Assignment 4 | Dynamic Linear Models with R

Stefano Graziosi

```
# Time series
library(dlm)
library(TSstudio)
library(feasts)
```

```
Loading required package: fabletools
```

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

```
library(tseries)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

```
  # Necessary packages for quantmod
  library(zoo)
```

```
Attaching package: 'zoo'
```

```
The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

```
  library(xts)
library(quantmod)
```

```
Loading required package: TTR
```

```
#Specifically for Assignment 2
library(depmixS4)
```

```
Loading required package: nnet
```

```
Loading required package: MASS
```

```
Loading required package: Rsolnp
```

```
Loading required package: nlme
```

```

Attaching package: 'nlme'
```

```
The following object is masked from 'package:feasts':

    ACF
```

```
library(HiddenMarkov)

# Datasets
library(readr)
library(fpp3)
```

```
-- Attaching packages --------------------------------------- fpp3 1.0.1 --
```

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

```
-- Conflicts ------------------------------------------------ fpp3_conflicts --
x ggplot2::%+%()       masks dlm::%+%()
x nlme::ACF()          masks feasts::ACF()
x dplyr::collapse()    masks nlme::collapse()
x lubridate::date()    masks base::date()
x dplyr::filter()      masks stats::filter()
x dplyr::first()       masks xts::first()
x tsibble::index()     masks zoo::index()
x tsibble::intersect() masks base::intersect()
x tsibble::interval()  masks lubridate::interval()
x dplyr::lag()         masks stats::lag()
x dplyr::last()        masks xts::last()
x dplyr::select()      masks MASS::select()
x tsibble::setdiff()   masks base::setdiff()
x tsibble::union()     masks base::union()
```

```
# For fancy plots
library(ggthemes)
  # Necessary packages for viridis
  library(viridisLite)
library(viridis)
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```
library(magrittr)
```

```
Attaching package: 'magrittr'

The following object is masked from 'package:tidyr':

    extract
```

```
library(textab)

# Packages related to tidyverse, for data manipulation
library(tidyverse) # includes (lubridate), (dplyr), (ggplot2), (tidyr), (tidyselect)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats 1.0.0     v stringr 1.5.1
v purrr   1.0.4

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x ggplot2::%+%()       masks dlm::%+%()
x dplyr::collapse()    masks nlme::collapse()
x gridExtra::combine() masks dplyr::combine()
x magrittr::extract()  masks tidyr::extract()
x dplyr::filter()      masks stats::filter()
x dplyr::first()       masks xts::first()
x tsibble::interval()  masks lubridate::interval()
x dplyr::lag()         masks stats::lag()
x dplyr::last()        masks xts::last()
x dplyr::select()      masks MASS::select()
x purrr::set_names()   masks magrittr::set_names()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(tinytex)

# To handle time changes
library(timechange)


# To solve conflicts
library(conflicted)
conflicts_prefer(dplyr::filter)
```

```
[conflicted] Will prefer dplyr::filter over any other package.
```

Kalman filter for the random walk plus noise model

Consider the Nile data (measurements of the annual ow of the river Nile at Ashwan 1871-1970), available in R (`> ?Nile`).

First, plot the data. The series clearly appears non-stationary, presenting a quite evident change point. A local level model, i.e. a random walk plus noise, may be used to capture the main change point and other minor changes in the level of the Nile river. Let us consider the following random walk plus noise model.

$$
\begin{aligned}
y_t &= \theta_t + v_t \qquad v_t \sim \mathcal{N}(0, V) \\
\theta_t &= \theta_{t-1} + w_t \quad w_t \sim \mathcal{N}(0, W)
\end{aligned}
$$

