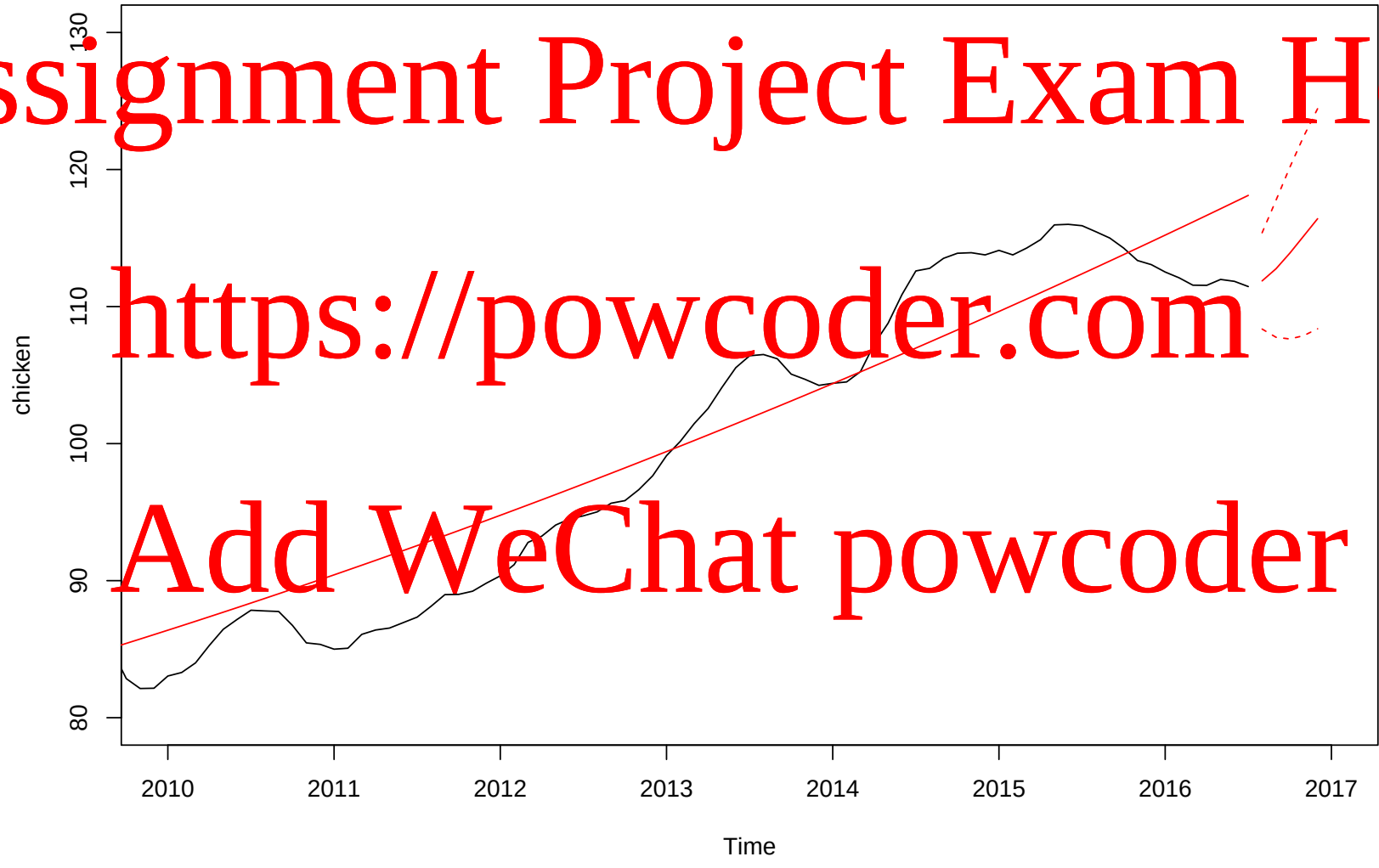
ARMA Models

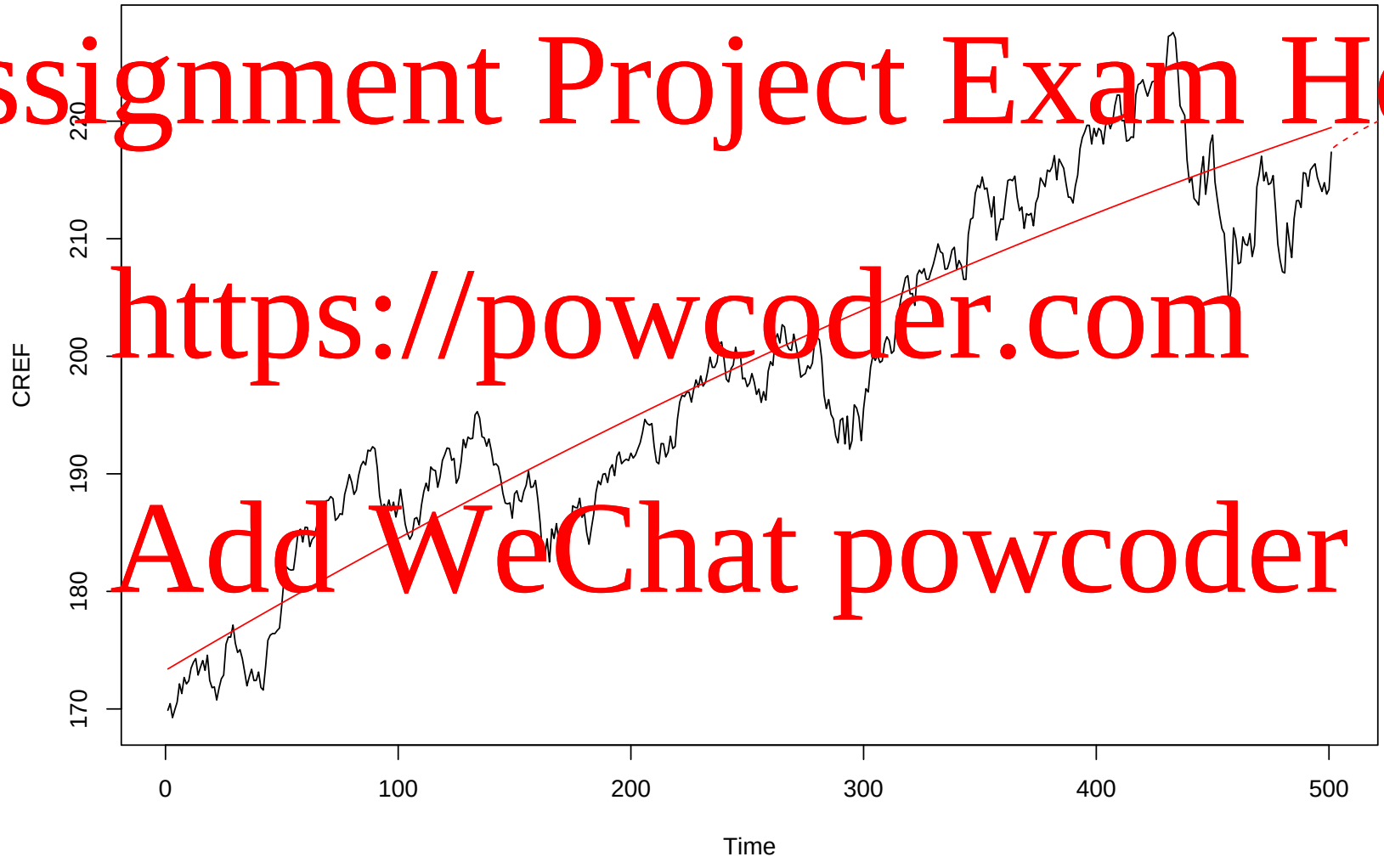
MAS 640

1/24/2018

Chicken Price



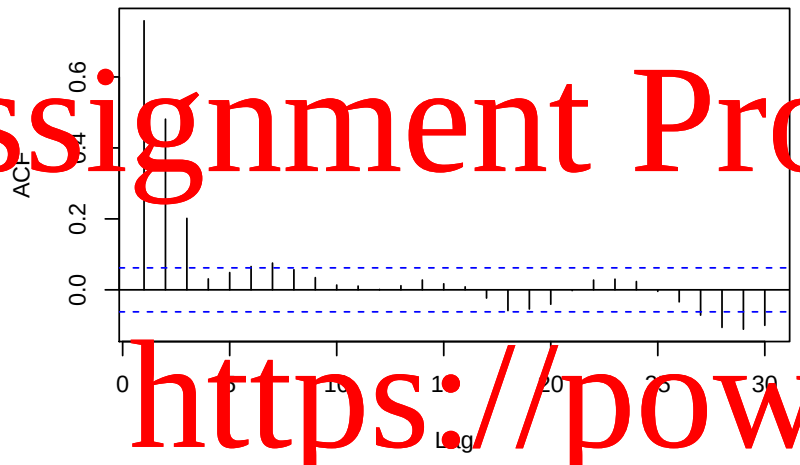
CREF Data Example

