simulation exercise

linyiguo

2019.5.24

我在想借助 **R Markdown** 这个工具来记录我平时学习的心路历程。在做研究的过程中，势必会产生很多感想，不论是感性上还是理性上，我都愿意把这些 ideas 记录下来，这样也许更有助于我理清自己的思路，不至于脑子里总是一盆浆糊。在以后，我可能更想用英语来写，但目前个人能力有限，就慢慢来吧。

After meeting with Aaron， I think I still need to spend more time on research, although I have other plans on working out and ielts. But research should be my first goal, given my current situation. Well, let’s say what we were talking about in this morning: I am trying to simulate some data from different models and these data will be used in my later research(since I do not have real data now).

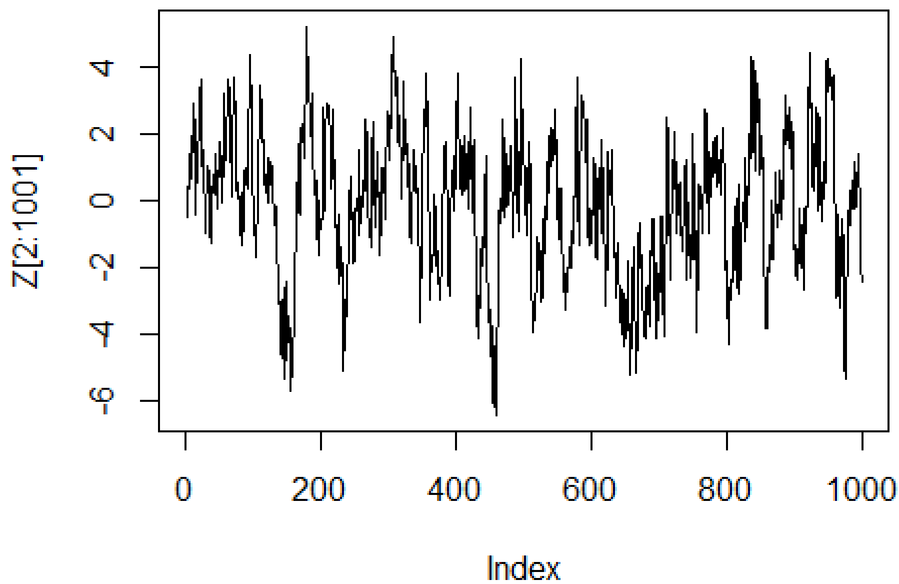
Auto-regressive model

Let’s say we want to simulate data corresponding to model *AR(1)*:

