Day 11 - Basic stochastic models lab

Introduction

The purpose of today's lab is to explore basic stochastic models using simulation. Since the exam is currently out, this lab will be graded on completion.

Warm-up

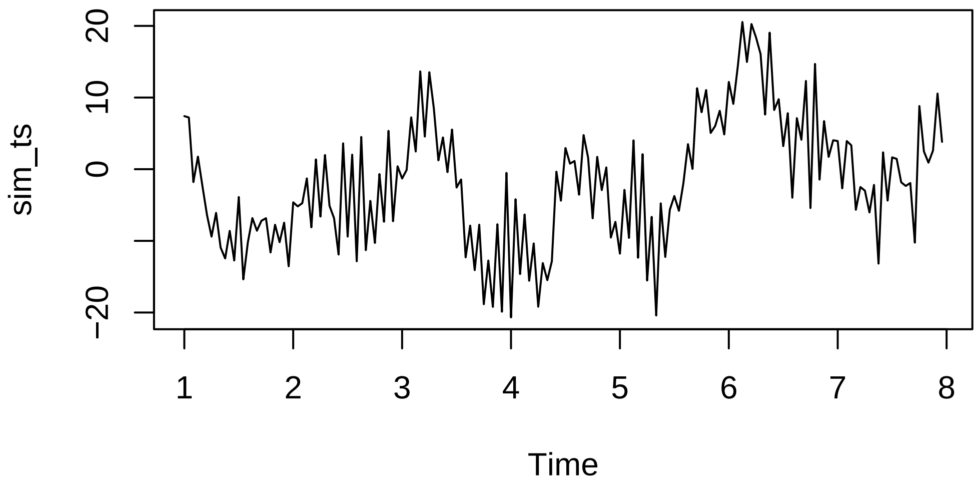
Consider the AR(2) model $x_t = .2x_{t-1} + .7x_{t-2} - .05x_{t-3} + w_t$ where w_t is Gaussian noise with a variance of 25. Express this model in terms of the backshift operator, and use that expression to determine whether the model is stationary.

```
polyroot(c(1, -.2, -.7, .05))
```

```
[1]  1.099711+0i -1.282317+0i 14.182606+0i
```

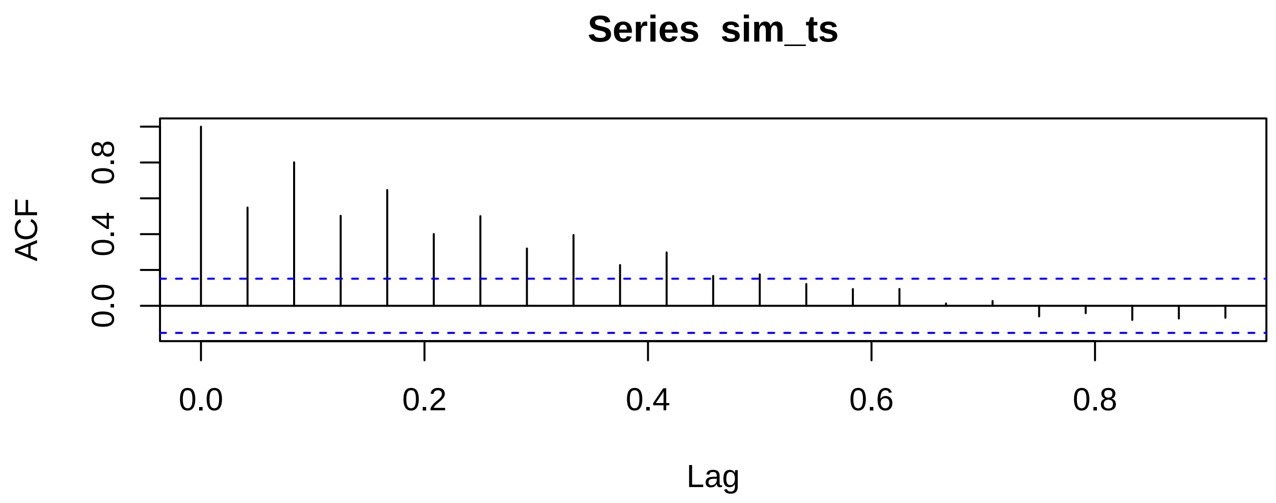
1. Simulate seven days of hourly data from the AR(2) model described in the warm-up, create a `ts` object called `sim_ts` from that series, and plot the series.

