Online trend estimation and detection of deviations

Julia Schedler

2023-09-26

Table of contents

1 Code + Synthetic Data for "Online trend estimation and detection of trend deviations in sub-sewershed time series of SARS-CoV-2 RNA measured in wastewater"

> Paper: Online trend estimation and detection of trend deviations in sub-sewershed time series of SARS-CoV-2 RNA measured in wastewater
>
> Katherine B. Ensor, Julia Schedler, Thomas Sun, Rebecca Schneider, Anthony Mulenga, Jingjing Wu, Lauren B. Stadler, Loren Hopkins
> medRxiv 2023.10.26.23297635; doi: https://doi.org/10.1101/2023.10.26.23297635

This website provides details on the application of the state-space modeling and statistical process control frameworks to the time series of Sars-cov2 viral load for various sampling sites in the City of Houston.

The purpose of this analysis is to develop a method that

- compares series from two different sampling sites
- accounts for measurement error
- is comparable to the existing B-spline method employed by the City.

Intended audience: those who wish to replicate the analyses demonstrated here on their own WW epi data.

> Assumed knowledge + resources to fill gaps
>
> - ability to interpret algebraic equations
> - basic familiarity with R programming
>   - Suggested external learning resource: R for Data Science 2E by Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund.
> - an understanding of linear regression

  - Suggested external learning resource: Chapter 7 of Introduction to Modern Statistics by Mine Çetinkaya-Rundel and Johanna Hardin.

- a willingness to learn some basic time series techniques.
  - Suggested external learning resource: Time Series: A Data Analysis Approach Using R by Robert Shumway and David Stoffer.
  - Suggested external learning resource: Forecasting: Principles and Practice 3E by Rob J Hyndman and George Athanasopoulos.

Contact us!

- https://hou-wastewater-epi.org/contact-us

Part I

Trend Estimation

2 Algorithm 1

This code depends on just a few packages.

```
library(tidyverse)
library(patchwork) ## to get nice rendered visuals in quarto
library(KFAS)
```

2.1 1. Data Processing

We summarize our data cleaning process below to serve as an example, but the data cleaning needs will be different depending on the particulars of the wastewater surveillance system.

- **inputs:** raw lab values of viral load observations
- **data processing steps:**
  - identify observations below the level of detection using statistical analysis
  - align all observations to Mondays
  - transform copies per L to a log10 scale
  - average replicates to give one weekly measurement per week per location
  - only use locations where the primary WWTP has at least 85% coverage and observations within 1 month of last date
  - ensure there is a row for each week, even if the observation is missing
  - create an indicator of missing values
  - remove irrelevant features/variables
- **output:** a data frame with four variables:
  - date
  - name of location
  - log10 copies per/L

– missing data indicator

```
# load the cleaned/prepped data
all_ts_observed <- read.csv("Data/synthetic_ww_time_series.csv")
all_ts_observed$dates <- as.Date(all_ts_observed$dates)
head(all_ts_observed)
```

```
       dates           name    value ts_missing  colors
1 2021-05-24 Lift station A 3.397031      FALSE #44AA99
2 2021-05-31 Lift station A       NA       TRUE #44AA99
3 2021-06-07 Lift station A       NA       TRUE #44AA99
4 2021-06-14 Lift station A       NA       TRUE #44AA99
5 2021-06-21 Lift station A 4.543146      FALSE #44AA99
6 2021-06-28 Lift station A 4.356128      FALSE #44AA99
```

2.2 2. Initialize Model

The state space model we are fitting needs a certain number of observations to initialize the model. Sometimes this is called the "burnin" period. We have found that about 10 weeks of complete observations are necessary to obtain a good model fit. However, since some of the sampling locations have missing data at the beginning of the series, we set the burnin period to 15 weeks for all series. The code chunk below identifies the dates which will be considered part of the burnin period.

```
burnin <- 15
date_burnin <- all_ts_observed %>% dplyr::filter(name == 'WWTP') %>% dplyr::select(dates) %>%
init_vals <- c(1, .1) ## set observation variance to be larger than state variance.
```

2.3 3. Online Estimates

The model is fit using the KFAS package in R, which can fit any state space model, not just our smoothing spline model. However, KFAS does not use rolling estimation.

We have written wrapper functions which fits the smoothing spline state space model and which performs the rolling estimation.

`KFAS_rolling_estimation.r` and `KFAS_state_space_spline.r`

State space spline using `KFAS::fitSSM` . Note the specification of matrices– this is what gives the smoothing spline structure. Different choice of matrices will give a different model structure, e.g. AR(1), ARIMA, etc.

```
KFAS_state_space_spline <- function(ts_obs, name, ts.missing, ts_dates, init_par){

## Specify model structure
A = matrix(c(1,0),1)
Phi = matrix(c(2,1,-1,0),2)
mu1 = matrix(0,2)
P1 = diag(1,2)
v = matrix(NA)
R = matrix(c(1,0),2,1)
w = matrix(NA)

#function for updating the model
update_model <- function(pars, model) {
  model["H"][1] <- pars[1]
  model["Q"][1] <- pars[2]
  model
}

#check that variances are non-negative
check_model <- function(model) {
  (model["H"] > 0 && min(model["Q"]) > 0)
}

# Specify the model
mod <- KFAS::SSModel(ts_obs ~ -1 +
                 SSMcustom(Z = A, T = Phi, R = R, Q = w, a1 = mu1, P1 = P1), H = v)

# Fit the model
fit_mod <- KFAS::fitSSM(mod, inits = init_par, method = "BFGS",
                  updatefn = update_model, checkfn = check_model, hessian=TRUE,
                  control=list(trace=FALSE,REPORT=1))

## Format for output
ts_len <- length(ts_obs)
smoothers <- data.frame(est = KFAS::KFS(fit_mod$model)$alphahat[,1],
                  lwr = KFAS::KFS(fit_mod$model)$alphahat[,1]- 1.96*sqrt(KFAS::KFS(fit_mod$
                  upr = KFAS::KFS(fit_mod$model)$alphahat[,1]+ 1.96*sqrt(KFAS::KFS(fit_mod$
                  ts_missing = ts.missing,
                  name = rep(name[1], times = ts_len),
                  fit = rep("smoother", times = ts_len),
                  date = ts_dates,
                  sigv = rep(fit_mod$optim.out$par[1], times = ts_len),
                  sigw = rep(fit_mod$optim.out$par[2], times = ts_len),
                  obs = ts_obs,
                  resid = rstandard(KFAS::KFS(fit_mod$model), type = "recursive"),
                  conv = fit_mod$optim.out$convergence)

filters <- data.frame(est = KFAS::KFS(fit_mod$model)$att[,1],
            lwr = KFAS::KFS(fit_mod$model)$att[,1]- 1.96*sqrt(KFAS::KFS(fit_mod$model)$Ptt[1
            upr = KFAS::KFS(fit_mod$model)$att[,1]+ 1.96*sqrt(KFAS::KFS(fit_mod$model)$Ptt[1
            ts_missing = ts.missing,
            name = rep(name[1], times = ts_len),
            fit = rep("filter", times = ts_len),
```

Rolling estimation code: Note that this calls `KFAS_state_space_spline.r` multiple times: once for the initialization using the burnin period set above, and the once for each subsequent time point.

```
KFAS_rolling_estimation <- function(init_vals_roll,
                               ts_obs_roll,
                               ts_name_roll,
                               dates_roll,
                               init.par_roll,
                               ts.missing_roll){

  ## perform initial fit on "burnin" of first init_vals_roll time points
  fits_rolling<- KFAS_state_space_spline(ts_obs = ts_obs_roll[1:init_vals_roll],
                                    name = ts_name_roll,
                                    ts.missing = ts.missing_roll[1:init_vals_roll],
                                    ts_dates = dates_roll[1:init_vals_roll],
                                    init_par = init.par_roll)


  # just keep estimates for dates in burnins
  # smoother need not be kept
  fits_rolling <- dplyr::filter(fits_rolling,
                          date == dates_roll[1:init_vals_roll],
                          fit == "filter")


  # use variance estimates from burnin fit to initialize model for next time point
  next.par <- c(fits_rolling$sigv[init_vals_roll], fits_rolling$sigw[init_vals_roll])

  ## perform rolling estimation for each time point
  for(i in (init_vals_roll +1):length(ts_obs_roll)){
    # just looking to current time point
    ts_partial <- ts_obs_roll[1:i]

    # fit the model for the next time point
    ith_fit <- KFAS_state_space_spline(ts_obs = ts_partial,
                                    name = ts_name_roll,
                                    ts.missing = ts.missing_roll[1:i],
                                    ts_dates = dates_roll[1:i],
                                    init_par = next.par)
    # save results of model fit
    if(exists("ith_fit")){
      fits_rolling <- rbind(fits_rolling, dplyr::filter(ith_fit, date == dates_roll[i], fit
      # get updated variance estimates for observation and state
      next.par <- c(ith_fit$sigv[nrow(ith_fit)], ith_fit$sigw[nrow(ith_fit)])
      ## compute smoother at final time point
      if(i == length(ts_obs_roll)){
        fits_rolling <- rbind(fits_rolling, dplyr::filter(ith_fit, fit == "smoother"))
      }
      rm(ith_fit)
    }else{ ## I don't know how to error handling, feel free to do a pull request
      print(rep("FAIL", times = 100))
    }
  }
  ## give the user an update once each series' estimation is complete
  print(paste("Model fit complete: ", ts_name_roll[1]))
  return(fits_rolling)
```

```r
## source the custom functions
source("Code/KFAS_state_space_spline.R")
source("Code/KFAS_rolling_estimation.R")

## Rolling estimation for each series
fits_rolling_KFAS <- all_ts_observed %>%
  dplyr::group_nest(name, keep = T) %>%
  tibble::deframe() %>%
  purrr::map(., ~ {
      KFAS_rolling_estimation(ts_obs_roll= .x$value,
                              init_vals_roll = burnin,
                              ts_name_roll = .x$name,
                              dates_roll = .x$dates,
                              ts.missing_roll = .x$ts_missing,
                              init.par_roll = c(1,.2))
  })
```

```r
save(fits_rolling_KFAS, file = "Data/fits_rolling_KFAS")
```

```r
online_estimates <- fits_rolling_KFAS %>% dplyr::bind_rows() %>% dplyr::filter(fit == "filte
head(online_estimates)
```

One-step-ahead estimates

In state space models, in addition to obtaining the best estimate of "today's" state based on data through "today", the one-step-ahead forecasts can also be obtained: The estimate of tomorrow's state based on data through today.
The KFAS package makes this estimation simple. The function `KFAS_state_space_spline.R` can be augmented to return the one-step-ahead predictions from the `KFS` function by accessing the element `a`, i.e. `KFS(fit_mod$model)$a`

