Analysis of Time Series Data Using R

Preliminaries

Always remember to load the packages that we need to use for analyzing time series in R. I've listed them below:

```
library(tidyverse)
library(tidyquant)
library(gridExtra)
library(tibbletime)
library(forecast)
library(itsmr)
library(tsibble)
library(fpp2)
library(knitr)
library(kableExtra)

knitr::opts_chunk$set(comment=NA,tidy=FALSE)

#library(future) Not needed yet
#library(doFuture) Not needed yet
#library(rbenchmark) Not needed yet
```

Developing intuition on simulated data

- Can use the `arima.sim` function to generate data according to various ARMA models
- `arima.sim` requires two arguments (there are others, but defaults are OK for our purposes): a model specification and a number of points in the series to generate
- Specify models by using a named `list`
- Ex.

```
true_ar_coef=c(0.6,0.2)
true_ma_coef=c(0.4)
my_ARMA_2_1_model = list(ar=true_ar_coef, ma=true_ma_coef)
my_ARMA_2_1_model
```

```
$ar
[1] 0.6 0.2

$ma
[1] 0.4
```

- The above code specifies an ARMA(2,1) model where $\phi = (\phi_1, \phi_2) = (0.6, 0.2)$ and $\theta = (\theta_1) = 0.4$. If you want only an AR or MA model, you can either leave out that component of the vector or set it to NULL.
- We can find (and plot the inverse of) the roots of the AR and MA polynomials using the code below. Note that we plot the *inverse* of the roots, as in most cases we will have causal and/or invertible series, so this keeps the plots nice as the roots will be inside of the circle instead of potentially far outside of it.

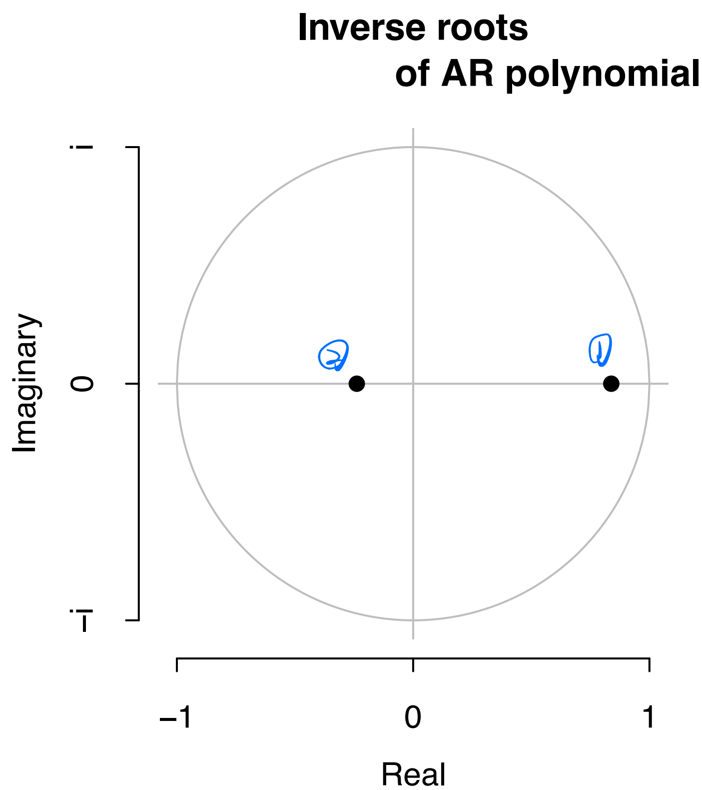
Find roots of poly

```
ar_roots<-polyroot(c(1,-my_ARMA_2_1_model$ar))
ar_roots
```

① ②
```
[1]  1.192582-0i -4.192582+0i
```

```
forecast:::plot.armaroots(structure(list(roots=ar_roots),
                          class = "armaroots"),xlab="Real", ylab="Imaginary",main="Inverse roots
                          of AR polynomial")
```

plot inverse of roots

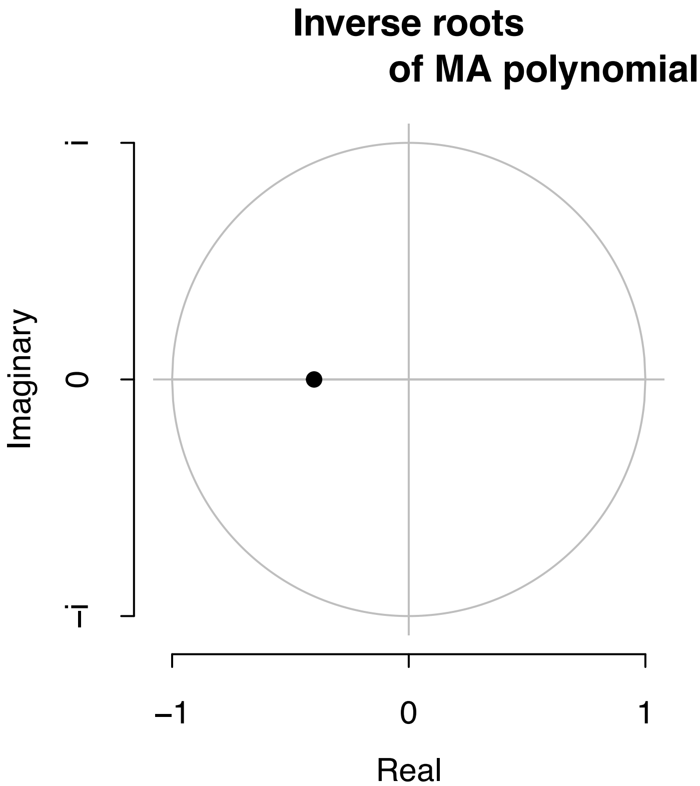


causal
⇒ inverse is inside

```
ma_roots<-polyroot(c(1,my_ARMA_2_1_model$ma))
ma_roots
```

```
[1] -2.5+0i
```

```
forecast:::plot.armaroots(structure(list(roots=ma_roots),
                          class = "armaroots"),xlab="Real", ylab="Imaginary",main="Inverse roots
                          of MA polynomial")
```



invertible

$\Rightarrow$ inverse is inside.

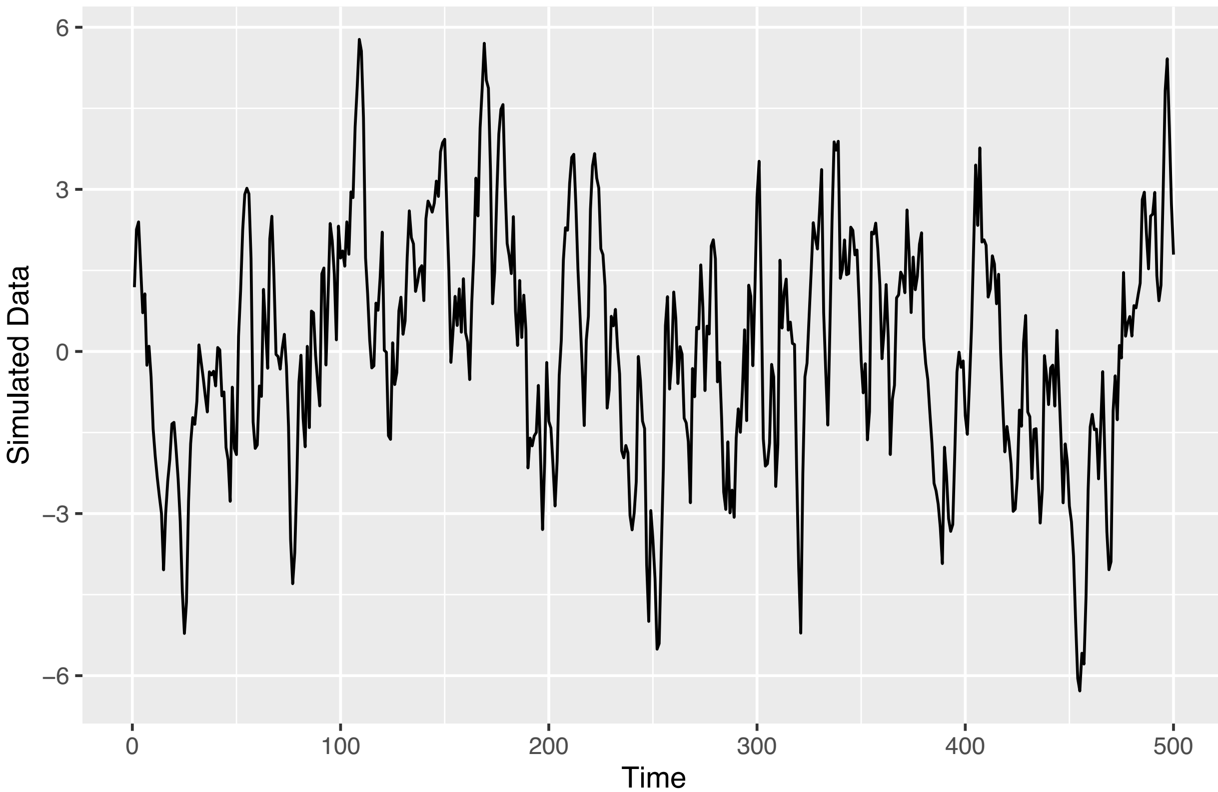
Simulating the data

- With these cofficients, we can now generate an ARMA(2,1) model using the `arima.sim` function:

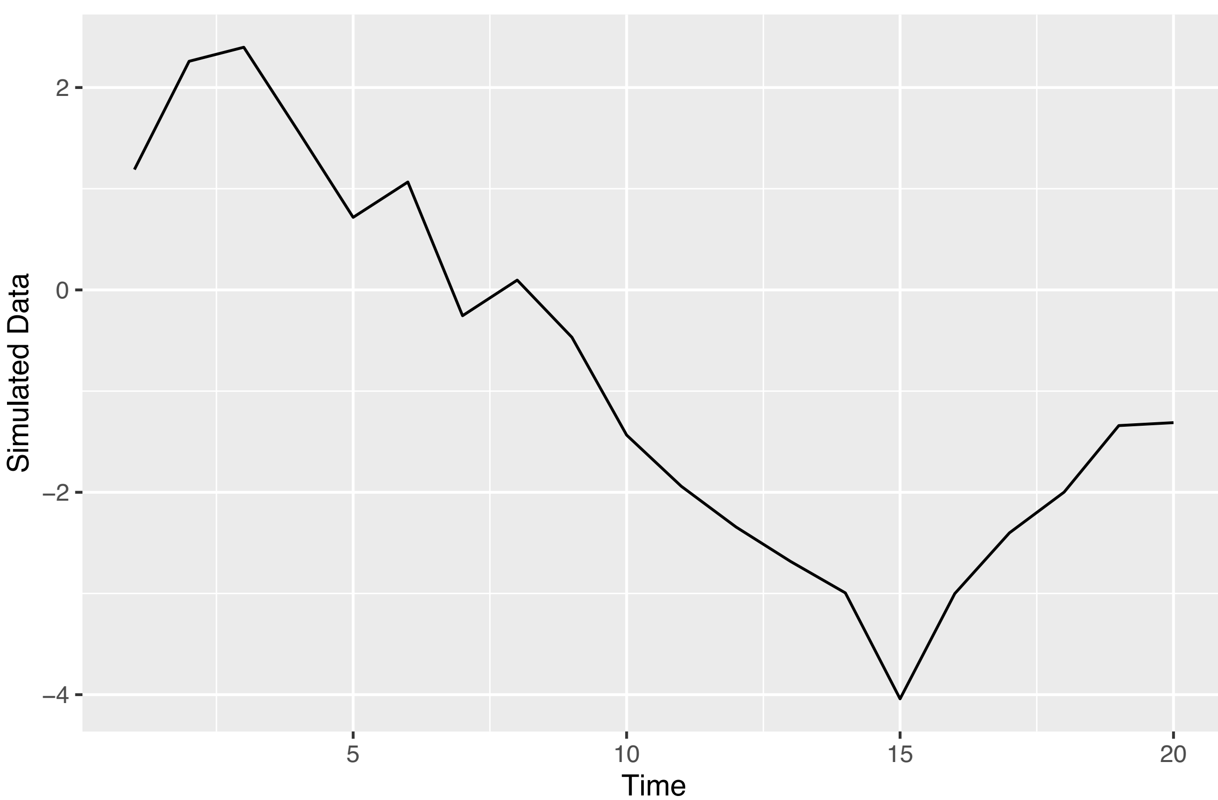
```
my_arma_2_1_data = arima.sim(my_ARMA_2_1_model,n=500)
glimpse(my_arma_2_1_data)
```

```
 Time-Series [1:500] from 1 to 500: 1.19 2.259 2.398 1.565 0.717 ...
```

```
autoplot(my_arma_2_1_data) + ylab("Simulated Data")
```




```r
autoplot(window(my_arma_2_1_data,end=20)) + ylab("Simulated Data")
```



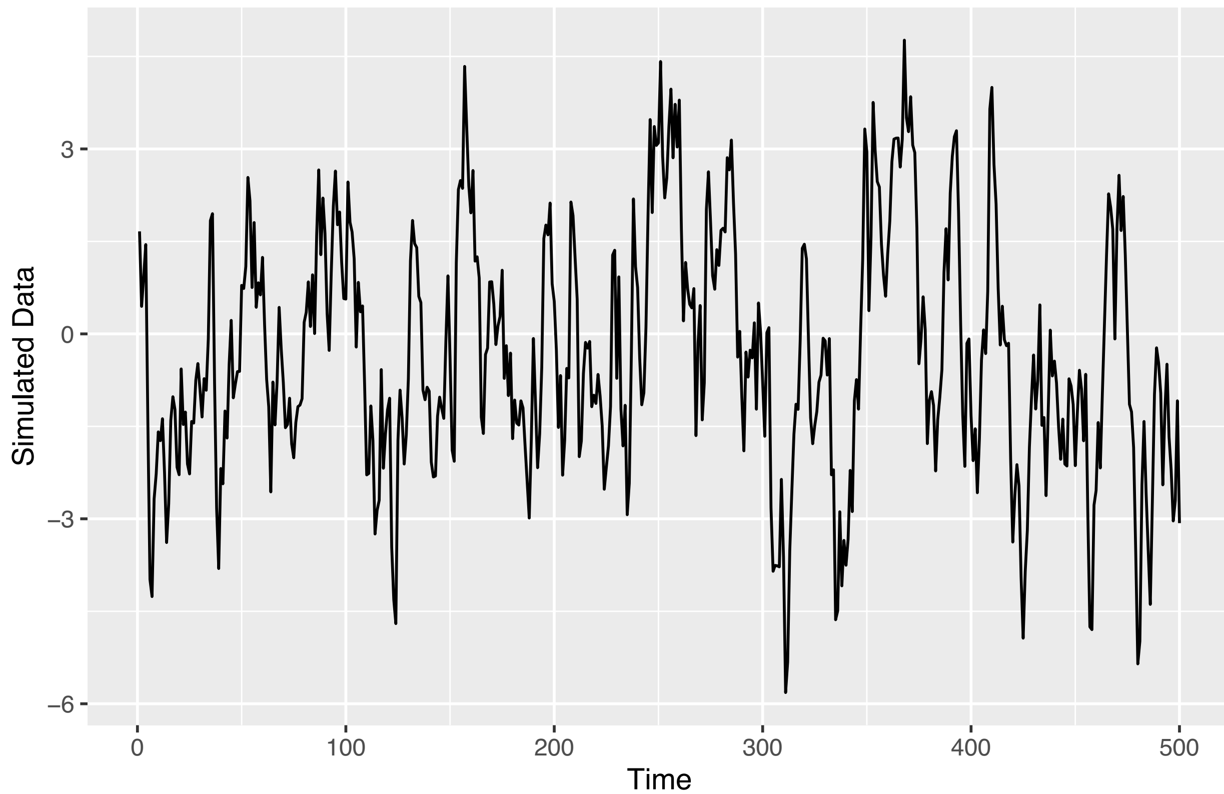
- Note if we want to be able to replicate our simulations (i.e. obtain the same dataset each time we run our code), we need to set a random seed (different seeds or running two sims without resetting the seed leads to different data):

```
set.seed(19750606)
my_arma_2_1_data = arima.sim(my_ARMA_2_1_model,n=500)
glimpse(my_arma_2_1_data)
```

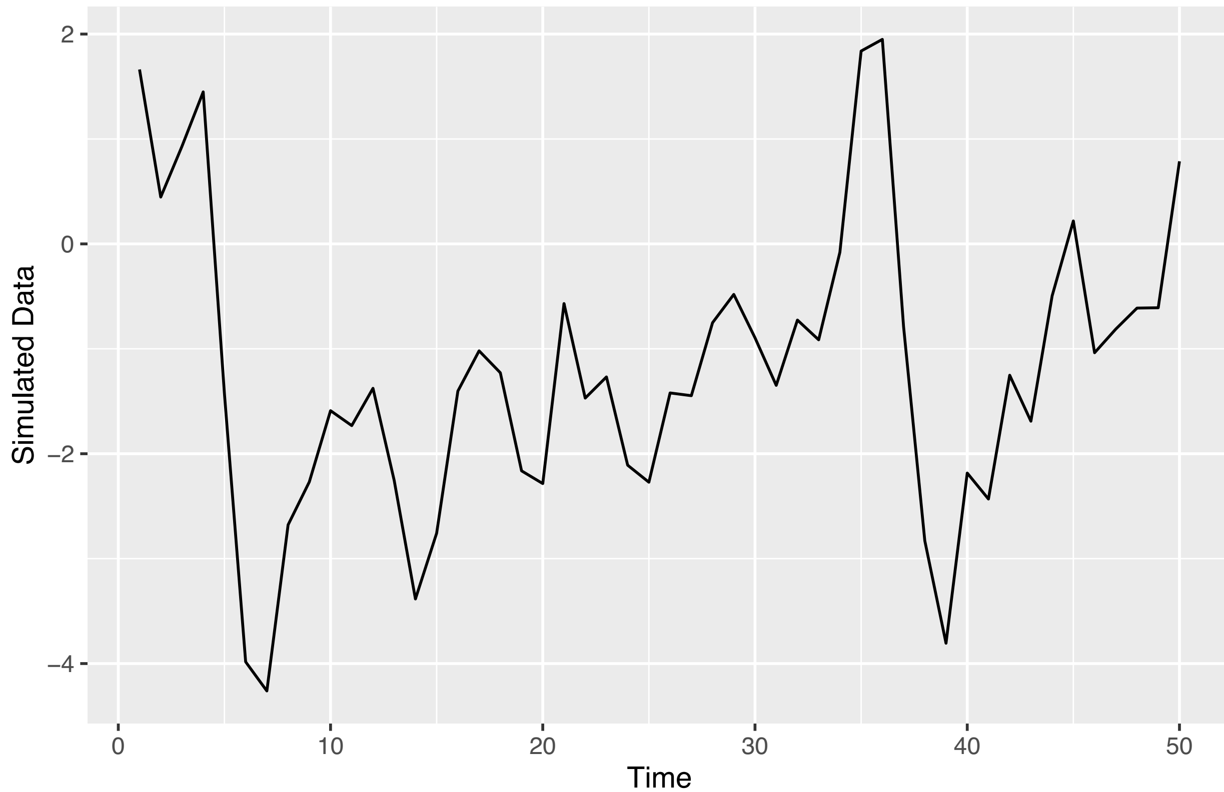
replicate simulations

```
 Time-Series [1:500] from 1 to 500: 1.663 0.446 0.93 1.449 -1.426 ...
```

```
autoplot(my_arma_2_1_data) + ylab("Simulated Data")
```



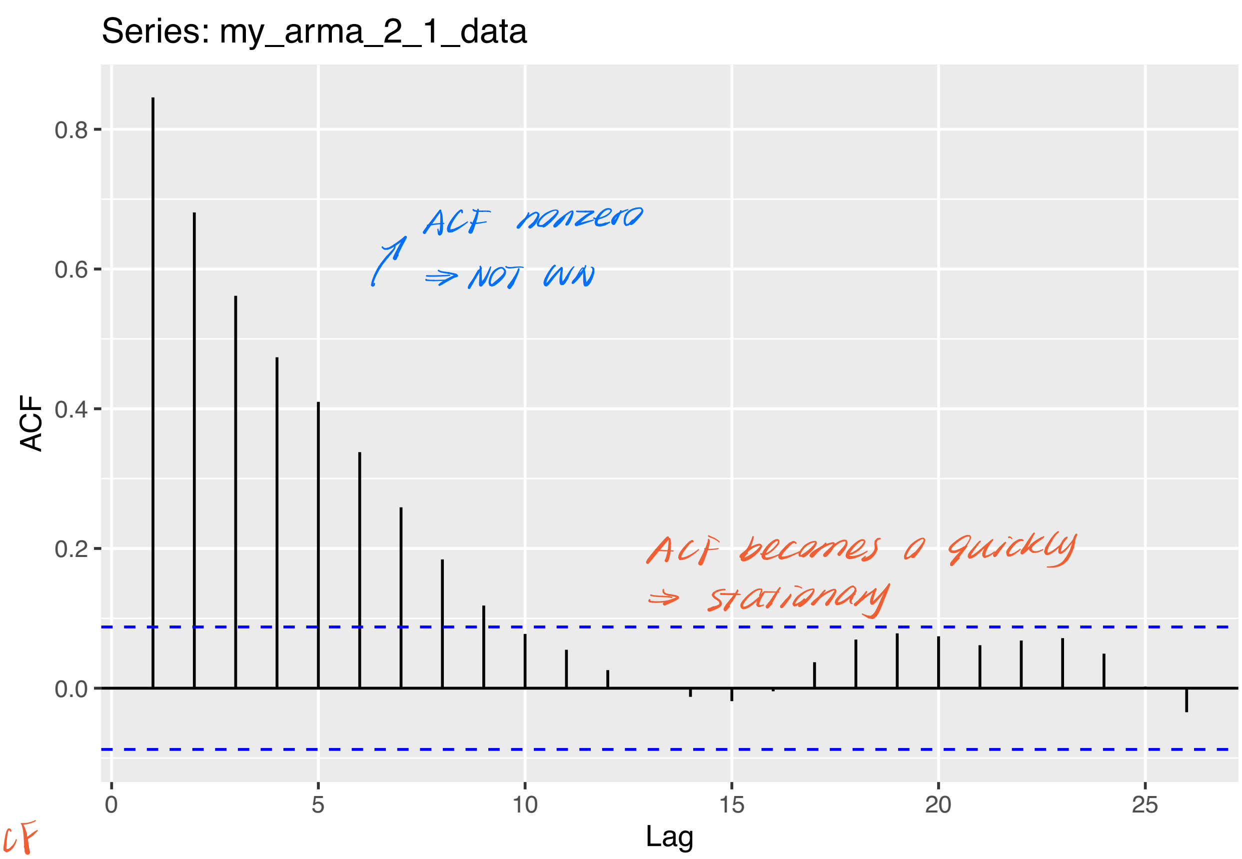

```
autoplot(window(my_arma_2_1_data,end=50)) + ylab("Simulated Data")
```



Sample autocorrelation

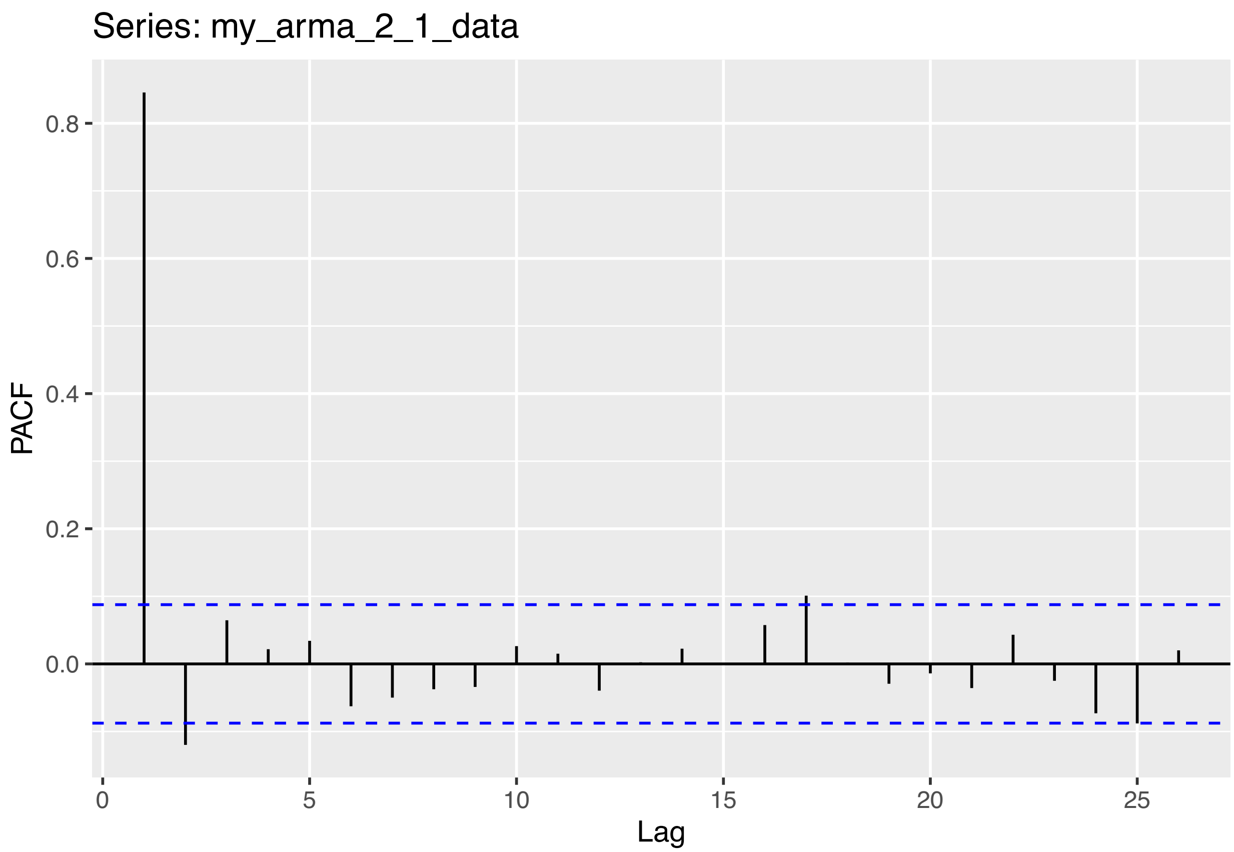
- Recall we can plot the autocorrelation for various lags using the `ggAcf` function:

```
ggAcf(my_arma_2_1_data)
```



PACF

```
ggPacf(my_arma_2_1_data)
```



ACF

- Can compute theoretical ACF and PACF values using the `ARMAacf` function and them to the plot:

```
ARMAacf(ar=true_ar_coef,ma=true_ma_coef,lag.max=10)
```

```
        0         1         2         3         4         5         6         7
1.0000000 0.8693182 0.7215909 0.6068182 0.5084091 0.4264091 0.3575273 0.2997982
        8         9        10
0.2513844 0.2107903 0.1767510
```

decrease over time

```
ARMAacf(ar=true_ar_coef,ma=true_ma_coef,lag.max=25) %>% head(.)
```

```
        0         1         2         3         4         5
1.0000000 0.8693182 0.7215909 0.6068182 0.5084091 0.4264091
```

