  )

# 4. Build ensemble of the 3 regression models using a simple average
fit_ens <- mutate(fit_10,
                  ensemble = (log_tslm + log_ets + log_arima) / 3)

# 5. Forecast the next 12 months (1968) and BACK-TRANSFORM log models
fc_1968 <- fit_ens |>
  forecast(h = "12 months") |>
  mutate(
    .mean = if_else(.model == "snaive", .mean, exp(.mean)))

# 6. Compute accuracy on 1968 and sort by RMSE  (now in ppm)
fc_1968 |>
  accuracy(co2_test_1968) |>
  arrange(RMSE) |>
  select(.model:MAPE) |>
  knitr::kable(
    digits = 2,
    caption = "Forecast accuracy for 1968 using the first 10 years (1958-1967) as training da
```

Table 3: Forecast accuracy for 1968 using the first 10 years (1958–1967) as training data

.model	.type	ME	RMSE	MAE	MPE	MAPE
log_ets	Test	-0.15	0.23	0.21	-0.05	0.06
ensemble	Test	0.17	0.25	0.17	0.05	0.05
log_tslm	Test	0.31	0.36	0.31	0.10	0.10
log_arima	Test	0.33	0.41	0.35	0.10	0.11
snaive	Test	0.87	0.94	0.87	0.27	0.27

```
# 7. Plot Forecast for 1968 (simple clean version, in ppm)
autoplot(fc_1968, co2) +
  labs(title = "Forecasts for 1968 (Training: 1958-1967)",
       x = "Month", y = "CO2 (ppm)") +
  coord_cartesian(xlim = c(yearmonth("1958-03"), yearmonth("1968-12")))
```



Step2: Rolling-Origin Cross-Validation (10-year initial window, 1-year increments)

- Create a tsibble with cross-validation series that starts with 10 years (118 months) of data and adds one year at time through 2024.

- Fit the same models from the previous step
- Report the model metrics sorted by RMSE for each model across all validation intervals.

```
# 1. Define initial window length: first 10 calendar years
# 1958-03 to 1967-12 (118 months)
co2_train_10 |> nrow()
```

```
[1] 118
```

```
# 2. Create rolling-origin cross-validation tsibble
#     start with first 10 years (118 obs)
#     add 1 year (12 months) at a time

co2_cv <- co2 |>
  stretch_tsibble(.init = nrow(co2_train_10), .step = 12)

# 3. Fit the same models as before (log models + snaive)
# Note: The following rolling-origin cross-validation chunk may take several minutes
# to run in a cloud environment due to repeated ARIMA/ETS refitting across many windows.
co2_cv_fit <- co2_cv |>
  model(log_tslm  = TSLM(log(average) ~ trend() + season()),
        log_ets   = ETS(log(average) ~ error("A") + trend("A") + season("A")),
        log_arima = ARIMA(log(average) ~ pdq(1,1,1) + PDQ(2,1,2)),
        snaive    = SNAIVE(average))

# 4. Add ensemble AFTER fitting base models
co2_cv_fit_ens <- co2_cv_fit |>
  mutate(
    ensemble = (log_tslm + log_ets + log_arima) / 3
  )

# 5. Generate 1-year-ahead forecasts for each rolling window
#   h = 12 months and back-transform log models

co2_cv_fc <- co2_cv_fit_ens |>
  forecast(h = "12 months") |>
  mutate(
    .mean = if_else(.model == "snaive", .mean, exp(.mean))
  )