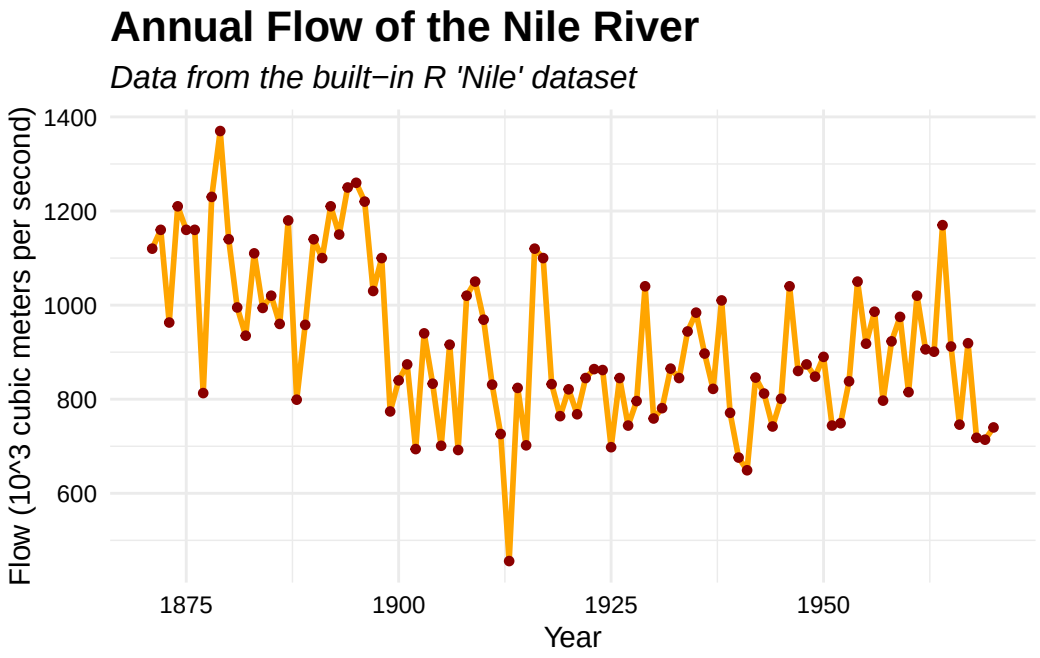
with the due assumptions.

To start with, assume that the variances are known, $V = 15100$, $W = 1470$. In fact, they will have to be estimated (next assignment). As the initial distribution, let $\theta_0 \sim \mathcal{N}(1000, 1000)$.

```
df <- Nile

nile_df <- data.frame(
  Y = as.numeric(time(Nile)),
  Flow = as.numeric(Nile)
)
```

```
ggplot(nile_df, aes(x = Y, y = Flow)) +
  geom_line(color = "orange", size = 1) +     # Draw the line for the time series
  geom_point(color = "darkred", size = 1.2) +        # Add points to highlight each annual me
  labs(
    title = "Annual Flow of the Nile River",
    subtitle = "Data from the built-in R 'Nile' dataset",
    x = "Year",
    y = "Flow (10^3 cubic meters per second)"
  ) +
  theme_minimal() +                                  # Use a minimal theme for a clean appearan
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic"),
    axis.text = element_text(color = "black")
  )
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



Step 1: Filtering

1. Compute the filtering estimates

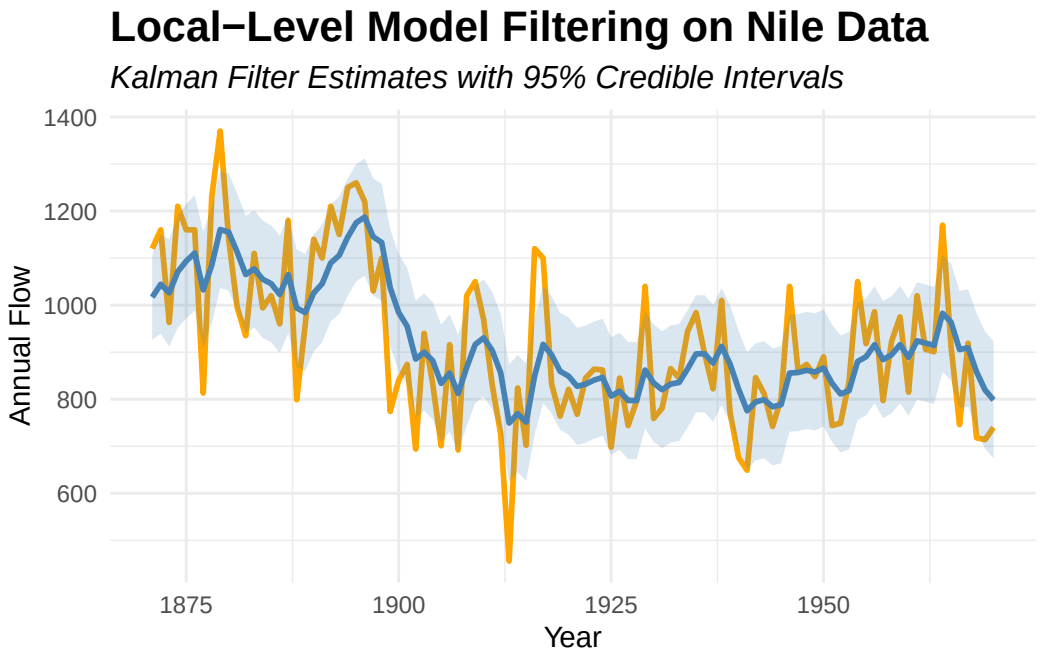
```
dlm_s1 <- dlm(m0 = 1000, C0 = 1000, FF = 1, V = 15100, GG = 1, W = 1470)

filtered_s1 <- dlmFilter(Nile, dlm_s1)
m_filtered_s1 <- dropFirst(filtered_s1$m)
```

2. Compute the corresponding standard deviations and plot them. Comment briefly.

