Name: Your name here
Due: 2024/11/04

Homework 6

Be sure to submit **both** the .pdf and .qmd file to Canvas by Monday, November 4th at 11:59 pm. The purpose of this assignment is to use simulation to better understand why we need to account for serial correlation in a time series.

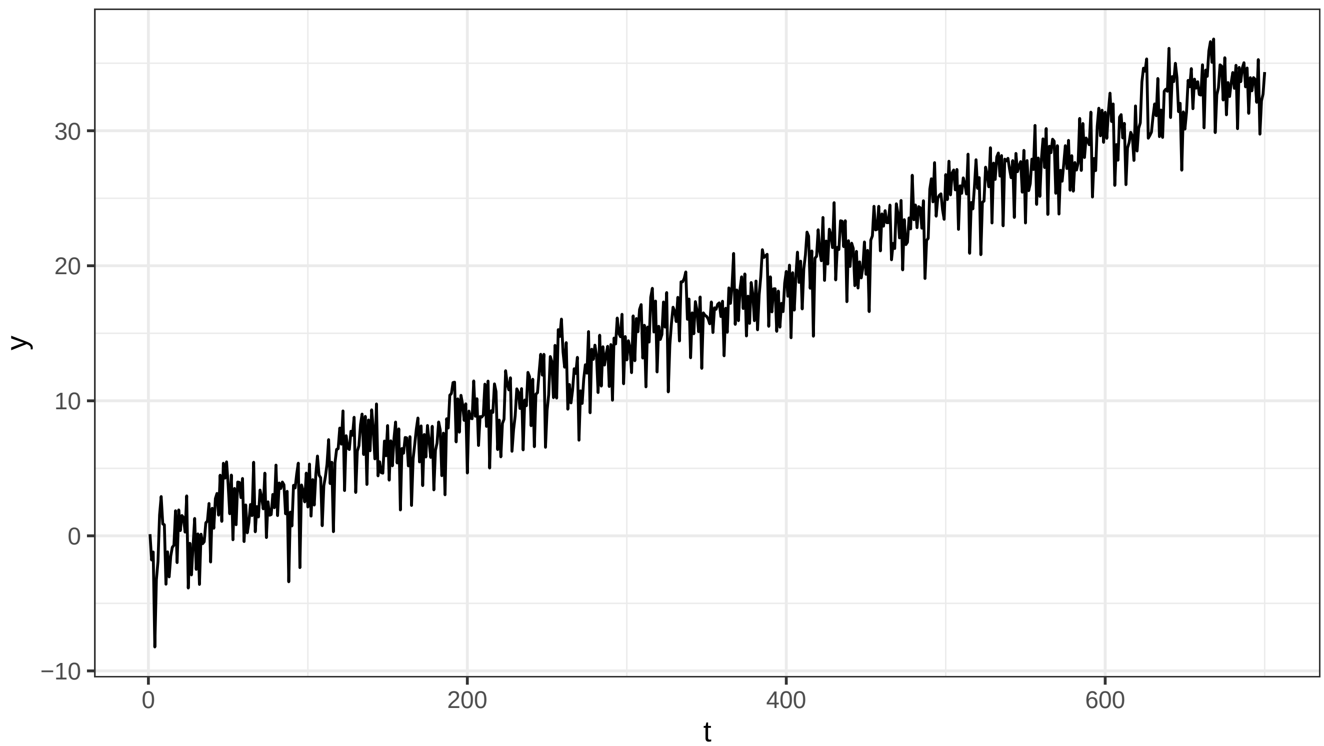
For this assignment, you may use the `generate_ts_reg` function contained in the `helpers.R` script to generate time series with trend, seasonality, and autocorrelated errors from an AR(1) process.

```r
source("helpers.R")

# example: 50 weeks of data
ex_ts <- generate_ts_reg(
  1027204,
  n = 100*7,
  freq = 7,
  betas = c(-1, .05, rnorm(6, sd = .5)) # beta0, beta1, and 6 harmonic cycles
)
str(ex_ts)
```

```
List of 4
 $ df    : tibble [700 x 8] (S3: tbl_df/tbl/data.frame)
  ..$ t    : int [1:700] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ sin1t: num [1:700] 0.782 0.975 0.434 -0.434 -0.975 ...
  ..$ sin2t: num [1:700] 0.975 -0.434 -0.782 0.782 0.434 ...
  ..$ sin3t: num [1:700] 0.434 -0.782 0.975 -0.975 0.782 ...
  ..$ cos1t: num [1:700] 0.623 -0.223 -0.901 -0.901 -0.223 ...
  ..$ cos2t: num [1:700] -0.223 -0.901 0.623 0.623 -0.901 ...
  ..$ cos3t: num [1:700] -0.901 0.623 -0.223 -0.223 0.623 ...
  ..$ y    : num [1:700] 0.124 -1.789 -1.192 -8.23 -3.253 ...
 $ X     : num [1:700, 1:8] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:700] "1" "2" "3" "4" ...
  .. ..$ : chr [1:8] "(Intercept)" "t" "sin1t" "sin2t" ...
  ..- attr(*, "assign")= int [1:8] 0 1 2 3 4 5 6 7
 $ mean  : num [1:700] 0.29 -1.369 0.892 -3.783 -0.615 ...
 $ params:List of 3
  ..$ betas: num [1:8] -1 0.05 0.838 -0.456 1.684 ...
  ..$ sigma: num 1
  ..$ alpha: num 0.662
```

```r
ex_ts$df %>%
  ggplot() +
  geom_line(aes(x = t, y = y)) +
  theme_bw()
```




```r
lm(y ~ ., ex_ts$df)
```

```
Call:
lm(formula = y ~ ., data = ex_ts$df)

Coefficients:
(Intercept)            t        sin1t        sin2t        sin3t        cos1t  
   -1.39613      0.05156      0.82175     -0.48562      1.72099      0.72309  
      cos2t        cos3t  
    0.04943      0.16477  
```

```r
ex_ts$params$betas
```