# 6. Output cross-validation accuracy table
```

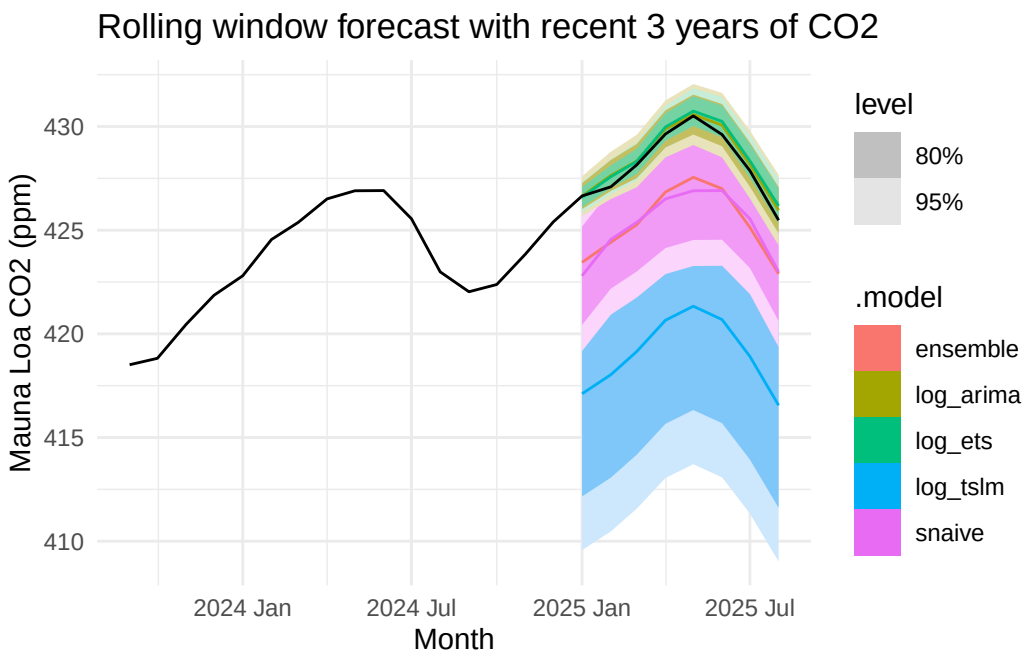
```r
last_obs <- max(co2$Month)

co2_cv_fc_trim <- co2_cv_fc |>
  filter(Month <= last_obs)

co2_cv_acc <- co2_cv_fc_trim |>
  accuracy(co2, by = ".model") |>
  arrange(RMSE) |>
  select(.model:MAPE)

co2_cv_acc |>
  knitr::kable(
    digits = 2,
    caption = "Rolling-origin cross-validation accuracy (h = 12, initial 10-year window)")
```

Table 4: Rolling-origin cross-validation accuracy (h = 12, initial 10-year window)

.model	.type	ME	RMSE	MAE	MPE	MAPE
log_arima	Test	0.08	0.50	0.39	0.02	0.11
log_ets	Test	0.06	0.50	0.38	0.02	0.11
ensemble	Test	1.17	1.47	1.19	0.31	0.31
snaive	Test	1.81	1.97	1.81	0.49	0.49
log_tslm	Test	3.38	4.13	3.38	0.88	0.88

```r
# 7. plot
#  most recently 24 month data
co2_recent <- co2 |>
  filter(Month > last_obs - 24)
#  last id of rolling window
last_id<- max(co2_cv_fc_trim$.id)