PACF

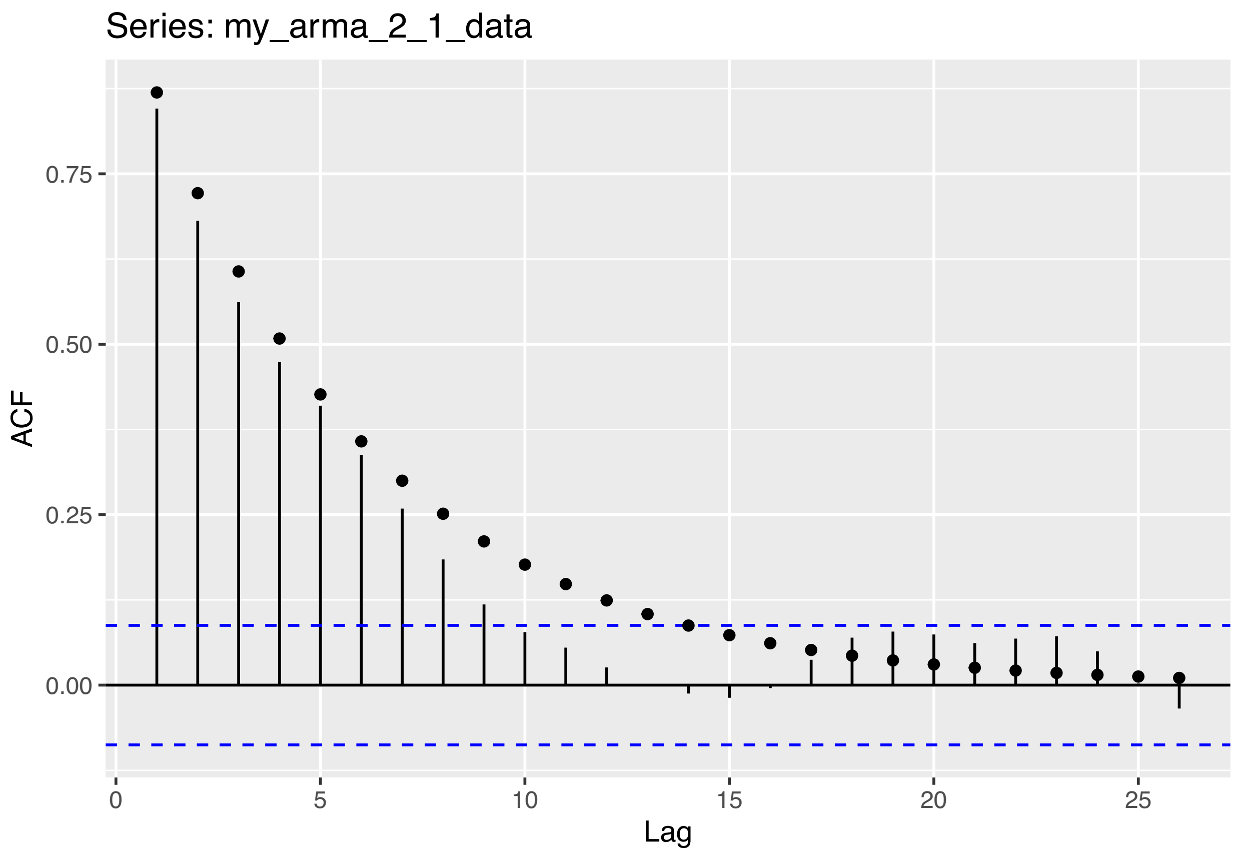
```
ARMAacf(ar=true_ar_coef,ma=true_ma_coef,lag.max=25,pacf=TRUE) %>% head(.)
```

```
[1]  0.869318182 -0.139685476  0.055668203 -0.022254154  0.008900822
[6] -0.003560275
```

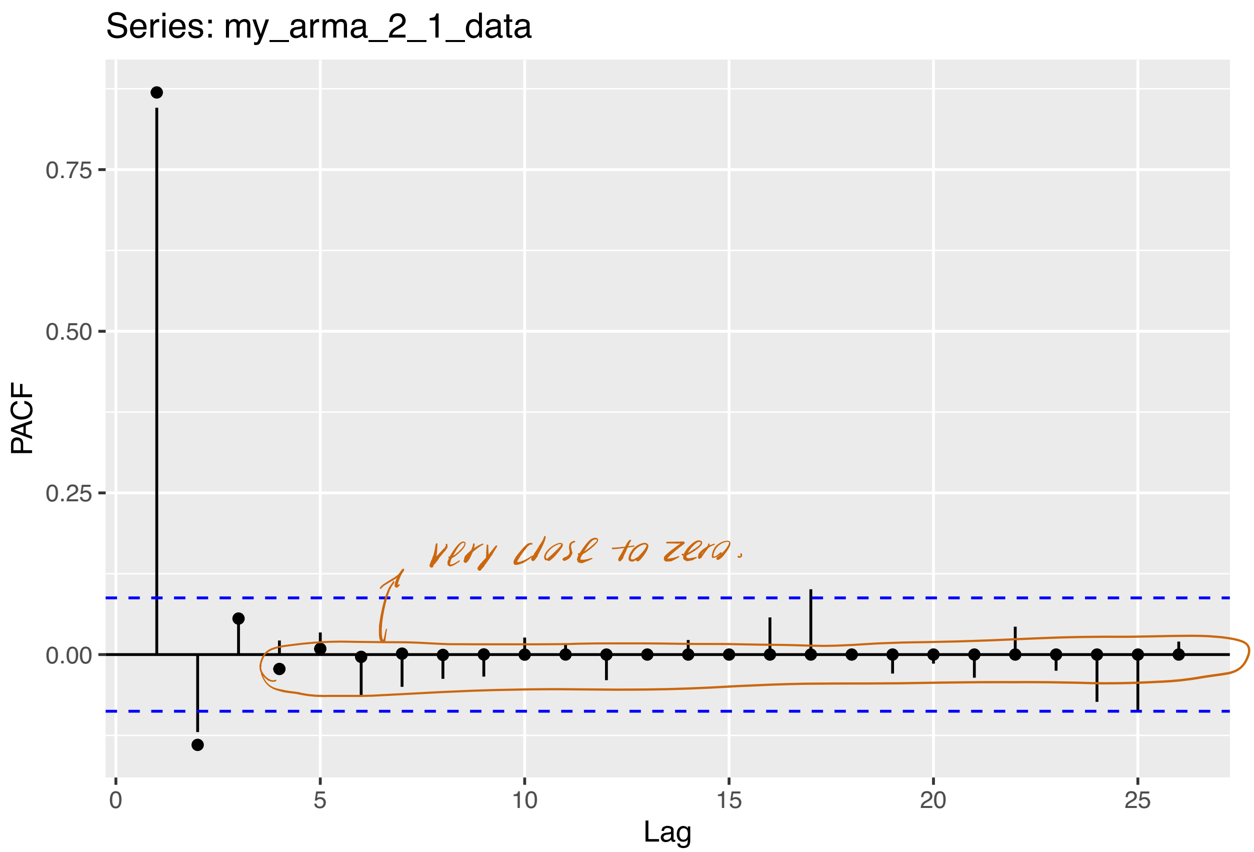
sample

```
ggAcf(my_arma_2_1_data) +
  geom_point(aes(y=ARMAacf(ar=true_ar_coef,ma=true_ma_coef,lag.max=26)[-1]))
```

theratical




```
ggPacf(my_arma_2_1_data) +
  geom_point(aes(y=ARMAacf(ar=true_ar_coef,ma=true_ma_coef,
                           lag.max=26,pacf=TRUE)))
```



Functions for estimating coefficients

AR estimation

- First let's generate some true AR(4) data:

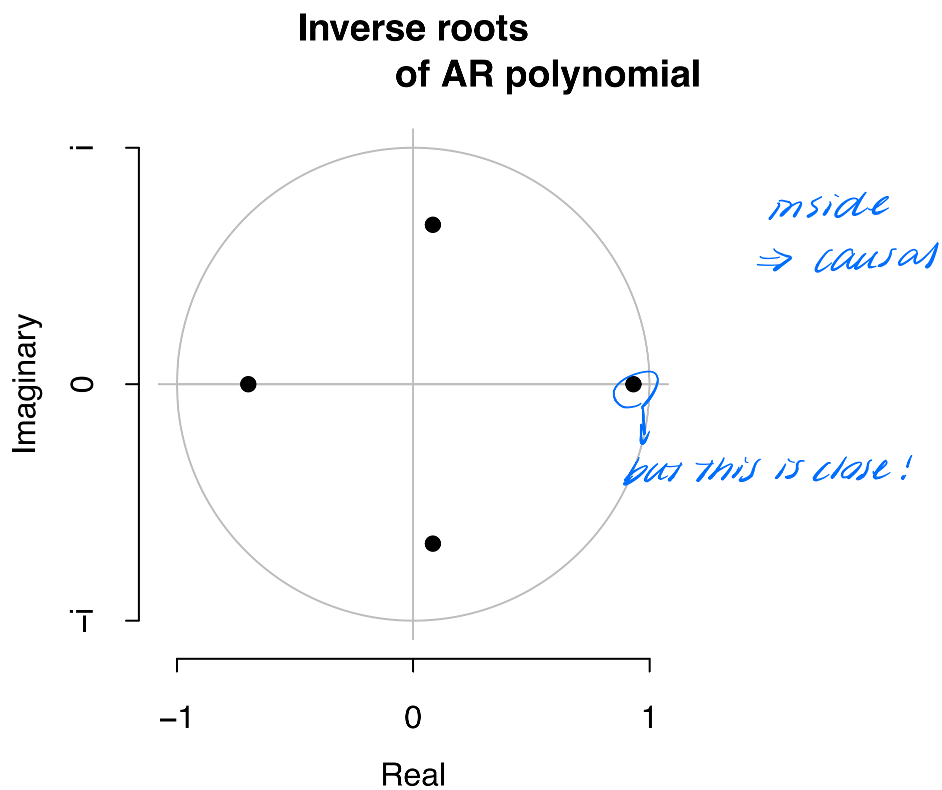
```
true_ar_coef_2=c(0.4,0.15,0,0.3)
my_AR4_model = list(ar=true_ar_coef_2,ma=NULL)
```

- Check the roots to make sure it is stationary

```
ar_roots<-polyroot(c(1,-my_AR4_model$ar))
ar_roots
```

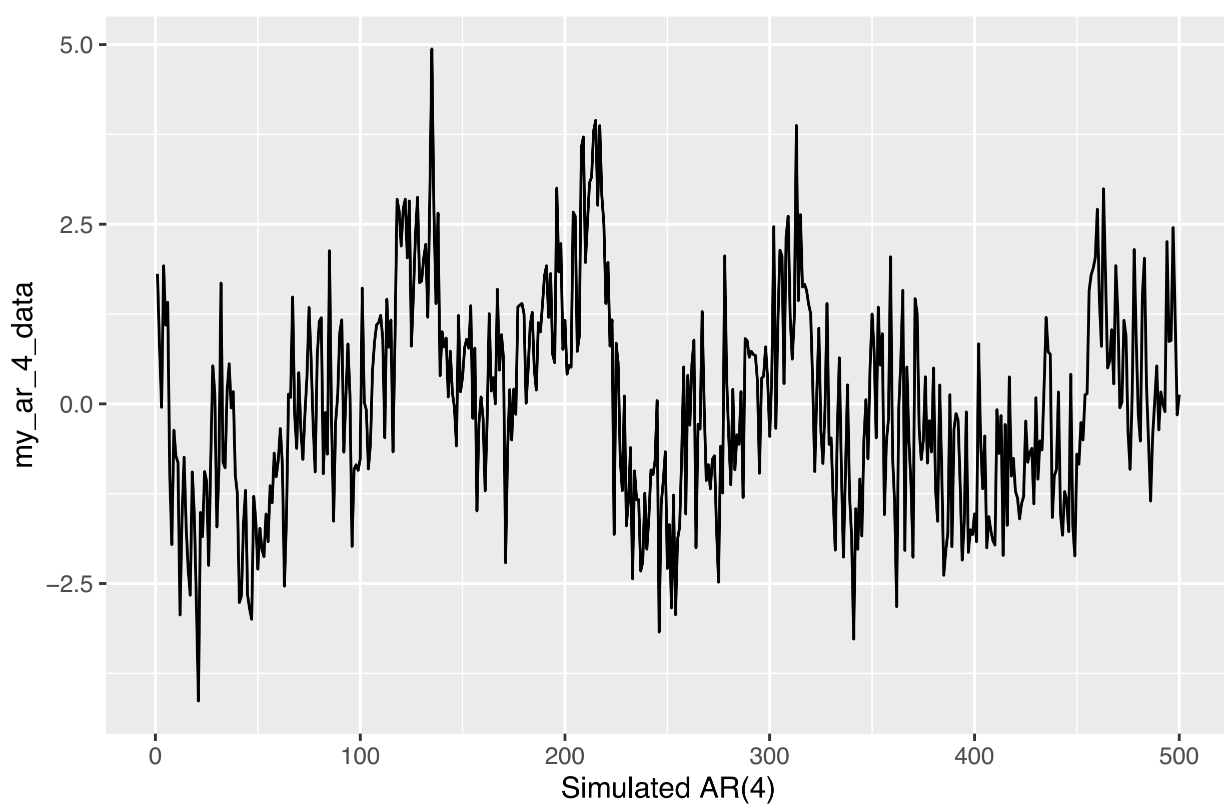
```
[1]  1.073196-0.000000i -1.433037+0.000000i  0.179920-1.461179i
[4]  0.179920+1.461179i
```

```
forecast:::plot.armaroots(structure(list(roots=ar_roots),
                          class = "armaroots"),xlab="Real", ylab="Imaginary",main="Inverse roots
                          of AR polynomial")
```

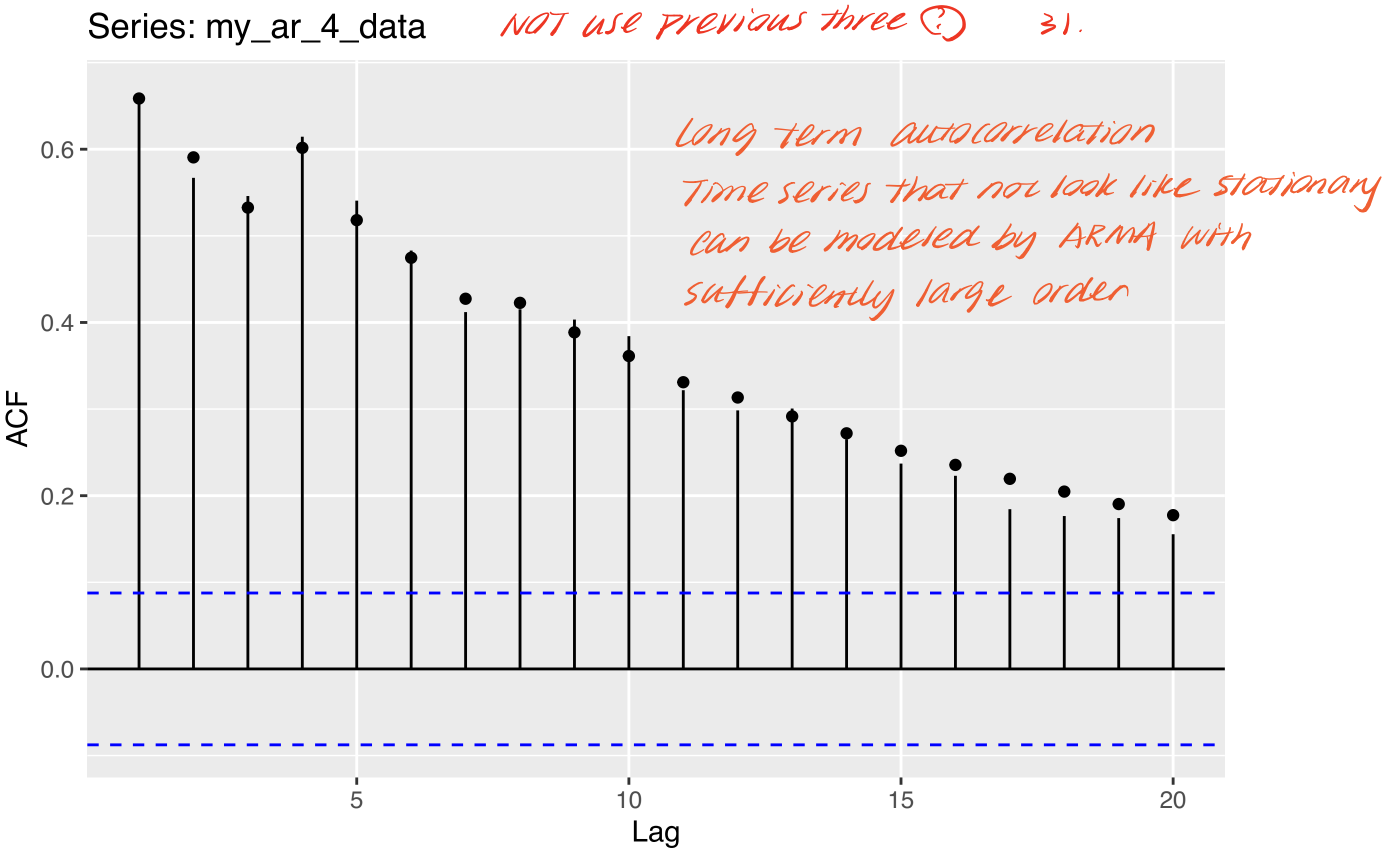


- Now generate some data and compare sample ACF and PACF to truth

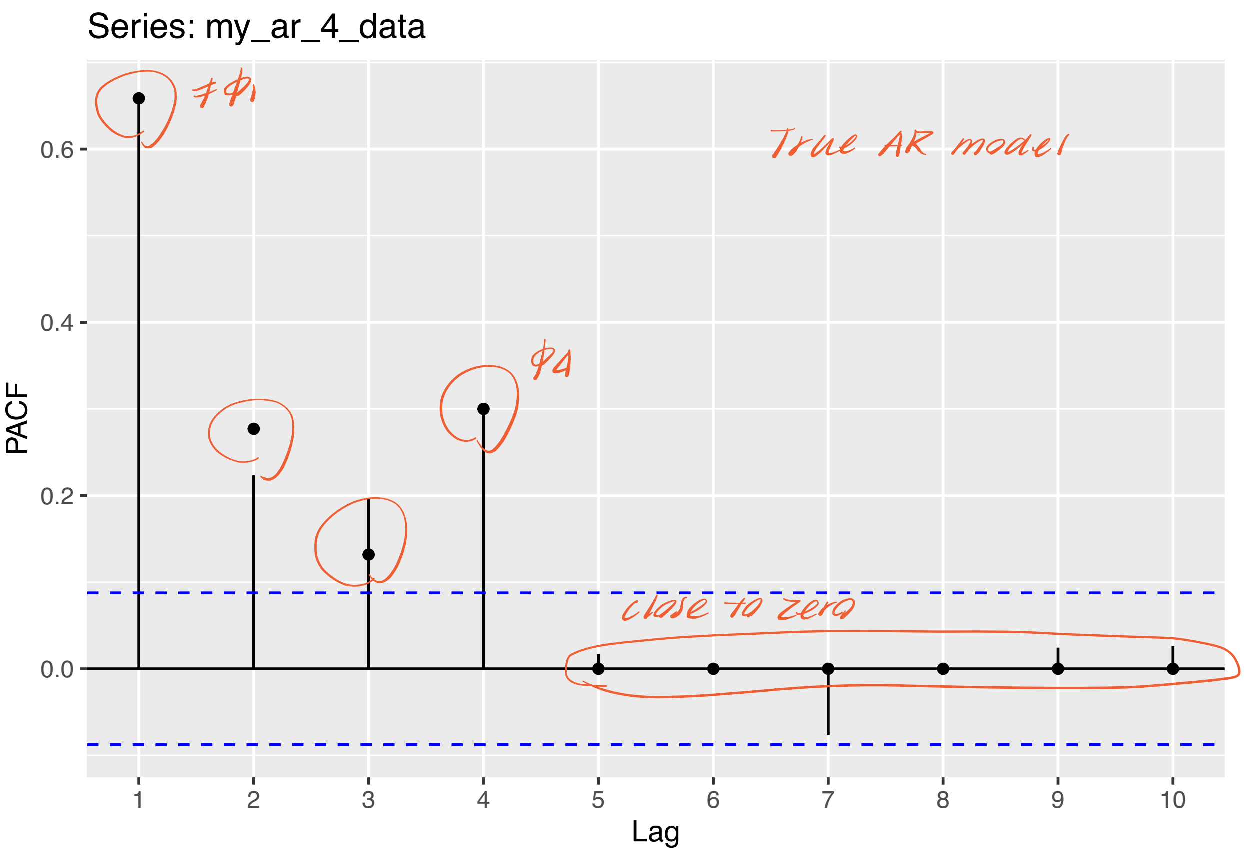
```
my_ar_4_data = arima.sim(my_AR4_model,n=500)
autoplot(my_ar_4_data) + xlab("Simulated AR(4)")
```



```
ggAcf(my_ar_4_data,lag.max=20) +
  geom_point(aes(y=c(ARMAacf(ar=true_ar_coef_2,ma=0,lag.max=20)[-1])))
```




```
ggPacf(my_ar_4_data,lag.max=10) +
  geom_point(aes(y=c(ARMAacf(ar=true_ar_coef_2,pacf=TRUE),rep(0,6))))
```



Estimation via Yule-Walker, Burg, and MLE

```
my_ar_mod_4_yw<-ar(my_ar_4_data,method="yw",aic=FALSE,order.max=10)
my_ar_mod_4_burg<-ar(my_ar_4_data,method="burg",aic=FALSE,order.max=10)
my_ar_mod_4_<-ar(my_ar_4_data,method="burg",aic=FALSE,order.max=10)
                                        mle

my_ar_map<-map_df(
  c("yw","burg","mle"),
  function(x){
    myfit=ar(my_ar_4_data,method=x,aic=FALSE,order.max=10)
    tibble(method=x,coef_ind=1:10,truth=c(true_ar_coef_2,rep(0,6)),
           coef=myfit[["ar"]],se=sqrt(diag(myfit[["asy.var.coef"]])))}
  )

my_ar_map %>% filter(method=="yw") %>% kable(.)
```

Fit a order ten model.

method	coef_ind	truth	coef	se
yw	1	0.40	0.4064531	0.0452058
yw	2	0.15	0.0847056	0.0487957
yw	3	0.00	0.0744958	0.0489427
yw	4	0.30	0.3021620	0.0489233
yw	5	0.00	0.0127394	0.0507804
yw	6	0.00	0.0275170	0.0507804
yw	7	0.00	-0.0805052	0.0489233
yw	8	0.00	-0.0121743	0.0489427
yw	9	0.00	0.0136945	0.0487957
yw	10	0.00	0.0263943	0.0452058

large se

⇒ hard to estimate coef that is close to zero.

```
my_ar_map %>% filter(method=="burg") %>% kable(.)
```

method	coef_ind	truth	coef	se
burg	1	0.40	0.4055918	0.0445495
burg	2	0.15	0.0834481	0.0480873
burg	3	0.00	0.0710490	0.0482322
burg	4	0.30	0.3115129	0.0482130
burg	5	0.00	0.0139990	0.0500432
burg	6	0.00	0.0339335	0.0500432
burg	7	0.00	-0.0774369	0.0482130
burg	8	0.00	-0.0203386	0.0482322
burg	9	0.00	0.0136697	0.0480873
burg	10	0.00	0.0191291	0.0445495

similar

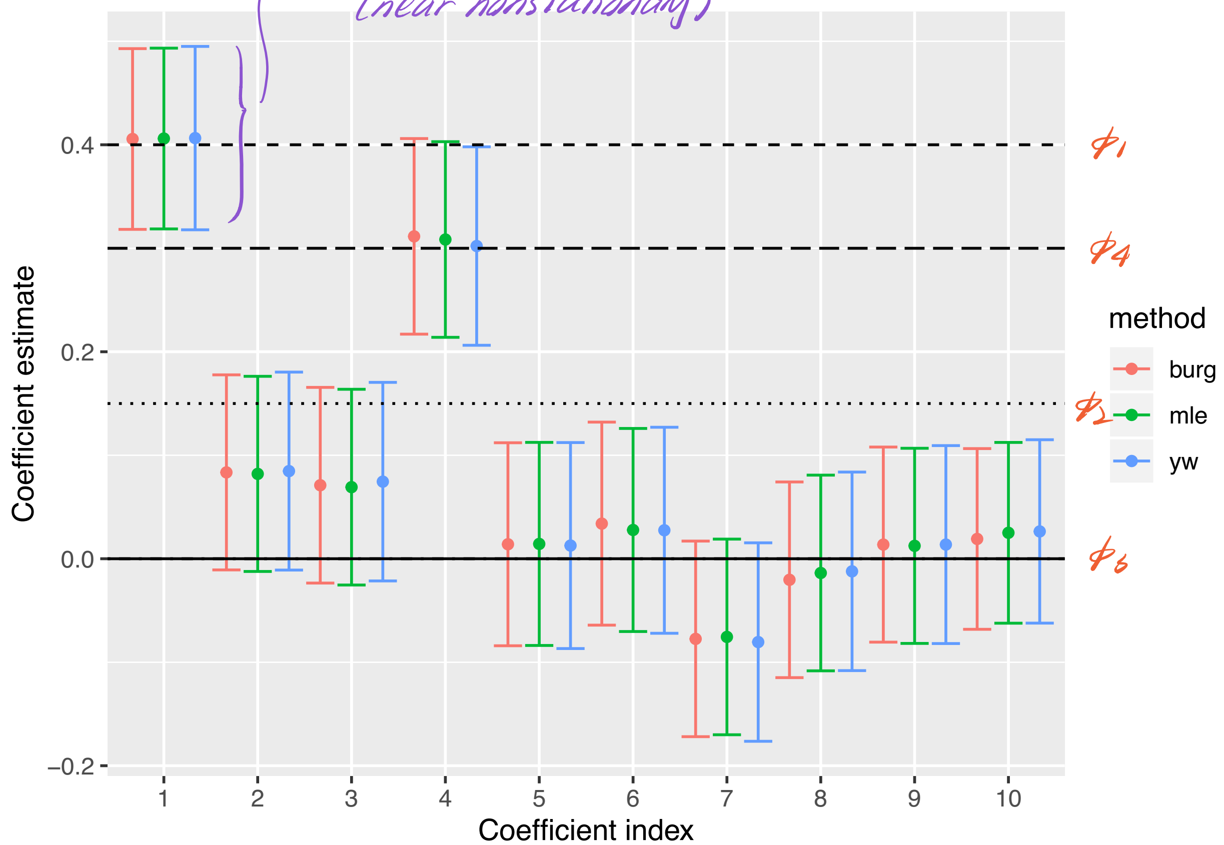
```
my_ar_map %>% filter(method=="mle") %>% kable(.)
```

method	coef_ind	truth	coef	se
mle	1	0.40	0.4060394	0.0445626
mle	2	0.15	0.0820030	0.0481015
mle	3	0.00	0.0691747	0.0482464
mle	4	0.30	0.3084719	0.0482272
mle	5	0.00	0.0143238	0.0500579
mle	6	0.00	0.0277853	0.0500579
mle	7	0.00	-0.0755489	0.0482272
mle	8	0.00	-0.0137561	0.0482464
mle	9	0.00	0.0124724	0.0481015
mle	10	0.00	0.0250573	0.0445626

similar

```
ggplot(my_ar_map,aes(x=factor(coef_ind),y=coef,colour=method,ymin=coef-1.96*se, ymax=coef+1.96*se)) +
  geom_errorbar(position=position_dodge(width=1)) + geom_point(position=position_dodge(width=1)) +
  xlab("Coefficient index") + ylab("Coefficient estimate") +
  geom_hline(yintercept=c(0,true_ar_coef_2),linetype=1:5)
```

CI width
① sample size
② complicated model
(near nonstationary)