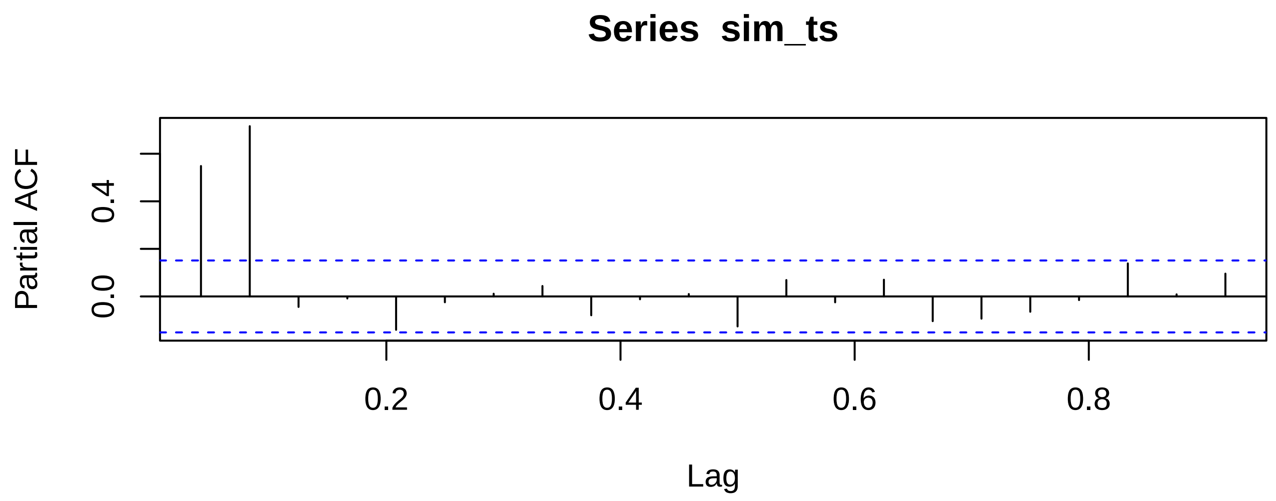
```r
# generate series
x <- w <- rnorm(24*7, sd = 5)
for(t in 4:(24*7)) {
  x[t] <- .2*x[t-1] + .7*x[t-2]  - .05*x[t-3] + w[t]
}

# create ts
sim_ts <- ts(
  x,
  start = c(1,1),
  freq = 24
)
plot(sim_ts)
```



2. Create acf and pacf plots of `sim_ts` and comment on what the plots suggest about serial autocorrelation in the series.

```r
par(mfrow = c(2, 1))
acf(sim_ts)
pacf(sim_ts)
```





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

The acf plot is interesting, and is what typically results from a series that contains a relatively strong positive autocorrelation at lags one and two, with lag two being stronger.

The pacf plot makes this abundantly clear, with spikes at lag one and two, with two much greater than one.

