Time Series Midterm Review

David Josephs

2019-06-28

Contents

Chapter 1

Preparing for the test

Chapter 2

Expected Values, Variance, etc.

2.1 Notation and Some Definitions

In time series, we denote the response variable as X_t, and clearly the explanatory variable as t. To denote a specific point in time, let us say $t = 9$, we would say X_9.

The collection of all X_n is a stochastic process known as a time series.

2.1.1 Realizations and Ensembles

A **realization** is a particular instance of a time series. It is one of the infinitely many possible time series that could have been observed. As a time series is a stochastic process, each one could be entirely different. The **realization** Is the one that was *actually* observed.

Sometimes a realization is the only realization possible. We can theorize what an infinite population of those realizations, but it is often the case that there is only one.

An **ensemble** is the theoretical totality of all possible realizations. It is the *ensemble* of *realizations*. We can think of that as the *population* of realizations. When we are thinking of means and variances, we think of them in terms of the ensemble, even though we may only be able to look at a single realization.

2.2 Population Means and Variances

It is important to note that when dealing with an ensemble of realizations, each time point is allowed to have its own mean and variance.

2.2.1 The Mean of X_t

μ_t is the mean of all possible realizations of X_t ***for a fixed t***.

2.2.2 Variance of X_t

${\sigma_t}^2$ is the variance of all possible realizations of X_t ***for a fixed t***

2.3 Estimating Mean and Variance for Multiple Realizations

Let us say we have n realizations of X_t at time t. The average can be then written as

$\hat{\mu_t} = \frac{1}{n} \sum_{i=1}^{n} X_{t_n}$

Variance is calculated the same way. We are making sample means and sample variances. If we assume the population is normally distributed, this allows us to take confidence intervals on the mean of our set of realizations.

However, We have a problem: What to do when we only have one realization?

2.4 Expected Value

In short, the expected value $(E[X])$ of a random variable (RV) denoted by X is the **mean** or more intuitively the ***long run average*** of the event that variable represents.

2.4.1 Discrete RVs

A discrete RV is an RV in which X cannot take on any value, it has specific values it can exist at and that is it.

The formula for EV of a discrete RV is as follows:

$$E\left[X\right] = \sum XP\left(X\right) = \mu$$

2.4.2 Continuous RVs

A continuous RV has instead of discrete values a ***probability distribution/density function,*** $f(x)$.

The formula for EV of a continuous RV is as follows:

$$E\left[X\right] = \int_a^b xf\left(x\right)dx = \mu$$

Note that is directly analagous to the discrete RV, and that a and b can span to $\pm\infty$

Important Note

$$E\left[X\right] = \mu$$

That is the expected value of a RV is equivalent to the population mean

2.4.3 Some Rules

Let a and b be constants

$$E\left[a\right] = a$$

$$E[aX] = aE[X] = a\mu$$

$$E[aX + b] = aE[X] + E[b] = aE[x] + b = a\mu + b$$

Knowing these rules, lets try out a challenge:

2.4.4 A challenge in Expected Values

Find $E[\bar{x}]$ where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

E.G find the expected value of an average

2.4.5 A solution

First, we can factor out the 1/n, so we have

$$\frac{1}{n}E\left[\sum_{i=1}^{n} x_i\right]$$

then we can say that the sum is just $x_1 + x_2 + x_3 + ... + x_n$. This would lead us to the conclusion that

$$E[\bar{x}] = \frac{1}{n}\sum_{i=1}^{n} E\left[x_i\right]$$

If we let each x_i be a constant then we have:

$$E[\bar{x}] = \frac{1}{n}\sum_{i=1}^{n} x_i = \bar{x}$$

This makes sense as $\bar{x}$ is the average value of x, and so the expected value of x for a very high n is the average value of x

2.5 Variance and Covariance

2.5.1 Variance

The variance is the *dispersion*. Variance is defined as

$$\mathrm{Var}\left(X\right) = E\left[\left(X-\mu\right)^2\right] = \int_{-\infty}^{\infty}\left(x-\mu\right)^2 f\left(x\right)dx = \sigma^2$$

$$\widehat{\mathrm{Var}\left(X\right)} = \sum\left(x_i - \bar{x}\right)^2$$

2.5.2 Covariance

Assume we have two RVs, X and Y:

$$\mathrm{Cov}(X,Y) = E\left[\left(X-\mu_x\right)\left(Y-\mu_y\right)\right]$$

$$\widehat{\mathrm{Cov}\left(X,Y\right)} = \sum\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)$$

The variance is, in essence, the sum of cross products.

2.5.3 Some generalizations on covariance:

If y tends to increase with x, then $\mathrm{Cov}(X,Y) > 0$

If Y tends to decrease with X, then $\mathrm{Cov}(X,Y)$ is ***Less than 0***

If it appears as a random cloud, then $\mathrm{Cov}(X,Y)$ is approximately 0

2.5.4 A challenge

Let X be a RV. What is $\mathrm{Cov}(X,X)$?

2.5.5 A solution

$$\mathrm{Cov}(X,X) = E\left[\left(X-\mu_x\right)\left(X-\mu_X\right)\right]$$

We see that the arguments in the left and right parens are exactly the same. Thus, we can rewrite the equation as:

$$\mathrm{Cov}(X,X) = E\left[\left(X-\mu_x\right)^2\right] = \mathrm{Var}(X)$$

2.5.6 A note on covariance and correlation

Correlation is covariance but standardized.

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\mathrm{SD}(X)\mathrm{SD}(Y)}$$

$$= \frac{E[X-\mu_x)(Y-\mu_y)]}{\sigma_x\sigma_y}$$

$$\widehat{\mathrm{Corr}(X,Y)} = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{ns_xs_y}$$

Chapter 3

A Brief Discussion of Stationarity

In order for a time series to be considered stationary, it must satisfy ***three*** conditions:

- Constant Mean with Time
- Constant Variance with Time
- Constant Autocorellation with Time

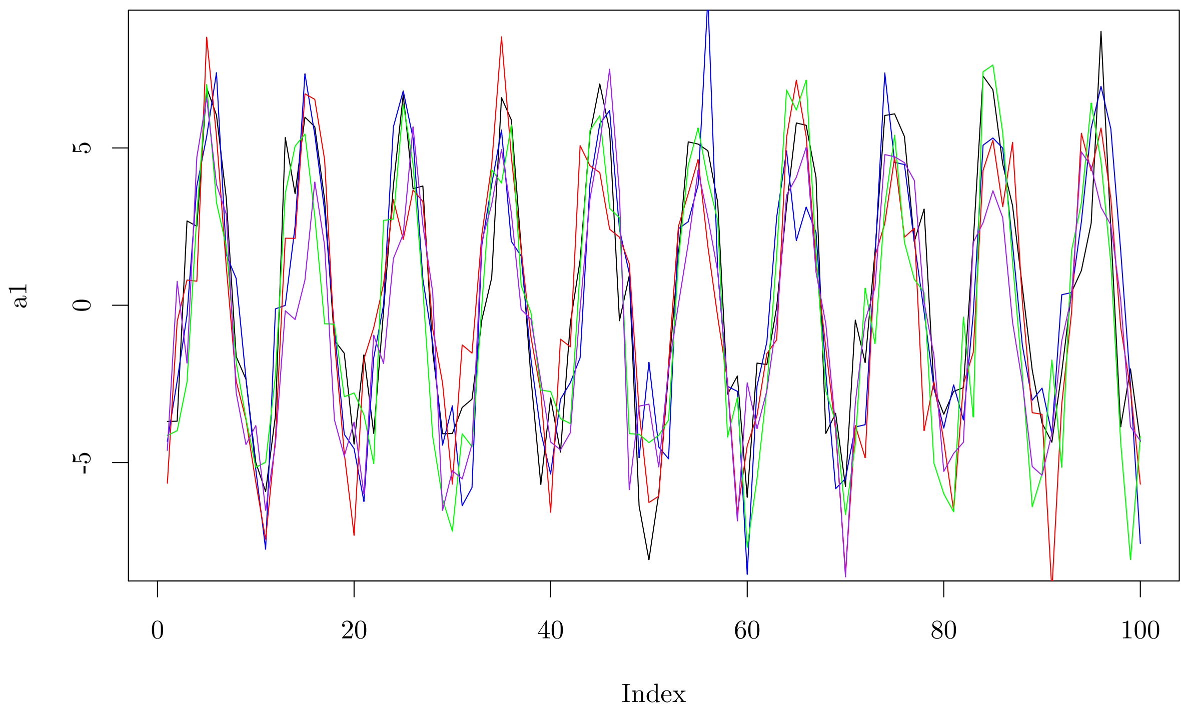
3.1 Condition One: Constant Mean

Mean is constant with time. That is $E[X_t] = \mu$. Note the lack of a little tiny t in mu! It is indepoent of tine! That is stationarity condition number 1!

Important result: **If we assume constant mean, we can use *all* of the data to estimate the mean**

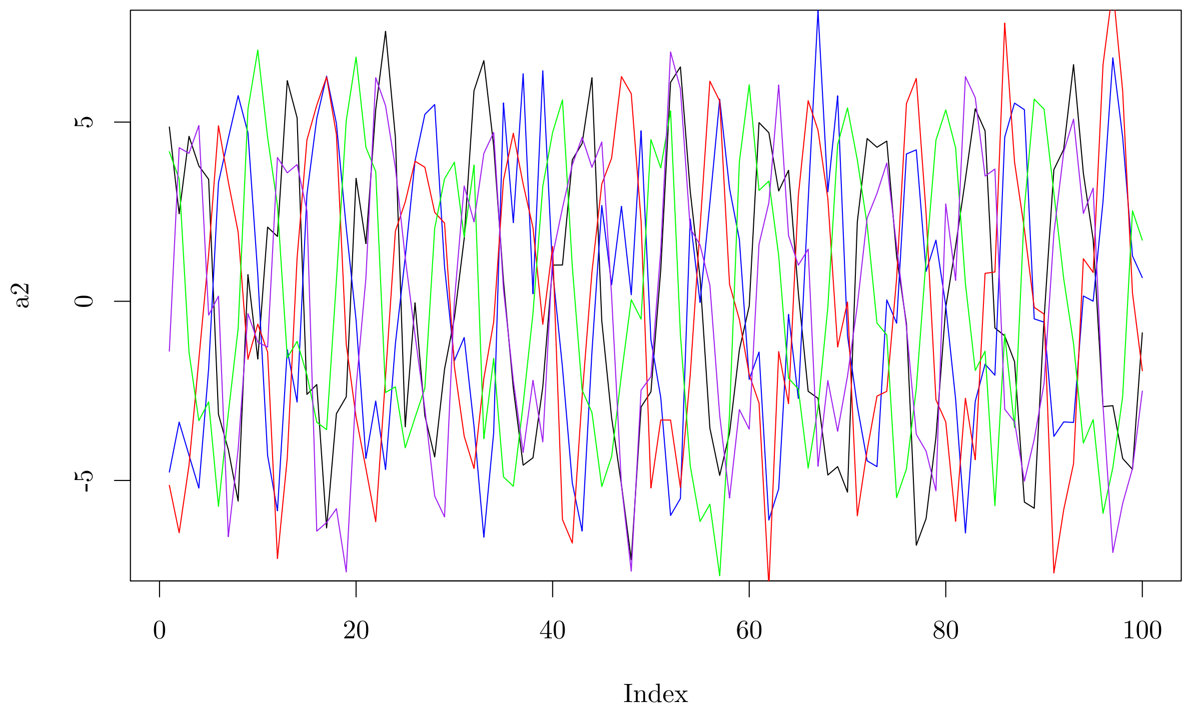
Here we can see after 5 realizations that the mean is clearly not constant with time

```r
library(tswge)
a1 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(3,
    0), vara = 3, plot = F)
b1 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(3,
    0), vara = 3, plot = F)
c1 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(3,
    0), vara = 3, plot = F)
d1 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(3,
    0), vara = 3, plot = F)
e1 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(3,
    0), vara = 3, plot = F)
```



Now lets look at 5 realizations which come from the same realization, which might not have a constant mean:

```
a2 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(runif(1,
    0, 2 * pi), 0), vara = 3, plot = F)
b2 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(runif(1,
    0, 2 * pi), 0), vara = 3, plot = F)
c2 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(runif(1,
    0, 2 * pi), 0), vara = 3, plot = F)
d2 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(runif(1,
    0, 2 * pi), 0), vara = 3, plot = F)
e2 = gen.sigplusnoise.wge(100, coef = c(5, 0), freq = c(0.1, 0), psi = c(runif(1,
    0, 2 * pi), 0), vara = 3, plot = F)
```



We cannot say for sure whether the mean is constant or not in this case!

3.2 Condition Two: Constant Variance

Variance does not depend on time

Variance is *constant* and finite*

That is :

$$Var[X_t] = \sigma^2 \neq \infty$$

If we can make this assumption, then we can use all of the data to make the variance. This is typically the hardest one to tell, and it will take some practice to see.

3.3 Condition 3: Constant autocorrelation

Correlation of $X_{t_1} and X_{t_2}$ only depends on $t_2 - t_1$

That is, the correlation between points depends only on how far apart they are in time, not where they are in time.

To describe this mathematically:

Let

$$h = t_2 - t_1$$

then

$$\text{Cor}\left(X_t, X_{t+h}\right) = \rho_h$$

ρ represents the population correlation coefficient.

Chapter 4

Autocorrelation

```r
library(tswge)
library(ggplot2)
acfdf <- function(vec) {
    vacf <- acf(vec, plot = F)
    with(vacf, data.frame(lag, acf))
}
ggacf <- function(vec) {
    ac <- acfdf(vec)
    ggplot(data = ac, aes(x = lag, y = acf)) + geom_hline(aes(yintercept = 0)) +
        geom_segment(mapping = aes(xend = lag, yend = 0))
}

tplot <- function(vec) {
    df <- data.frame(X = vec, t = seq_along(vec))
    ggplot(data = df, aes(x = t, y = X)) + geom_line()
}
```

4.1 Independence

Two events are independent in a time series if the probability that an event at time t occurs in no way depends on the ocurrence of any event in the past or affects any event in the future. Mathematically, this is written as:

$$\text{Independence} : P\left(x_{t+1} | X_t\right) = P\left(X_{t+1}\right)$$

4.2 Serial Dependence / Autocorrelation

4.2.1 A definition

If two events are independent, their corellation is 0 That is **if X_t and X_{t+k} are independent, $\rho_{x_t,x_{t+k}} = 0$**

*Corollary: If the correlation between two variables is not zero, then they are not **independent***

In other words if $\rho_{x_t,x_{t+k}} \neq 0$, they are ***not*** independent.

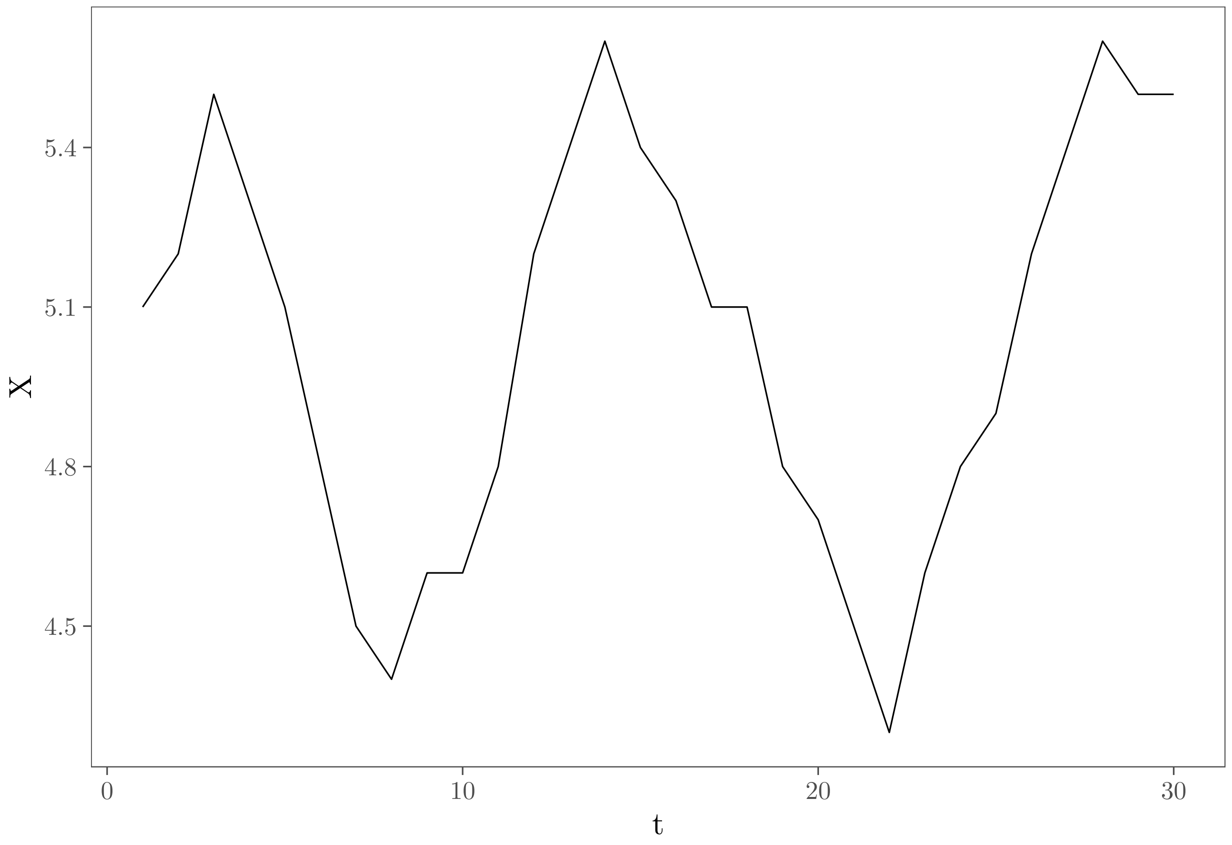
4.2.2 Autocorrelation Plots

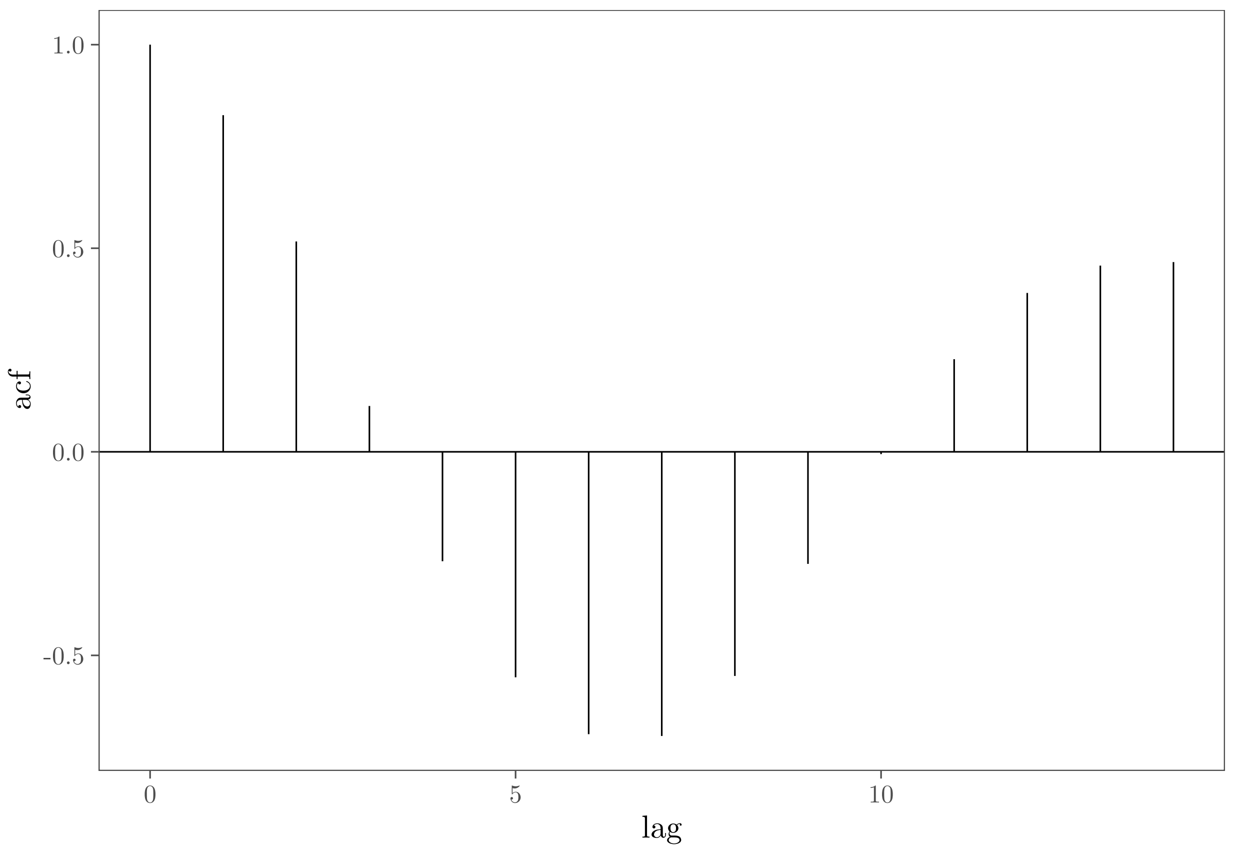
4.2.3 Dependent(ish) data

In time series we look at the **autocorrelation** between X_t and X_{t+1} etc (with t and t+1 it is lag 1 autocorrelation)

For example, visually, let us define a vector, $Y5$ and take its autocorrelation:

```r
Y5 <- c(5.1, 5.2, 5.5, 5.3, 5.1, 4.8, 4.5, 4.4, 4.6, 4.6, 4.8, 5.2, 5.4, 5.6,
    5.4, 5.3, 5.1, 5.1, 4.8, 4.7, 4.5, 4.3, 4.6, 4.8, 4.9, 5.2, 5.4, 5.6, 5.5,
    5.5)
tplot(Y5) + ggthemes::theme_few()
ggacf(Y5) + ggthemes::theme_few()
```





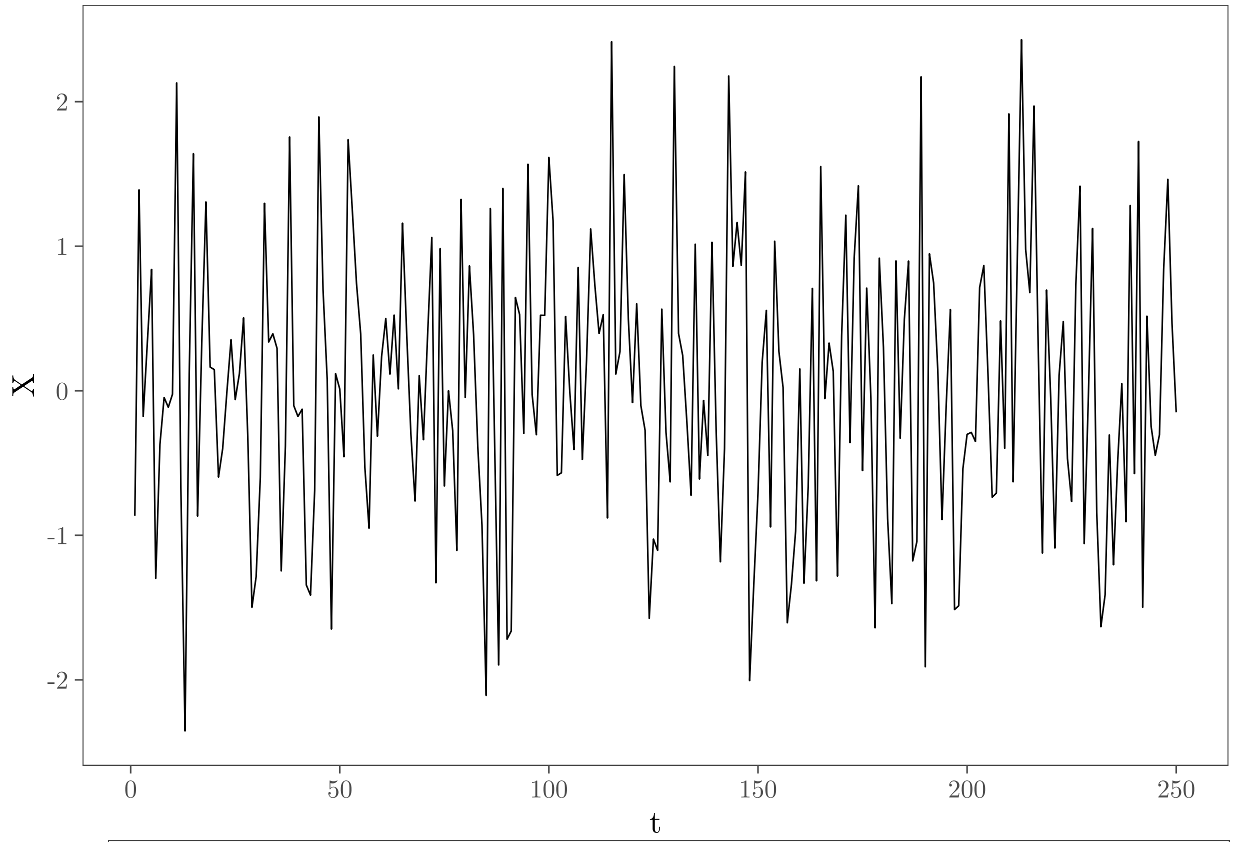
4.2.4 Independent(ish) data!

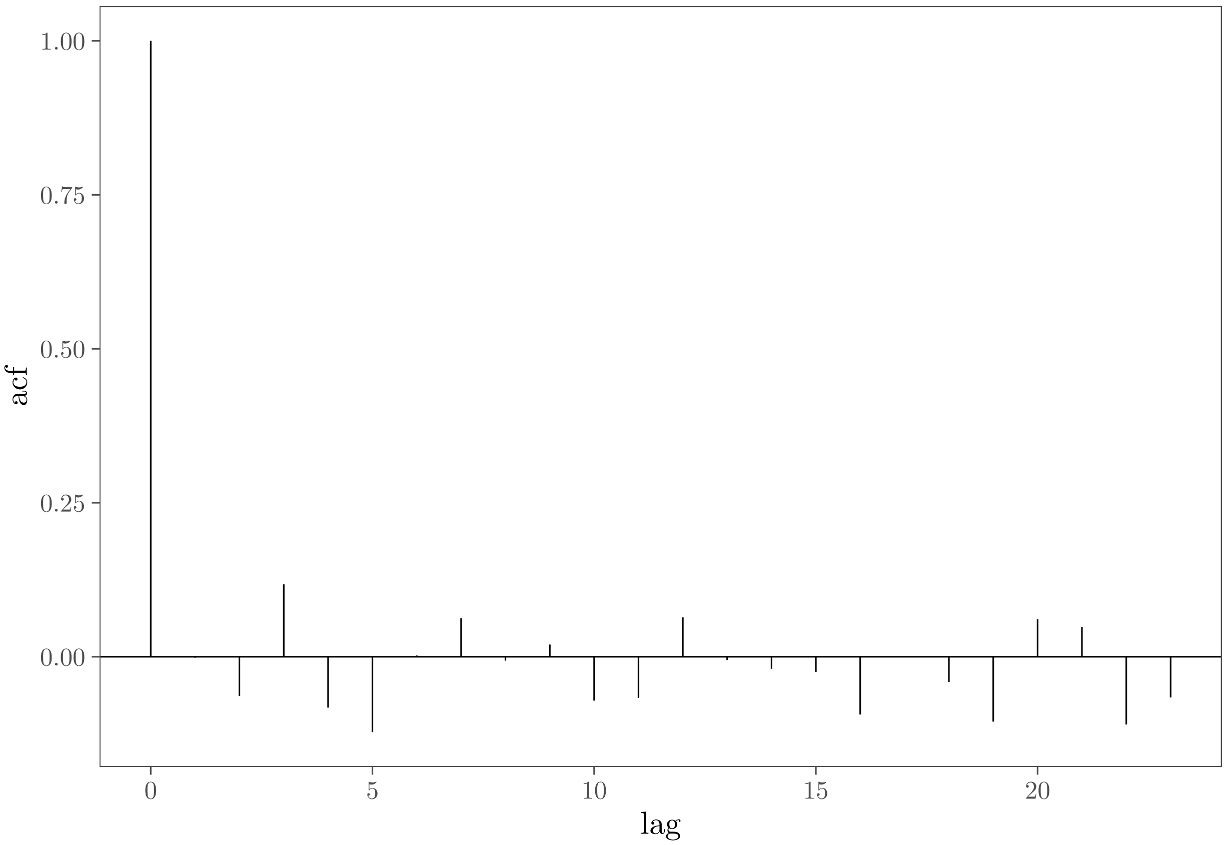
Now let us look at autocorrelation of ***independent*** data

```
xs = gen.arma.wge(n = 250)
```

Let's check it out

```
tplot(xs) + ggthemes::theme_few()
ggacf(xs) + ggthemes::theme_few()
```





We see that the autocorrelation is more or less zero

Chapter 5

Autocorrelation Concepts and Notation

5.1 Theoretical concept of ρ

$$\rho = \frac{E\left[\left(X - \mu_X\right)\left(Y - \mu_Y\right)\right]}{\sigma_X \sigma_Y} = \frac{\text{covariancebetweenX} + \text{y}}{\sigma_X \sigma_Y}$$

In a stationary time series, we have autocorrelation and autocovariance

5.2 autocovariance

$$\gamma_k = E\left[\left(X_t - \mu\right)\left(X_{t+k} - \mu\right)\right]$$

5.3 autocorrelation

$$\rho_k = \frac{E\left[\left(X_t - \mu\right)\left(X_{t+k} - \mu\right)\right]}{\sigma_{X_t}\sigma_{X_{t+k}}} = \frac{E\left[\left(X_t - \mu\right)\left(X_{t+k} - \mu\right)\right]}{\sigma_X^2} = \frac{\gamma_k}{\sigma_X^2}$$

This works because of constant variance and constant mean!

