STAT 5511 Homework 7
Solution

General formatting guidelines:

The usual formatting rules:

- Your homework (HW) should be formatted to be easily readable by the grader.
- You may use knitr or Sweave in general to produce the code portions of the HW. However, the output from knitr/Sweave that you include should *be only what is necessary to answer the question*, rather than just any automatic output that R produces. (You may thus need to avoid using default R functions if they output too much unnecessary material, and/or should make use of `invisible()` or `capture.output()`.)
  - For example: for output from regression, the main things we would want to see are the estimates for each coefficient (with appropriate labels of course) together with the computed OLS/linear regression standard errors and p-values. If other output is not needed to answer the question, it should be suppressed!
- Code snippets that directly answer the questions can be included in your main homework document; ideally these should be preceded by comments or text at least explaining what question they are answering. Extra code can be placed in an appendix.
- All plots produced in R should have appropriate labels on the axes as well as titles. Any plot should have explanation of what is being plotted given clearly in the accompanying text.
- Plots and figures should be *appropriately sized,* meaning they should not be too large, so that the page length is not too long. (The arguments fig.height and fig.width to knitr chunks can achieve this.)
- **Directions for "by-hand" problems:** In general, credit is given for (correct) shown work, not for final answers; so show **all** work for each problem and explain your answer fully.

Instructions: For this homework, you will analyze two data sets. Find the file `hw7dat.rsav` on the course webpage. Load it into R by running `load("path/hw7dat.rsav")` where 'path/hw7dat.rsav' is replaced by the full path on your hard drive to the file `hw7dat.rsav` (the syntax for which is operating system dependent). The file contains multiple data objects: `dat1`,`dat2`. Each of the two datasets is a separate homework question. The analysis for each dataset should begin on a new page and should have as label the name of the dataset (dat1, dat2). Your tasks are different for the three datasets and they are as follows.

1. For `dat1`: `dat1` has two columns, `yy` and `xx`, which are both time series. Your job is to regress `yy` on `xx` using the ("transfer modelling") methodology we discussed in class.
2. For `dat2` your job is to use the local level type models that were demonstrated in class, using the `dlm` R package. The dataset has 4 columns. The data is related to wastewater measurements about COVID-19 from the Twin Cities wastewater treatment plant, used for monitoring the disease in the Twin Cities area. The first column may be ignored. The second column is the date of each measurement. The third and fourth columns are the measurements.
   (a) Plot the data and the local level filter and the local level smoother like we did in class for just the third column of data. That is there is a single (one dimensional) observation equation.
   (b) Explain the differences between the smoother output and the filter output.
   (c) Plot the data and the local level filter and the local level smoother like we did in class based on both the third and fourth columns of the data. That is there is a bivariate (two dimensional) observation equation and two observations at each time point.
   (d) Explain the differences between the output based on just one of the series and the smoother/filter output based on both.

Presentation/formatting rules: for question 1, your output should be in the following format. Points will be deducted if it is not.

- On the first page of the assignment you should state on which page each question begins. [Note: One simple way to automate this in LaTeXis to use `\section{}` to start each dataset and then include a `\tableofcontents`. You could also use `\label{}` and `\pagref{}` commands.]

- On the first page of output for each problem, you should first have a summary (labeled "Summary") that provides the model chosen, parameter estimates, standard errors, and p-values in that model. Specify explicitly if you exclude a constant term. For example, "For the series $Y_t = X_t^{1/2}$, I chose an SARIMA$(1, 2, 3) \times (4, 5, 6)_7$ model, including intercept term. The parameter estimates were ...". If you believe the data cannot distinguish between two (or more) models you should describe both (all) of them in this manner here. In the case of a regression model, you should explain the full model, meaning which lags of which variables are included in the regression model as well as what the ARMA model of the errors is.

- After the summary should be an explanation (labeled "Explanation"). Provide a clear explanation of why you selected the model you selected. Refer to the output of your analysis, which will be below. The model selection and diagnostic techniques we have discussed in class can be discussed here. You do not need to (and should not) provide an exhaustive list of all possible models, but should rather provide explanation for which models were reasonable contenders (and why), and which model (or models) were the best out of those contenders (and why).

- After the explanation is the "Output" you refer to in your explanation. (The output may be plots or output from various commands.) All of it should be clearly formatted, and labeled or described. You do not need to provide exhaustive output from every command you have run, but you should include enough to justify all the arguments you make in your summary.

