Homework 3
MATH/STAT 4540/5540 Spr 2022 Time Series

Due date: Friday, March 4, before midnight, on Canvas/Gradescope **Instructor**: Prof. Becker
Theme: Seasonality estimation, and forecasting ARMA processes.

Instructions Collaboration with your fellow students is OK and in fact recommended, although direct copying is not allowed. The internet is allowed for basic tasks (e.g., looking up definitions on wikipedia) but it is not permissible to search for proofs or to *post* requests for help on forums such as `http://math.stackexchange.com/` or to look at solution manuals. Please write down the names of the students that you worked with.

An arbitrary subset of these questions will be graded.

Reading You are responsible for chapters 2.4–2.5 and 3.2–3.3 in our textbook (Brockwell and Davis, "Intro to Time Series and Forecasting", 3rd ed).

Homework Formatting As this is mid-semester in a 4/5000 level class, we now expect homework to be nicely formatted. Code should *not* be screen-shots; PDFs should be letter paper sized. It *is* acceptable to work out theory problems by hand (like Problem 2) and then take a photo or scan your work. However, all computer generated output (figures, code, etc.) should be professional looking. Ask the instructor or TA if you are unsure if your output is well-formatted.

Problem 1: We covered several types of trend/seasonality removal in class. For a time series with daily data for 3 years, we don't have many observations of the full period (just three), so it's often reasonable to assume a parametric form for the seasonality component (which is what we did when we modeled it as a mixture of sines and cosines).

However, if you have many periods worth of observations, and the periods themselves are not too fine-grained (e.g., for annual seasonality, this might mean monthly data rather than daily data), a very nice alternative is to model the seasonality via linear regression in dummy variables (aka indicator variables), as well as with any other terms for the trend that you want to include.

To see how this works, suppose you have daily data that you suspect shows a weekly trend. Then the regression model is

$$X_t = \beta_0 + \beta_1 t + \beta_{\text{Mon}} \mathbb{1}_{\text{Mon}}(t) + \beta_{\text{Tue}} \mathbb{1}_{\text{Tue}}(t) + \ldots + \beta_{\text{Sun}} \mathbb{1}_{\text{Sun}}(t) + \epsilon(t)$$

where

$$\mathbb{1}_{\text{Mon}}(t) = \begin{cases} 1 & t \text{ is a Monday} \\ 0 & \text{else} \end{cases}$$

and similarly for the other days of the week (and $\epsilon(t)$ is noise that our model cannot explain)[1]. You use linear regression to solve for the coefficients $\beta_0, \beta_1, \beta_{\text{Mon}}, \ldots, \beta_{\text{Sun}}$.

For this problem, use the `CO-GHCND-TN-TX.RData` dataset like we did in the TrendRemoval.ipynb demo. Your dataset should be the minimum daily temperature in Fort Collins from 1900 to 2010, i.e., this is part ofo column 125 of the `TN` variable. Then, *unlike* the demo (but similar to your last homework), **aggregate** the data by month. The dataset has variables `da`, `mo` and `yr` which are the day, month and year (of each observation) which can help you do the aggregation (tools from the `lubridate` package can also be helpful).

[1] You actually only need to include 6 days of the week, since the other date can be inferred, but either way is OK since `lm` will automatically remove anything redundant.

a) Perform a trigonometric regression (as well as a linear and offset term), i.e., using the `lm` package, regress the temperature X_t onto the model $\beta_0 + \beta_1 t + \beta_2 \cos(2\pi t/12) + \beta_3 \sin(2\pi t/12) + \epsilon(t)$. Show your code and report the value of the coefficient for the linear term in the regression (using correct units).

b) Now perform regression using the dummy variables as suggested above. As before, show your code and the value of the linear term.

c) For the range 1900 to 1901, plot both seasonality estimates. Show the figure.

d) Comment on the differences between the two methods: do they both give similar estimate for the linear trend? Do the residuals behave differently? Which one do you think does a better job (there's no right answer; you get credit for your explanation). Include a plot of the ACF for the residuals of both models (showing lags up to 2 years).

Problem 2: Let (X_t) be a causal ARMA(1,1) time series with ARMA parameters ϕ and θ, and noise $Z_t \sim \mathrm{WN}(0, \sigma^2)$. Show that

$$\gamma_X(0) = \sigma^2 \left(1 + \frac{(\phi+\theta)^2}{1-\phi^2}\right)$$

Hint: rewrite X_t as a MA(∞) process.

Problem 3: One-step predictions of an ARMA(1,1) process

a) On the computer, simulate 11 steps of an ARMA(1,1) process with parameters $\phi = 0.9$ and $\theta = 0.7$. Using the first $n = 10$ data points from the simulation, create the one-step prediction $P_n X_{n+1}$ to predict X_{11}. Include your code, as well as a plot showing all 11 data points and your prediction.

Tips in R: use `arima.sim` to simulate the ARMA process; use `ARMAacf` to calculate the theoretical ACF ρ for the ARMA process; use `toeplitz` to simplify construction of the Γ_n matrix; use `solve` to invert the matrix (see note later on other ways to do this).

b) Repeat the above simulation 10,000 times (every time, keep all details the same, but using new simulations/noise), and report a histogram of the error $X_{11} - P_n X_{n+1}$. Is your estimate unbiased (and did you expect it to be)? What is the mean and variance of the error $X_{11} - P_n X_{n+1}$?

c) Compare the variance of the error you found via simulation with the expected variance of the error. *Hint:* use equation (2.5.9) and the result from Problem 1 to calculate the expected variance of the error.

Note about inverting Γ_n. You can solve an equation like $\Gamma_n \boldsymbol{a} = \boldsymbol{\gamma}$ in R using $\boldsymbol{a} = \texttt{solve}(\Gamma_n, \boldsymbol{\gamma})$ or (less preferred) $\Gamma_n^{-1} \leftarrow \texttt{solve}(\Gamma)$ followed by $\boldsymbol{a} = \Gamma_n^{-1}\boldsymbol{\gamma}$. These are easy to program and you can use either method on your homework.

However, if n is very large, or Γ_n is ill-conditioned, then you may want to use better methods [*not* necessary to do for this homework]. One method is to exploit the fact that Γ_n is symmetric nonnegative definite (aka postive semidefinite), and in almost all cases, it is in fact symmetric positive definite, hence a Cholesky decomposition is an accurate method to use. Similar to a LU decomposition used in Gaussian elimination, it factors the matrix as a product of upper and lower triangular matrices, and then you use it for forward and backward substitution. For example, to solve $\Gamma \boldsymbol{a} = \boldsymbol{y}$ to find $\boldsymbol{a}$, in R you can do the following:

```
GC <- chol(Gamma) # upper triangular
a  <- backsolve( GC, forwardsolve( GC,y,transpose=TRUE,upper.tri=TRUE ) )
```

When n is extremely large, the above methods are slow because they take $\mathcal{O}(n^3)$ time (where Γ_n is a $n \times n$ matrix). That's the point of the Durbin-Levinson algorithm, which is a specific case of the Levinson recursion algorithm used to "invert" any Toeplitz matrix (in Python, this is `scipy.linalg.solve_toeplitz`). It runs in $\mathcal{O}(n^2)$ time (and $\mathcal{O}(n)$ memory), though it's not particularly stable. Other algorithms like the Bareiss algorithm also run in $\mathcal{O}(n^2)$ time and are more stable (though require $\mathcal{O}(n^2)$ memory).