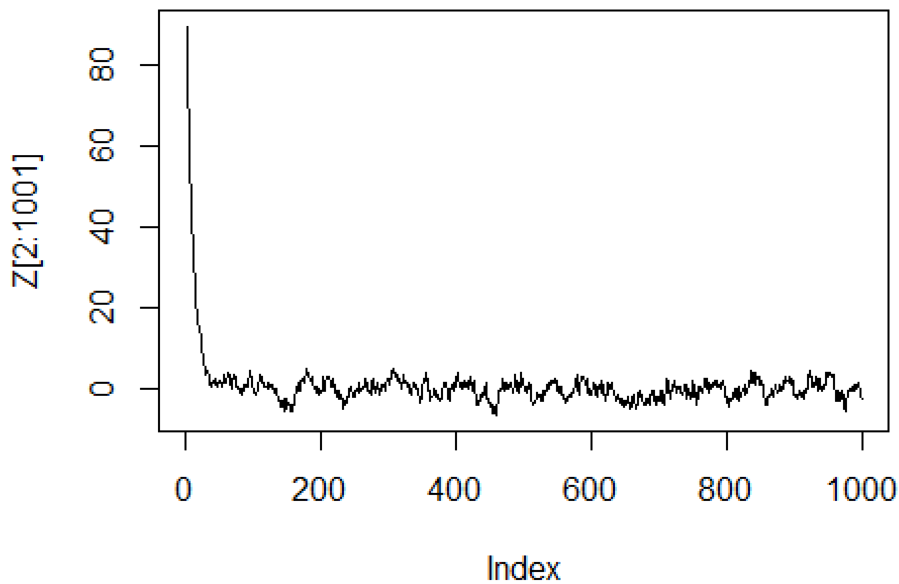
$$Z_t = 0.9 * Z_{t-1} + \epsilon_t$$

where $\epsilon \sim N(0,1), Z_0 = 1$

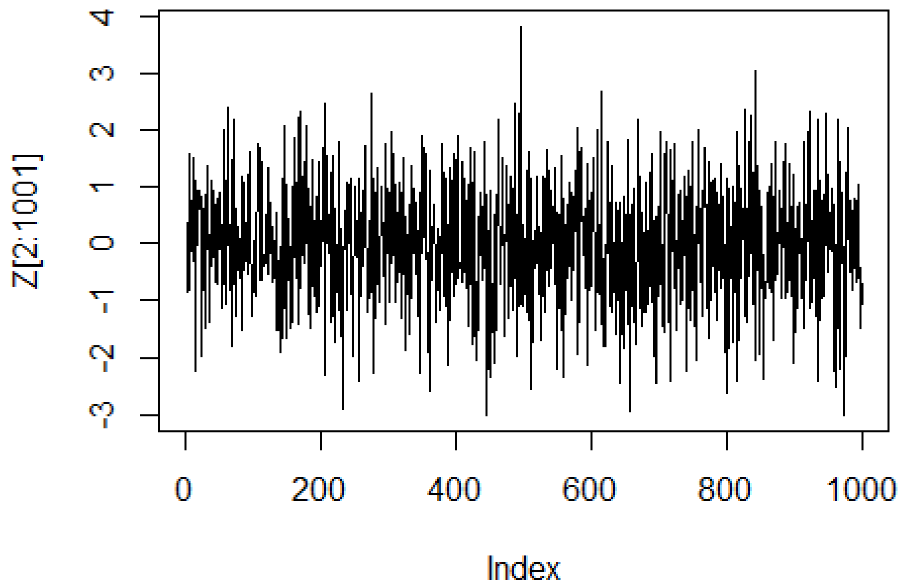
```
set.seed(1)
epsilon <- rnorm(1000)
Z <- rep(0,1001)
Z[1] <- 1
for(i in 2:1001)  Z[i] <- 0.9*Z[i-1] + epsilon[i-1]
plot(Z[2:1001], type="l")
```



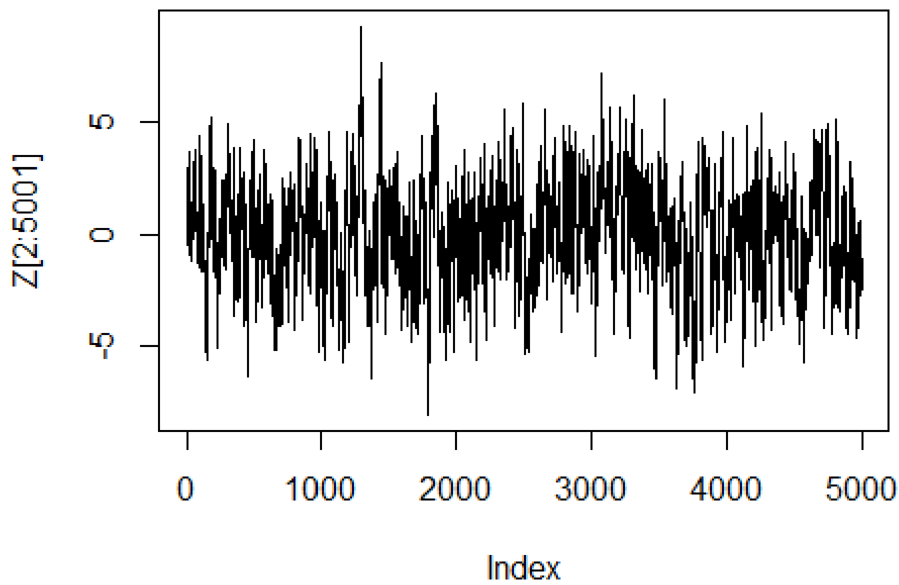

```r
set.seed(1)
epsilon <- rnorm(1000)
Z <- rep(0,1001)
Z[1] <- 100
for(i in 2:1001)  Z[i] <- 0.9*Z[i-1] + epsilon[i-1]
plot(Z[2:1001], type="l")
```




```r
set.seed(1)
Z <- rep(0,1001)
Z[1] <- 1
for(i in 2:1001)  Z[i] <- 0.01*Z[i-1] + epsilon[i-1]
plot(Z[2:1001], type="l")
```




```r
set.seed(1)
Z <- rep(0,5001)
e <- rnorm(5000)
Z[1] <- 1
for(i in 2:5001)  Z[i] <- 0.9*Z[i-1] + e[i-1]
plot(Z[2:5001], type="l")
```



comment: 从上面的图像可以看出，，振幅就越大；同时，初始值Z_0不会对最终的结果产生影响。我在想一个问题，为什么ϕ_1的值和 ts 的 amplitude 有关？ Intuitively，Z_{t-1} is also a variable, which has a normal dist'n as well(since we assume noises are normal), so Z_t is the sum of several normal dist'n, but is mainly determined by the first ones(since $\phi < 0$, efficients converge to 0).