```
Ct_list <- dlmSvd2var(filtered_s1$U.C, filtered_s1$D.C)
sd_filtered_s1 <- sqrt(unlist(Ct_list))[-1]

upper_95 <- m_filtered_s1 + 1.96 * sd_filtered_s1
lower_95 <- m_filtered_s1 - 1.96 * sd_filtered_s1
```

3. Finally, plot the data together with the ltering state estimates and their 0.95 credible intervals.

```r
df_nile <- data.frame(
  Year      = as.numeric(time(Nile)),
  Flow      = as.numeric(Nile),
  m_filt    = m_filtered_s1,
  sd_filt   = sd_filtered_s1,
  upper_95  = upper_95,
  lower_95  = lower_95
)

ggplot(df_nile, aes(x = Year)) +
  geom_line(aes(y = Flow), color = "orange", size = 1) + # Original data (gray line)
  geom_line(aes(y = m_filtered_s1), color = "steelblue", size = 1) + # Filtered estimates (bl
  geom_ribbon(aes(ymin = lower_95, ymax = upper_95), # 95% credibility interval around the fi
              fill = "steelblue", alpha = 0.2) +
  labs(
    title = "Local-Level Model Filtering on Nile Data",
    subtitle = "Kalman Filter Estimates with 95% Credible Intervals",
    x = "Year",
    y = "Annual Flow"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic")
  )
```



Step 2: Online forecasting

1. Compute the one-step ahead forecasts

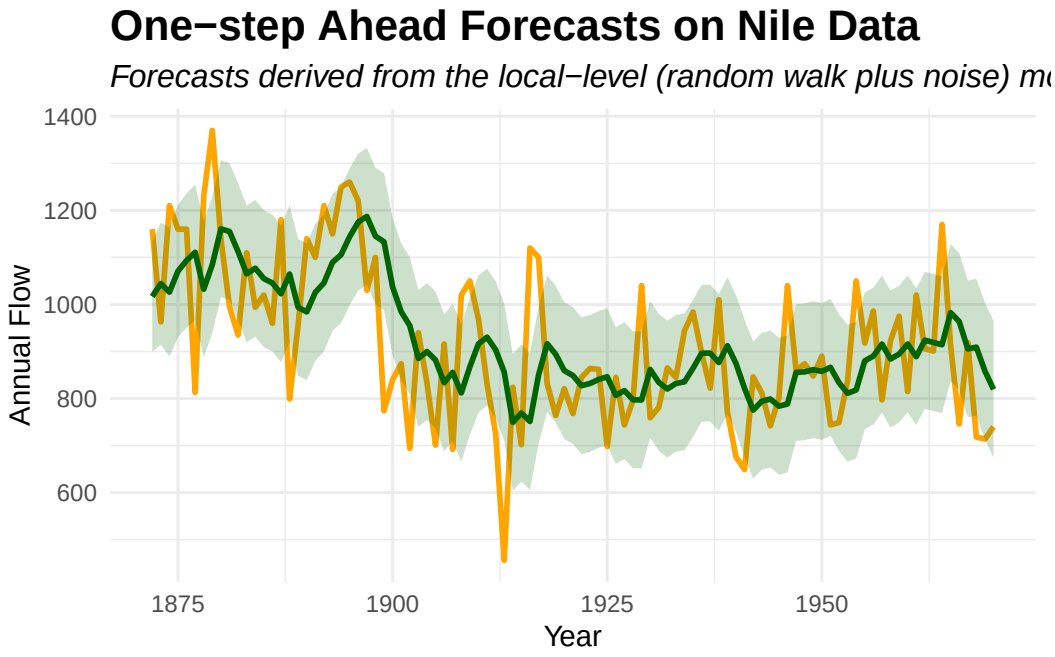
```
forecast_means <- dropFirst(filtered_s1$a)  # one-step ahead forecast means

forecast_var <- dlmSvd2var(filtered_s1$U.R, filtered_s1$D.R)
forecast_sd <- sqrt(unlist(forecast_var))[-1]
```

```
upper_forecast <- forecast_means + 1.96 * forecast_sd
lower_forecast <- forecast_means - 1.96 * forecast_sd
```

2. Plot the data, together the one-step-ahead forecasts and their 0.95 credible intervals

```
df_forecast <- data.frame(
  Year       = as.numeric(time(Nile))[-1],
  Flow       = as.numeric(Nile)[-1],
  Forecast   = forecast_means,
  sd_Forecast = forecast_sd,
  Lower_95 = lower_forecast,
  Upper_95 = upper_forecast
)
```