2.4 4. Retrospective Estimates

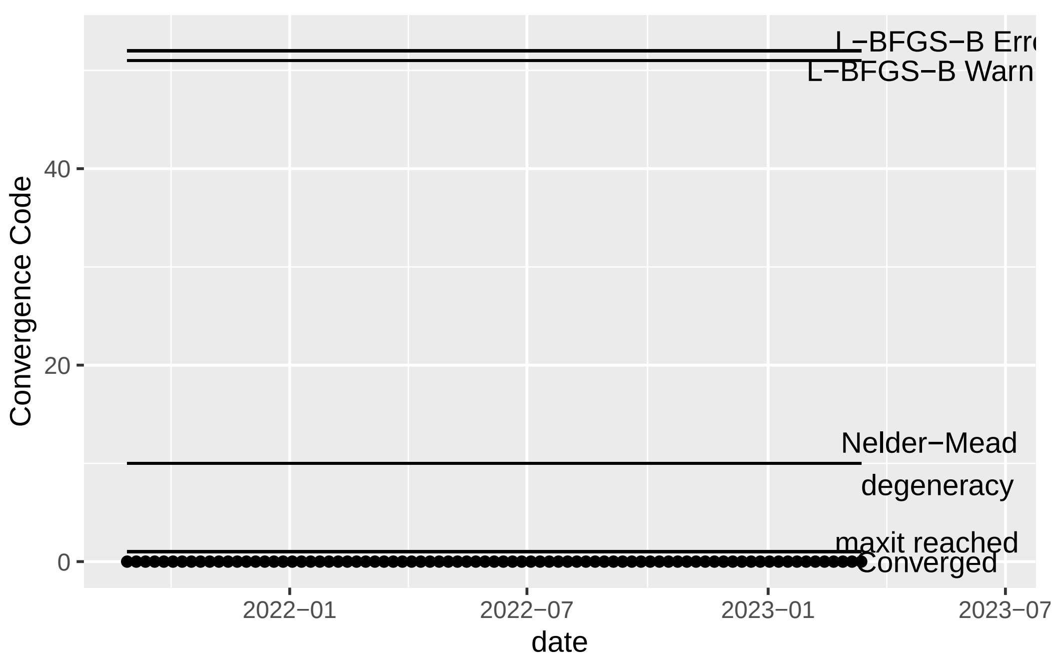
```
retro_estimates <- fits_rolling_KFAS %>% dplyr::bind_rows() %>% dplyr::filter(fit == "smoothe
head(retro_estimates)
```