Alternate method for estimating AR models with diagnostics

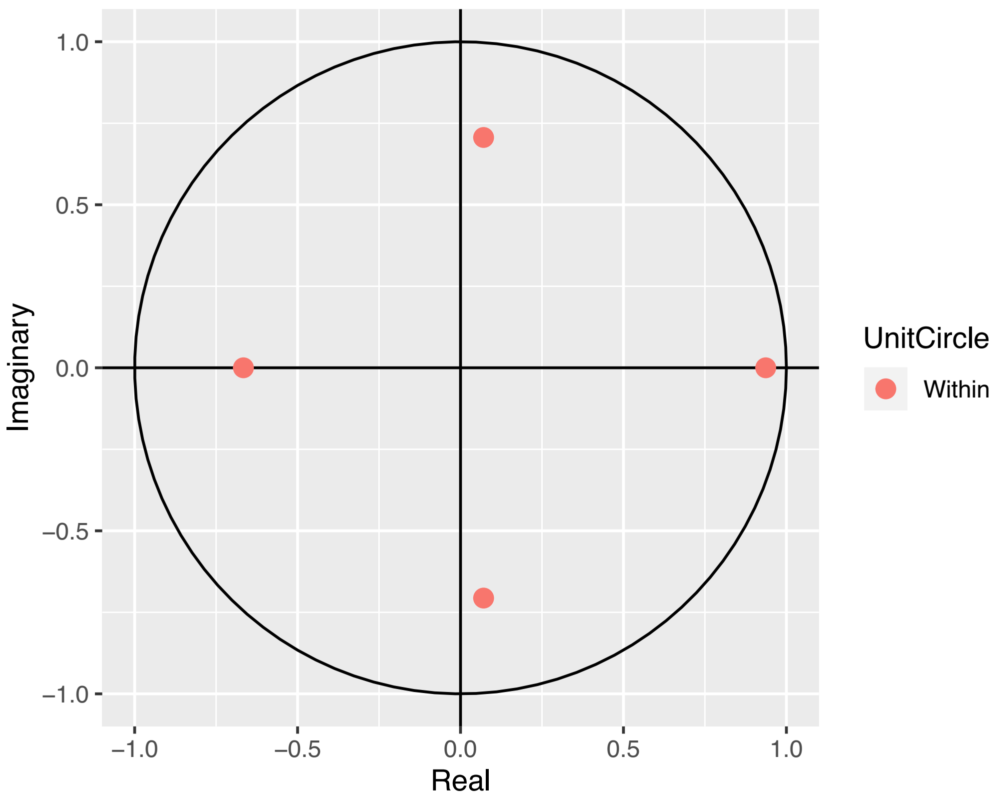
```
arima_mod_ar4<-Arima(my_ar_4_data,order=c(4,0,0))
summary(arima_mod_ar4)
```

```
Series: my_ar_4_data
ARIMA(4,0,0) with non-zero mean

Coefficients:
         ar1     ar2     ar3     ar4    mean
      0.4120  0.0817  0.0479  0.3144  0.0370
s.e.  0.0424  0.0462  0.0463  0.0426  0.2976

sigma^2 estimated as 0.9757:  log likelihood=-701.42
AIC=1414.84   AICc=1415.01   BIC=1440.13

Training set error measures:
                       ME      RMSE       MAE       MPE     MAPE      MASE
Training set -0.008055167 0.9828311 0.7737122 -19.79253 205.9686 0.8244087
                     ACF1
Training set -0.007352168
```

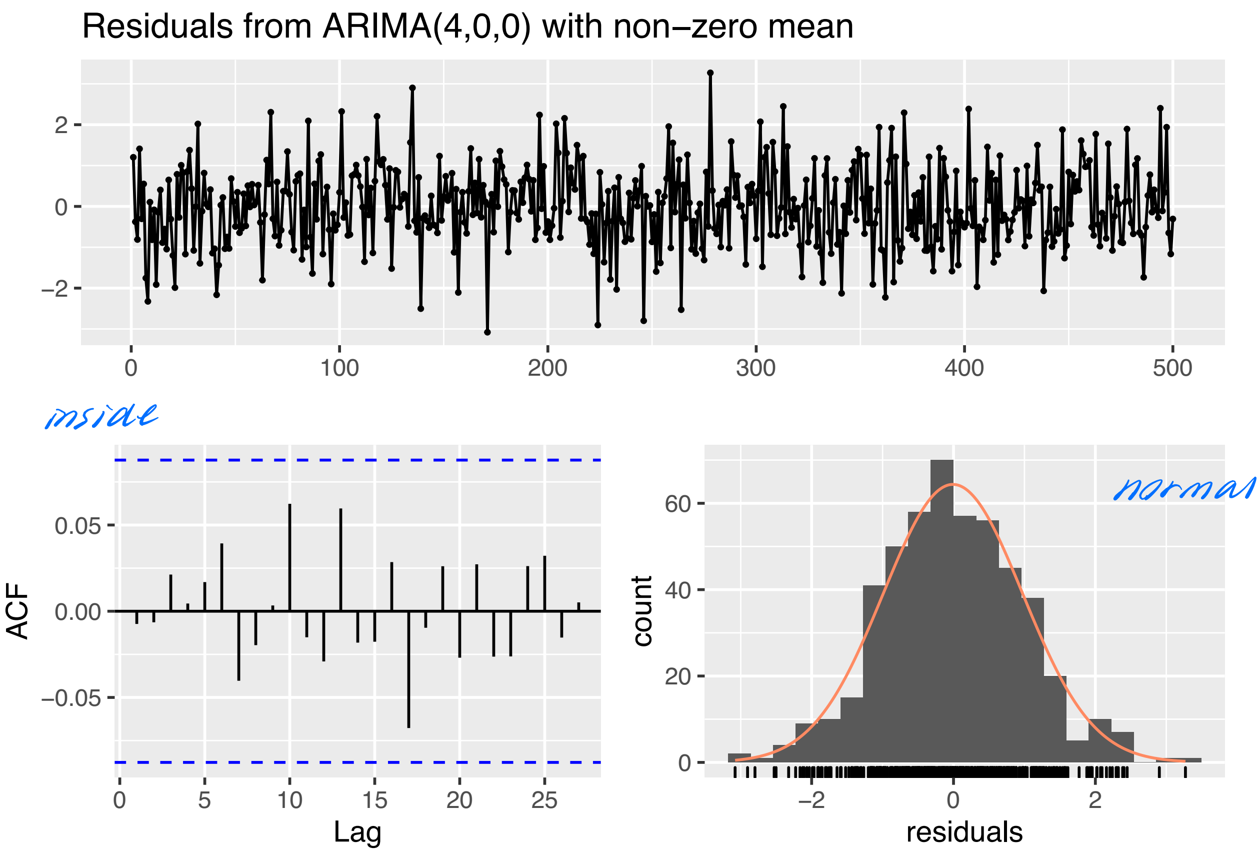
ARIMA

```
autoplot(arima_mod_ar4)
```

(estimated) Inverse AR roots




```
checkresiduals(arima_mod_ar4)
```



inside

normal

```
    Ljung-Box test
```

NOT reject WN

```
data:  Residuals from ARIMA(4,0,0) with non-zero mean
Q* = 4.2365, df = 5, p-value = 0.5159

Model df: 5.   Total lags used: 10
```

Estimating MA model via Innovations algorithm

- First simulate some MA data:

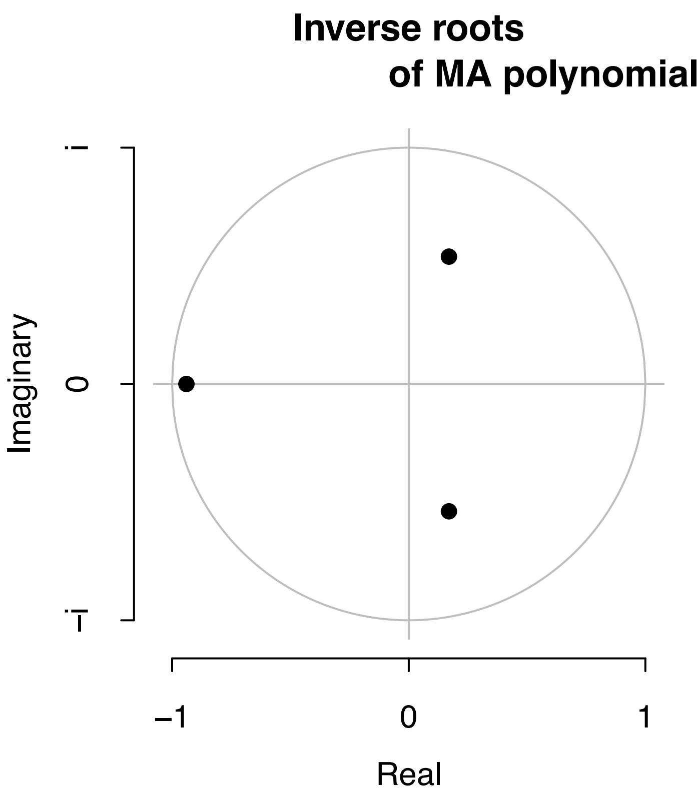
```
true_ma_coef_2=c(0.6, 0, 0.3)
my_MA3_model = list(ma=true_ma_coef_2,ar=NULL)
```

- Check the roots to make sure it is invertible

```
ma_roots<-polyroot(c(1,true_ma_coef_2))
ma_roots
```

```
[1]  0.532073+1.687988i -1.064145+0.000000i  0.532073-1.687988i
```

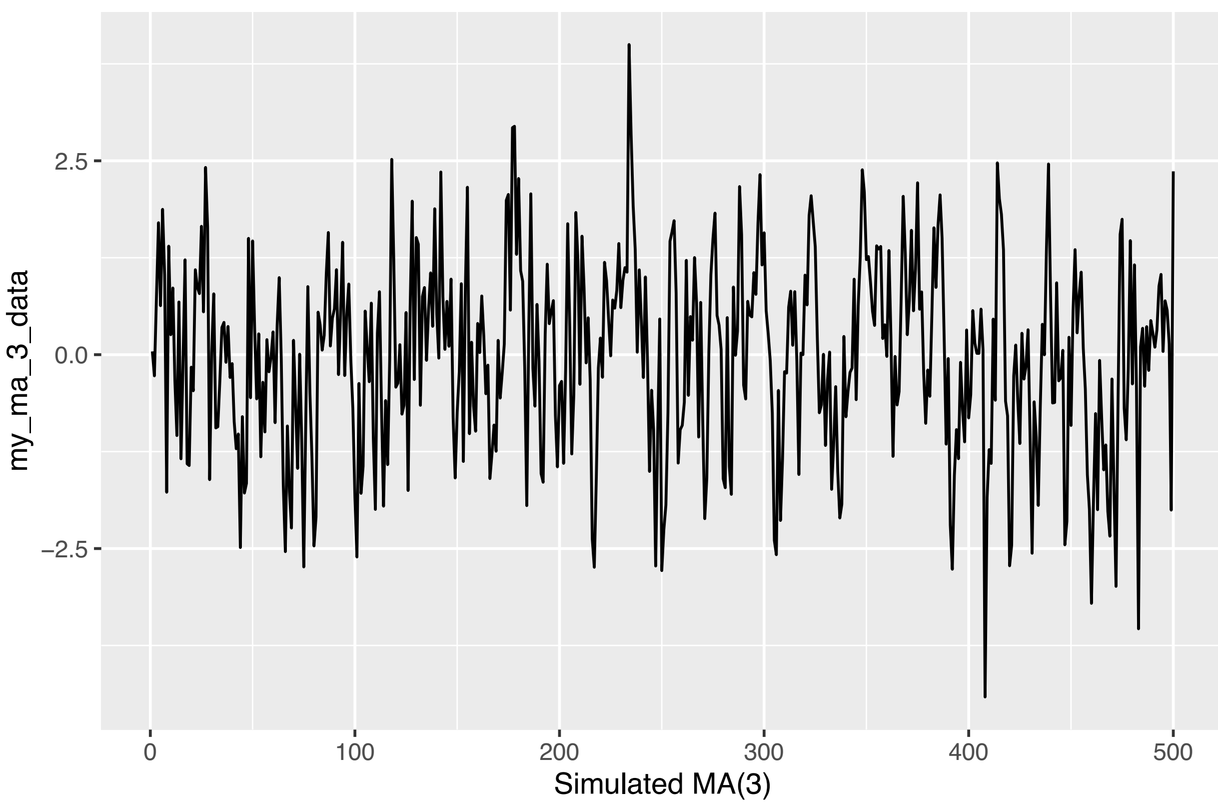
```
forecast:::plot.armaroots(structure(list(roots=ma_roots),
                          class = "armaroots"),xlab="Real", ylab="Imaginary",main="Inverse roots
                          of MA polynomial")
```



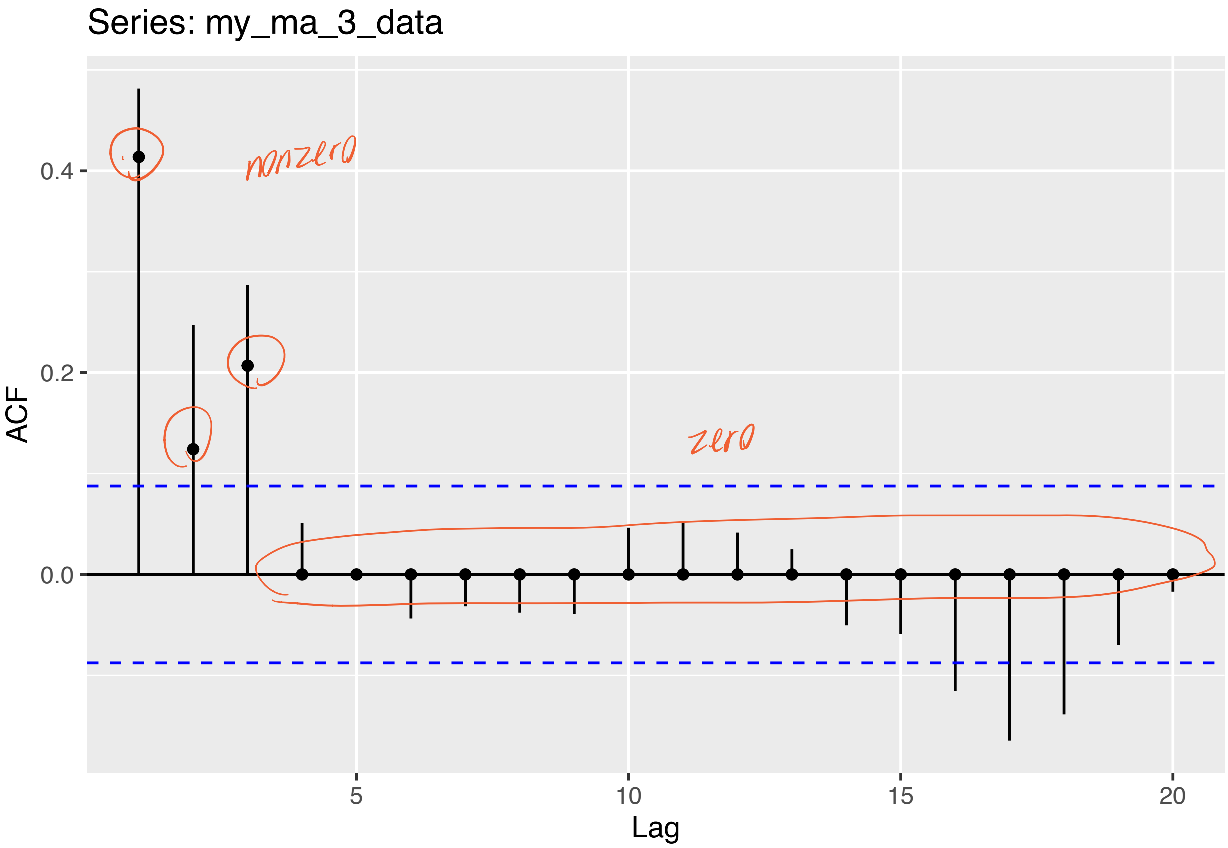
invertible

- Now generate some data and compare sample ACF and PACF to truth

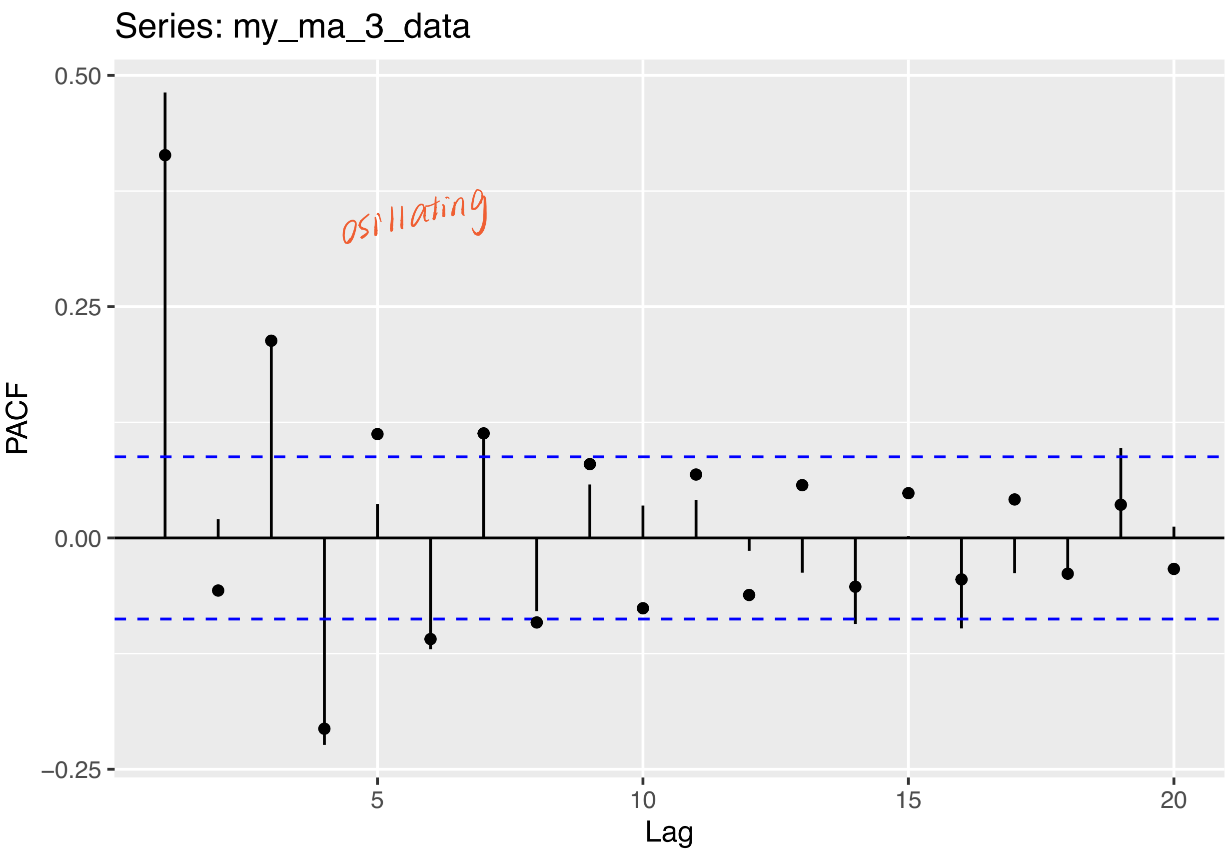
```
my_ma_3_data = arima.sim(my_MA3_model,n=500)
autoplot(my_ma_3_data) + xlab("Simulated MA(3)")
```



```r
ggAcf(my_ma_3_data,lag.max=20) +
  geom_point(aes(y=c(ARMAacf(ma=true_ma_coef_2,ar=0,lag.max=20)[-1])))
```




```r
ggPacf(my_ma_3_data,lag.max=20) +
  geom_point(aes(y=c(ARMAacf(ma=true_ma_coef_2,ar=0,pacf=TRUE,lag.max=20))))
```



- Now estimate MA coefficients from innovations algorithm

```
ia(my_ma_3_data,q=4,m=17)
```

$\theta_{17,1}$ ... $\theta_{17,4}$

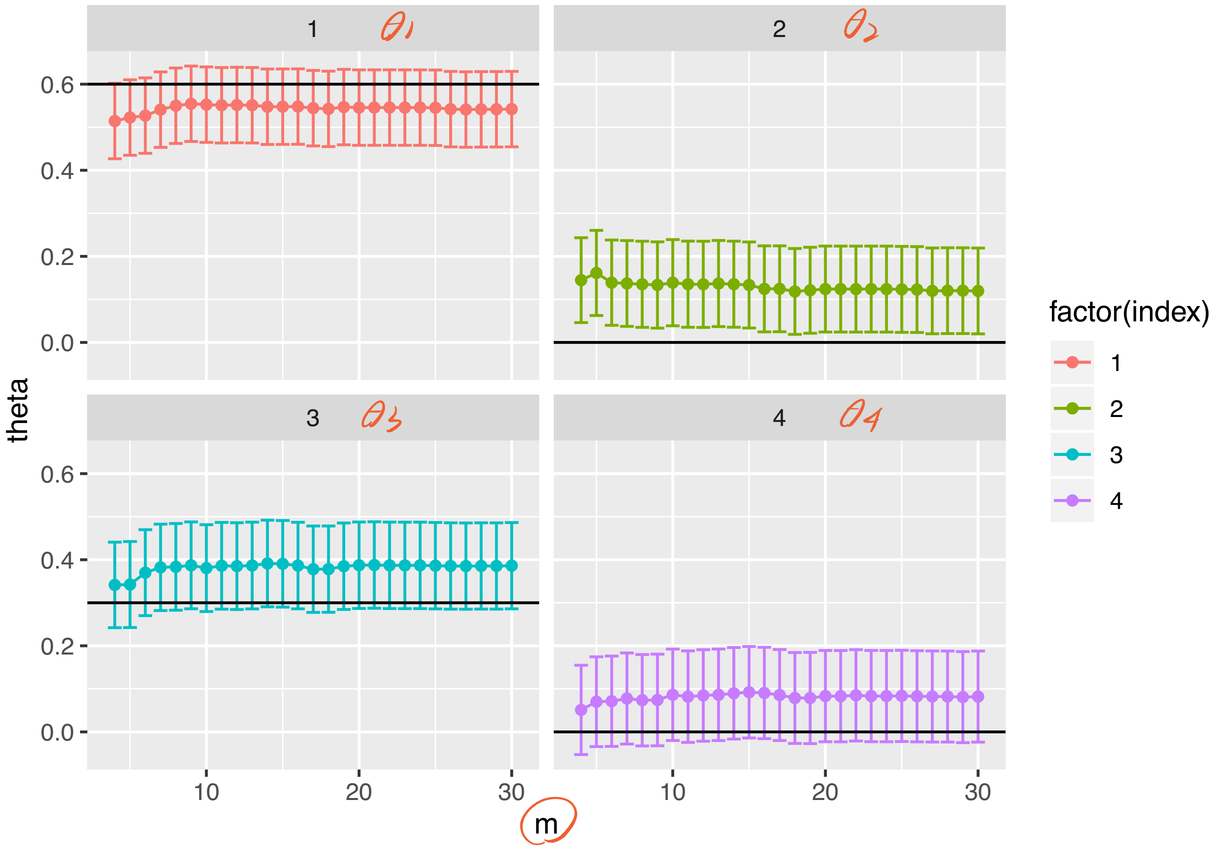
larger than 3

```
$phi
[1] 0

$theta
[1] 0.54422161 0.12462154 0.37800418 0.08567481

$sigma2
[1] 1.0003

$aicc
[1] 1430.003

$se.phi
[1] 0

$se.theta
[1] 0.04472136 0.05091517 0.05121929 0.05393690
```

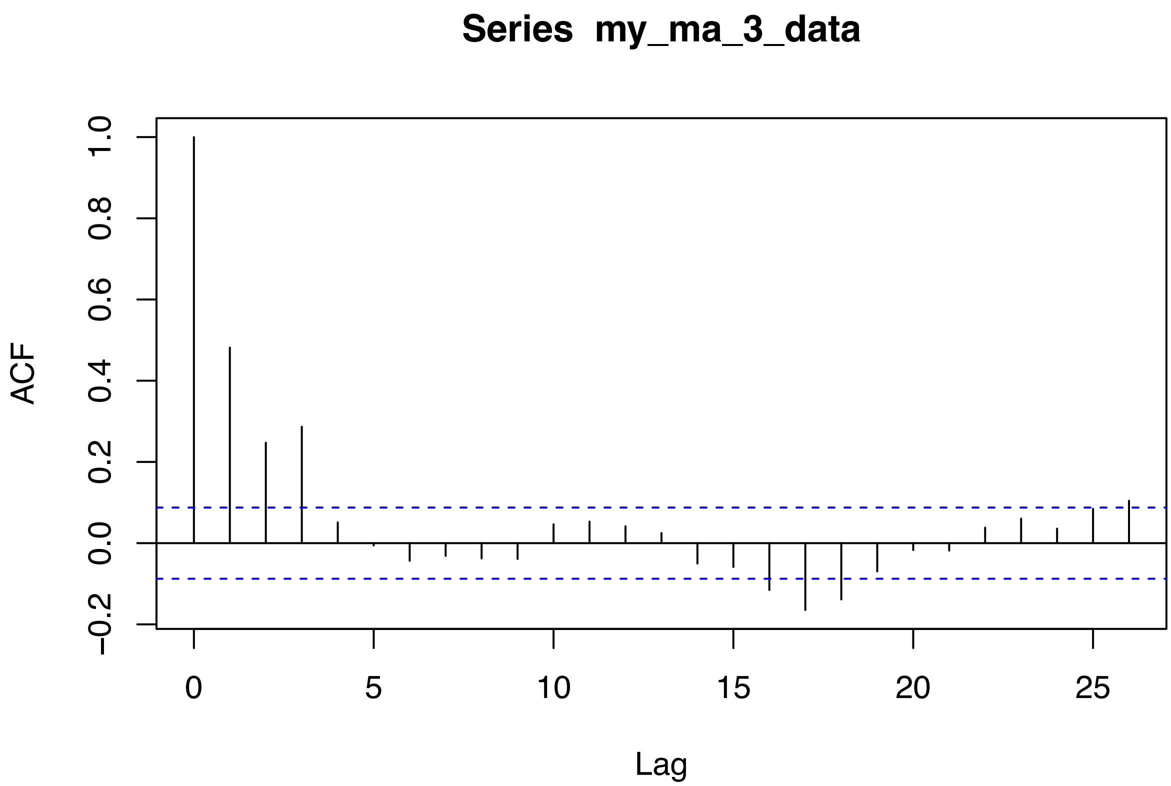
```
my_theta_paths<-map_df(4:30,~bind_cols(tibble(m=rep(.x,4),index=c(1:4),truth=c(true_ma_coef_2,0)),
                                        as_tibble(ia(my_ma_3_data,q=4,m=.x)[c("theta","se.theta")])))
```

```
ggplot(my_theta_paths,aes(x=m,y=theta,group=factor(index),colour=factor(index))) +
  geom_errorbar(aes(ymin=theta-1.96*se.theta,ymax=theta+1.96*se.theta)) +
  facet_wrap(~factor(index)) + geom_point() + geom_line() +
```

```r
  geom_hline(aes(yintercept=truth))
```




```r
acf_vals<-acf(my_ma_3_data)
```




```
acf_vals
```

```
Autocorrelations of series 'my_ma_3_data', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.482  0.247  0.287  0.051 -0.006 -0.044 -0.032 -0.038 -0.039  0.046
    11     12     13     14     15     16     17     18     19     20     21
 0.053  0.042  0.025 -0.050 -0.059 -0.115 -0.165 -0.139 -0.070 -0.017 -0.018
    22     23     24     25     26
 0.038  0.061  0.036  0.085  0.105
```