Note that in a stationary time series:

$$\sigma_X^2 = 1E[(X_t - \mu)^2] = E[(X_t - \mu)(X_t - \mu)] = \gamma_0$$

Therefore:

$$\rho_k = \frac{\gamma_k}{\gamma_0}$$

5.4 Stationary Covariance

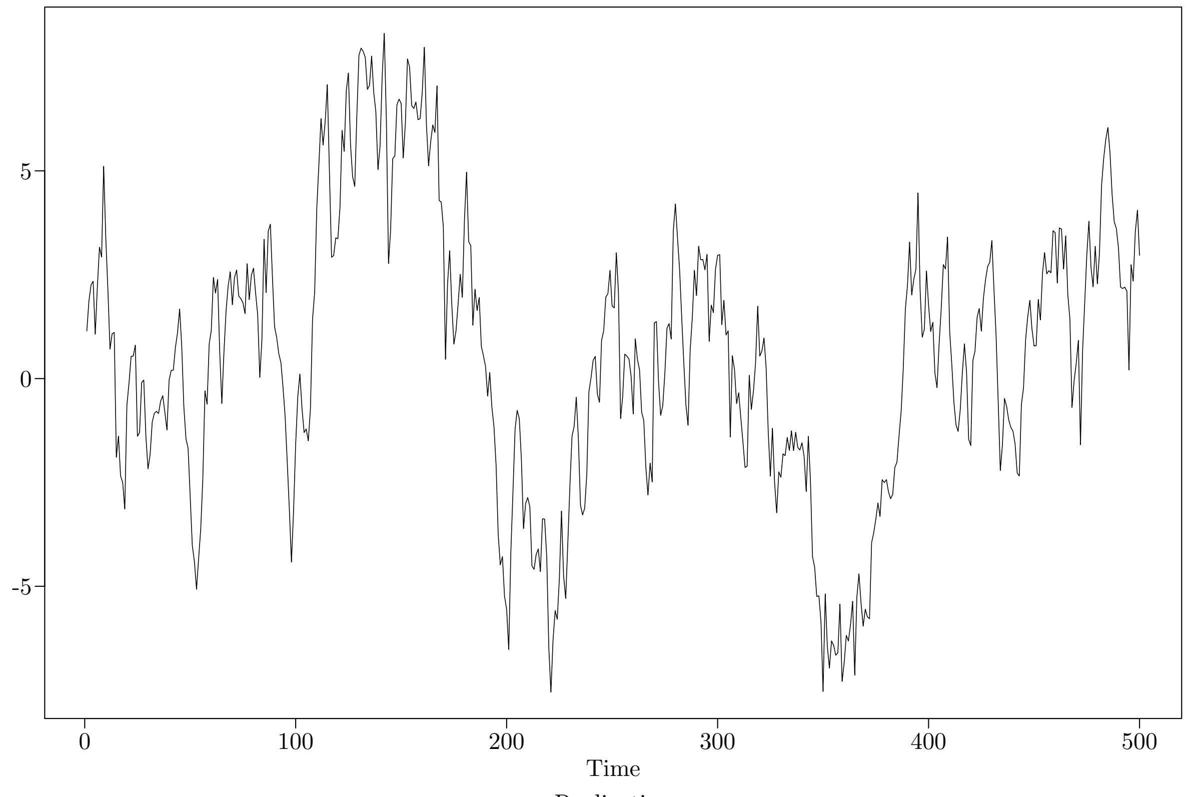
Correlation is not affected by where, only by how far apart, that is:

$$\rho_h = \operatorname{Cor}\left(X_t, X_{t+h}\right)$$

Let us try it out with this little snippet:

```
ggsplitacf <- function(vec) {
    h1 <- vec[1:(length(vec)/2)]
    h2 <- vec[(length(vec)/2):length(vec)]
    first <- ggacf(vec) + ggthemes::theme_few()
    second <- ggacf(h1) + ggthemes::theme_few()
    third <- ggacf(h2) + ggthemes::theme_few()
    cowplot::plot_grid(first, second, third, labels = c("original", "first half",
        "second half"), nrow = 2, align = "v")
}
```

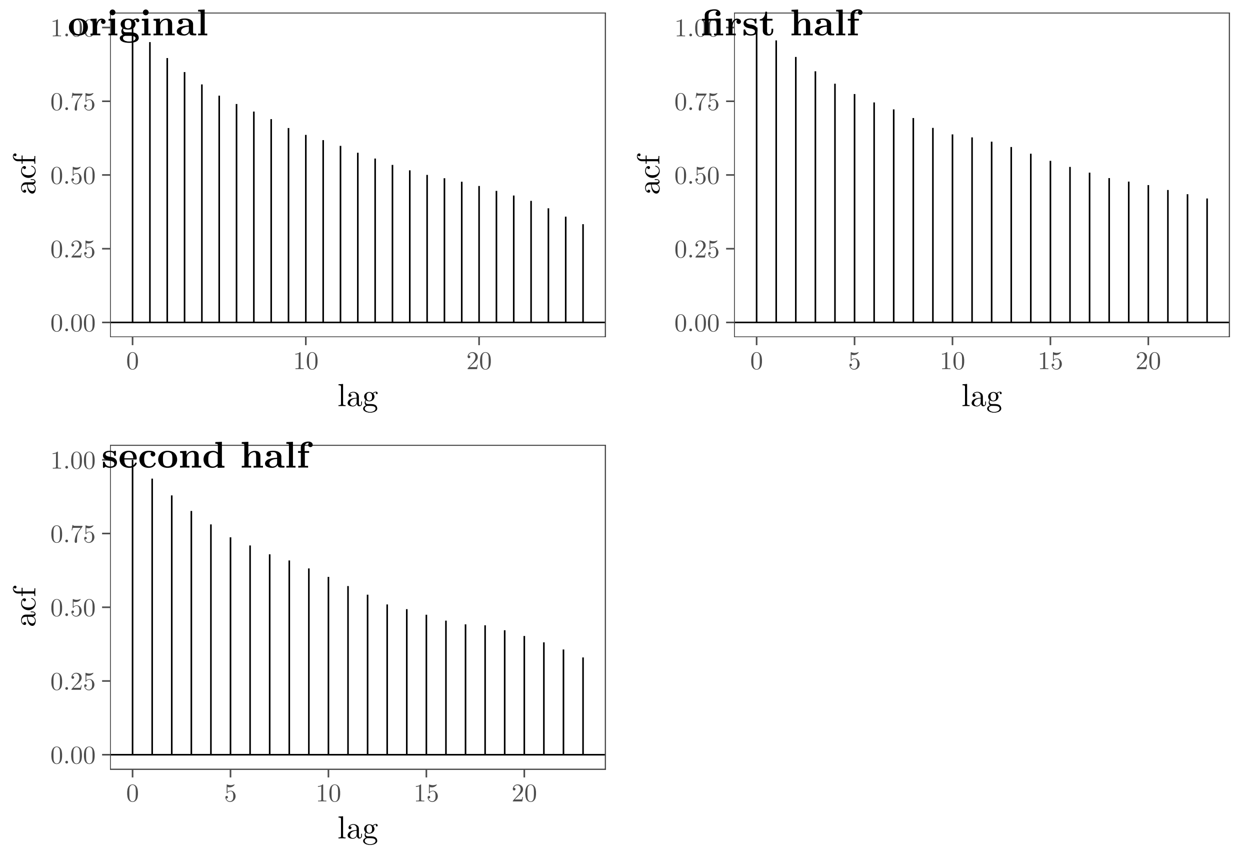
```
Realize = gen.arma.wge(500, 0.95, 0, sn = 784)
```



Realization

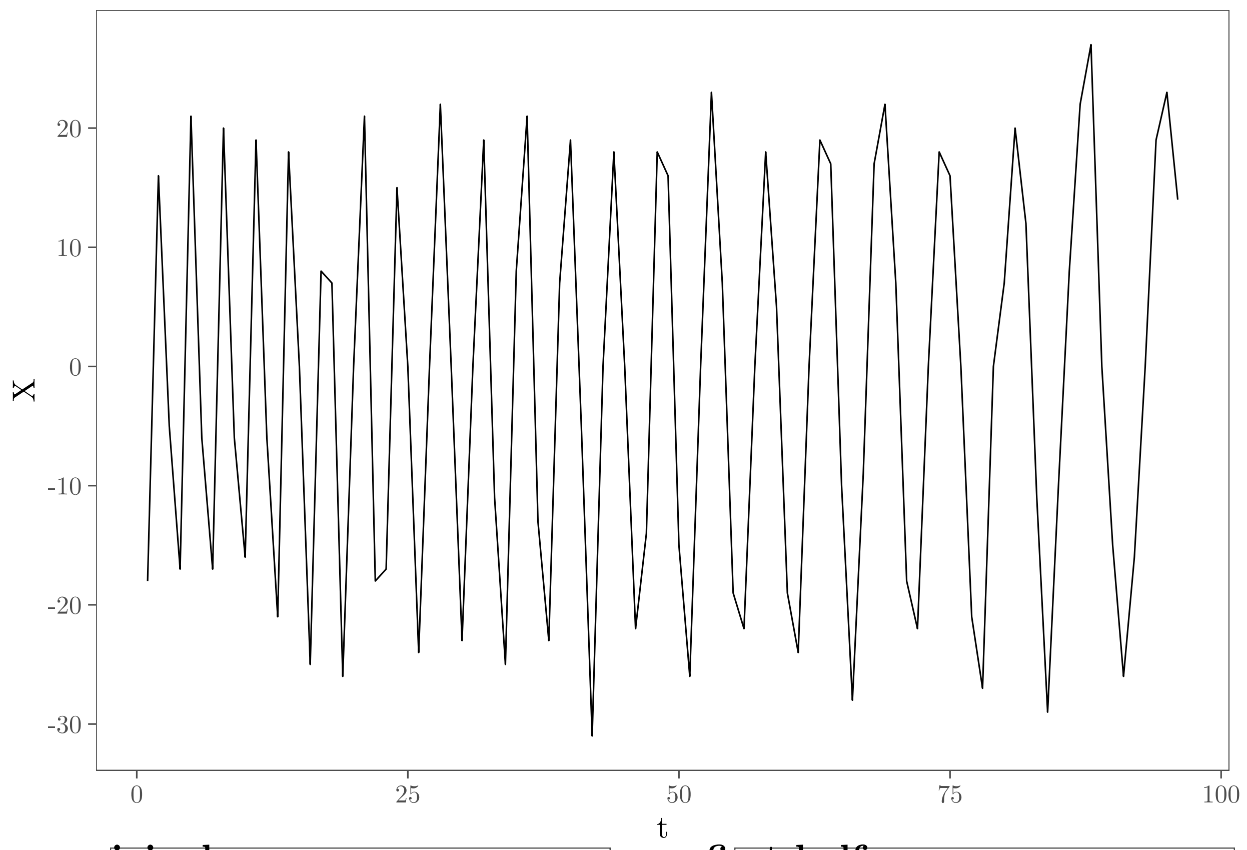
Lets check out the ACF

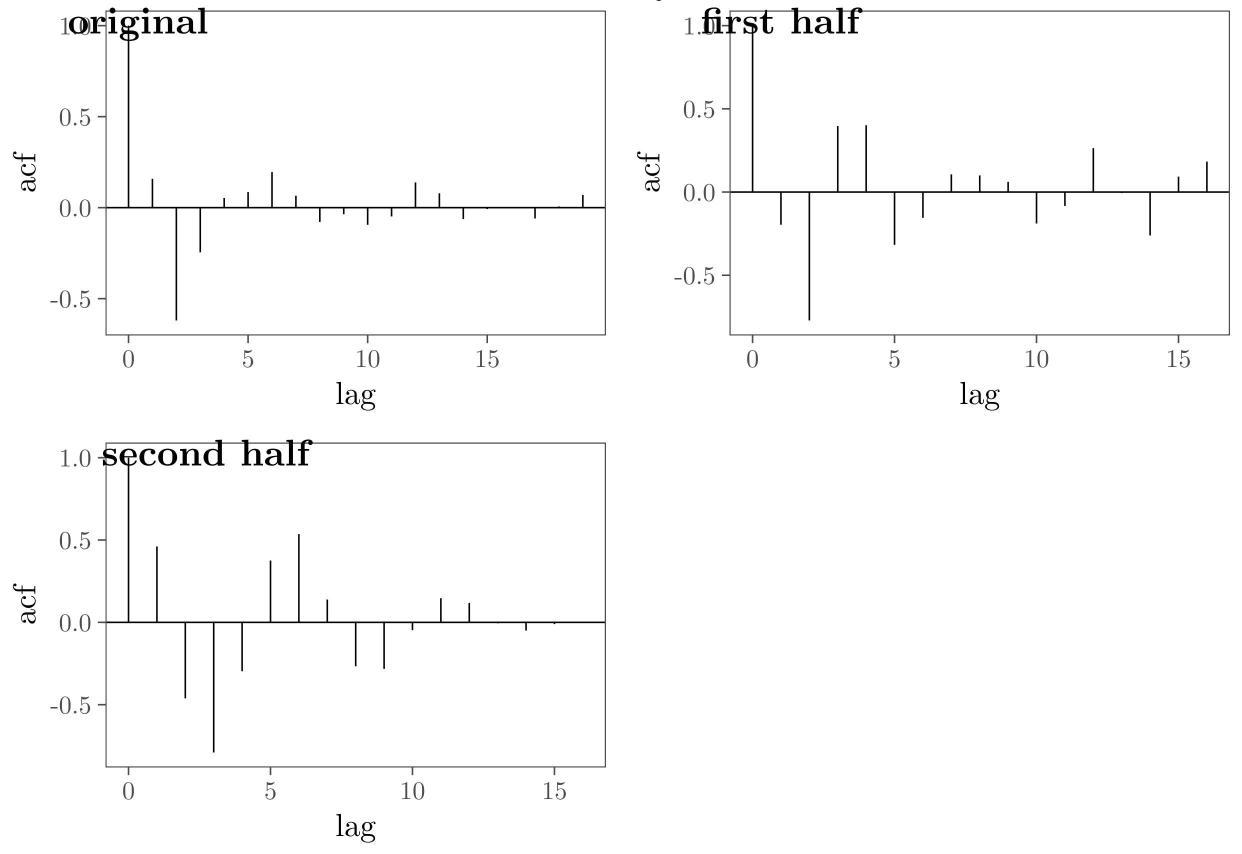
```
ggsplitacf(Realize)
```



This looks pretty stationary to me! Let's try with some sample data:

```r
data("noctula")
tplot(noctula) + ggthemes::theme_few()
ggsplitacf(noctula)
```

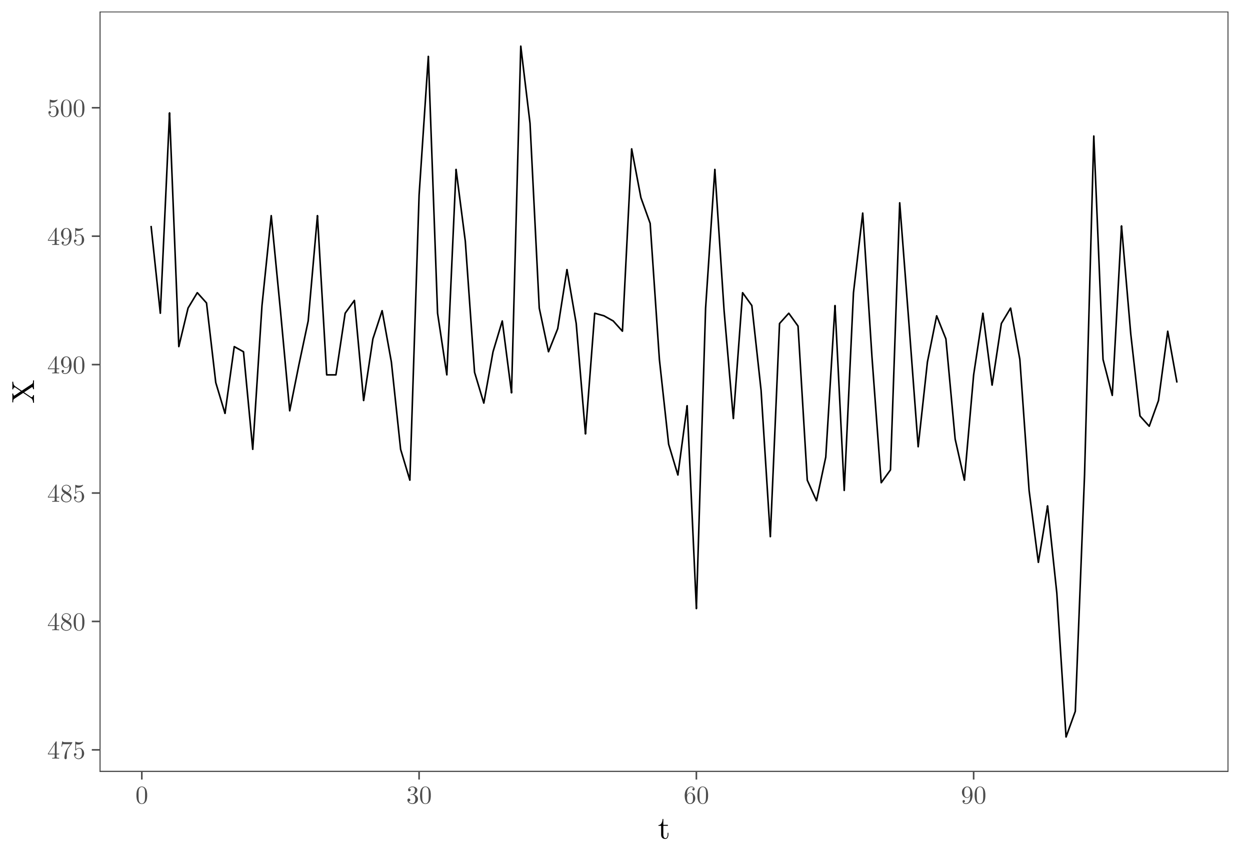


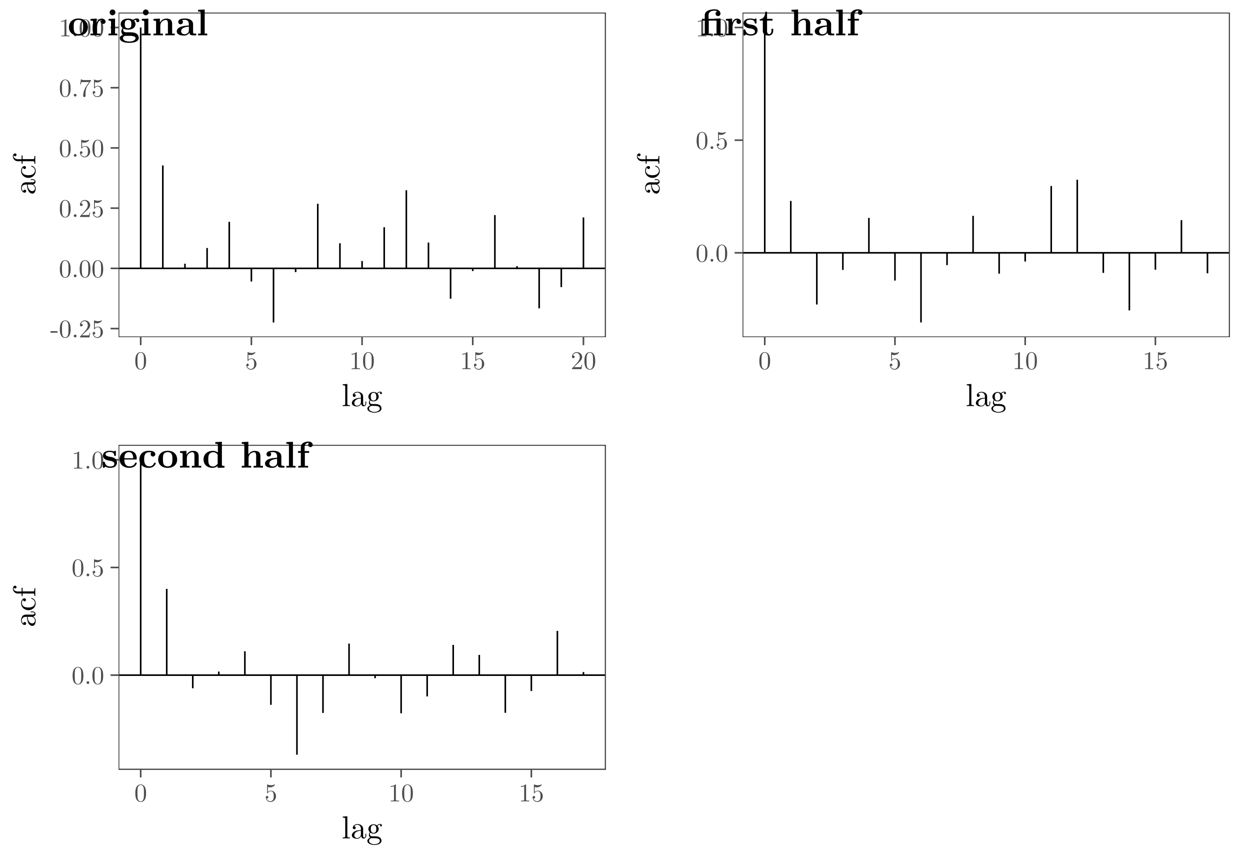


So in this case our mean probably looks constant, our variance could be constant as well, however the ACFs are clearly different.

Let's look at a different dataset:

```
data(lavon)
tplot(lavon) + ggthemes::theme_few()
ggsplitacf(lavon)
```





In this case, our mean looks maybe to be constant around 495, our variance does not appear to be constant, and our ACF looks pretty good, however I would say overall this probably is not stationary, due to the variance and slightly off ACF.

Chapter 6

Practical Autocorrelation

6.1 Estimation!

6.1.1 $\rho_k \to 0$

For a stationary time series, if the autocorrelation approaches zero, then :

A Single realization lets us estimate mean, variance, and autovvariance!

Remember that $\rho_k = \frac{\gamma_k}{\gamma_0}$, and we use n-k pairs to calculate it (the summation)

6.1.1.1 Mean

Just calculate the mean normally for this case.

6.1.1.2 Variance

$$\mathrm{Var}\left(\bar{X}\right) = \frac{\sigma^2}{n} \sum_{k=-(n-1)}^{n-1} \left(1 - \frac{\mid k \mid}{n}\right) \rho_k$$

σ^2 is calculated as normal, we will see rhok next!

remember !

Now it is time for some code!

```r
library(glue)
```

```
##
## Attaching package: 'glue'

## The following object is masked from 'package:dplyr':
##
##     collapse
```

```r
xbar <- function(xs) {
    mean(xs)
}

ghat_zero <- function(xs) {
    summand <- (xs - xbar(xs))^2
    mean(summand)
}

ghat_one <- function(xs) {
    lhs <- xs[1:(length(xs) - 1)] - xbar(xs)
    rhs <- xs[2:length(xs)] - xbar(xs)
    summand <- lhs * rhs
    summate <- sum(summand)
    summate/length(xs)

}
rhohat_zero <- 1
rhohat_one <- function(xs) {
    ghat_one(xs)/ghat_zero(xs)
}
v <- c(76, 70, 66, 60, 70, 72, 76, 80)
xbar(v)
```

```
## [1] 71.25
```

```
ghat_zero(v)
```

```
## [1] 34.9375
```

```
ghat_one(v)
```

```
## [1] 14.74219
```

```
rhohat_one(v)
```

```
## [1] 0.4219589
```

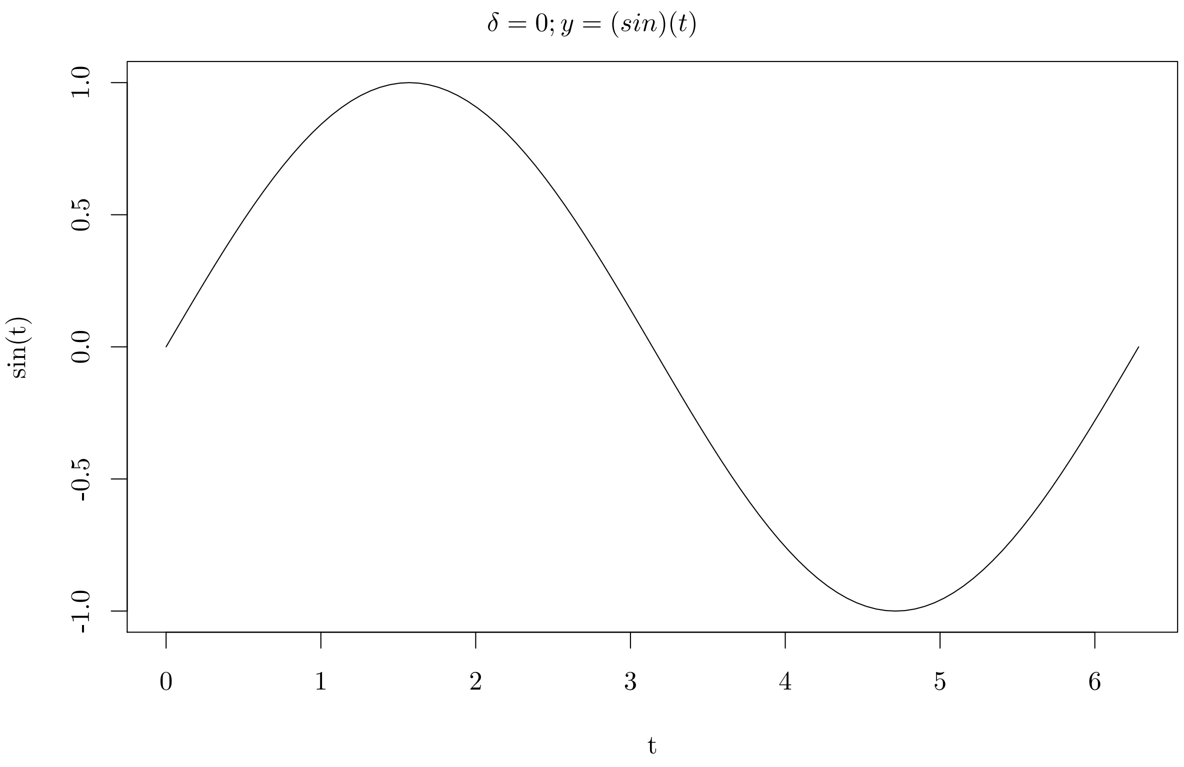
Chapter 7

The Frequency Domain

7.1 Review of sine and cosine

7.1.1 Phase shifts

This is what a sine wave looks like



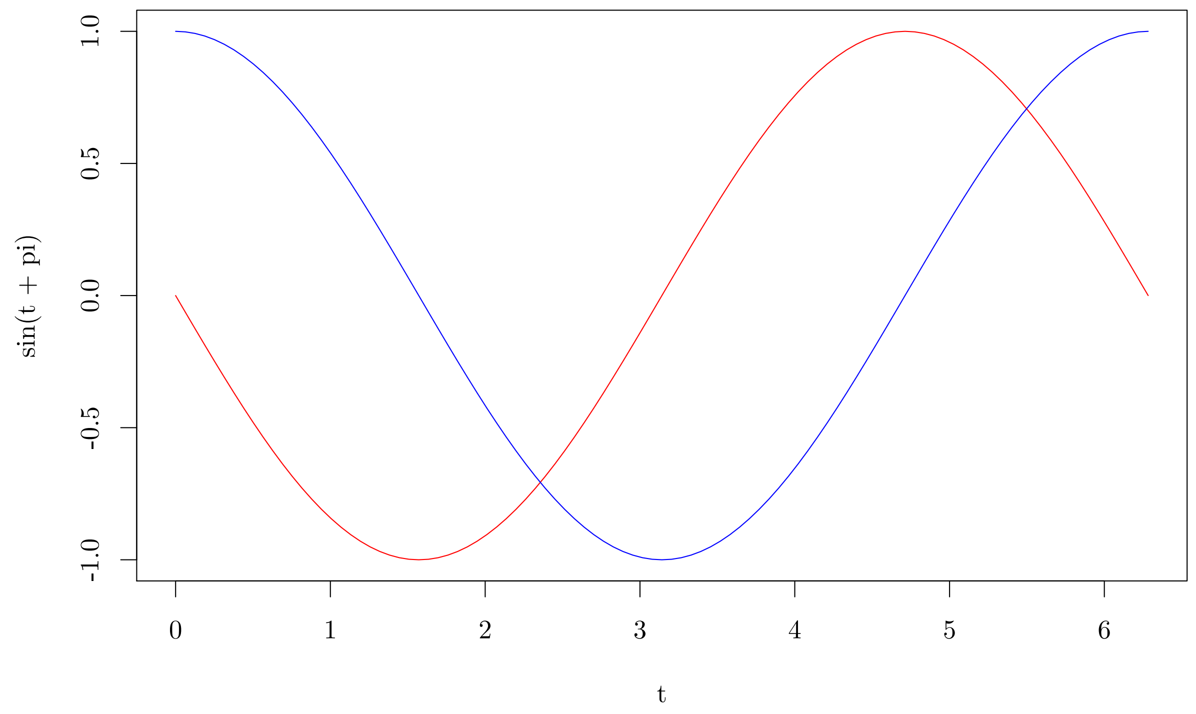
We can phase shift the sin wave by adding

RED:

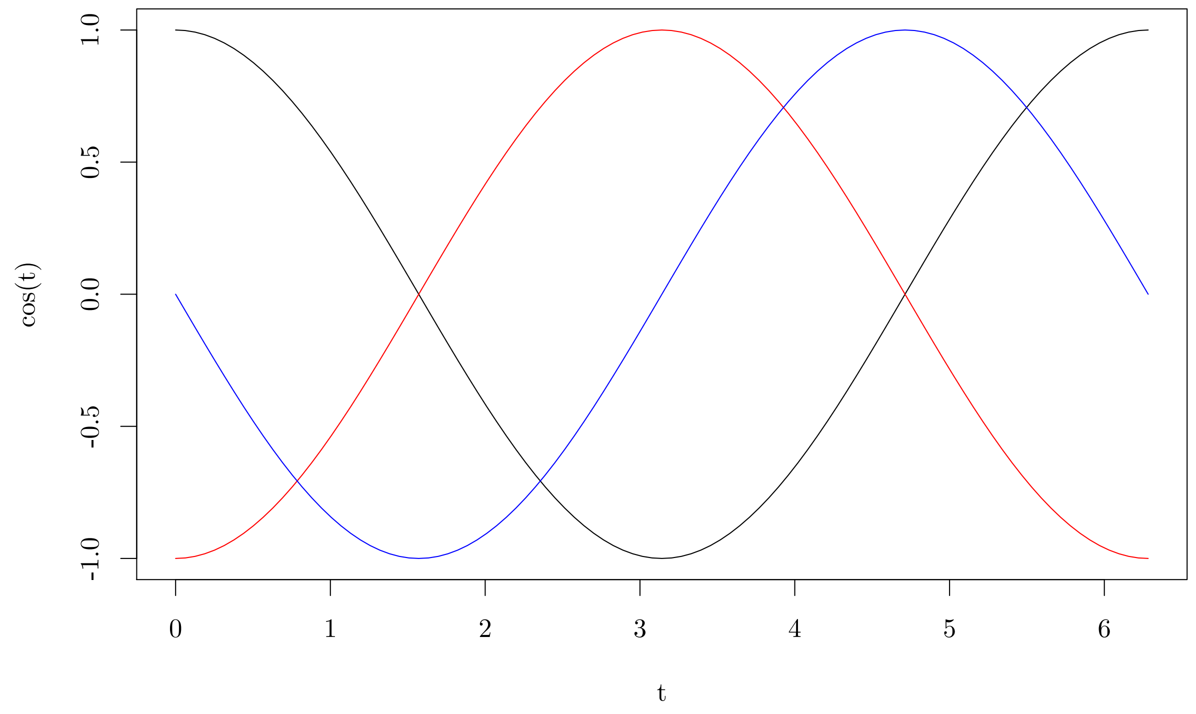
$$\delta = \pi; y = \sin(t + \pi)$$

BLUE:

$$\delta = \frac{\pi}{2}; y = \sin(t + \frac{\pi}{2})$$



Now cosine



7.2 Frequency and Period

Let us say $f(x)$ is a periodic function. If $f(x)$ has period p, then p is the smallest value S.T.

$$f(x) = f(x + kp)$$

Cool

7.3 psueudo-periodicity

Data are *psuedo-periodic* with period p if p is the smallest value such that a cycle appears to repeat itself.

7.4 Aperiodicity

Aperiodic data is data in which no p exists.