Finally, *in Question 1 the two series are named xx and yy and you should not rename them.*

Solution:
Please note: this solution is not formatted according to the necessary guidelines.

```r
load("hw7.rsav")
library(astsa)
```

Dat1

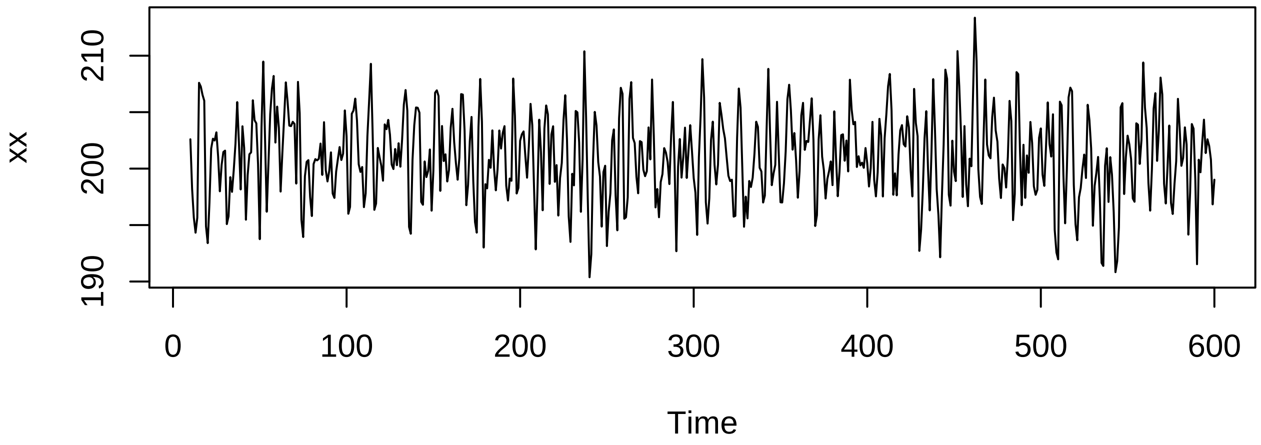
Summary: I choose to fit the model $Y_t = -6.5233 + 1.2813 X_{t-3} + U_t$. Here $U_t = \eta_t + 0.1878\eta_{t-1} + 0.5652\eta_{t-2}$ where η_t's are white noises. See the table below for the parameter estimates, standard errors, and p-values.

Coefficient	Estimate	Standard Error	p-value
MA_1	0.188	0.035	0.00
MA_2	0.565	0.033	0.00
intercept	-6.52	5.90	0.27
xreg	1.28	0.029	0.00

Table 1: Parameter estimates, standard errors, and p-values for dat3

Explanation: We first plot the sample ACF and PACF of X_t and arrive at an AR(2) model with fitted residuals $\tilde{W}_t$ to this input series. Then we use this AR operator of X_t to get the transformed output series $\tilde{Y}_t$. From the plot of the cross-correlation between $\tilde{Y}_t$ and $\tilde{W}_t$ we see that the third leg of X_t should be used. We fit the model $Y_t = \alpha + \delta X_{t-3} + U_t$ where U_t has ARMA behavior. We regress Y_t on X_{t-3} and the residuals suggest we can fit an MA(2) model for U_t.
Output:

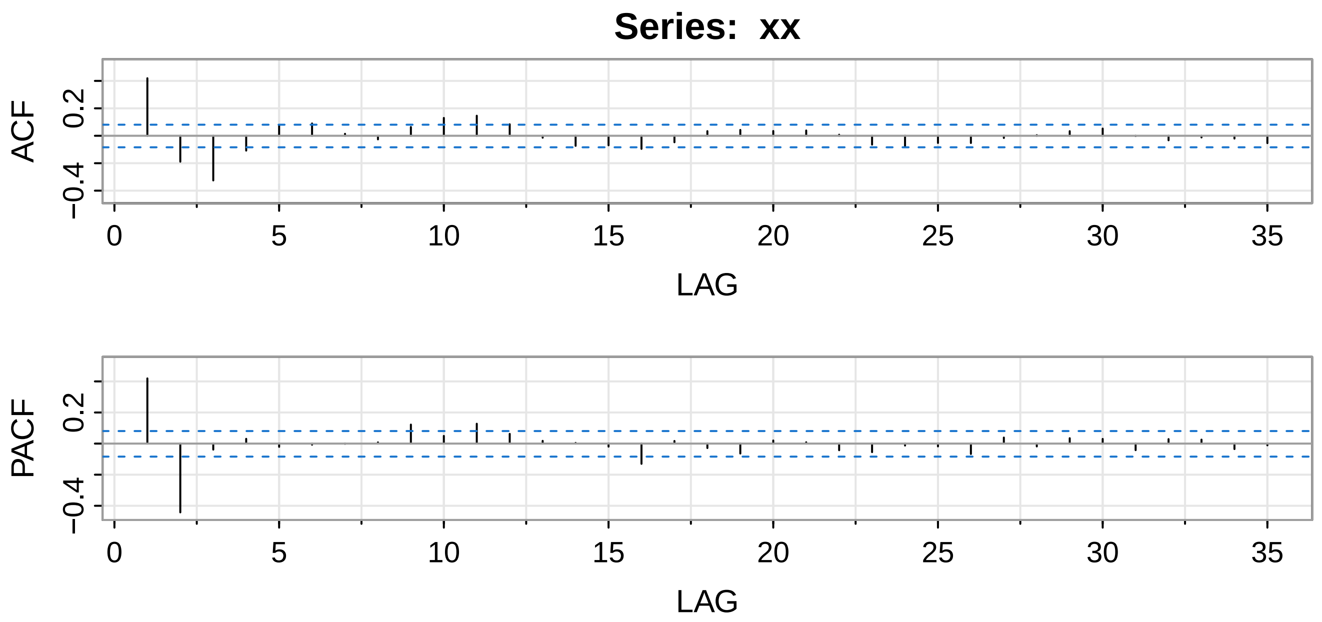
```r
yy = dat1[, 1]
xx <- dat1[, 2]
plot(xx)
```



```r
nn <- length(xx)
```