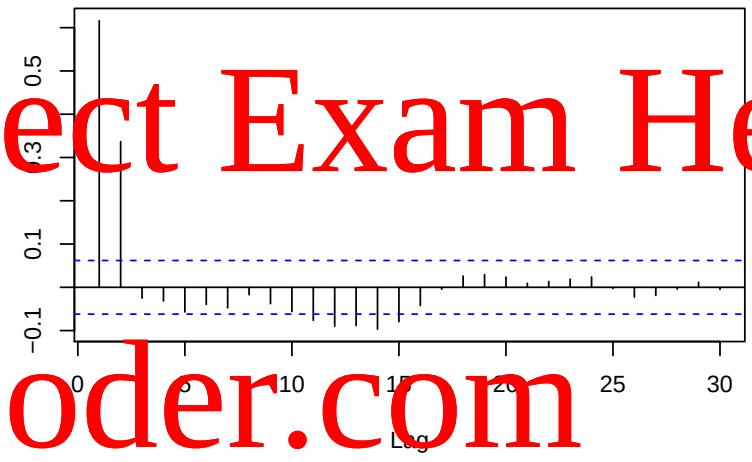


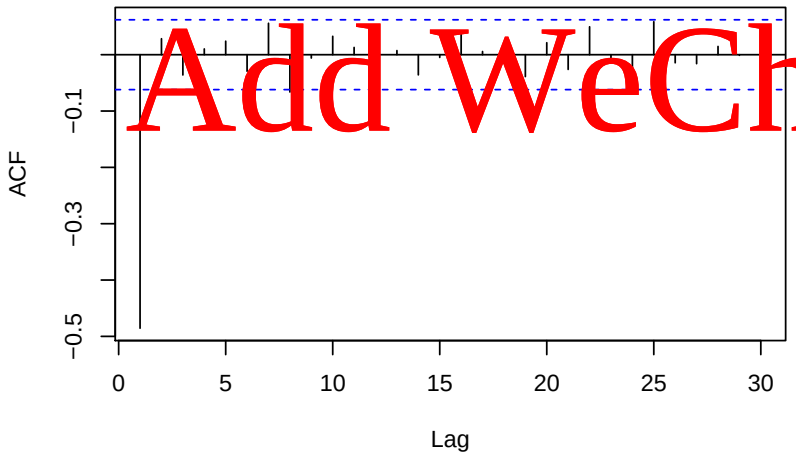
MA(q) Model

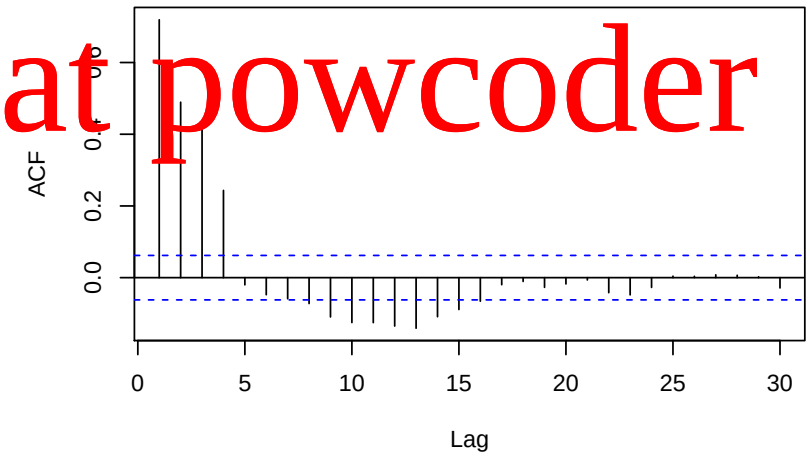
- MA(q) models have 0 autocorrelation beyond lag q