2.5 5. Verify model fit

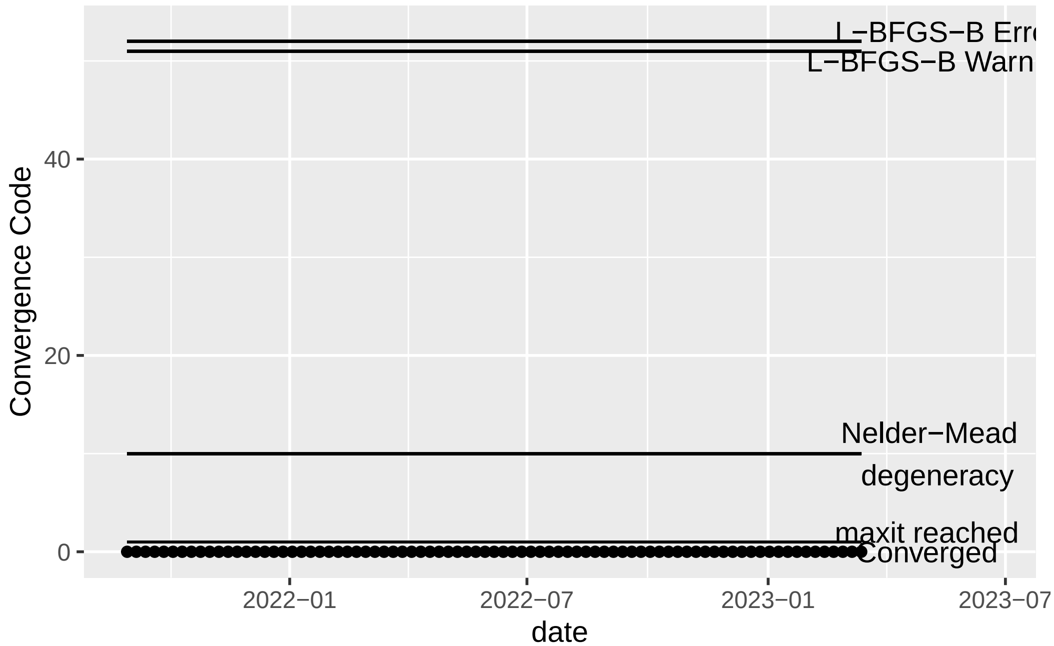
2.5.1 Convergence

2.5.1.1 Visuals

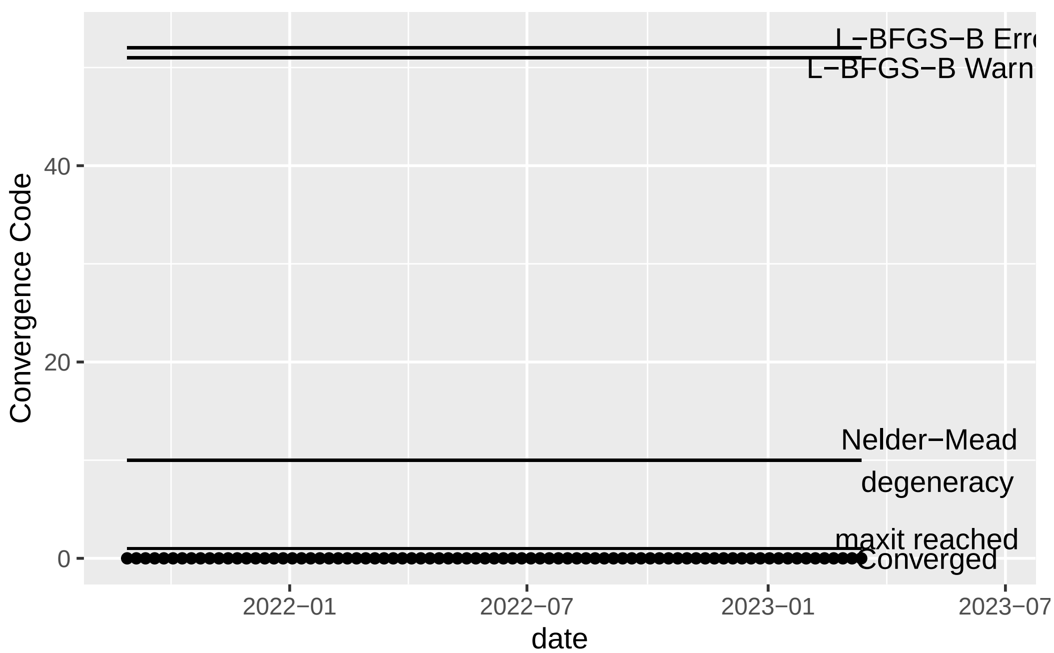
2.5.1.2 Lift station A



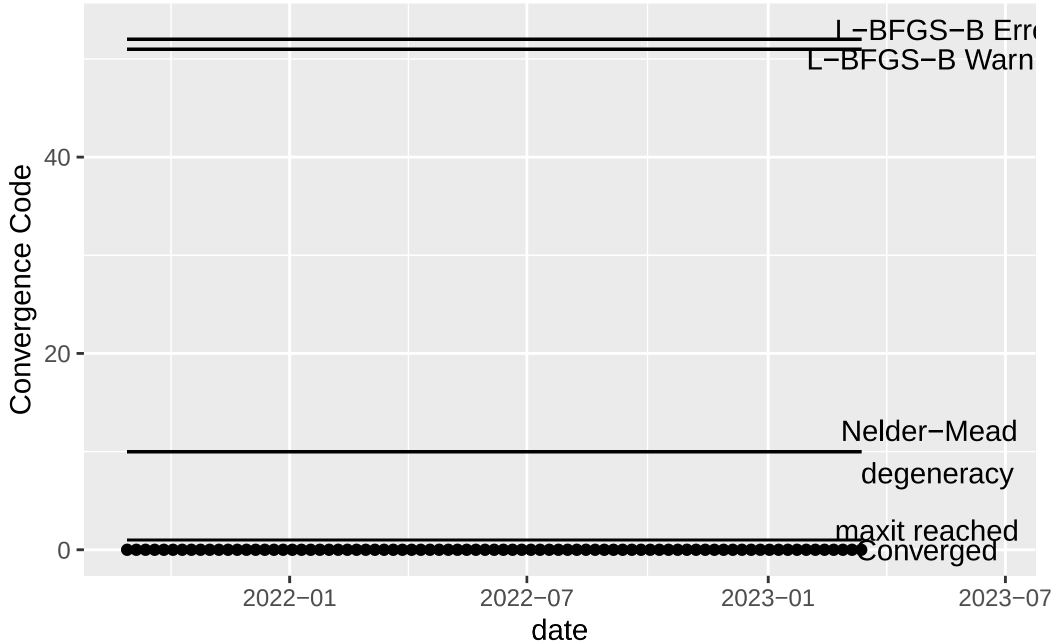
2.5.1.3 Lift station B



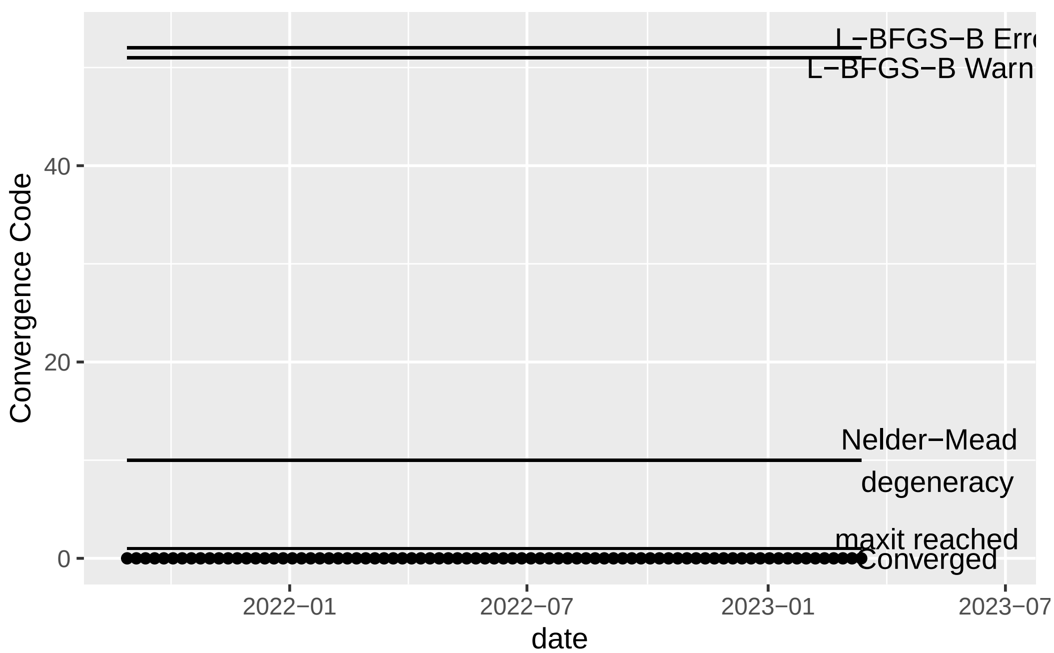
2.5.1.4 Lift station C



2.5.1.5 Lift station D



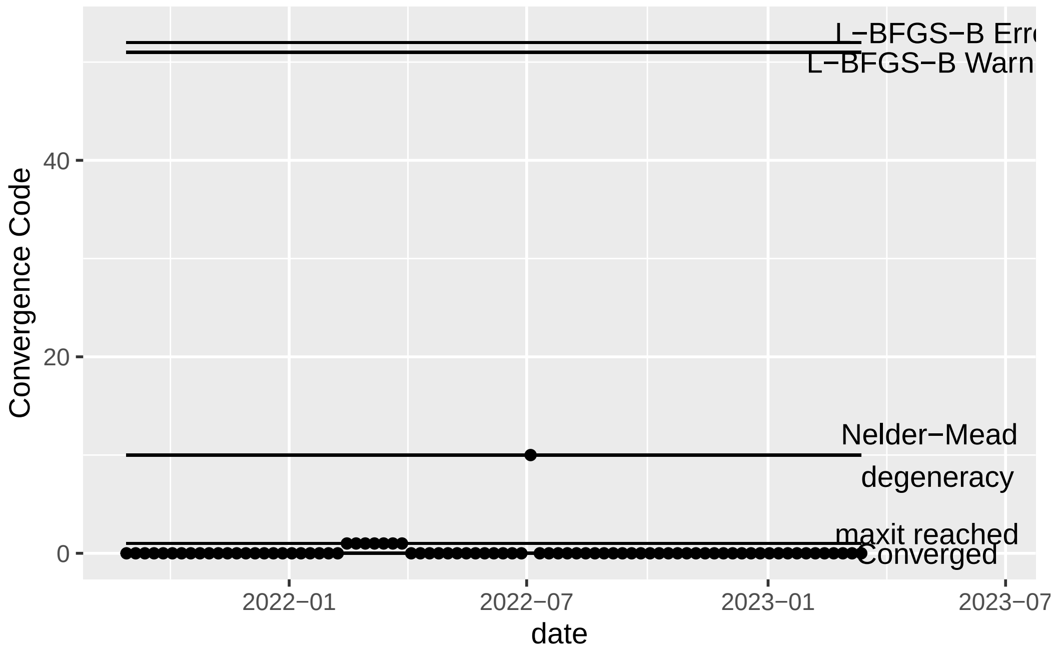
2.5.1.6 WWTP



Troubleshooting model convergence (hypothetical example)

If any of the models have not converged, you should not use the output of those models. Here's an example of what the above plots might look like if the model has not converged for some dates. In the hypothetical example below, the `maxit` parameter should be increased for the dates with error code 1 and the model corresponding to the date which gave error code 10 should be explored– perhaps there is a lot of missing data or an error was made in the data cleaning step, resulting in extreme values. Note that the `L-BFGS-B` error codes will only show up if the optimization method is changed to `L-BFGS-B`.

```
example <- fits_rolling_KFAS %>% dplyr::bind_rows() %>% dplyr::filter(fit == "filter" & dat
example$conv[25:31] <- 1
example$conv[45] <- 10

  ggplot2::ggplot(example, aes(x = date, y = conv)) +
                            ggplot2::geom_point() +
                            ggplot2::xlim(min(example$date), max(example$date) + 100) +
                            ggplot2::geom_segment(x = min(example$date), xend = max(examp
                            ggplot2::annotate("text", x = max(example$date) + 50, y = 0,
                            ggplot2::geom_segment(x = min(example$date), xend = max(examp
                            ggplot2::annotate("text", x = max(example$date) + 50, y = 2,
                            ggplot2::geom_segment(x = min(example$date), xend = max(examp
                            ggplot2::annotate("text", x = max(example$date) + 55, y = 10,
                            ggplot2::geom_segment(x = min(example$date), xend = max(examp
                            ggplot2::annotate("text", x = max(example$date) + 60, y = 50,
                            ggplot2::geom_segment(x = min(example$date), xend = max(examp
                            ggplot2::annotate("text", x = max(example$date) + 65, y = 53,
                          ggplot2::ylab("Convergence Code")