We check the diagnostic plots and get the coefficients.

```r
invisible(acf2(xx))
```



To fit the model for xx, we try $p = 0, 1, 2$ and $q = 0, 1, 2, 3$.

```
maxAR <- 2 + 1
maxMA <- 3 + 1
fits_all <- vector("list", length = maxAR)
AICmin <- BICmin <- Inf
for (ii in 1:maxAR) {
    fits_all[[ii]] <- vector("list", length = maxMA)
    for (jj in 1:maxMA) {
        fits_all[[ii]][[jj]] <- sarima(xx, p = ii - 1, d = 0, q = jj - 1, no.constant = FALSE,
            details = FALSE)
        if (fits_all[[ii]][[jj]]$AICc < AICmin)
            AICmin <- fits_all[[ii]][[jj]]$AICc
        if (fits_all[[ii]][[jj]]$BIC < BICmin)
            BICmin <- fits_all[[ii]][[jj]]$BIC
    }
}
print("check AICc's")

[1] "check AICc's"

for (ii in 1:maxAR) {
    for (jj in 1:maxMA) {
        print(paste0(ii - 1, "", jj - 1))
        print(((fits_all[[ii]][[jj]])$AICc - AICmin) * nn)
    }
}

[1] "00"
[1] 239.0987
[1] "01"
[1] 54.6937
[1] "02"
[1] 49.93804
[1] "03"
```

```
[1] 17.94168
[1] "10"
[1] 126.9123
[1] "11"
[1] 54.36552
[1] "12"
[1] 50.12545
[1] "13"
[1] 12.3458
[1] "20"
[1] 0
[1] "21"
[1] 1.193013
[1] "22"
[1] 2.695938
[1] "23"
[1] 4.402948
```

```r
print("check BICc's")
```

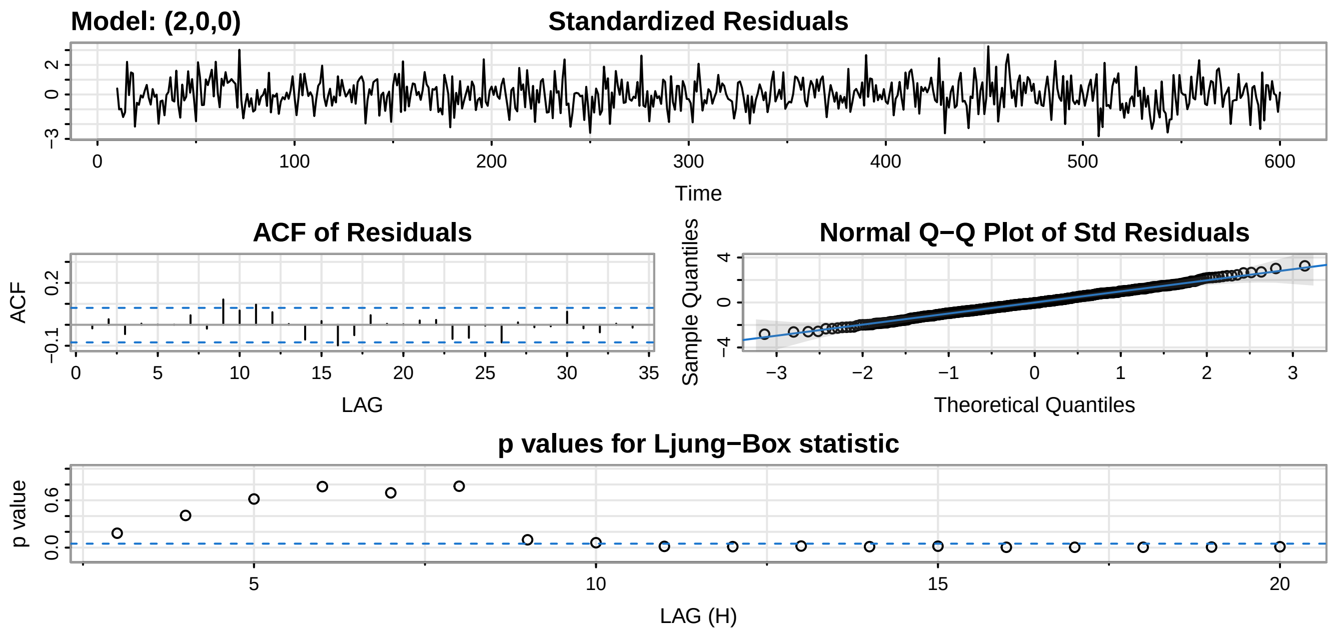
```
[1] "check BICc's"
```

```r
for (ii in 1:maxAR) {
    for (jj in 1:maxMA) {
        print(paste0(ii - 1, "", jj - 1))
        print(((fits_all[[ii]][[jj]])$BIC - BICmin) * nn)
    }
}
```