#  last window of forecast
co2_cv_fc_last<- co2_cv_fc_trim|>filter(.id== last_id)

#  most recent year with id
co2_recent_id<- co2_recent|>
  mutate(.id =last_id)|>
  as_tsibble(index =Month,key =.id)

#  autoplot
autoplot(co2_cv_fc_last, co2_recent_id)+
```

```
labs(title ="Rolling window forecast with recent 3 years of CO2",
     x ="Month",y ="Mauna Loa CO2 (ppm)") +theme_minimal()
```



Because rolling-origin cross-validation re-estimates all models for each expanding window, this step is relatively computationally expensive and may take several minutes to run in a cloud environment.

4.7 Modeling Summary:

To structure the forecasting analysis, I evaluate the models under three complementary settings.

1. We begin with a standard 80/20 hold-out split to compare model performance on both the original and log-transformed CO2 series, establishing a baseline for in-sample and out-of-sample accuracy.
2. Next, We use a fixed 10-year training window (1958–1967) to forecast the following 12 months, replicating a realistic short-horizon forecasting scenario based on limited historical data.
3. Finally, We extend this design with a rolling-origin cross-validation framework, starting with the same 10-year window and expanding it one year at a time, which provides a more robust assessment of model stability and average forecast accuracy across multiple validation intervals.

5. Model Comparison and Selection

Our model comparisons produced different rankings depending on the validation structure.

This is expected because each framework emphasizes a different forecasting challenge.

1. **Using a simple 80/20 split**, the log-transformed ARIMA and ETS models performed best.This setup involves forecasting nearly ten years ahead, where capturing the accelerating trend is essential.
2. **When evaluating only 1968 using the initial ten years as training**, ETS achieved the lowest error. The early CO2 series exhibits almost linear trend and stable seasonal amplitude, which makes short-horizon forecasting comparatively easy.
3. **Under rolling-origin cross-validation**, log-ETS again emerged as the most consistently accurate model, followed closely by log-ARIMA. This method averages performance across multiple decades and provides the most realistic assessment of generalization over changing trend and seasonality patterns.

Across all metrics, **log-ets and log_arima offer the strongest balance of adaptability, stability, and interpretability**, making it the most reliable choice for CO2 forecasting in this project.

6. Final Model Diagnostics

Residual diagnostics for final log_ets and log_arimamodel

```
# Extract the final model: log_ets
final_fit_ets <- co2_fit |>
  select(log_ets)

report(final_fit_ets)
```

```
Series: average
Model: ETS(A,A,A)
Transformation: log(average)
  Smoothing parameters:
    alpha = 0.6030969
    beta  = 0.004123315
    gamma = 0.0469199

  Initial states:
```

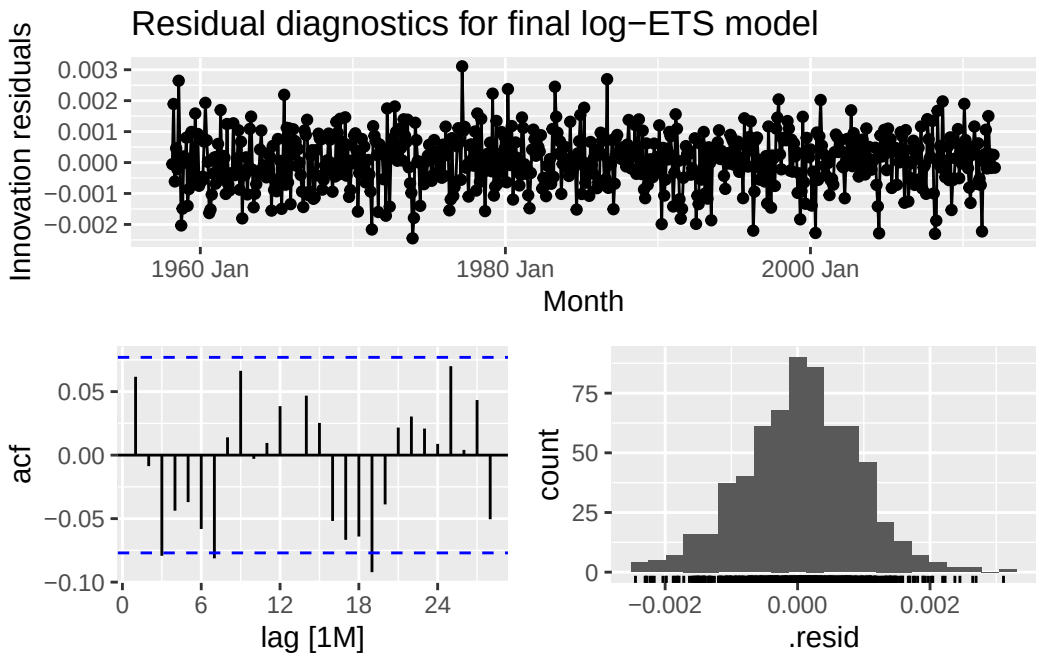
```
     l[0]          b[0]         s[0]         s[-1]         s[-2]         s[-3]
 5.750849 0.0002652315 0.001807899 4.562435e-06 -0.002697169 -0.005905782
       s[-4]         s[-5]         s[-6]        s[-7]        s[-8]        s[-9]
 -0.009221136 -0.008505131 -0.003648267 0.002334863 0.006627789 0.008358711
      s[-10]       s[-11]
 0.007080457 0.003763202