ACF Plots for MA(q) Processes





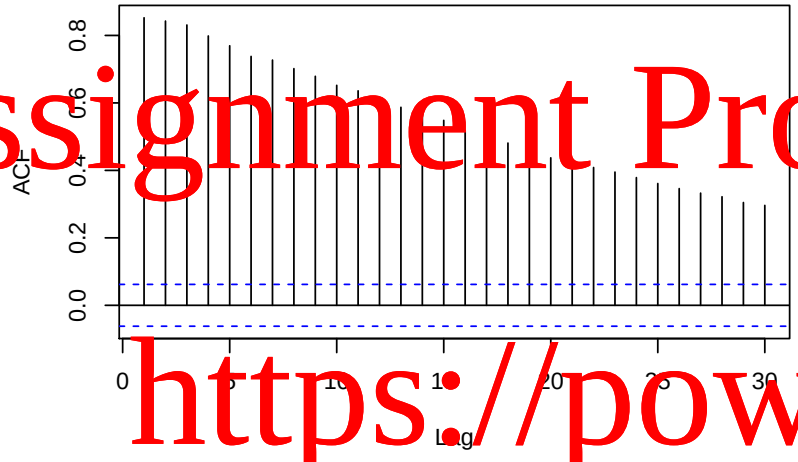


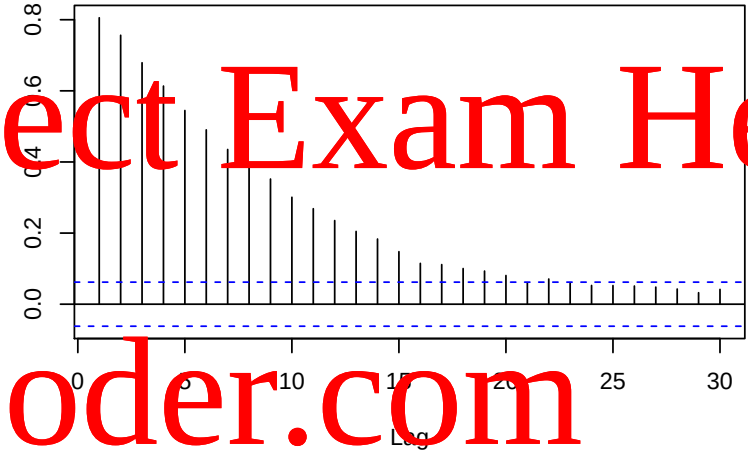


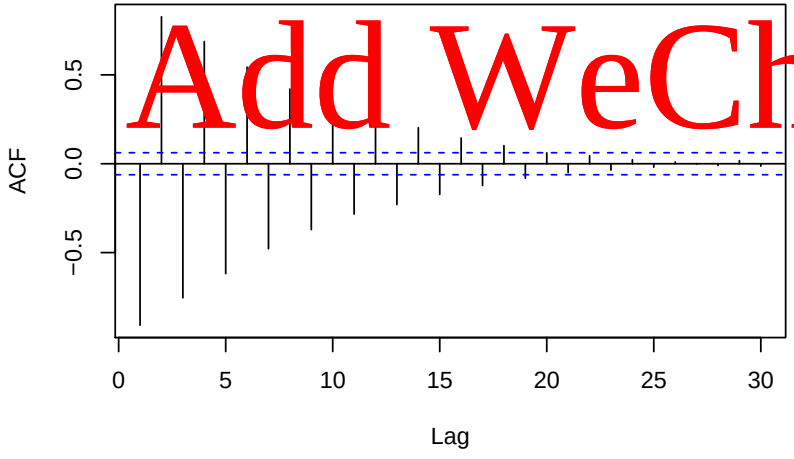
AR(p) Model

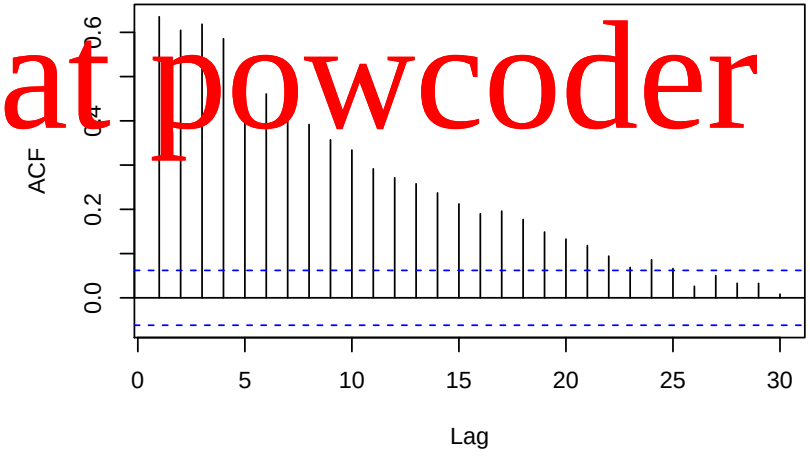
- Autocorrelation for AR(p) process decays over time, but does not cut off to 0