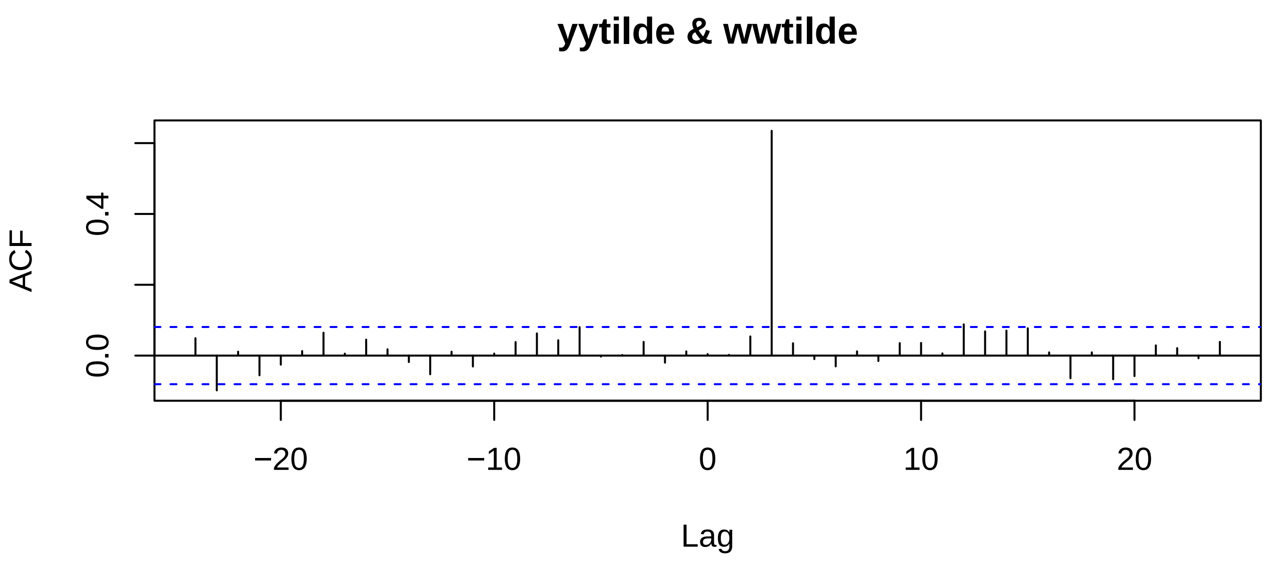
```
[1] "00"
[1] 230.3692
[1] "01"
[1] 50.33236
[1] "02"
[1] 49.93804
[1] "03"
[1] 22.29612
[1] "10"
[1] 122.551
[1] "11"
[1] 54.36552
[1] "12"
[1] 54.47989
[1] "13"
[1] 21.04775
[1] "20"
[1] 0
[1] "21"
[1] 5.547456
[1] "22"
[1] 11.39789
[1] "23"
[1] 17.44545
```

AICc hits (2,0) and (2,1).
BIC hits (2,0).
I choose (2,0) since it has the smallest AIC and the smallest BIC. The diagnostic plots are not worse for this model than for others.