```r
# Plot the one-step forecasts and the corresponding 95% credible intervals.
ggplot(df_forecast, aes(x = Year)) +
  geom_line(aes(y = Flow), color = "orange", size = 1) +
  geom_line(aes(y = Forecast), color = "darkgreen", size = 1) +
  geom_ribbon(aes(ymin = Lower_95, ymax = Upper_95), fill = "darkgreen", alpha = 0.2) +
  labs(
    title = "One-step Ahead Forecasts on Nile Data",
    subtitle = "Forecasts derived from the local-level (random walk plus noise) model",
    x = "Year",
    y = "Annual Flow"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic")
  )
```



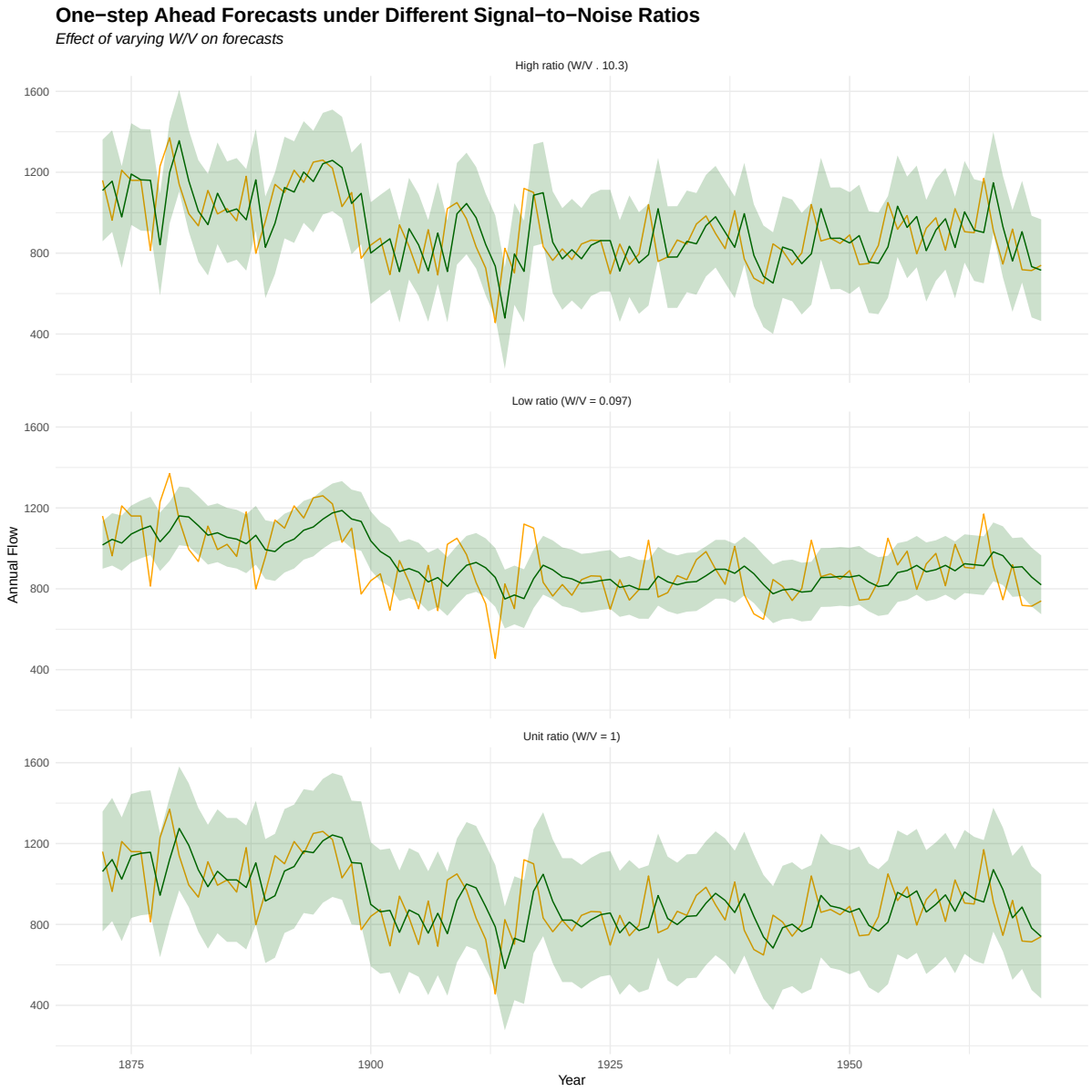
Step 3: Signal-to-noise Ratio

What is the eect of the signal-to-noise ratio (i.e. the ratio W/V) on the forecasts? Repeat the exercise with dierent choices of V (observation variance) and W (evolution variance) and comment briey.