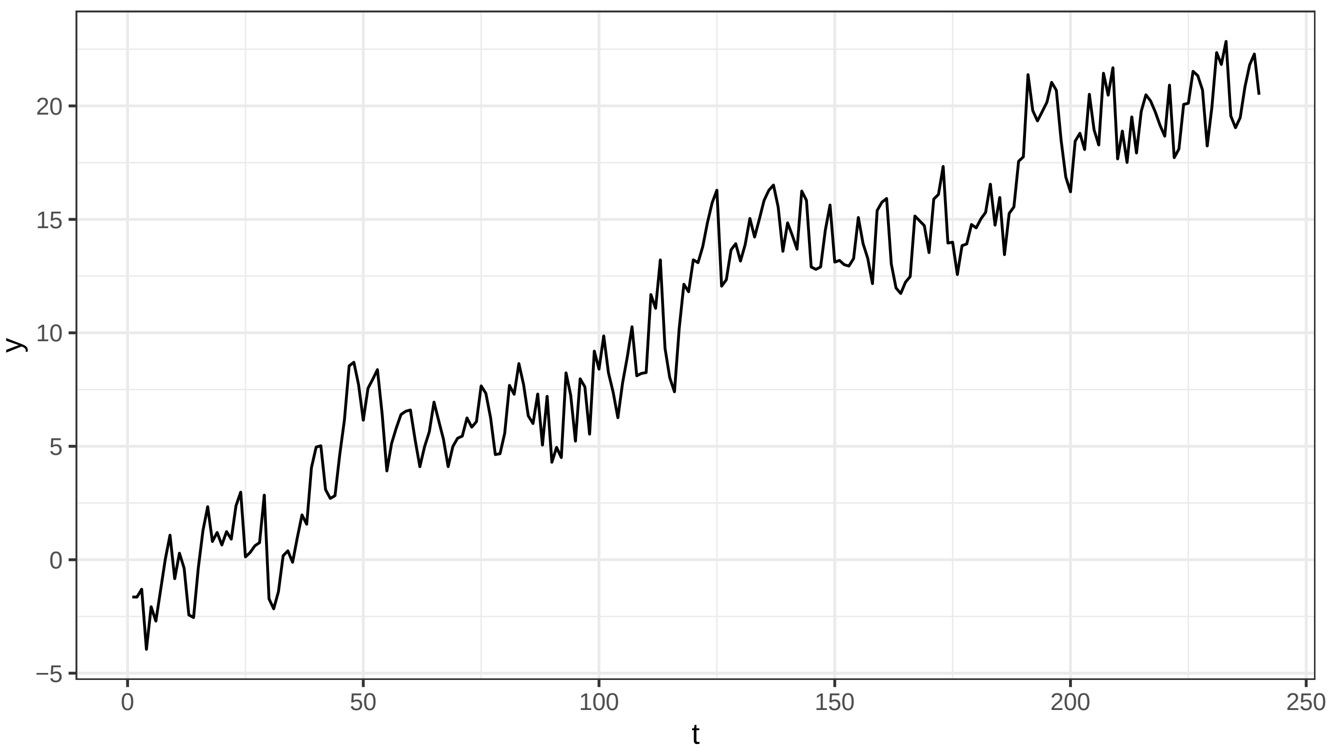
```
[1] -1.0000000  0.0500000  0.8379243 -0.4563659  1.6843704  0.7266140  0.1059672
[8]  0.1445958
```