ACF Plots for AR(p) Processes

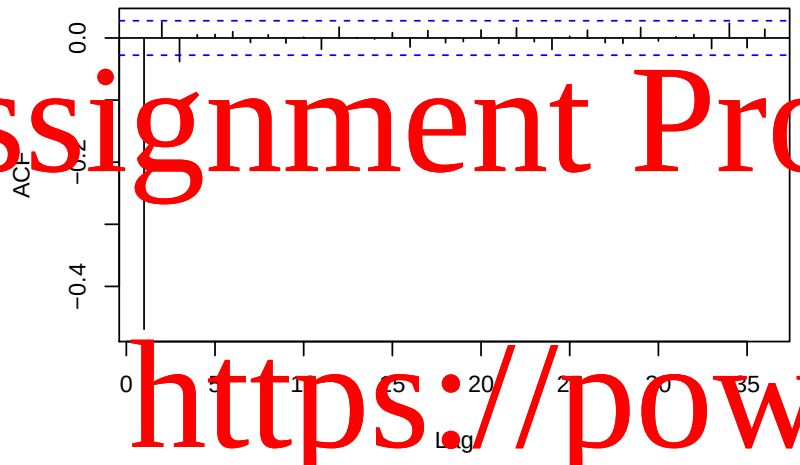


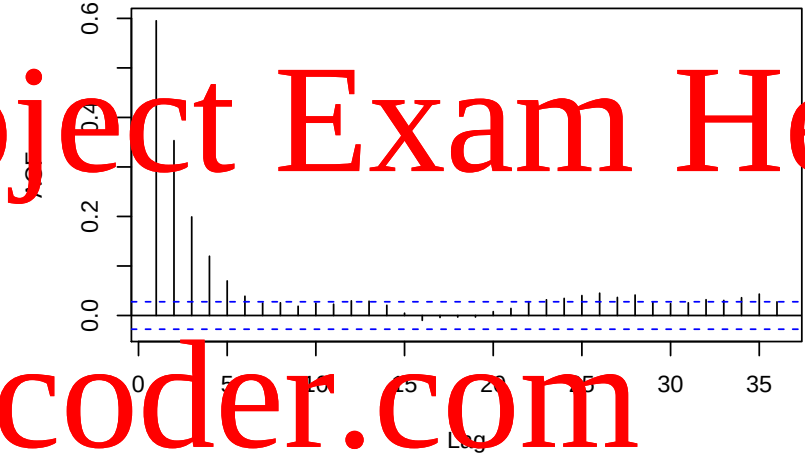


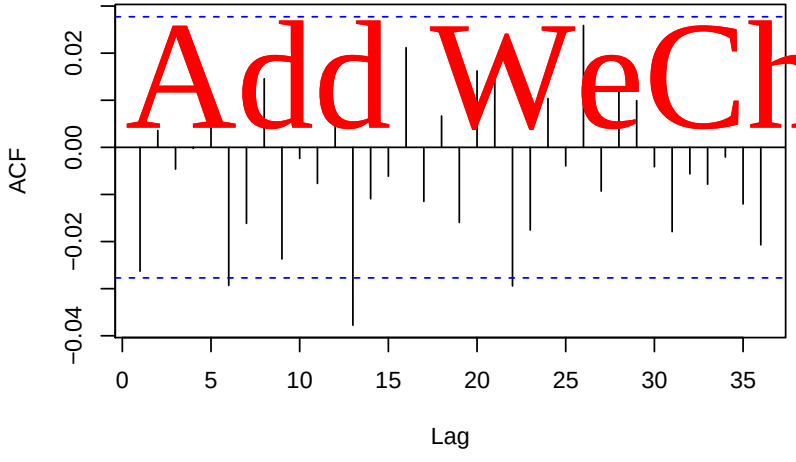


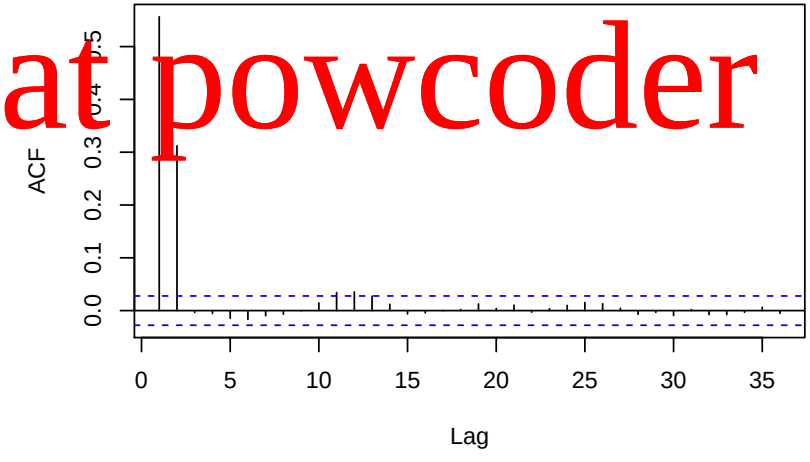


Which Model Generated These Plots?









Reading the ACF Plot

	AR(p)	MA(q)
ACF	Tails off	Cuts off after lag q

- Knowing this, we can quickly determine...
  - If we need MA components in our model (and how many)
  - If we need AR components
- Unfortunately, due to the nature of ACF for AR processes, we cannot determine the order, p

Motivation

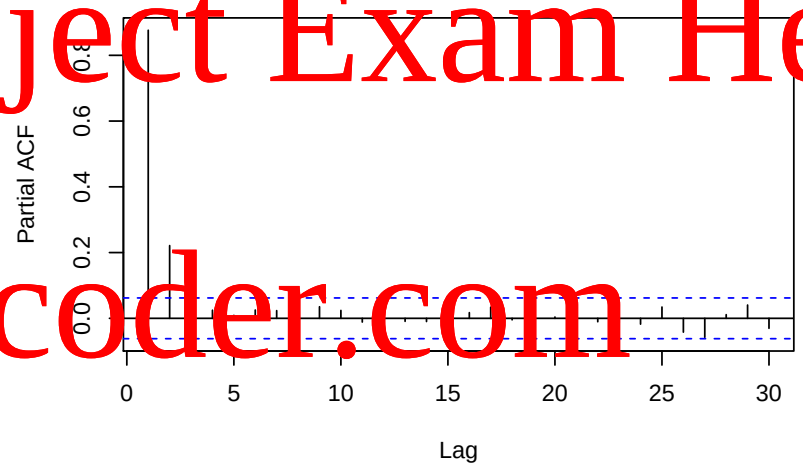
- The ACF provides a considerable amount of info about the order of the dependence or MA processes
- Unfortunately, it doesn't for AR processes
- Need a new function that behaves like the ACF for MA models, but for use with AR models instead.
  - This function is called the partial autocorrelation function (PACF).