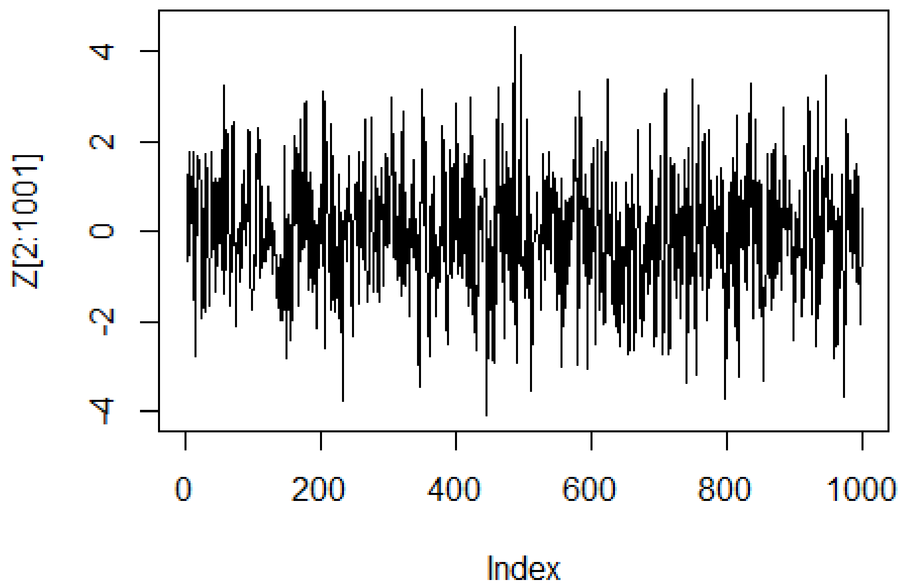
Moving-average model

Here, we simulate data to fit the model *MA(1)*:

$$Z_t = \epsilon_t + 0.9 * \epsilon_{t-1}$$

where $\epsilon \sim N(0,1), Z_0 = 1$

```
set.seed(1)
Z <- rep(0,1001)
epsilon <- rnorm(1001)
Z[1] <- 0
for(i in 2:1001) Z[i] = epsilon[i]+0.9*epsilon[i-1]
plot(Z[2:1001],type="l")
```



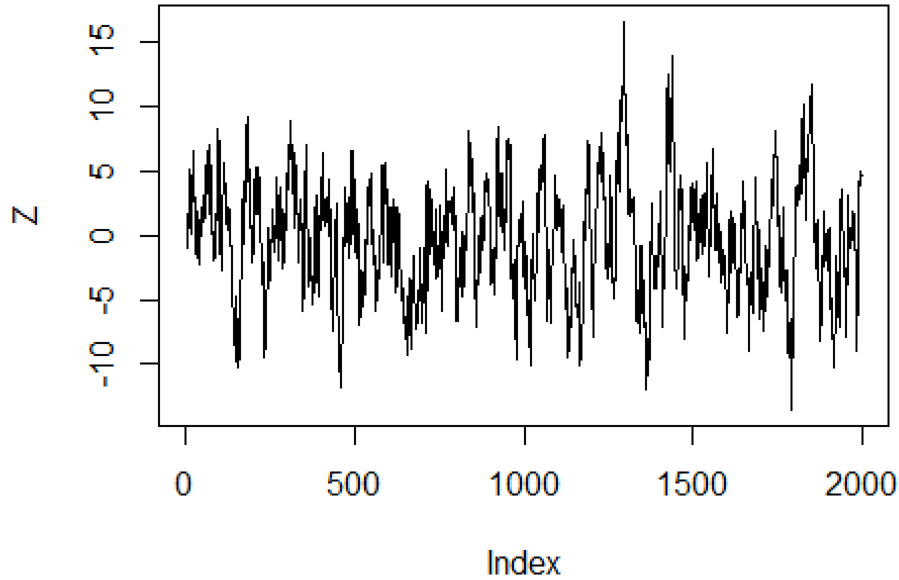
Auto-regressive moving-average model

Let's say we want to simulate data according to an *ARMA(1,1)* model:

$$Z_t - 0.9 * Z_{t-1} = \epsilon_t + 0.9 * \epsilon_{t-1}$$

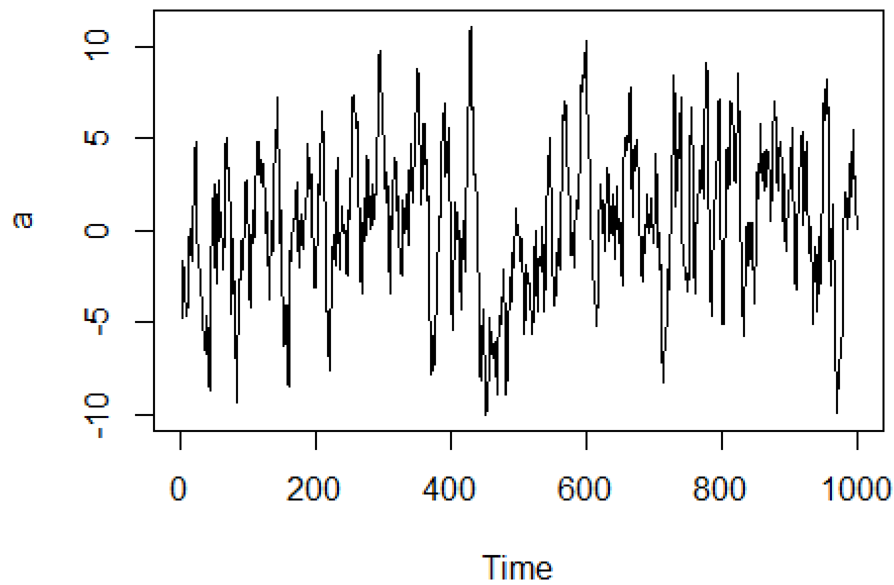
where $\epsilon \sim N(0,1), Z_0 = 1$

```
set.seed(1)
Z <- rep(0,2001)
e <- rnorm(2000)
for(i in 2:2001) Z[i] = 0.9*Z[i-1]+e[i]+0.9*e[i-1]
plot(Z,type="l")
```