```



If `example` is the value of `online_estimates`for one location, the dates corresponding to models with convergence issues can be returned using the following code snippet.

```
example %>% dplyr::filter(conv >0) %>% dplyr::pull(date, conv)
```

```
           1            1            1            1            1            1
"2022-02-14" "2022-02-21" "2022-02-28" "2022-03-07" "2022-03-14" "2022-03-21"
           1           10
"2022-03-28" "2022-07-04"
```

2.5.2 Residuals

Chapter 2 of Time Series: A Data Analysis Approach Using R

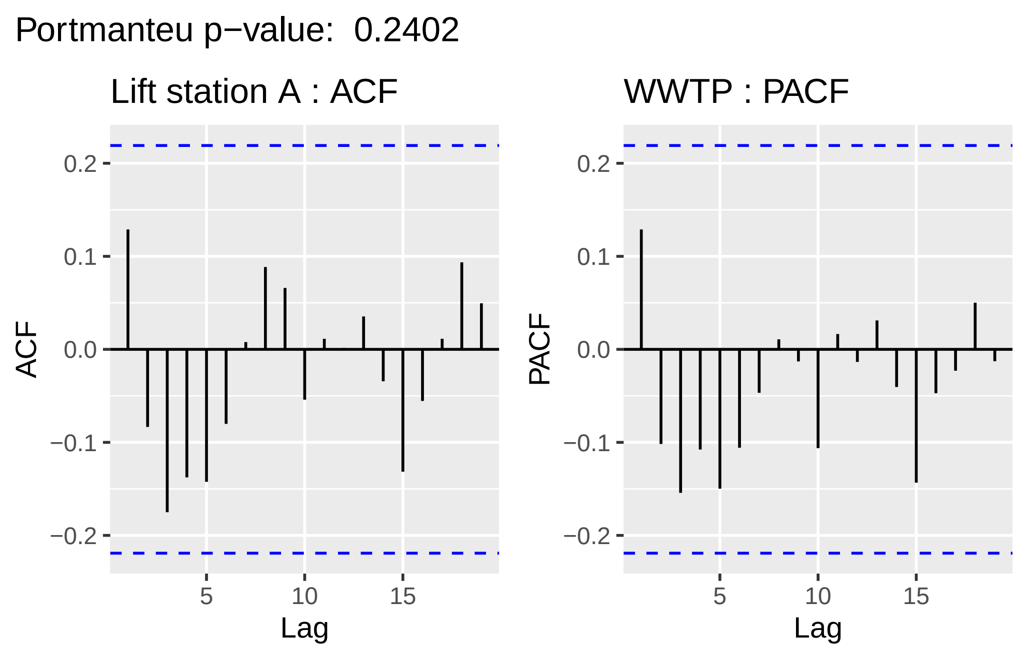
```
resid <- fits_rolling_KFAS$`WWTP` %>%  ## just look at the WWTP
        dplyr::filter(date > date_burnin & fit == "filter") %>% ## filter estimates beyond b
        dplyr::mutate(resid = obs-est) %>% dplyr::pull()# calcualte residual

# Ljung-Box test
LB_test <- Box.test(resid, type = "Ljung-Box")

# make acf plot
#resid %>% acf1(main = "Autocorrleation plot for Resid = 69th St. Observed - 69th St. Filter
#text(x = 1, y = .35, labels = paste("Portmanteau test p-value#: ", round(LB_test$p.value, 4
```

Residual plots for all series

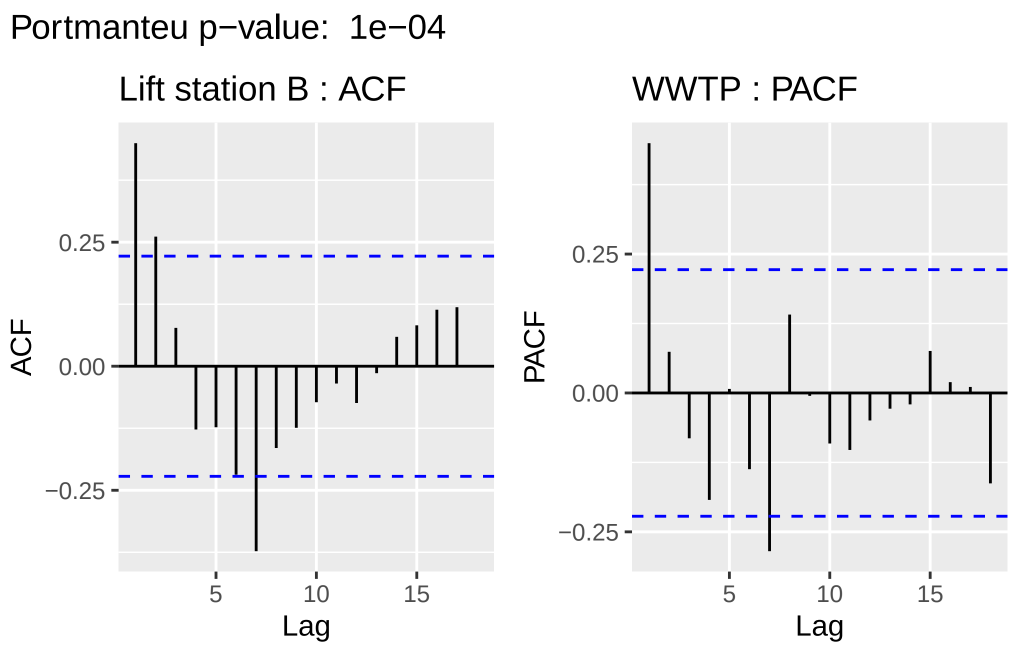
```
TS <- fits_rolling_KFAS %>%
    dplyr::bind_rows() %>%
    dplyr::filter(fit == "filter" & date >= date_burnin & name == "WWTP")

resid_plots <- fits_rolling_KFAS %>%
                        dplyr::bind_rows() %>%
                        dplyr::filter(fit == "filter" & date >= date_burnin) %>%
                        dplyr::group_nest(name, keep = T) %>%
                        tibble::deframe() %>%
                        purrr::map(., ~{

                          ## impute missing values in .x$resid
                          resid = zoo::na.approx(.x$resid)
                          # compute p-value
                          LB_test <- stats::Box.test(resid, type = "Ljung-Box")
                          # create acf and pacf plots
                          acf <- forecast::ggAcf(resid, main = paste(.x$name[1], ": ACF"))
                          pacf <- forecast::ggPacf(resid, main = paste(TS$name[1], ": PACF
                          # output single visual for rendering in tabs
                          acf + pacf + patchwork::plot_annotation(title = paste("Portmante
                        })
```