7.5 Frequency

Frequency is calculated as : $\frac{1}{p}$

It symbolizes the number of cycles per unit(number of periods per unit)

7.6 Frequency Terminology

Frequency of a trigonometruc function:

Let $f(x) = \sin(2\pi f t)$

Then we have that :

$$p = 1/f$$

$$\nu = f$$

Consider:

$$\sin(Bt + C)$$

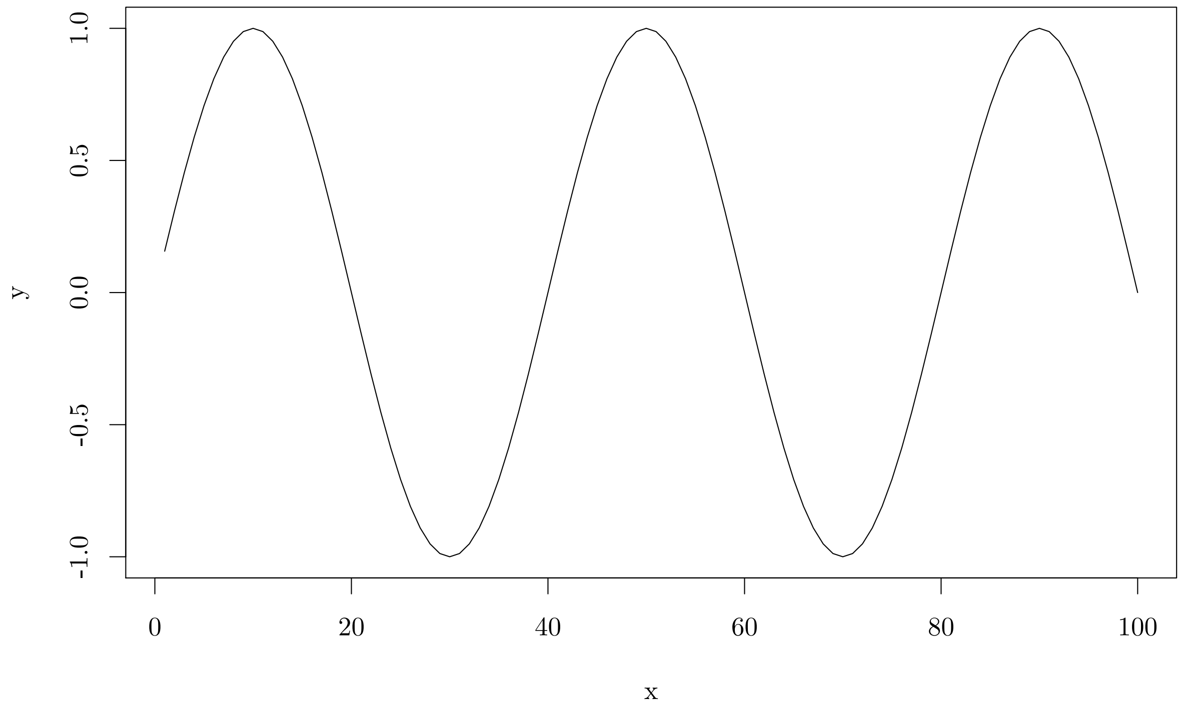
Then

$$p = \frac{2\pi}{B}$$

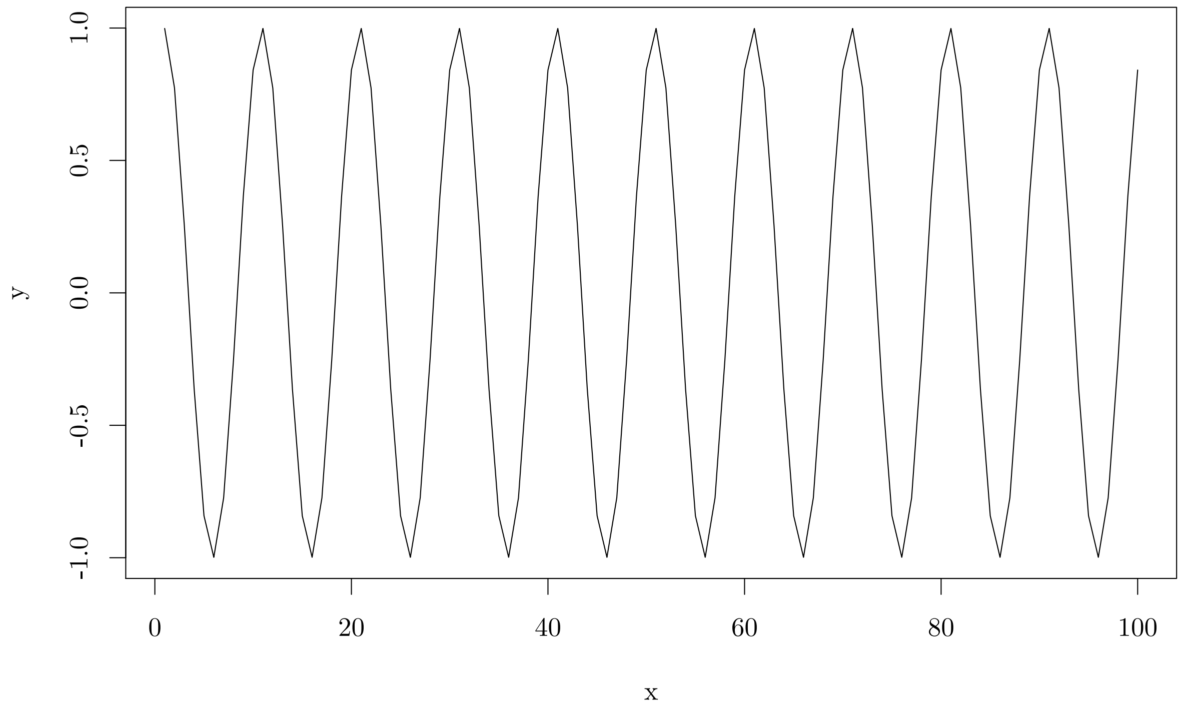
$$\nu = \frac{B}{2\pi}$$

7.7 Composite sine function

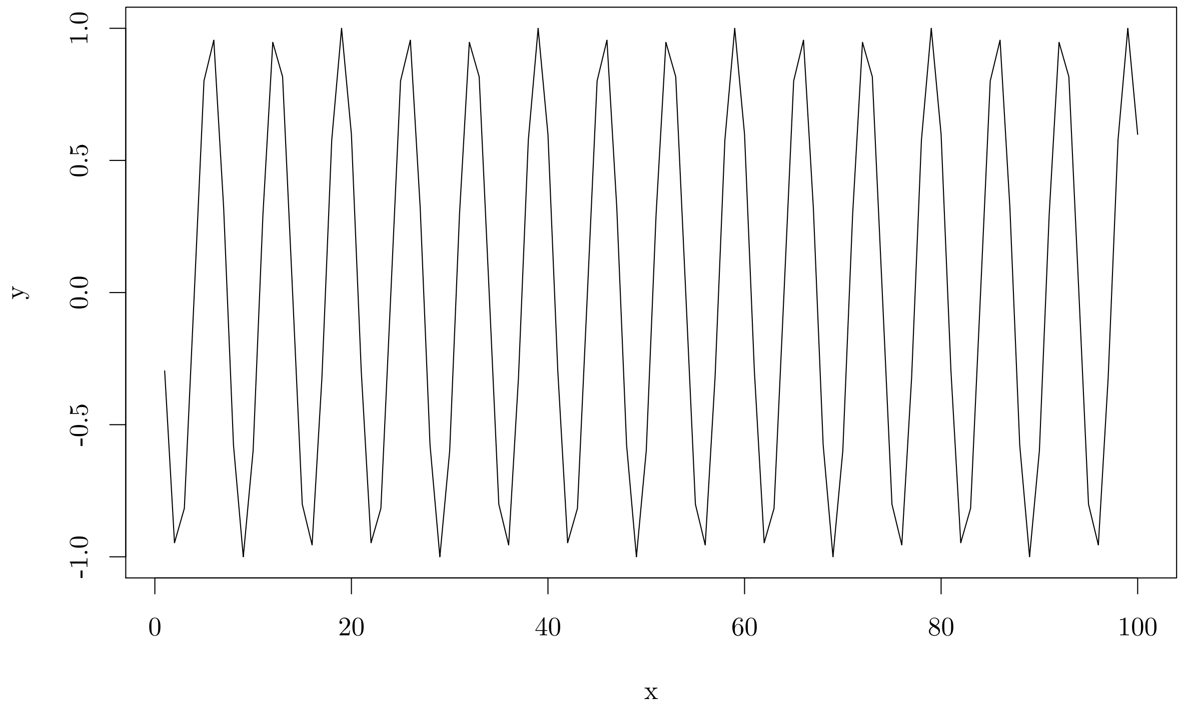
```r
t <- seq(1, 100, length = 100)
y1 <- sin(2 * pi * 0.025 * t)
y2 <- sin(2 * pi * 0.1 * t + 1)
y3 <- sin(2 * pi * 0.15 * t + 2.5)
lineplotter <- function(x, y) {
    plot(x, y, type = "l")
}

lineplotter(t, y1)
```



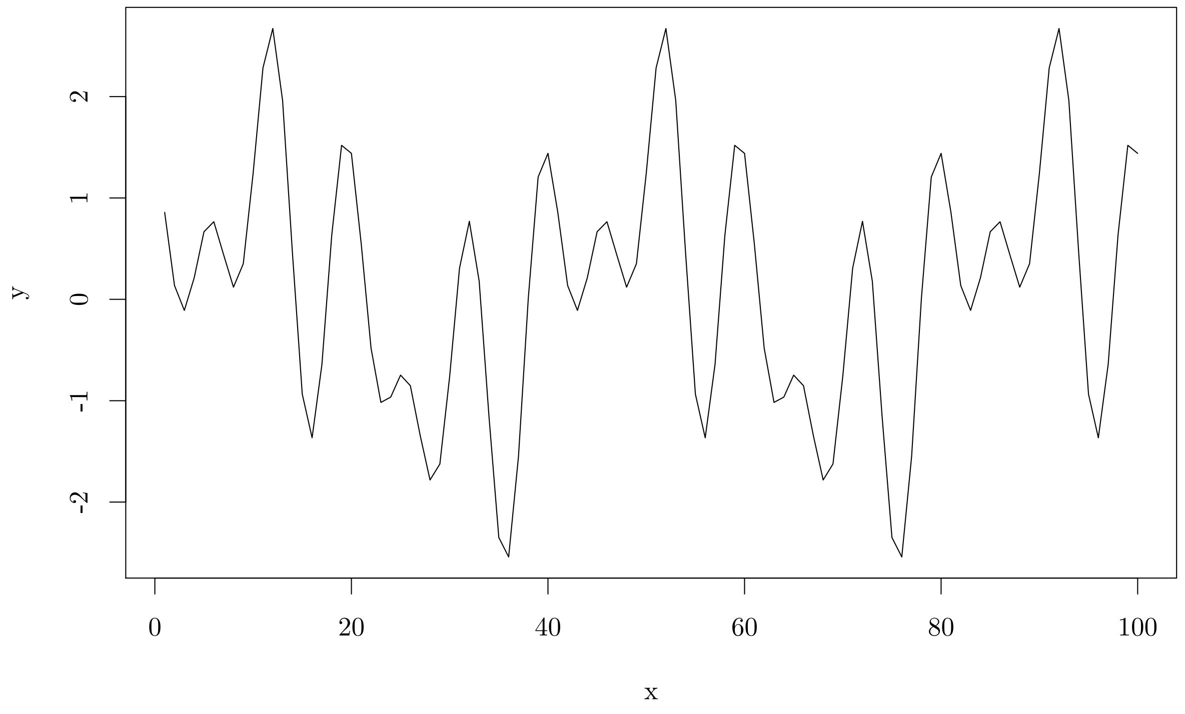
```r
lineplotter(t, y2)
```



```r
lineplotter(t, y3)
```



```r
lineplotter(t, y1 + y2 + y3)
```



this is a signal with three frequencies!

7.8 Fundamental idea

A lot of realizations are made up of components of various frequencies, and some do not have frequency at all. Uncovering the frequency content of a dataset (or lack thereof) allows us to better understand the process generating the data.

7.8.1 A Brief Look at Fourier Series

When we do spectral analysis, we are generally going to be decomposing functions into sine and cosine terms, in order to better analyze the frequency of the data.

7.8.1.1 Fourier Series

You should just know this by this point, if you dont, google it. Too lazy to tex up the equations. Basically, we can decompose any function into a bunch of sines and cosines. It is pretty rad. The important conclusion of the existence of fourier series is that we can represent non cyclic functions as a bunch of sines and cosines, which will allow us to do spectral analysis of them.

Chapter 8

Spectral Density

A tool to identify the frequency content of a (stationary) time series.

8.1 Derivation of spectral density of white noise

$$S_x(f) = 1 + 2\sum_{k=1}^{\infty} \rho_k \cos\left(2\pi f k\right), \mid f \mid = 0.5$$

For white noise, $\rho_0 = 1$, if $k \neq 0$, $\rho_k = 0$

This tells us that for white noise:

$$S_x(f) = 1 + 2 * 0 = 1$$

In other words: **All frequencies are equally present in white noise**

8.2 Estimating Spectral Density from Real Data

With real data, we don't have an infinite amount, and we do not know what the true autocorrelation is. So we are probably going to have to make some estimates.

Lets first estimate autocorrelation:

$$S_x(f) = 1 + 2\sum_{k=1}^{\infty} \widehat{\rho_k} \cos\left(2\pi f k\right), \mid f \mid = 0.5$$

Now lets deal with the infinity, which is killing us. We know that:

$$\hat{\rho}_1 = \frac{\frac{1}{n}\sum_{t=1}^{n} -1(X_t - \bar{X})(X_{t+1} - \bar{X})}{\hat{\gamma}_0}$$

And (check yourself to see the math works)

$$\hat{\rho}_{n-1} = (X_1 - \bar{X})(X_n - \bar{(}x))/(n\hat{\gamma}_0)$$

We can then rewrite the sample spectral density/sample spectrum as!:

$$S_x(f) = 1 + 2\sum_{k=1}^{n-1} \widehat{\rho_k} \cos\left(2\pi f k\right), \mid f \mid = 0.5$$

However, this estimate of autocorrelation cannot be very good, since it is estimated using one cross product/pair. Thats like estimating the mean with one value!

So, lets stop this sum somewhere more reasonable.

$$\widehat{S_x(f)} = 1 + 2\sum_{k=1}^{M} \widehat{\rho_k}\cos\left(2\pi f k\right), \mid f \mid = 0.5$$

Where M is less than $n-1$. This lets us have a bit more freedom to decide what M should be.

Let us now rewrite the estimate of spectral density as:

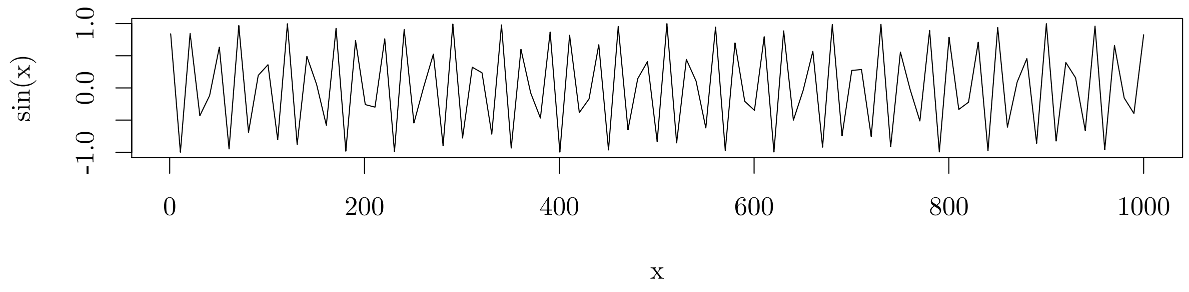
$$\widehat{S_x(f)} = \lambda_0 1 + 2\sum_{k=1}^{M} \lambda_k\widehat{\rho_k}\cos\left(2\pi f k\right), \mid f \mid = 0.5$$

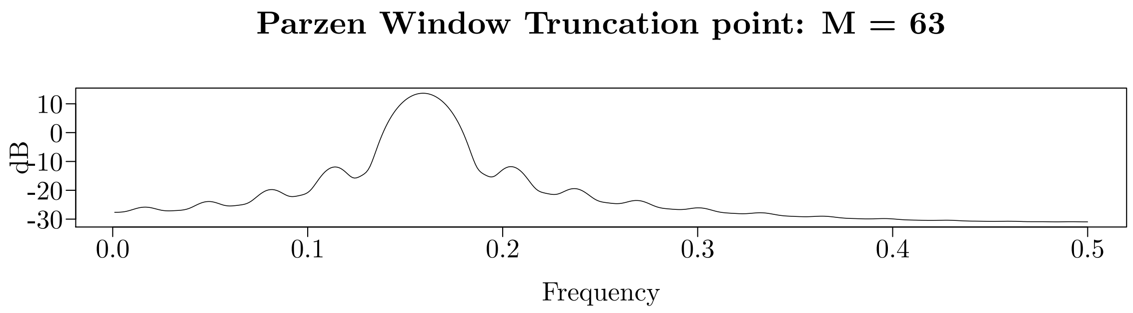
Where $\mid \lambda_k \mid$ decreases as k increases. With this, we have M minimizing and limiting the impact of sample autocorrelations when k gets pretty close to n, and lambda further decreases the impact of the poor sample autocorrelations at high values of k. These are called ***window functions***. In our case, we will use what is called the Parzen window. Other windows (every statistician in the 60's had their own window) include: the Tukey window, the Bartlett window, etc. We call this the the smooth spectral density estimator. M is frequently chosen to be $2\sqrt{n}$

8.3 Spectral Density in the Wild

First, lets look at a simple example:

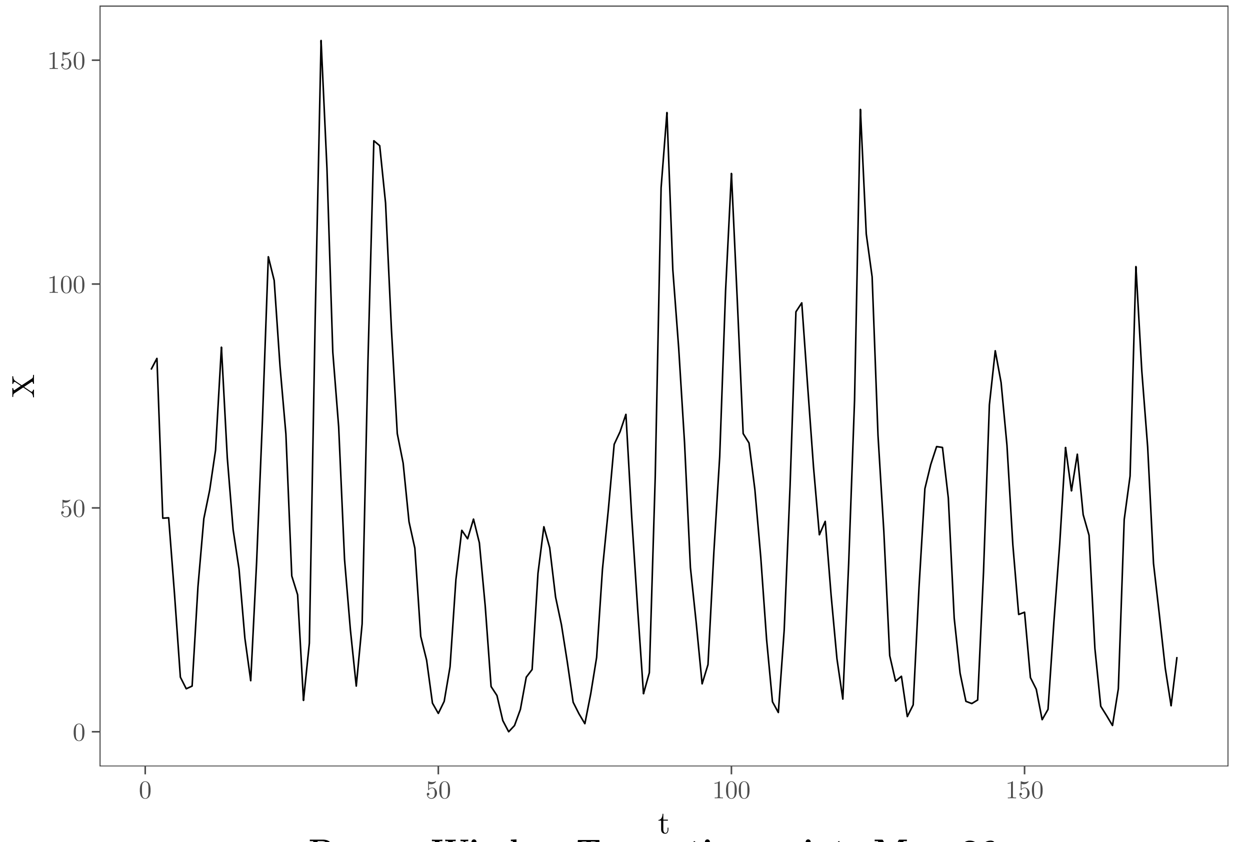
```r
x <- 1:1000
y <- sin(x)
par(mfrow = c(2, 1))
curve(sin(x), 1, 1000)
parzen.wge(y)
```

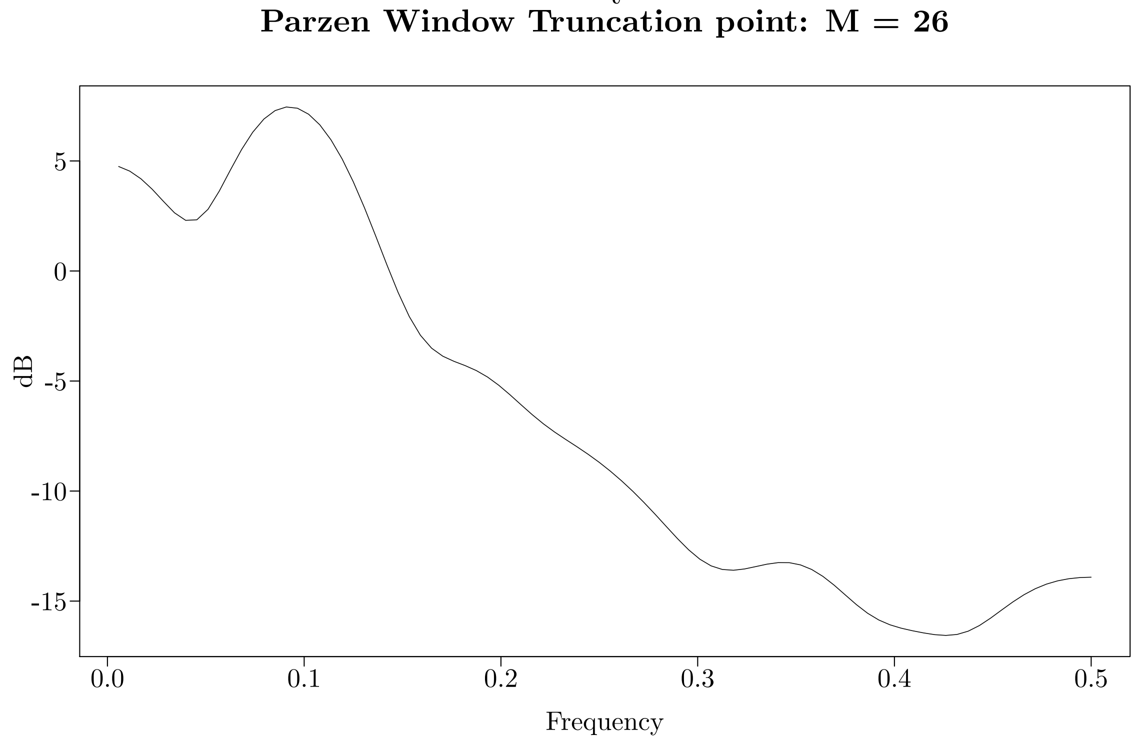




Shockingly, we have a frequency peak at roughly $\frac{1}{2\pi} = 0.1591549$. If I did a better job of generating the data, it would be a super shark peak and nothing else, however I did not. Lets look at sunspot data now:

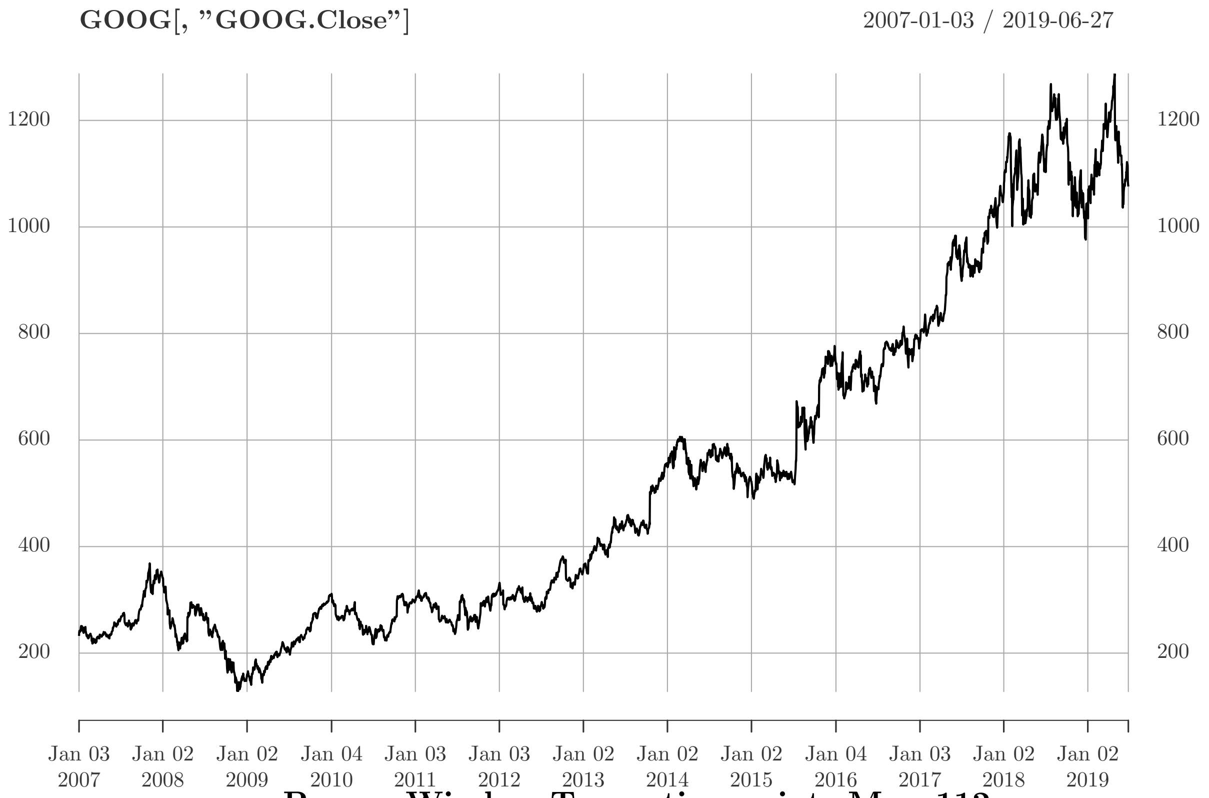
```r
par(mfrow = c(1, 1))
data(sunspot.classic)
tplot(sunspot.classic) + ggthemes::theme_few()
parzen.wge(sunspot.classic)
```

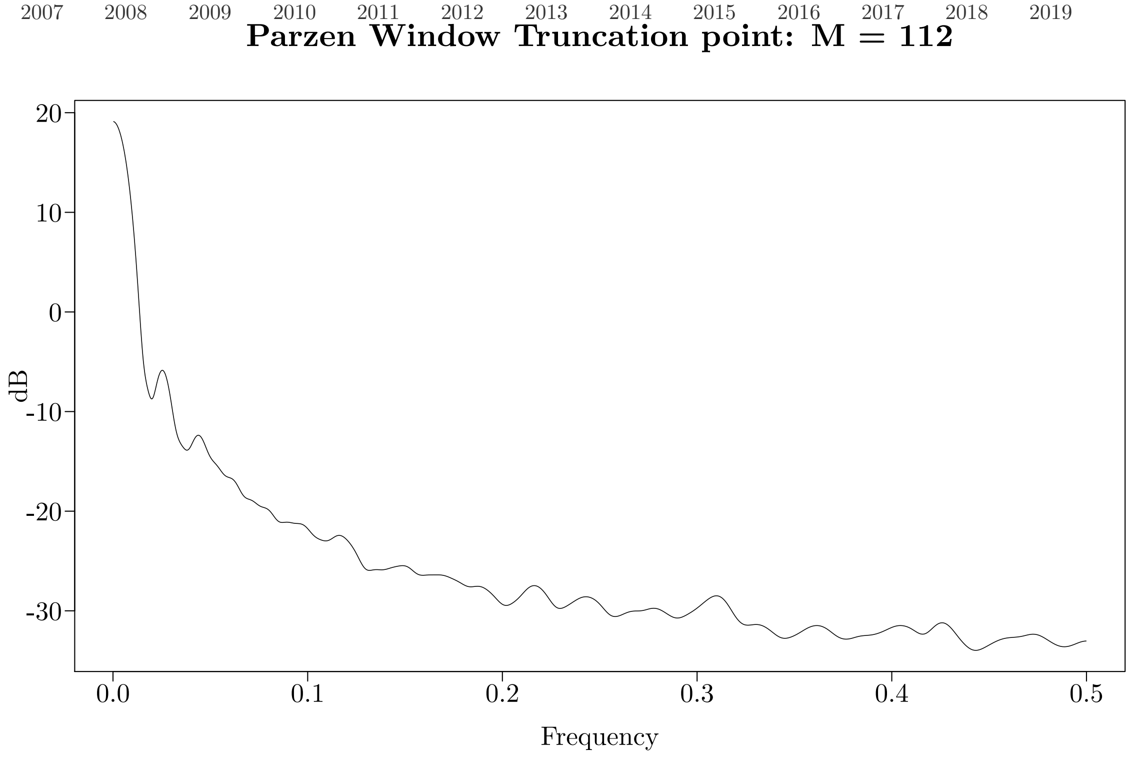




It looks like we have a frequency peak at about 0.1, which makes sense because the classic sunspot psuedoperiod is about 11 years. Lets look at some aperiodic stock data now:

```r
getSymbols("GOOG", src = "yahoo")
plot(GOOG[, "GOOG.Close"])
parzen.wge(GOOG$GOOG.Close)
```





Sadly, there appears to be no real periodicity in google's stock price, no free money for us :(. The peak at 0 means that the data is either aperiodic or it has not been long enough to show a period.

8.4 Why do we do spectral density from -0.5 to 0.5?

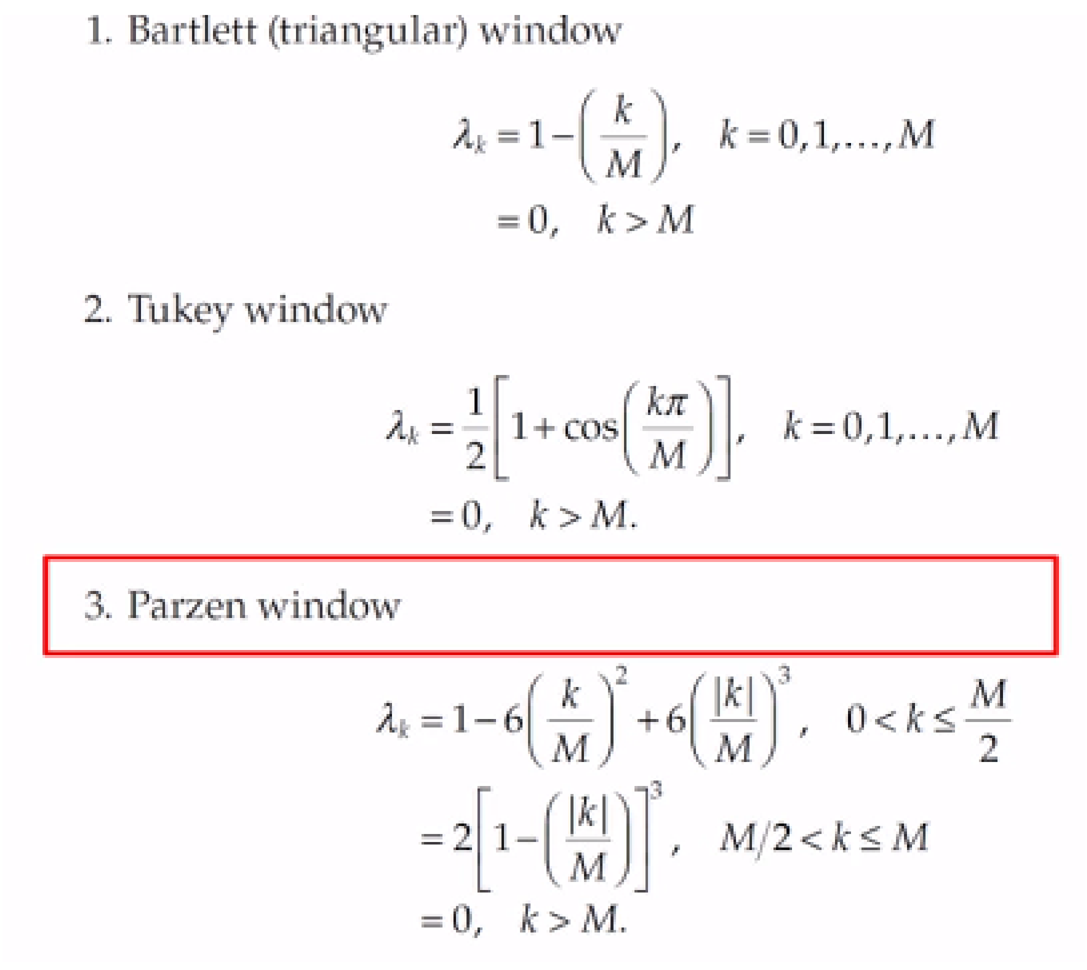
Time series data is taken in integers. What is the shortest distance between two integeres that we can complete a period in? It is 2. This means the highest obesrvable frequency is 0.5. Interestingly, if you look at sin(2pi.3t) and sin(2pi1.3t) at the integers, we will have exactly the same value. This means that even if the frequency is higher, we will have it be aliased to be exactly the same as the lower frequency function:

```r
x <- 0:100
y1 <- sin(2 * pi * 0.3 * x)
y2 <- sin(2 * pi * 1.3 * x)
round(y2 - y1, 2)
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

This is called ***aliasing***. These two functions will have the same spectral densities. 0.5 is known as the ***Nyquist Frequency***

```r
knitr::include_graphics("fig/windows.png")
```



1. Bartlett (triangular) window

$$\lambda_k = 1 - \left(\frac{k}{M}\right), \quad k = 0, 1, \ldots, M$$
$$= 0, \quad k > M$$

2. Tukey window

$$\lambda_k = \frac{1}{2}\left[1 + \cos\left(\frac{k\pi}{M}\right)\right], \quad k = 0, 1, \ldots, M$$
$$= 0, \quad k > M.$$

3. Parzen window

$$\lambda_k = 1 - 6\left(\frac{k}{M}\right)^2 + 6\left(\frac{|k|}{M}\right)^3, \quad 0 < k \le \frac{M}{2}$$
$$= 2\left[1 - \left(\frac{|k|}{M}\right)\right]^3, \quad M/2 < k \le M$$
$$= 0, \quad k > M.$$

A note on plots of spectral density: Normally we do this in a logarithmic scale, so we can see whats going on.

Chapter 9

AR(1) Models and Filtering

9.1 Algebra review

Be able to find the roots of a 2nd order polynomial

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$z = a + bi$$

$$i = \sqrt{-1}$$

a is real b is imaginary

$$z^* = a - bi$$

z* is complex conjugate

It is basically a vector so absolute value is just the magnitude

Unit circle is the values in the complex plane in which the magnitude of z is equal to one. If Z is a complex number with magnitude greater than one, it is outside the unit circle.