[update 2019.5.26] 其实我发现 simulate 数据并不是我想的这么麻烦，在 R 中有一些 code 可以帮助我们很轻易地模拟得到想要的数据，比如：

```
a <- arima.sim(model=list(ar=c(0.9),ma=c(0.9)), n=1000,innov = rnorm(10
00))
plot(a)
```



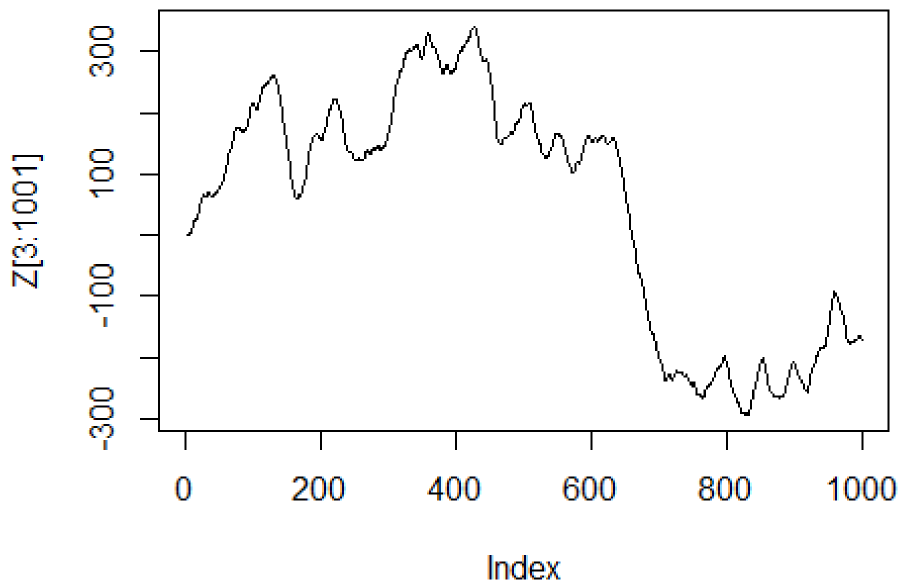
Auto-regressive integrated moving-average model

Suppose we want to simulate data to fit *ARIMA(1,1,1)* model:

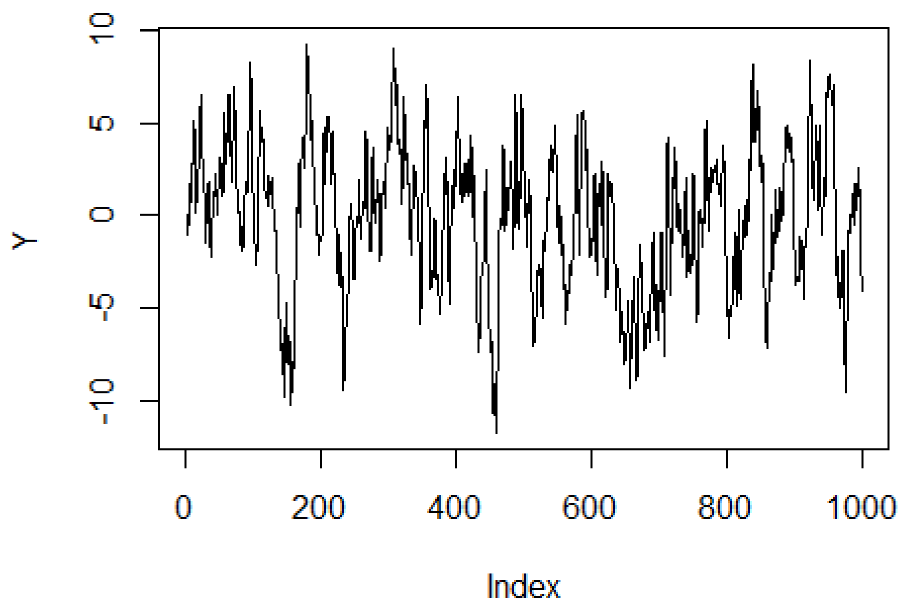
$$(1-0.9B)(1-B)Z_t = \epsilon_t + 0.9 * \epsilon_{t-1}$$

where $\epsilon \sim N(0,1), Z_0 = Z_1 = 1$

```
set.seed(1)
Z <- rep(0,1001)
e <- rnorm(1000)
Z[1] <- 1
Z[2] <- 1
for (i in 3:1001) Z[i] <- 1.9*Z[i-1]-0.9*Z[i-2]+e[i-1]+0.9*e[i-2]
plot(Z[3:1001],type="l")
```




```r
Y <- Z[2:1001] - Z[1:1000]
plot(Y, type="l")
```



comment:

Seasonal Autoregressive Moving-average Model

Now, we consider $SARIMA$ model， which can be viewed as an expanded model of $ARIMA$. Let's say our model is $ARIMA(1,1,1) * (1,1,1)_4$, that is,