2.5.2.1 Visuals

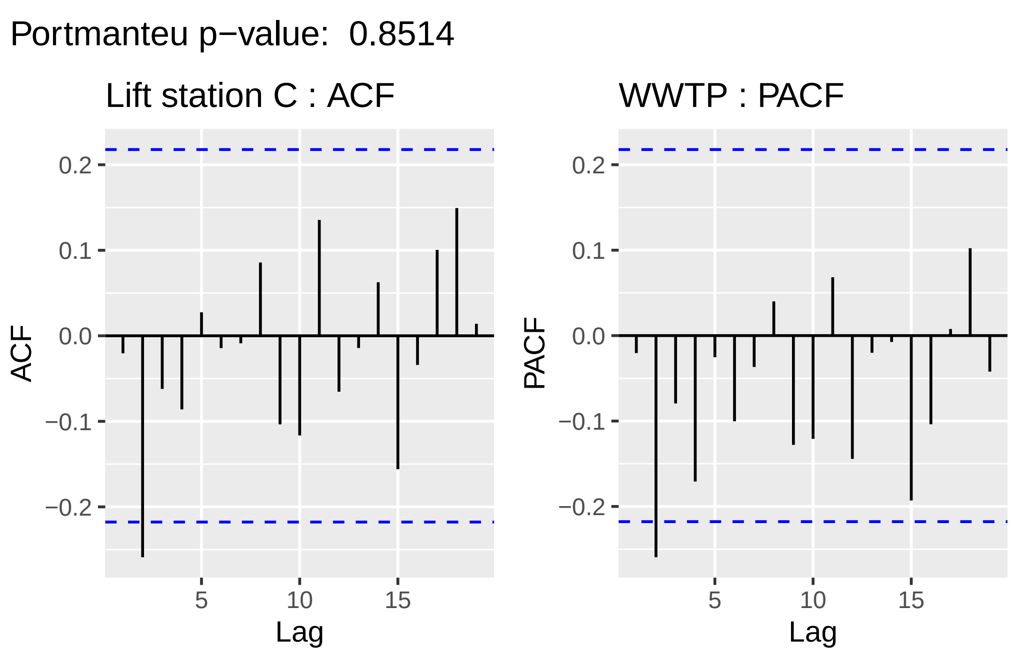
2.5.2.2 Lift station A



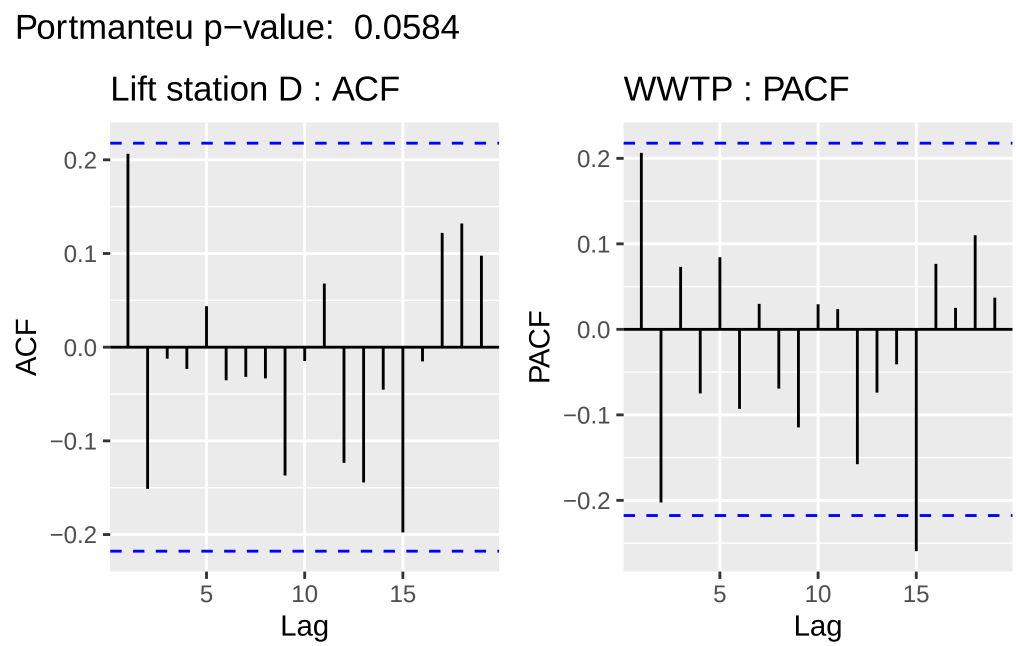
2.5.2.3 Lift station B



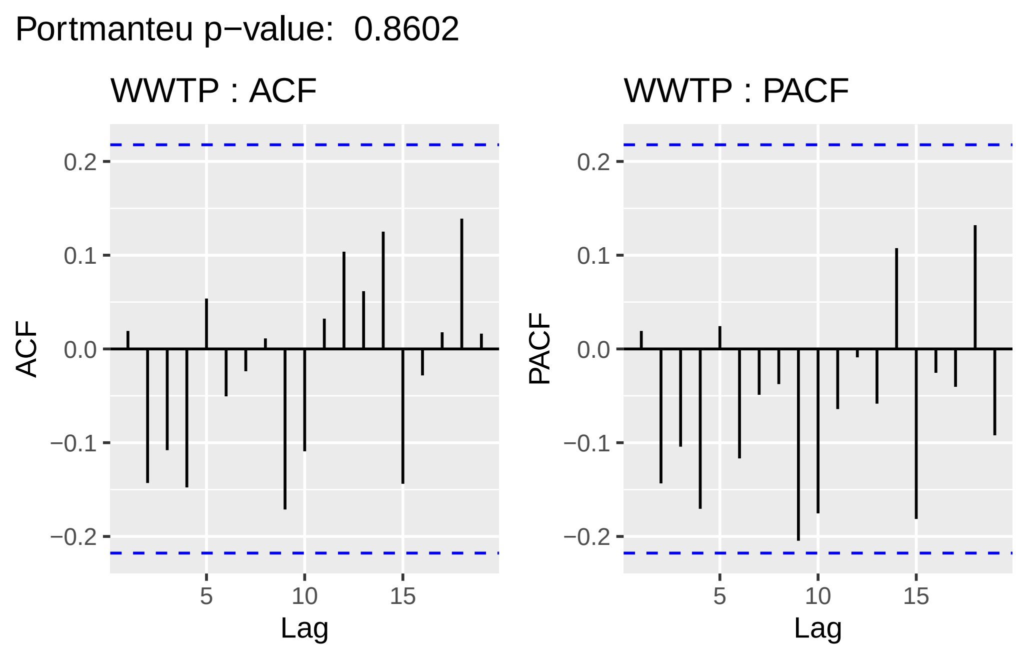
2.5.2.4 Lift station C



2.5.2.5 Lift station D



2.5.2.6 WWTP



Why is lift station B showing significant autocorrelation?

This is cold be due to a linear imputation of almost half of the missing values, which are missing in a big chunk.

```
all_ts_observed %>% dplyr::group_by(name) %>%dplyr::summarise(missing = sum(ts_missing),
```

```
# A tibble: 5 x 4
  name           missing total percent
  <chr>            <int> <int>   <dbl>
1 Lift station B      42    95   44.2
2 Lift station A      28    95   29.5
3 Lift station D      18    95   18.9
4 Lift station C       9    95    9.47
5 WWTP                 4    95    4.21
```

2.6 6. Compare the variances

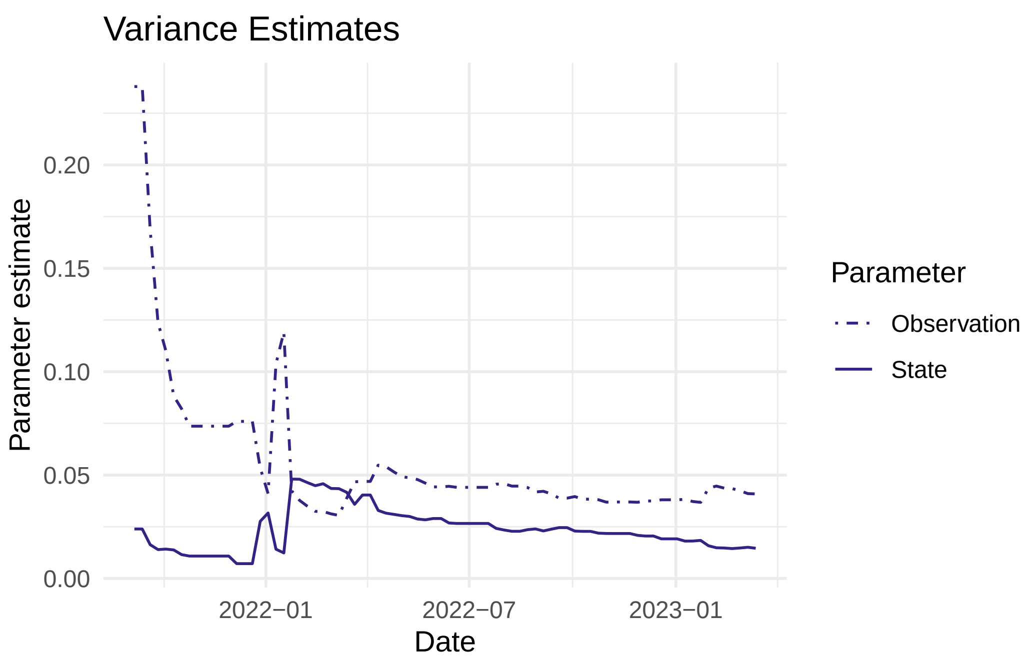
```
par_est_plots <- fits_rolling_KFAS %>% dplyr::bind_rows() %>%
                  dplyr::mutate(colors = rep(c("#332288", "#AA4499","#44AA99","#88CCEE", "#DD
                 dplyr::filter(fit == "filter" & date >date_burnin) %>%
                 tidyr::pivot_longer(cols = c("sigv", "sigw"), names_to = "par", values_to =
                 dplyr::group_nest(name, keep = T) %>%
                 tibble::deframe() %>%
                 purrr::map(., ~{
                    ggplot2::ggplot(.x, aes(x = date, y = var_est, lty = par)) +
                      ggplot2::geom_line(col = .x$colors[1]) +
                      ggplot2::theme_minimal() +
                      ggplot2::scale_linetype_manual(values = c(4,1), labels = c("Observation
                      ggplot2::labs(linetype = "Parameter", x = "Date", y = "Parameter estima
                          #missing_dates_lwr <- unique(.x$date[which(.x$ts_missing)])
                  #missing_dates_upr <- unique(.x$date[which(.x$ts_missing)+1])[1:length(miss

                  #p+ geom_vline(xintercept = missing_dates_lwr)
```

```
                  })

## Final variance estimates
fits_rolling_KFAS %>% dplyr::bind_rows() %>% dplyr::filter(fit == "filter" & date == "2023-03
```