```r
quad_form <- function(a, b, c) {
    rad <- b^2 - 4 * a * c
    if (is.complex(rad) || all(rad >= 0)) {
        rad <- sqrt(rad)
    } else {
        rad <- sqrt(as.complex(rad))
    }
    round(cbind(-b - rad, -b + rad)/(2 * a), 3)
}

quad_form(c(1, 4), c(1, -5), c(6, 2))
```

```
##                [,1]          [,2]
## [1,] -0.500-2.398i -0.500+2.398i
## [2,]  0.625-0.331i  0.625+0.331i
```

9.2 Linear filters

A filter will turn Z_t into X_t

for example:

$$\text{difference} : X_t = Z_t - Z_{t-1}$$

Moving average also a filter (smoother)

9.2.1 Example

Consider $ Z_1 = 8 , Z_2 =14, Z_3 = 14, Z_4 = 7$

If we apply the difference filter to this, we have that:

$$X_2 = 6, X_3 = 0, X_4 = 7$$

Note: the differenced data are a realization of length n-1

We can use the difference filter to remove the wandering behjaviior

9.2.2 5 point moving average

We can only get a realization of n-4. (n-nopoints-1). What does the 5 point moving average do (it averages a point and two points ahead and two behind). This filter can filter out some frequencies

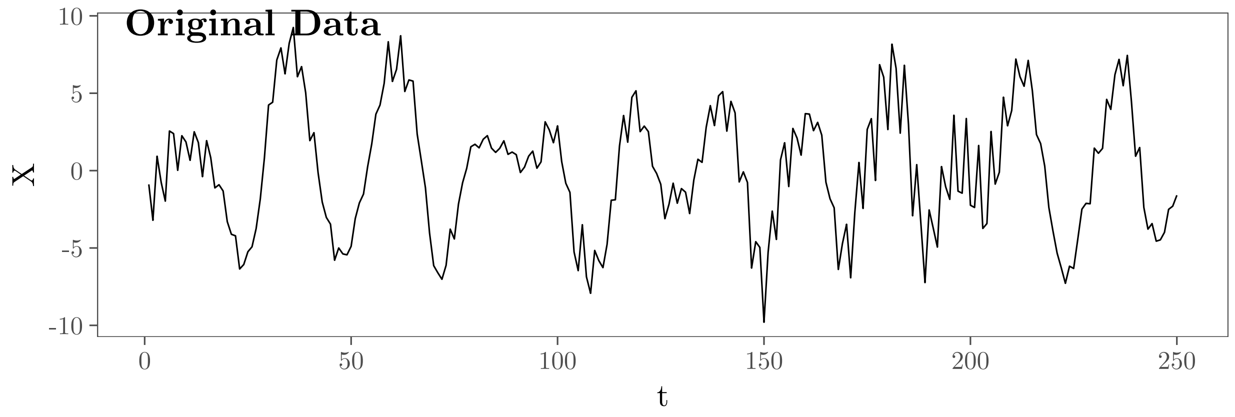
9.3 Types of filters:

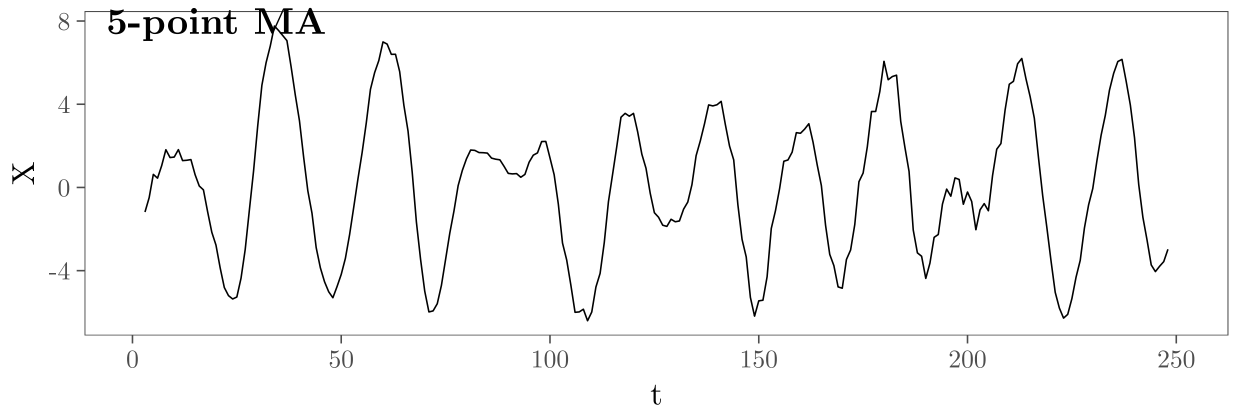
- Low pass filter
  - Filters out high frequency
  - Such as 5 point moving average, it smooths
- High pass filter
  - Leaves high freq but removes low freq
  - Such as differencing

9.3.1 An example in R

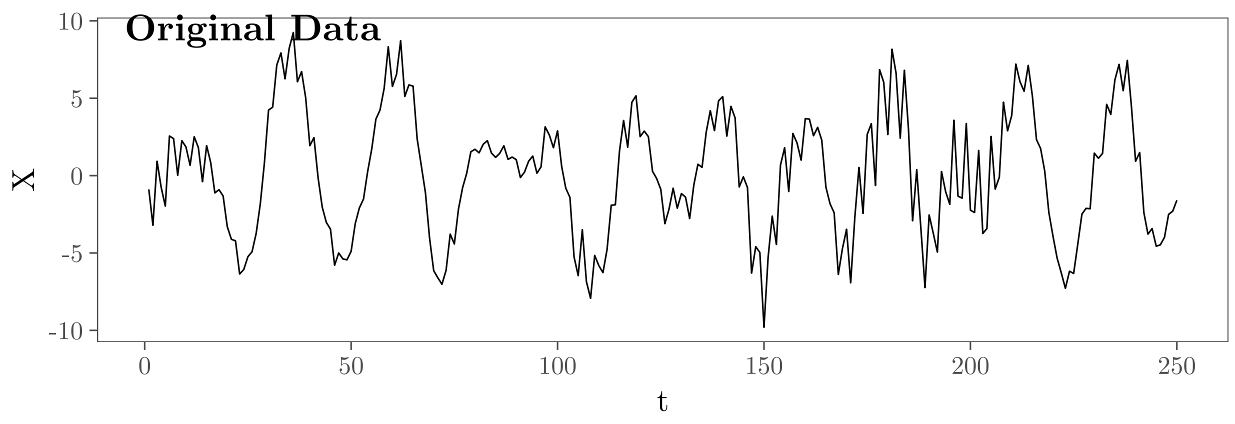
```r
mafun <- function(xs, n) {
    stats::filter(xs, rep(1, n))/n
}
dfun <- function(xs, n) {
    diff(xs, lag = n)
}
th <- ggthemes::theme_few()
data(fig1.21a)
ma <- mafun(fig1.21a, 5)
d <- dfun(fig1.21a, 1)
fp <- tplot(fig1.21a) + th
mp <- tplot(ma) + th
dp <- tplot(d) + th
```

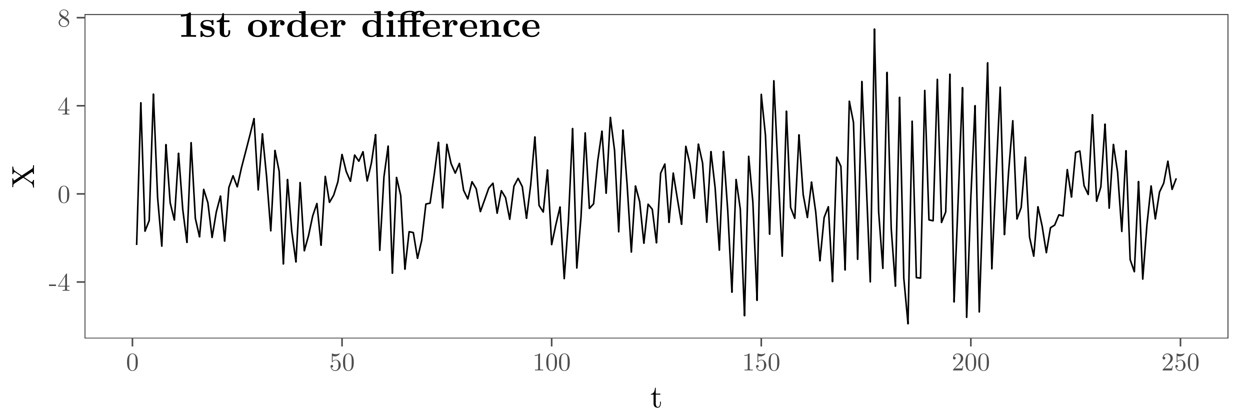
And here lets view the 5 point moving average





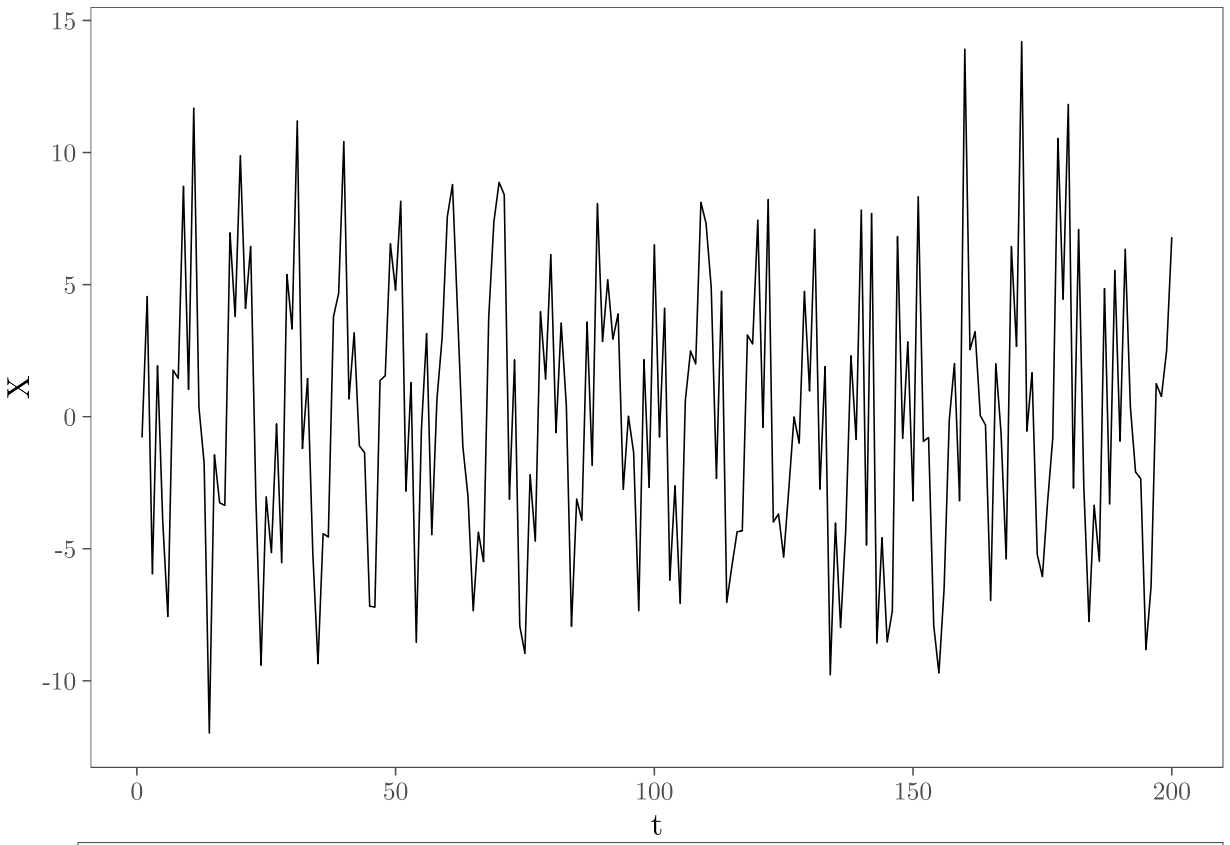
And now a look at the difference filter

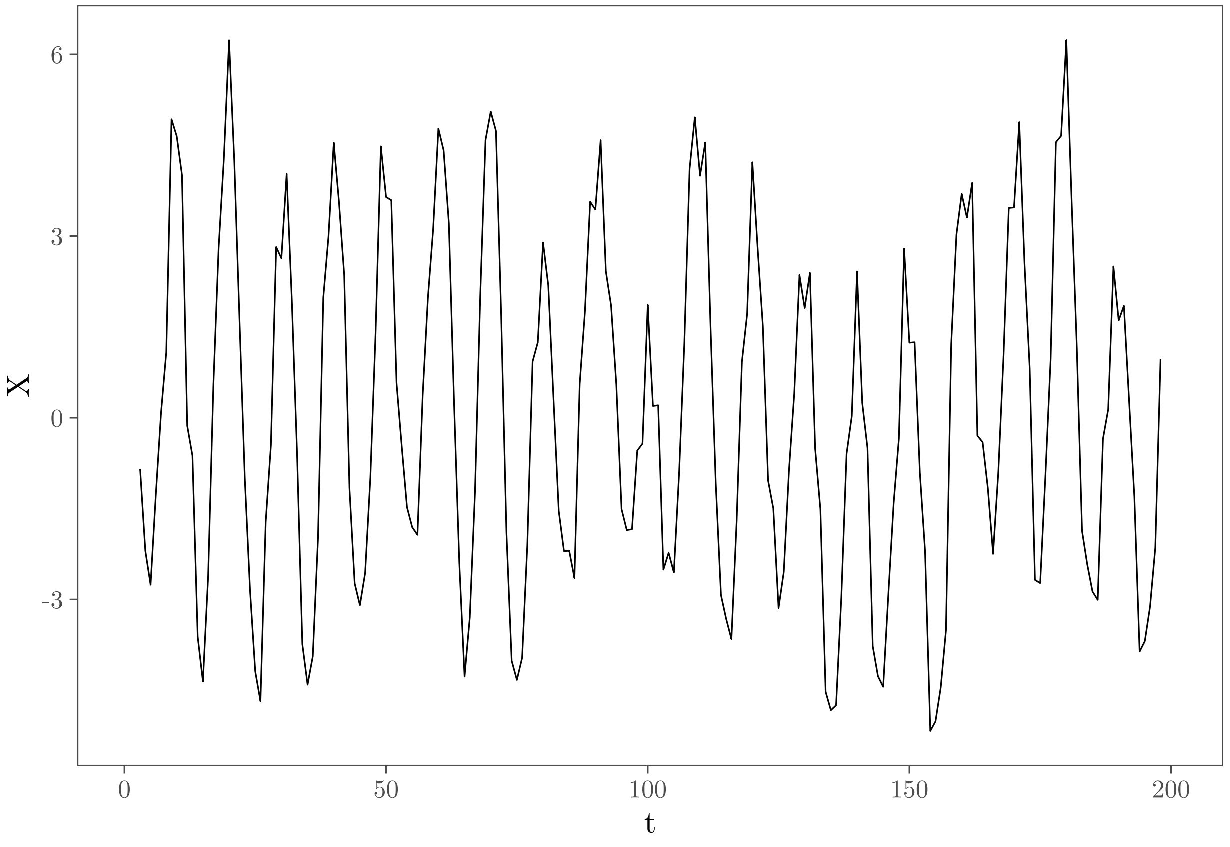


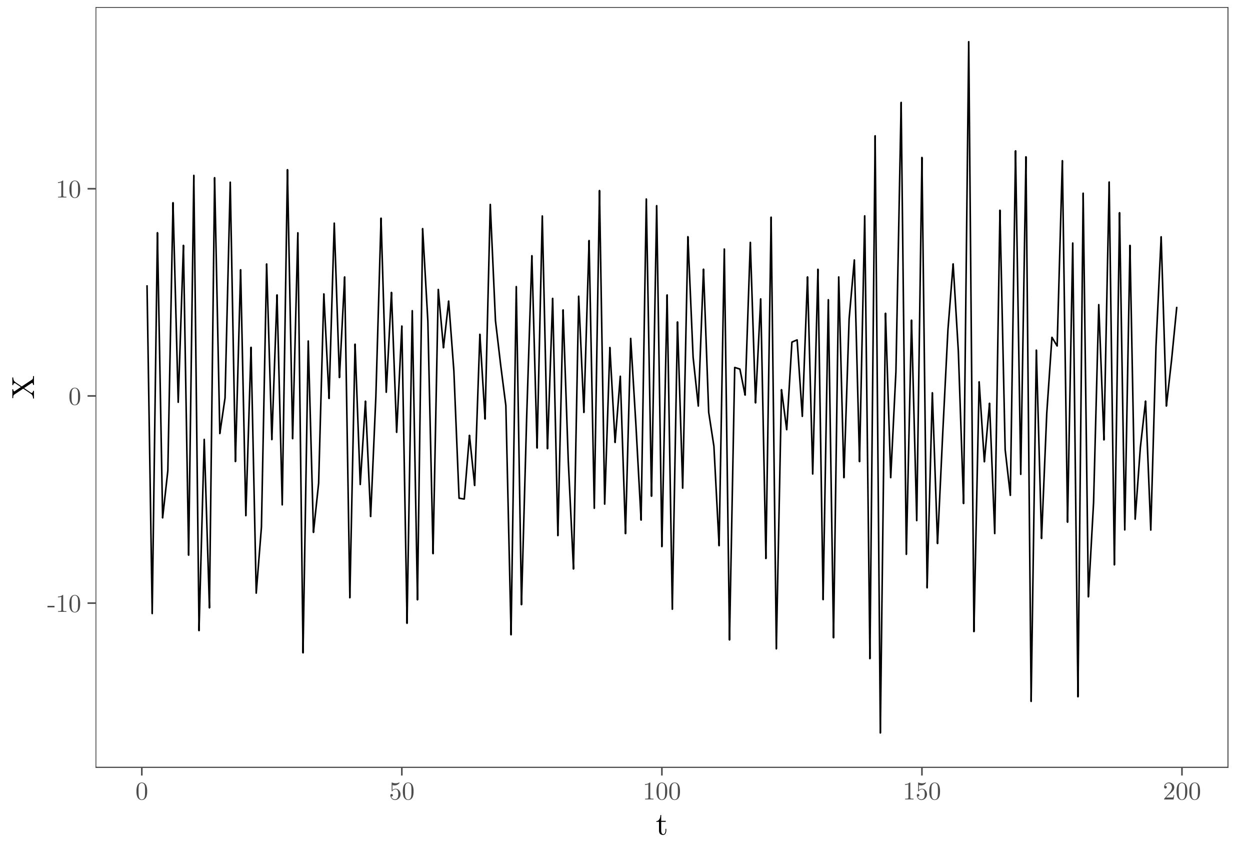


```r
rlz <- gen.sigplusnoise.wge(200, coef = c(5, 3), freq = c(0.1, 0.45), vara = 10,
    sn = 1, plot = F)
```

```r
pfun <- function(x, n, l) {
    fp <- tplot(x) + th
    mp <- tplot(mafun(x, n)) + th
    dp <- tplot(dfun(x, l)) + th
    list(original = fp, MA = mp, dif = dp)
}
plts <- pfun(rlz, 5, 1)
plts$original
plts$MA
plts$dif
```

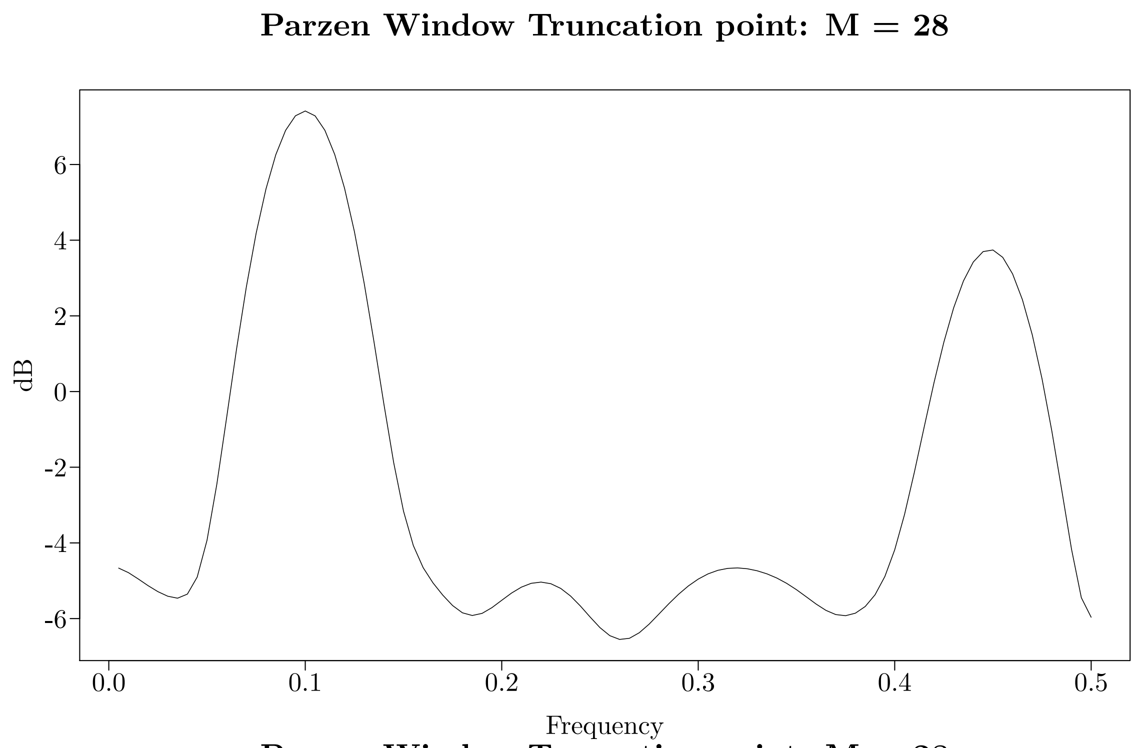

15
10
5
0
-5
-10
X
0
50
100
150
200
t

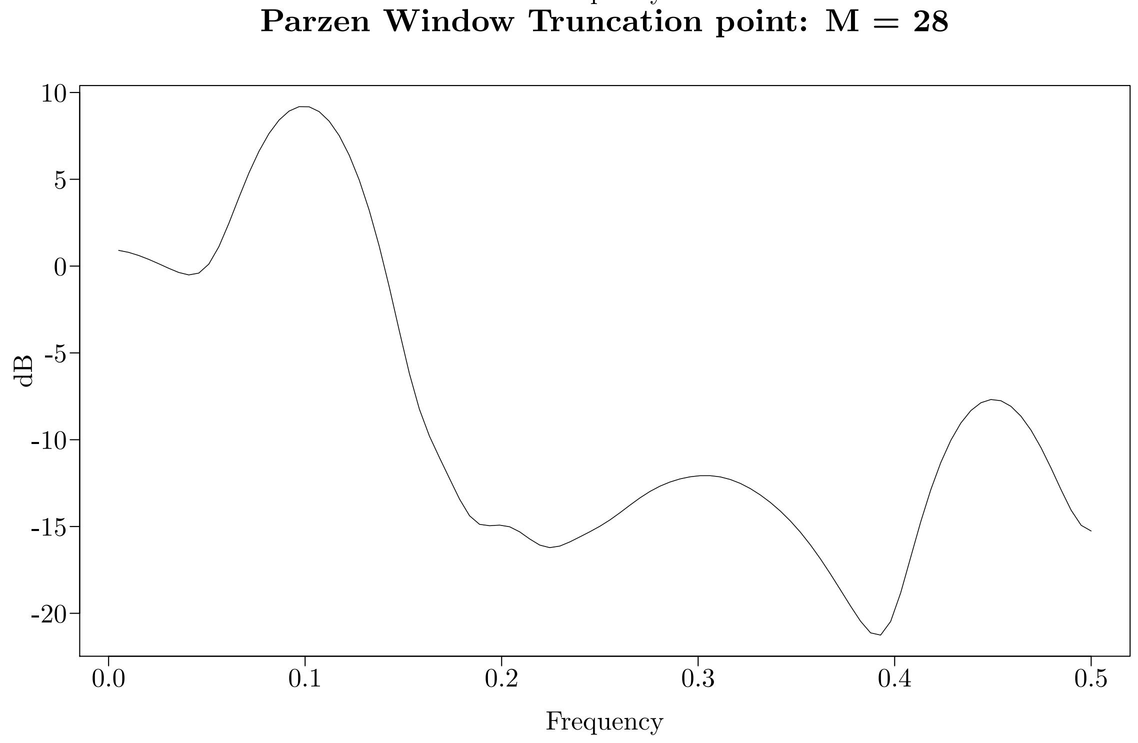

6
3
0
-3
X
0
50
100
150
200
t

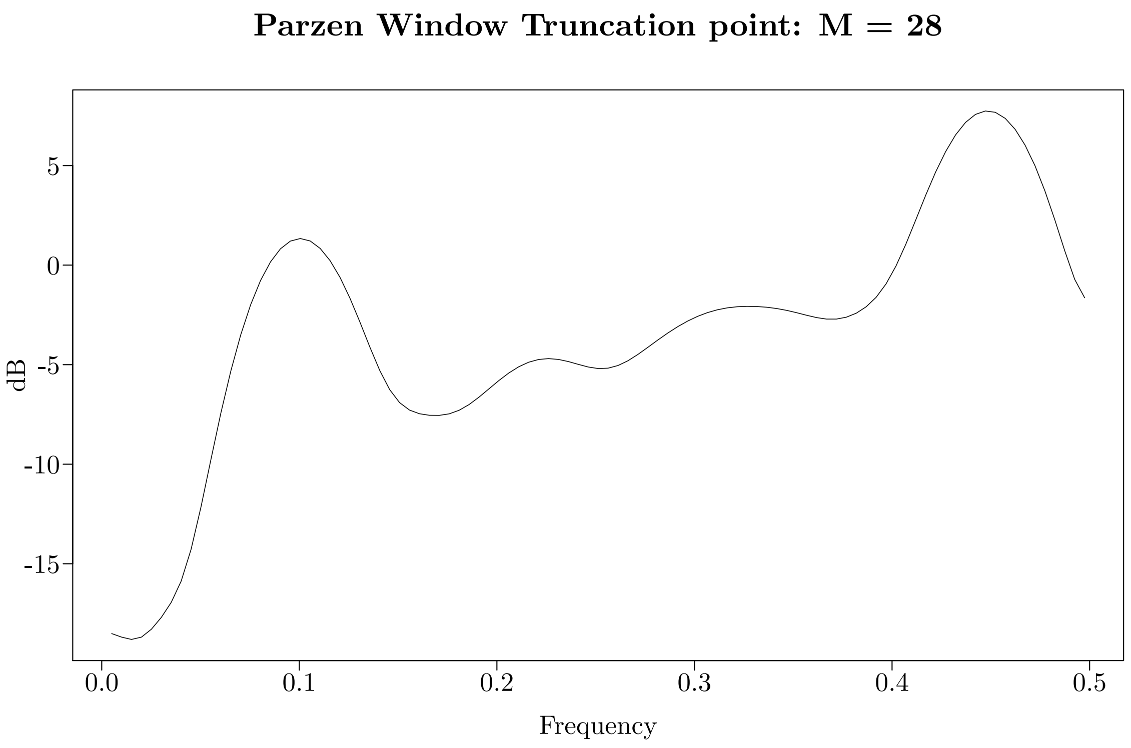


```
parzen.wge(rlz)
parzen.wge(na.omit(mafun(rlz, 5)))
parzen.wge(na.omit(dfun(rlz, 1)))
```

9.3.2.2 Some Parzen Windows


Parzen Window Truncation point: M = 28
6
4
2
0
-2
-4
-6
dB
0.0
0.1
0.2
0.3
0.4
0.5
Frequency


Parzen Window Truncation point: M = 28
10
5
0
-5
-10
-15
-20
dB
0.0
0.1
0.2
0.3
0.4
0.5
Frequency


Parzen Window Truncation point: M = 28
dB
5
0
-5
-10
-15
0.0
0.1
0.2
0.3
0.4
0.5
Frequency

Chapter 10

GLPs

a GLP is a linear filter with a ***white noise input***

We take white noise input and put it through a filter and get something that isnt white noise.

$$\sum_{j=0}^{\infty} \psi_j a_{t-j} = X_t - \mu$$

GLP's are an infinite sum of white noise terms. This might be weird now but later on this concept will return.

The ψ_j's are called psi weights, these will be useful again later. AR, MA, and ARMA are all special cases of GLP's, and will be useful when we study confidence intervals.

10.1 AR(1) Intro

AR(p) in which p = 1

We will go through well known forms of the AR(1) model.

$$X_t = \beta + \phi_1 X_{t-1} + a_t$$

$$\beta = (1 - \phi_1)\mu$$

Beta is moving average constant

We are saying that the vluse of x depends on some constant, the previous value of x, and some random white noise. Looks a lot like regression except there is a wierd variable

we can rewirte tis as

$$X_t = (1 - \phi_1)\mu + \phi_1 X_{t-1} + a_t$$

An are 1 process is stationary iff the magnitude of ϕ_1 is less than 1

We will deal with this a lot in the near future

10.1.1 AR(1) math zone

$$E\left[X_t\right] = \mu?$$

$$E\left[X_t\right] = E\left[1-\phi_1\mu\right] + E\left[\phi_1 X_{t-1}\right] + E[a_t]$$

We can rewrite this as

$$E\left[X_t\right] = 1-\phi_1\mu + \phi_1 E\left[X_t\right] + 0$$

$$E\left[X_t\right]\left(1-\phi_1\right) = 1-\phi_1\mu$$

$$E\left[X_t\right] = \mu$$

Mean does not depend on T

The variance also does not, and if phi1 is less than one variance is finite

$$\sigma_X^2 = \frac{\sigma_a^2}{1-\phi_1^2}$$

and rhok is phi1 to the k

$$\rho_k = \phi_1^k, k \geq 0$$

Spectral Density of AR(1) also does not depend on time, it just monotonically increases or decreases depending on phi1:

$$S_X(f) = \frac{\sigma_a^2}{\sigma_X^2}\left(\frac{1}{\mid 1-\phi_1 e^{-2\pi i f}\ \mid^2}\right)$$

10.1.2 the zero mean form of AR1

with zero mean,

$$X_t = \phi_1 X_{t-1} + a_t$$

OR

$$X_t - \phi_1 X_{t-1}$$

A

10.1.3 AR1 with positive phi

remember we have

$$\rho_k = \phi_1^k$$

With positive ϕ_1, we have:

realizations are wandering and aperiodic

Autocorrelations are damped exponentials

Spectral density Peaks at zero.

Higher values = stronger versions of these characterstics

10.1.4 AR1 with negative phi

We have:

realization are oscillating

Autocorrelations are damped and alternating(negative to a power you fool)

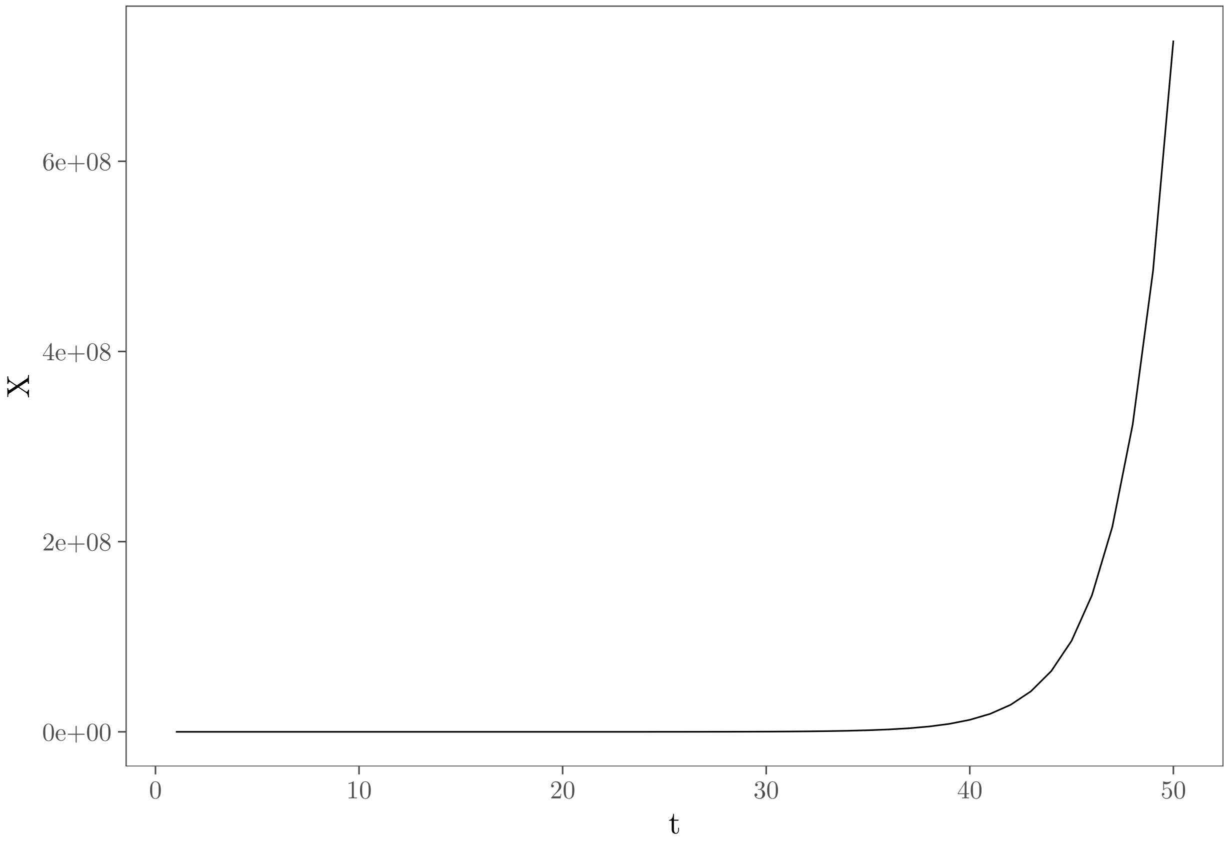
Spectral density* has a peak at f = 0.5 (cycle length of 2)

Higher magnitude = stronger characteristics

10.1.5 Nonstationary

if phi = 1 or phi> 1, we have that realizations are from a nonstationary process. with it equal to one, it looks ok, and is actualy a special ARIMA model. WIth it > 1, we have crazy explosive realizations. Check this out:

```
nonstarma <- function(n, phi) {
    x <- rep(0, n)
    a <- rnorm(n)
    x[1:n] <- 0
    for (k in 2:n) {
        x[k] = phi * x[k - 1] + a[k]
    }
    tplot(x)
}

nonstarma(n = 50, phi = 1.5) + th
```



Chapter 11

Characteristic Equations for AR(1) Models

Consider the zero-mean form of an AR(1) model

$$X_t - \phi_1 X_{t-1} = a_t$$

We can rewtite in operator form as

$$(1 - \phi_1 B)X_t = a_t$$

Where B is an operator that turns X_t into X_{t-1}. We can then rewrite this akgebraically (for solving purposes) as a characteristic equation:

$$1 - \phi_1 Z = 0$$

.

Now recall that AR(1) is stationary $iff \mid \phi_1 \mid$ is less than 1. Lets solve the characteristic equation for the root now:

$$r = z = \frac{1}{\phi_1}$$

If X_t is stationary , $\mid r \mid$ is greater than one. This does not feel important now, but when we get to higher order time series this will save us a lot of thinking. Lets look at a numerical example just to make sure we got it:

$$X_t = -1.2X_{t-1} + a_t$$
$$X_t + 1.2X_{t-1} = a_t$$
$$(1 + 1.2B)X_t = a_t$$
$$1 + 1.2Z = 0$$
$$r = z = -\frac{1}{1.2} = -0.8333333$$

Note that in this case, $\phi_1 = -1.2$\$. This is a weird thing with this notation, get used to it.

Chapter 12

AR(2) models

$$X_t = \beta + \phi_1 X_{t-1} + \phi_2 X_{t-2} + a_t$$

where $\beta = (1 - \phi_1 - \phi_2)\,\mu$ is the moving average constant.

12.1 Notes

It looks like MLR, however the independetn variables are values of the dependent variable at two previous time periods

12.2 Stationarity in AR(2)

$$E[X_t] = \mu$$

$$\sigma_X^2 = \frac{\sigma_a^2}{1 - \phi_1\rho_1 - \phi_2\rho_2}$$

Instead of depending on positive or negative roots, like AR(1), the behavior of the autocorrelations and spectral density depends on whether or not the roots of the characteristic equation are complex. This means it is already hard to tell from the equation whether or not it is stationary just by looking at the equation.

12.3 AR(2) zero mean form

If $\mu = 0$, then the AR(2) model takes the form :

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + a_t$$

We can study this much easier and then add the mean back later. A more common way of writing this, and more useful to us is the form:

$$X_t - \phi_1 X_{t-1} - \phi_2 X_{t-2} = a_t$$

12.4 More on backshift operator notation

Remember, we defined the backshift operator as B such that $BX_t = X_{t-1}$

We can extend this and say:

$$B^k X_t = X_{t-k}$$

This means we can rewrite:

$$X_t - \phi_1 B X_t - \phi_2 B^2 X_t = a_t$$

$$X_t \left[1 - \phi_1 B - \phi_2 B^2\right] = a_t$$

or, more tersely:

$$\phi(B) X_t = a_t$$

Where $\phi(B) = 1 - \phi_1 B - \phi_2 B^2$

We can rewrite this $\phi(B)$ as a characteristic equation now:

$$1 - \phi_1 Z - \phi_2 Z^2 = 0$$

Lets quickly expand on our quadratic solver to get the roots of an AR(2) model:

```
ar2solver <- function(p1, p2) {
    res <- as.complex(quad_form(p2, p1, 1))
    mag <- Mod(res)
    absrecip <- 1/mag
    data.frame(roots = res, magnitude = mag, abs.recip = absrecip)
}
ar2solver(0.4, 0.8)
```

```
##          roots magnitude abs.recip
## 1 -0.25-1.09i  1.118302 0.8942126
## 2 -0.25+1.09i  1.118302 0.8942126
```

12.5 Characteristic equation

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$1 - \phi_1 Z - \phi_2 Z^2 = 0$$

Just solve the quadratic equation using the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Two cases: * Two real roots * Two complex roots

12.6 Key result

An AR(2) is stationary ***iff*** all roots are outside the unit circle

12.6.1 Two real roots

We can factor the AR(2) into two AR(1)s

- one part positive one part negative
  - Wandering but oscillating
  - Damping ACF but slightly oscillatory
  - S(f) has a peak at 0 and at 0.5
- Two Positive roots
  - Super wandering
  - Super Damped
  - S(f) peak at 0
- Two Negative
  - super oscillating

12.6.2 Complex conjugate roots

Stationary AR(2)'s with complex conjugate roots have an autocorrelation function which looks more like a damped sinusoid instead of a damped exponential, and a system frequency, f_0 defined as+

$$f_0 = \frac{1}{2\pi}\cos^{-1}\left(\frac{\phi_1}{2\sqrt{-\phi_2}}\right)$$

. Pretty easy stuff.

Chapter 13

AR(p)

This is basically the same thing, with the same conclusion, except instead of having clearly defined behavior based on the sine of the roots, the closer the absolute value of the roots is to one the more dominant that behavior is in the realization. To solve these, we will use this brilliant `factor.wge` function:

```r
library(tswge)
factor.wge(phi = c(0.7, 0.4, 0.3, -0.4, -0.7, 0.1, -0.1))
```

```
##
## Coefficients of Original polynomial:
## 0.7000 0.4000 0.3000 -0.4000 -0.7000 0.1000 -0.1000
##
## Factor                 Roots                Abs Recip    System Freq
## 1-2.0903B+1.2507B^2    0.8357+-0.3182i      1.1183       0.0579
## 1+0.7598B+0.8037B^2   -0.4727+-1.0104i      0.8965       0.3196
## 1+0.8141B             -1.2283               0.8141       0.5000
## 1-0.1836B+0.1222B^2    0.7512+-2.7602i      0.3496       0.2077
##
##
```

We can use the absolute reciporacal (speling) of the root to tell if it is stationary easily. If it is less than one, we have stationarity. It has the same property in that if it is closer to one, that is a stronger characteristic in the realization.

Chapter 14

MA(q) Models

14.1 Properties and Characteristics

We use MA models to model stationary data. They are not quite as useful as AR models, but in conjunction with them, we can create ARMA(p,q) models and so forth.

14.1.1 A quote for your thoughts:

"All models are wrong... but some are useful" - George Box

14.2 MA(q)

First of all, we need to define the equation for a Moving Average Model of order q:

$$X_t = \mu + a_t - \theta_1 a_{t-1} - ... - \theta_q a_{t-q}$$

Inspecting this equation, we see that a MA(q) model is a finite GLP, and therefore ***Always stationary***. **reminder**: a GLP is defined as :

$$X_t = \mu + \sum_{j=0}^{\infty} \psi_j a_{t-j}$$

We can now define a MA(q) model as a GLP in which:

$$\psi_0 = 1, \psi_1 = -\theta_1, ..., \psi_q = -\theta_q, \psi_k = 0, k > q$$

14.3 Operator Zero-Mean Form

$$X_t = (1 - \theta_1 B - ... - \theta_q B^q) a_t$$

14.3.1 Characteristic Equation

$$1 - \theta_1 z - ... - \theta_q z^q = 0$$

We can solve this just like the AR side, except this is with white noise terms.

14.3.2 Some definitions:

$$E(X_t) = \mu$$

14.3.2.1 For MA(1)

$$\sigma_X^2 = \sigma_a^2(1+\theta_1^2)$$

$$\rho_0 = 1, \rho_1 = \frac{-\theta_1}{1+\theta_1^2}, \rho_k = 0, k > 1$$

For pure MA they do not damp exponentially! (theoretically) This is a defining piece of info.

$$S_X(f) = \frac{\sigma_a^2}{\sigma_X^2} \mid 1 - \theta_1 e^{-2\pi i f} \mid^2$$

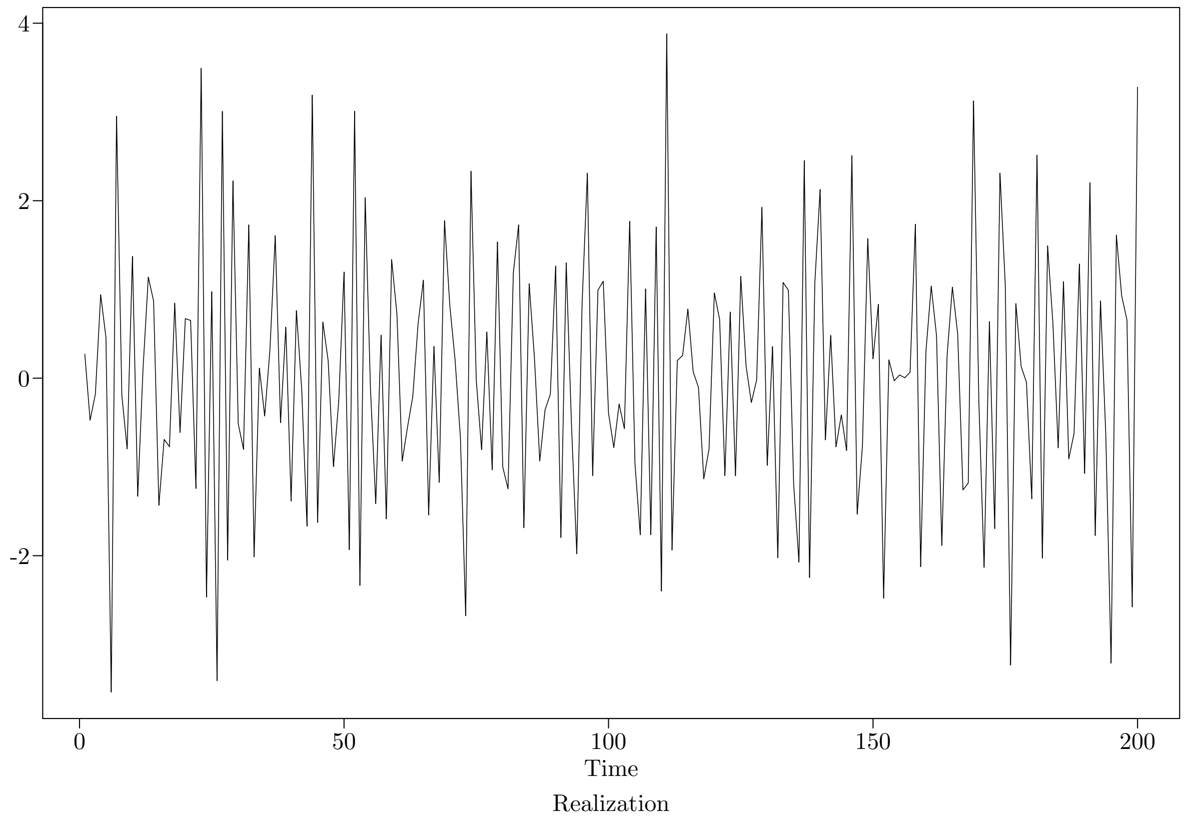
From this we will see that MA spectral densities will have dips rather than peaks (again a function of white noise).

Chapter 15

Some examples of MA(1)

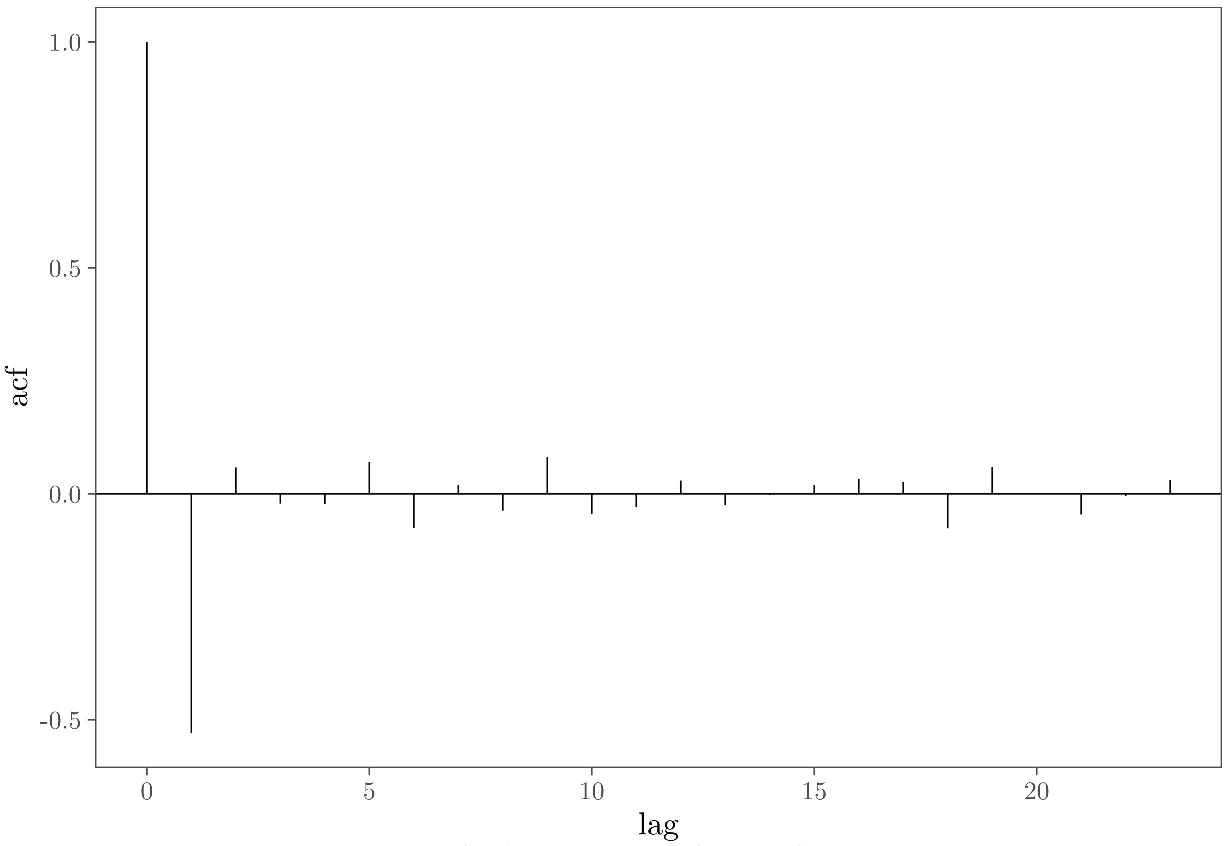
15.0.1 Positive theta1

```r
xs <- gen.arma.wge(n = 200, theta = 0.99)
```

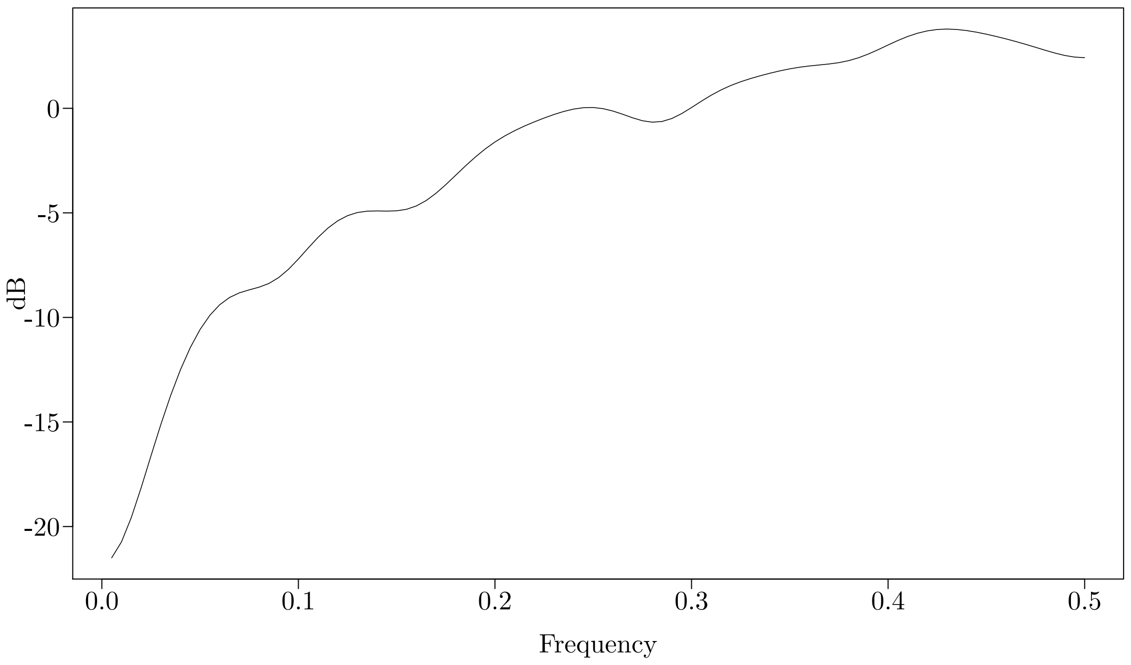


Lets check this out now

```r
ggacf(xs) + th
parzen.wge(xs)
```



Parzen Window Truncation point: M = 28



Lets get some math done too and check out the ACF at lag 1

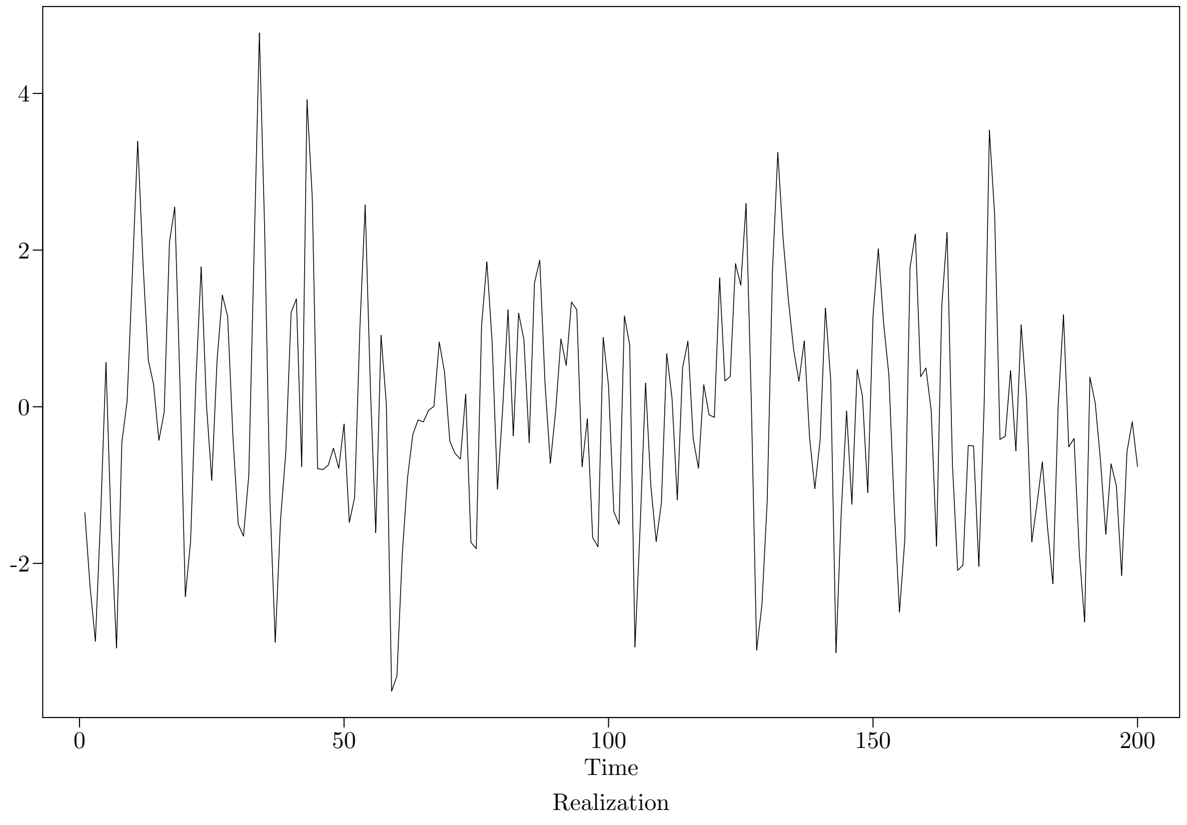
```
macf1 <- function(theta) {
    num <- -theta
    denom <- 1 + theta^2
    num/denom
}
macf1(0.99)
# [1] -0.4999747
```

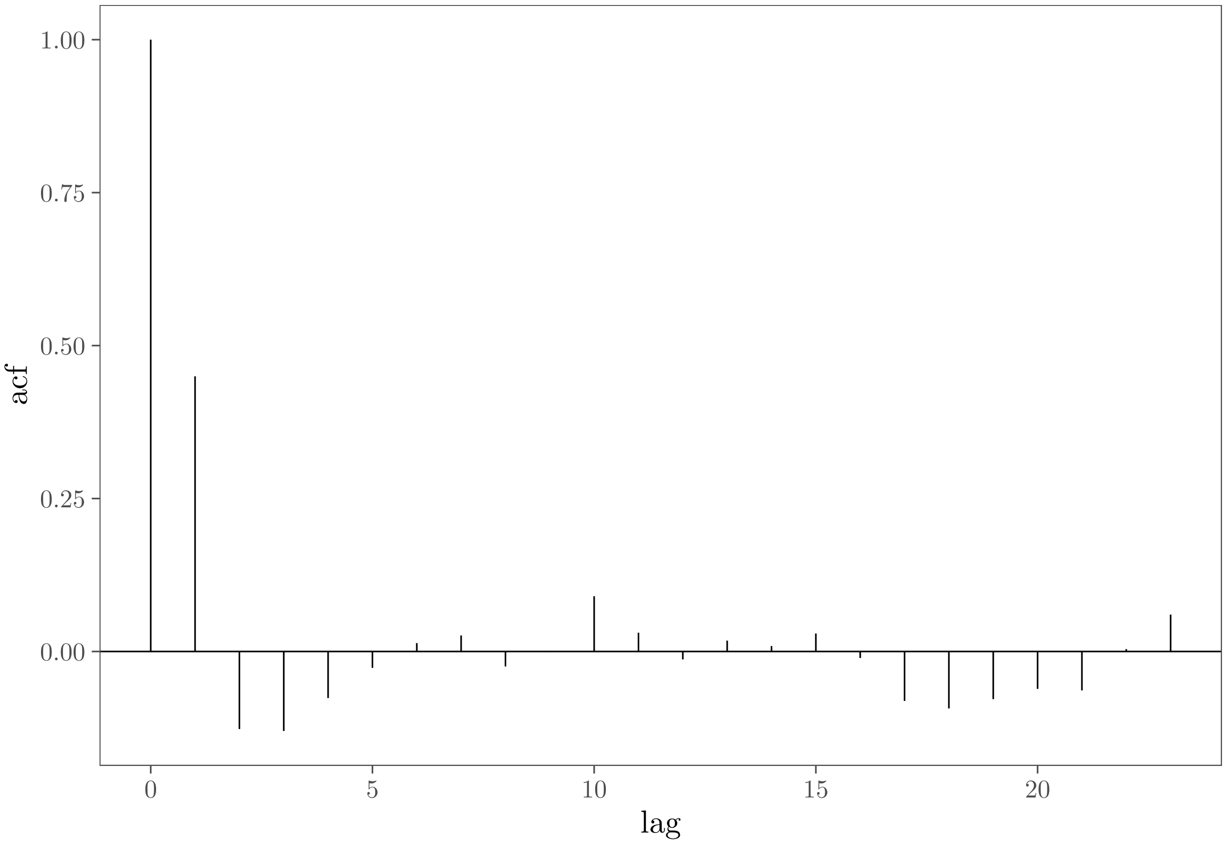
From this we can also see that the maximum possiblevalue of autocorrelation at lag 1 is 0.5.

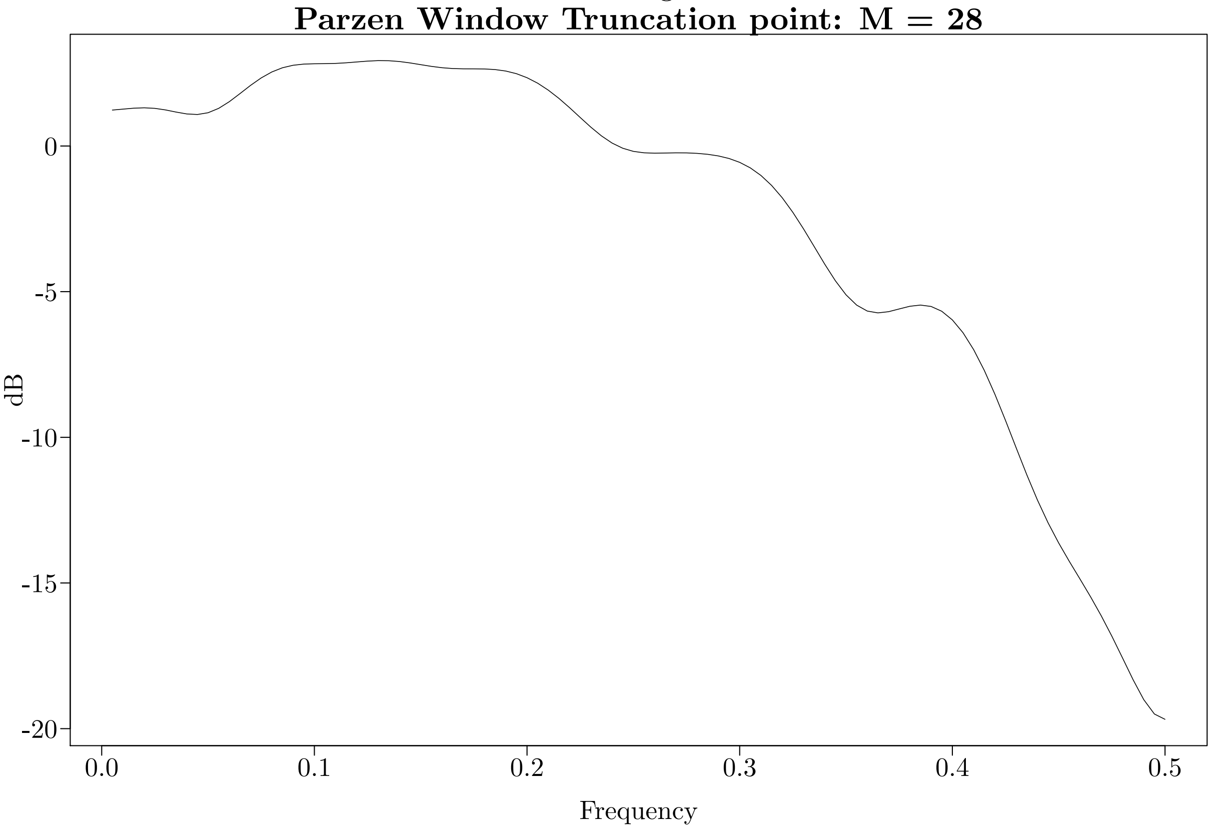
15.0.2 Negative theta

Now lets try it out with a negative theta:

```
xs <- gen.arma.wge(n = 200, theta = -0.99)
ggacf(xs) + th
parzen.wge(xs)
macf1(-0.99)
# [1] 0.4999747
```

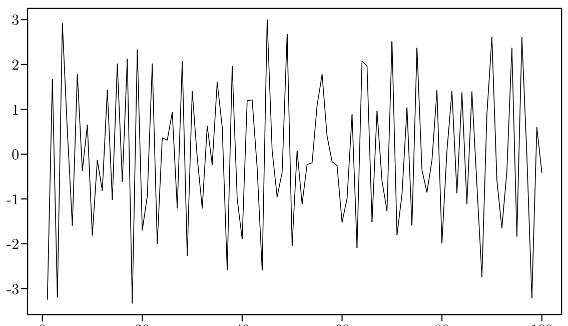




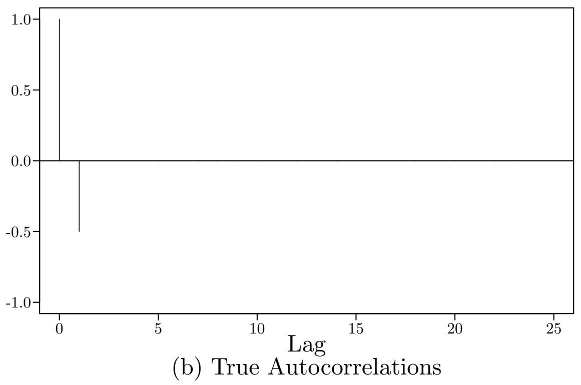


Note: the negative autocorrelation of a positive theta, as well as the dip at 0 in the spectral density tell us it is going to oscillate a bit more than the negative theta. Lets check out the true autocorrelations and spectral densities of these models:

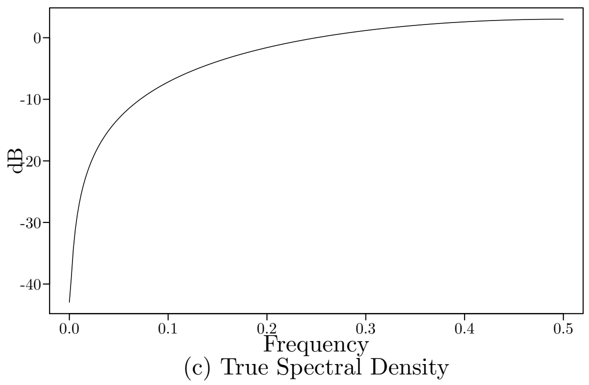
```
plotts.true.wge(theta = 0.99)
plotts.true.wge(theta = -0.99)
```



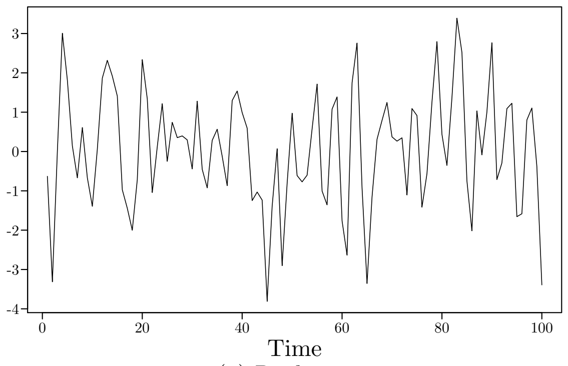
(a) Realization



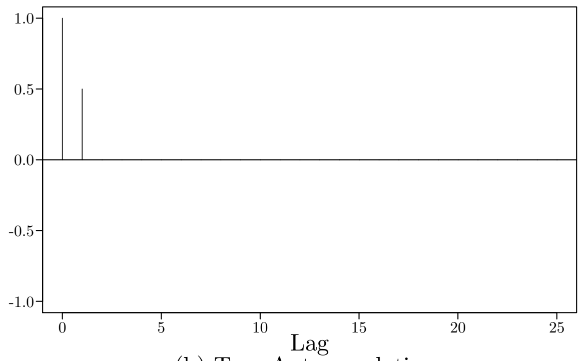
(b) True Autocorrelations



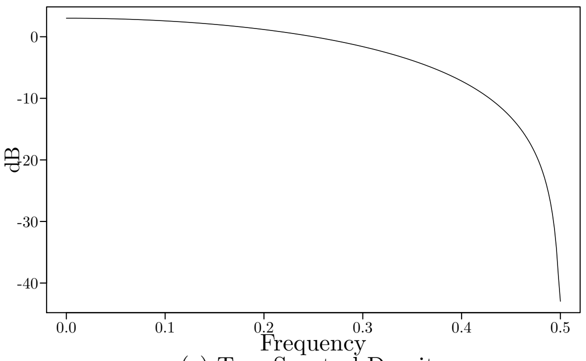
(c) True Spectral Density



(a) Realization



(b) True Autocorrelations



(c) True Spectral Density

Chapter 16

MA(2)

A MA(2) model behaves pretty similarly, however for our autocorrelations we have:

$$\rho_0 = 1$$

$$\rho_1 = \frac{-\theta_1 + \theta_1\theta_2}{1 + \theta_1^2 + theta_2^2}$$

$$\rho_2 = \frac{-\theta_2}{1 + \theta_1^2 + theta_2^2}$$

$$\rho_k = 0; k > 2$$

This weird behavior for rho_k is why we dont really use this on real data.

16.1 Invertibility

Consider two MA(1) models, with theta equal to 0.8 and 1.25. Then, let us consider their autocorrelations: for theta = 0.8, $rho_1 = -0.4878049$, and for theta = 1.25, it is -0.4878049 (use the `macf1` function). These two models can have the same ACF, which is not good. Model multiplicity is undesirable. This happens I believe due to some magic, similar to aliasing. How do we tell them apart?

16.1.1 Criteria for invertibility

A MA model is invertible iff all roots of the model are outside of the unit circle. We can just solve the MA equations as we do with the AR equations, to check for invertibility, we use the characteristic equation trick. We can also use factor.wge

Chapter 17

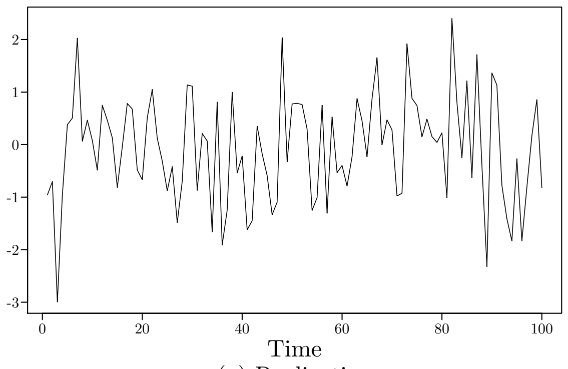
ARMA(p,q)

An ARMA model is an autoregressive moving average model. Written in operator form we have:

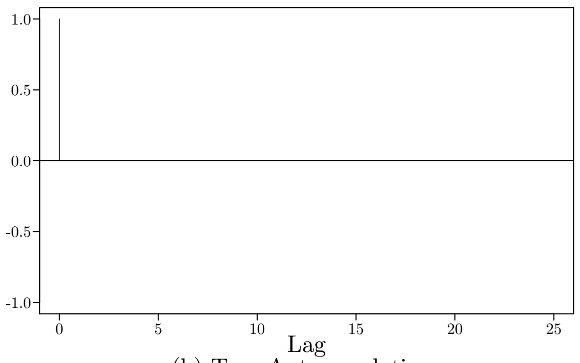
$$(1 - \phi_1 B - ... - \phi_p B^p)X_t = (1 - \theta_1 B - ... - \theta_q B^q)$$

If we remember the `plotts.true.wge` plots we did in a previous section, you may have wondered what happens when the AR and MA chave common factors? Well, they cancel out conveniently! Lets look at an example:

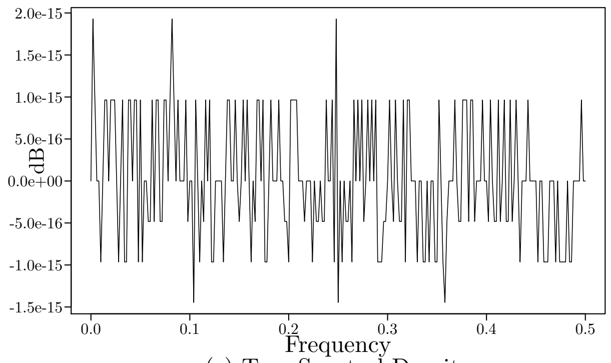
```
plotts.true.wge(phi = c(1.1, -0.9), theta = c(1.1, -0.9))
```



(a) Realization



(b) True Autocorrelations



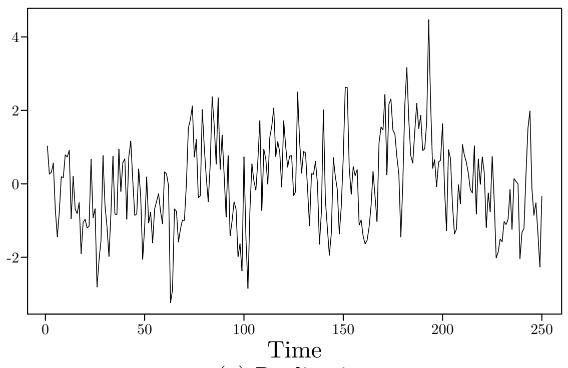
(c) True Spectral Density

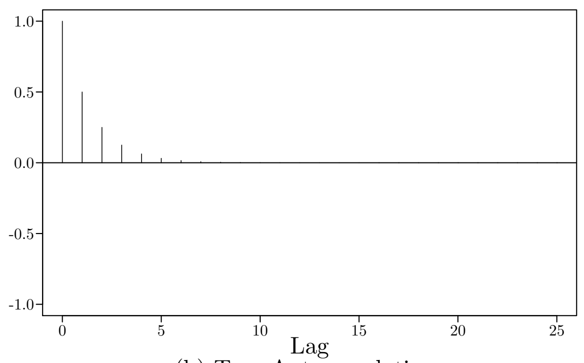
Wow! It is zero everywhere on the spectral density! BUT, it is not actually zero, as this is a logarithmic scale. This means it is actually one everywhere, and this leads us to the conclusion, along with the ACF and common sense that if all the factors between the

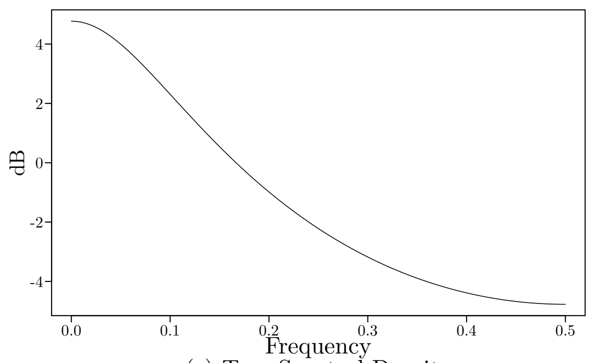
AR and MA components are the same, we are left with $X_t = a_t$, pure white noise! This allows us to conclude that factors can cancel out, at least with my pea brain.

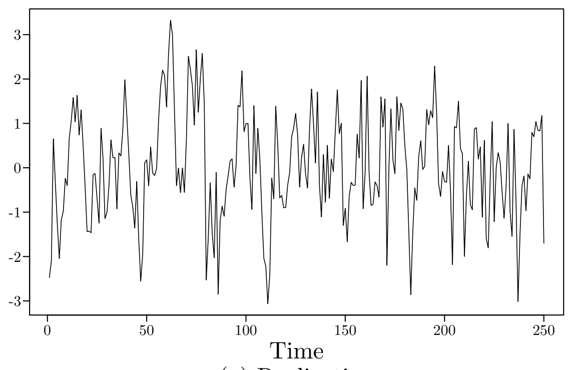
Another example:

```
plotts.true.wge(250, phi = c(1.3, -0.4), theta = 0.8)
plotts.true.wge(250, phi = 0.5)
```

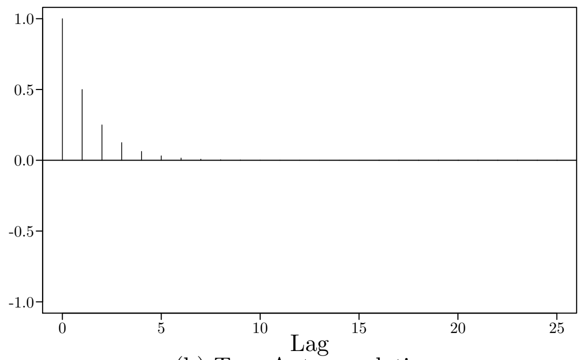

(a) Realization


(b) True Autocorrelations

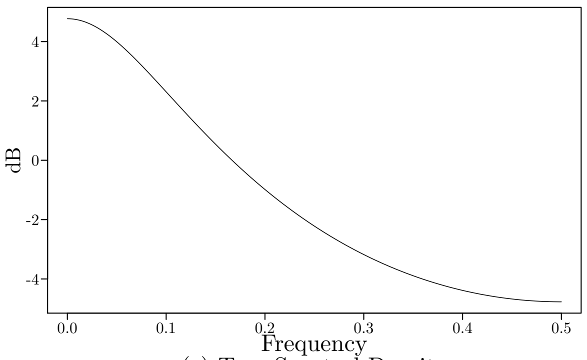

(c) True Spectral Density



(a) Realization



(b) True Autocorrelations



(c) True Spectral Density

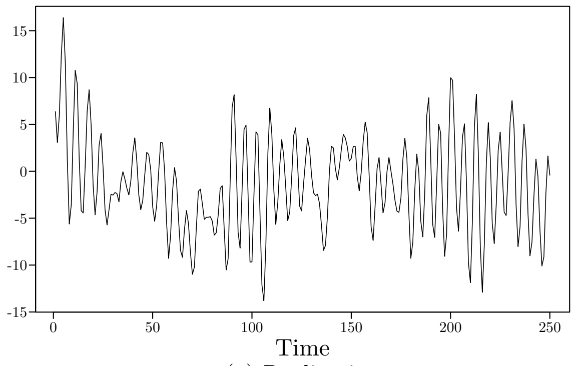
Bam! They are the same!!! Wow!

Chapter 18

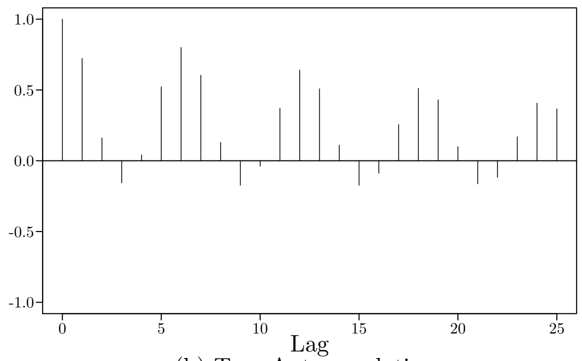
Blending AR and MA components

An ARMA(p,q) model is invertible iff the roots of the AR side ***and*** the MA side are invertible. For a process to be considered arma it is stationary and invertible. Lets check it out with plotts.true again:

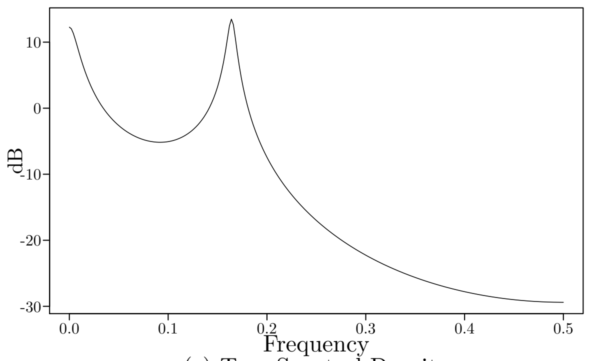
```r
plotts.true.wge(250, phi = c(1.95, -1.9, 0.9025))
plotts.true.wge(250, theta = 0.95)
plotts.true.wge(250, phi = c(1.95, -1.9, 0.9025), theta = 0.95)
```



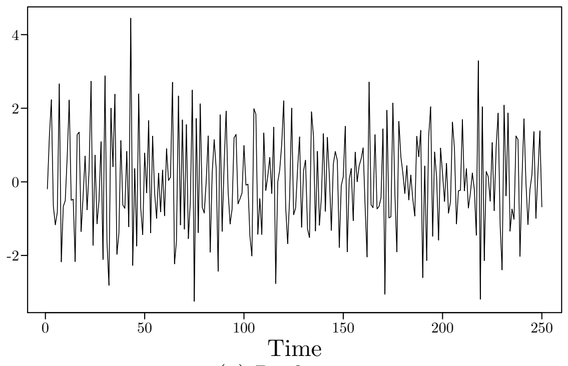
(a) Realization



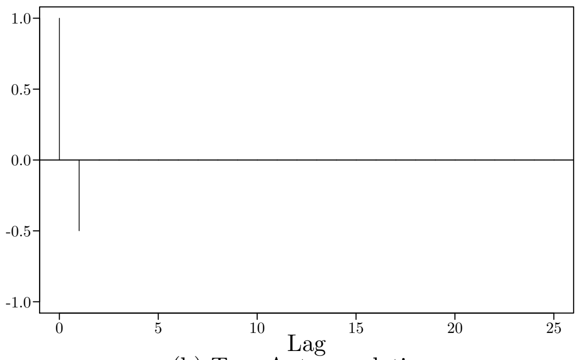
(b) True Autocorrelations



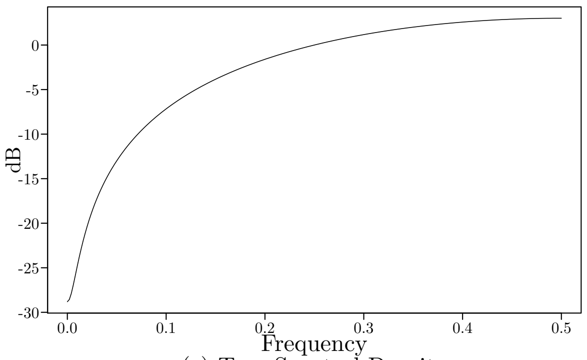
(c) True Spectral Density



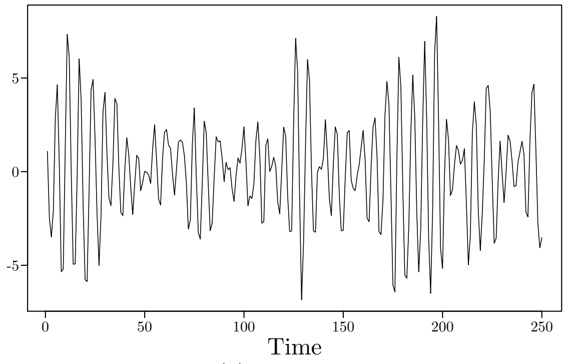
(a) Realization



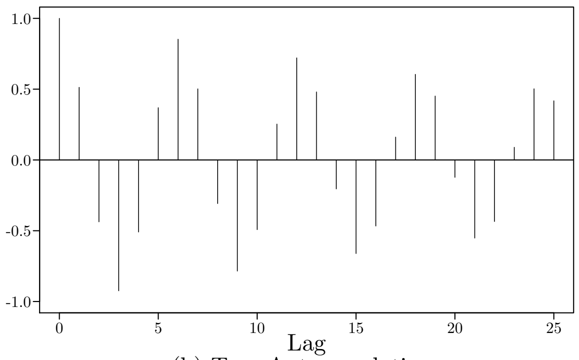
(b) True Autocorrelations



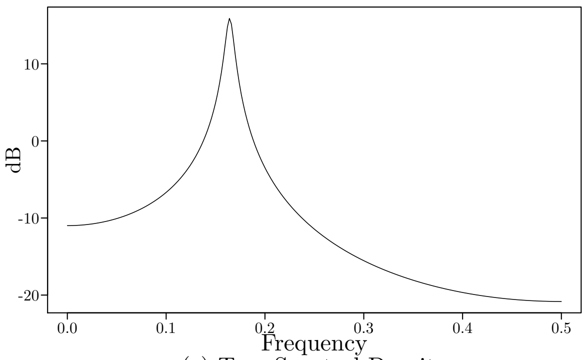
(c) True Spectral Density



(a) Realization



(b) True Autocorrelations



(c) True Spectral Density

Lets also just take a moment that aic and aic5.wge exist and tell us things. There is no need to write them here because thats a lot of work

Chapter 19

Psi weights with AR models:

$$\psi_0 = 1$$

$$\psi_1 = \phi_1$$

$$\psi_2 = \phi_1\psi_1 + \phi_2$$

$$\psi_3 = \phi_1\psi_2 + \phi_2\psi_1 + \phi_3$$

$$\psi_\ell = \sum_{i=1}^{p} \phi_i\psi_{\ell-i}$$

```r
multfun <- function(phi, l, psi) {
    sum(phi[1:l] * psi[1:l])
}
phitest <- function(phi, l) {
    if (l > length(phi)) {
        return(append(phi, rep(0, l - length(phi))))
    } else {
        return(phi)
    }
}
arpsiweights <- function(phi, theta = as.numeric(0), l) {
    phi <- phitest(phi, l)
    psi <- as.numeric(1)
    for (i in 2:l) {
        psi[i] <- multfun(phi, i - 1, rev(psi))
    }
    psi
}
```

19.1 Psi weights for ARMA

$$\psi_0 = 1$$

$$\psi_1 = \phi_1 - \theta_1$$

$$\psi_2 = \phi_1\psi_1 + \phi_2 - \theta_2$$

$$\psi_3 = \phi_1\psi_2 + \phi_2\psi_1 + \phi_3 - theta_3$$

$$\psi_\ell = \sum_{i=1}^{p} \phi_i \psi_{\ell-i} - \theta_i$$

```
psiweights <- function(phi, theta = as.numeric(0), l) {
    phi <- phitest(phi, l)
    psi <- as.numeric(1)
    theta <- c(0, theta)
    theta <- phitest(theta, l)
    for (i in 2:l) {
        psi[i] <- multfun(phi, i - 1, rev(psi)) - theta[i]
    }
    psi
}
```

Chapter 20

ARIMA and ARUMA

Ok this is gonna be a light week, basically, in our notation, we have that ARIMA is an ARMA model with a wandering term, eg an equation with a root at exactly 1, and just one of them, while an ARUMA is a model where the backshift operator is squared. Mathematically we have:

20.1 ARIMA

$$(1-B)^n X_t = a_t$$

is an ARIMA (0,n,0) model. Note that because this is not stationary, we cannot view the plotts.true.wge :(

20.2 ARUMA

$$(1-B^n) X_t = a_t$$

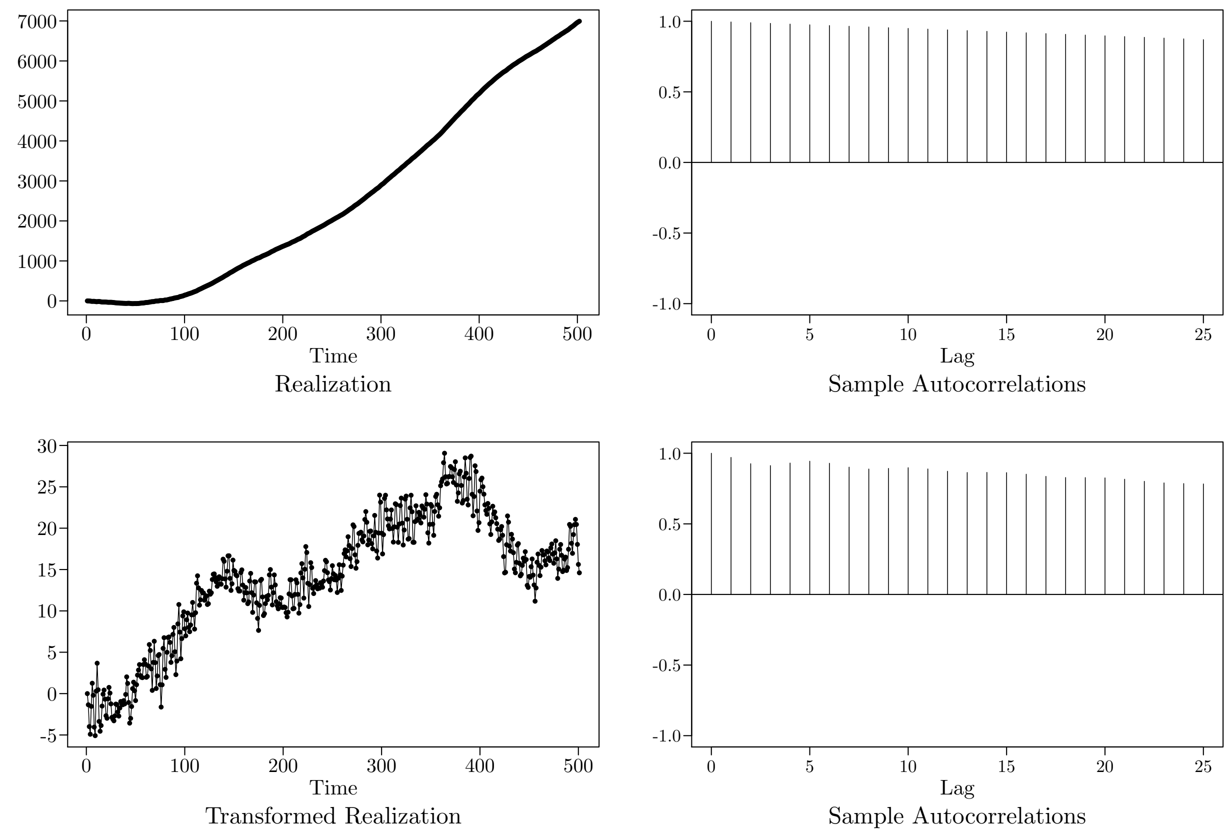
is an ARUMA(0,n,0) model, or an ARIMA(0,0,0) model with a seasonal component of order n. This is pretty easy.

20.3 But what do they do????

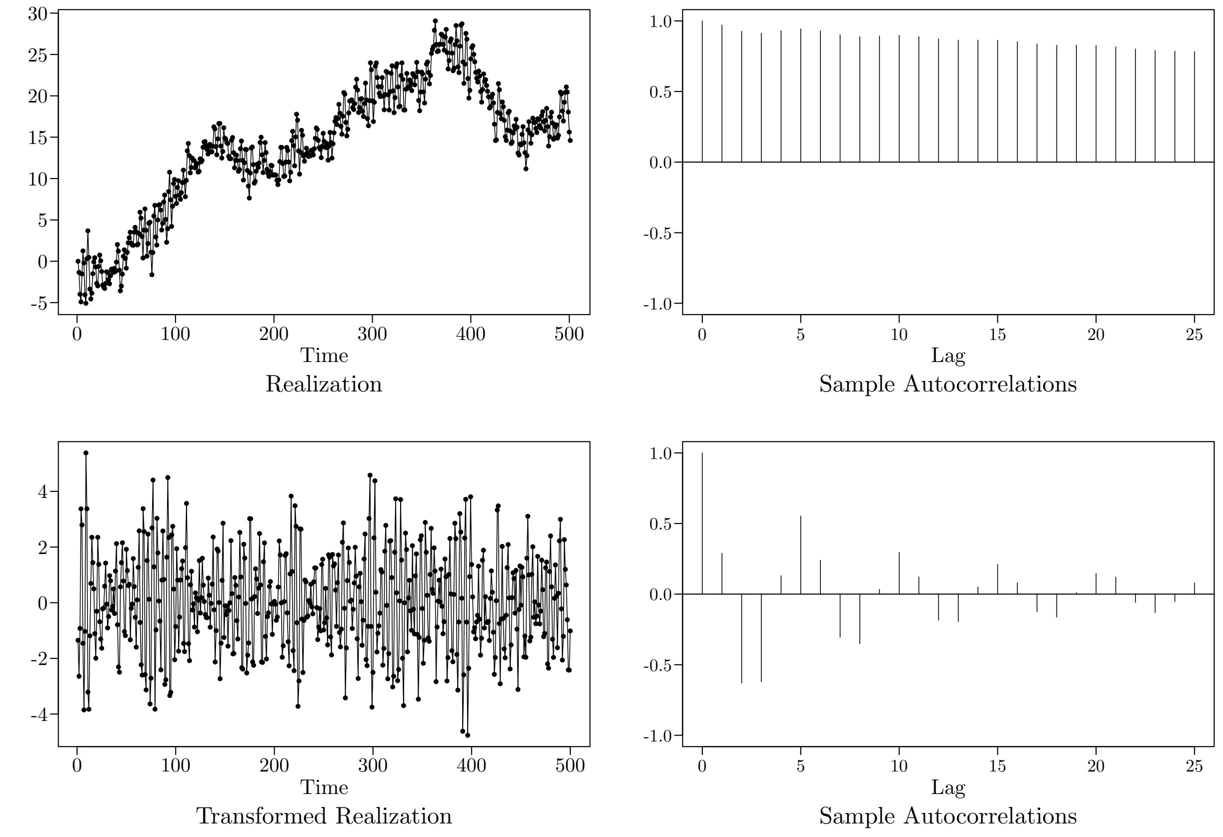
ARIMA adds a wandering component, since the root is exactly one, while ARUMA is a seasonal model, because of sines and cosines and Euler's theorem etc. Dont worry too much about it, try some stuff out with `artrans.wge`, `gen.arima.wge`, `gen.aruma.wge` etc. Here are some wrapper functions for artrans.wge:

20.3.1 ARIMA Transform

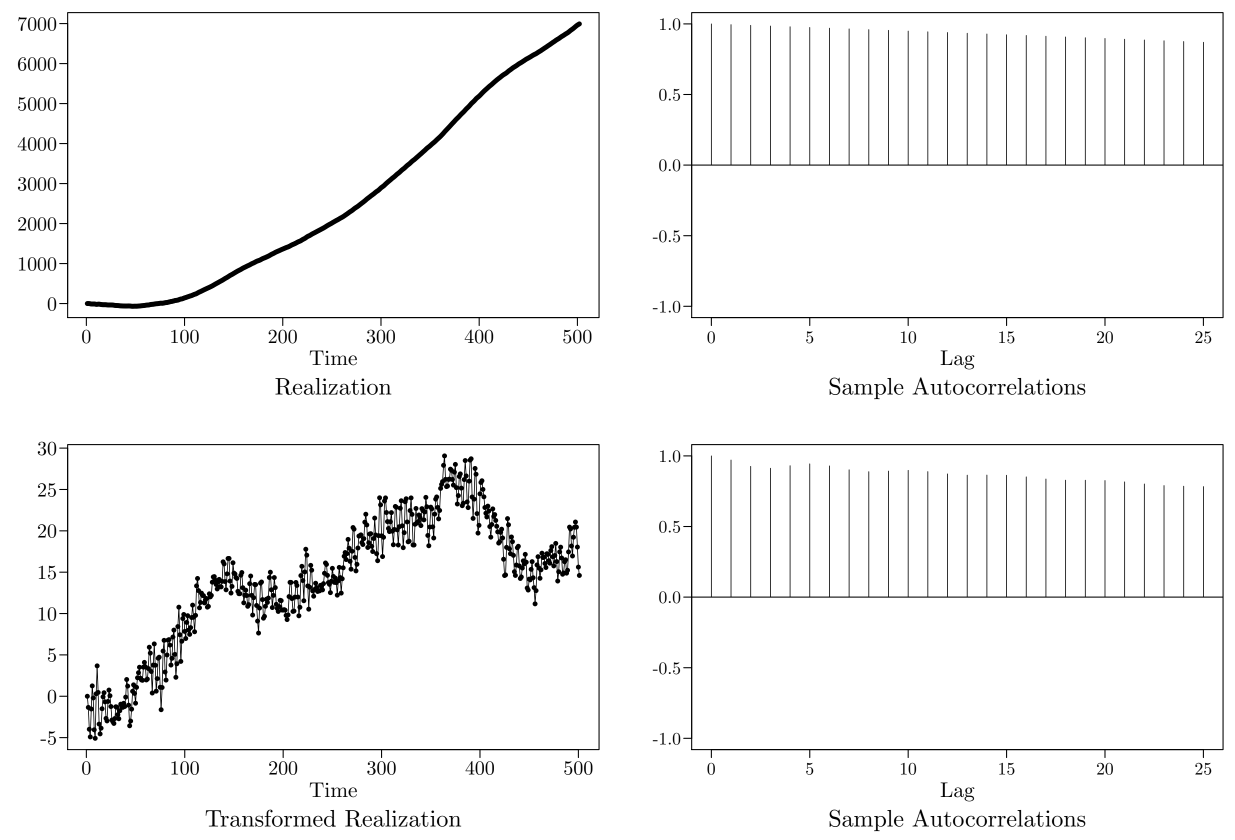
```r
library(tswge)
arimatr <- function(xs, d) {
    f <- artrans.wge(xs, phi.tr = 1)
    if (d == 1) {
        f
    } else {
        arimatr(f, d - 1)
    }
}

x = gen.arima.wge(500, phi = c(0.6, -0.8), d = 2, theta = 0.3, sn = 35, plot = F)
d <- artrans.wge(x, phi.tr = 1)
```

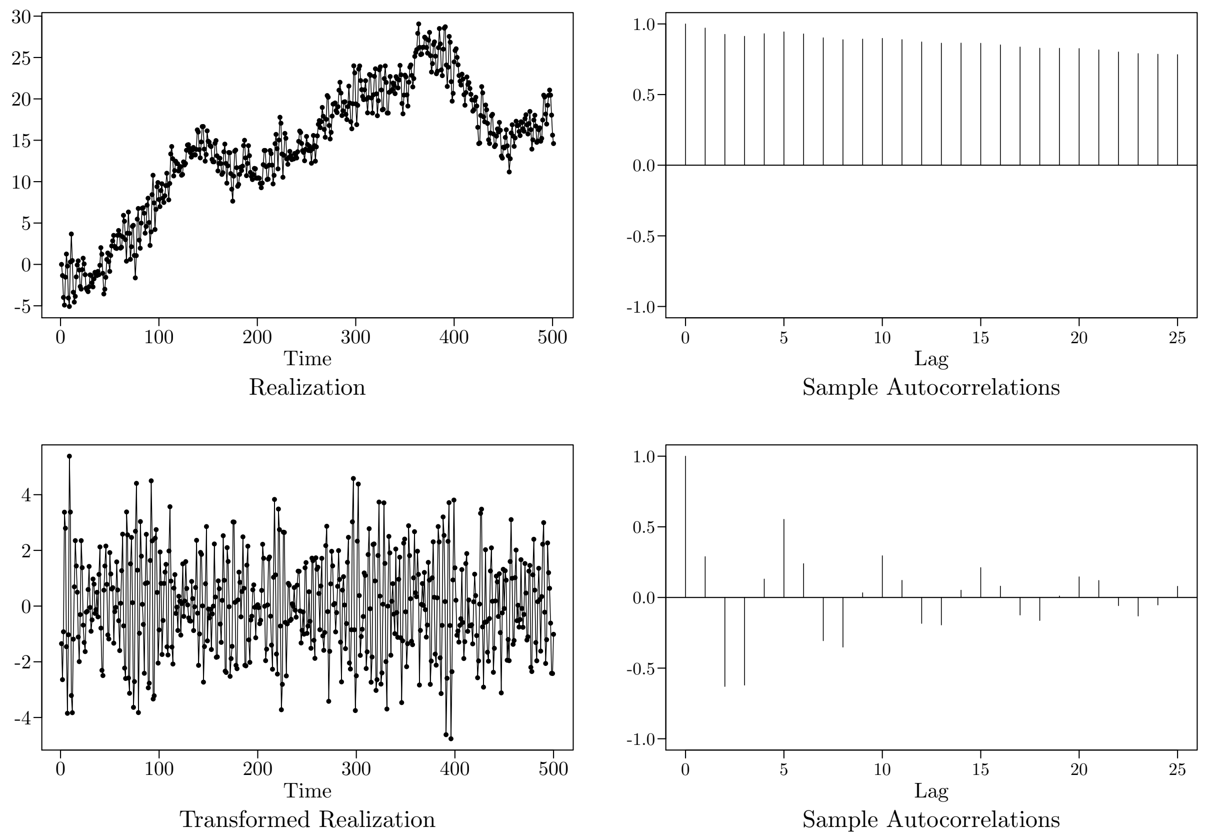


```r
d2 <- artrans.wge(d, phi.tr = 1)
```

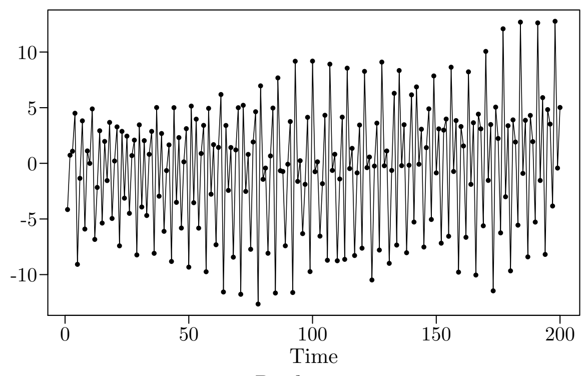