3. Fit an AR(2) model to the observed series, called `ar_fit`, and write out the estimated model.

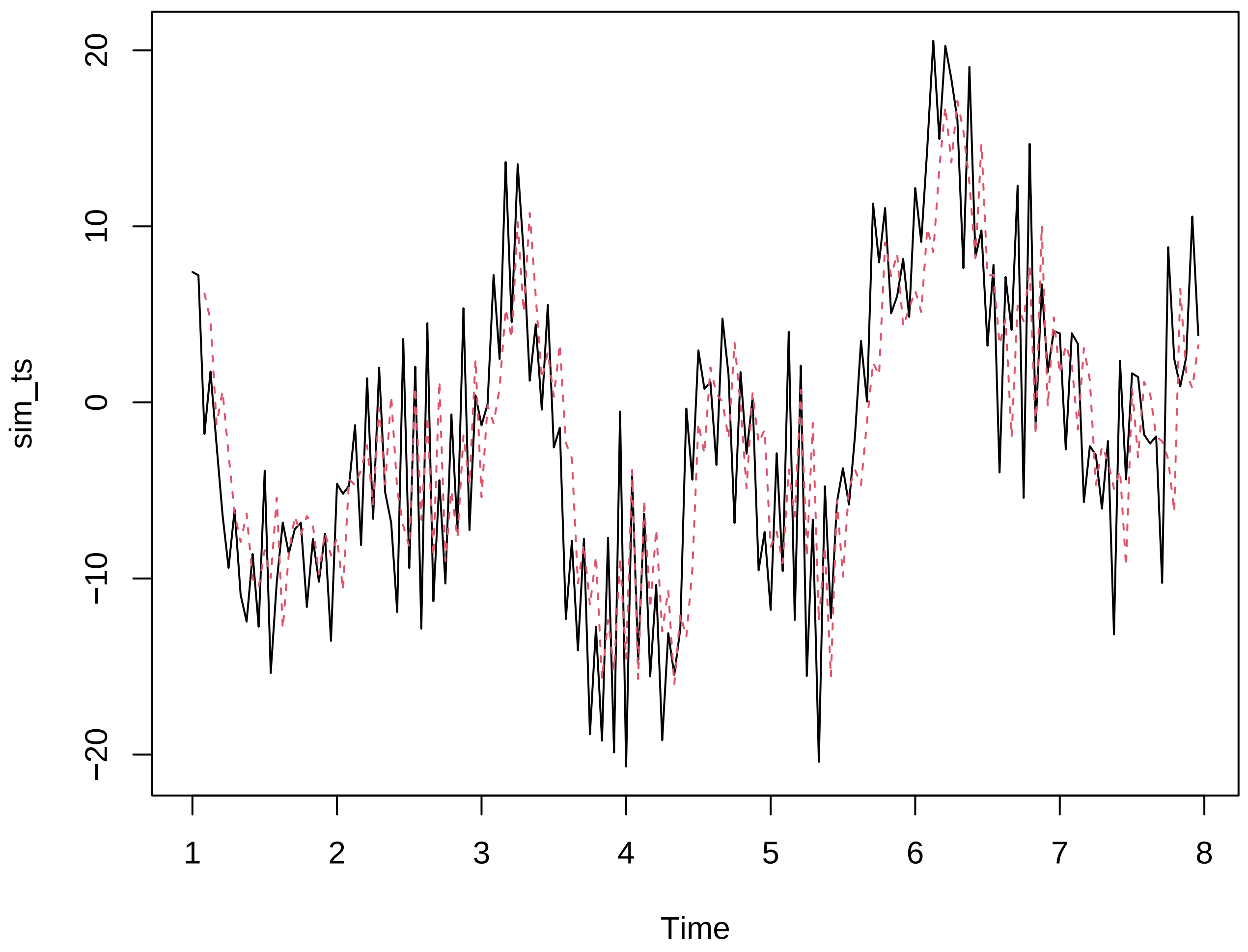
```r
ar_fit <- ar(sim_ts, order.max = 2)
ar_fit
```

```

Call:
ar(x = sim_ts, order.max = 2)

Coefficients:
     1       2  
0.1558  0.7159  

Order selected 2  sigma^2 estimated as  26.56
```