```r
scenarios <- data.frame(
  Scenario = c("Low ratio (W/V = 0.097)", "Unit ratio (W/V = 1)", "High ratio (W/V   10.3)"),
  V = c(15100, 15100, 1470),
  W = c(1470, 15100, 15100)
)

forecast_list <- list()

for(i in 1:nrow(scenarios)) {
  V_val <- scenarios$V[i]
  W_val <- scenarios$W[i]
  scenario_name <- scenarios$Scenario[i]

  dlm_model <- dlm(m0 = 1000, C0 = 1000, FF = 1, V = V_val, GG = 1, W = W_val)
  filter_mod <- dlmFilter(Nile, dlm_model)

  # Compute one-step forecasts
  f_mean <- dropFirst(filter_mod$a)
  f_var <- dlmSvd2var(filter_mod$U.R, filter_mod$D.R)
  f_sd <- forecast_sd <- sqrt(unlist(f_var))[-1]

  scenario_df <- data.frame(
    Year = as.numeric(time(Nile))[-1],
    Flow = as.numeric(Nile)[-1],
    Forecast = f_mean,
    sd_Forecast = f_sd,
    Lower_95 = f_mean - 1.96 * f_sd,
    Upper_95 = f_mean + 1.96 * f_sd,
    Scenario = scenario_name
  )

  forecast_list[[i]] <- scenario_df
}
```

```r
df_forecast_all <- bind_rows(forecast_list)

ggplot(df_forecast_all, aes(x = Year)) +
  geom_line(aes(y = Flow), color = "orange", size = 0.5) +
  geom_line(aes(y = Forecast), color = "darkgreen", size = 0.5) +
  geom_ribbon(aes(ymin = Lower_95, ymax = Upper_95), fill = "darkgreen", alpha = 0.2) +
  facet_wrap(~ Scenario, ncol = 1) +
  labs(
```

```
    title = "One-step Ahead Forecasts under Different Signal-to-Noise Ratios",
    subtitle = "Effect of varying W/V on forecasts",
    x = "Year",
    y = "Annual Flow"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic")
  )
```



Effect of the Signal-to-Noise Ratio on One-Step Forecasts

From the three panels above we see:

- **Low ratio (W/V = 0.097):**
  Here the evolution variance W is small relative to the observation noise V. The model "believes" the underlying level moves very slowly and that most of the variation in the

data is noise. As a result, the one-step forecasts are very 'flat' (they stick close to the previous level) and the 95% intervals are relatively narrow.

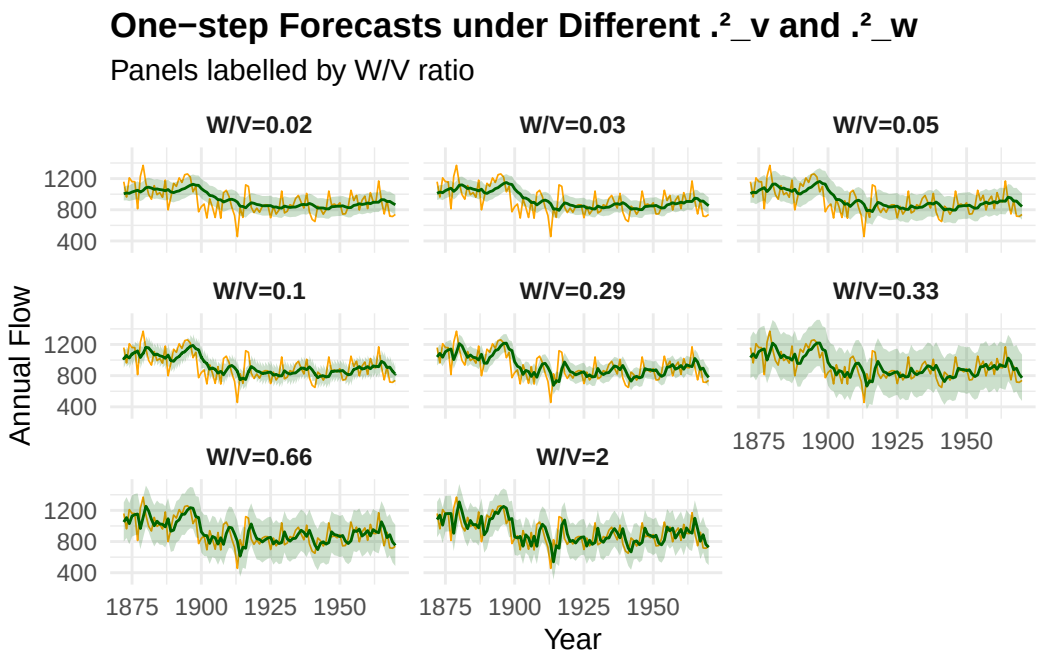
- **Unit ratio (W/V = 1):**
  When W and V are equal, the model balances changes in the level against observation noise. Forecasts react moderately to recent observations and the uncertainty bands widen at an intermediate rate.

- **High ratio (W/V 10.3):**
  Now the model thinks the level is very volatile (large W) but observations are quite precise (small V). Forecasts jump almost immediately to follow the last observation, and the 95% bands expand quickly because the state is assumed to wander freely.

In practice:
- A **low** signal-to-noise ratio produces **smooth, stable** forecasts with **narrow** intervals, essentially ignoring small fluctuations as noise.
- A **high** ratio produces **responsive**, but **uncertain**, forecasts with **wide** intervals, since the level is free to change rapidly.
- Choosing W/V is therefore a trade-off between **tracking** real shifts in the level and **dampening** transient noise.

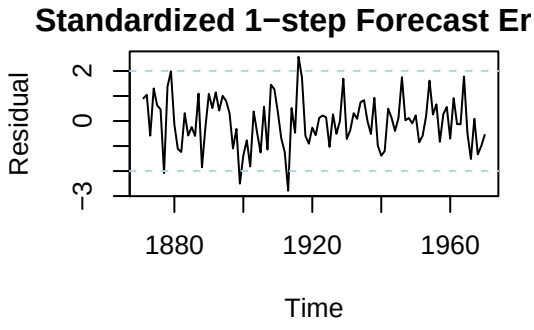
```
V_vals <- c(5000, 15100, 30000)      # try low, baseline, high observation noise
W_vals <- c( 500,  1470, 10000)      # and low, baseline, high evolution noise