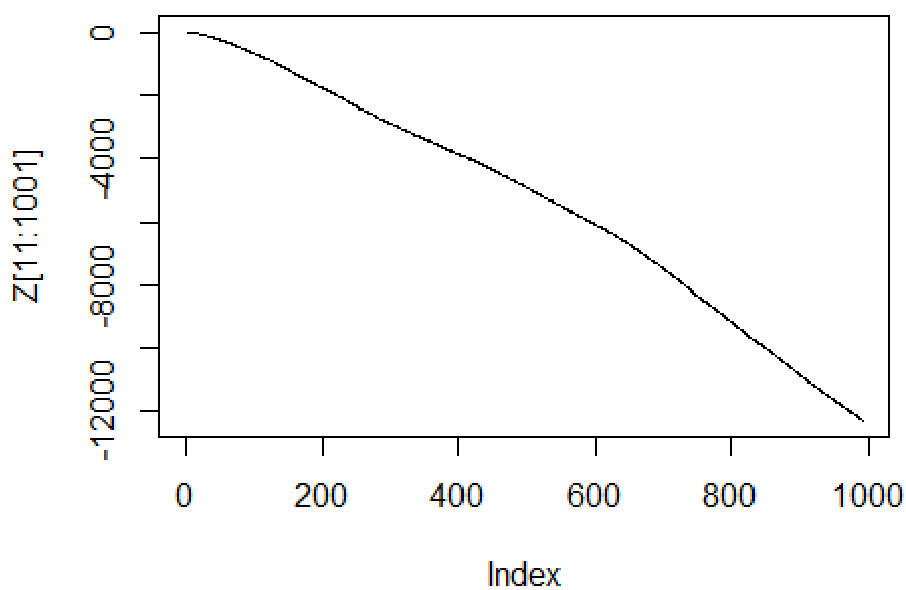
$$(1-0.9B)(1-0.9B^4)(1-B)(1-B^4)Z_t = (1-0.9B)(1-0.9B^4)\epsilon_t$$

After expansion, we have

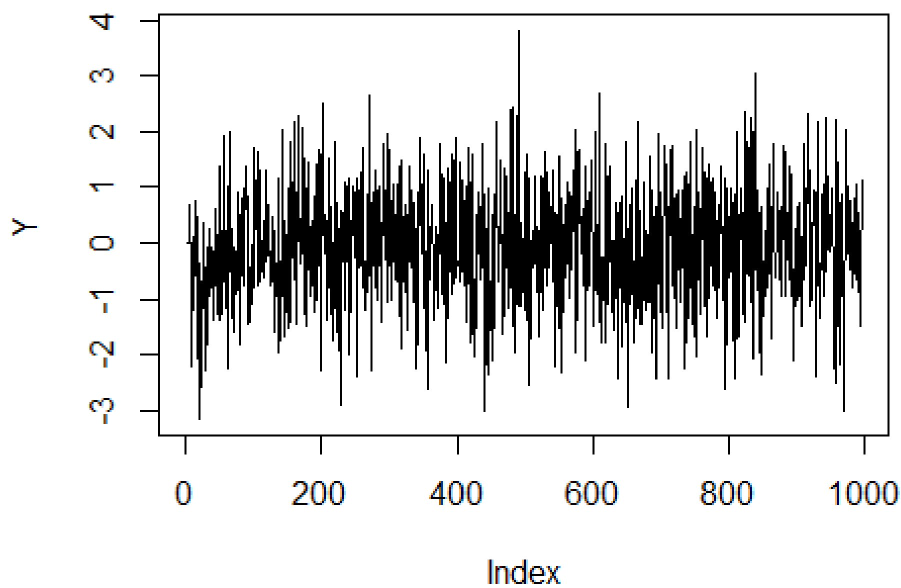
$$(1-1.9B+0.9B^2-1.9B^4+3.61B^5-1.71B^6+0.9B^8-1.71B^9+0.81B^{10}) * Z_t$$
$$= (1-0.9B-0.9B^4+0.81B^5) * \epsilon_t$$

where$\epsilon \sim N(0,1)$, we take $Z_1, Z_2, \ldots, Z_{10}$ randomly from normal distribution N(0,1).