- Need more data (n) to be more accurate, not necessarily larger m

```
my_ma_3_data_big = arima.sim(my_MA3_model,n=1000)
```

increase n

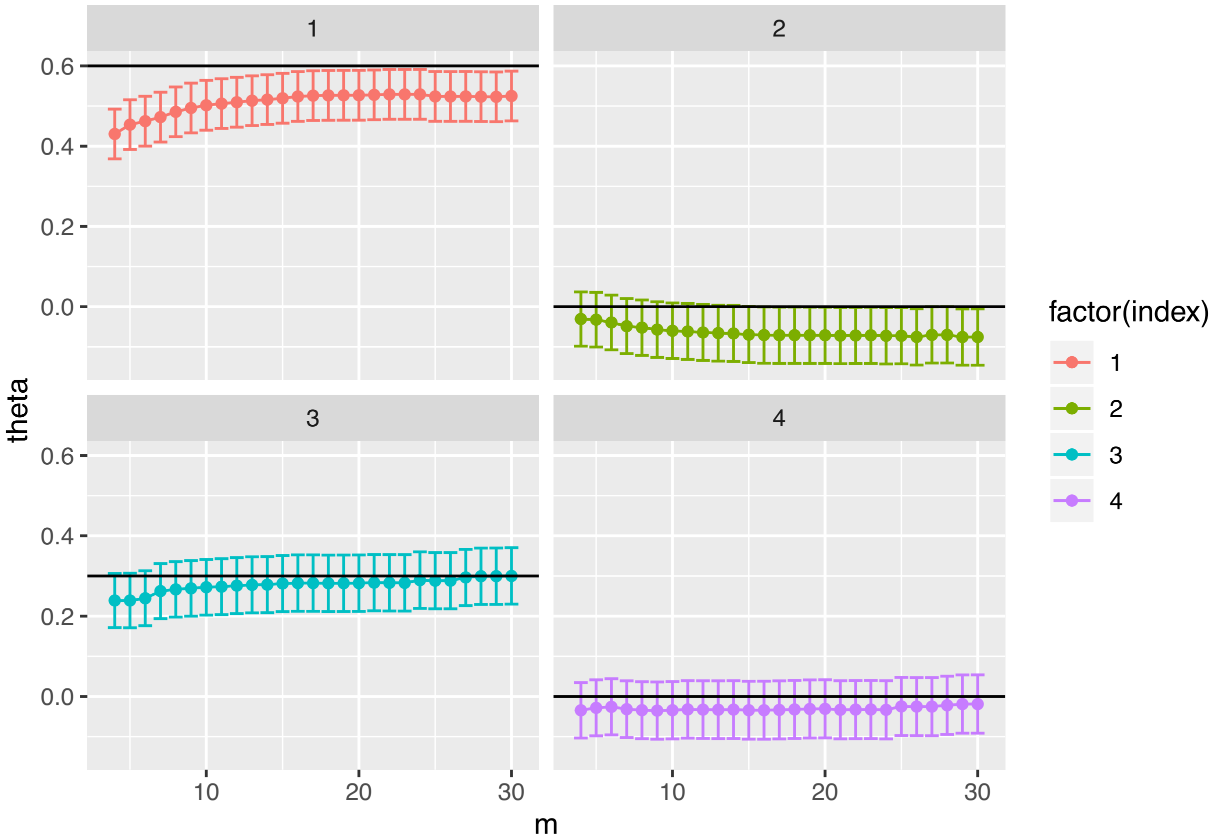
```
ia(my_ma_3_data_big,q=4,m=17)
```

```
$phi
[1] 0

$theta
[1]  0.52581663 -0.07084455  0.28234489 -0.03352451

$sigma2
[1] 0.9363118

$aicc
[1] 2783.702

$se.phi
```

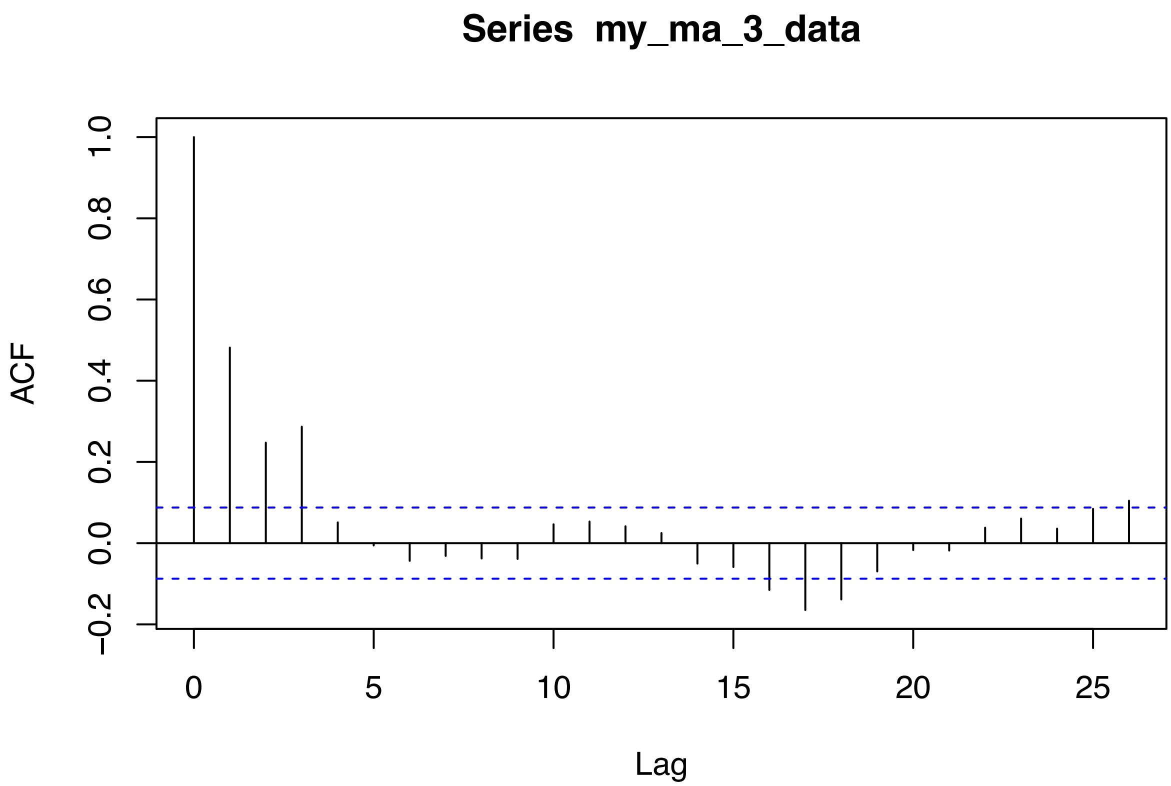
```
[1] 0

$se.theta
[1] 0.03162278 0.03572790 0.03579807 0.03689472
```

```
my_theta_paths<-map_df(4:30,~bind_cols(tibble(m=rep(.x,4),index=c(1:4),truth=c(true_ma_coef_2,0)),
                                        as_tibble(ia(my_ma_3_data_big,q=4,m=.x)[c("theta","se.theta")])))
```

```
ggplot(my_theta_paths,aes(x=m,y=theta,group=factor(index),colour=factor(index))) +
  geom_errorbar(aes(ymin=theta-1.96*se.theta,ymax=theta+1.96*se.theta)) +
  facet_wrap(~factor(index)) + geom_point() + geom_line() +
  geom_hline(aes(yintercept=truth))
```




```
acf_vals<-acf(my_ma_3_data)
```




```
acf_vals
```

```
Autocorrelations of series 'my_ma_3_data', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.482  0.247  0.287  0.051 -0.006 -0.044 -0.032 -0.038 -0.039  0.046
    11     12     13     14     15     16     17     18     19     20     21
 0.053  0.042  0.025 -0.050 -0.059 -0.115 -0.165 -0.139 -0.070 -0.017 -0.018
    22     23     24     25     26
 0.038  0.061  0.036  0.085  0.105
```

- Compare these to the results using the MLE:

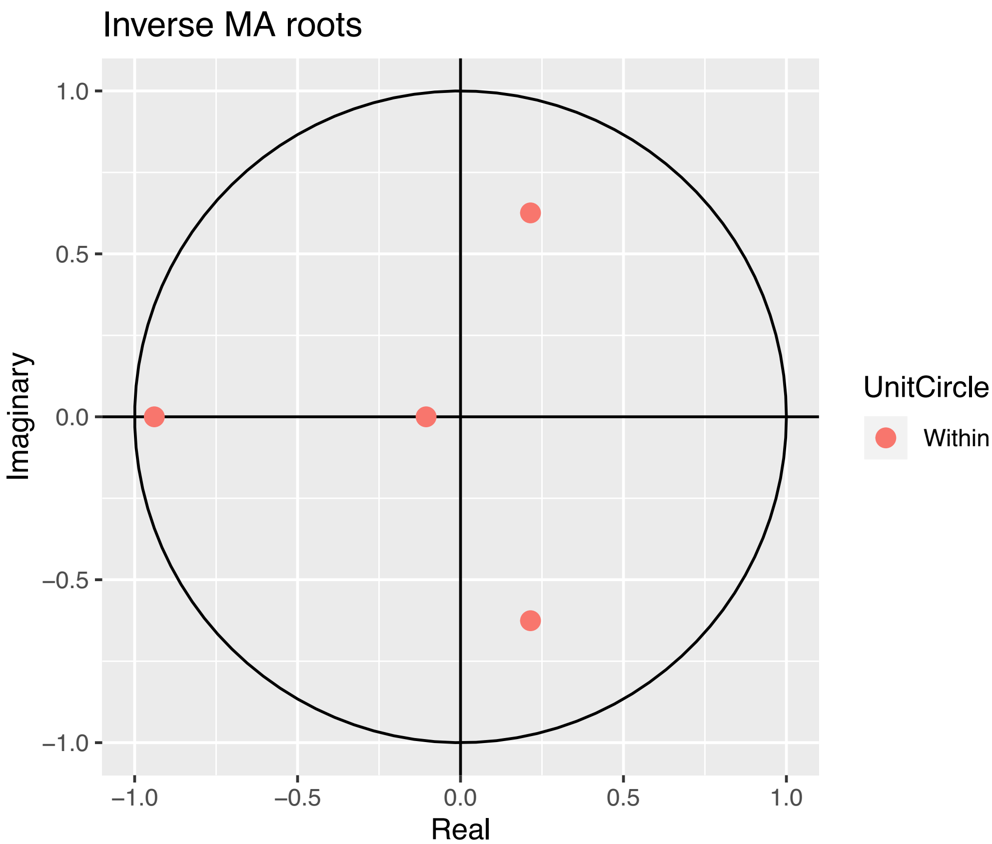
```
arima_mod_ma3<-Arima(my_ma_3_data,order=c(0,0,4))
summary(arima_mod_ma3)
```

```
Series: my_ma_3_data
ARIMA(0,0,4) with non-zero mean

Coefficients:
         ma1     ma2     ma3     ma4     mean
      0.6158  0.0880  0.4149  0.0434  -0.0218
s.e.  0.0447  0.0494  0.0471  0.0440   0.0951

sigma^2 estimated as 0.9803:  log likelihood=-702.79
AIC=1417.59   AICc=1417.76   BIC=1442.88

Training set error measures:
                        ME      RMSE       MAE      MPE     MAPE      MASE
Training set -0.0002456246 0.9851384 0.7765872 229.0824 326.0979 0.7659998
```

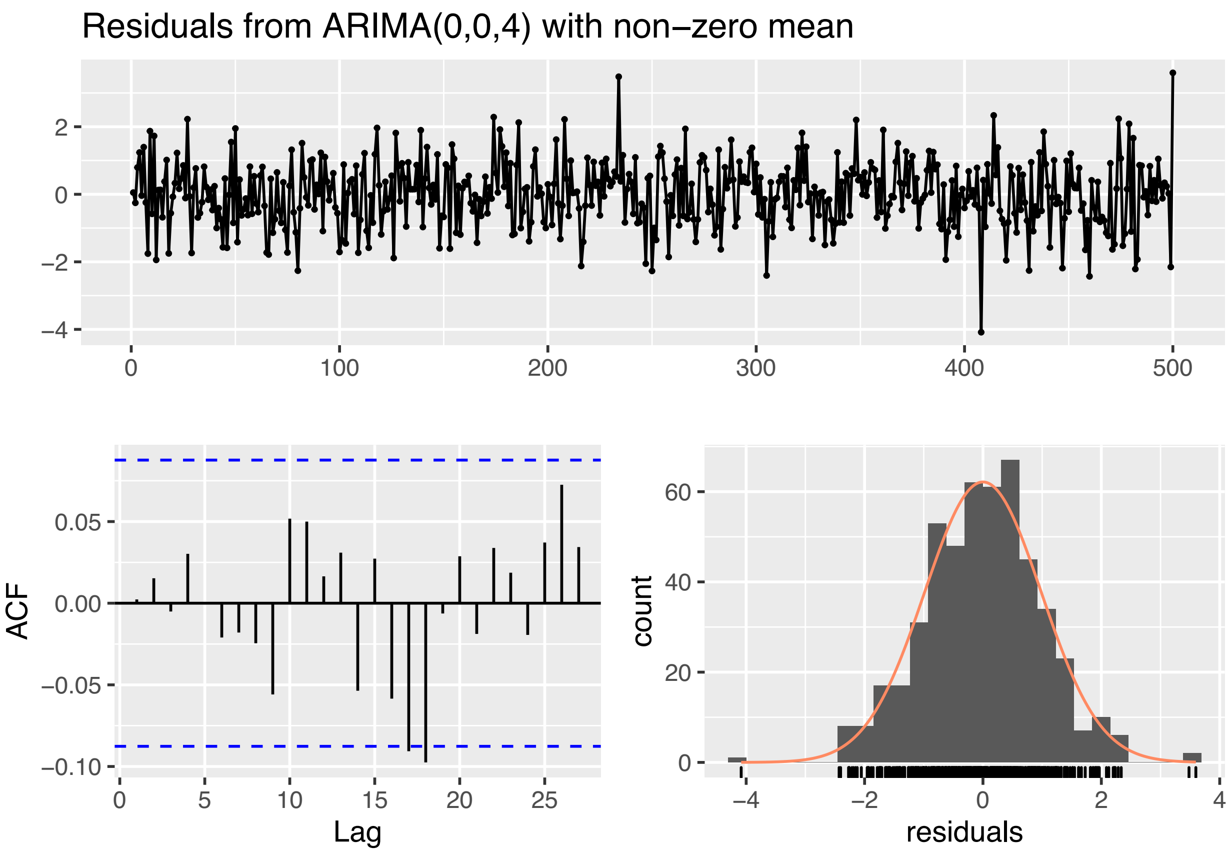
close to true value

```
                    ACF1
Training set 0.002313249
```

```r
autoplot(arima_mod_ma3)
```




```r
checkresiduals(arima_mod_ma3)
```



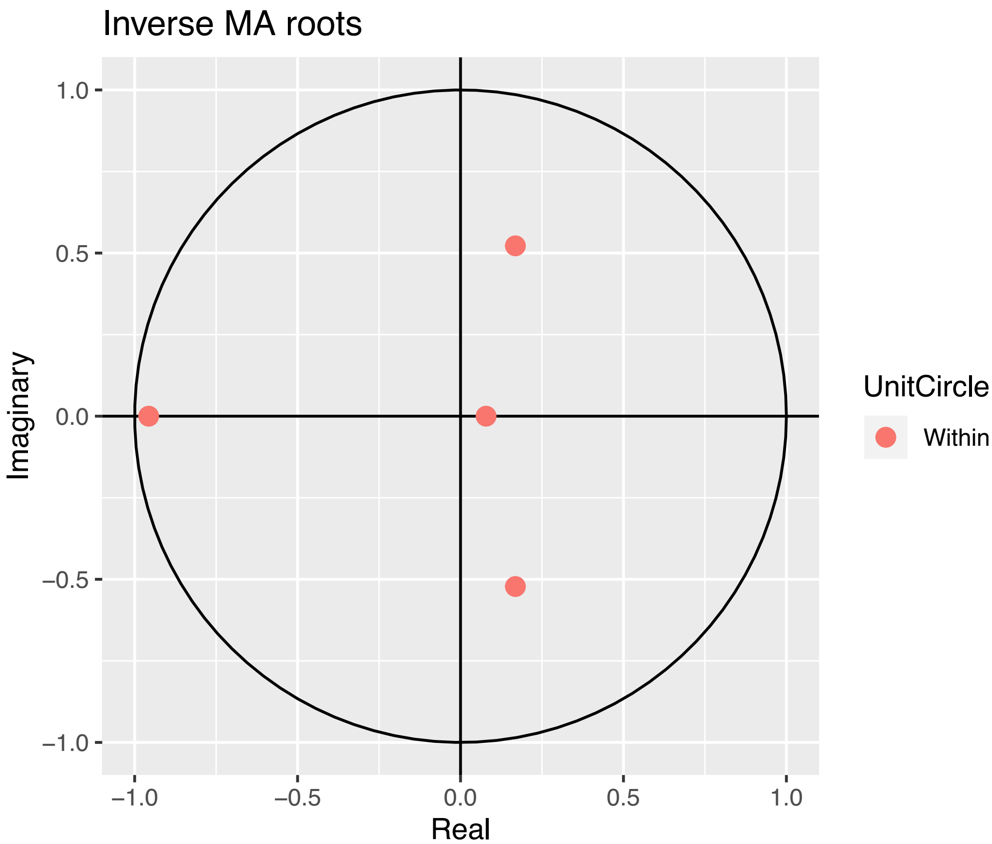
WN residuals.

```
    Ljung-Box test

data:  Residuals from ARIMA(0,0,4) with non-zero mean
Q* = 4.2611, df = 5, p-value = 0.5125

Model df: 5.   Total lags used: 10
```

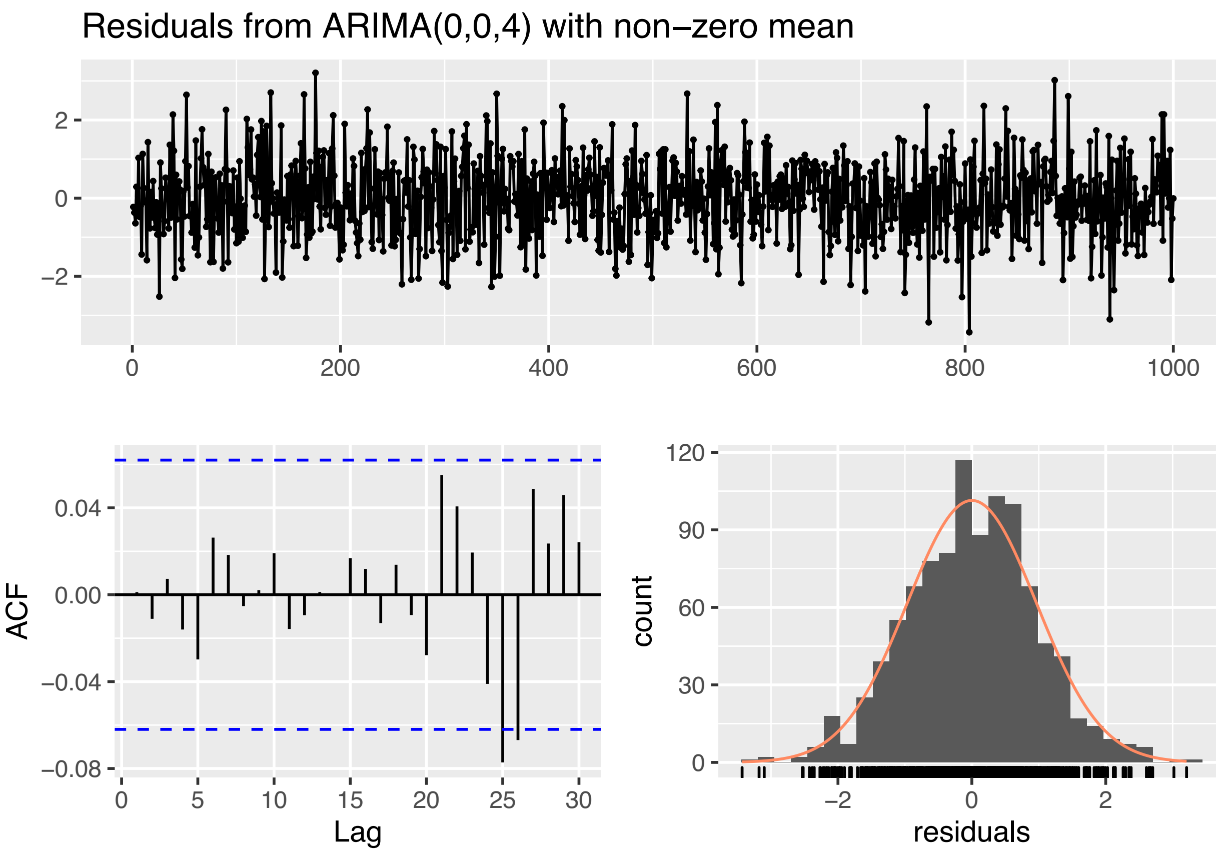
```r
arima_mod_ma3_big<-Arima(my_ma_3_data_big,order=c(0,0,4))
summary(arima_mod_ma3_big)
```

```
Series: my_ma_3_data_big
ARIMA(0,0,4) with non-zero mean

Coefficients:
         ma1      ma2     ma3      ma4     mean
      0.5416  -0.0700  0.2897  -0.0226  -0.0379
s.e.  0.0320   0.0356  0.0351   0.0319   0.0531

sigma^2 estimated as 0.9403:  log likelihood=-1386.5
AIC=2785   AICc=2785.09   BIC=2814.45

Training set error measures:
                       ME      RMSE       MAE      MPE     MAPE      MASE
Training set 0.0003526257 0.9672585 0.7649265 403.2864 637.0501 0.7351561
                    ACF1
Training set 0.001163003
```

```r
autoplot(arima_mod_ma3_big)
```




```r
checkresiduals(arima_mod_ma3_big)
```




```
    Ljung-Box test

data:  Residuals from ARIMA(0,0,4) with non-zero mean
Q* = 2.7655, df = 5, p-value = 0.7361

Model df: 5.   Total lags used: 10
```

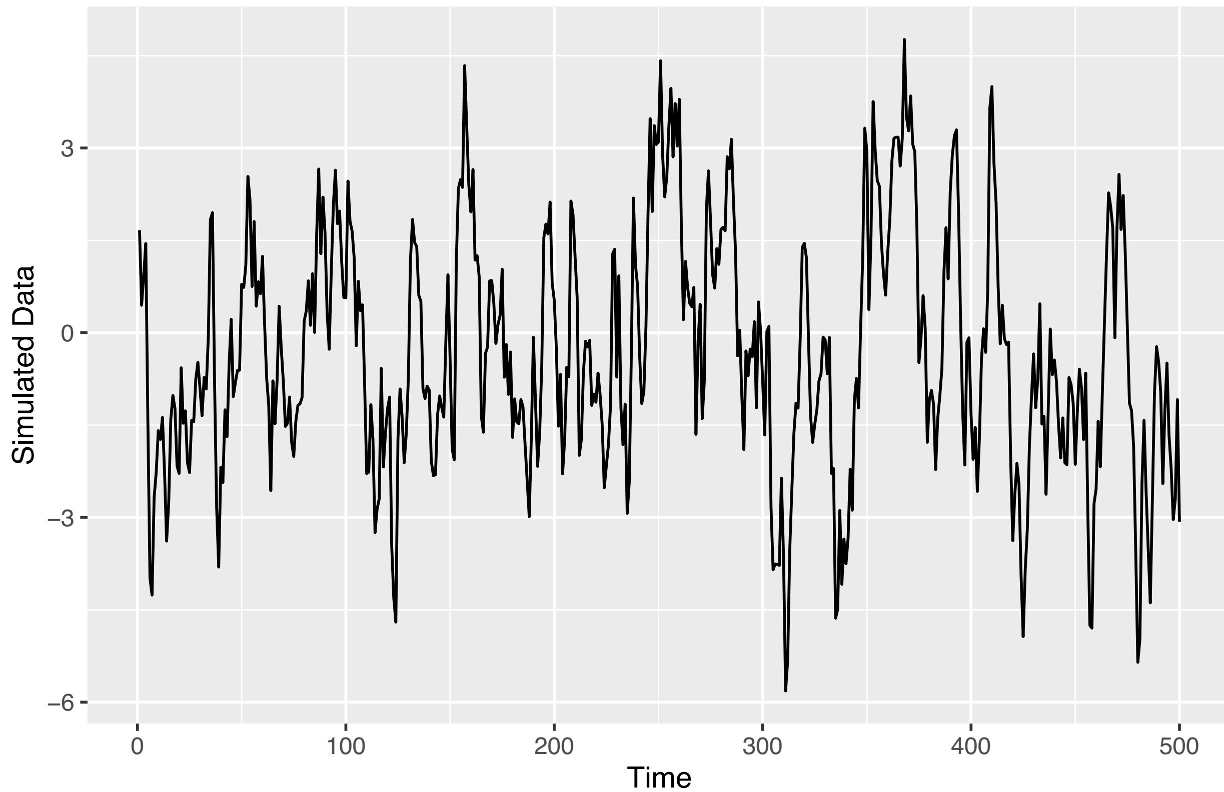
ARMA(p,q) via Hannen Rissanen

- Let's go back to our ARMA(2,1) dataset:

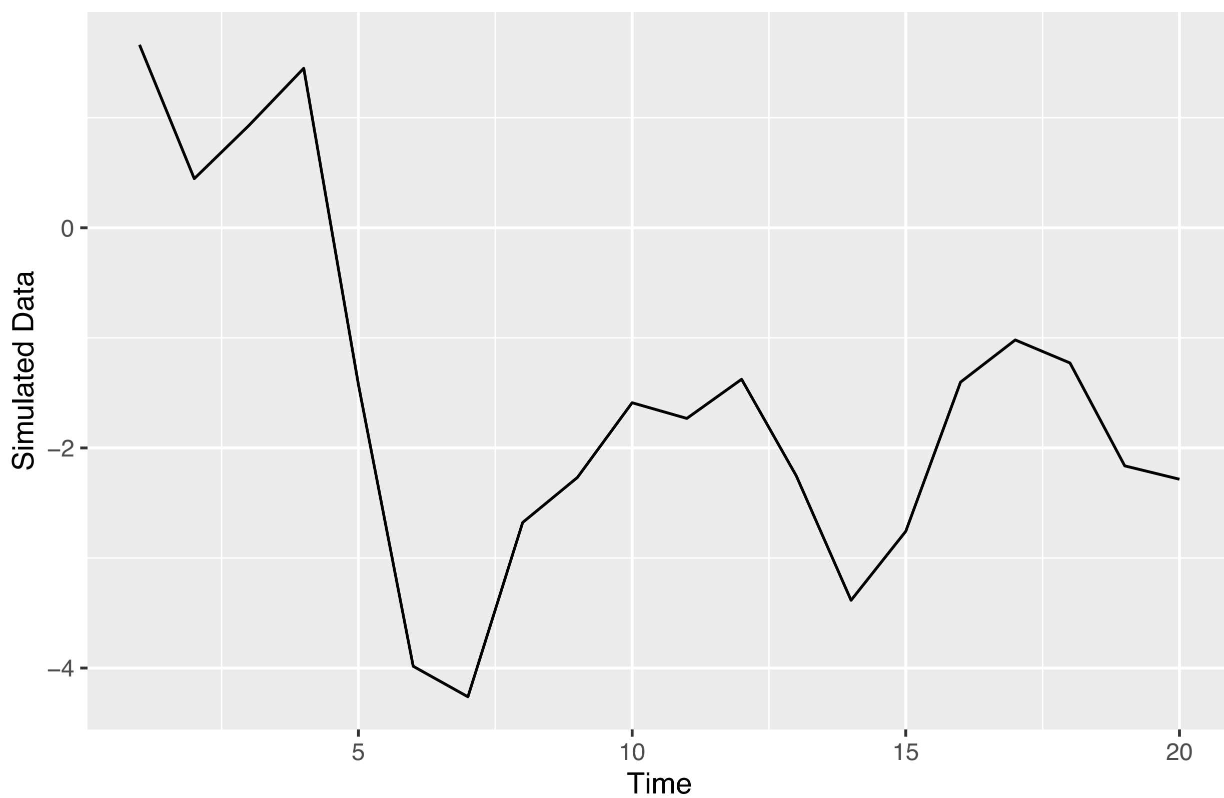
```
autoplot(my_arma_2_1_data) + ylab("Simulated Data")
```

true AR 0.6 0.2

true MA 0.4




```r
autoplot(window(my_arma_2_1_data,end=20)) + ylab("Simulated Data")
```



- We can use Hannan-Rissanen to estimate the ARMA coefficients

```r
hanr_arma_2_1<-hannan(my_arma_2_1_data - mean(my_arma_2_1_data),p=2,q=1)
hanr_arma_2_1
```

```
$phi
[1]  0.9883515 -0.1513998

$theta
[1] -0.03545169

$sigma2
[1] 1.057283

$aicc
[1] 1456.182

$se.phi
[1] 0.2058250 0.1758256

$se.theta
[1] 0.211752
```

- We can then compare this to the MLE:

```r
arima_mod_arma_2_1<-Arima(my_arma_2_1_data,order=c(2,0,1))
summary(arima_mod_arma_2_1)
```