Question 1 [11 pt] The goal of this assignment is to conduct a *simulation study* that demonstrates why we need to account for serial autocorrelation when fitting regression models. The purpose of this first question is to get our feet wet with the idea of a simulation study.

1. [2 pt] Use the `generate_ts_reg` function to generate a time series that represents 20 years of monthly data. Set the `beta` vector equal to `c(-1, 0.05, 1, -1, rnorm(10, sd = .5))` and the autocorrelation parameter to `.8`. Plot the resulting time series.

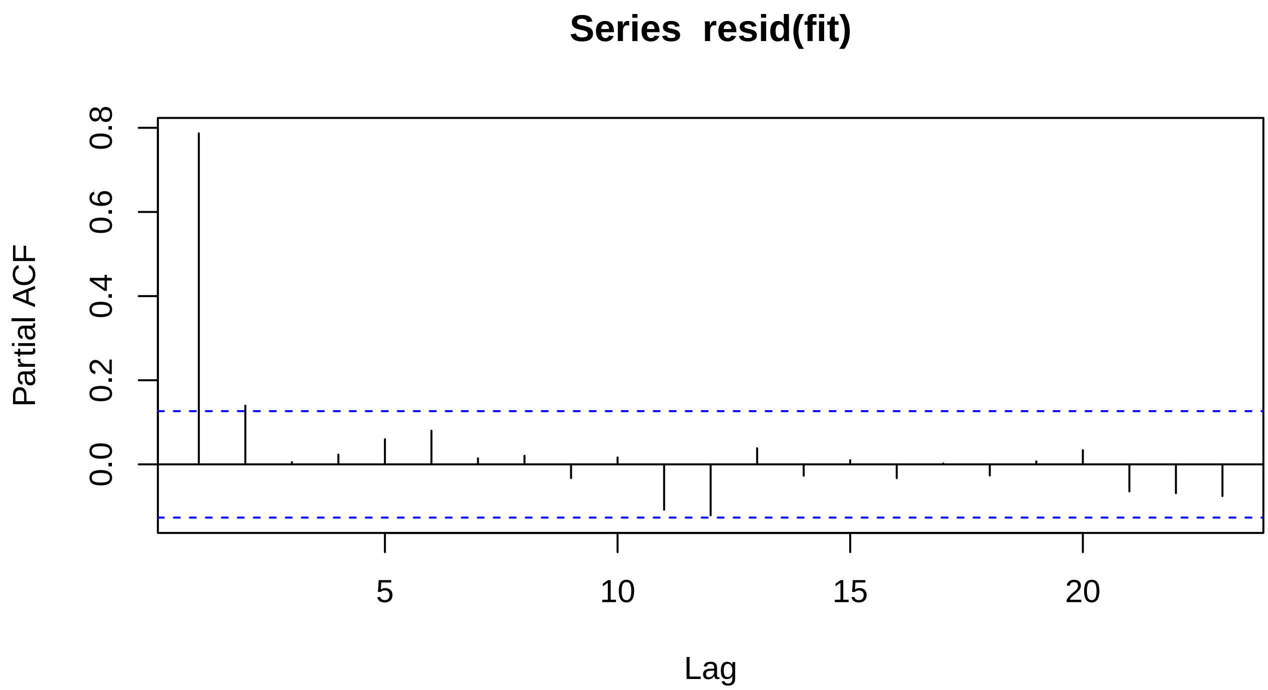
```r
ex_ts <- generate_ts_reg(
  1027204,
  n = 20*12,
  freq = 12,
  betas = c(-1, 0.1, 1, -1, rnorm(10, sd = .25)),
  alpha = 0.8
)

ex_ts$df %>%
  ggplot() +
  geom_line(aes(x = t, y = y)) +
  theme_bw()
```



2. [2 pt] Fit a linear regression model that includes the time index and all 12 harmonic seasonal cycles. Create a PACF plot of the residuals and comment on how the ACF plot relates to the way you generated the series.

```r
fit <- lm(y ~ ., data = ex_ts$df)
pacf(resid(fit))
```



> The PACF plot suggests that the residuals are an AR(1) series with $\alpha = .8$, which makes sense because that is exactly what we simulated.