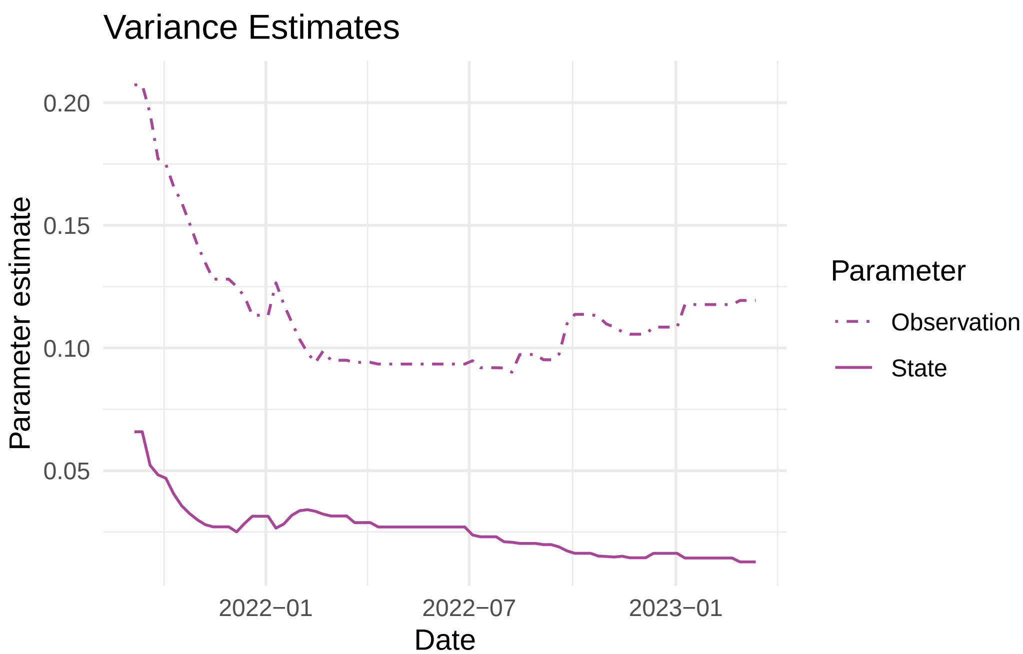
Variance visualizations

2.6.0.1 Visuals

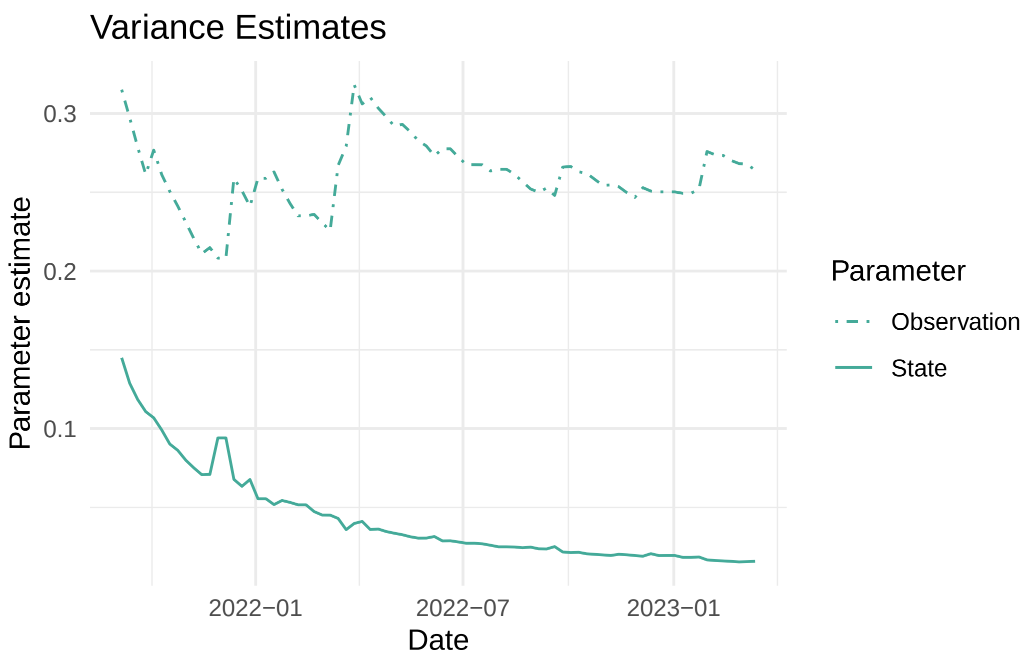
2.6.0.2 Lift station A



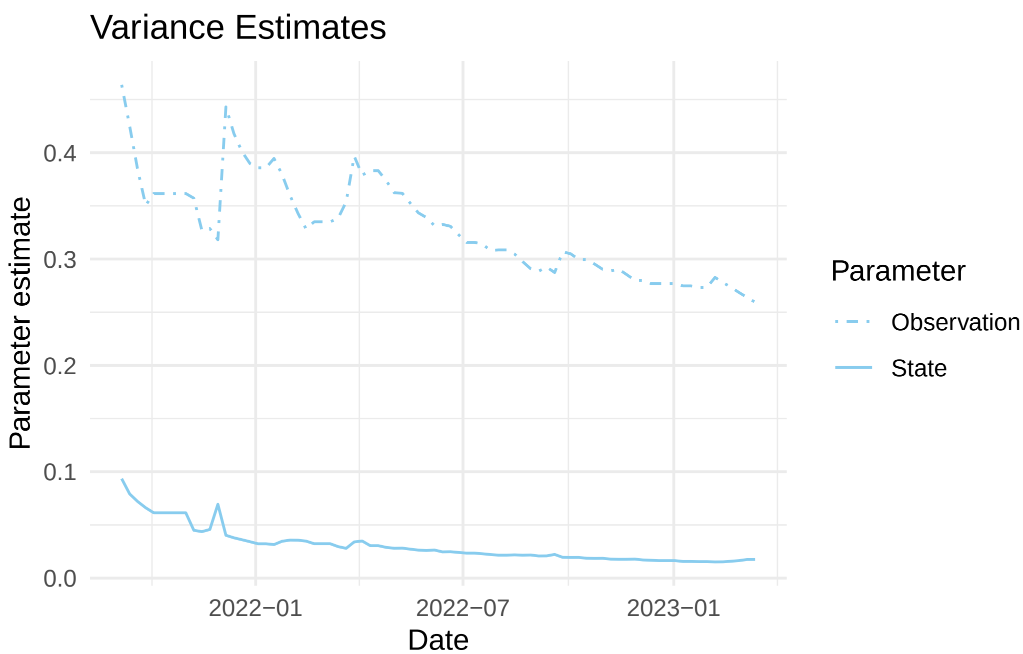
2.6.0.3 Lift station B



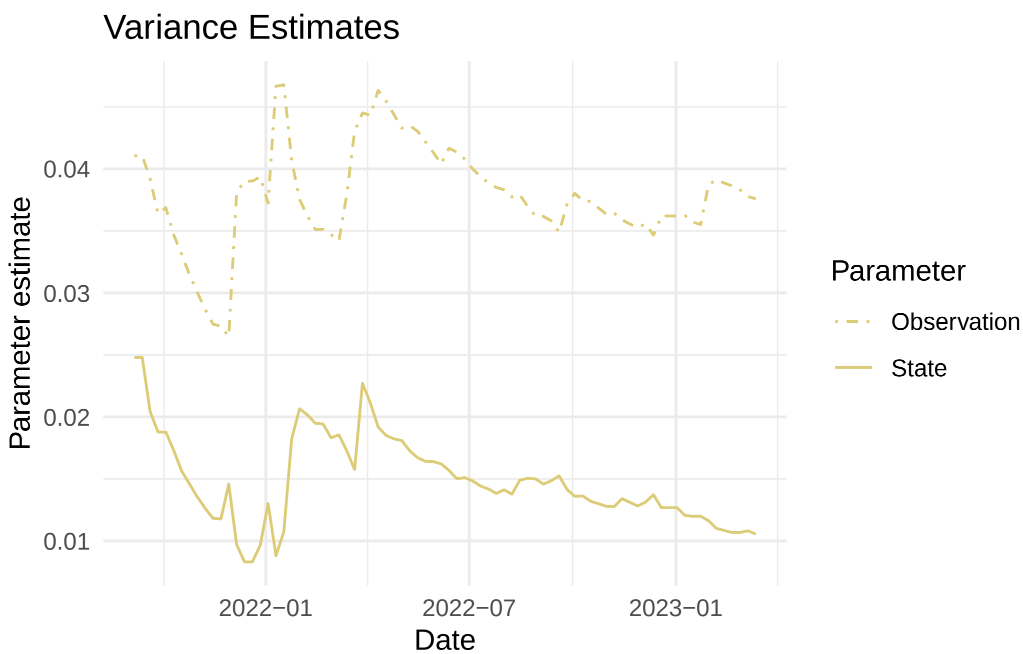
2.6.0.4 Lift station C



2.6.0.5 Lift station D



2.6.0.6 WWTP



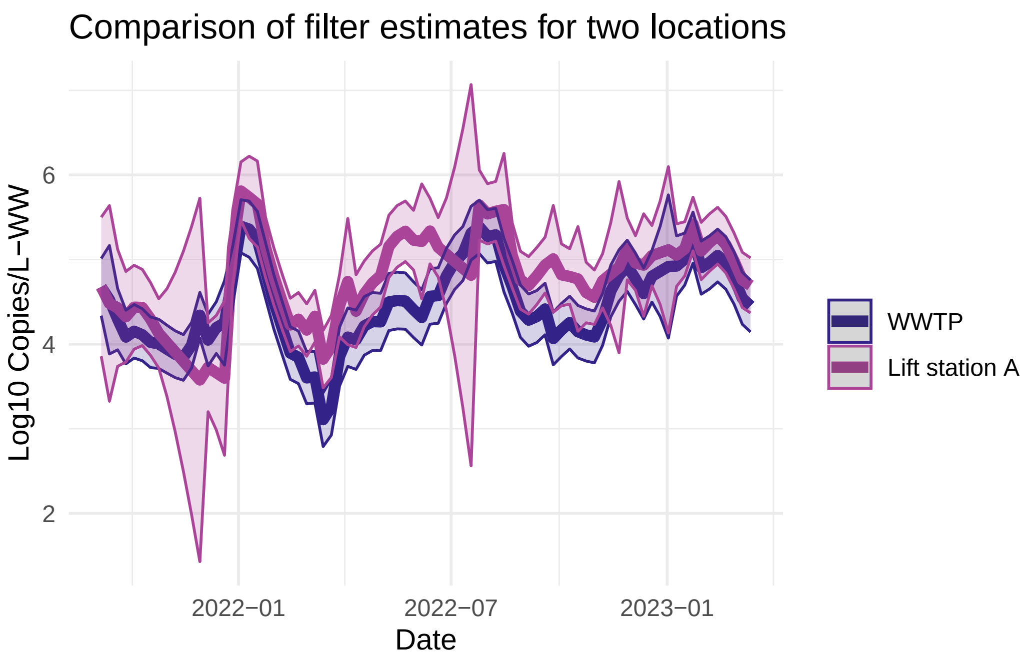
2.7 7. Visualize estimates

2.7.1 Filter estimates

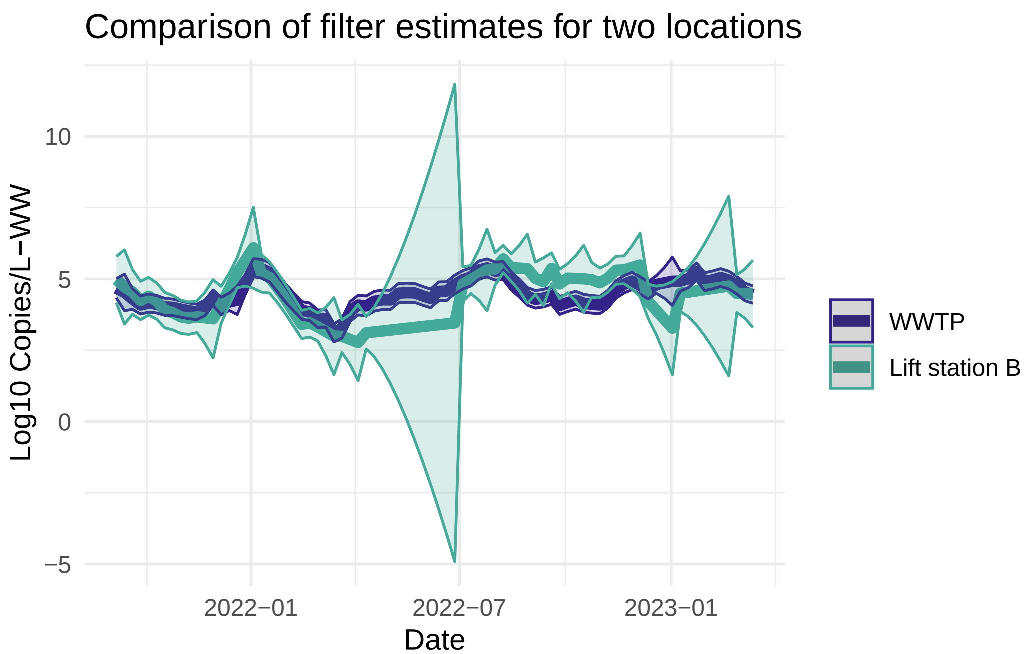
```
#state filter?

library(ggplot2)
library(tidyverse)
source("Code/fplot.R")
load("Data/fits_rolling_KFAS")
# plotting the smoothers for all the series
filter_plots <- fits_rolling_KFAS %>% dplyr::bind_rows() %>%
          dplyr::mutate(colors = rep(c( "#AA4499","#44AA99","#88CCEE", "#DDCC77", "#332288")
          dplyr::filter(name != "WWTP" & fit == "filter" & date > date_burnin) %>%
          dplyr::group_nest(name, keep = T) %>%
          tibble::deframe() %>%
          purrr::map(., ~ {
            plot.dat <- dplyr::bind_rows(dplyr::filter(fits_rolling_KFAS$`WWTP`, fit == "fil
                        plot.dat$name <- factor(plot.dat$name, levels(factor(plot.dat$name))
            #fplot(f= plot.dat, title_char = "Comparison of filter estimates for two locatio
            ggplot2::ggplot(plot.dat, aes(x = date, y = est, color = name, fill = name)) +
             ggplot2::geom_line(linewidth=2) +

        ggplot2::theme_minimal()+

        ggplot2::geom_ribbon(aes(ymin=lwr,ymax=upr),alpha=.2) +

        ggplot2::scale_color_manual(values = c("#332288", .x$colors[1])) +

        ggplot2::scale_fill_manual(values = c(paste("#332288", "50", sep = ""), paste(.x$col
                ggplot2::labs(title = "Comparison of filter estimates for two locations", x=

          })
```