scenarios2 <- expand.grid(V = V_vals, W = W_vals) %>%
  mutate(Scenario = paste0("W/V=", round(W/V, 2)))

forecast_list2 <- lapply(seq_len(nrow(scenarios2)), function(i) {
  V_val <- scenarios2$V[i]
  W_val <- scenarios2$W[i]
  scen  <- scenarios2$Scenario[i]

  model_i <- dlm(m0 = 1000, C0 = 1000, FF = 1, V = V_val, GG = 1, W = W_val)
  filt_i  <- dlmFilter(Nile, model_i)

  f_mean <- dropFirst(filt_i$a)
  f_var  <- dlmSvd2var(filt_i$U.R, filt_i$D.R)
  f_sd   <- sqrt(unlist(f_var))[-1]

  data.frame(
    Year     = as.numeric(time(Nile))[-1],
    Flow     = as.numeric(Nile)[-1],
    Forecast = f_mean,
    Lower95  = f_mean - 1.96*f_sd,
```

```
    Upper95  = f_mean + 1.96*f_sd,
    Scenario = scen
  )
})

df2 <- bind_rows(forecast_list2)

library(ggplot2)
ggplot(df2, aes(x = Year)) +
  geom_line(aes(y = Flow), color = "orange",   size = 0.3) +
  geom_line(aes(y = Forecast), color = "darkgreen", size = 0.5) +
  geom_ribbon(aes(ymin = Lower95, ymax = Upper95),