$$\hat{x}_t = -1.91 + .156(x_{t-1} + 1.91) + .716(x_{t-2} + 1.91)$$

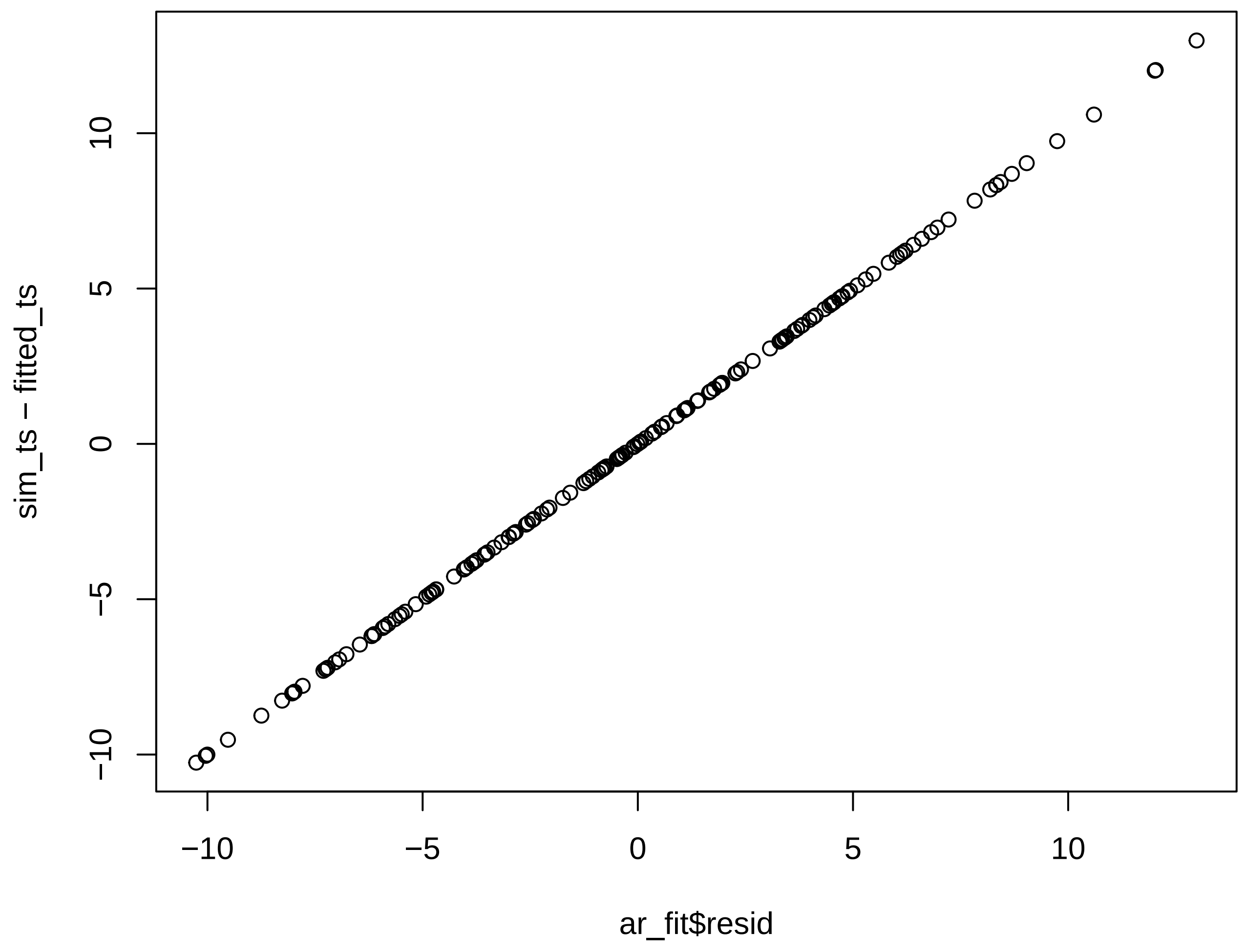
4. Calculate the fitted AR(2) model described in question three, and plot the observed and fitted series on the same plot.

```
fitted <- rep(NA, length(sim_ts))
for(t in 3:length(sim_ts)){
  fitted[t] <- ar_fit$x.mean +
    ar_fit$ar[1]*(sim_ts[t-1] - ar_fit$x.mean) +
    ar_fit$ar[2]*(sim_ts[t-2] - ar_fit$x.mean)
}
fitted_ts <- ts(
  fitted,
  start = c(1,1),
  freq = 24
)

# plot
plot(sim_ts)
lines(fitted_ts, lty = 2, col = 2)
```



5. Verify that the observed AR(2) model you have written out is correct by calculating the residuals (observed - fitted) by hand, and comparing them with `ar_fit$resid`.