  sigma^2:  0

      AIC      AICc       BIC
-4915.095 -4914.123 -4839.038
```

```r
# Extract the final model: log_arima
final_fit_arima <- co2_fit |>
  select(log_arima)

report(final_fit_arima)
```

```
Series: average
Model: ARIMA(1,1,1)(2,1,2)[12]
Transformation: log(average)

Coefficients:
         ar1      ma1     sar1     sar2     sma1     sma2
      0.2057  -0.5561  -0.3316  -0.0277  -0.5479  -0.2991
s.e.  0.1129   0.0974   1.6936   0.0545   1.6939   1.5106

sigma^2 estimated as 1.441e-06:  log likelihood=3567.05
AIC=-7120.11   AICc=-7119.93   BIC=-7088.93
```

ljung-box test with sufficient lags=24

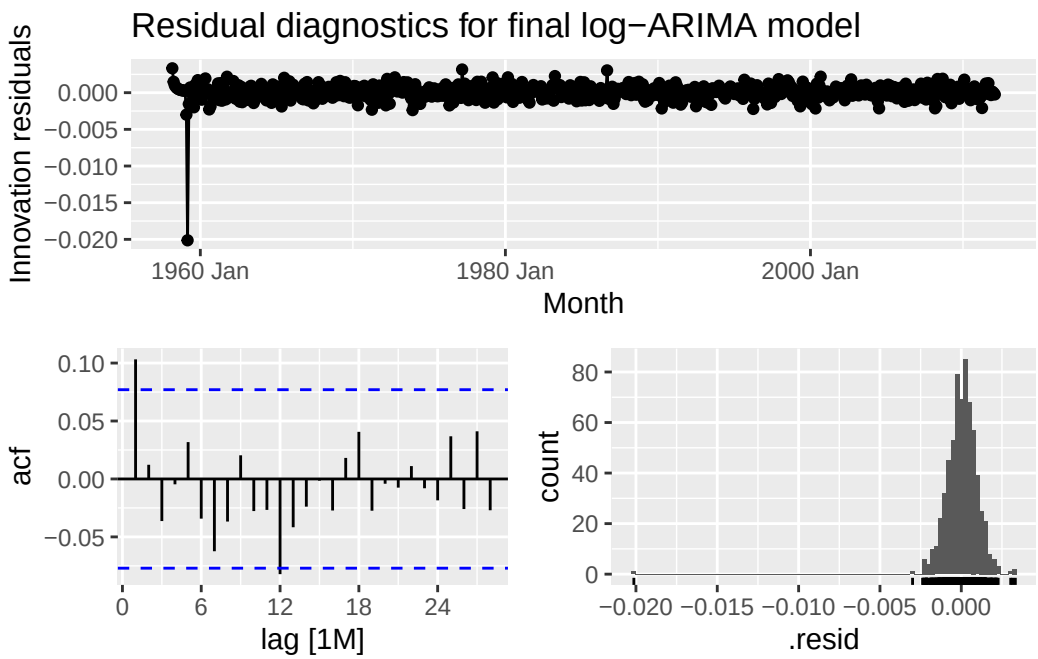
```r
# Residual diagnostics for final log-ETS model
final_fit_ets |>
  gg_tsresiduals() +
  labs(title = "Residual diagnostics for final log-ETS model")
```




```
# perform a ljung-box test with sufficient lags
final_fit_ets|>
  augment() |>
  features(.innov, ljung_box, lag=24)
```