3. [2 pt] Create confidence intervals for the regression coefficients using `confint` and determine which intervals captured the generating parameters (which you can find contained within the output of the `generate_ts_reg` function).

```r
ints <- confint(fit)
ex_ts$params$betas > ints[,1] & ex_ts$params$betas < ints[,2]
```

```
(Intercept)           t       sin1t       sin2t       sin3t       sin4t 
       TRUE       FALSE       FALSE        TRUE        TRUE        TRUE 
      sin5t       sin6t       cos1t       cos2t       cos3t       cos4t 
       TRUE        TRUE       FALSE        TRUE        TRUE        TRUE 
      cos5t       cos6t 
       TRUE        TRUE 
```

4. [4 pt] Fit a GLS model with an AR(1) correlation structure, create confidence intervals for the regression coefficients, and again determine which intervals captured the generating values.

```
library(nlme)
gls_fit <- gls(y ~ ., correlation = corARMA(p = 1), data = ex_ts$df)
gls_ints <- confint(gls_fit)
ex_ts$params$betas > gls_ints[,1] & ex_ts$params$betas < gls_ints[,2]
```

```
(Intercept)           t       sin1t       sin2t       sin3t       sin4t
       TRUE        TRUE       FALSE        TRUE        TRUE        TRUE
      sin5t       sin6t       cos1t       cos2t       cos3t       cos4t
       TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
      cos5t       cos6t
       TRUE        TRUE
```

5. [1 pt] If we were to repeatedly simulate time series like in question 1-1 through 1-4 and calculate 95% confidence intervals for the regression coefficients in the model, approximately what percent of the constructed intervals should capture the generating values, if we are appropriately modeling the uncertainty?

> About 95% - that is the definition of confidence!

Question 2 [**10 pt**] We are now ready to conduct our simulation study. To do so, repeat the following process 100 times:

- simulate a new data with a distinct seed (using the same `betas` and `alpha` as before)
- fit an OLS model and determine for which parameters the confidence interval captures the generating values
- fit a GLS model and determine for which parameters the confidence interval captures the generating values

Then, calculate the proportion of times (out of 100 simulations) that each model captured the generating values for each of the regression coefficients. Create a visual that displays the resulting proportions and comment on what the results suggest about the importance of accounting serial autocorrelation when estimating regression coefficients.

```
# simulation study
ols_capture <- matrix(NA, 100, 14)
gls_capture <- matrix(NA, 100, 14)
pb <- txtProgressBar(min = 0, max = 100, style = 3, width = 50, char = "=")
for(i in 1:100){
  # data first
  dat <- generate_ts_reg(
    seed = i,
    n = 20*12,
    freq = 12,
    betas = c(-1, 0.1, 1, -1, rnorm(10, sd = .5)),
    alpha = 0.8
  )

  # ols fit
  ols_fit <- lm(y ~ ., dat$df)
  ols_ints <- confint(ols_fit)
  ols_capture[i,] <- dat$params$betas > ols_ints[,1] & dat$params$betas < ols_ints[,2]

  # gls fit
  gls_fit <- gls(y ~ ., correlation = corARMA(p = 1), dat$df)
  gls_ints <- confint(gls_fit)
  gls_capture[i,] <- dat$params$betas > gls_ints[,1] & dat$params$betas < gls_ints[,2]

  # progress
  setTxtProgressBar(pb, i)
}
close(pb)
save(ols_capture, gls_capture, file = "sim.rdata")
```

```
# analyze
load("sim.rdata")
colMeans(ols_capture)
```