2.7.1.1 Visualizations

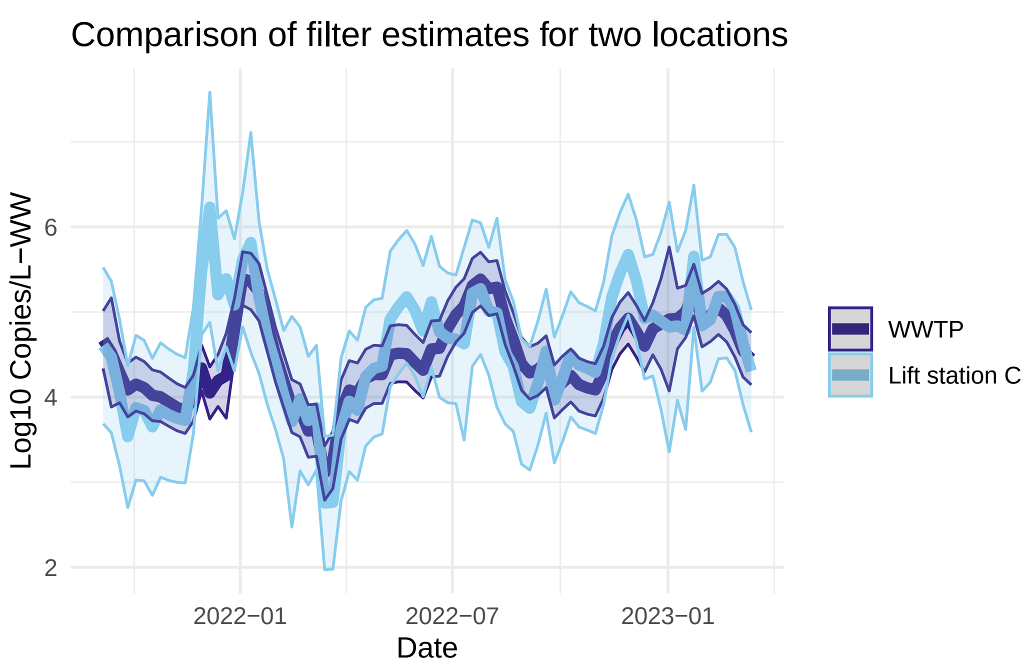
2.7.1.2 Lift station A



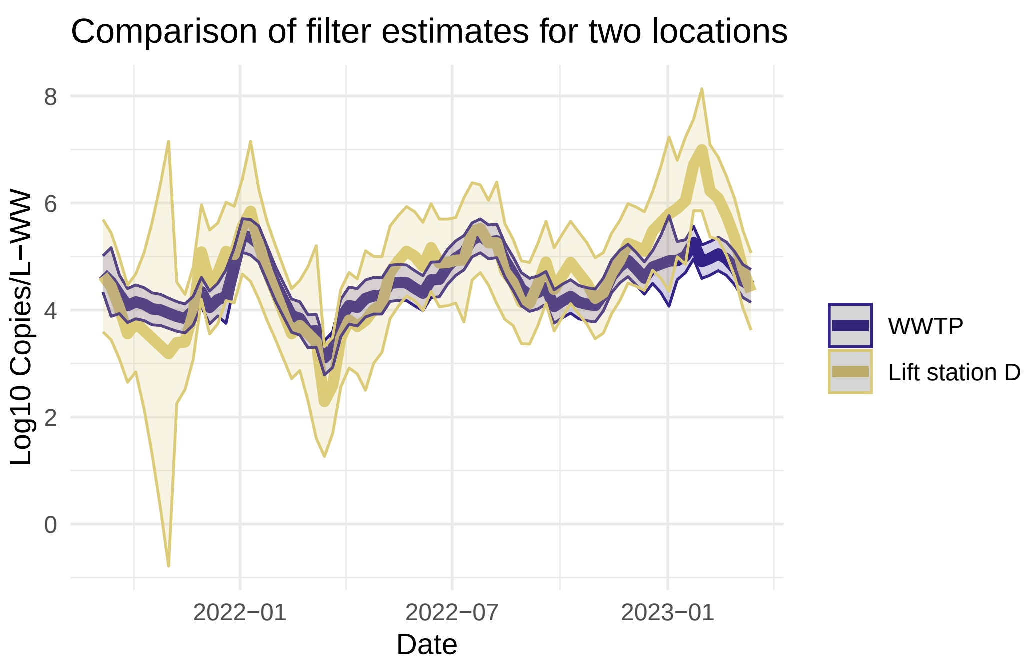
2.7.1.3 Lift station B



2.7.1.4 Lift station C



2.7.1.5 Lift station D

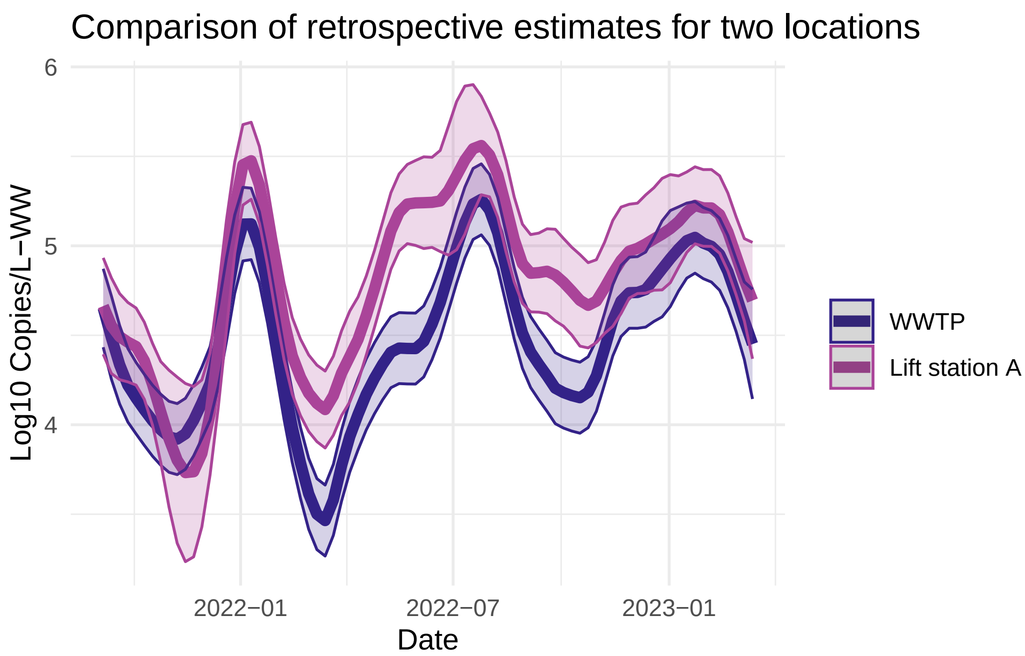


2.7.2 Smoother estimates

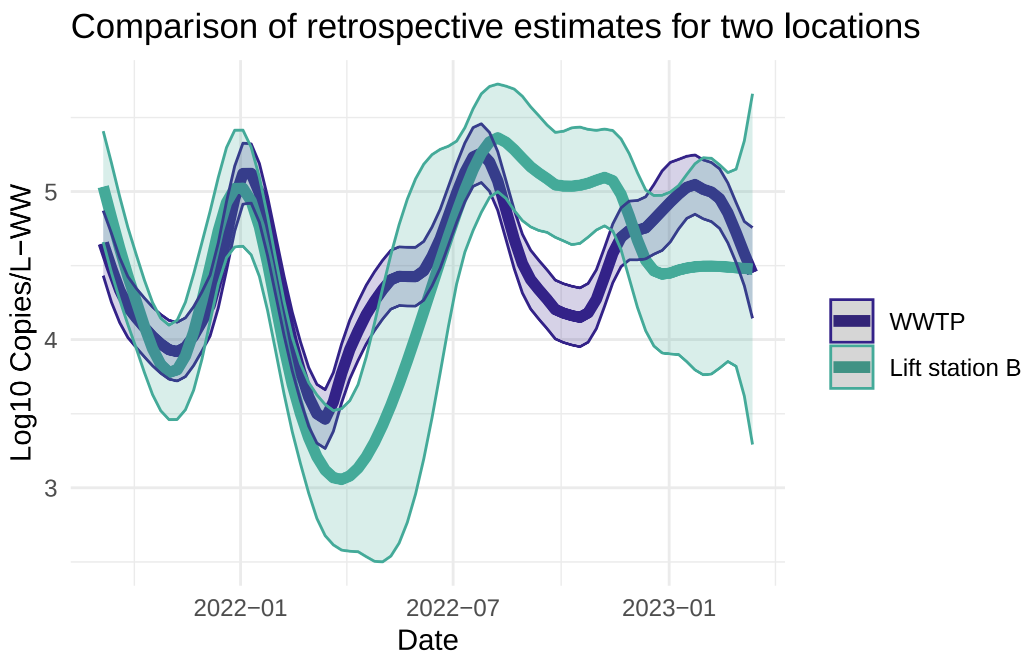
```
library(ggplot2)
source("Code/fplot.R")
# plotting the smoothers for all the series
smoother_plots <- fits_rolling_KFAS %>% dplyr::bind_rows() %>%
                  dplyr::mutate(colors = rep(c( "#AA4499","#44AA99","#88CCEE", "#DDCC77", "#3
          dplyr::filter(name != "WWTP" & fit == "smoother"& date > date_burnin) %>%
          dplyr::group_nest(name, keep = T) %>%
          tibble::deframe() %>%
          purrr::map(., ~ {
            plot.dat <- dplyr::bind_rows(filter(fits_rolling_KFAS$`WWTP`, fit == "smoother" &
            plot.dat$name <- factor(plot.dat$name, levels(factor(plot.dat$name))[2:1])
            fplot(f= plot.dat, title_char =  "Comparison of retrospective estimates for two l
          })
```

2.7.2.1 Visualizations

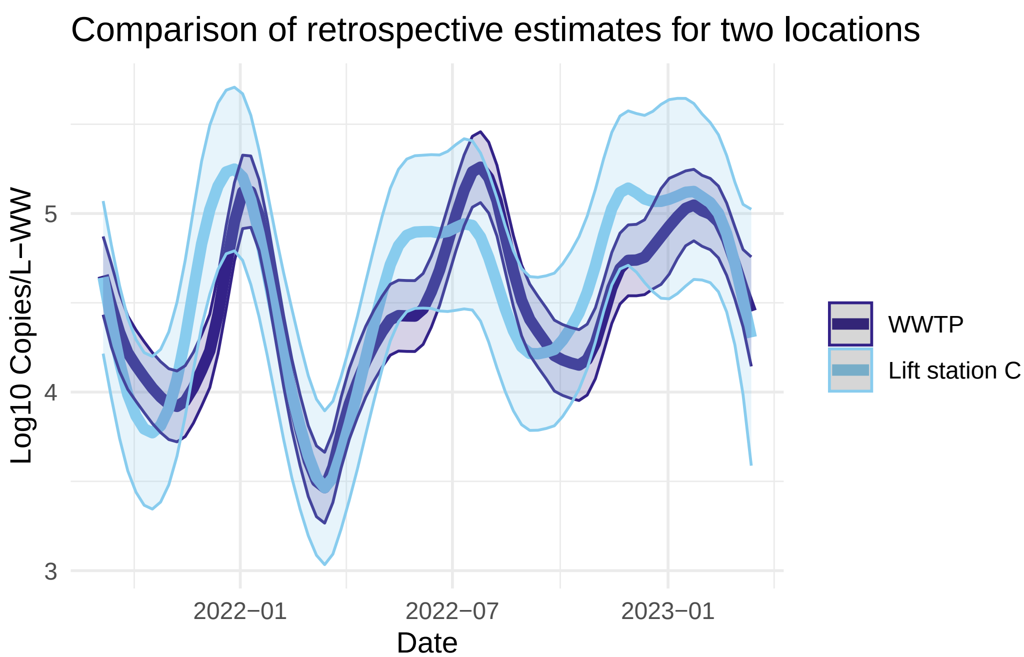
2.7.2.2 Lift station A



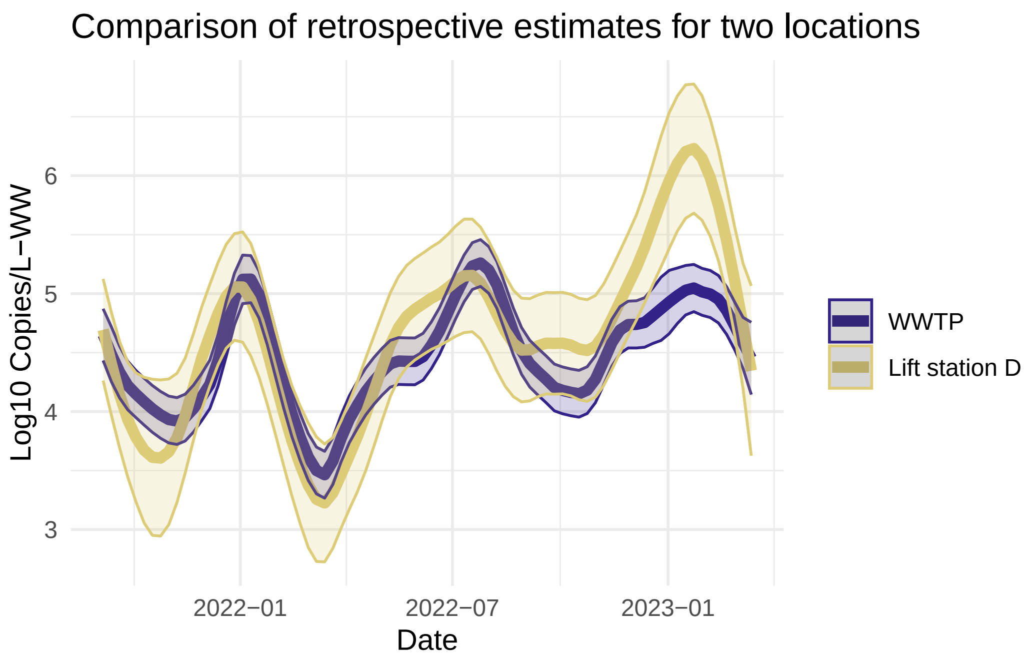
2.7.2.3 Lift station B



2.7.2.4 Lift station C



2.7.2.5 Lift station D



Part II

Detection of Deviations

3 1. Obtain trend estimates

```r
library(tidyverse)
library(qcc)
library(ggnewscale)
load("Data/fits_rolling_KFAS")
```

3.1 2. Handle missing data in LS series

```r
# Subset to WWTP
mu_t <- fits_rolling_KFAS$`WWTP` %>% dplyr::filter(date> "2021-08-30"& fit == "filter")


# Observed LS series and dates
y_t_all <- fits_rolling_KFAS$`Lift station B` %>% dplyr::filter(date> "2021-08-30" & fit == "

#Replace missing values in lift station series
y_t_all[y_t_all$ts_missing, "obs"] <- dplyr::left_join(y_t_all[y_t_all$ts_missing,], mu_t, by

# Keep just the series of observations and filled-in missing values
y_t <- y_t_all %>% dplyr::pull(obs)

# Just the online estimates for WWTP
mu_t <- fits_rolling_KFAS$`WWTP` %>% dplyr::filter(date> "2021-08-30"& fit == "filter") %>% d
```

3.2 3. Create difference time series

```
## compute the raw differences (numerator of d_tilde)
diff <- y_t-mu_t
```

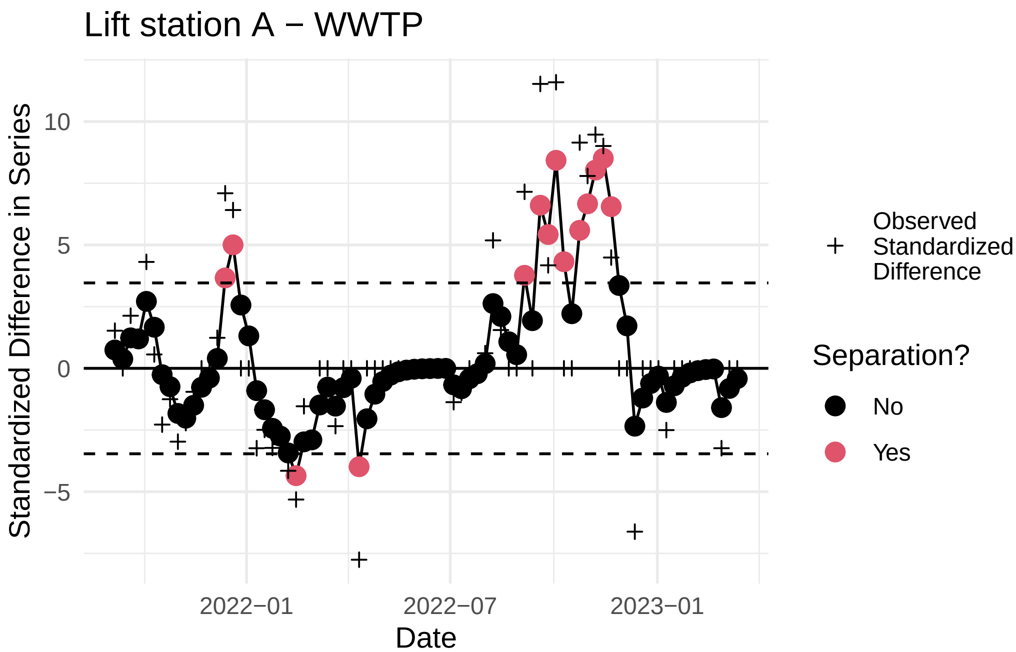
3.3 4. Standardize the difference series

```
var_y <- fits_rolling_KFAS$`Lift station B` %>%
          dplyr::filter(date > "2021-08-30" & fit == "filter")%>%
          dplyr::mutate(var_est = sigv^2) %>% select(var_est)
var_mu <- fits_rolling_KFAS$`WWTP` %>%
            dplyr::filter(date > "2021-08-30" & fit == "filter") %>%
            dplyr::mutate(var_est = ((upr-est)/2)^2) %>%
            dplyr::select(var_est)

## compute the estimated covariance (scaled product of variances)
cor_estimate <- cor(y_t, mu_t, use ="pairwise.complete.obs")
cov_est <- as.numeric(cor_estimate)*sqrt(var_y$var_est)*sqrt(var_mu$var_est)

## compute approximate variance using covariance (this is the correct variance for the numera
var_est <- var_y$var_est + var_mu$var_est -2*cov_est

## standardize
standardized_diff <- diff/sqrt(var_est)
```