```
# A tibble: 1 x 3
  .model  lb_stat lb_pvalue
  <chr>     <dbl>     <dbl>
1 log_ets    36.8    0.0458
```

```
# Residual diagnostics for final log-arima model
final_fit_arima |>
  gg_tsresiduals() +
  labs(title = "Residual diagnostics for final log-ARIMA model")
```




```
# perform a ljung-box test with sufficient lags
final_fit_arima|>
  augment() |>
  features(.innov, ljung_box, lag=24)
```

```
# A tibble: 1 x 3
  .model    lb_stat lb_pvalue
  <chr>       <dbl>     <dbl>
1 log_arima    22.8     0.533
```

Both models capture the long-term upward trend and seasonal pattern of atmospheric CO2, but the log-ARIMA model provides noticeably cleaner residuals. Its ACF shows no remaining autocorrelation and the Ljung–Box p-value (0.53) indicates white-noise errors, while the log-ETS model shows borderline autocorrelation ($p = 0.046$). Therefore, log-ARIMA offers a more statistically adequate fit for this series.

1. Residual time series (ETS / ARIMA)

ETS: Residuals show small clusters and mild structure across time.

ARIMA: Residuals fluctuate randomly around zero with no visible patterns.

2. ACF plot (ETS / ARIMA)

ETS: Several spikes approach or exceed the confidence bands, indicating remaining autocorrelation.