              fill = "darkgreen", alpha = 0.2) +
  facet_wrap(~ Scenario, ncol = 3) +
  labs(
    title    = "One-step Forecasts under Different ²_v and ²_w",
    subtitle = "Panels labelled by W/V ratio",
    x        = "Year",
    y        = "Annual Flow"
  ) +
  theme_minimal() +
  theme(
    strip.text = element_text(face = "bold"),
    plot.title = element_text(face = "bold")
  )
```

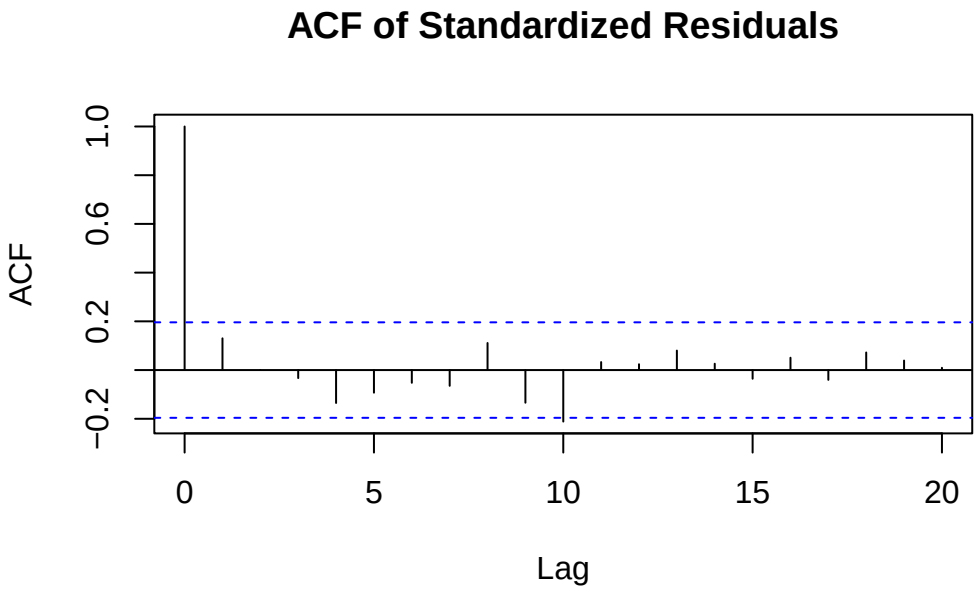


Step 4: Model Checking

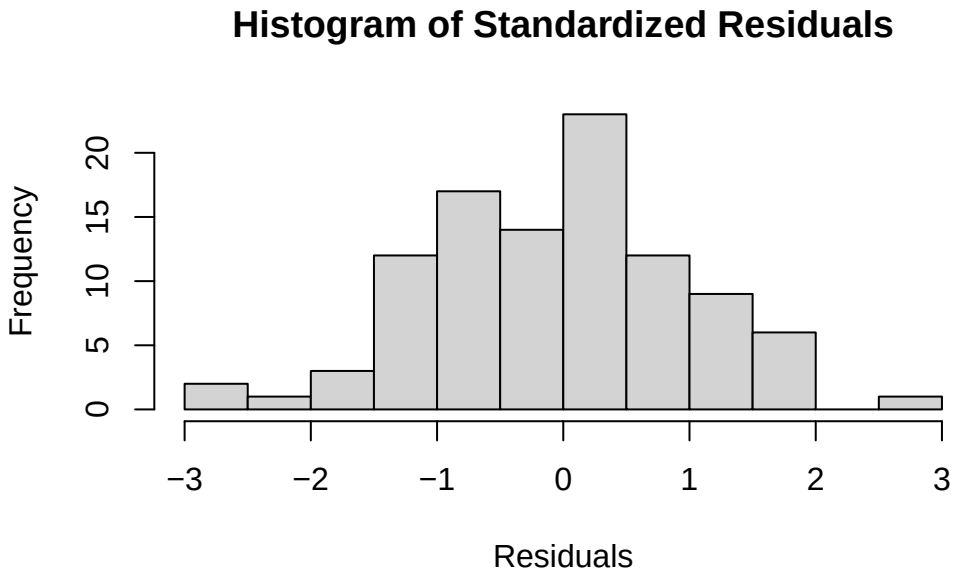
```
std_list <- residuals(filtered_s1, sd = TRUE)
std_resid <- std_list$res

par(mfrow = c(2,2), mar = c(4,4,2,1))
ts.plot(std_resid,
        main = "Standardized 1-step Forecast Errors",
        ylab = "Residual", xlab = "Time")
abline(h = c(-2, 2), lty = 2, col = "lightblue")
```



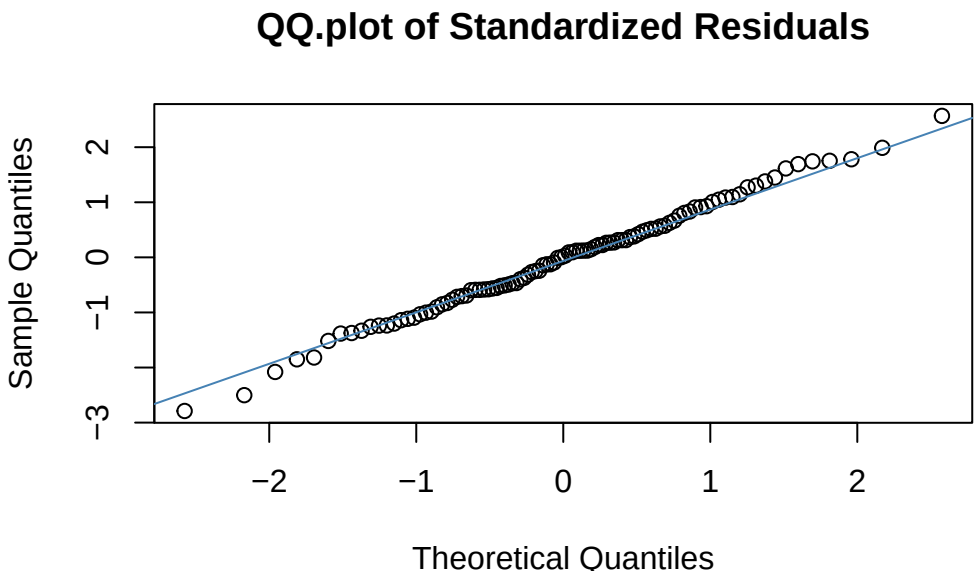
```r
acf(std_resid, main = "ACF of Standardized Residuals")
```




```r
hist(std_resid, breaks = 12,
     main = "Histogram of Standardized Residuals",
     xlab = "Residuals")
```