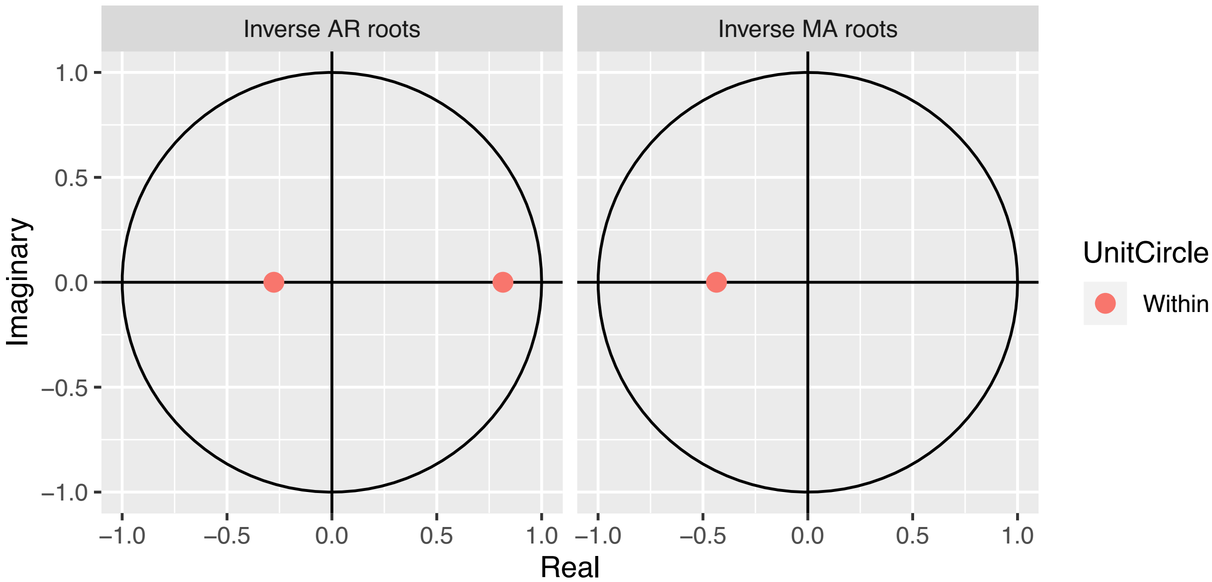
p q

close to true value

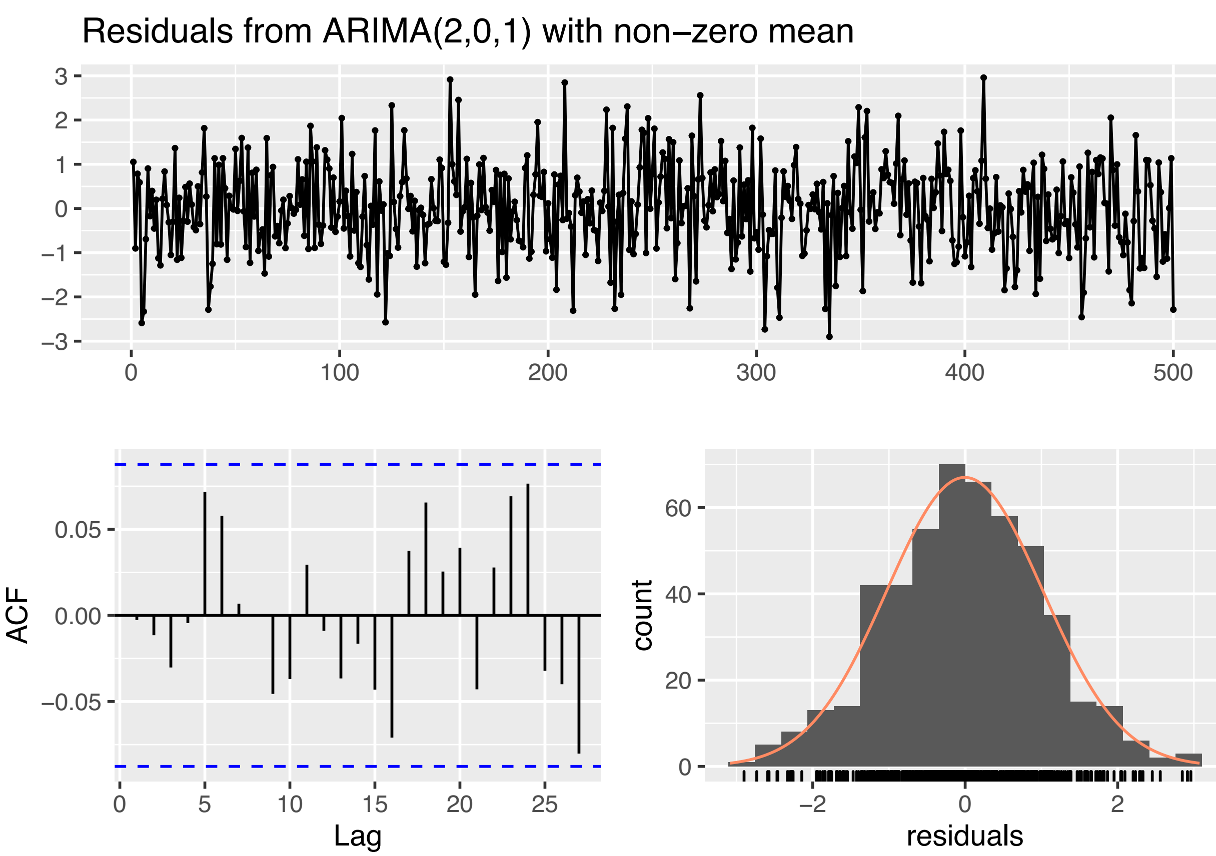
0.6 0.2 0.4

```
Series: my_arma_2_1_data
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1     ar2     ma1     mean
      0.5389  0.2257  0.4355  -0.3547
s.e.  0.2575  0.2246  0.2447   0.2771

sigma^2 estimated as 1.06:  log likelihood=-722.59
AIC=1455.17   AICc=1455.29   BIC=1476.24

Training set error measures:
                       ME     RMSE       MAE       MPE     MAPE      MASE
Training set -0.004817993 1.025217 0.8120458 -6.175719 165.1011 0.9410304
                     ACF1
Training set -0.002660299
```

```r
autoplot(arima_mod_arma_2_1)
```




```
checkresiduals(arima_mod_arma_2_1)
```



WN noise

```
    Ljung-Box test

data:  Residuals from ARIMA(2,0,1) with non-zero mean
Q* = 6.6368, df = 6, p-value = 0.3557

Model df: 4.   Total lags used: 10
```

```
tibble(coef=c("phi[1]","phi[2]","ma[1]"), Truth=c(true_ar_coef,true_ma_coef),
       HR = round(c(hanr_arma_2_1$phi, hanr_arma_2_1$theta),2),
       seHR = round(c(hanr_arma_2_1$se.phi, hanr_arma_2_1$se.theta),2),
       MLE = round(coef(arima_mod_arma_2_1)[-4],2),
       seMLE = round(sqrt(diag(vcov(arima_mod_arma_2_1)))[-4],2)) %>%
  kable(.)
```

coef	Truth	HR	seHR	MLE	seMLE
phi[1]	0.6	0.99	0.21	0.54	0.26
phi[2]	0.2	-0.15	0.18	0.23	0.22
ma[1]	0.4	-0.04	0.21	0.44	0.24

- Now compare with larger values of n with modified hannan_fast function to speed things along (does not do expensive estimation of σ^2):

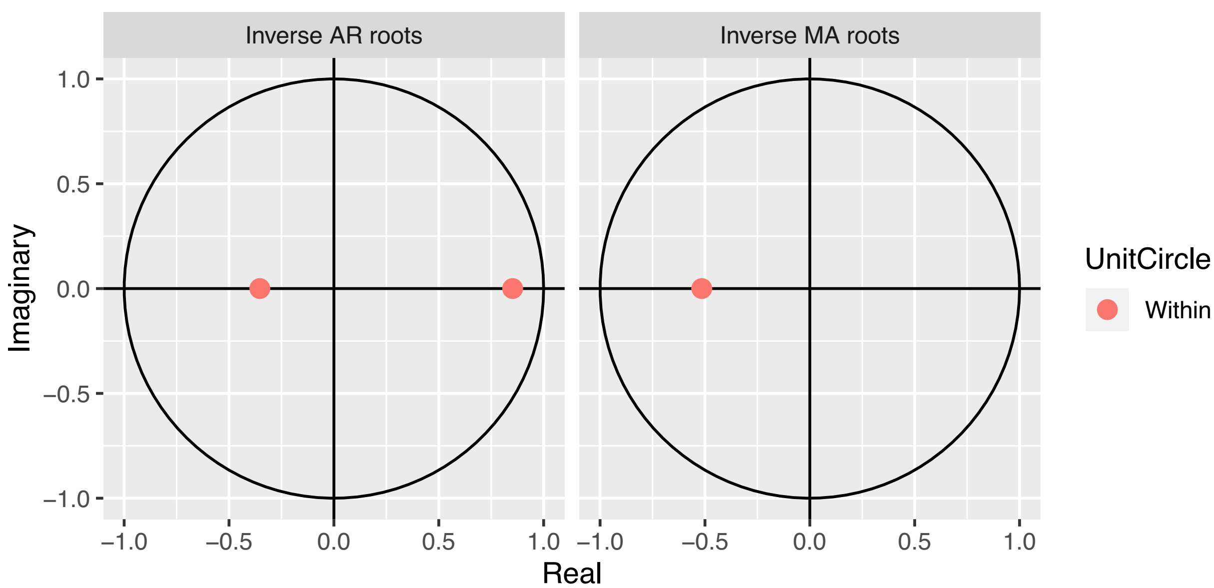
```
hannan_fast<-function (x, p, q)
{
    if (q < 1)
        stop("q < 1")
    n = length(x)
    x = x - mean(x)
    m = 20 + p + q
    k = max(p, q)
    phi = ar.yw(x, aic = FALSE, order.max = m, demean = FALSE)$ar
    F1 = function(t) x[t] - sum(phi * x[(t - 1):(t - m)])
    z = sapply((m + 1):n, F1)
    z = c(numeric(m), z)
    F2 = function(i) z[(m + k + i - 1):(m + k + i - q)]
    Z = sapply(1:(n - m - k), F2)
    Z = matrix(Z, nrow = n - m - k, ncol = q, byrow = TRUE)
    if (p > 0) {
        F3 = function(i) x[(m + k + i - 1):(m + k + i - p)]
        X = sapply(1:(n - m - k), F3)
        X = matrix(X, nrow = n - m - k, ncol = p, byrow = TRUE)
        Z = cbind(X, Z)
    }
    G = qr.solve(qr(t(Z) %*% Z))
    b = G %*% t(Z) %*% x[(m + 1 + k):n]
    xhat = Z %*% b
    e = x[(m + 1 + k):n] - xhat
    sigma2 = sum(e^2)/(n - m - k)
    se = sqrt(sigma2 * diag(G))
    if (p == 0) {
        phi = 0
        se.phi = 0
    }
    else {
        phi = b[1:p]
        se.phi = se[1:p]
    }
    theta = b[p + (1:q)]
    se.theta = se[p + (1:q)]
    a = list(phi = phi, theta = theta, sigma2 = NA, aicc = NA,
             se.phi = se.phi, se.theta = se.theta)
a
```

```
}
```

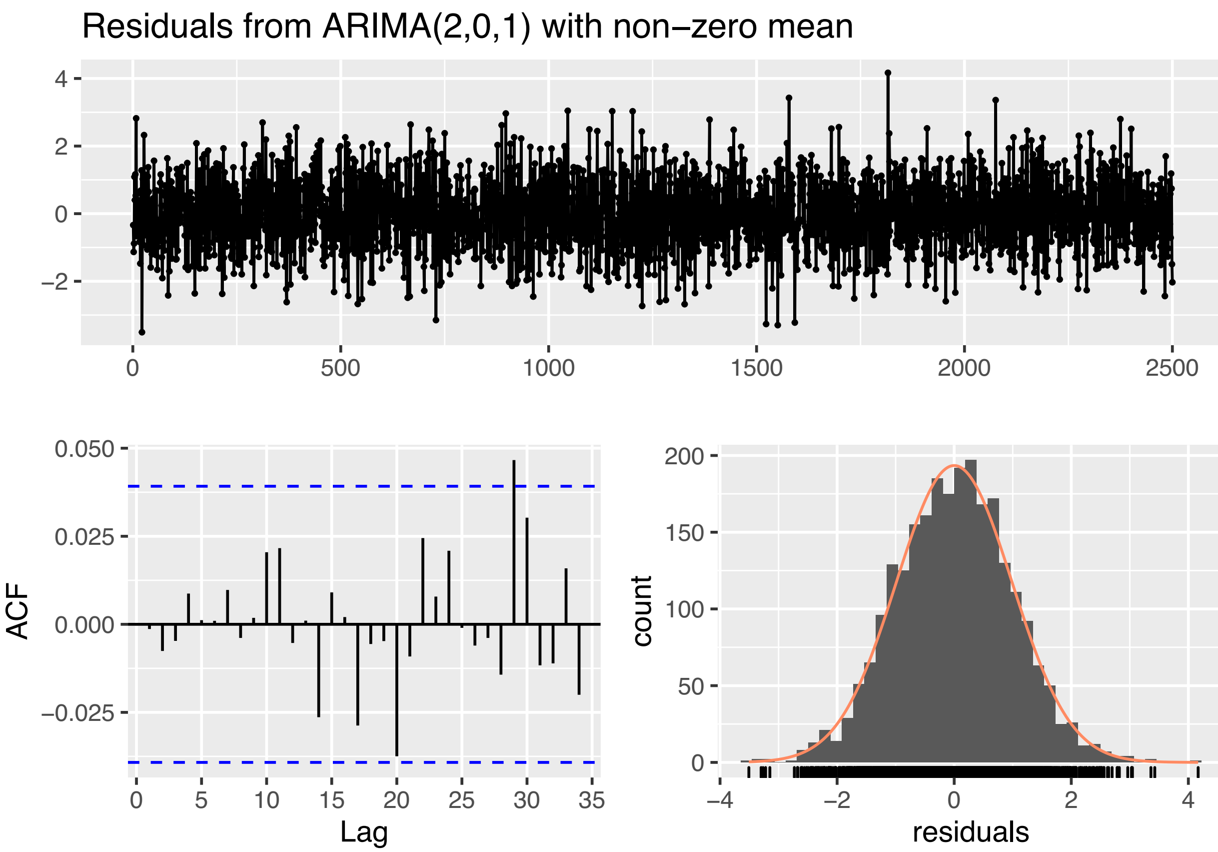
```
set.seed(94014) ## Note with the old seed, HR is still terrible!

my_big_arma_2_1_data = arima.sim(my_ARMA_2_1_model,n=2500)

hanr_big_arma_2_1<-hannan_fast(my_big_arma_2_1_data - mean(my_big_arma_2_1_data),p=2,q=1)

arima_mod_big_arma_2_1<-Arima(my_big_arma_2_1_data,order=c(2,0,1))
summary(arima_mod_big_arma_2_1)
```

```
Series: my_big_arma_2_1_data
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1     ar2     ma1    mean
      0.4991  0.3010  0.5152  0.0304
s.e.  0.0960  0.0867  0.0890  0.1496

sigma^2 estimated as 0.9799:  log likelihood=-3520.68
AIC=7051.36   AICc=7051.38   BIC=7080.48

Training set error measures:
                       ME      RMSE       MAE      MPE     MAPE      MASE
Training set 7.761268e-05 0.9890855 0.7896762 269.3692 477.2689 0.9582402
                     ACF1
Training set -0.001331118
```

```
autoplot(arima_mod_big_arma_2_1)
```




```
checkresiduals(arima_mod_big_arma_2_1)
```




```
    Ljung-Box test

data:  Residuals from ARIMA(2,0,1) with non-zero mean
Q* = 1.7366, df = 6, p-value = 0.9423

Model df: 4.   Total lags used: 10
```

```
tibble(coef=c("phi[1]","phi[2]","ma[1]"), Truth=c(true_ar_coef,true_ma_coef),
       HR = round(c(hanr_big_arma_2_1$phi, hanr_big_arma_2_1$theta),2),
       seHR = round(c(hanr_big_arma_2_1$se.phi, hanr_big_arma_2_1$se.theta),2),
       MLE = round(coef(arima_mod_big_arma_2_1)[-4],2),
       seMLE = round(sqrt(diag(vcov(arima_mod_big_arma_2_1)))[-4],2)) %>%
  kable(.)
```

closer to truth

coef	Truth	HR	seHR	MLE	seMLE
phi[1]	0.6	0.59	0.11	0.50	0.10
phi[2]	0.2	0.22	0.10	0.30	0.09
ma[1]	0.4	0.43	0.12	0.52	0.09

HR is faster than MLE for large dataset

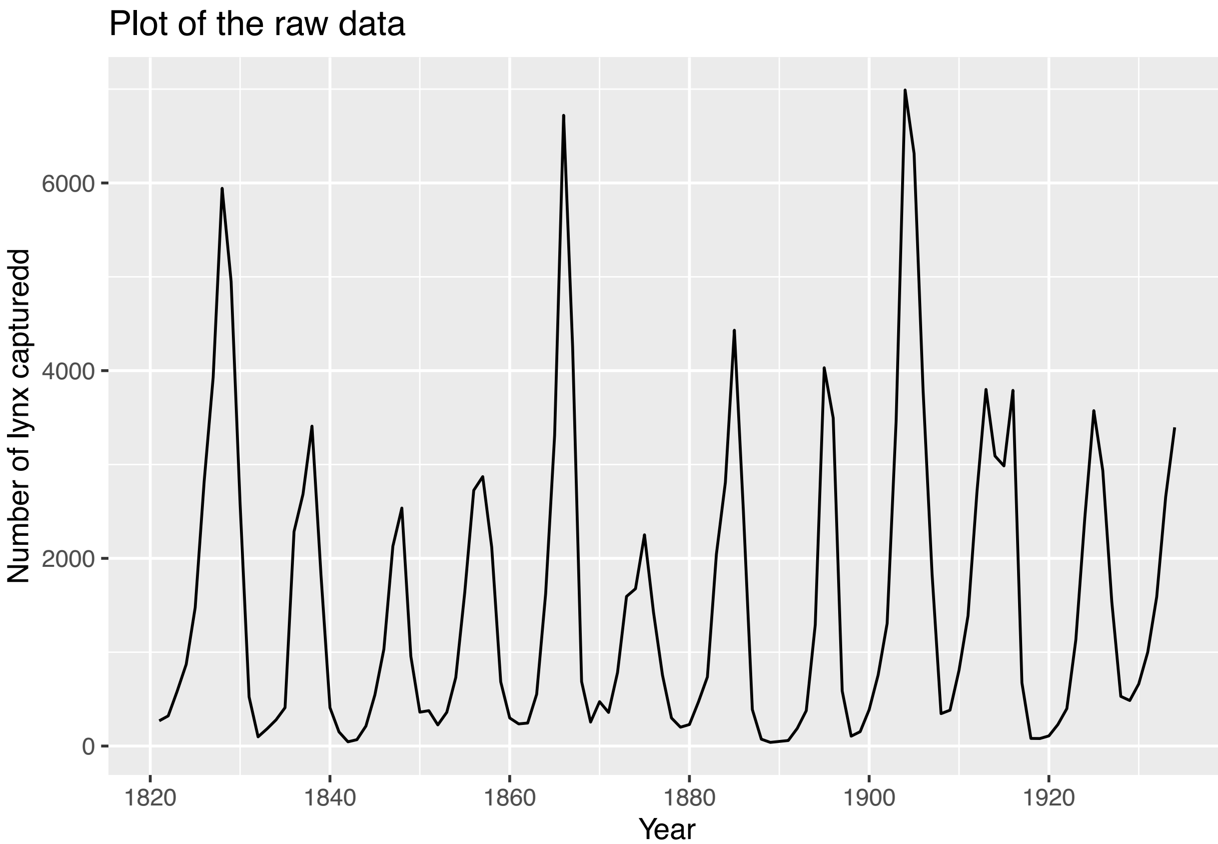
And now for some real data

- Canadian Lynx data (1821-1934)

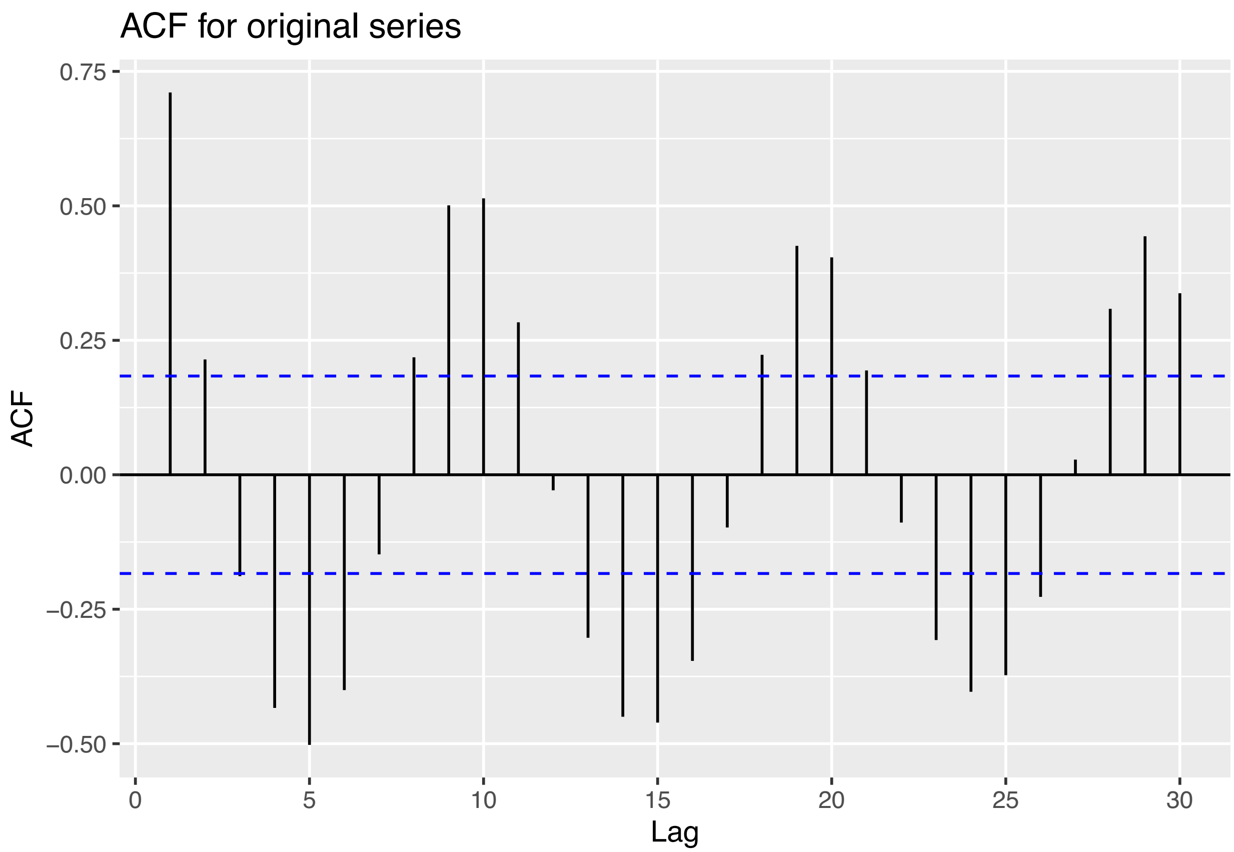
```
data(lynx)
class(lynx)
```

```
[1] "ts"
```

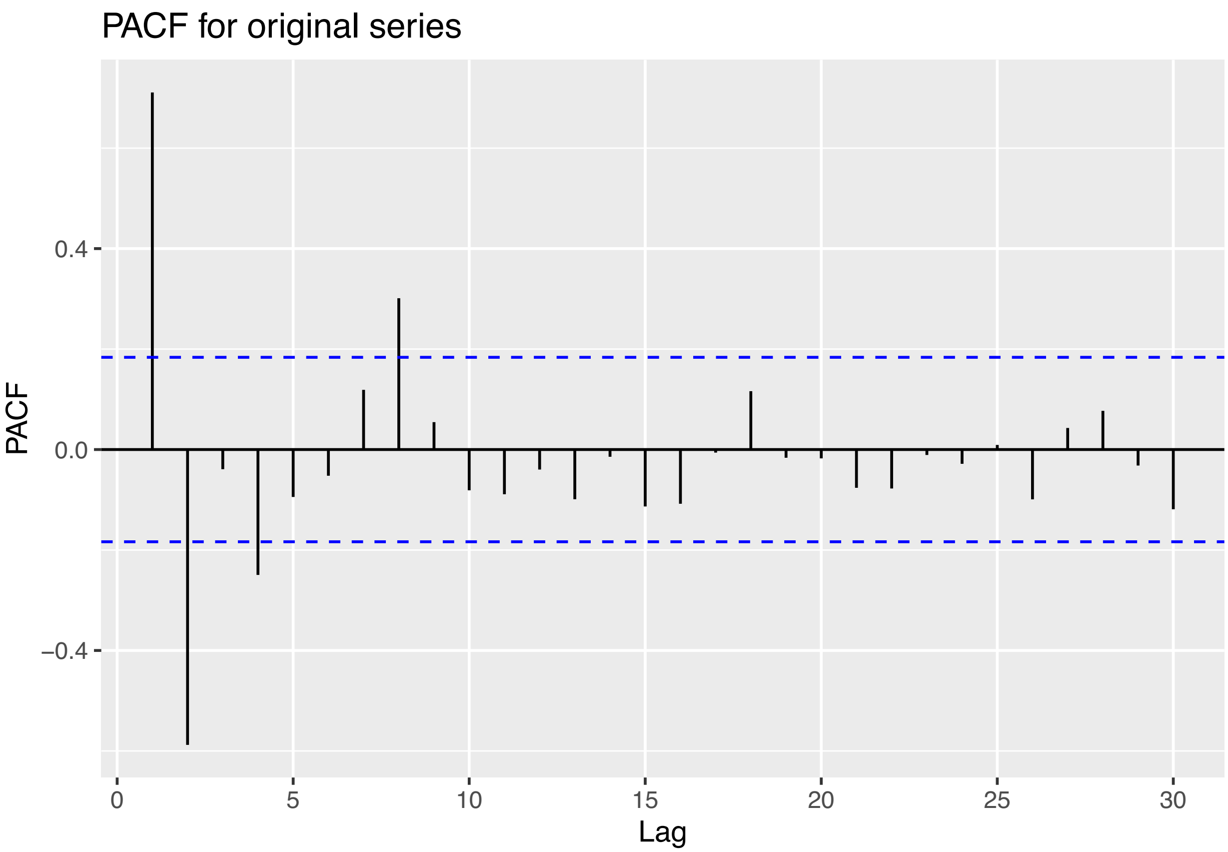
```r
autoplot(lynx) + ggtitle("Plot of the raw data") + ylab("Number of lynx capturedd") + xlab("Year")
```




```r
ggAcf(lynx,lag=30) + ggtitle("ACF for original series")
```