ARIMA: All autocorrelations fall well within the bands, suggesting no remaining time dependence.

3. Residual histogram (ETS / ARIMA)

ETS: Residuals are roughly symmetric but slightly more dispersed.

ARIMA: Residuals form a tighter, more centered distribution consistent with white noise.

4. Ljung–Box test (ETS / ARIMA)

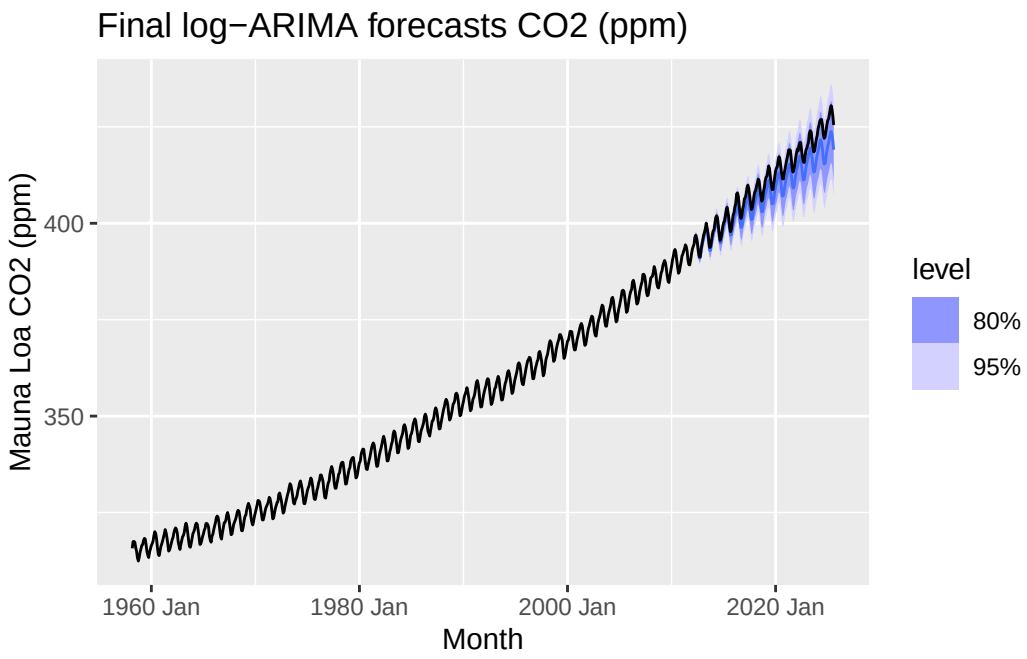
ETS: p 0.046, indicating borderline residual autocorrelation.

ARIMA: p 0.53, confirming residuals behave like white noise.

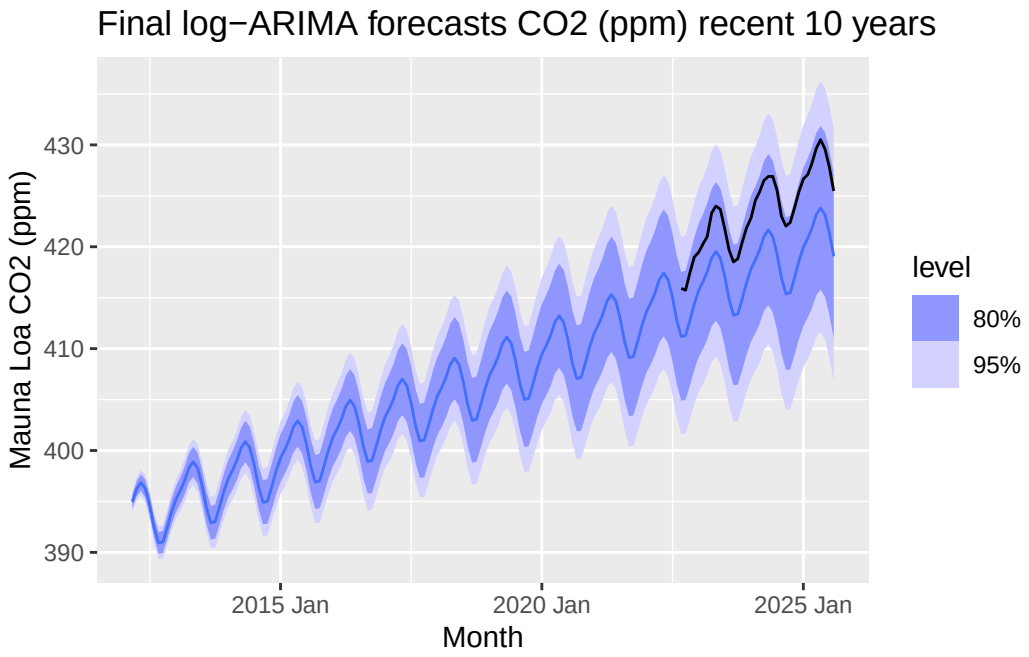
We performed a Ljung–Box test at lag 24, following the common guideline of using roughly two seasonal cycles for monthly data. The resulting p-value (0.53) provides no evidence of remaining autocorrelation, indicating that the log_arima model has adequately captured the trend and seasonality and that the residuals behave like white noise.

Finally, we generate forecasts from the log_arima model and back-transform them to the original ppm scale for interpretation. The forecast path smoothly extends the historical Keeling Curve, and the prediction intervals widen gradually into the future, reflecting increasing uncertainty while remaining consistent with the long-term upward trend observed in the data.