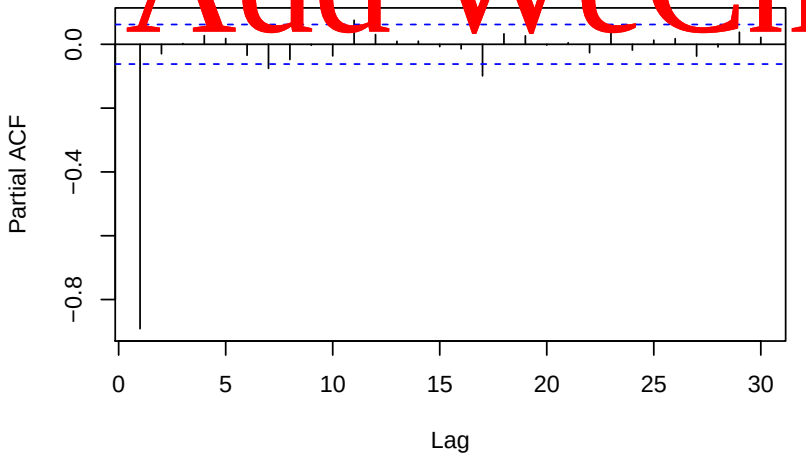
Partial Autocorrelation Function (PACF)

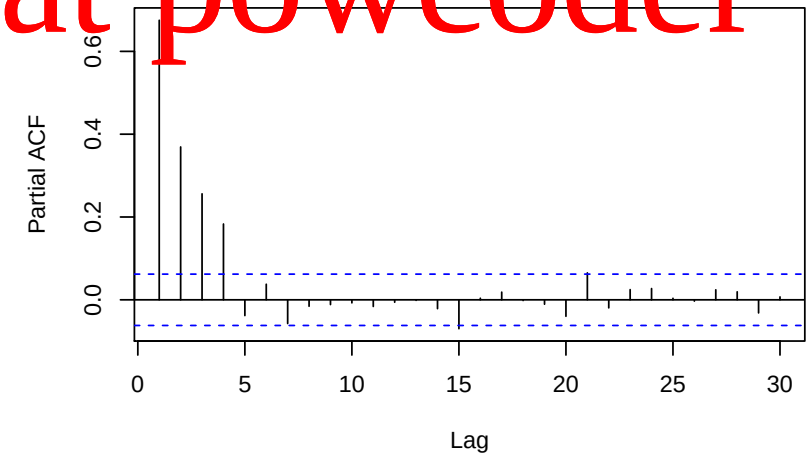
- **Basic idea:** What is the correlation between Y_t and Y_{t-k} after taking into account $Y_{t-1}, Y_{t-2}, \cdots Y_{t-k+1}$
- **Technically:**
  1. Regress Y_t on $Y_{t-1}, Y_{t-2}, \cdots Y_{t-k+1}$ and find $\hat{Y}_t$
  2. Regress Y_{t-k} on $Y_{t-1}, Y_{t-2}, \cdots Y_{t-k+1}$ and find $\hat{Y}_{t-k}$
  3. Use correlation between $\hat{Y}_t$ and $\hat{Y}_{t-k}$
- "Partial" because I'm removing everything in the middle...
- `pacf()` for PACF, or `acf2()` for both ACF and PACF

PACF for AR(p) Models

- Can you guess the AR order, p?