```r
d3 <- arimatr(x, 2)
```

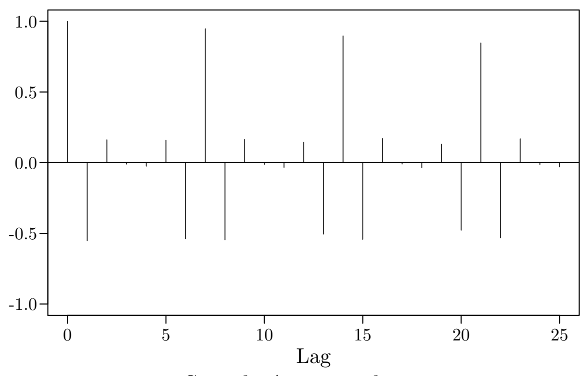




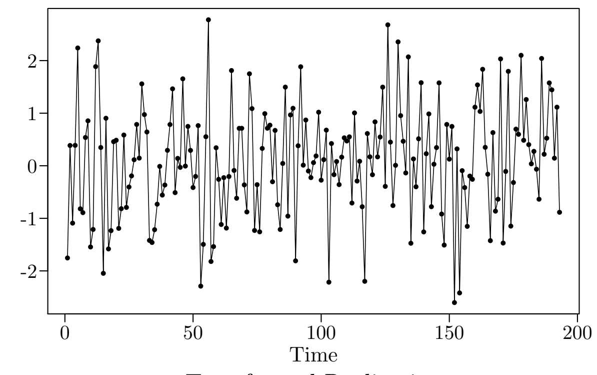
```r
arumatr <- function(xs, s) {
    artrans.wge(xs, phi.tr = c(rep(0, s - 1), 1))
}

x2 <- gen.aruma.wge(n = 200, s = 7, plot = F)
arumatr(x2, 7)
```



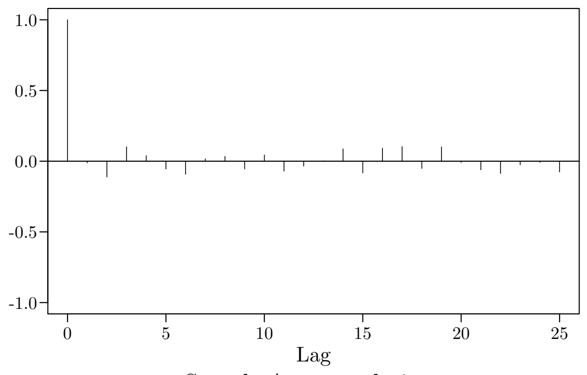
Realization



Sample Autocorrelations



Transformed Realization



Sample Autocorrelations

Again by transforming away the non stationary nonsense we end up with pure white noise in this case. We can check that with aic5.wge, it should be very confused

Chapter 21

Forecasting

Some helpers:

```r
tswgen <- function(n, sn = 0) {
    sig <- function(...) {
        gen.sigplusnoise.wge(n = n, ..., sn = sn)
    }
    ari <- function(..., dif) {
        gen.arima.wge(n = n, ..., sn = sn)
    }
    aru <- function(...) {
        gen.aruma.wge(n = n, ..., sn = sn)
    }

    arma <- function(...) {
        gen.arma.wge(n = n, ..., sn = sn)
    }
    list(sig = sig, ari = ari, aru = aru, arma = arma)
}
```

21.1 Forecasts from Signal-Plus-Noise

$$X_t = s_t + Z_t$$

Where:

- s_t is a deterministic signal
- Z_t is a zero-mean, stationary process

Examples:

- $s_t = a + bt$ is a linear signal
- $s_t = a + bt + ct^2$ is a quadratic signal
- $s_t = \cos(2\pi f t + C)$

Chapter 22

Forecasting setting

Forecast predicts future behavior given finite reaizations. ARMA forecasting has gotten really populat lately.

- Curve fitting
    - Has the underlying assumption that future behavior follows a deterministic path
- Time series based forecasting (ARMA et al)
    - Underlying assumption that the future is only guided by its correlation to the past.

22.1 Box-Jenkins approach

We use this

Chapter 23

Strategy + Notation

We normally have:

$$X_1, X_2, ..., X_{t_0}$$

Our goal is to forecast a future value, say $X_{t_{0+4}}$

Typicall $t_0 = n$ ie it is the last value of the realization. However, sometimes to evaluate our forecast, we compare it to the last few values of the realization.

23.1 Notation

$\widehat{X_{t_0}}(\ell)$ is the forecast of $X_{t_{0+\ell}}$ given data up to time t_0

- t_0 is called the **forecast origin**
- ℓ is called the lead time, the number of steps ahead which we want to forecast

23.2 some math

$$X_t - \phi_1 X_{t-1} = (1 - \phi_1)\mu + a_t$$

- assume we know ϕ_1
- we don-t know X11, we dont know mu, and we dont know a12

Iterative forecasting

We assume a in the future is 0, then we have

```
ar1forcastgen <- function(phi, mu) {
    fun <- function(xprev, l = 1) {
        if (l == 1)
            return(phi * xprev + mu * (1 - phi)) else return(phi * fun(xprev, l - 1) + mu * (1 - phi))
    }
    fun
}
```

```
ex74 <- ar1forcastgen(phi = 0.8, mu = 24.17)
times <- c(1, 2, 3, 4, 5)
lapply(times, ex74, xprev = 22.93) %>% as.data.frame
```

```
##   X23.178 X23.3764 X23.53512 X23.662096 X23.7636768
## 1  23.178  23.3764  23.53512    23.6621    23.76368
```

```
ex74(22.93, l = 2)
```

```
## [1] 23.3764
```

Chapter 24

Discussion of Ar1 forcasts

AR1 forcasts damp to the sample mean.

24.1 AR(p) forecasts:

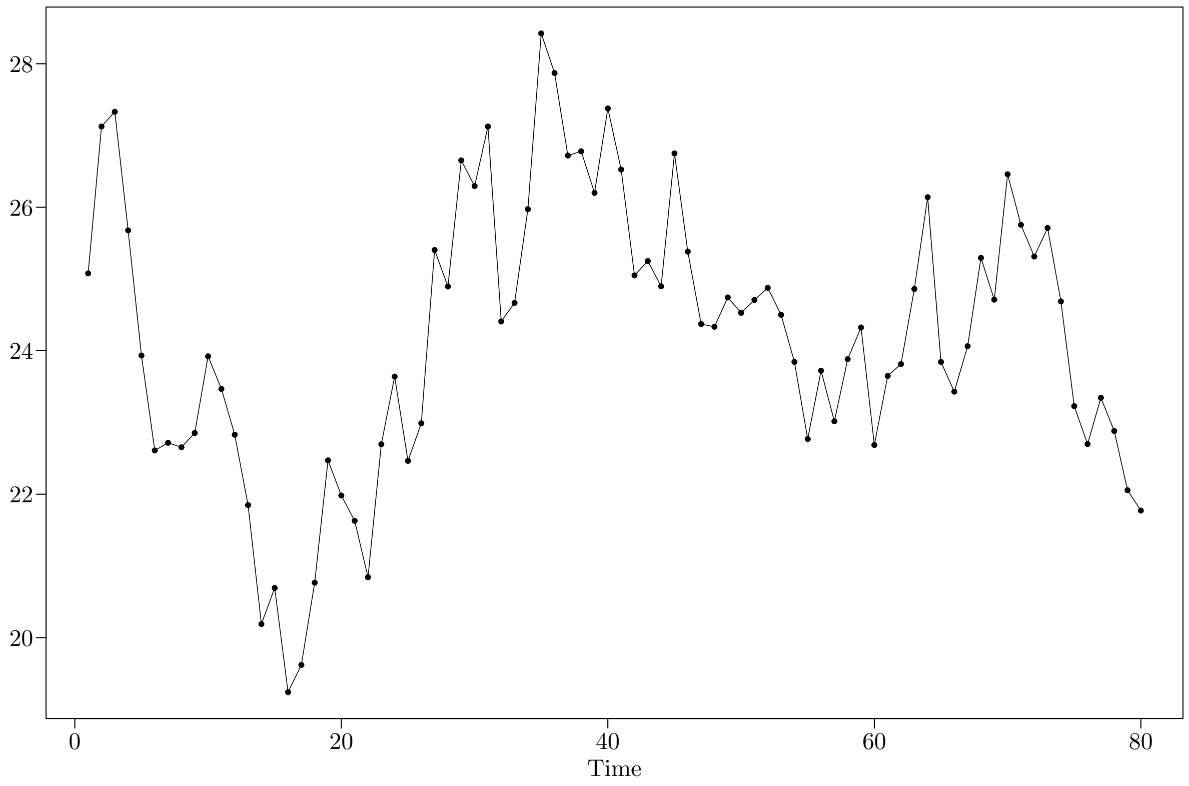
```r
arpgen <- function(vec, l, mu, phi) {
    v <- vec[length(vec):(length(vec) - (length(phi) - 1))]
    f <- sum(phi * (v)) + mu * (1 - sum((phi)))
    if (l == 1)
        return(f)
    return(arpgen(append(vec, f), l - 1, mu, phi))

}

arp <- function(vec, l, mu, phi) {
    as.numeric(lapply(1:l, arpgen, phi = phi, vec = vec, mu = mu))
}
x <- c(27.7, 23.4, 21.2, 21.1, 22.7)
p <- c(1.6, -0.8)
avg <- 29.4
arp(phi = p, vec = x, l = 5, mu = avg)
```

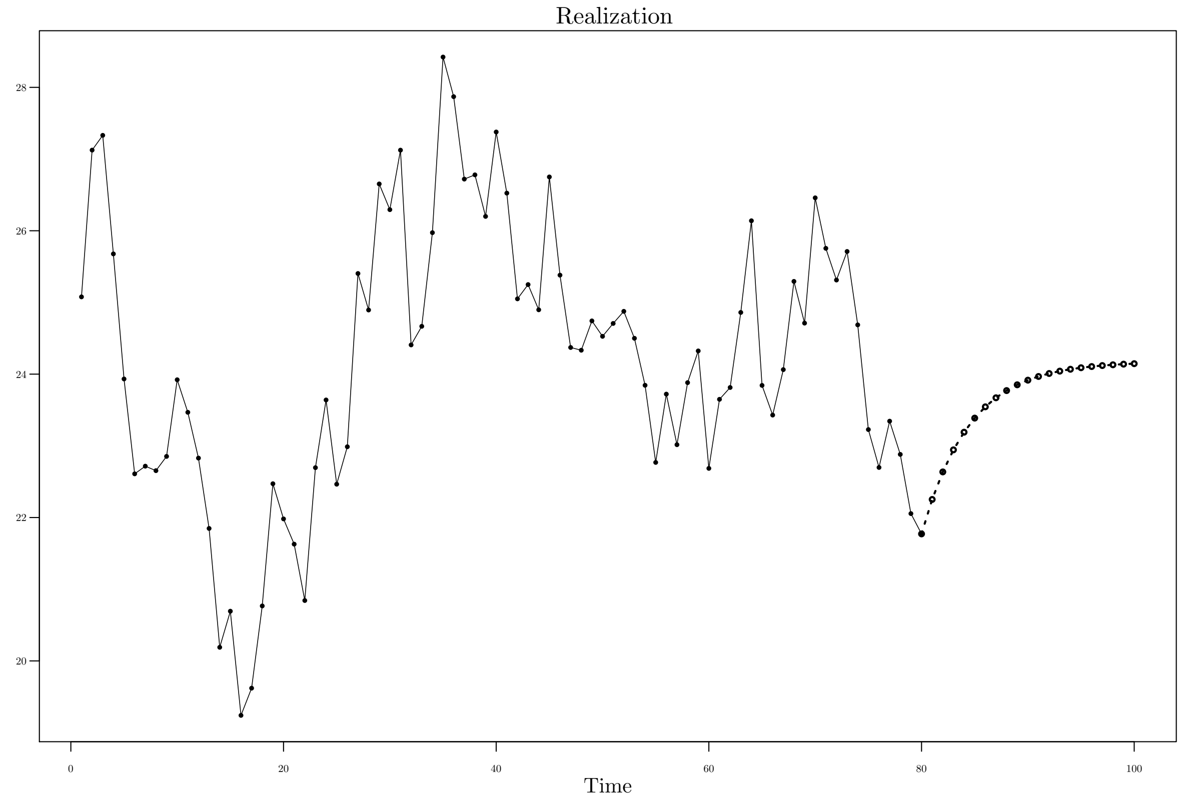
```
## [1] 25.32000 28.23200 30.79520 32.56672 33.35059
```

24.1.1 With tswge

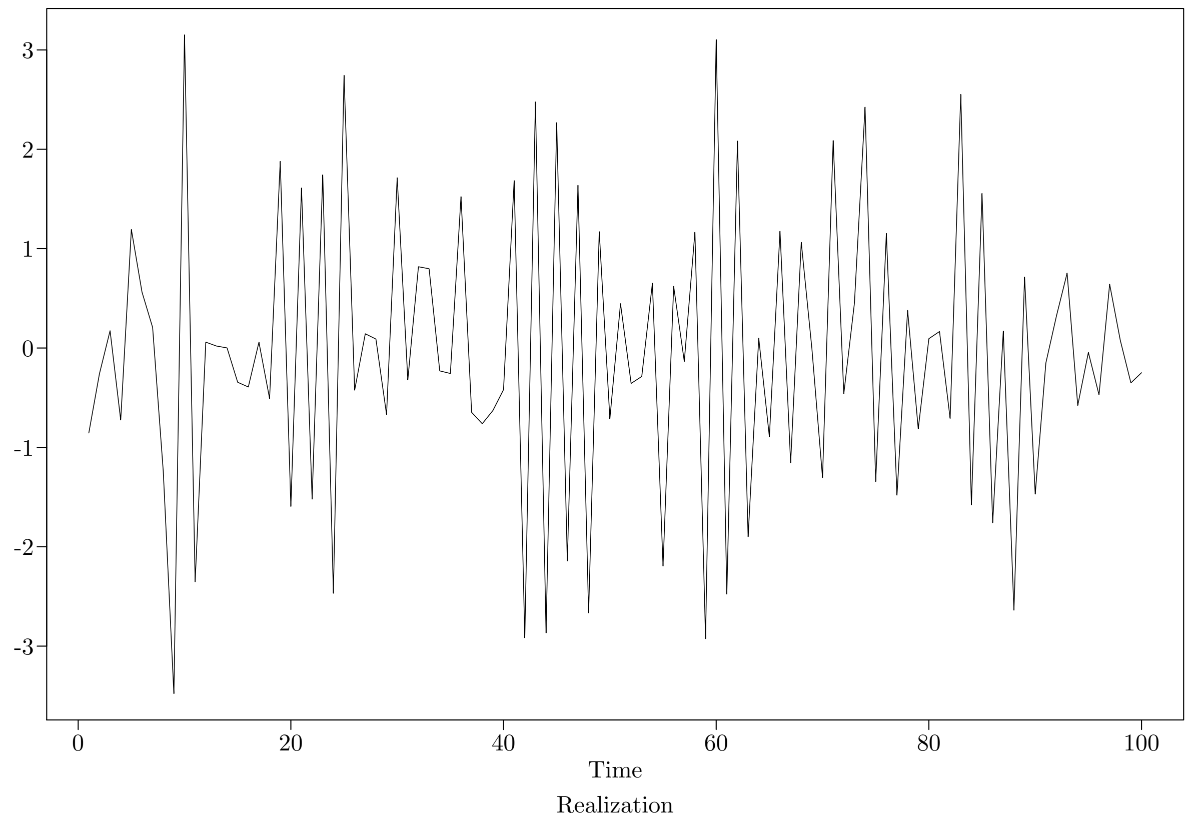
24.1.1.1 positive phi, AR(1)

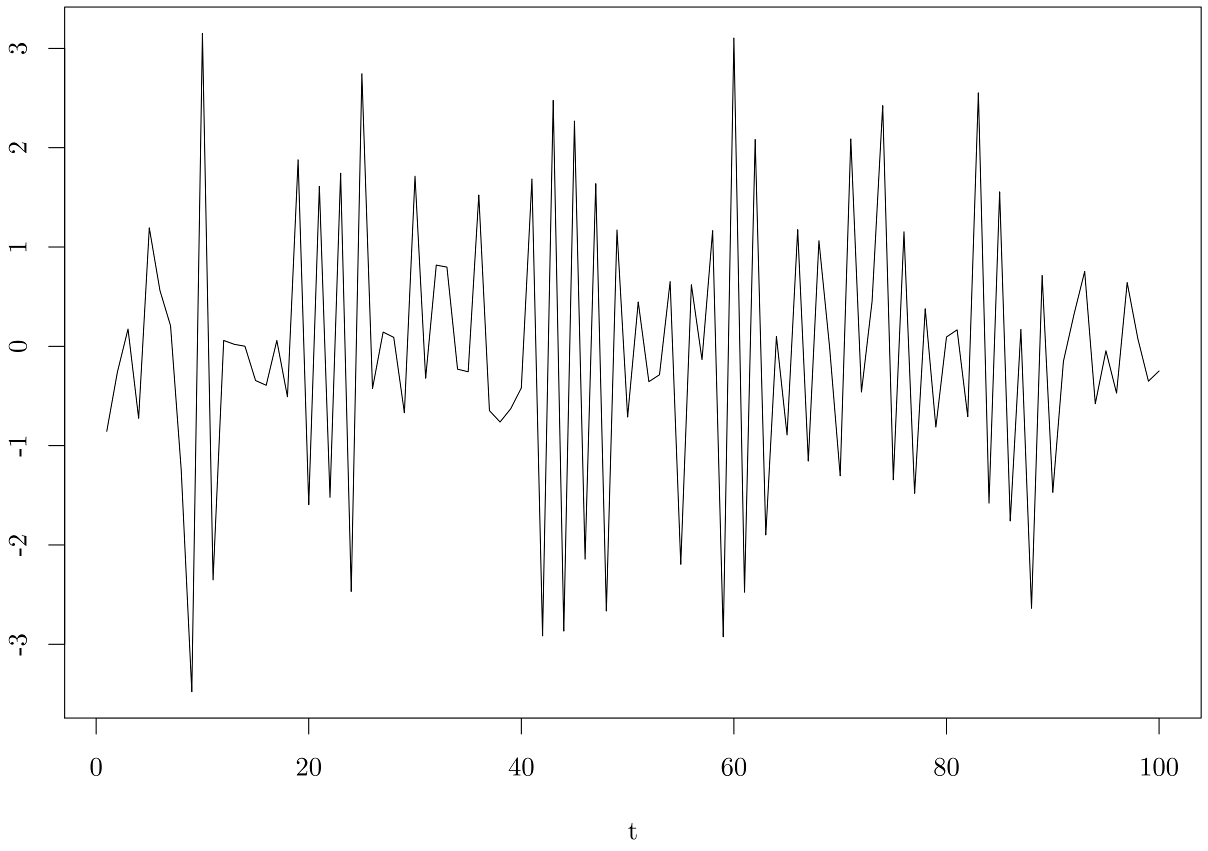
```r
data(fig6.1nf)
plotts.wge(fig6.1nf)
fore.arma.wge(fig6.1nf, phi = 0.8, n.ahead = 20, limits = F)
```


28
26
24
22
20
0
20
40
60
80
Time

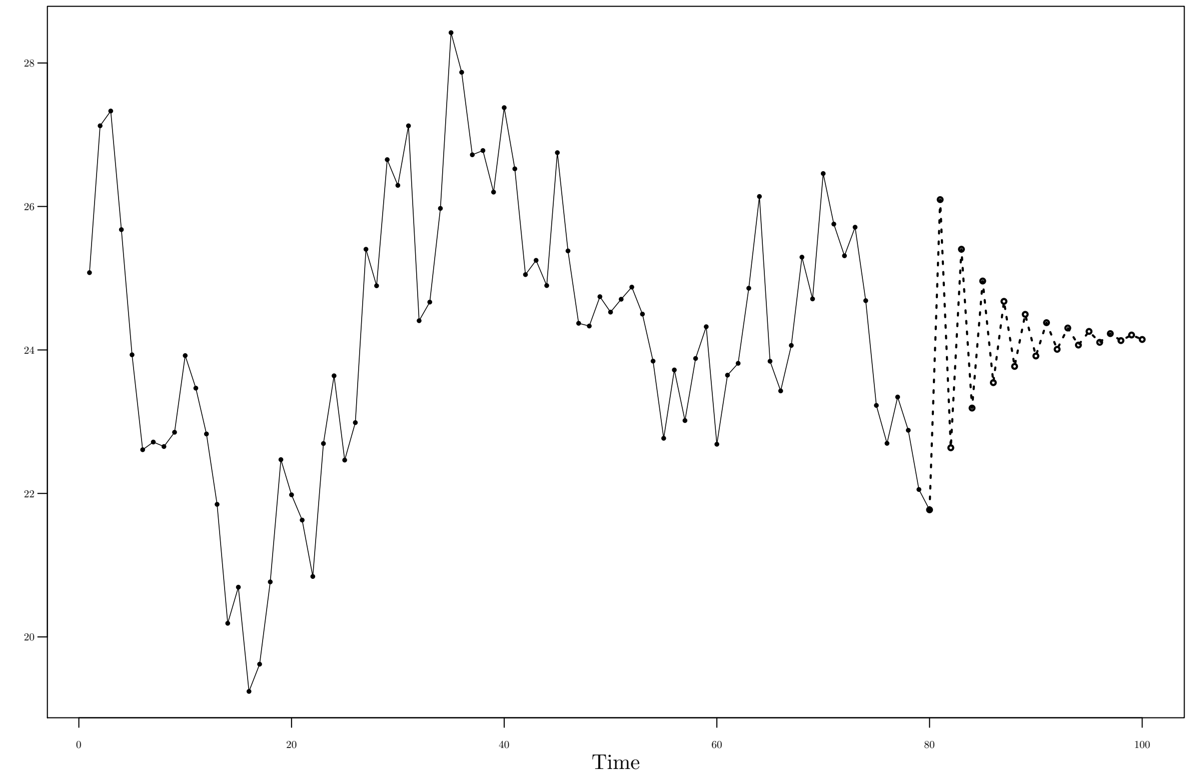

Realization
28
26
24
22
20
0
20
40
60
80
100
Time

24.1.1.2 Negative phi, AR(1)



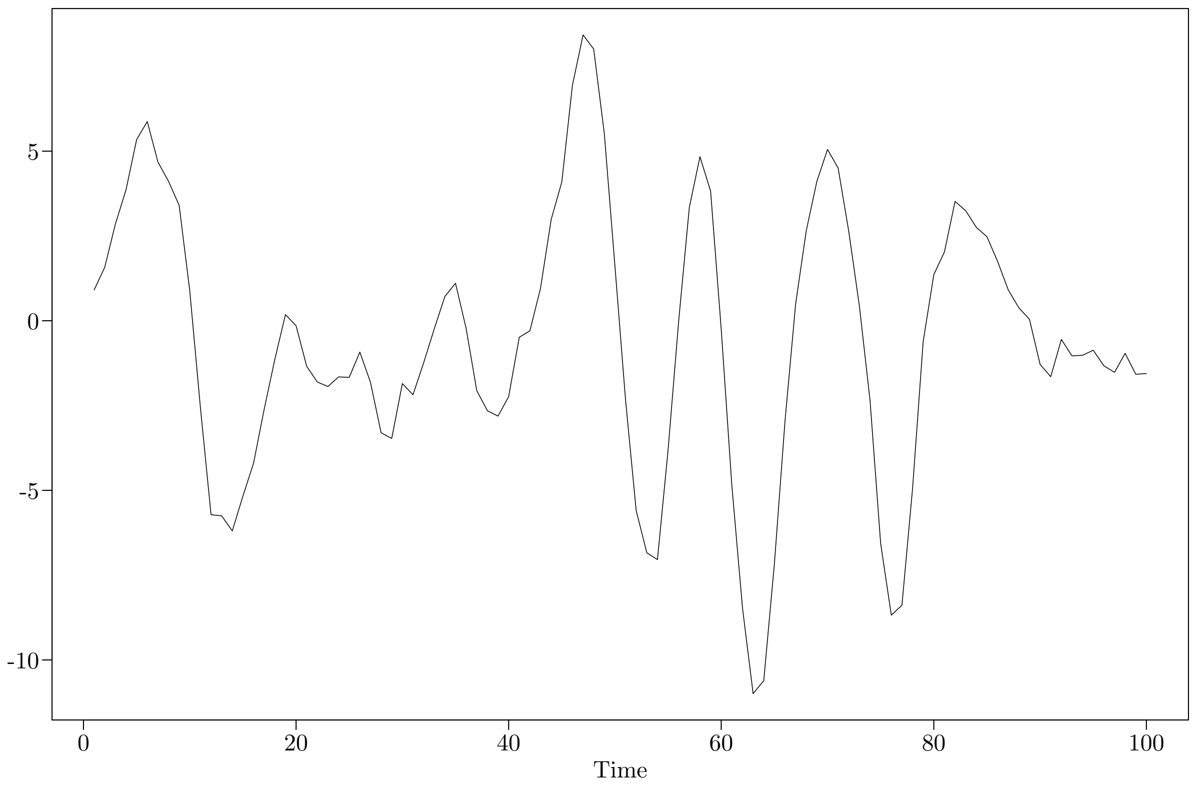


```
fore.arma.wge(fig6.1nf, phi = -0.8, n.ahead = 20, limits = F)
```

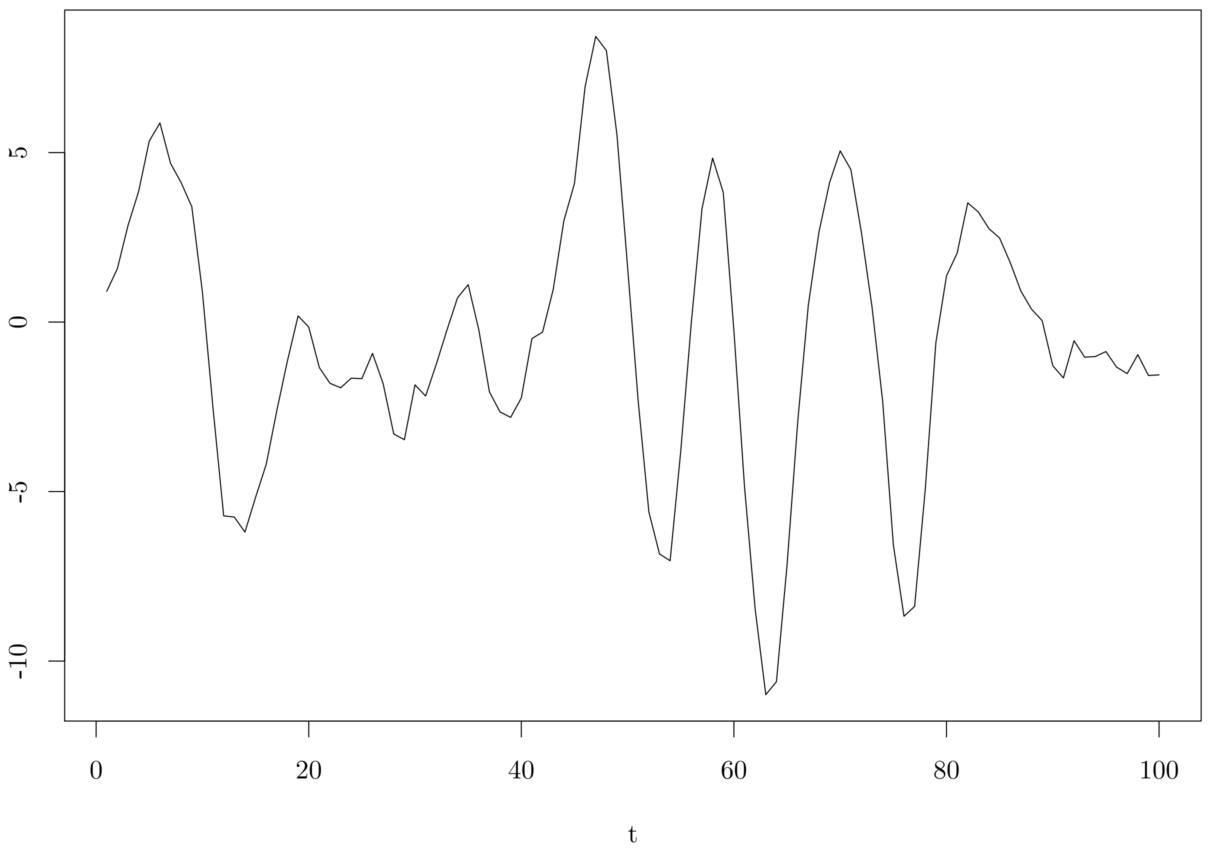


It has the same behavior as the AR(functions). Sweet.

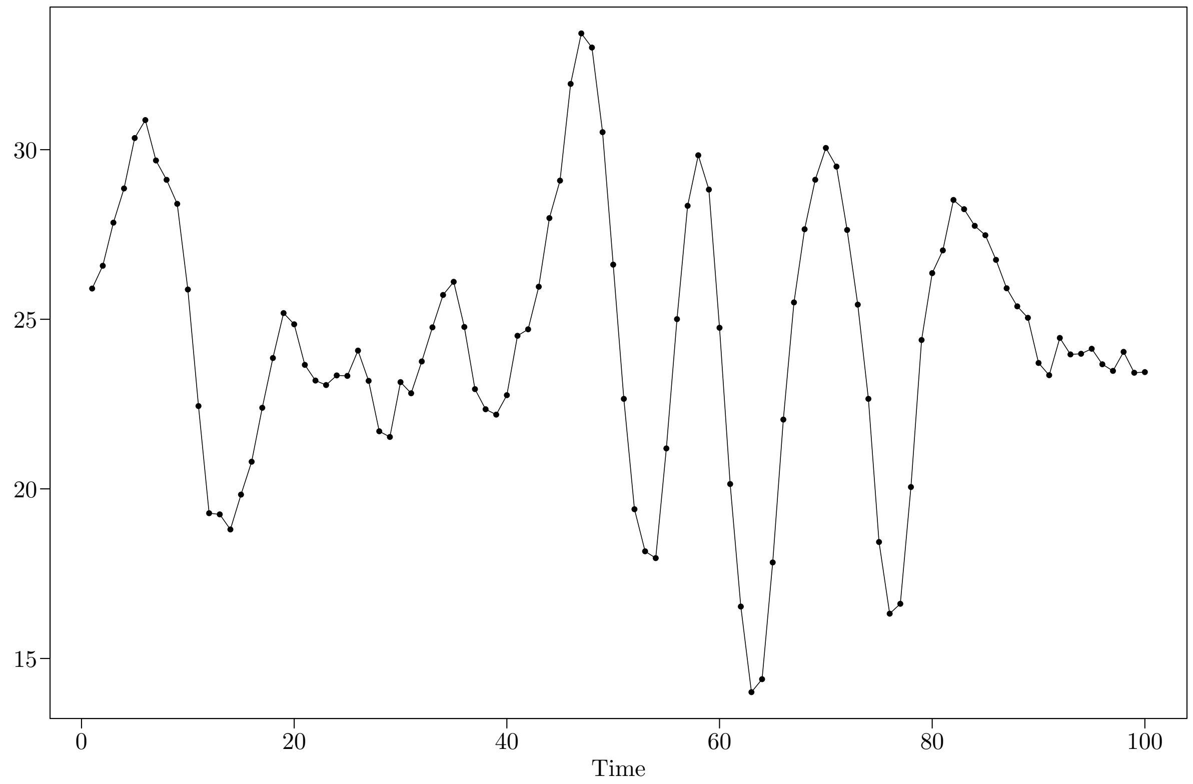
24.1.1.3 AR(2)



Realization

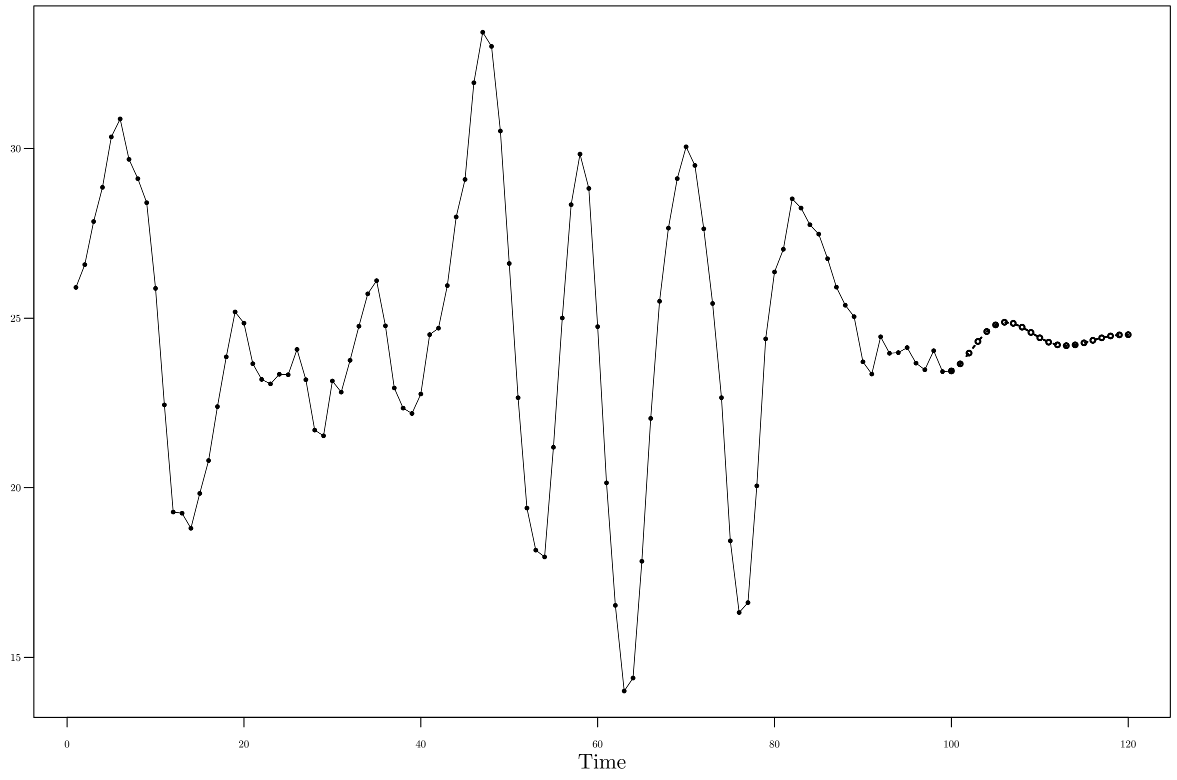