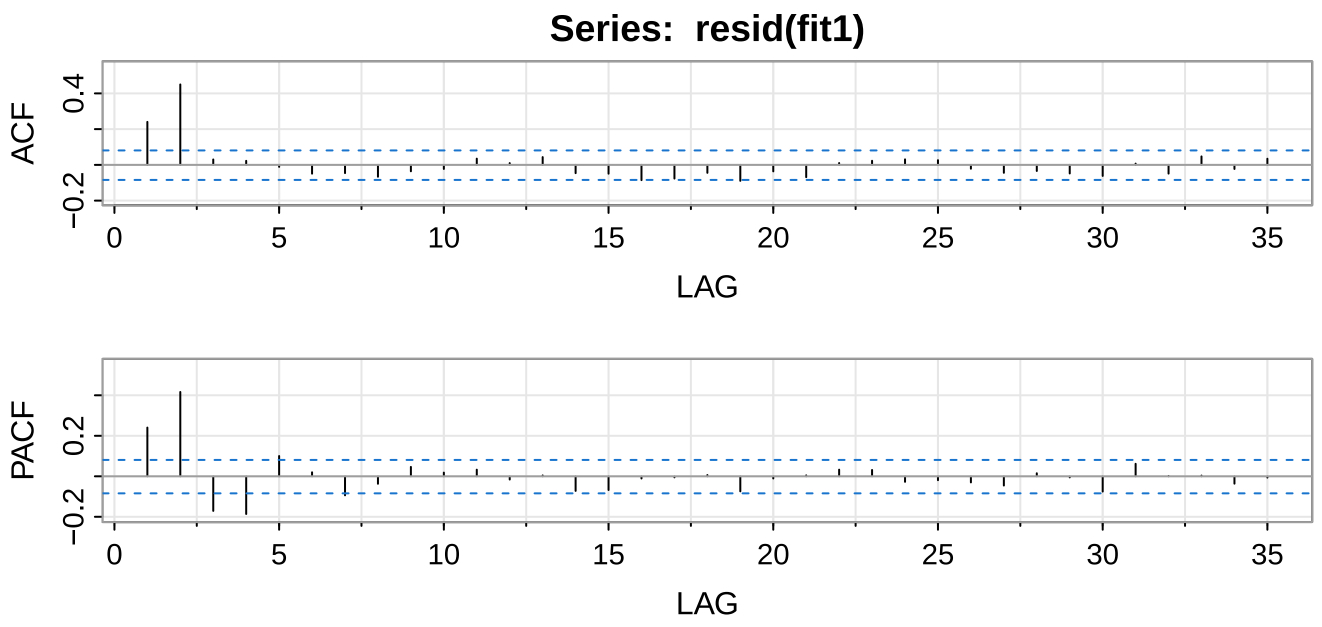
```r
invisible(capture.output(astsa::sarima(xx, p = 2, d = 0, q = 0, no.constant = FALSE)))
```



```r
xxfit <- astsa::sarima(xx, p = 2, d = 0, q = 0, no.constant = FALSE, details = F)

aa <- xxfit$fit$coef
int <- aa[3] * (1 - aa[1] - aa[2])

wwtilde <- resid(xxfit$fit)
yytilde <- filter(yy, filter = c(1, -aa[1:2]), sides = 1, method = "convolution")


ccf(yytilde, wwtilde, na.action = na.omit)
```



```
XL3 <- lag(xx, -3)
dat <- ts.intersect(yy, XL3)


fit1 <- lm(yy ~ XL3, data = dat)
invisible(acf2(resid(fit1)))
```



To fit the model for the errors, we try $p = 0, 1, 2, 3, 4$ and $q = 0, 1, 2$.

```
maxAR_e <- 4 + 1
maxMA_e <- 2 + 1
fits_all_e <- vector("list", length = maxAR)
AICmin_e <- BICmin_e <- Inf
for (ii in 1:maxAR_e) {
    fits_all_e[[ii]] <- vector("list", length = maxMA_e)
    for (jj in 1:maxMA_e) {
        fits_all_e[[ii]][[jj]] <- sarima(dat[, 1], p = ii - 1, d = 0, q = jj - 1,
            xreg = dat[, 2], details = F)
        if (fits_all_e[[ii]][[jj]]$AICc < AICmin_e)
            AICmin_e <- fits_all_e[[ii]][[jj]]$AICc
        if (fits_all_e[[ii]][[jj]]$BIC < BICmin_e)
            BICmin_e <- fits_all_e[[ii]][[jj]]$BIC
    }
}
print("check AICc's")
```

```
[1] "check AICc's"
```

```
for (ii in 1:maxAR_e) {
    for (jj in 1:maxMA_e) {
        print(paste0(ii - 1, "", jj - 1))
        print(((fits_all_e[[ii]][[jj]])$AICc - AICmin_e) * nn)
    }
}
```

```
[1] "00"
[1] 191.6673
[1] "01"
[1] 174.9844
[1] "02"
[1] 0
[1] "10"
[1] 158.2235
[1] "11"
[1] 123.3187
[1] "12"
[1] 1.722056
[1] "20"
[1] 47.63955
[1] "21"
[1] 41.21284
[1] "22"
[1] 2.538945
[1] "30"
[1] 32.54057
[1] "31"
[1] 25.26943
[1] "32"
[1] 4.590049
[1] "40"
[1] 13.55923
[1] "41"
[1] 11.90428
[1] "42"
[1] 6.549623
```

```
print("check BICc's")
```