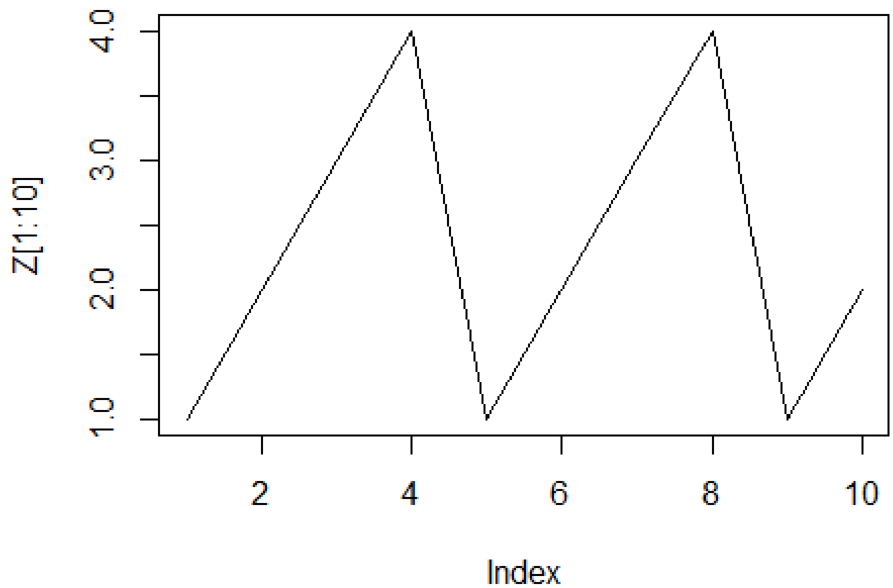
```
set.seed(1)
Z <- rep(0,1001)
e <- rnorm(1001)
#Z[1:10] <- rnorm(10)
Z[1:10] <- c(1,2,3,4,1,2,3,4,1,2)
for (i in 11:1001) {
  Z[i] <- e[i]-0.9*e[i-1]-0.9*e[i-4]+0.81*e[i-5]+1.9*Z[i-1]-0.9*Z[i-2]+
1.9*Z[i-4]-3.61*Z[i-5]+1.71*Z[i-6]-0.9*Z[i-8]+1.71*Z[i-9]-0.81*Z[i-10]
}
plot(Z[11:1001],type="l")
```



```
Y <- Z[6:1001]-Z[5:1000]-Z[2:997]+Z[1:996]
plot(Y,type="l")
```




```
plot(Z[1:10],type="l")
```



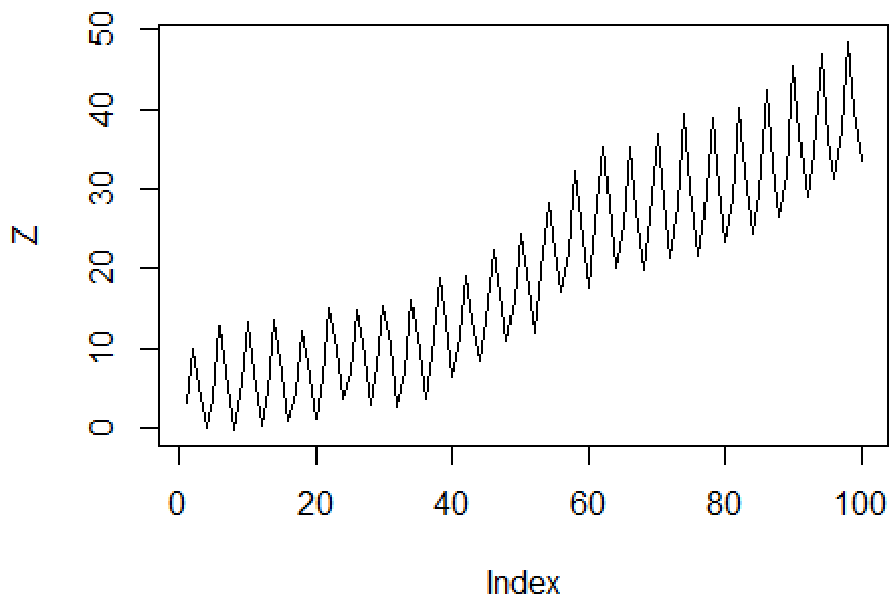
Well, this is a little weird, cause the curve I expect should be with obvious seasonal fluctuations. Let's try another *SARIMA* model:

$$(1-B)(1-B^4)Z_t = (1-0.4B)(1-0.6B^4)\epsilon_t$$

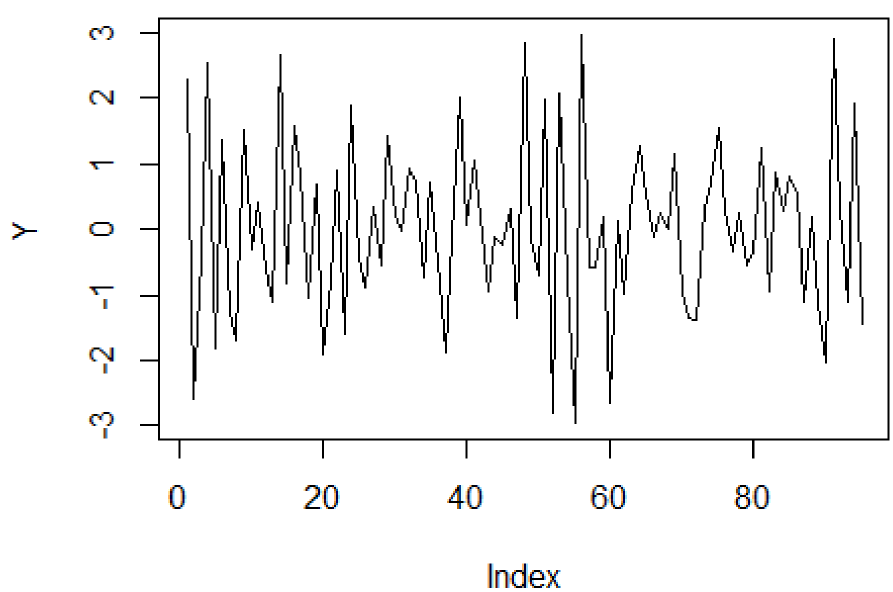
which is equal to

$$Z_t = Z_{t-1} + Z_{t-4} - Z_{t-5} + \epsilon_t - 0.4 * \epsilon_{t-1} - 0.6 * \epsilon_{t-4} + 0.24 * \epsilon_{t-5}$$

```
Z <- rep(0,100)
e <- rnorm(100)
Z[1:5] <- c(3,10,5,0.1,3.5)
for (i in 6:100)   Z[i] <- Z[i-1]+Z[i-4]-Z[i-5]+e[i]-0.4*e[i-1]-0.6*e[i
-4]+0.24*e[i-5]

plot(Z,type="l")
```

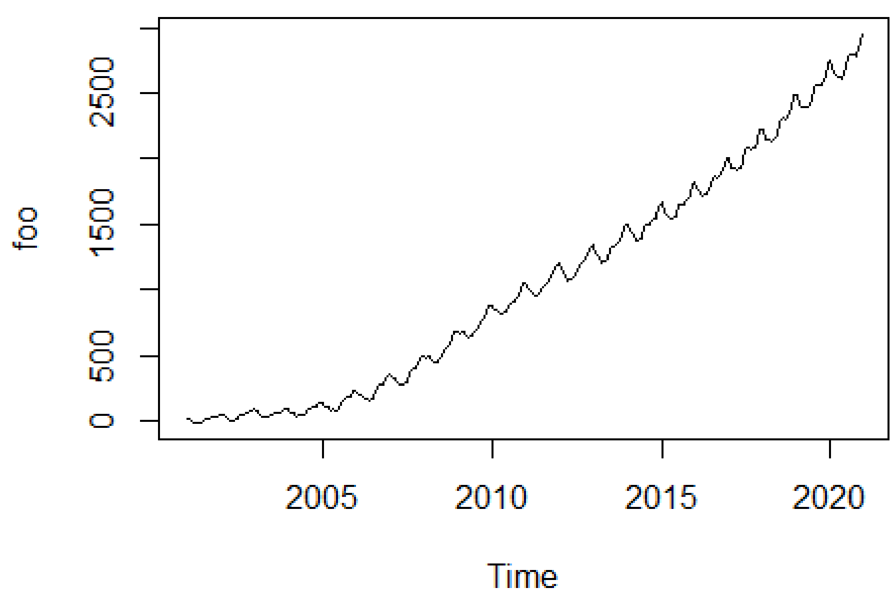



```r
Y <- Z[6:100]-Z[5:99]-Z[2:96]+Z[1:95]
plot(Y,type="l")
```