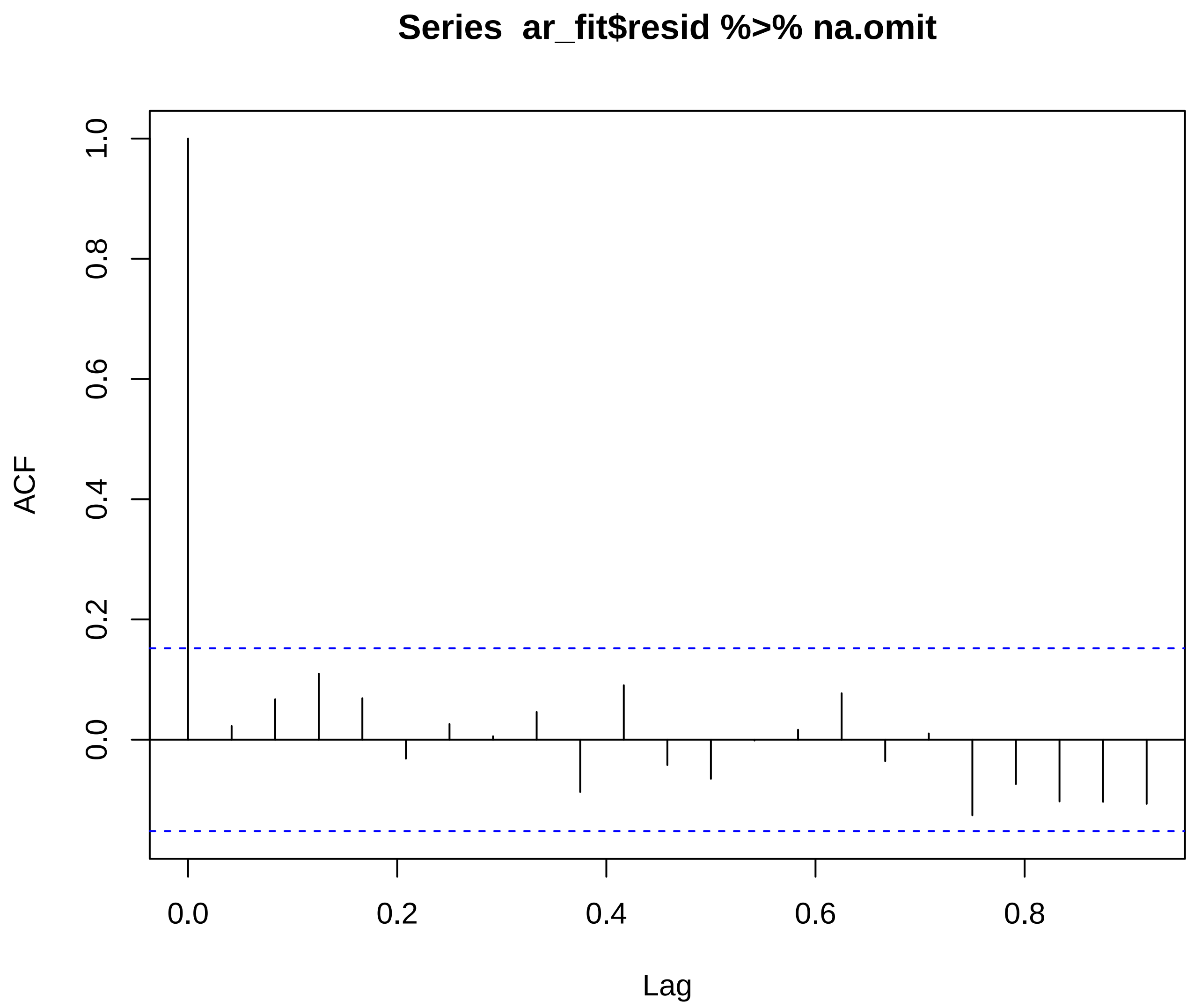
```
qqplot(ar_fit$resid, sim_ts - fitted_ts)
```



```
# nice
```

6. Create an acf plot of the residual error series from the `ar_fit` and comment on what the plot suggests about residual autocorrelation.

```
acf(ar_fit$resid %>% na.omit)
```