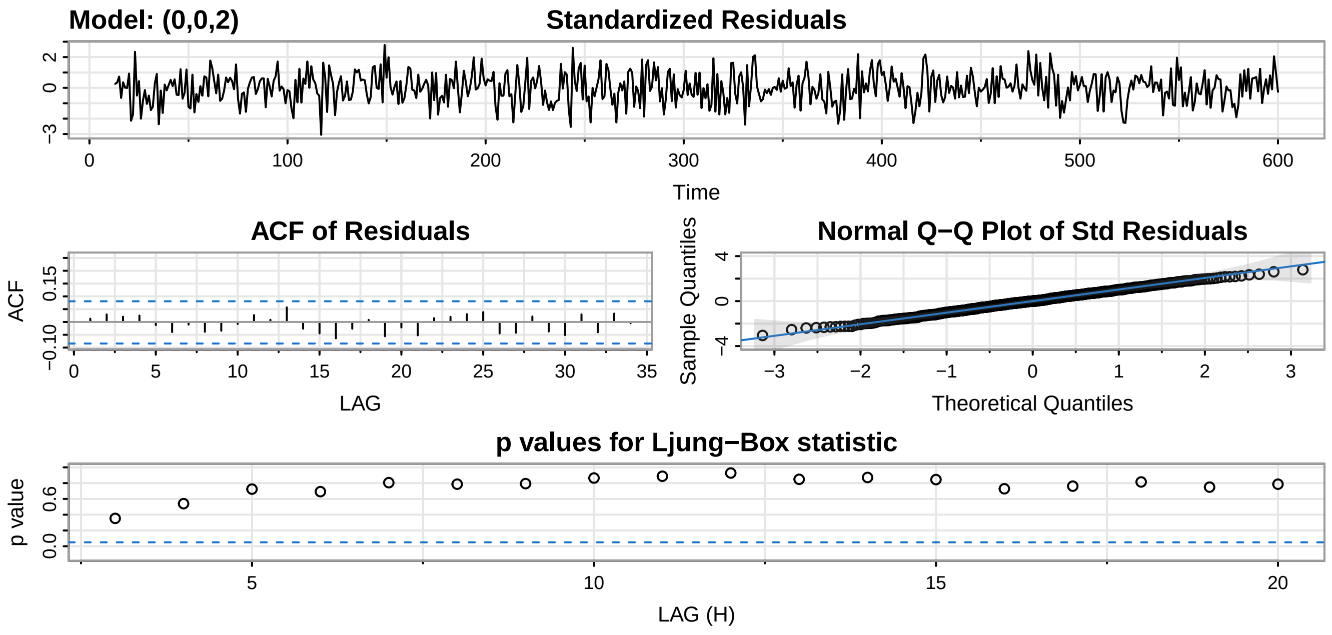
```
[1] "check BICc's"
```

```r
for (ii in 1:maxAR_e) {
    for (jj in 1:maxMA_e) {
        print(paste0(ii - 1, "", jj - 1))
        print(((fits_all_e[[ii]][[jj]])$BIC - BICmin_e) * nn)
    }
}
```

```
[1] "00"
[1] 182.9175
[1] "01"
[1] 170.613
[1] "02"
[1] 0
[1] "10"
[1] 153.8521
[1] "11"
[1] 123.3187
[1] "12"
[1] 6.086455
[1] "20"
[1] 47.63955
[1] "21"
[1] 45.57724
[1] "22"
[1] 11.2607
[1] "30"
[1] 36.90497
[1] "31"
[1] 33.99119
[1] "32"
[1] 17.66209
[1] "40"
[1] 22.28099
[1] "41"
[1] 24.97632
[1] "42"
[1] 23.96484
```

AICc hits (0,2) and (1,2).
BIC hits (0,2).
I choose (0,2) since it has the smallest AIC and the smallest BIC. The diagnostic plots support this model.

```r
invisible(capture.output(astsa::sarima(dat[, 1], p = 0, d = 0, q = 2, xreg = dat[,
    2])))
```



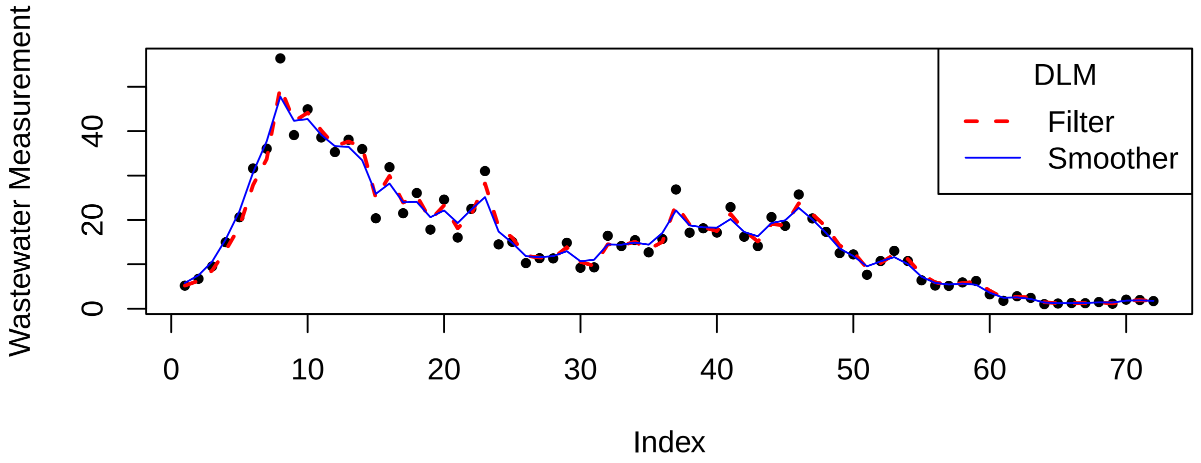
```
astsa::sarima(dat[, 1], p = 0, d = 0, q = 2, xreg = dat[, 2], details = F)$ttable

          Estimate     SE t.value p.value
ma1         0.1878 0.0347  5.4136  0.0000
ma2         0.5652 0.0329 17.1608  0.0000
intercept  -6.5233 5.9041 -1.1049  0.2697
xreg        1.2813 0.0293 43.6641  0.0000
```