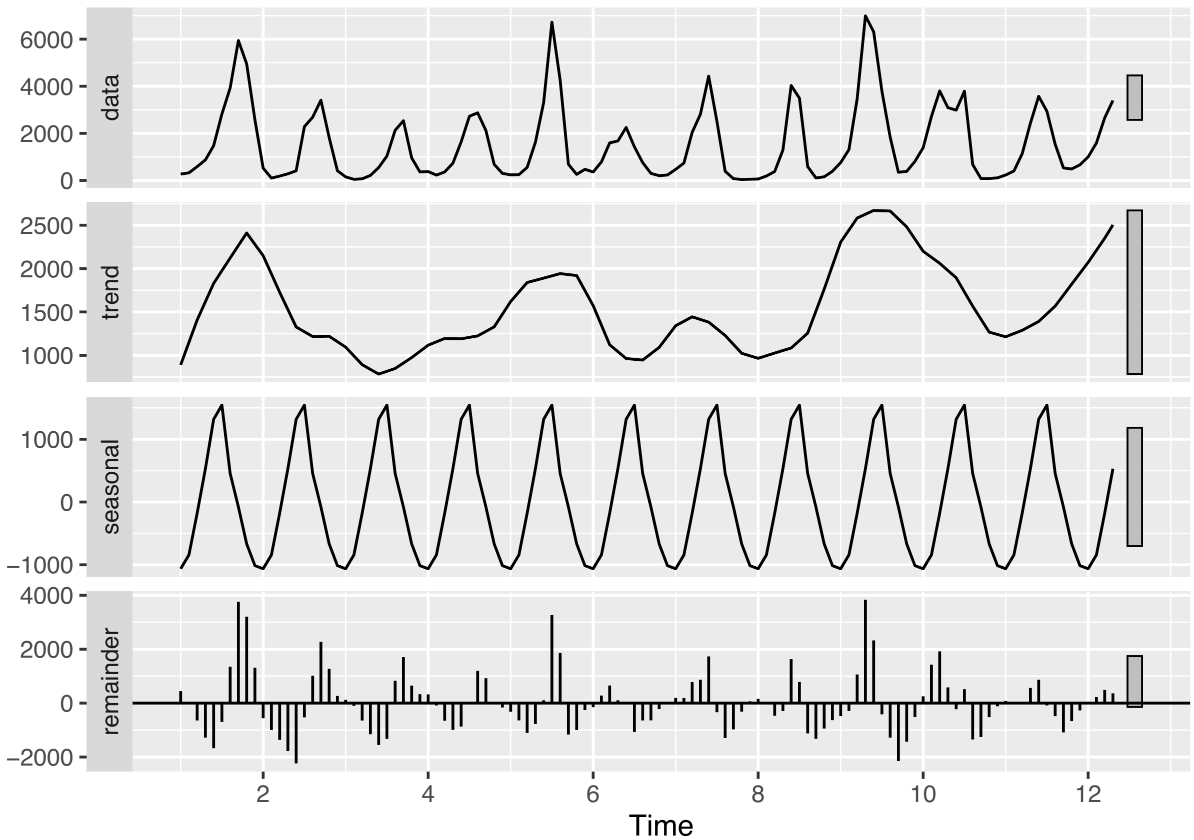



```
ggPacf(lynx,lag=30)+ ggtitle("PACF for original series")
```



- Approach 1: STL (Seasonal and Trend removal via Loess) – Replaces the moving average vs. polynomial trend removal with LOESS, which is a local regression modelling approach that uses a moving average model that fits locally smooth polynomials to allow for interpolation

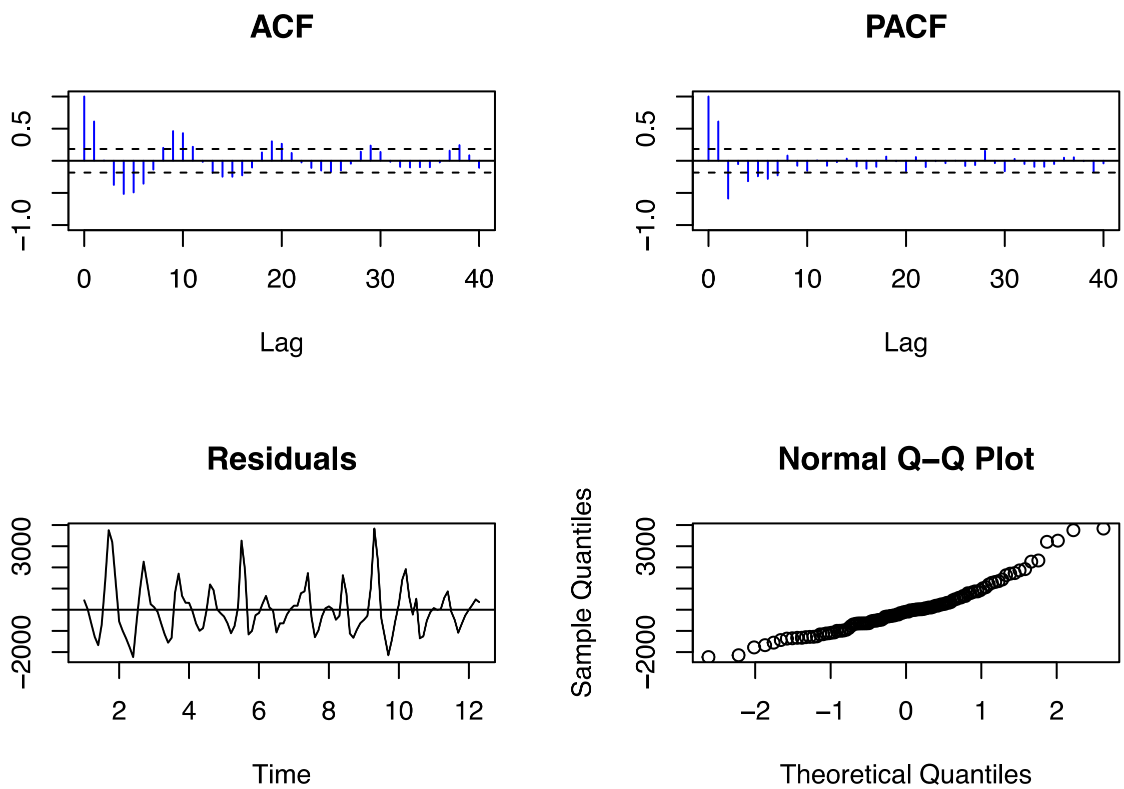
```
lynx_freq_10 = lynx %>% ts(., frequency=10)

lynx_stl=stl(lynx_freq_10,s.window="periodic")

autoplot(lynx_stl)
```



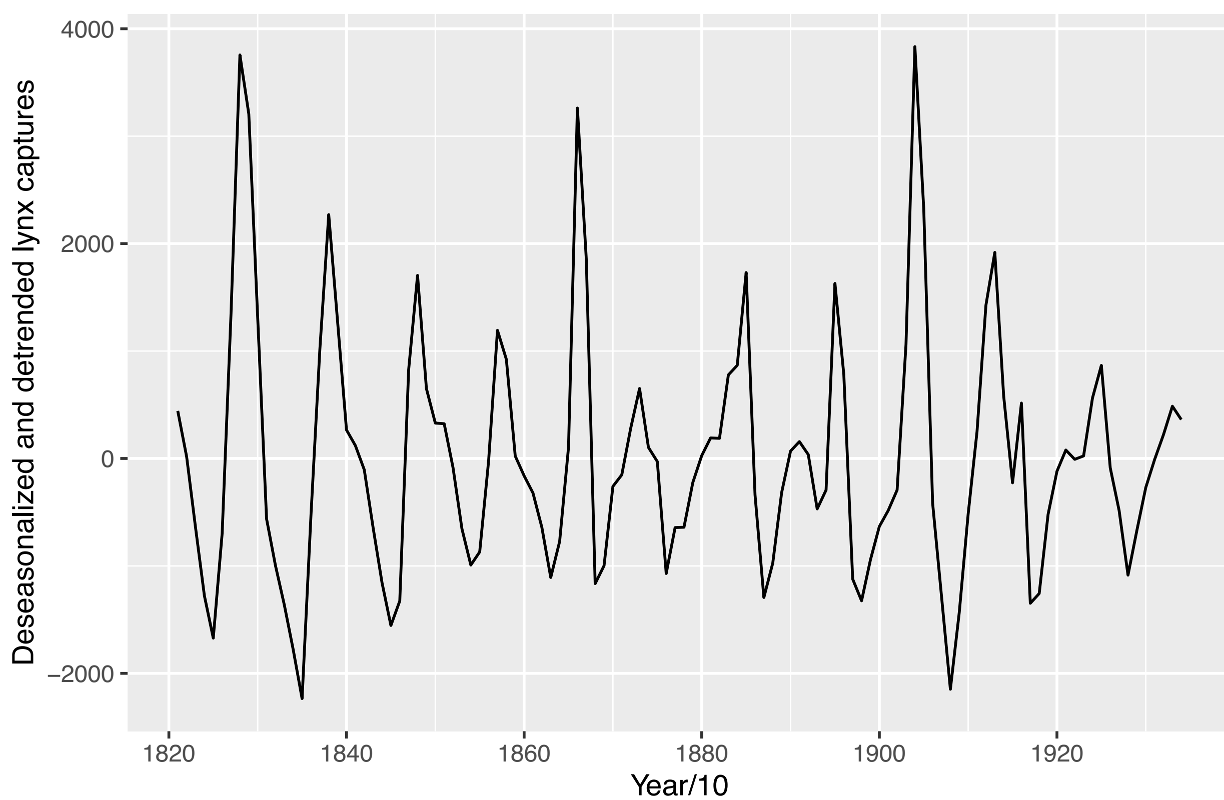

```
lynx_stl_remainder = lynx_stl$time.series[,"remainder"] %>% ts(.,frequency=1,start=1821)
test(lynx_stl$time.series[,"remainder"])
```

```
Null hypothesis: Residuals are iid noise.
Test                          Distribution Statistic   p-value
Ljung-Box Q                  Q ~ chisq(20)    254.66         0 *
McLeod-Li Q                  Q ~ chisq(20)     27.21    0.1294
Turning points T   (T-74.7)/4.5 ~ N(0,1)         32         0 *
Diff signs S       (S-56.5)/3.1 ~ N(0,1)         58     0.628
Rank P         (P-3220.5)/204.2 ~ N(0,1)       3279    0.7745
```

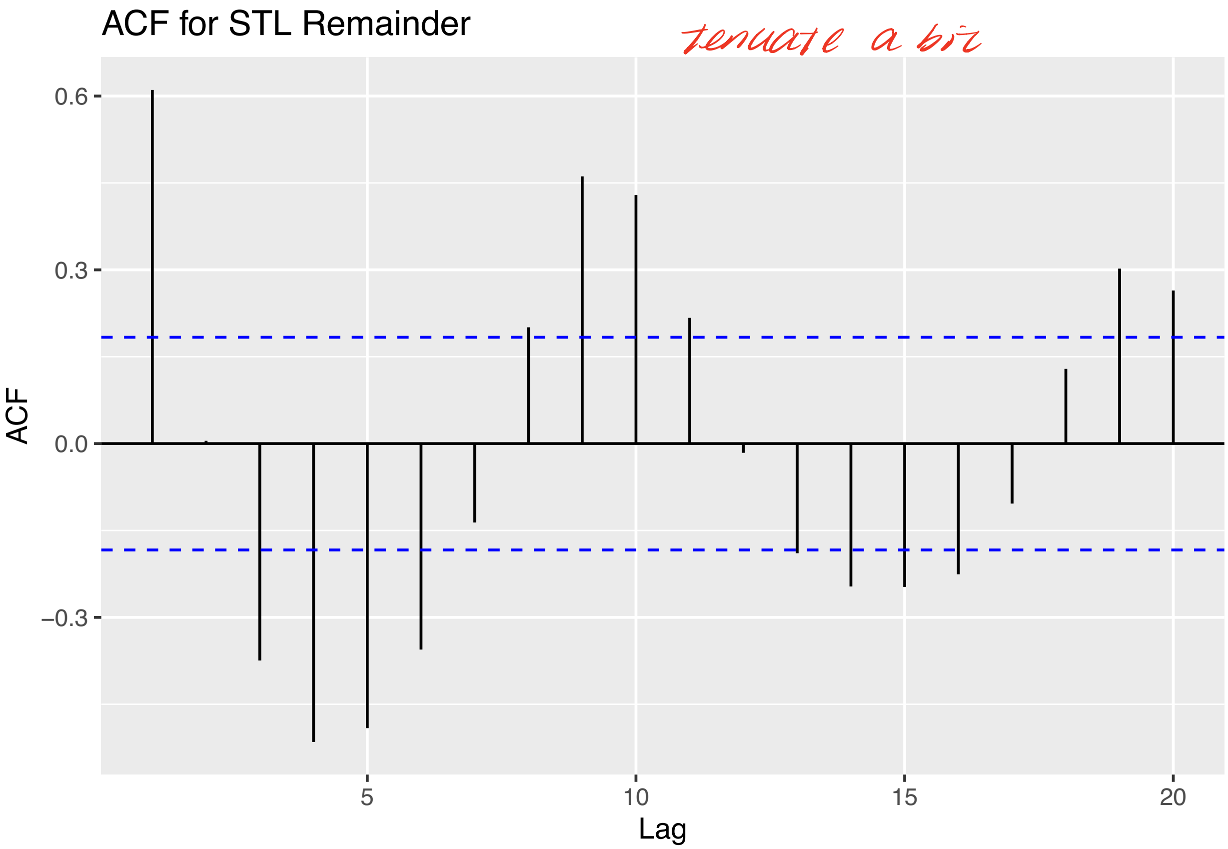
against WN




```r
autoplot(lynx_stl_remainder) + ylab("Deseasonalized and detrended lynx captures") + xlab("Year/10")
```



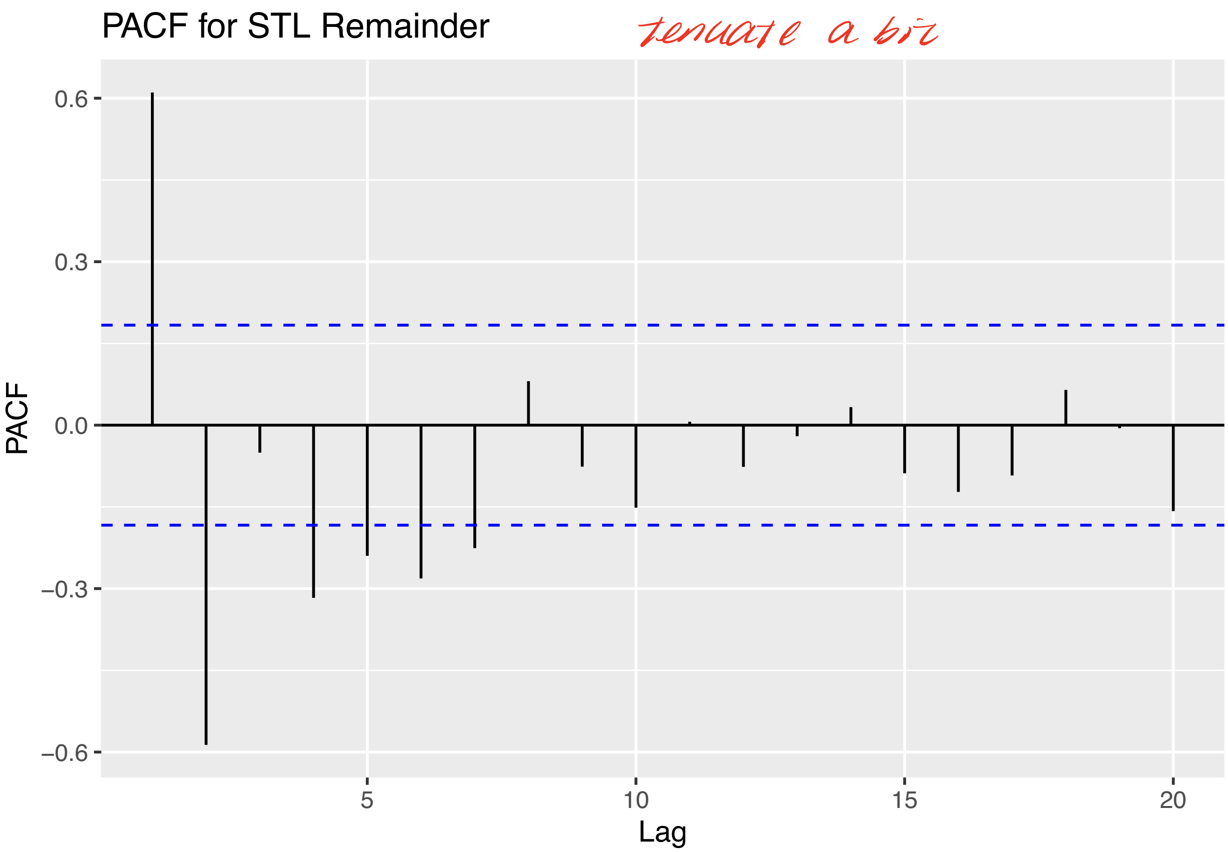
```r
ggAcf(lynx_stl_remainder) + ggtitle("ACF for STL Remainder")
```



tenuate a bit

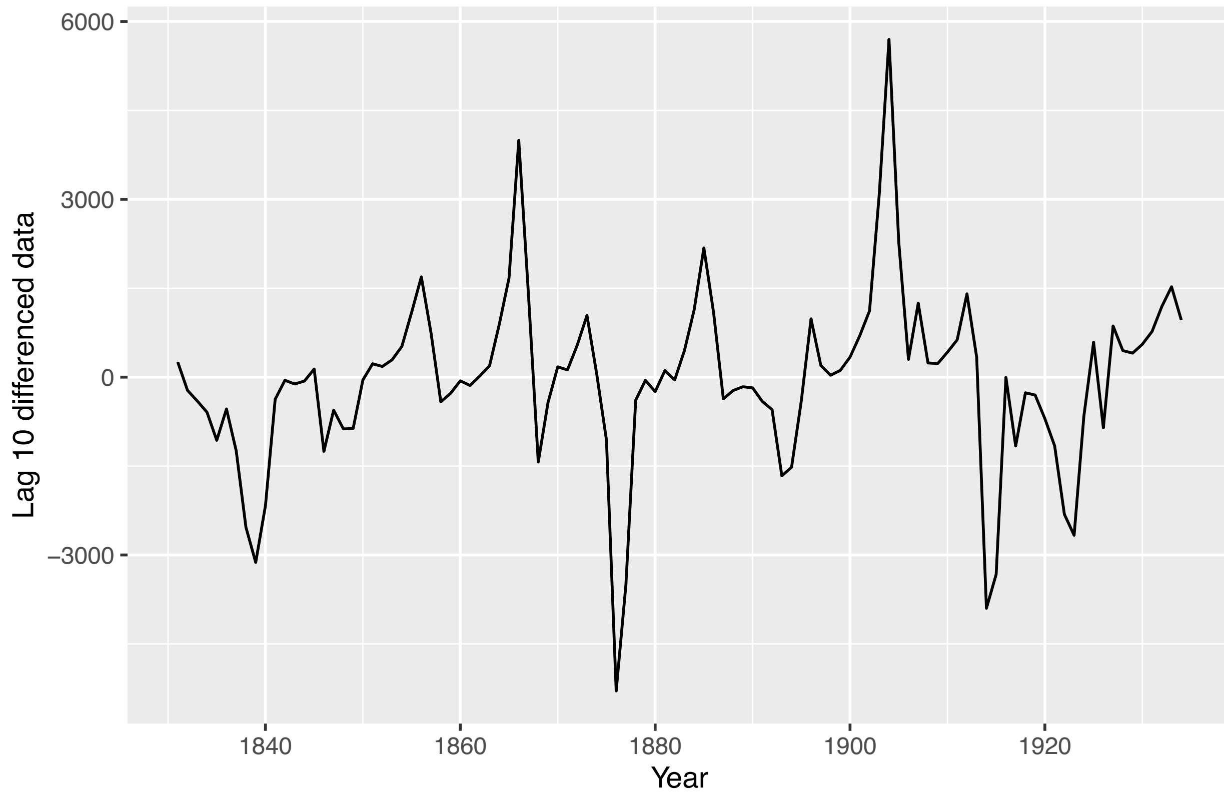
```r
ggPacf(lynx_stl_remainder) + ggtitle("PACF for STL Remainder")
```

tenuate a bit

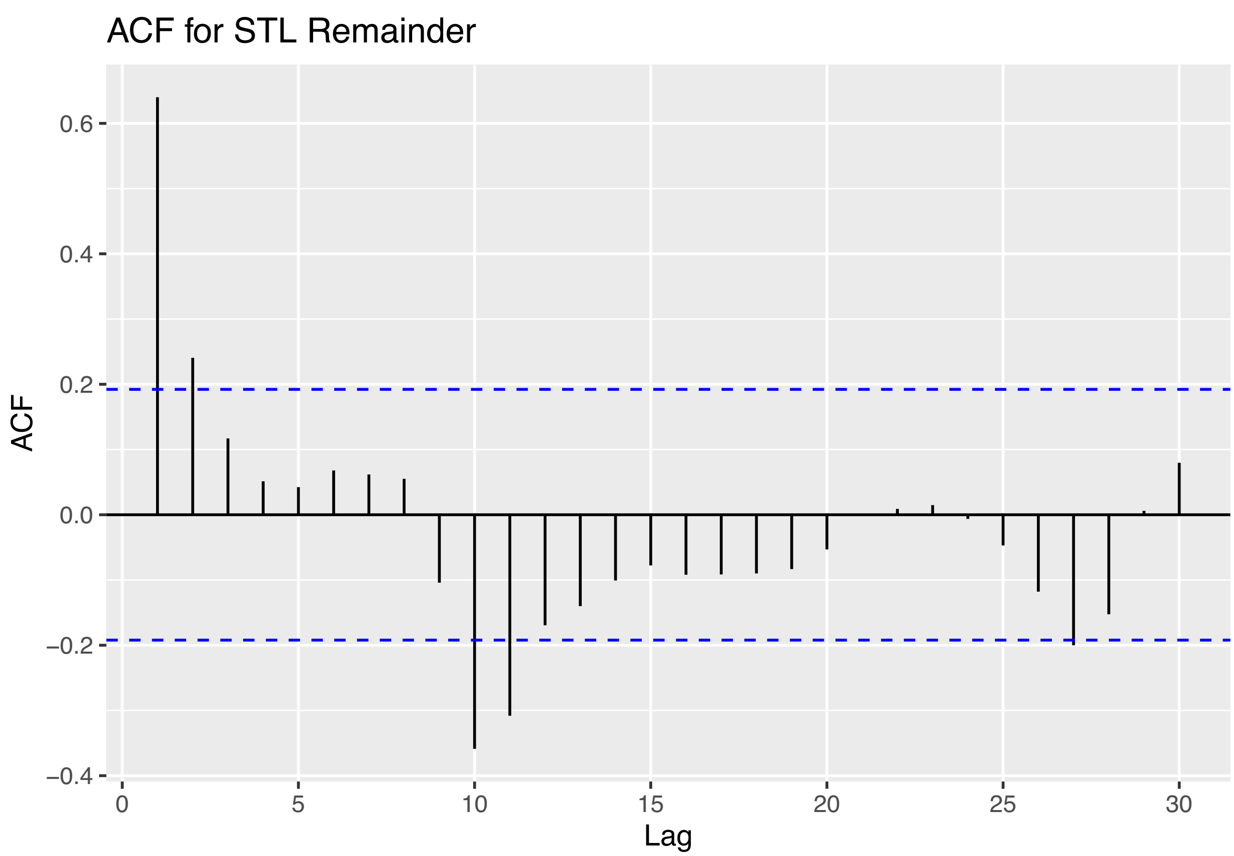


- Approach 2: Classic differencing approach

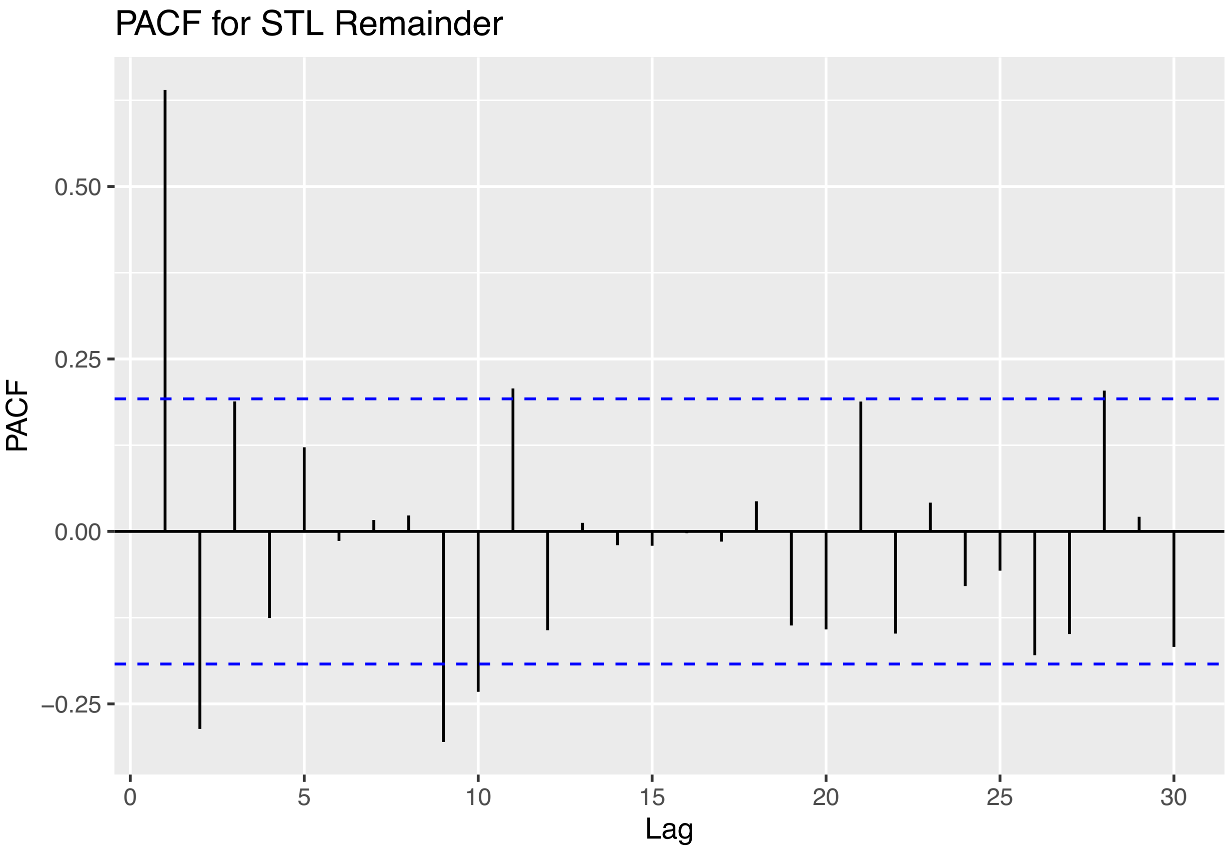
```
lynx_diff_10 = lynx %>% diff(.,lag=10)

autoplot(lynx_diff_10) + ylab("Lag 10 differenced data") + xlab("Year")
```




```
ggAcf(lynx_diff_10,lag=30) + ggtitle("ACF for STL Remainder")
```