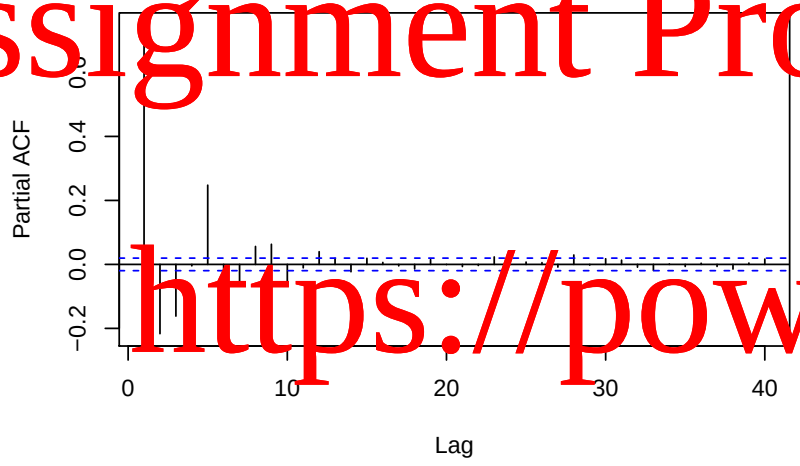


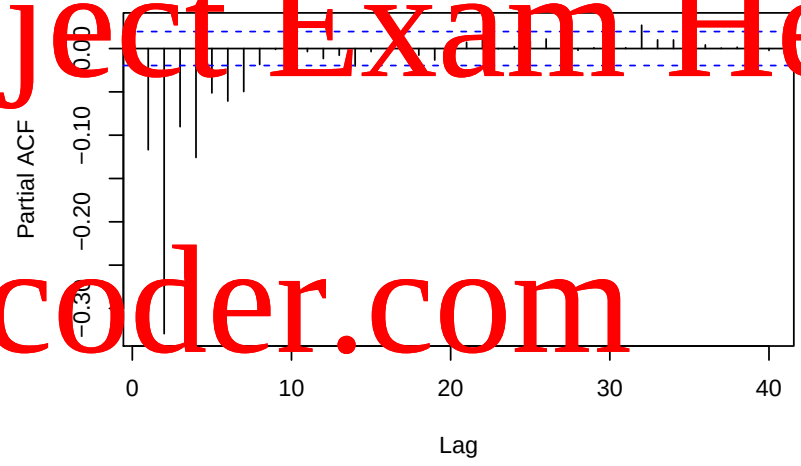


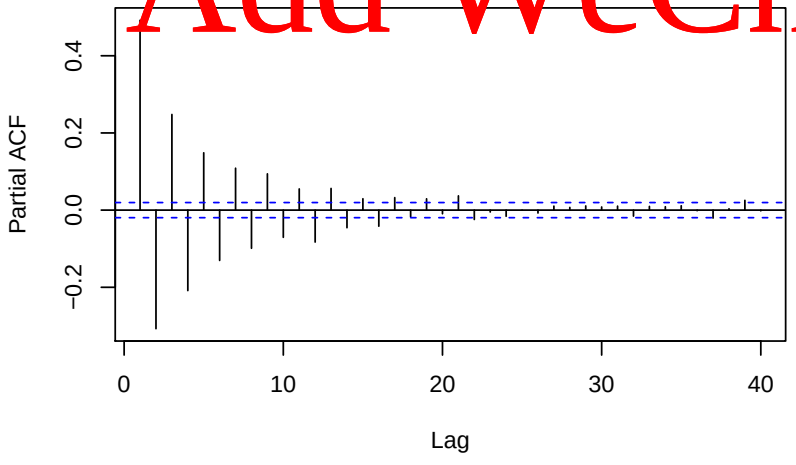


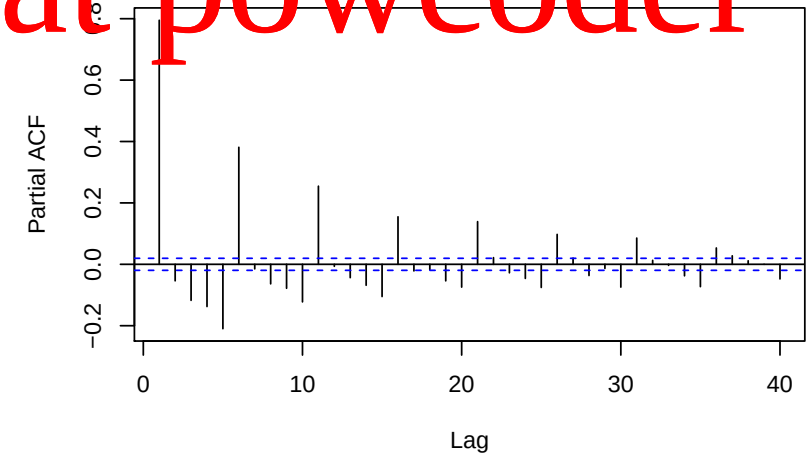
ACF Plots for MA(q) Processes

- Can you guess the MA order q?









Determing AR and MA Components

	AR(p)	MA(q)
ACF	Tails off	Cuts off after lag q
PACF	Cuts off after lag p	Tails off

- Now we are comfortable with these!
- Let's use them to build more complicated models

Autoregressive Moving Average (ARMA) Process

Suppose $\{e_t\}$ is a zero-mean white noise process with $var(e_t) = \sigma_e^2$. The process

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + e_t + \theta_1 e_{t-1} + \ldots + \theta_q e_{t-q}$$

is an autoregressive moving average process with AR order p and MA order q, written as ARMA(p,q).

ARMA(p,q) Process

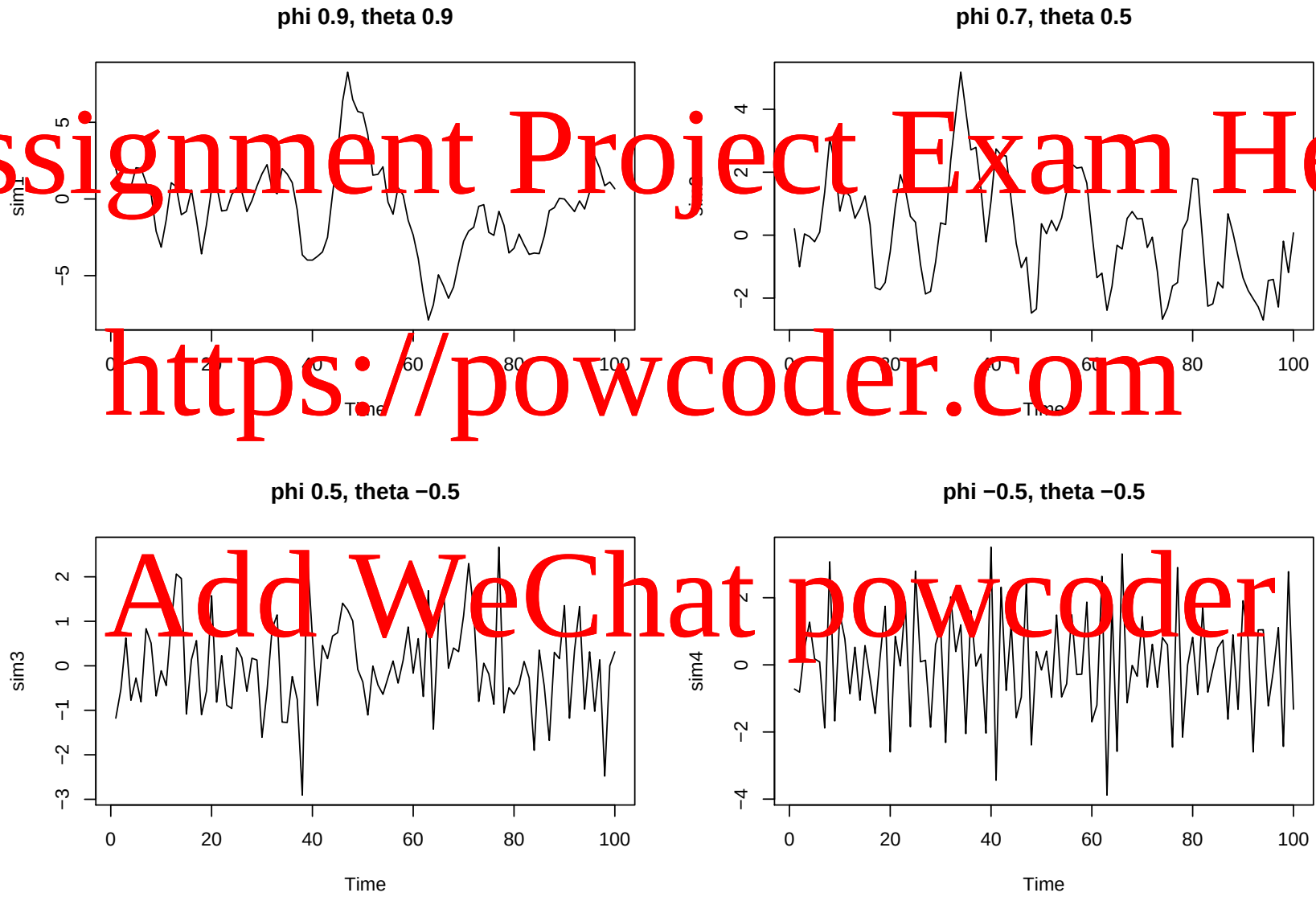
- Special cases
  - ARMA(0, 0):
  - ARMA(p,0):
  - ARMA(0,q):