3.4 5. Construct EWMA chart

```
 # compute lag 1 autocorrelation of standardized difference series
  lag1_est <- acf(standardized_diff, plot=F, na.action = na.pass)$acf[2] ## we could do some

  # use qcc package to make ewma plot
```

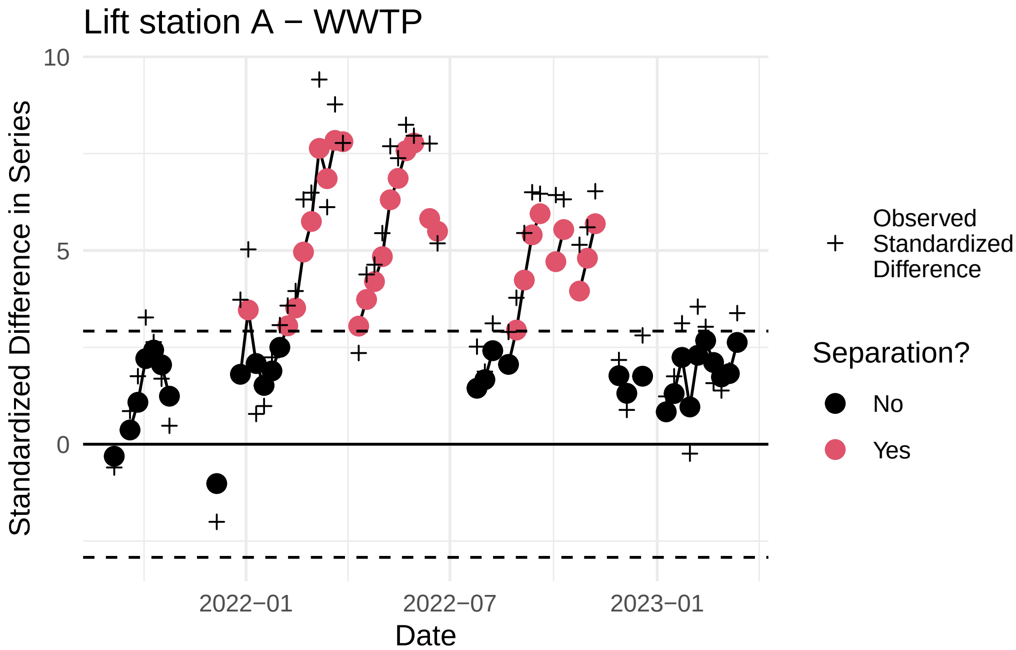
```r
out <- qcc::ewma(standardized_diff, center = 0, sd = 1,
                 lambda = lag1_est, nsigmas = 3, sizes = 1, plot = F)



## put NAs where we had missing values for either series
out$y[is.na(y_t)] <- NA
out$data[is.na(y_t),1]<- NA

dates <- dplyr::filter(fits_rolling_KFAS$WWTP,
                       date > "2021-08-30" & fit == "filter") %>% dplyr::pull(date)

# create plot
dat <- data.frame(x = dates,
                  ewma = out$y,
                  y = out$data[,1],
                  col = out$x %in% out$violations,
                  lwr = out$limits[20,1],
                  upr = out$limits[20,2])
obs_dat <- data.frame(x = dat$x, y = out$data[,1], col = "black")
p <- ggplot2::ggplot(dat, aes(x = x, y = ewma)) +
  ggplot2::geom_vline(xintercept = NULL,
             col = "darkgrey",
             lwd = 1) +
  ggplot2::geom_line()+
  ggplot2::geom_point(aes(col = dat$col), size = 3) +
  ggplot2::scale_color_manual(values = c(1,2), label = c("No", "Yes"), name = "Separation?"
  ggnewscale::new_scale_color() +
  ggplot2::geom_point(data = obs_dat, aes(x = x, y = y, col =col), shape = 3) +
  ggplot2::scale_color_manual(values = "black", label = "Observed \nStandardized \nDifferen
  ggplot2::geom_hline(aes(yintercept = out$limits[20,1]), lty = 2) +
  ggplot2::geom_hline(aes(yintercept = out$limits[20,2]), lty = 2) +
  ggplot2::geom_hline(aes(yintercept = 0), lty = 1) +
  ggplot2::ggtitle("Lift station A - WWTP")+
  ggplot2::xlab("Date") + ggplot2::ylab("Standardized Difference in Series") +
  ggplot2::theme_minimal()

p
```



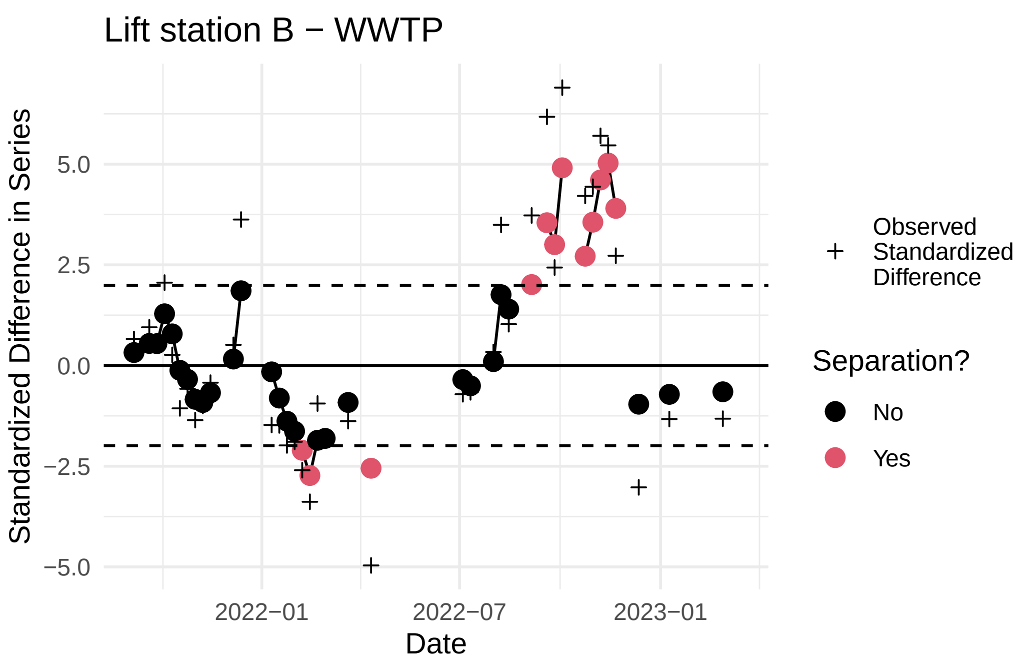
3.5 6. Inspect EWMA chart

```
source("Code/ww_ewma.r")

mu2 <- fits_rolling_KFAS$`WWTP` %>% dplyr::filter(fit == "filter" & date > "2021-08-30")

ewma_plots <- fits_rolling_KFAS %>% dplyr::bind_rows() %>%
          dplyr::filter(name != "WWTP" & fit == "filter" & date > "2021-08-30") %>%
          dplyr::group_nest(name, keep = T) %>%
          tibble::deframe() %>%
          purrr::map(., ~ {
               ww_ewma(.x, mu2, paste(.x$name[1], "-", mu2$name[1]))

                          })
```

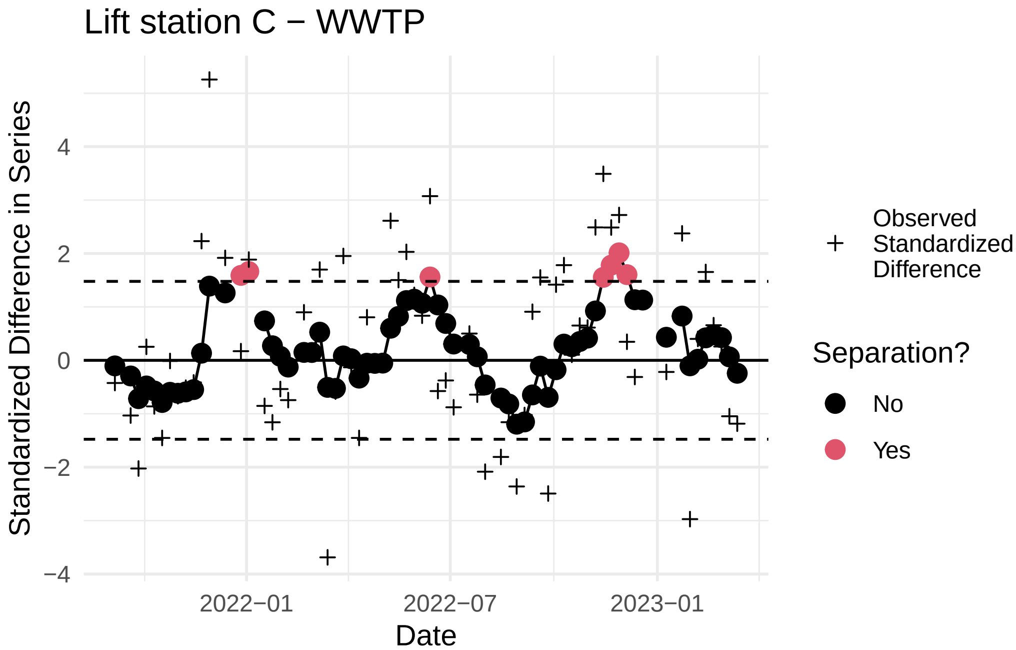
3.5.0.1 Lift station A



3.5.0.2 Lift station B



3.5.0.3 Lift station C



3.5.0.4 Lift station D