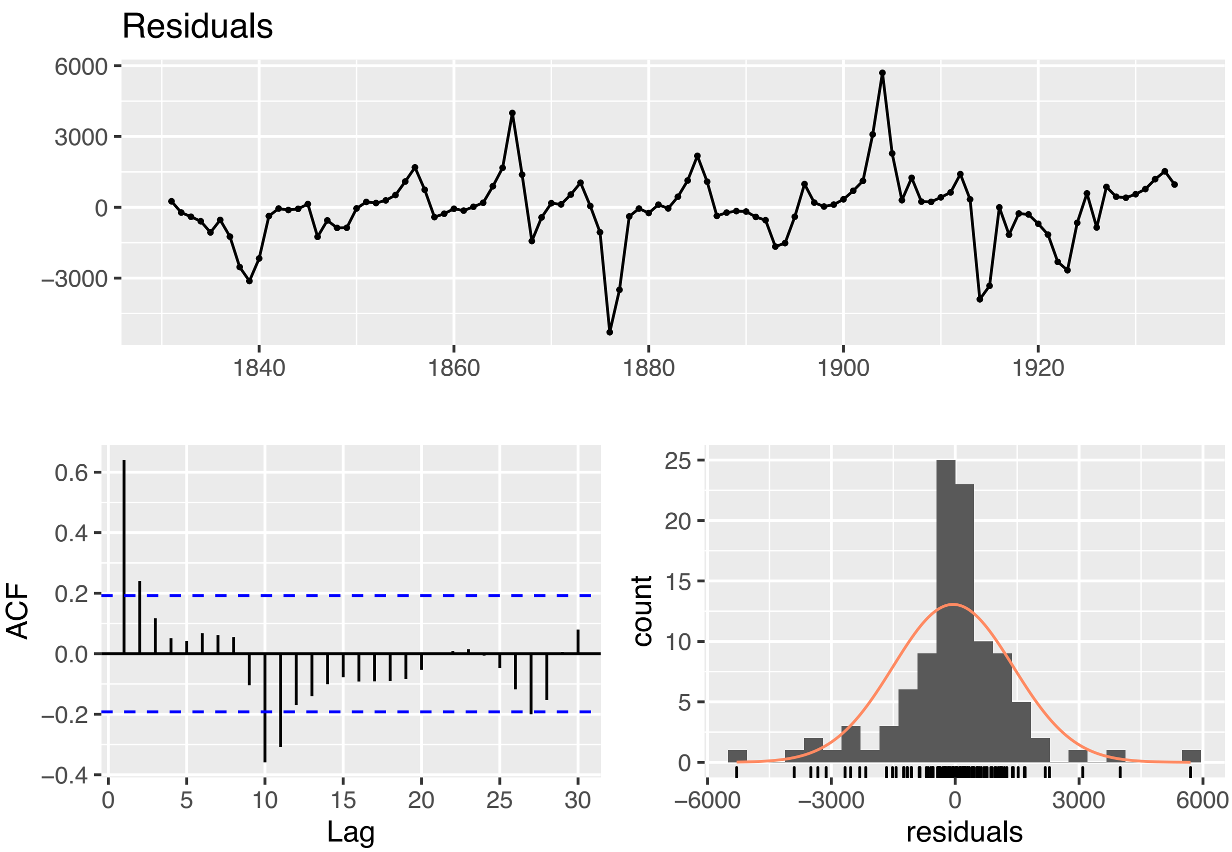


```r
ggPacf(lynx_diff_10,lag=30) + ggtitle("PACF for STL Remainder")
```




```r
checkresiduals(lynx_diff_10,lag.max=30)
```

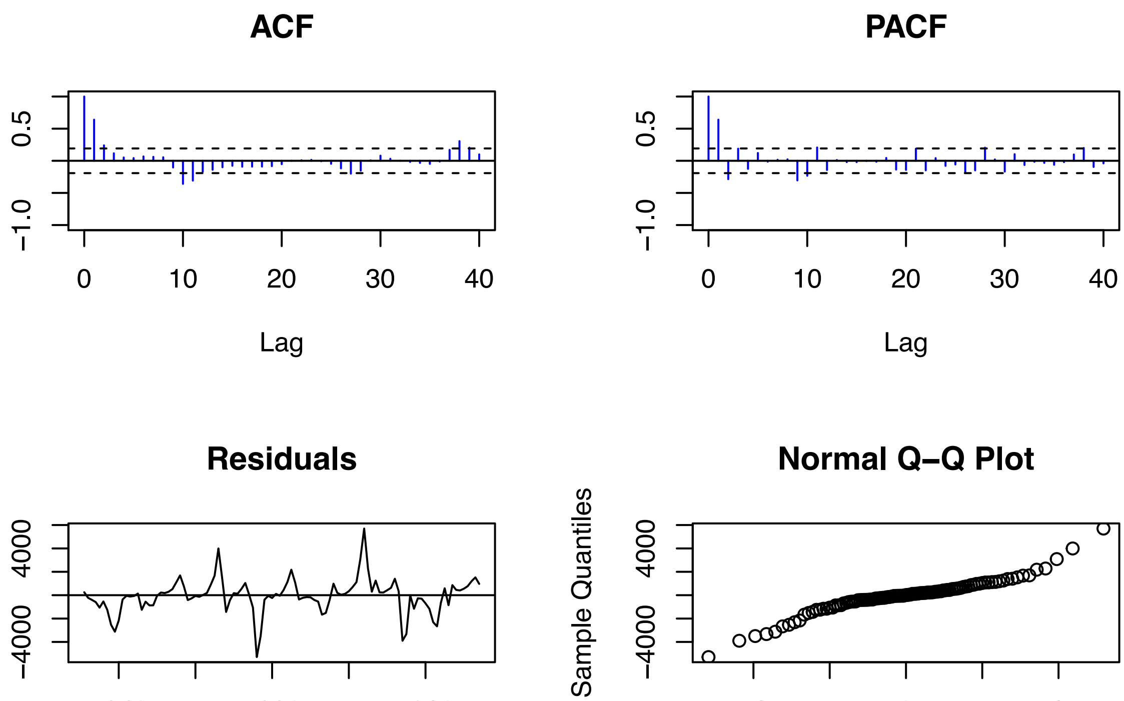
```
Warning in modeldf.default(object): Could not find appropriate degrees of
freedom for this model.
```




```r
test(lynx_diff_10)
```

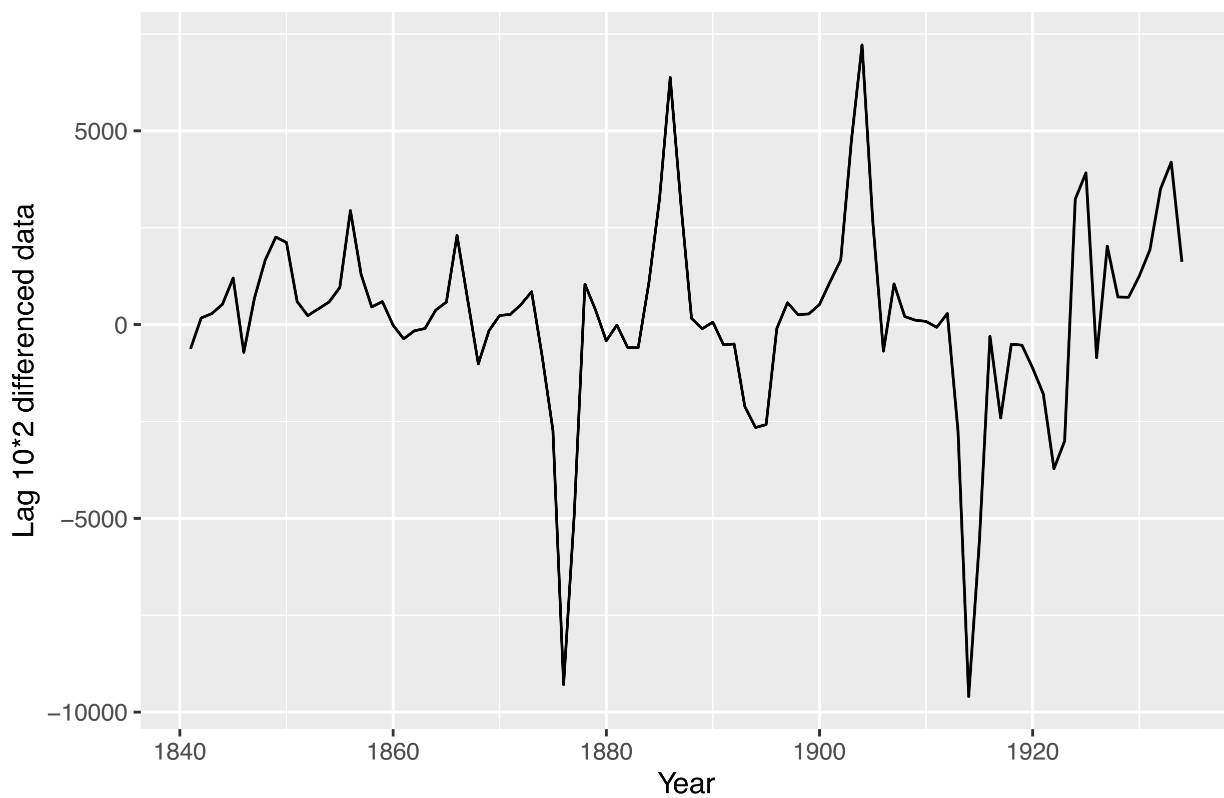
```
Null hypothesis: Residuals are iid noise.
Test                        Distribution Statistic   p-value
Ljung-Box Q                Q ~ chisq(20)     93.24         0 *
McLeod-Li Q                Q ~ chisq(20)     50.57     2e-04 *
Turning points T      (T-68)/4.3 ~ N(0,1)       46         0 *
Diff signs S          (S-51.5)/3 ~ N(0,1)       57     0.063
Rank P              (P-2678)/178 ~ N(0,1)     3118    0.0134 *
```

No evidence for WN

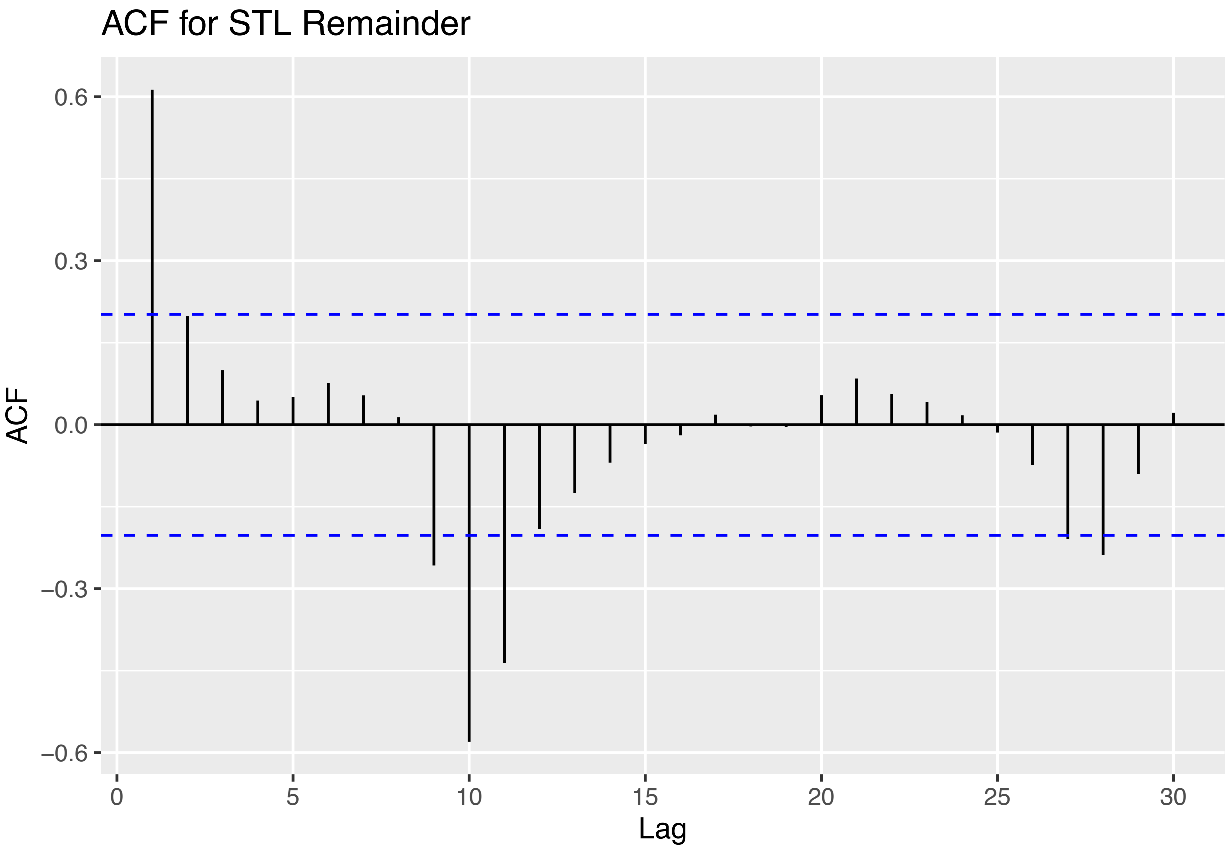


second diff

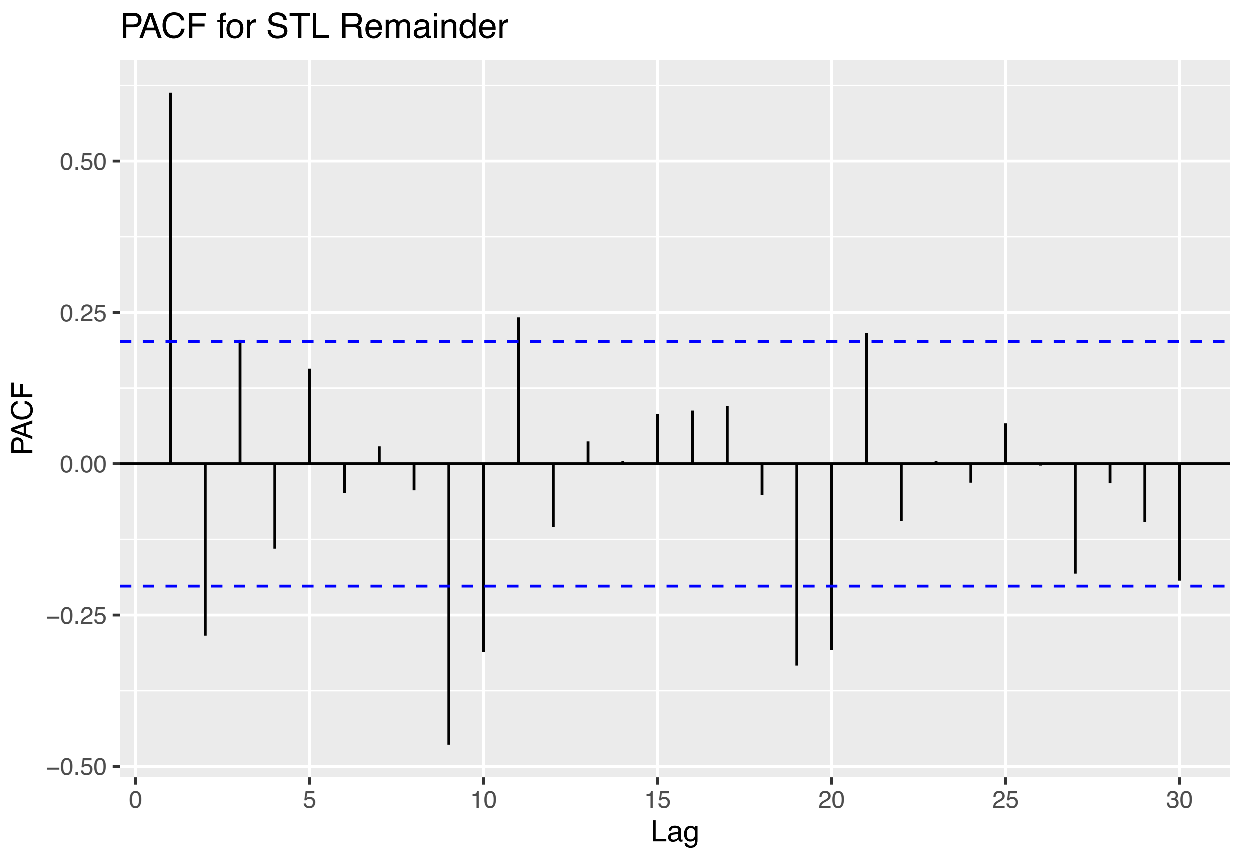
```
lynx_diff_10_2 = lynx_diff_10 %>% diff(.,lag=10)
autoplot(lynx_diff_10_2) + ylab("Lag 10*2 differenced data") + xlab("Year")
```



```
ggAcf(lynx_diff_10_2,lag=30) + ggtitle("ACF for STL Remainder")
```

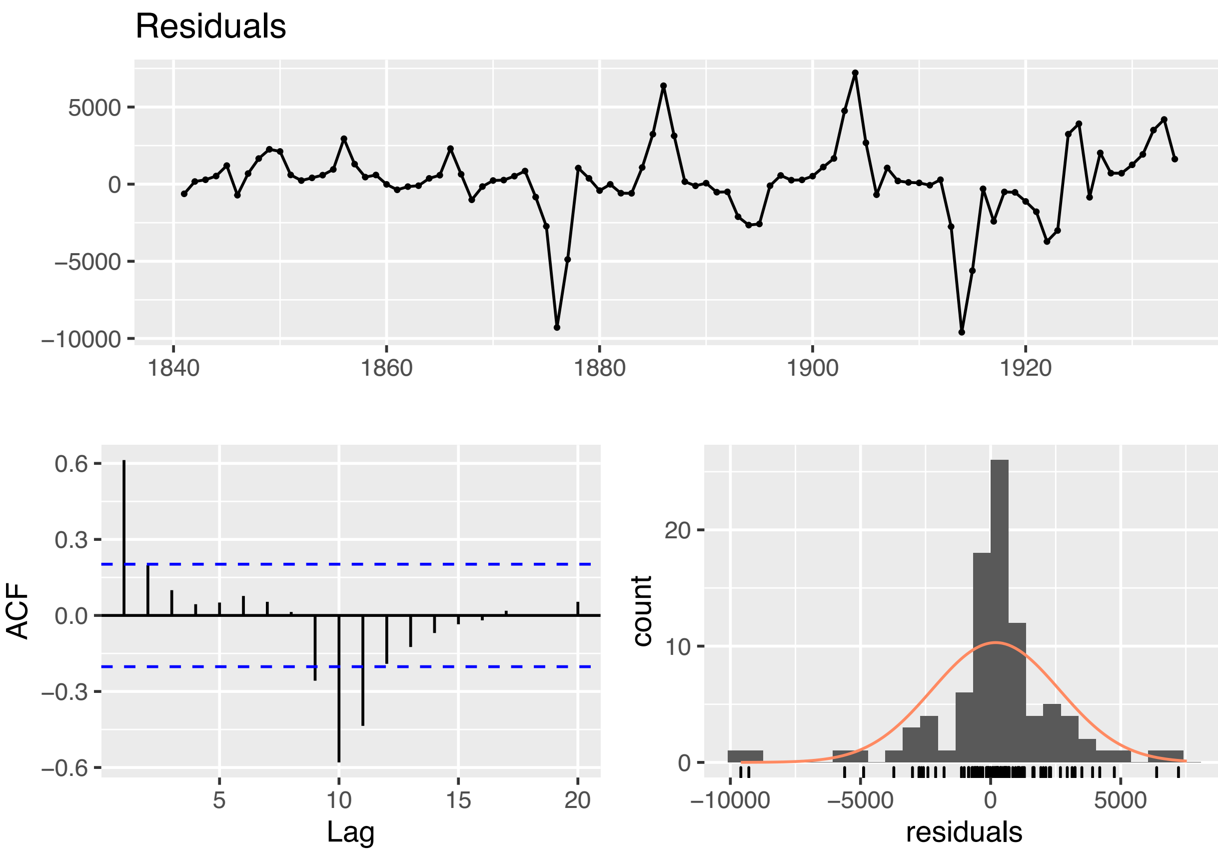



```
ggPacf(lynx_diff_10_2,lag=30) + ggtitle("PACF for STL Remainder")
```




```
checkresiduals(lynx_diff_10_2)
```

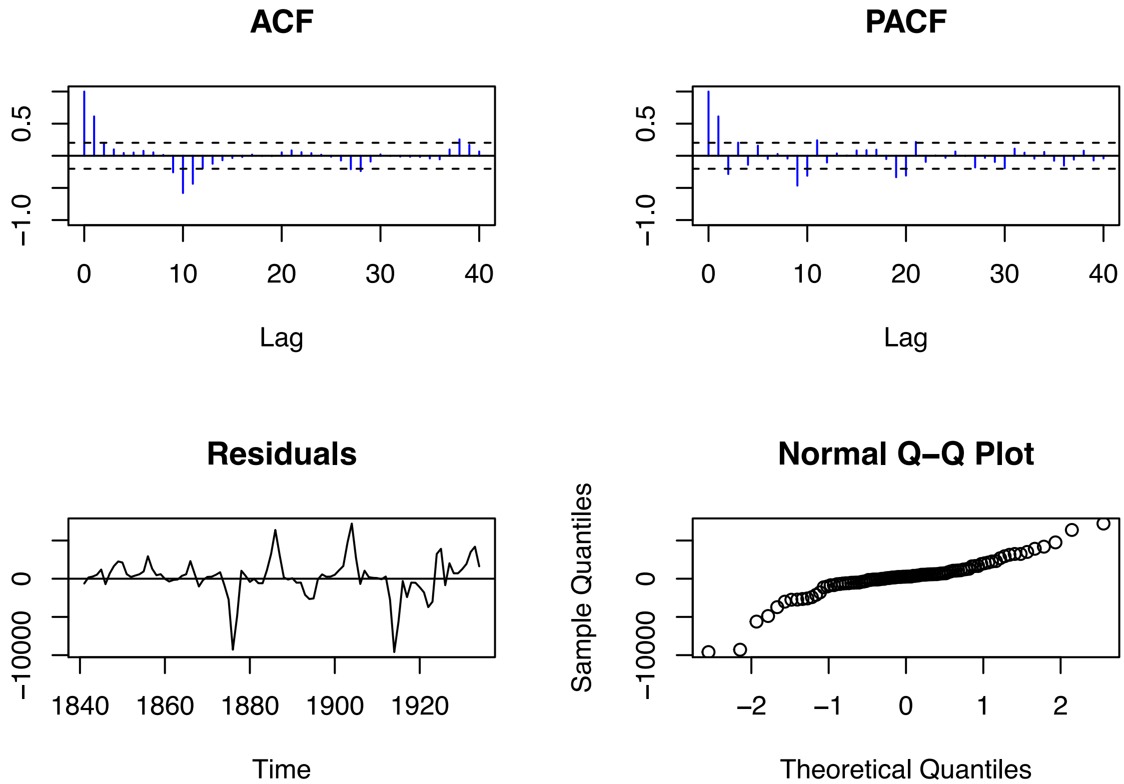
```
Warning in modeldf.default(object): Could not find appropriate degrees of
freedom for this model.
```




```r
test(lynx_diff_10_2)
```

```
Null hypothesis: Residuals are iid noise.
Test                          Distribution Statistic   p-value
Ljung-Box Q                  Q ~ chisq(20)    113.35         0 *
McLeod-Li Q                  Q ~ chisq(20)     51.16     2e-04 *
Turning points T      (T-61.3)/4 ~ N(0,1)        39         0 *
Diff signs S        (S-46.5)/2.8 ~ N(0,1)        52    0.0506
Rank P          (P-2185.5)/153.1 ~ N(0,1)      2083    0.5031
```

NOT improve



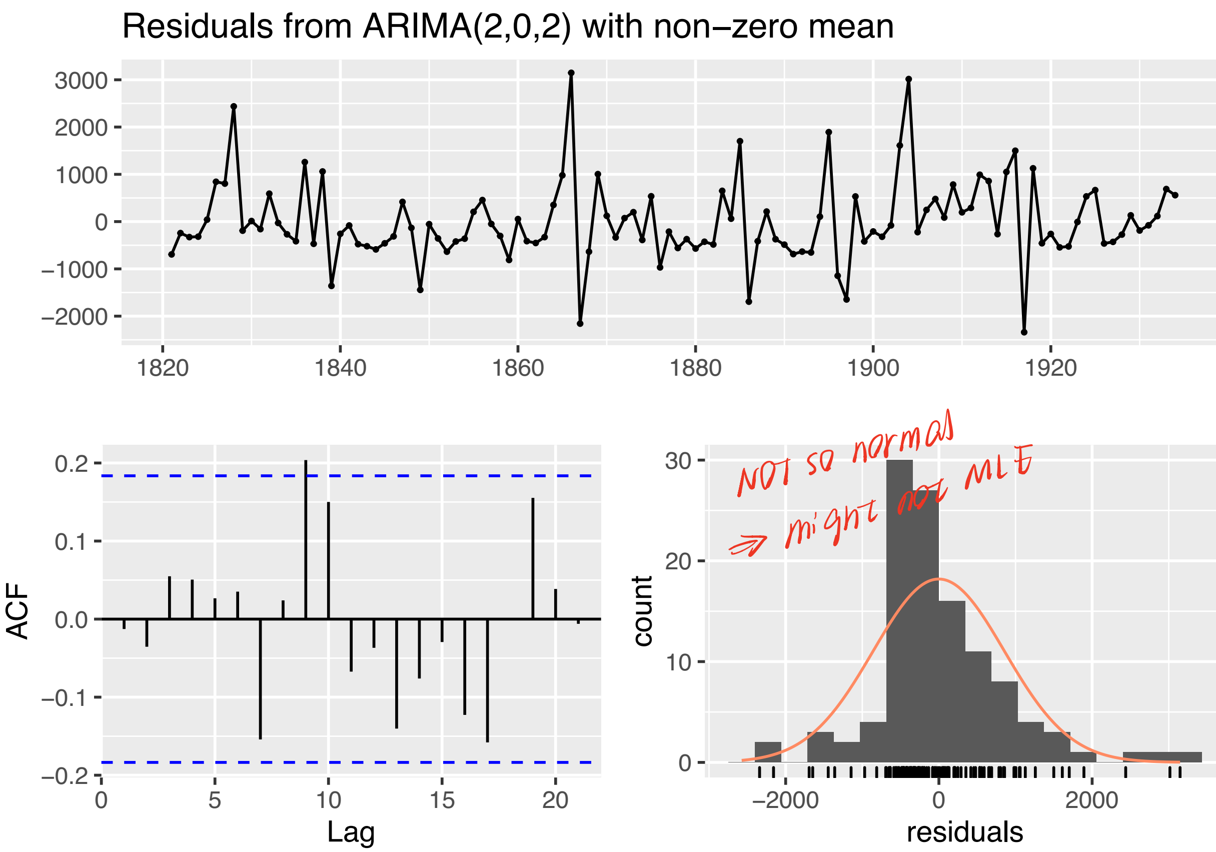
Using ARMA

```
### Orig Data
lynx_orig_2_2 <- arima(lynx,c(2,0,2))
summary(lynx_orig_2_2)
```

```
Call:
arima(x = lynx, order = c(2, 0, 2))

Coefficients:                NOT significantly from zero
         ar1      ar2      ma1      ma2  intercept
      1.3421  -0.6738  -0.2027  -0.2564  1544.4039
s.e.  0.0984   0.0801   0.1261   0.1097   131.9242

sigma^2 estimated as 728546:  log likelihood = -932.08,  aic = 1876.17

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE
Training set -1.608903 853.5488 610.1112 -63.90926 140.7693 0.7343143
                    ACF1
Training set -0.01267127
```

```
checkresiduals(lynx_orig_2_2,2,2)
```

```
Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
```