```r
qqnorm(std_resid, main = "QQ-plot of Standardized Residuals")
qqline(std_resid, col = "steelblue")
```




```r
lb_test <- Box.test(std_resid, type = "Ljung-Box", lag = 10)
sw_test <- shapiro.test(as.numeric(std_resid))

print(lb_test)
```

```
    Box-Ljung test

data:  std_resid
X-squared = 13.947, df = 10, p-value = 0.1754
```

```r
print(sw_test)
```

```
    Shapiro-Wilk normality test

data:  as.numeric(std_resid)
W = 0.99576, p-value = 0.99
```

```r
par(mfrow = c(1,1))
```

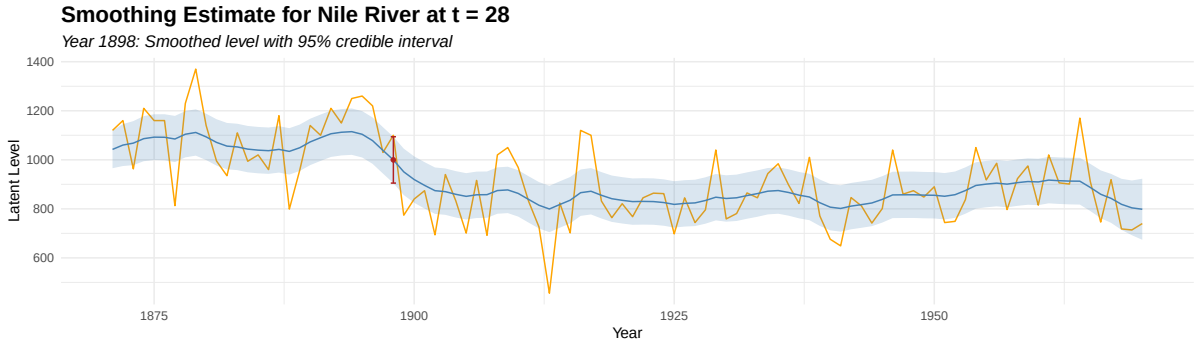
Step 5: Smoothing

So far, for computations, we pretended that the data arrived sequentially. Now consider $(y_1, \ldots, y_T)$ and provide and plot the smoothing estimate of the Nile level θ_t at time t = 28 together with its 95% credible interval.

```r
smoothed <- dlmSmooth(Nile, dlm_s1)

theta_smoothed <- dropFirst(smoothed$s)

Ct_smooth_list <- dlmSvd2var(smoothed$U.S, smoothed$D.S)

var_smoothed <- unlist(Ct_smooth_list)[-1]
sd_smoothed <- sqrt(var_smoothed)

theta_t28 <- theta_smoothed[28]
sd_t28 <- sd_smoothed[28]
upper_t28 <- theta_t28 + 1.96 * sd_t28
lower_t28 <- theta_t28 - 1.96 * sd_t28

df_smooth <- data.frame(
  Year = as.numeric(time(Nile)),
  Flow = as.numeric(Nile),
  Smoothed = theta_smoothed,
  Lower_95 = theta_smoothed - 1.96 * sd_smoothed,
  Upper_95 = theta_smoothed + 1.96 * sd_smoothed
)
```

```r
year_t28 <- as.numeric(time(Nile))[28]

ggplot() +
  geom_line(data = df_smooth, aes(x = Year, y = Flow), color = "orange", size = 0.5) +
  geom_line(data = df_smooth, aes(x = Year, y = Smoothed), color = "steelblue", size = 0.5) +
  geom_ribbon(data = df_smooth, aes(x = Year, ymin = Lower_95, ymax = Upper_95), fill = "stee
  # Highlight the smoothed estimate at t = 28 with error bars.
  geom_point(data = df_smooth %>% filter(Year == year_t28), aes(x = Year, y = Smoothed), colo
  geom_errorbar(data = df_smooth %>% filter(Year == year_t28), aes(x = Year, ymin = Lower_95
  labs(
    title = "Smoothing Estimate for Nile River at t = 28",
```

```
    subtitle = paste0("Year ", year_t28, ": Smoothed level with 95% credible interval"),
    x = "Year",
    y = "Latent Level"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic")
  )
```



Step 6: Maximum Likelihood Estimators

```
mle_fit <- StructTS(Nile, type = "level")

sigma2_w <- mle_fit$coef["level"]    # evolution variance Ŵ
sigma2_v <- mle_fit$coef["epsilon"]  # observation variance V

cat("MLE of evolution variance ( ²_w):", round(sigma2_w, 2), "\n")
```

```
MLE of evolution variance ( ²_w): 1469.15
```

```
cat("MLE of observation variance ( ²_v):", round(sigma2_v, 2), "\n")
```

```
MLE of observation variance ( ²_v): 15098.58
```