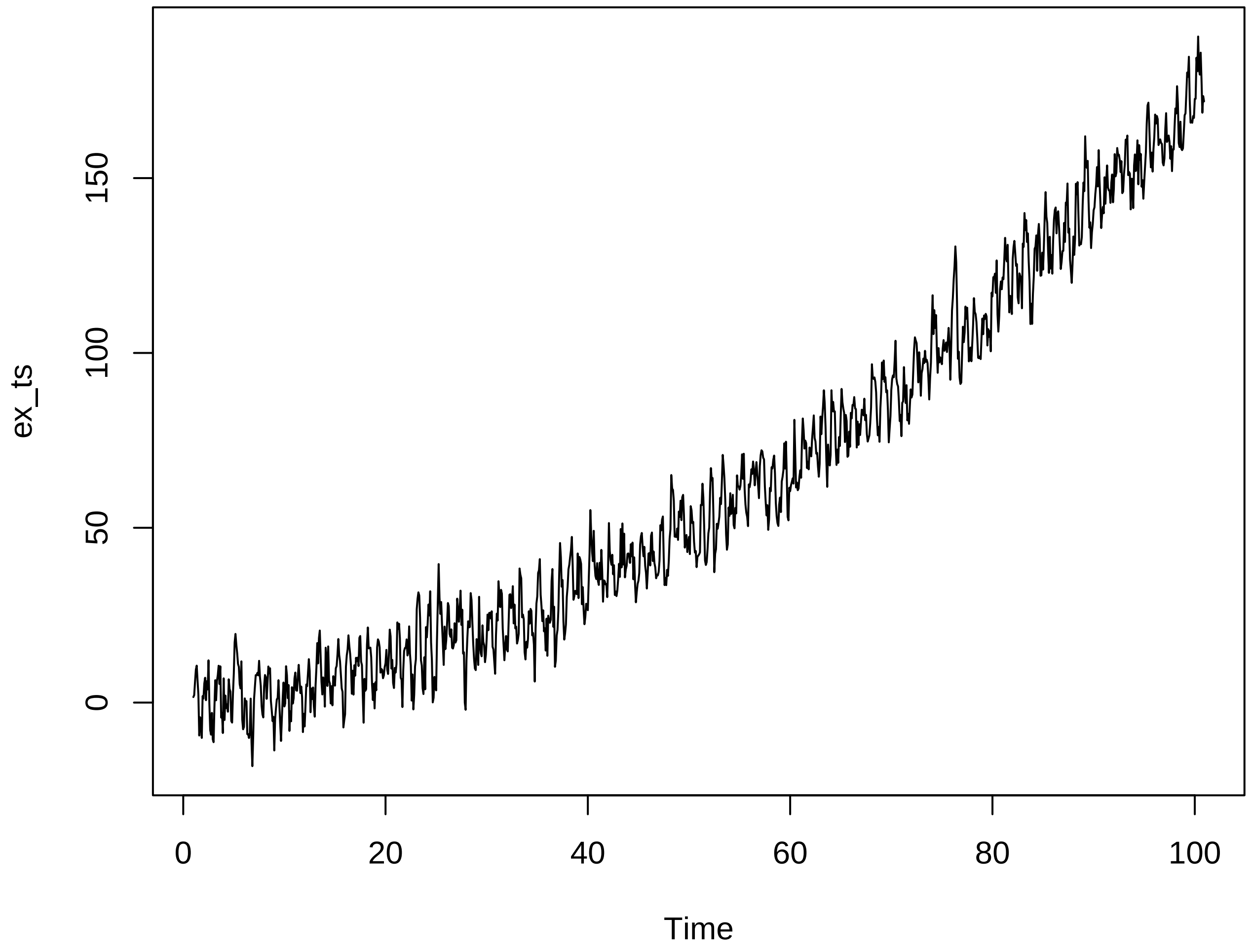
Looks pretty good! There does not appear to be any serial autocorrelation leftover in the residual error series.