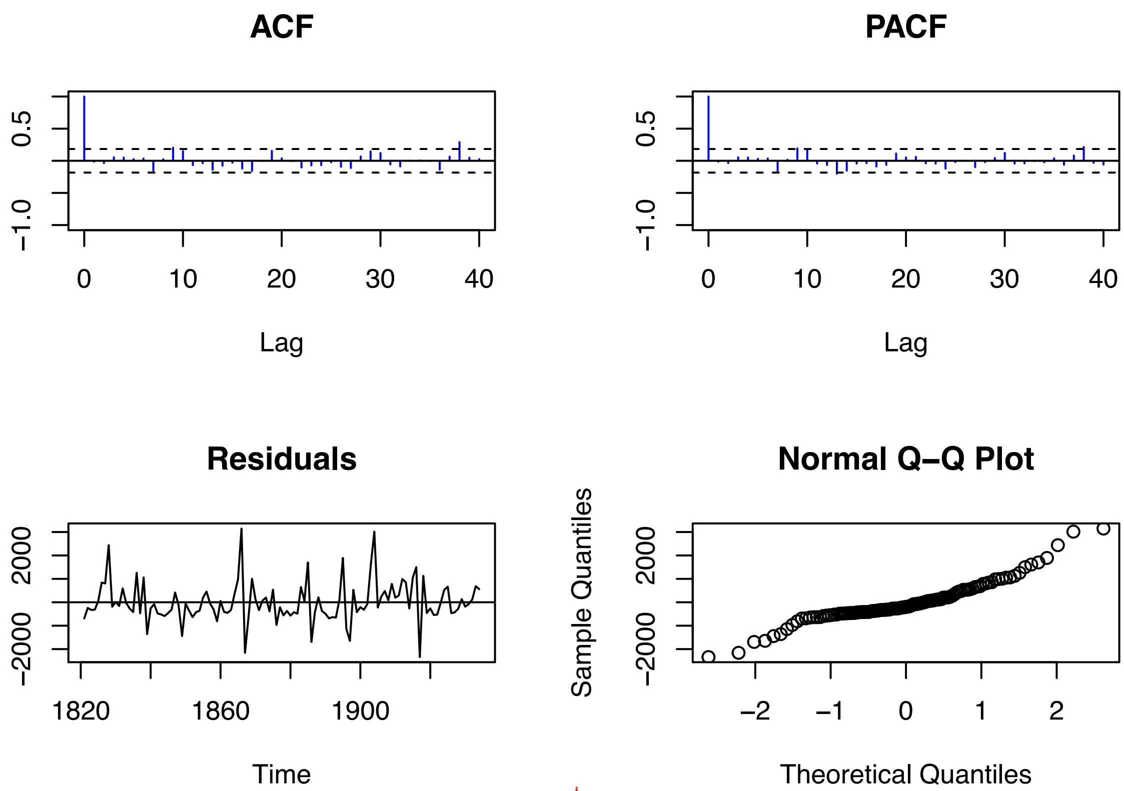

```
    Ljung-Box test

data:  Residuals from ARIMA(2,0,2) with non-zero mean
Q* = 0.16653, df = -3, p-value = NA

Model df: 5.   Total lags used: 2
```

```r
test(residuals(lynx_orig_2_2))
```

don't account for estimation of some parameters

```
Null hypothesis: Residuals are iid noise.
Test                        Distribution Statistic   p-value
Ljung-Box Q                Q ~ chisq(20)     25.38    0.1874
McLeod-Li Q                Q ~ chisq(20)     22.46     0.316
Turning points T   (T-74.7)/4.5 ~ N(0,1)        71    0.4116
Diff signs S       (S-56.5)/3.1 ~ N(0,1)        63    0.0358 *
Rank P          (P-3220.5)/204.2 ~ N(0,1)     3466    0.2292
```

weak evidence against WN




```
1 - pchisq(25.38,20-5)
```

```
[1] 0.04506707
```

adjusted p-value

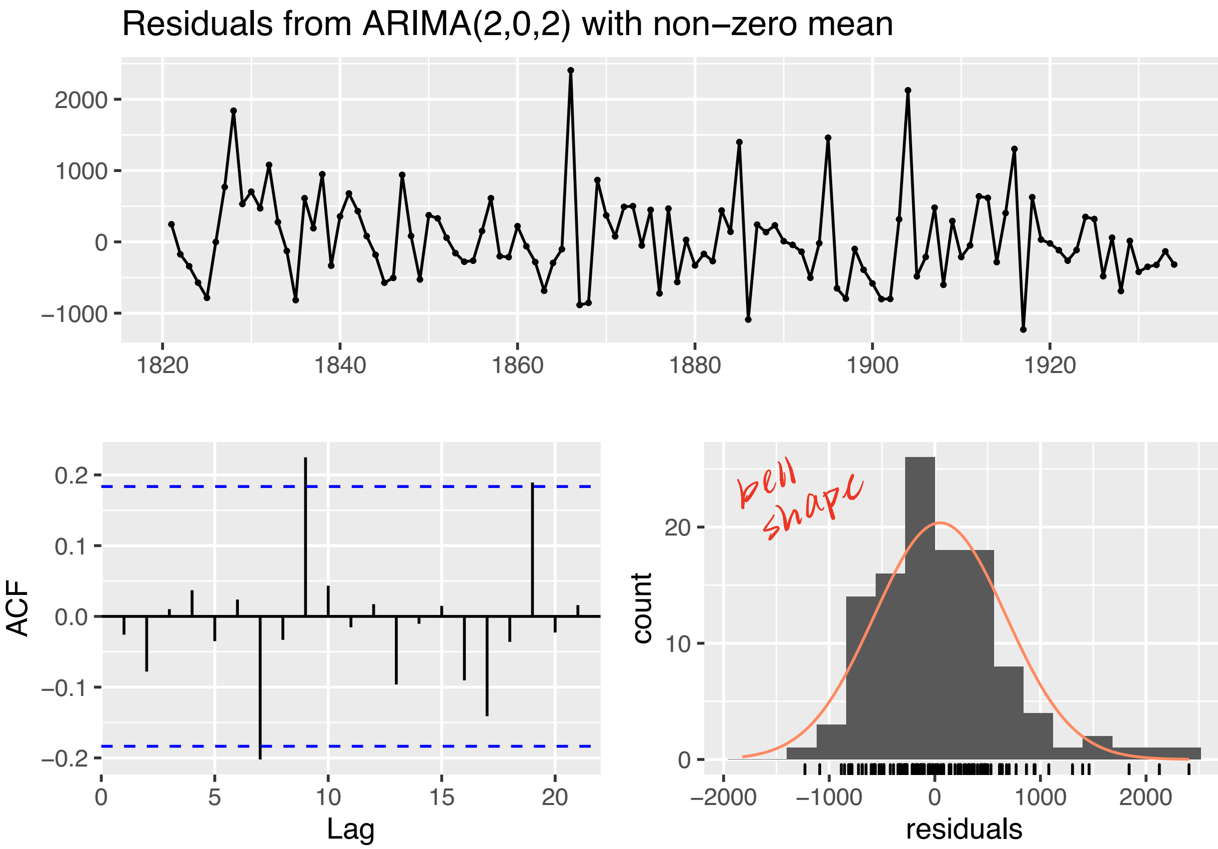
```
1 - pchisq(22.46,20-5)
```

```
[1] 0.09629755
```

```
### STL model
lynx_stl_2_2<-arima(lynx_stl_remainder,c(2,0,2))
summary(lynx_stl_2_2)
```

```
Call:
arima(x = lynx_stl_remainder, order = c(2, 0, 2))

Coefficients:
         ar1      ar2      ma1      ma2  intercept
      1.2728  -0.6695  -0.5854  -0.4146     1.0254
s.e.  0.0817   0.0790   0.1106   0.1082     6.2313

sigma^2 estimated as 389312:  log likelihood = -898.19,  aic = 1808.39

Training set error measures:
                   ME     RMSE    MAE      MPE     MAPE      MASE        ACF1
Training set 50.28488 623.9488 463.52 31.24224 192.0649 0.6265284 -0.02584277
```

```r
checkresiduals(lynx_stl_2_2)
```

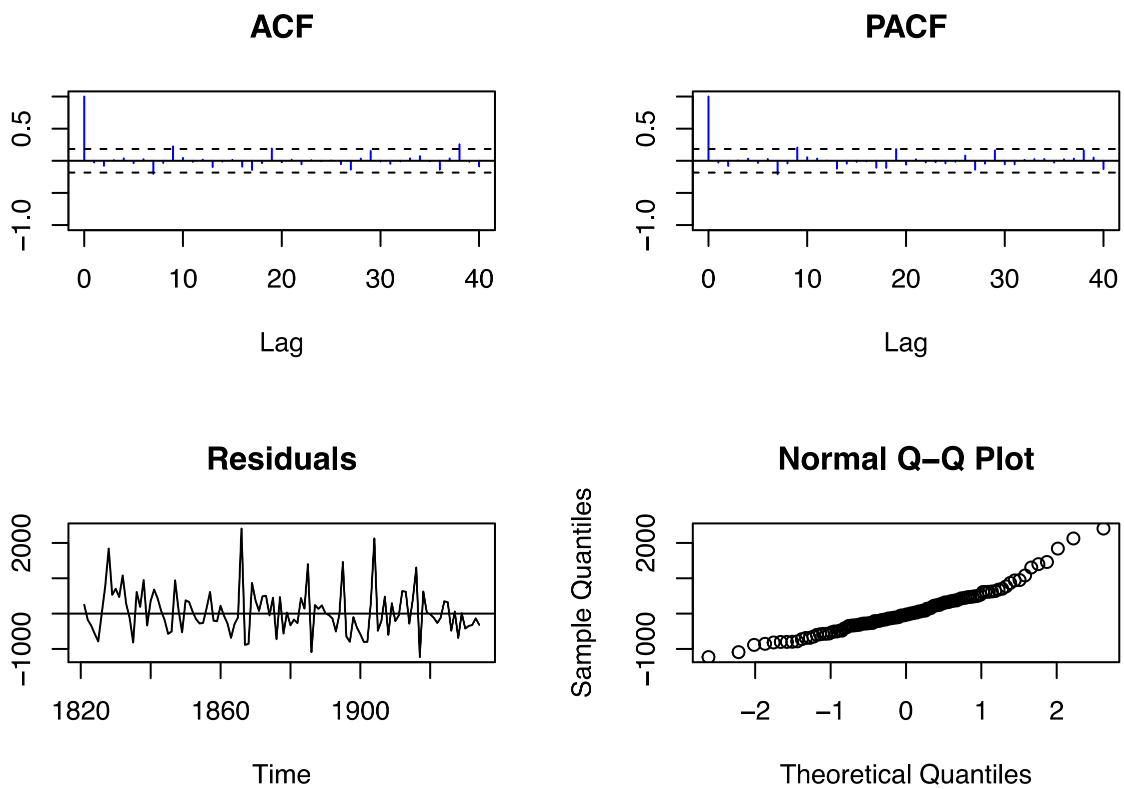



```
	Ljung-Box test

data:  Residuals from ARIMA(2,0,2) with non-zero mean
Q* = 12.986, df = 5, p-value = 0.02351

Model df: 5.   Total lags used: 10
```

```r
test(residuals(lynx_stl_2_2))
```

```
Null hypothesis: Residuals are iid noise.
Test                        Distribution Statistic   p-value
Ljung-Box Q                Q ~ chisq(20)     23.36    0.2714
McLeod-Li Q                Q ~ chisq(20)     20.34    0.4368
Turning points T   (T-74.7)/4.5 ~ N(0,1)        66    0.0523
Diff signs S       (S-56.5)/3.1 ~ N(0,1)        52     0.146
Rank P         (P-3220.5)/204.2 ~ N(0,1)      2843    0.0645
```

better




```
1 - pchisq(23.36,20-5)
```

```
[1] 0.07678976
```

```
1 - pchisq(20.34,20-5)
```

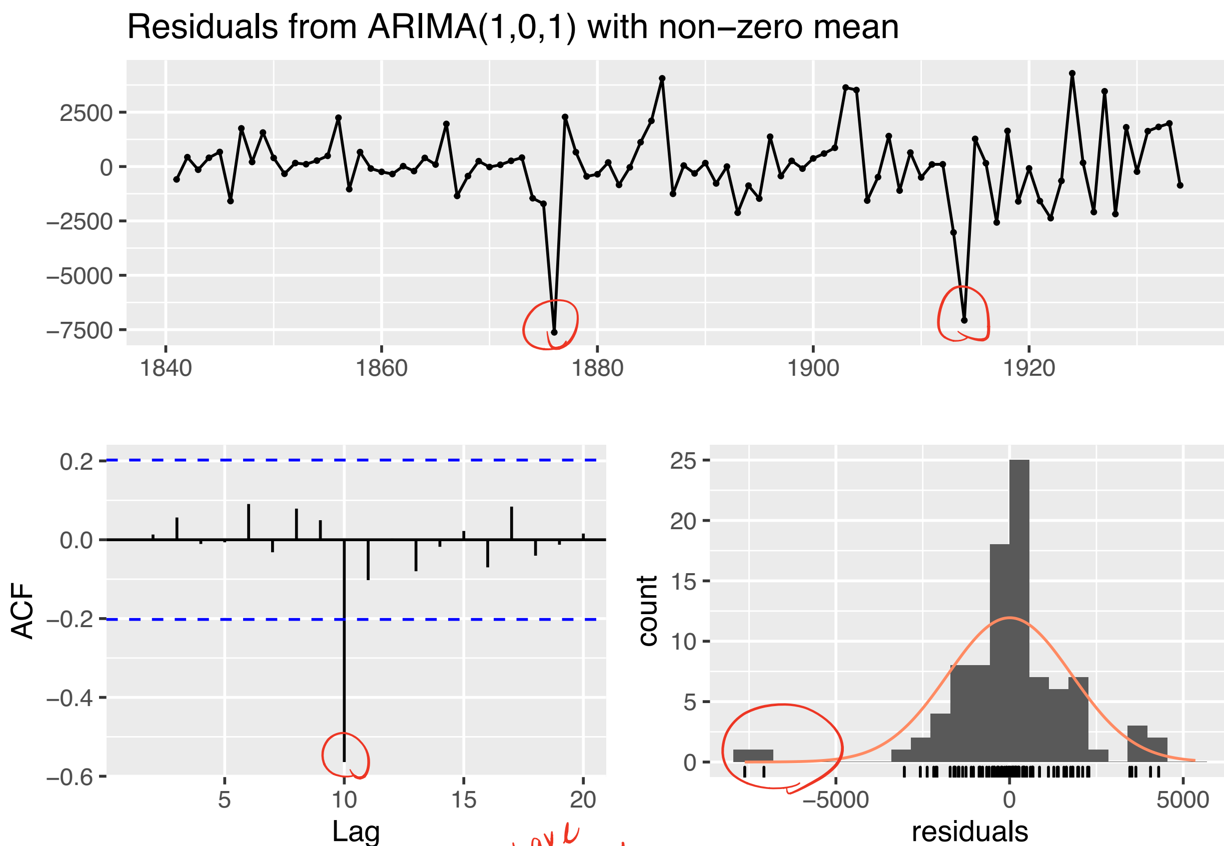
```
[1] 0.1592742
```

(2,2) NOT very good

```
### Using differencing
lynx_diff_arma<-arima(lynx_diff_10_2,c(1,0,1))
summary(lynx_diff_arma)
```

```
Call:
arima(x = lynx_diff_10_2, order = c(1, 0, 1))

Coefficients:
        ar1     ma1  intercept
      0.253  0.6490   182.2738
s.e.  0.132  0.1103   400.4635

sigma^2 estimated as 3141202:  log likelihood = -836.96,  aic = 1681.93

Training set error measures:
                   ME     RMSE      MAE      MPE     MAPE      MASE       ACF1
Training set 3.968555 1772.344 1157.686 26.81534 156.5061 0.8101618 0.00102156
```

```r
checkresiduals(lynx_diff_arma)
```

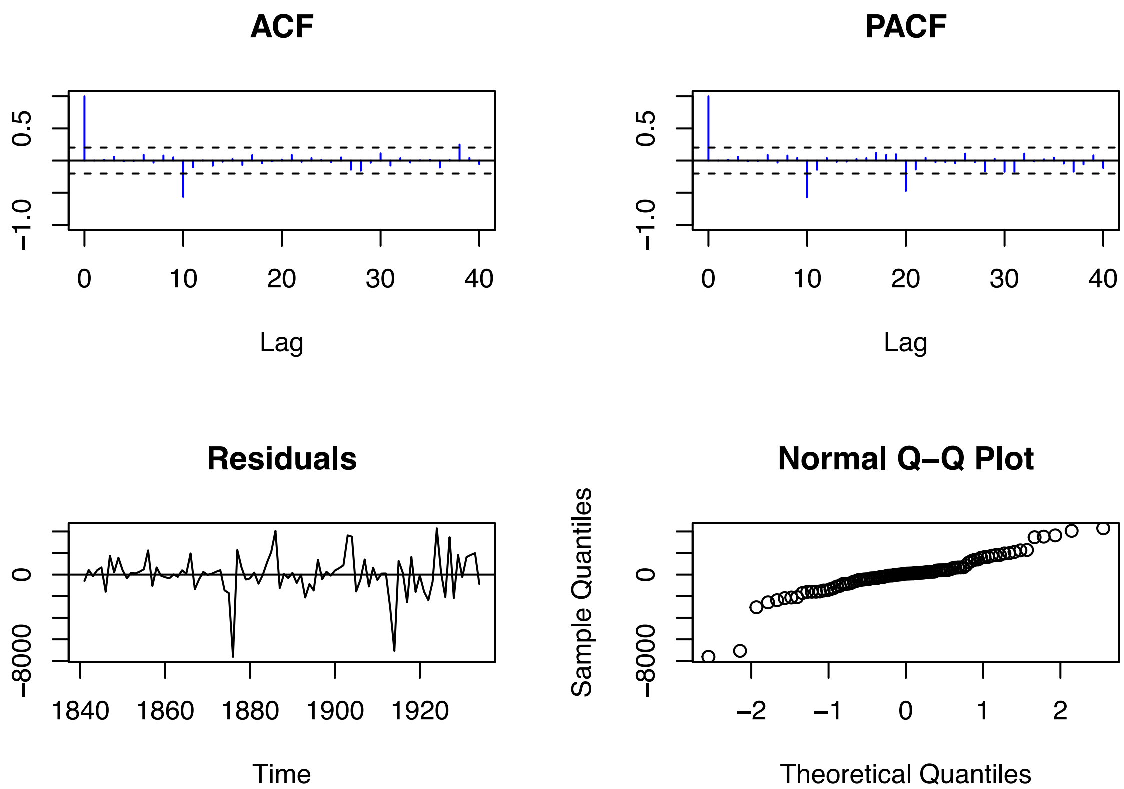


don't have
lag 10 diff

```
    Ljung-Box test

data:  Residuals from ARIMA(1,0,1) with non-zero mean
Q* = 36.377, df = 7, p-value = 6.153e-06

Model df: 3.   Total lags used: 10
```

```r
test(residuals(lynx_diff_arma))
```

```
Null hypothesis: Residuals are iid noise.
Test                        Distribution Statistic   p-value
Ljung-Box Q                Q ~ chisq(20)     39.96    0.0051 *
McLeod-Li Q                Q ~ chisq(20)     19.62    0.4821
Turning points T      (T-61.3)/4 ~ N(0,1)        63    0.6806
Diff signs S          (S-46.5)/2.8 ~ N(0,1)      50    0.2135
Rank P            (P-2185.5)/153.1 ~ N(0,1)    2070    0.4505
```




```
1 - pchisq(39.96,20-3)
```

```
[1] 0.001311118
```

```
1 - pchisq(19.62,20-3)
```

```
[1] 0.2941255
```

Forecasting

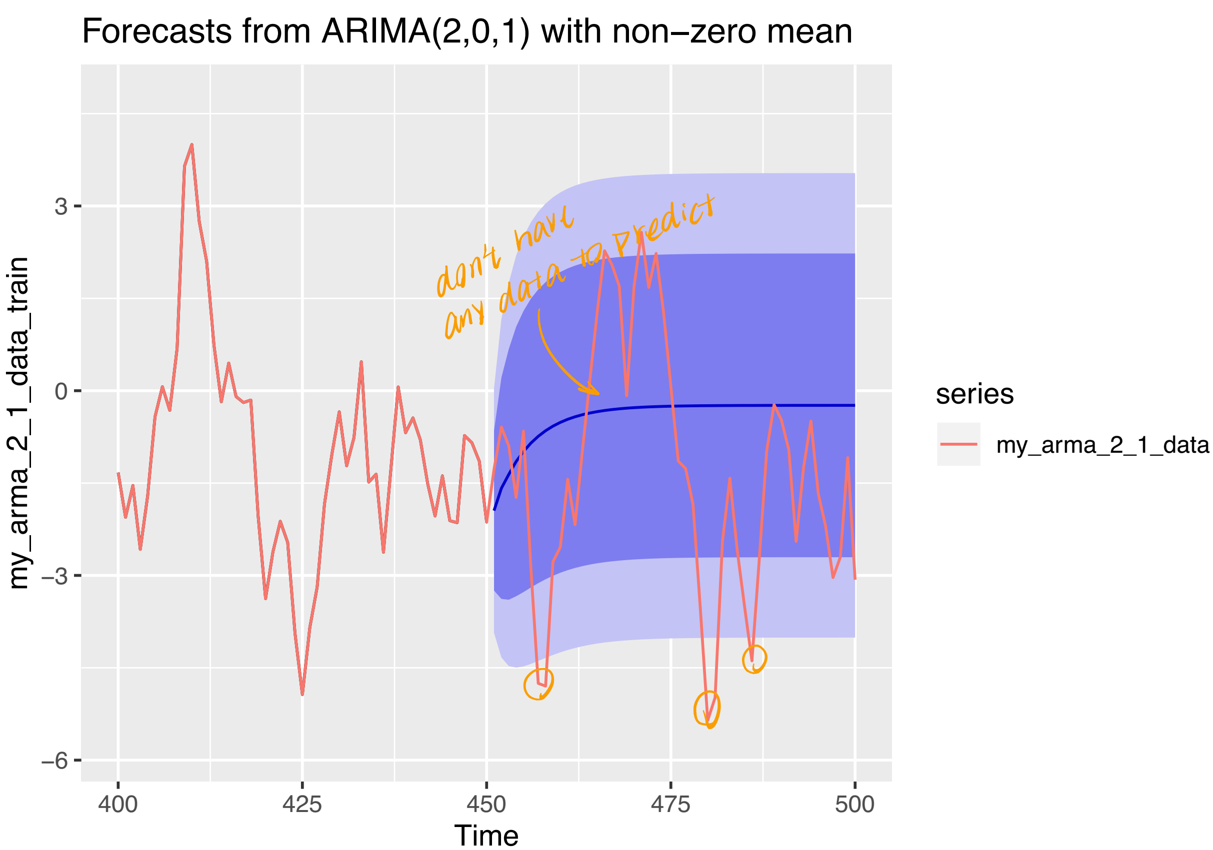
- First, let's see how our forecasting on the ARMA(2,1) data by fitting the ARMA model to only the first 450 observations and then examining the performance of the predictions on the last 50

```
my_arma_2_1_data_train <- window(my_arma_2_1_data,end=450)
my_arma_2_1_data_test <- window(my_arma_2_1_data,start=451,end=500)

my_arma_2_1_train_mod <- arima(my_arma_2_1_data_train,c(2,0,1))
myforecasts<-forecast::forecast(my_arma_2_1_train_mod,h=50)

autoplot(myforecasts) + autolayer(my_arma_2_1_data) + xlim(c(400,500))
```

```
Scale for 'x' is already present. Adding another scale for 'x', which will
replace the existing scale.
```

```
Warning: Removed 399 rows containing missing values (geom_path).
```

```
Warning: Removed 399 rows containing missing values (geom_path).
```




```
forecast_table<-print(myforecasts) %>% mutate(observed=my_arma_2_1_data_test,
                                              errors=`Point Forecast`-observed)
```

```
    Point Forecast     Lo 80      Hi 80     Lo 95      Hi 95
451     -1.9476985 -3.245901 -0.6494958 -3.933128 0.03773121
452     -1.5846613 -3.381277  0.2119546 -4.332348 1.16302561
453     -1.3600352 -3.394613  0.6745425 -4.471653 1.75158292
454     -1.1504955 -3.338224  1.0372334 -4.496338 2.19534715
455     -0.9877942 -3.270343  1.2947545 -4.478651 2.50306283
456     -0.8513689 -3.196017  1.4932788 -4.437198 2.73446026
457     -0.7407109 -3.125952  1.6445300 -4.388622 2.90720029
458     -0.6496735 -3.061822  1.7624755 -4.338737 3.03939005
459     -0.5752312 -3.005257  1.8547943 -4.291635 3.14117207
460     -0.5142007 -2.956154  1.9277522 -4.248845 3.22044400
461     -0.4642211 -2.914145  1.9857024 -4.211056 3.28261352
462     -0.4232720 -2.878530  2.0319862 -4.178266 3.33172146
463     -0.3897286 -2.848560  2.0691032 -4.150187 3.37073011
464     -0.3622492 -2.823476  2.0989779 -4.126371 3.40187283
465     -0.3397382 -2.802571  2.1230950 -4.106317 3.42684009
466     -0.3212972 -2.785208  2.1426132 -4.089523 3.44692864
467     -0.3061904 -2.770824  2.1584428 -4.075521 3.46314072
468     -0.2938149 -2.758933  2.1713031 -4.063887 3.47625778
469     -0.2836768 -2.749120  2.1817665 -4.054247 3.48689335
470     -0.2753718 -2.741033  2.1902899 -4.046276 3.49553227
471     -0.2685683 -2.734376  2.1972398 -4.039696 3.50255981
472     -0.2629948 -2.728901  2.2029116 -4.034273 3.50828358
473     -0.2584291 -2.724401  2.2075433 -4.029808 3.51295021
```

```
474     -0.2546888 -2.720705  2.2113278 -4.026136 3.51675818
475     -0.2516248 -2.717671  2.2144215 -4.023117 3.51986763
476     -0.2491148 -2.715181  2.2169515 -4.020638 3.52240816
477     -0.2470585 -2.713138  2.2190211 -4.018602 3.52448485
478     -0.2453741 -2.711463  2.2207146 -4.016931 3.52618304
479     -0.2439942 -2.710089  2.2221005 -4.015561 3.52757216
480     -0.2428638 -2.708962  2.2232350 -4.014436 3.52870877
481     -0.2419377 -2.708039  2.2241637 -4.013514 3.52963895
482     -0.2411791 -2.707282  2.2249241 -4.012759 3.53040035
483     -0.2405576 -2.706662  2.2255468 -4.012139 3.53102367
484     -0.2400486 -2.706154  2.2260567 -4.011631 3.53153402
485     -0.2396315 -2.705737  2.2264743 -4.011215 3.53195191
486     -0.2392899 -2.705396  2.2268164 -4.010874 3.53229413
487     -0.2390100 -2.705116  2.2270965 -4.010594 3.53257438
488     -0.2387807 -2.704887  2.2273259 -4.010365 3.53280391
489     -0.2385929 -2.704700  2.2275139 -4.010178 3.53299191
490     -0.2384390 -2.704546  2.2276678 -4.010024 3.53314588
491     -0.2383130 -2.704420  2.2277939 -4.009898 3.53327201
492     -0.2382097 -2.704317  2.2278972 -4.009795 3.53337531
493     -0.2381251 -2.704232  2.2279818 -4.009710 3.53345994
494     -0.2380558 -2.704163  2.2280511 -4.009641 3.53352925
495     -0.2379991 -2.704106  2.2281079 -4.009584 3.53358603
496     -0.2379526 -2.704060  2.2281544 -4.009538 3.53363255
497     -0.2379145 -2.704021  2.2281925 -4.009500 3.53367065
498     -0.2378833 -2.703990  2.2282237 -4.009468 3.53370186
499     -0.2378577 -2.703965  2.2282493 -4.009443 3.53372743
500     -0.2378368 -2.703944  2.2282702 -4.009422 3.53374837
```

nearly same

converge to sample mean (close to 0)

```
forecast_table%>% dplyr::slice(1:15) %>%  select(c(1,6,7,2:5)) %>%
  round(2) %>% kable()
```

Point Forecast	observed	errors	Lo 80	Hi 80	Lo 95	Hi 95
-1.95	-1.28	-0.67	-3.25	-0.65	-3.93	0.04
-1.58	-0.59	-0.99	-3.38	0.21	-4.33	1.16
-1.36	-0.89	-0.47	-3.39	0.67	-4.47	1.75
-1.15	-1.73	0.58	-3.34	1.04	-4.50	2.20
-0.99	-0.66	-0.33	-3.27	1.29	-4.48	2.50
-0.85	-2.87	2.02	-3.20	1.49	-4.44	2.73
-0.74	-4.75	4.01	-3.13	1.64	-4.39	2.91
-0.65	-4.80	4.15	-3.06	1.76	-4.34	3.04
-0.58	-2.78	2.20	-3.01	1.85	-4.29	3.14
-0.51	-2.54	2.03	-2.96	1.93	-4.25	3.22
-0.46	-1.44	0.97	-2.91	1.99	-4.21	3.28
-0.42	-2.18	1.75	-2.88	2.03	-4.18	3.33
-0.39	-0.97	0.58	-2.85	2.07	-4.15	3.37
-0.36	0.16	-0.52	-2.82	2.10	-4.13	3.40
-0.34	1.28	-1.62	-2.80	2.12	-4.11	3.43

CI very wide

```
forecast_table%>%  mutate(outside95=I(observed<`Lo 95` | observed>`Hi 95`)) %>%
  count(outside95)
```

count outside

```
# A tibble: 2 x 2
  outside95      n
  <I<lgl>>  <int>
1 FALSE         45
```

```
2 TRUE              5
```