Remembering the Goal

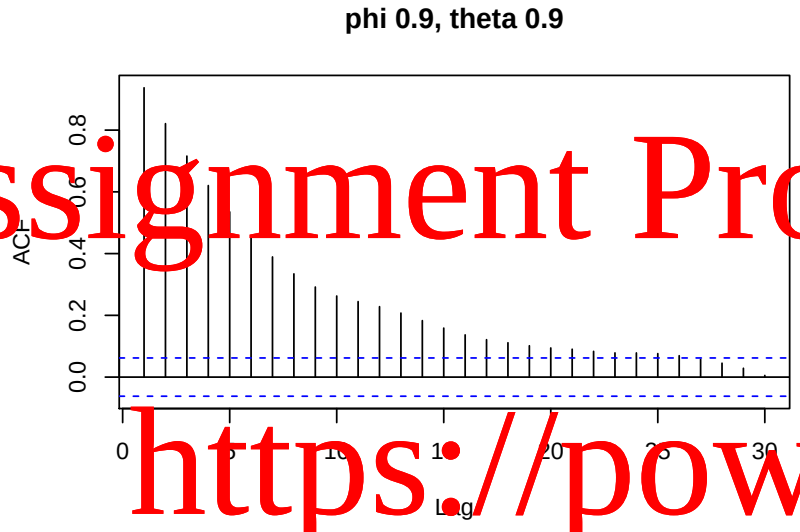
$$Y_t = \mu_t + e_t$$

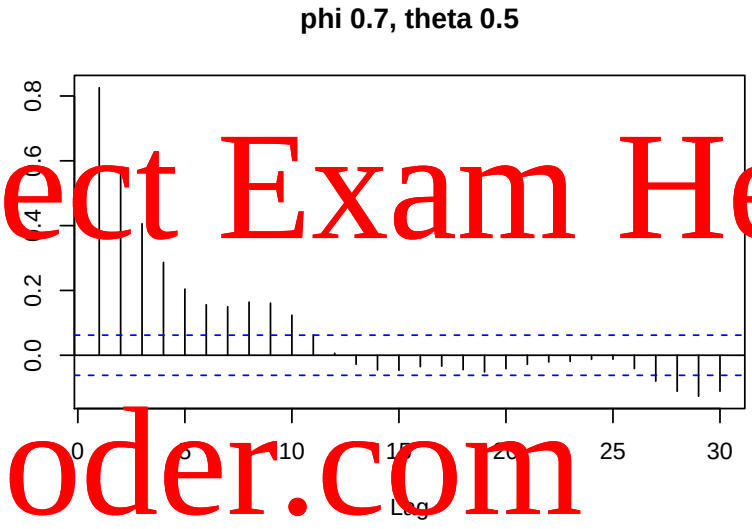
- We model μ_t with standard linear regression techniques
- We are left with the error terms
  - These errors are correlated
  - We want to model them
- These errors can be correlated with past values of itself (AR) or past errors (MA)
  - We just need to determine which components (AR or MA), and how many of each, to include in our model