```
X2 <- X2 + 25
plotts.wge(X2)
```



Realization

```r
fore.arma.wge(X2, phi = c(1.6, -0.8), n.ahead = 20, limits = F)
```



Chapter 25

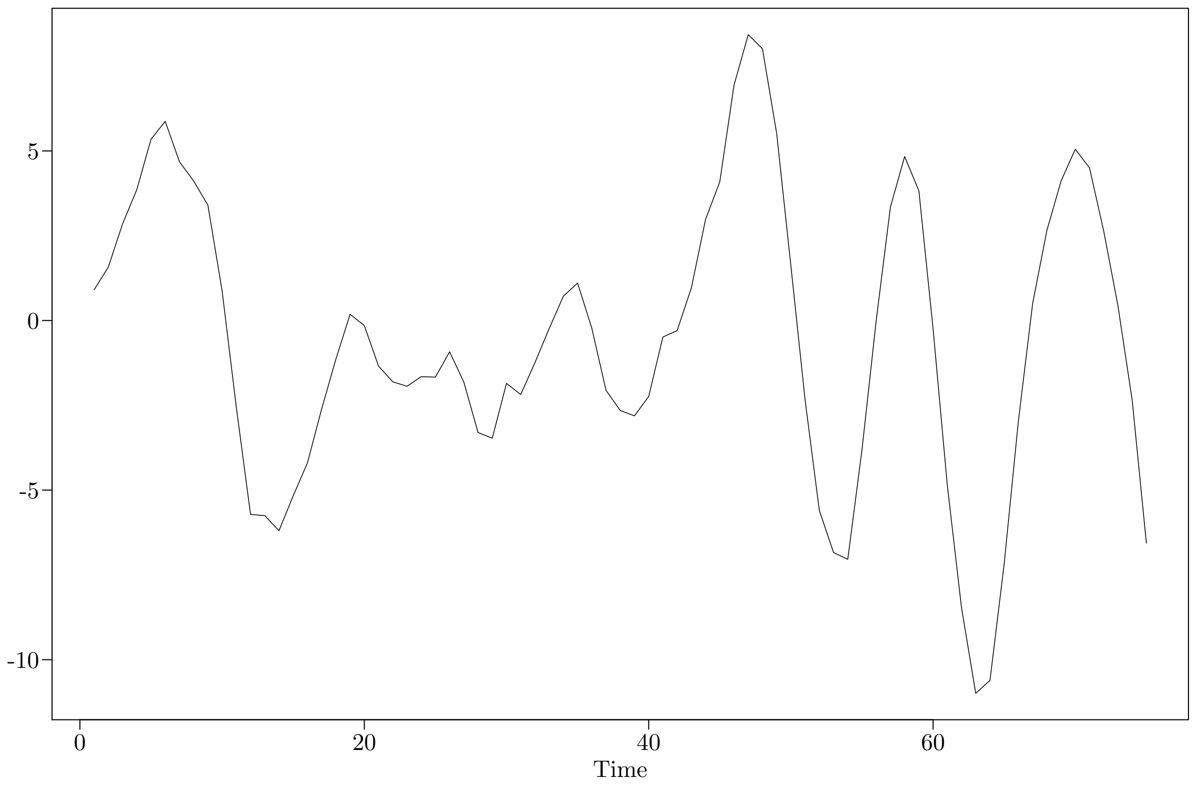
Eventual Forecast Function:

As ℓ gets large, it is most reasonable to simply forecast the sample mean, because of how they converge

Chapter 26

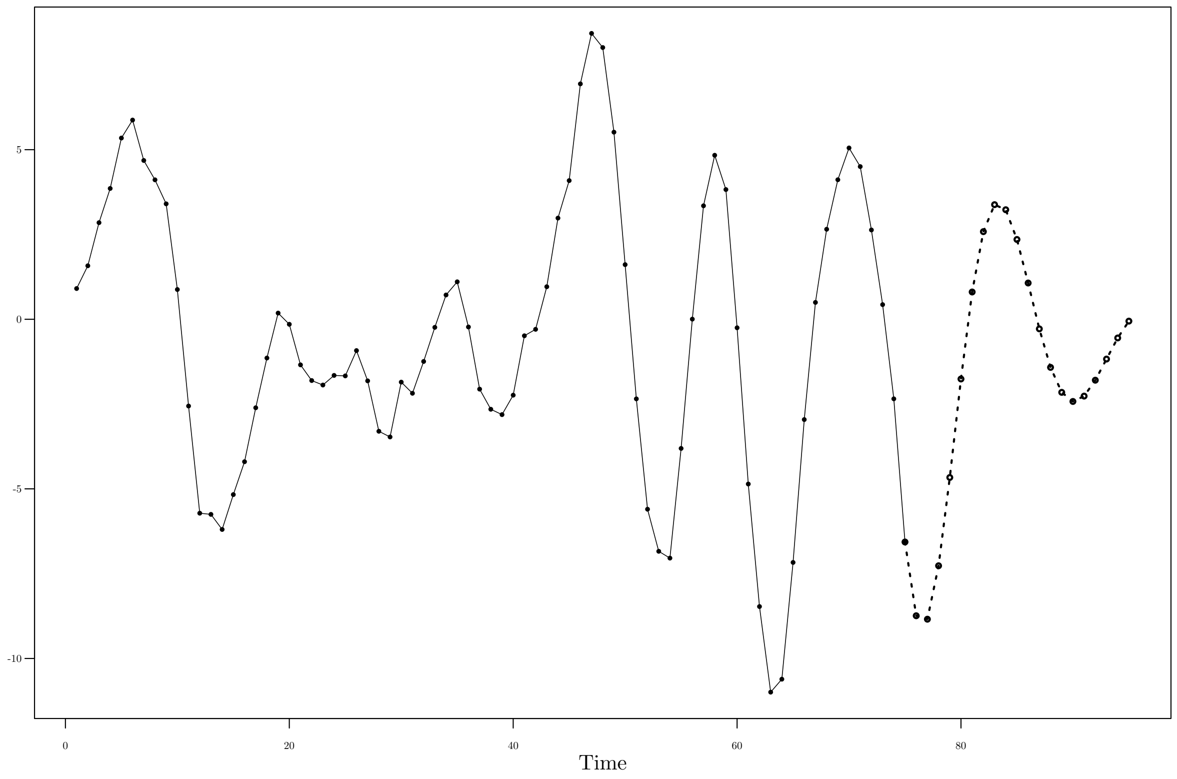
ARMA(p,q)

```r
ts7524 <- tswgen(75, sn = 24)
p <- c(1.6, -0.8)
thet <- (-0.9)
x1 <- ts7524$arma(phi = p)
```

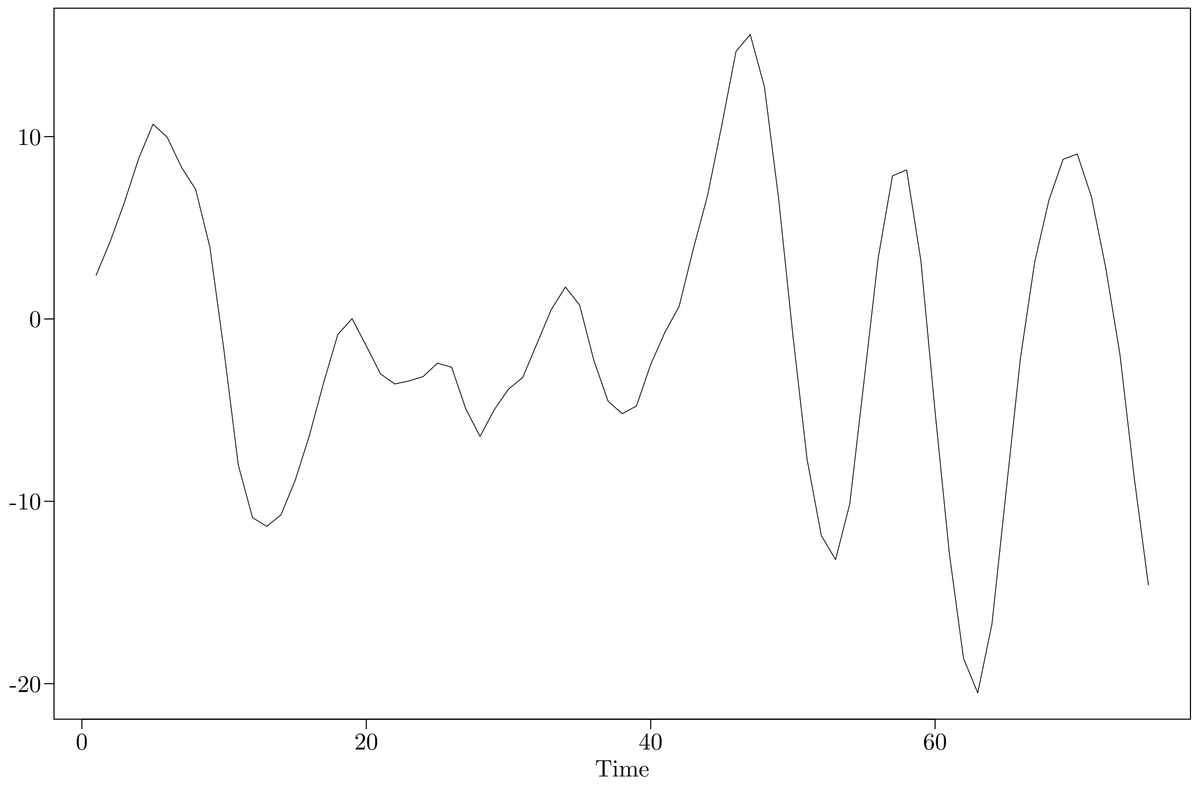


Realization

```r
fore.arma.wge(x1, phi = p, n.ahead = 20, limits = F)
```

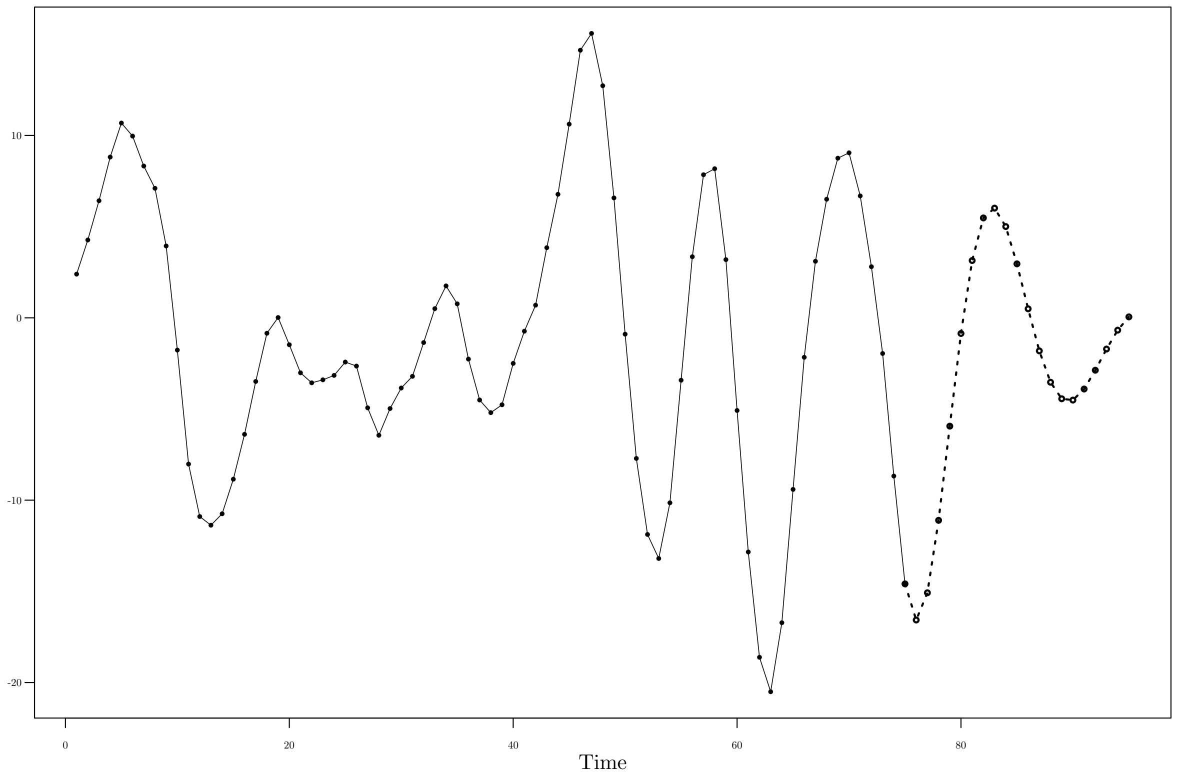


```
x2 <- ts7524$arma(phi = p, theta = thet)
```



Realization

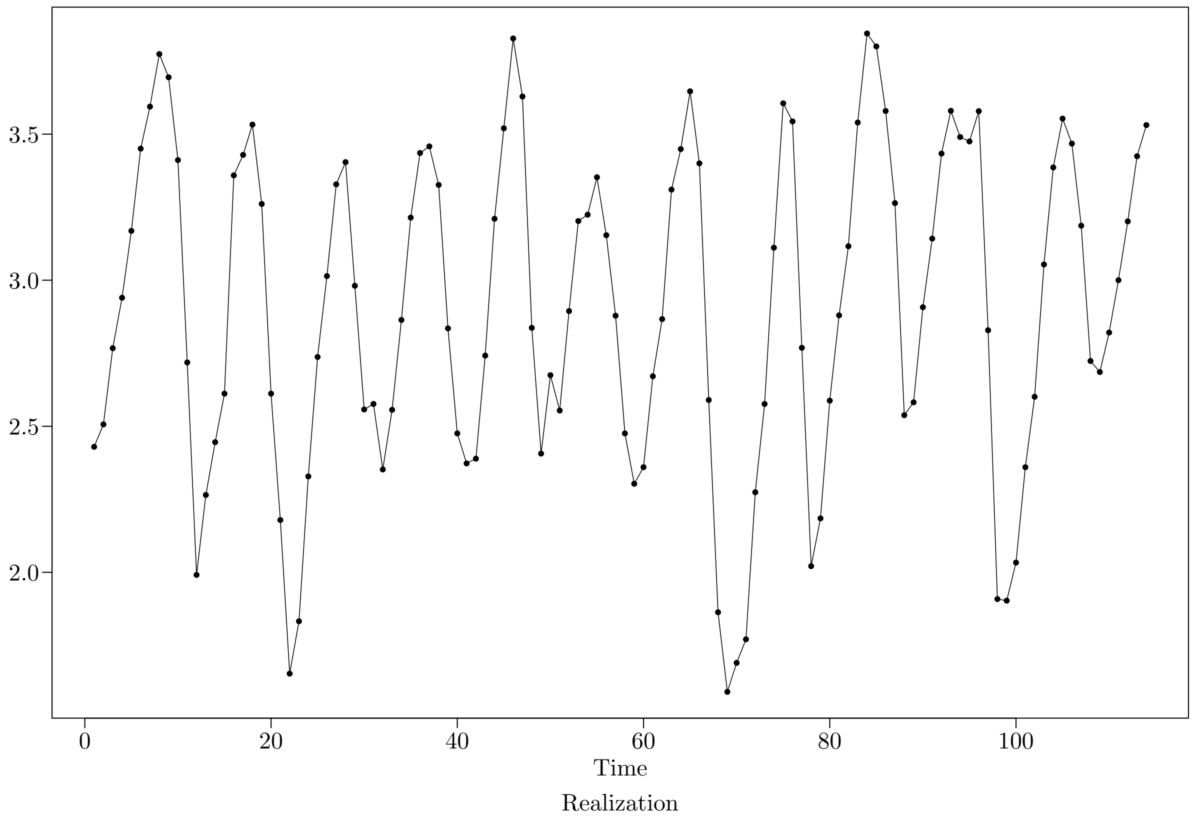
```r
fore.arma.wge(x2, phi = p, theta = thet, n.ahead = 20, limits = F)
```



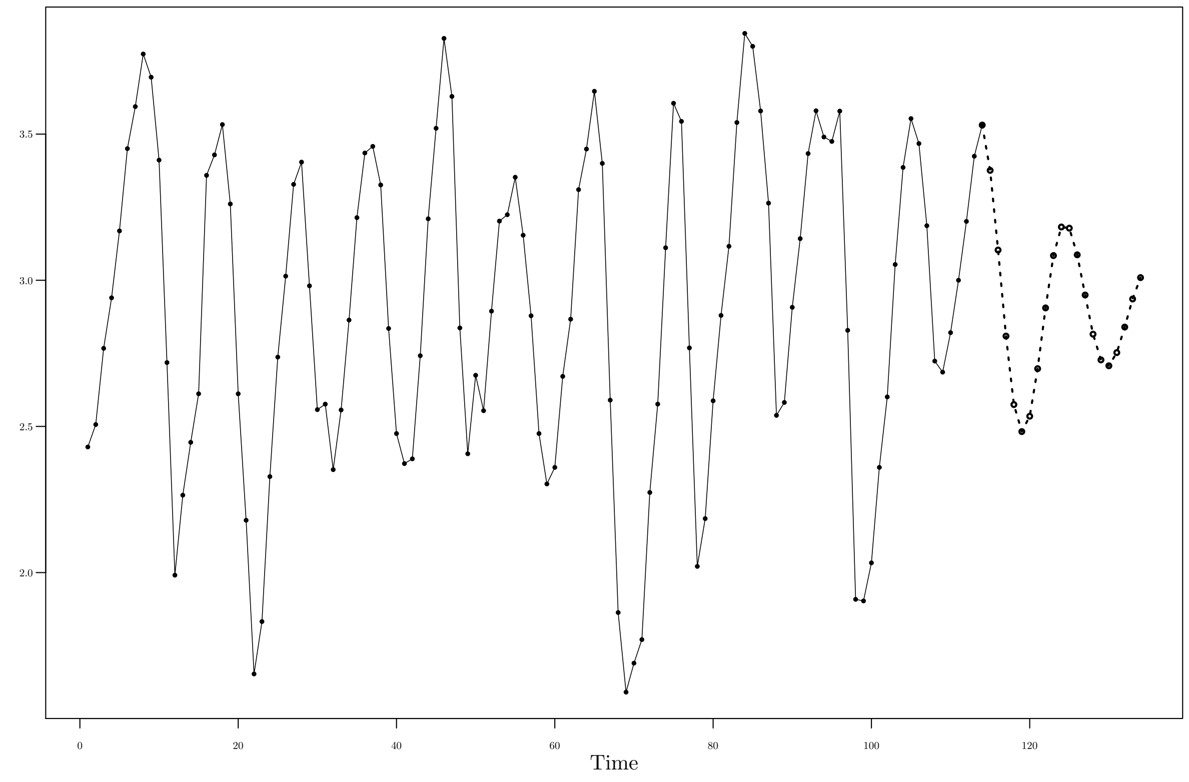
Chapter 27

Example

```
data(llynx)
plotts.wge(llynx)
```



```
fore.arma.wge(llynx, phi = c(0.7, 0.1, -0.2, -0.3), theta = -0.6, n.ahead = 20,
    limits = F)
```


3.5
3.0
2.5
2.0
0
20
40
60
80
100
120
Time

Chapter 28

psi weights

```r
psi.weights.wge(phi = c(0.4, -0.6, 0.8), lag.max = 5)
```

```
## [1]  0.40000 -0.44000  0.38400  0.73760 -0.28736
```

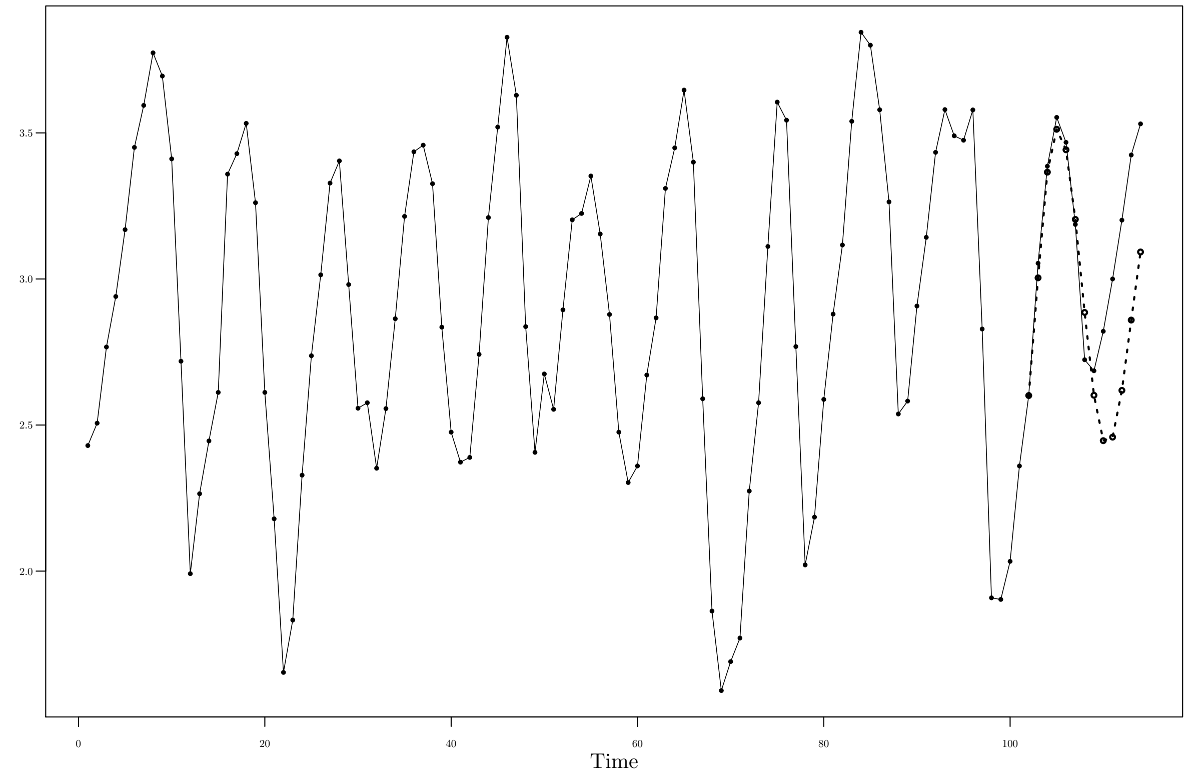
```r
psiweights(phi = c(0.4, -0.6, 0.8), l = 6)
```

```
## [1]  1.00000  0.40000 -0.44000  0.38400  0.73760 -0.28736
```

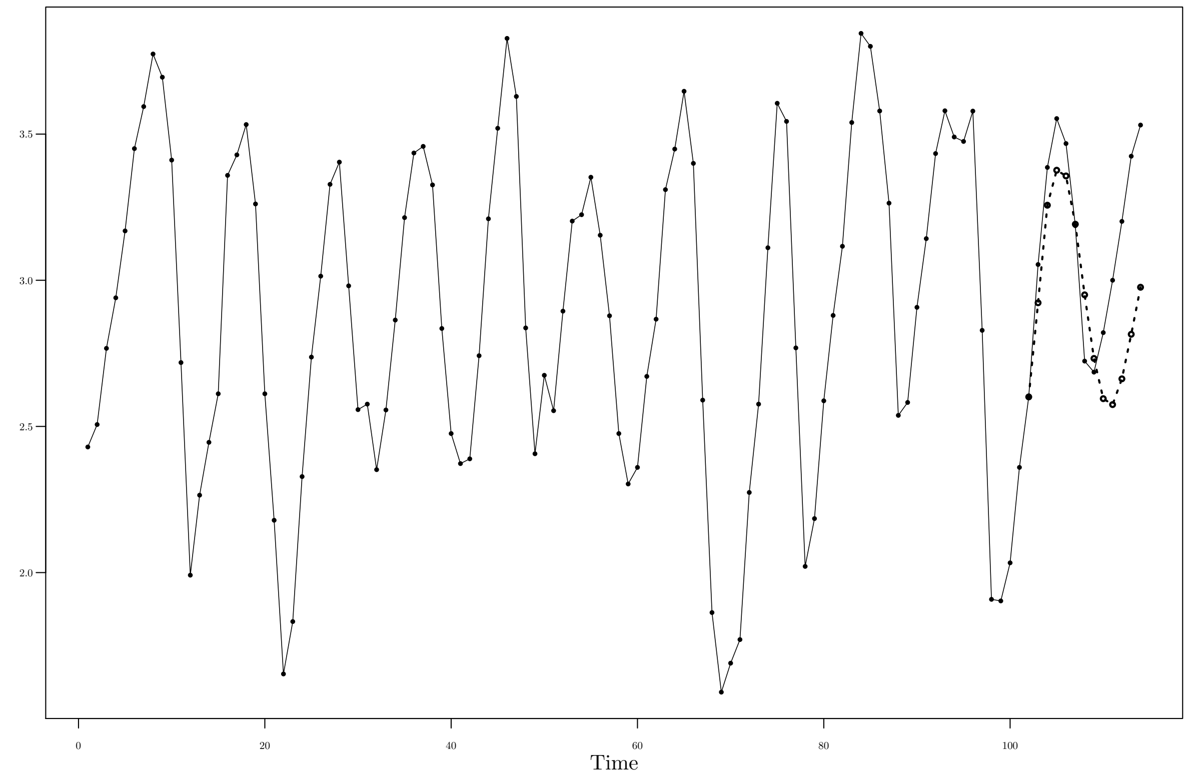
Chapter 29

ASE

```r
tswase <- function(vec, ..., ahead) {
    f <- fore.arma.wge(vec, ..., n.ahead = ahead)
    ASE <- mean((f$f - vec[(length(vec) - (ahead - 1)):(length(vec))])^2)
    res <- list(fore = f, ASE = ASE)
    return(res)
}
res1 <- tswase(vec = llynx, phi = c(1.3, -0.7, 0.1, -0.2), ahead = 12, lastn = T,
    limits = F)
```



```r
library(pander)
res2 <- tswase(vec = llynx, phi = c(0.7, 0.1, -0.2, -0.3), theta = -0.6, ahead = 12,
    lastn = T, limits = F)
```



```r
pander(res1)
pander(res2)
```

- **fore**:
    - **f**: *3.004*, *3.366*, *3.512*, *3.442*, *3.204*, *2.885*, *2.602*, *2.446*, *2.459*, *2.619*, *2.859* and *3.092*
    - **ll**: *2.566*, *2.647*, *2.673*, *2.577*, *2.336*, *1.968*, *1.597*, *1.375*, *1.366*, *1.526*, *1.749* and *1.941*
    - **ul**: *3.442*, *4.085*, *4.352*, *4.308*, *4.072*, *3.803*, *3.607*, *3.518*, *3.551*, *3.711*, *3.97* and *4.244*
    - **resid**: *0*, *0*, *0*, *0*, *0.06725*, *0.1617*, *0.1344*, *0.3362*, *0.1412*, *0.1288*, *-0.2401*, *-0.2215*, *0.5252*, *-0.1463*, *-0.08944*, *0.3955*, *-0.3528*, *0.2023*, *-0.1965*, *-0.2777*, *-0.05285*, *-0.4227*, *0.1478*, *-0.04402*, *-0.1883*, *-0.2188*, *0.007712*, *-0.07248*, *-0.3206*, *-0.1166*, *0.2116*, *-0.2759*, *0.1905*, *-0.01071*, *0.1089*, *0.02447*, *0.01511*, *0.03496*, *-0.2208*, *0.007828*, *0.04624*, *-0.03276*, *0.1647*, *0.1242*, *0.04974*, *0.2505*, *-0.1074*, *-0.363*, *0.1281*, *0.483*, *-0.2486*, *0.3216*, *-0.01041*, *-0.08519*, *0.1721*, *-0.1402*, *-0.008932*, *-0.2007*, *0.00302*, *-0.01021*, *0.09193*, *-0.1408*, *0.2258*, *-0.09438*, *0.2757*, *-0.1359*, *-0.412*, *-0.2504*, *-0.08081*, *-0.1047*, *-0.4328*, *-0.08309*, *-0.4431*, *0.06305*, *0.03913*, *-0.2203*, *-0.5617*, *-0.2877*, *0.4101*, *0.1424*, *-0.05529*, *-0.08196*, *0.2306*, *0.2021*, *0.0926*, *0.1474*, *0.1429*, *-0.2628*, *0.5179*, *0.2648*, *0.1174*, *0.1811*, *0.08981*, *0.05521*, *0.277*, *0.3812*, *-0.476*, *-0.365*, *0.2874*, *-0.1236*, *-0.0284*, *-0.3039*, *0.04991*, *-0.04449*, *0.04913*, *-0.01834*, *-0.01367*, *-0.1218*, *0.288*, *0.1588*, *0.1261*, *0.1003*, *0.1658* and *0.1326*
    - **wnv**: *0.05002*
    - **se**: *0.2237*, *0.3668*, *0.4285*, *0.4415*, *0.4427*, *0.4681*, *0.5127*, *0.5468*, *0.5576*, *0.5576*, *0.5664* and *0.5877*
    - **psi**: *1.3*, *0.99*, *0.477*, *-0.1429*, *-0.6807*, *-0.9351*, *-0.8489*, *-0.4885*, *0.00185*, *0.4465*, *0.7* and *0.6954*

- **ASE**: *0.1103*

- **fore**:
    - **f**: *2.923*, *3.257*, *3.376*, *3.357*, *3.192*, *2.95*, *2.733*, *2.595*, *2.575*, *2.663*, *2.815* and *2.976*
    - **ll**: *2.489*, *2.544*, *2.54*, *2.476*, *2.311*, *2.039*, *1.756*, *1.557*, *1.511*, *1.598*, *1.744* and *1.882*
    - **ul**: *3.357*, *3.969*, *4.213*, *4.238*, *4.073*, *3.862*, *3.709*, *3.633*, *3.638*, *3.728*, *3.886* and *4.07*
    - **resid**: *0*, *0*, *0*, *0*, *0.03173*, *0.192*, *0.1324*, *0.3168*, *0.1114*, *0.102*, *-0.2995*, *-0.2346*, *0.4979*, *-0.1032*, *-0.08355*, *0.354*, *-0.2599*, *0.1759*, *-0.2375*, *-0.2209*, *-0.1402*, *-0.3697*, *0.1473*, *-0.02132*, *-0.1114*, *-0.2381*, *0.07028*, *-0.05577*, *-0.3099*, *-0.1463*, *0.2225*, *-0.2556*, *0.1787*, *-0.018*, *0.1755*, *-0.022*, *0.05216*, *0.0001507*, *-0.2203*, *-0.01963*, *0.03829*,

-0.01013, *0.1514*, *0.1459*, *0.06813*, *0.2341*, *-0.1108*, *-0.3849*, *0.07767*, *0.5014*, *-0.2366*, *0.2809*, *-0.02308*, *-0.01238*, *0.09503*, *-0.09576*, *-0.03411*, *-0.229*, *0.02378*, *-0.02486*, *0.13*, *-0.1462*, *0.2542*, *-0.09775*, *0.3021*, *-0.1893*, *-0.3908*, *-0.3238*, *-0.03664*, *-0.08252*, *-0.4048*, *-0.04694*, *-0.3817*, *0.1381*, *0.04056*, *-0.1509*, *-0.6192*, *-0.2778*, *0.4174*, *0.1901*, *-0.06197*, *-0.1104*, *0.2769*, *0.2086*, *0.0845*, *0.09404*, *0.1203*, *-0.2961*, *0.4802*, *0.252*, *0.152*, *0.09705*, *0.1274*, *0.03258*, *0.2512*, *0.3599*, *-0.5004*, *-0.4197*, *0.2618*, *-0.03989*, *-0.0322*, *-0.3143*, *0.1308*, *-0.04089*, *0.09771*, *-0.05833*, *-0.0001889*, *-0.16*, *0.2836*, *0.1434*, *0.1389*, *0.05759*, *0.1862* and *0.1157*
  - **wnv**: *0.0491*
  - **se**: *0.2216*, *0.3634*, *0.4268*, *0.4495*, *0.4495*, *0.465*, *0.4983*, *0.5295*, *0.5426*, *0.5434*, *0.5464* and *0.5582*
  - **psi**: *1.3*, *1.01*, *0.637*, *-0.0131*, *-0.5375*, *-0.8079*, *-0.8078*, *-0.5348*, *-0.1323*, *0.2578*, *0.5165* and *0.5743*

- **ASE**: *0.111*