AR processes are nice tools to model serial autocorrelation at various lags, but they are not useful for modeling trends or seasonal variation. Typically, AR models are paired with regression models (or other more advanced stochastic models) to construct flexible time series models (which we will begin formally next week). For now, I would like to whet your appetite for time series regression with the following demonstration. The function below generates a time series regression model of the form:

$$y_t = b_0 + b_1 t + b_2 t^2 + a_1 \sin\left(\frac{2\pi t}{f}\right) + a_2 \cos\left(\frac{2\pi t}{f}\right) + z_t$$

where z_t is an AR(2) process.

```
gen_ts_reg <- function(
    N = 144,
    b = c(0, .5, .001),
    a = c(5, -5),
    f = 12,
    alpha = c(.5, .2),
    sigma2 = 25
){
  # function to generate time series of the form
  # y_t = b0 + b1*t + b2*t^2 + a1*sin(2*pi*f*t) + a2*cos(2*pi*f*t) + zt
  # where zt is an AR(2) model

  # generate error series first
  zt <- wt <- rnorm(N, 0, sqrt(sigma2))
  for(t in 3:N) zt[t] <- alpha[1]*zt[t-1] + alpha[2]*zt[t-2] + wt[t]

  # reg piece next
  t <- 1:N
  y <- b[1] + b[2]*t + b[3]*t^2 + a[1]*sin(2*pi*t/f) + a[2]*cos(2*pi*t/f) + zt
  y_ts <- ts(
    y,
    start = c(1, 1),
    freq = f
  )
  return(y_ts)
}

# example
ex_ts <- gen_ts_reg(
  N = 1200,
  b = c(0, .025, .0001),
  a = c(5, -5),
  f = 12,
  alpha = c(.5, .1),
  sigma2 = 25
)
plot(ex_ts)
```



7. Play around with the `gen_ts_reg` function arguments to generate a couple of different series and begin to develop intuition for how time series regression works.

8. The code below fits a regression model to the simulated time series `ex_ts` and prints out the coefficient estimates. What do you notice about how these values compare to the values used to generate `ex_ts` above?

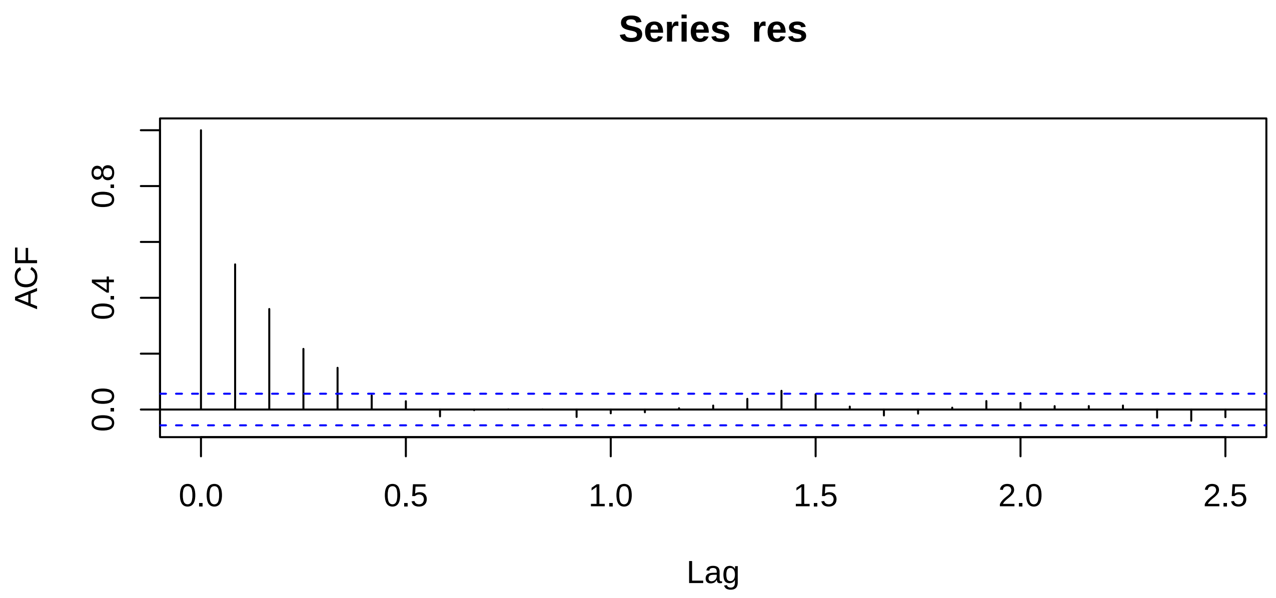
```
reg <- lm(
  y ~ t + I(t^2) + sint + cost,
  data = tibble(
    y = c(ex_ts)
  ) %>%
    mutate(
      t = 1:n(),
      sint = sin(2*pi*t/12),
      cost = cos(2*pi*t/12)
    )
)
reg
```