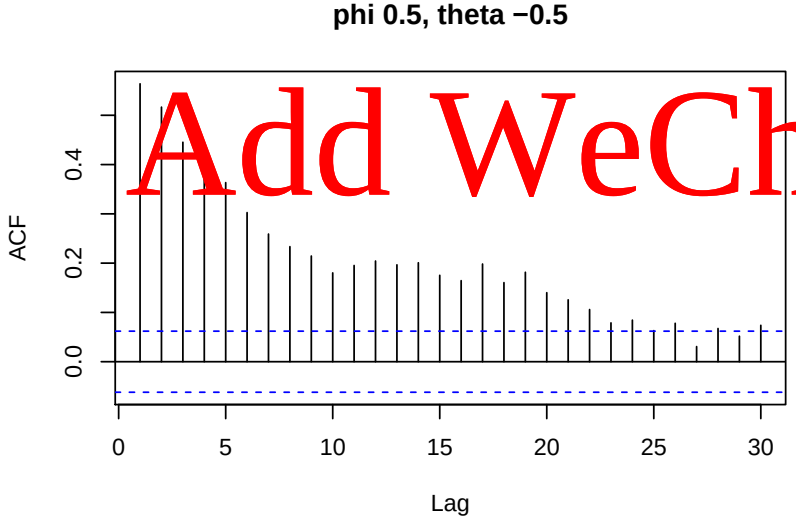
Some ARMA(1, 1) Processes

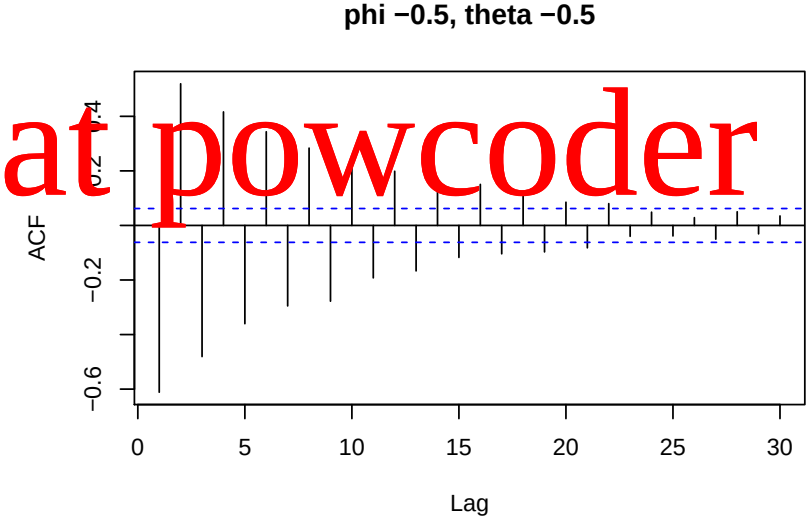


ACF for ARMA(1,1) Processes

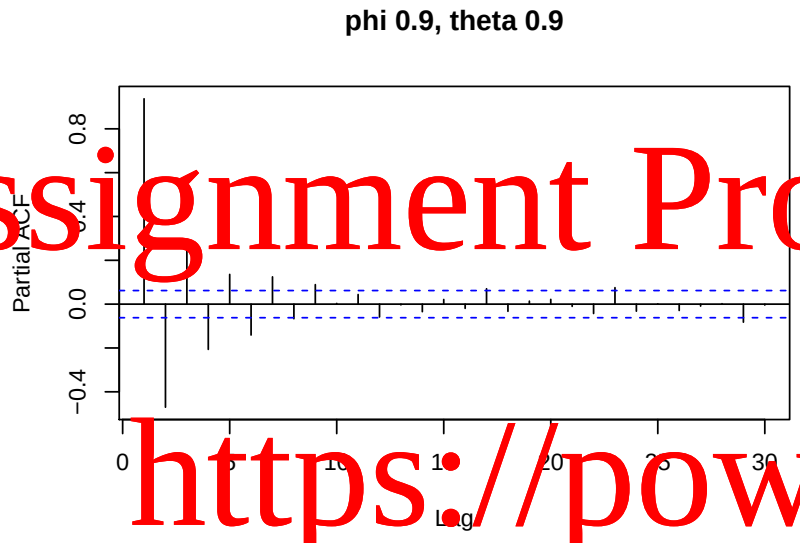


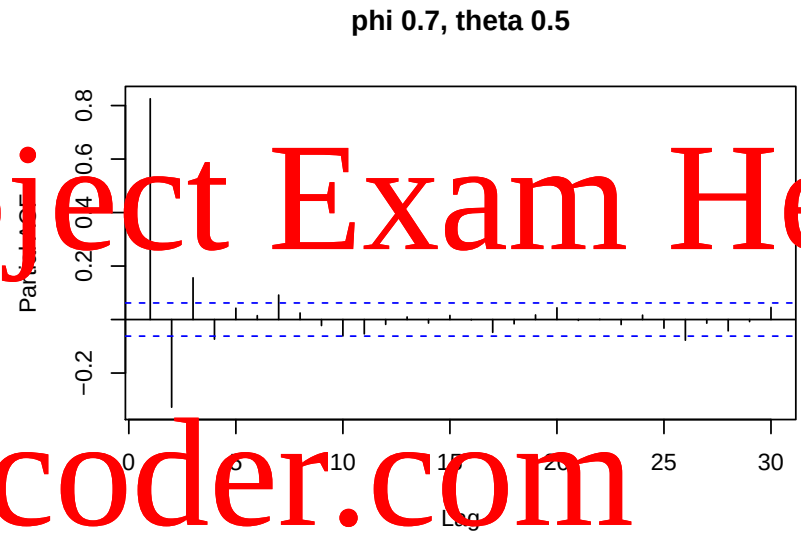


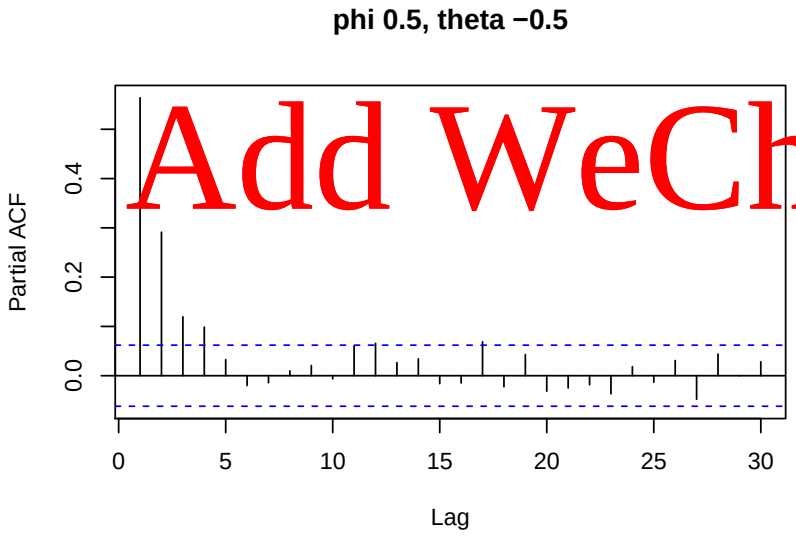


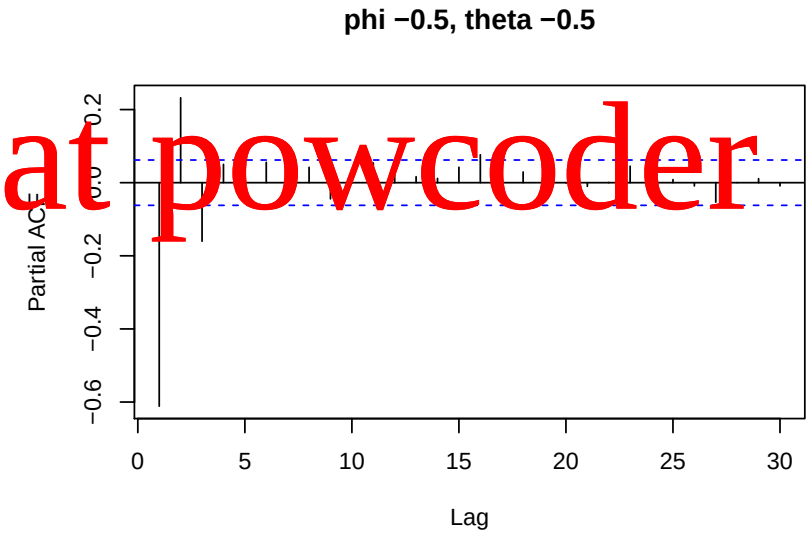


PACF for ARMA(1,1) Processes









Determining Order for ARMA(p, q)

- Unfortunately, no clear "drop-offs" are present in the ACF or PACF plots
- So we can't determine p or q if AR and MA components are both present in model
- We do have the following to guide the process:

	AR(p)	MA(q)	ARMA(p,q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

Fitting an ARMA(p, q) Model

- If the ACF and PACF both tail off, this is evidence of an ARMA model
- But how to determine p and q?
  - "Go fit an ARMA model to this data..."

Fitting ARMA Models in R

- Previously used `ar()` function to fit an AR model and let it decide the order p by comparing AIC values
- Need something more comprehensive for ARMA models
  - `arma()`
  - `arima()`
  - `sarima()`

- Each of these accomplish the same thing, some are easier to use
- I will stick with `sarima()` for the remainder of the course

The `sarima()` Function

Seasonal **A**uto**r**egressive **I**ntegrated **M**oving **A**verage

```
sarima(x, p, d, q, P=0, D=0, Q=0, S=-1, ...)
```

- Where
  - x is the data
  - p and q are the AR and MA orders (note that you now have to specify)
  - d tells the function to difference the data d times (next class, use 0 for now)
  - The capital letters are seasonal components that we ignore for now
  - x, p, d, and q are required

The 'sarima.for()' Function

- `sarima()` not compatible with `predict()`, but it does come with it's own function for forecasting


```
sarima.for(x, n.ahead, p, d, q, ...)
```

Examples Using `sarima()`

Write the code for fitting each of the following models to data x:

1. AR(1) -
2. MA(3) -

3. AR(2) -
4. ARMA(1, 1) -

5. ARMA(2, 1) -

Examples Using `sarima.for()`

Write the code for forecasting the next 6 time periods for each of the following models:

1. AR(1) -
2. MA(3) -
3. AR(2) -
4. ARMA(1, 1) -
5. ARMA(2, 1) -