Dat2

```r
library(dlm)



locallevel_2_1 <- function(x, m0) {
    Sigma.R <- diag(c(exp(x[1]), exp(x[2])))
    Sigma.Q <- matrix(exp(x[3]))
    m0 <- as.matrix(m0)
    C0 <- matrix(data = 0.1^2)
    GG <- matrix(data = 1)
    FF <- matrix(c(1, 1), ncol = 1)
    return(list(m0 = m0, C0 = C0, FF = FF, GG = GG, V = Sigma.R, W = Sigma.Q))
}

locallevel_2_1_v2 <- function(x, m0) {
    Sigma.R <- diag(c(exp(x[1]), exp(x[1])))
    Sigma.Q <- matrix(exp(x[2]))
    m0 <- m0
    C0 <- matrix(0.1^2)
    GG <- matrix(1)
    FF <- matrix(c(1, 1), ncol = 1)
    return(list(m0 = m0, C0 = C0, FF = FF, GG = GG, V = Sigma.R, W = Sigma.Q))
}


locallevel_1_1 <- function(x, m0) {
    Sigma.R <- matrix(exp(x[1]))
    Sigma.Q <- matrix(exp(x[2]))
    m0 <- m0
    C0 <- matrix(0.2^2)
    GG <- matrix(1)
    FF <- matrix(1)
    return(list(m0 = m0, C0 = C0, FF = FF, GG = GG, V = Sigma.R, W = Sigma.Q))
}
```

1.
```r
WW1 <- dat2$WW.weekly
dlm1 <- dlmMLE(y = WW1, parm = c(0, 0), control = list(maxit = 300), m0 = WW1[1],
    build = locallevel_1_1)
par1 <- dlm1$par
mod1 <- locallevel_1_1(x = par1, m0 = WW1[1])
filter1 <- dlmFilter(y = WW1, mod = mod1)
smoother1 <- dlmSmooth(filter1)
WW1.filter <- filter1$m[-1]
WW1.smoother <- smoother1$s[-1]

plot(WW1, cex = 0.75, pch = 16, ylab = "Wastewater Measurement")
lines(WW1.filter, col = "red", lty = "dashed", lwd = 2)
lines(WW1.smoother, col = "blue", lty = "solid", lwd = 1)
```

```r
legend(x = "topright", title = "DLM", lwd = c(2, 1), legend = c("Filter", "Smoother"),
    lty = c("dashed", "solid"), col = c("red", "blue"))
```



```r
print(par1)
```

```
[1] 2.285171 2.870191
```

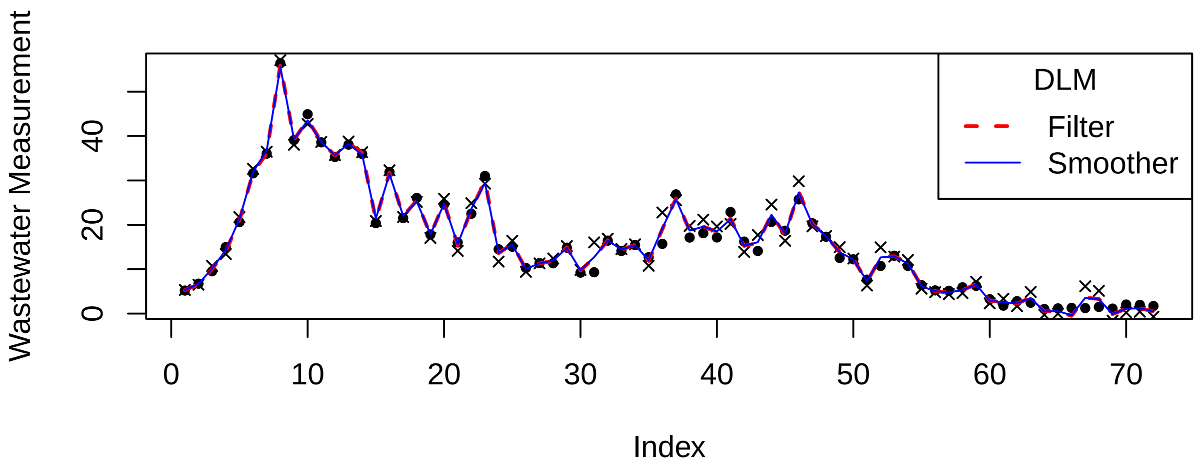
2. The filter (at time t) uses data only up to time t whereas the smoother uses all the data including data in the future of time t. This indeed makes the latter generally appear smoother (as the name suggests) because it is closer to interpolating the past and future data points.

   For instance, examining $t = 23$, the observed data point is larger than the neighboring ones; the filter jumps up to be close to that data point whereas the smoother jumps up less because it knows the series will drop down afterwards.