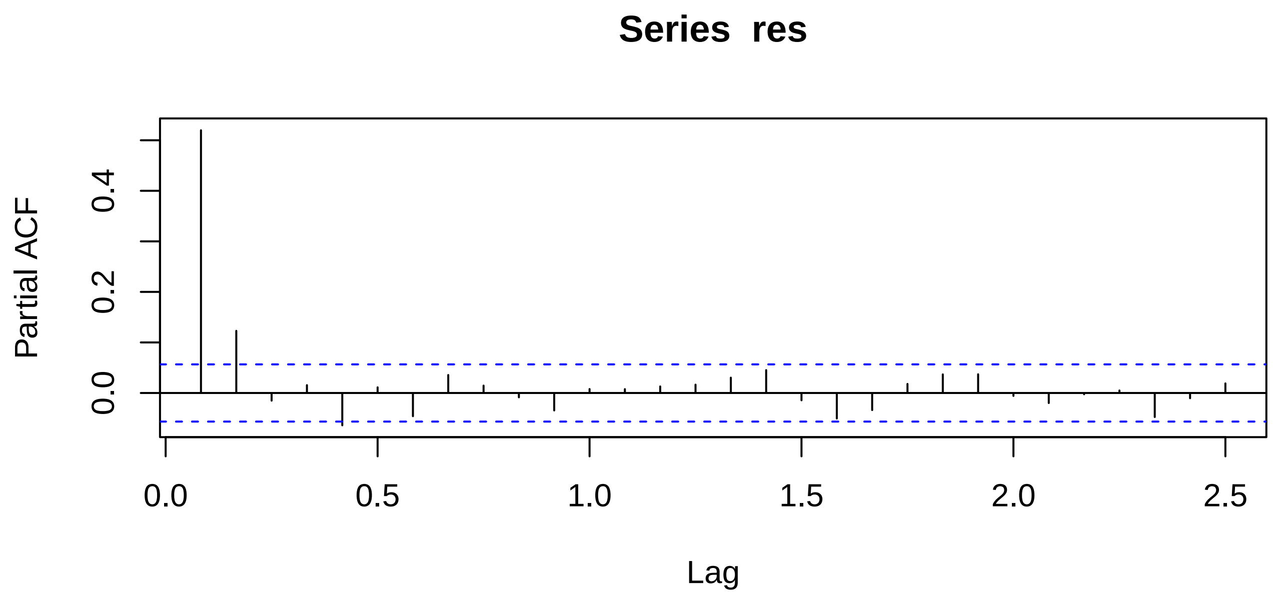
```
Call:
lm(formula = y ~ t + I(t^2) + sint + cost, data = tibble(y = c(ex_ts)) %>%
    mutate(t = 1:n(), sint = sin(2 * pi * t/12), cost = cos(2 *
        pi * t/12)))

Coefficients:
(Intercept)            t       I(t^2)         sint         cost
  0.3500663    0.0215831    0.0001039    4.6046568   -4.9299267
```

They are very similar to the generating values! Not exactly the same as a result of Monte Carlo error, but very similar. That is the purpose of simulation. :)

9. The code below extracts the residuals from the regression model, which represent our residual error series, and creates acf and pacf plots of the series. Do these plots look as we would expect? Why or why not?

```
res <- ts(
  resid(reg),
  start = c(1,1),
  freq = 12
)
par(mfrow = c(2,1))
acf(res)
pacf(res)
```





Yes! The pacf plot suggests that the auto correlation at lag 1 and 2 are around .5 and .1, respectively. These are exactly the values we specified.