I saw a method to simulate data for a known dist'n(source), let's try:

```
library(forecast)

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff

set.seed(1)
model <- Arima(ts(rnorm(24000),freq=12), order=c(0,1,1), seasonal=c(0,1,
1),fixed=c(theta=0.313, Theta=0.817))
foo <- simulate(model,nsim = 240)
plot(foo,type="l")
```



```
foo

##              Jan          Feb          Mar          Apr          Ma
y
## 2001   10.3408032   11.2264006   -0.9604257  -14.8070713  -13.577251
3
## 2002   50.3907762   47.6905099   23.2798942   -1.8270887   -0.901133
5
## 2003   84.4329722   69.5446959   48.6671463   22.8438096   24.718462
2
## 2004   96.8827042   68.1862340   64.5230956   37.1204034   43.996242
7
## 2005  138.6868890  103.2394906  109.1220867   79.7727820   87.352252
1
## 2006  233.8640890  194.4476100  192.8628578  163.9490385  165.399688
3
## 2007  348.3970770  316.0691704  317.1798696  289.8313020  281.908986
1
## 2008  493.5886701  475.7920618  487.7115568  463.7267696  451.259776
2
## 2009  676.5103128  660.5762374  675.2845740  651.1063055  639.490000
8
## 2010  878.5850876  845.7284190  854.2341604  824.1072746  816.815641
9
## 2011 1049.2271060 1000.3351041  993.5976658  955.4308774  951.592053
8
## 2012 1198.1862439 1136.2601561 1112.9924479 1070.4753178 1078.260328
2
## 2013 1347.2836581 1284.6252336 1252.4697929 1210.7021340 1221.882267
5
## 2014 1501.7267257 1446.8332758 1418.3647548 1375.6020213 1375.737216
4
## 2015 1660.1586641 1600.0516280 1576.2705779 1538.8688000 1538.447239
1
## 2016 1825.3464651 1758.1330828 1745.2148310 1720.1782383 1729.555636
5
## 2017 2006.9801360 1935.2096809 1932.4426273 1914.4567240 1930.408822
9
## 2018 2225.5190303 2146.9179456 2143.9723535 2127.2963478 2140.823137
7
## 2019 2488.8387882 2402.7384114 2395.8135960 2384.9781634 2390.651539
5
## 2020 2745.1812622 2649.9981864 2634.0703994 2622.4640545 2616.096318
5
##              Jun          Jul          Aug          Sep          Oc
t
## 2001  -12.2068474   -0.7280686   16.9448287   21.8299733   27.820928
7
## 2002    2.3337353   18.5976583   42.8542247   45.1201207   53.978382
5
```

```
## 2003   29.5488333   40.1019091   62.9246449   54.9975986   68.008791
3
## 2004   39.1243511   51.9636292   92.8812687   94.3217415  108.260445
2
## 2005   77.2960645   93.4422099  150.1555975  169.1391994  180.050281
8
## 2006  157.5114997  171.2907353  237.9187892  269.6925844  273.012664
7
## 2007  276.8036169  289.1070811  350.2236738  387.5357566  401.967440
0
## 2008  453.3386509  475.2320355  521.4923505  557.6723219  582.019189
0
## 2009  647.0350792  680.8852711  714.2948435  743.4982193  773.634679
9
## 2010  832.5891847  872.3028533  891.2863807  914.6763674  948.027106
5
## 2011  969.7088605 1016.6779244 1032.6215934 1061.7735120 1099.753181
8
## 2012 1087.7299796 1148.0961755 1168.3381098 1198.5489140 1236.733013
1
## 2013 1234.3568392 1310.1042178 1326.9537276 1342.0105923 1374.556429
0
## 2014 1394.2791691 1478.9510289 1492.1753387 1492.9543660 1522.955196
0
## 2015 1559.8835779 1643.9110050 1658.0537242 1653.4728151 1685.835231
6
## 2016 1763.8054172 1851.8340570 1869.1976235 1852.9965589 1881.653529
6
## 2017 1970.0569893 2064.1551227 2087.8783668 2066.0838530 2090.405801
2
## 2018 2180.4136732 2277.9086180 2310.8898600 2296.7906261 2318.625893
7
## 2019 2431.5680263 2525.0783066 2568.2517211 2556.6222628 2562.950300
8
## 2020 2661.2222267 2755.5171362 2800.4116153 2790.7061190 2783.052274
4
##               Nov          Dec
## 2001   24.4657549   36.3180075
## 2002   55.1437472   75.8807984
## 2003   71.1121628   93.5097456
## 2004  114.5743881  141.0896547
## 2005  187.6891221  230.0530013
## 2006  284.0037685  336.3064711
## 2007  427.8087441  486.0252321
## 2008  619.2790918  679.1338747
## 2009  820.7172141  881.9302128
## 2010  986.2552497 1049.0367844
## 2011 1126.1721616 1189.2242164
## 2012 1263.1629425 1329.4626998
## 2013 1399.9202331 1475.3637103
```

```
## 2014 1543.1112577 1630.0228110
## 2015 1707.4959638 1803.6917045
## 2016 1911.2588828 2000.7056088
## 2017 2129.5361416 2224.4124532
## 2018 2373.1193621 2485.5844926
## 2019 2617.7381824 2734.7282969
## 2020 2833.5408806 2949.9900040
```

```
summary(model)
```

```
## Series: ts(rnorm(24000), freq = 12)
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##    ma1   sma1
## 0.313  0.817
##
## sigma^2 estimated as 31.02:  log likelihood=-75233.5
## AIC=150469   AICc=150469   BIC=150477.1
##
## Training set error measures:
##                        ME     RMSE      MAE      MPE     MAPE     MA
SE
## Training set 0.0001462373 5.567472 4.429197 256.4385 4375.639 3.9162
37
##                     ACF1
## Training set -0.6522161
```

Comment: our final SARIMA model is

$$(1-B)(1-B^{12})Z_t = (1-0.5B)(1-0.5B^{12})\epsilon_t$$

. If we take one unit as one year(12 observations), then we have ten years' data.

—UPDATE 2019.5.26— I am trying to check the method from the answer, but:

```
# install.packages("devtools")
library("devtools")
devtools::install_github("smac-group/gmwm")

# Set seed for reproducibility
set.seed(1)

# Specify a SARIMA(0,1,1)(0,1,1)[12]
mod = SARIMA(i=1, ma=.5, si = 1, sma = .5, s = 12, sigma2 = 1.5)

# Generate the data
xt2 = gen.gts(mod, 1e3)

# Validate output
arima(xt2, order=c(0,1,1), seasonal=list(order=c(0,1,1), period = 12))
```

end(perhaps ?)

我感觉数据这块到这这儿就差不多了吧(**too young too naive**)，虽然最后 *SARIMA* 花了很长时间，走了很多弯路，而且最后的数据我现在还不是很确定能不能用，但是也只能先暂且相信网上的大牛们和自己的判断了。前路茫茫啊，年轻人，不要气馁，继续努力！感觉不能一直给自己说慢慢来，因为感觉目前自己的节奏真的有点太悠闲自在了些...不管怎样，还是要相信自己，坚持你的梦想，朝着梦想前进！别人能做到，为什么我不能呢！多思考，年轻人 :) Cheers ~