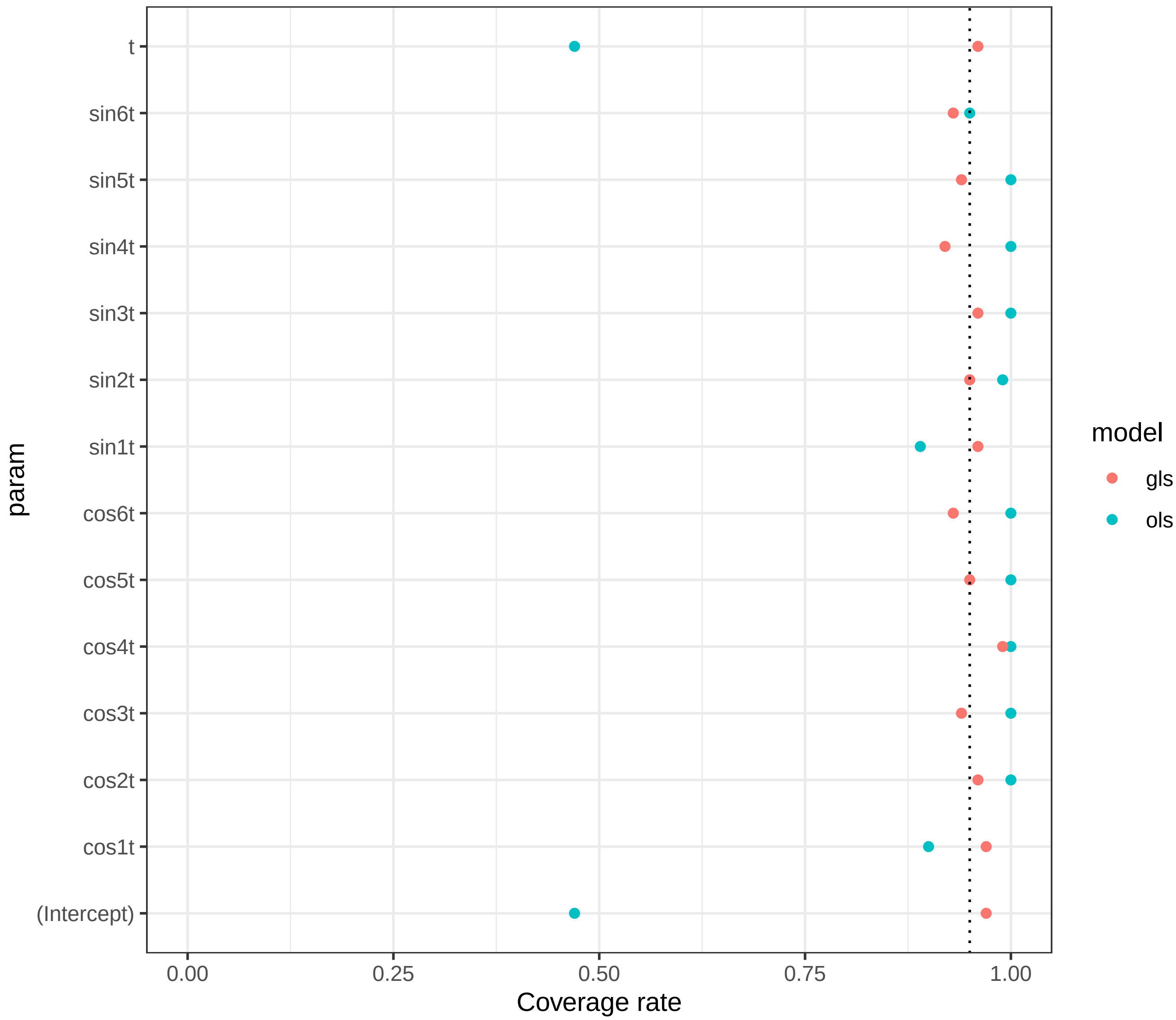
```
[1] 0.47 0.47 0.89 0.99 1.00 1.00 1.00 0.95 0.90 1.00 1.00 1.00 1.00 1.00
```

```
colMeans(gls_capture)
```

```
[1] 0.97 0.96 0.96 0.95 0.96 0.92 0.94 0.93 0.97 0.96 0.94 0.99 0.95 0.93
```

```
# grab some names
tmp <- generate_ts_reg(
    seed = 1,
    n = 20*12,
    freq = 12,
    betas = c(-1, 0.1, 1, -1, rnorm(10, sd = .5)),
    alpha = 0.8
  )

tibble(
  param = colnames(tmp$X),
  ols = colMeans(ols_capture),
  gls = colMeans(gls_capture)
) %>%
  pivot_longer(ols:gls, names_to = "model", values_to= "val") %>%
  ggplot() +
  geom_point(aes(y = param, x = val, col = model)) +
  theme_bw() +
  geom_vline(aes(xintercept = 0.95), linetype = "dotted") +
  lims(x = c(0, 1)) +
  labs(x = "Coverage rate")
```



Only the GLS estimates tend to achieve the nominal coverage rate (0.95) for all parameters in the model. The OLS fit has really poort coverage for the intercept and coefficient associated with time, meaning that we are not achieving nominal coverage for those parameters! That is bad news, since the regression coefficient associated with time is often of interest in these types of models.