```
# Optional: produce back-transformed forecasts from the final model
final_fc <- final_fit_arima |>
  forecast(new_data = co2_test) |>
  mutate(.mean = exp(.mean))

co2_recent <- co2 |> filter (Month > last_obs - 36)

autoplot(final_fc, co2) +
  labs(title = "Final log-ARIMA forecasts CO2 (ppm)",
       x = "Month",
       y = "Mauna Loa CO2 (ppm)")
```




```
autoplot(final_fc, co2_recent) +
  labs(
    title = "Final log-ARIMA forecasts CO2 (ppm) recent 10 years",
    x = "Month", y = "Mauna Loa CO2 (ppm)")
```



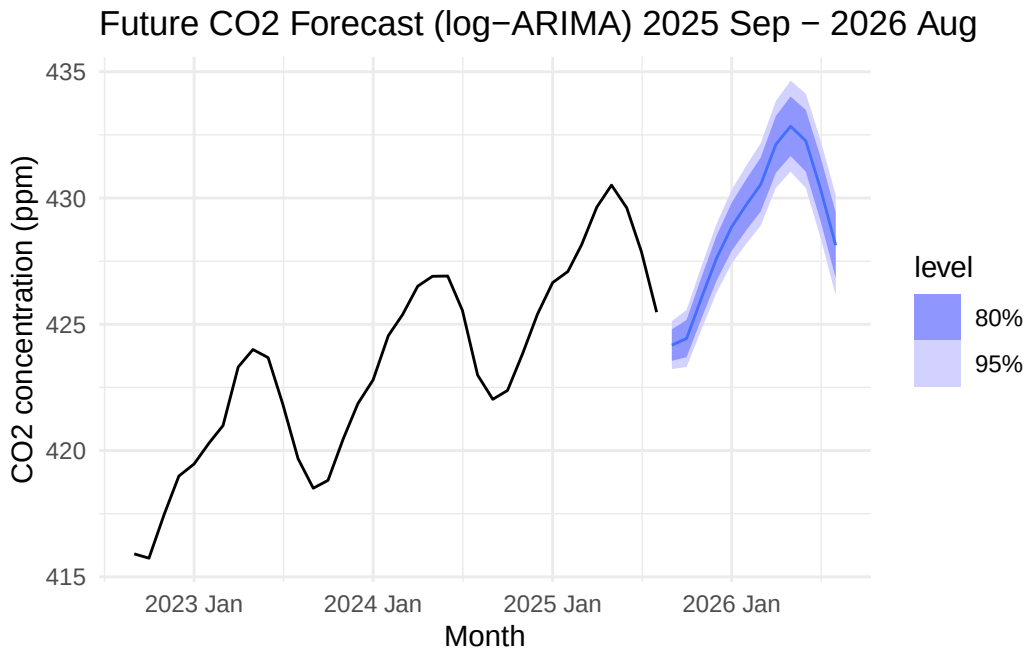
7. Final Model Forecasts (2025 Sep – 2026 Aug)

Use the best performing model to create a forecast for each month in next 12 month (2025 Sep to 2026 Aug).

- Based on the rolling-origin cross-validation results, the **log-ARIMA**model demonstrated the most stable and accurate performance across multiple decades of the CO2 series. We therefore selected log-ETS as the final forecasting model for the next stage of analysis.
- Using the full historical dataset (1958–2025 Aug), we refit a final log-ARIMA model and generated **12-month-ahead monthly forecasts from 2025 Sep through 2026 Aug**. The forecasts were back-transformed from the log scale to the original CO2 concentration units (ppm).
- The resulting trajectory continues the long-term upward trend in atmospheric CO2, with seasonal peaks early in the calendar year and troughs near late summer. The prediction intervals widen gradually over the forecast horizon, reflecting uncertainty in both the trend acceleration and seasonal amplitude as we project further into the future.

```
# 1. Refit the final model using full historical data
co2_final_fit <- co2 |>
  model(log_arima = ARIMA(log(average)))
```

```r
# 2. Forecast next 12 months (2025 Sep - 2026 Aug)
co2_final_fc <- co2_final_fit |>
  forecast(h = "12 months") |>
  mutate(.mean  = exp(.mean)
  )

# 3. Prepare recent historical data for comparison
co2_recent <- co2 |>
  filter(Month > last_obs - 36)

# 4. Plot (now in ppm correctly)
autoplot(co2_final_fc, co2_recent) +
  labs(
    title = "Future CO2 Forecast (log-ARIMA) 2025 Sep - 2026 Aug",
    x = "Month",
    y = "CO2 concentration (ppm)"
  ) +
  theme_minimal()
```



The 12-month forecast from **September 2025 to August 2026** shows:

1. **A continued upward trend**, consistent with the long-term trajectory of the Mauna Loa CO2 series.

2. **Seasonal oscillations**, with CO2 peaking around May and dipping around late summer.
3. **Prediction intervals that widen over time**, indicating greater forecast uncertainty at longer horizons.

This behavior is consistent with both the historical pattern of the Keeling Curve and the error structure identified in model diagnostics.