3.
```r
WW <- as.matrix(dat2[, c(3:4)])

dlm2 <- dlmMLE(y = WW, parm = c(0, 0), control = list(maxit = 300), m0 = mean(WW[1,
    ]), build = locallevel_2_1_v2)

par2 <- dlm2$par
mod2 <- locallevel_2_1_v2(x = par2, m0 = mean(WW[1, ]))
filter2 <- dlmFilter(y = WW, mod = mod2)
smoother2 <- dlmSmooth(filter2)
WW.filter <- filter2$m[-1]
WW.smoother <- smoother2$s[-1]

plot(WW[, 1], cex = 0.75, pch = 16, ylab = "Wastewater Measurement")
points(WW[, 2], pch = 4, cex = 0.75)
lines(WW.filter, col = "red", lty = "dashed", lwd = 2)
lines(WW.smoother, col = "blue", lty = "solid", lwd = 1)
print("The estimated parameters are:")
```

```
[1] "The estimated parameters are:"
```

```
print(par2)

[1] 0.8874717 3.6011137

legend(x = "topright", title = "DLM", lwd = c(2, 1), legend = c("Filter", "Smoother"),
    lty = c("dashed", "solid"), col = c("red", "blue"))
```

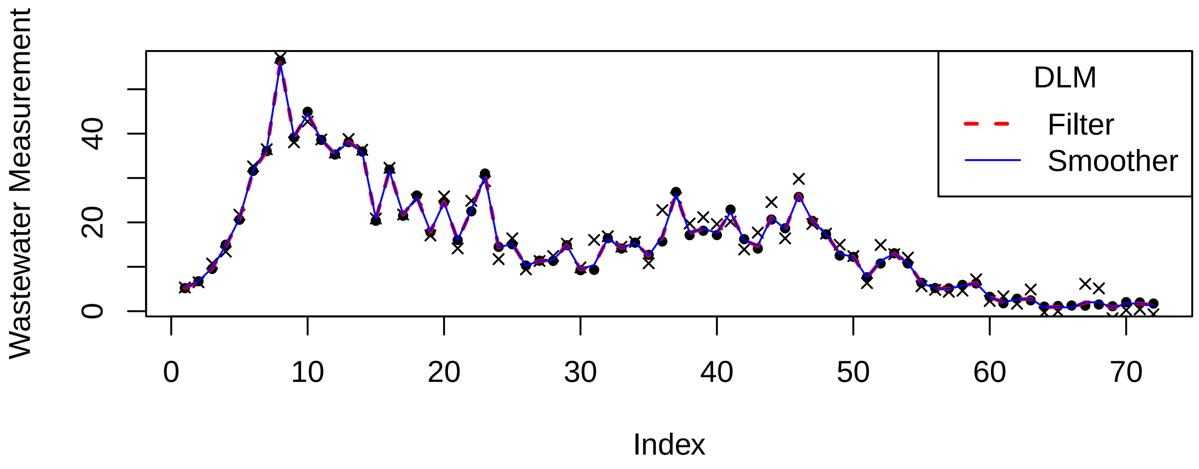


4. The filter and smoother in the bivariate DLM are much closer to each other than they were in the univariate DLM model. They almost overlap in this plot. The reason is that the observation noise is estimated as being smaller ($e^{.88} = 2.4$ versus $e^{2.29} = 9.9$). The reason for that is that at each time point we have two rather than one observation, and (in this dataset) those two observations are relatively close together which gives more information that the observation noise is small. When the observation noise is relatively small, the model will come close to interpolating the data for either the filter or the smoother since neither data ahead or behind time t are having a major effect on the expected latent state value (which is just assumed to be close to the observed value).

   Version 2 Below is a second version of the output/answer, using the local level model version with two different observation variance parameters. The question details did not specify what to do so this can receive full credit.

```
WW <- as.matrix(dat2[, c(3:4)])

dlm2 <- dlmMLE(y = WW, parm = c(0, 0, 0), control = list(maxit = 300), m0 = mean(WW[1,
    ]), build = locallevel_2_1)

par2 <- dlm2$par
mod2 <- locallevel_2_1(x = par2, m0 = mean(WW[1, ]))
filter2 <- dlmFilter(y = WW, mod = mod2)
smoother2 <- dlmSmooth(filter2)
WW.filter <- filter2$m[-1]
WW.smoother <- smoother2$s[-1]

plot(WW[, 1], cex = 0.75, pch = 16, ylab = "Wastewater Measurement")
```

```
points(WW[, 2], pch = 4, cex = 0.75)
lines(WW.filter, col = "red", lty = "dashed", lwd = 2)
lines(WW.smoother, col = "blue", lty = "solid", lwd = 1)
print("The estimated parameters are:")

[1] "The estimated parameters are:"

print(par2)

[1] -0.3040183  1.4022864  3.5995424

legend(x = "topright", title = "DLM", lwd = c(2, 1), legend = c("Filter", "Smoother"),
    lty = c("dashed", "solid"), col = c("red", "blue"))
```



The filter and smoother almost captures the third column of the data (the solid dots) because the corresponding estimated is very small: $exp(-0.304) \approx 0.738$ while both deviate from the observation in the first model.