MATH 8090: Stationary Processes: Properties, Mean, and Covariance Functions

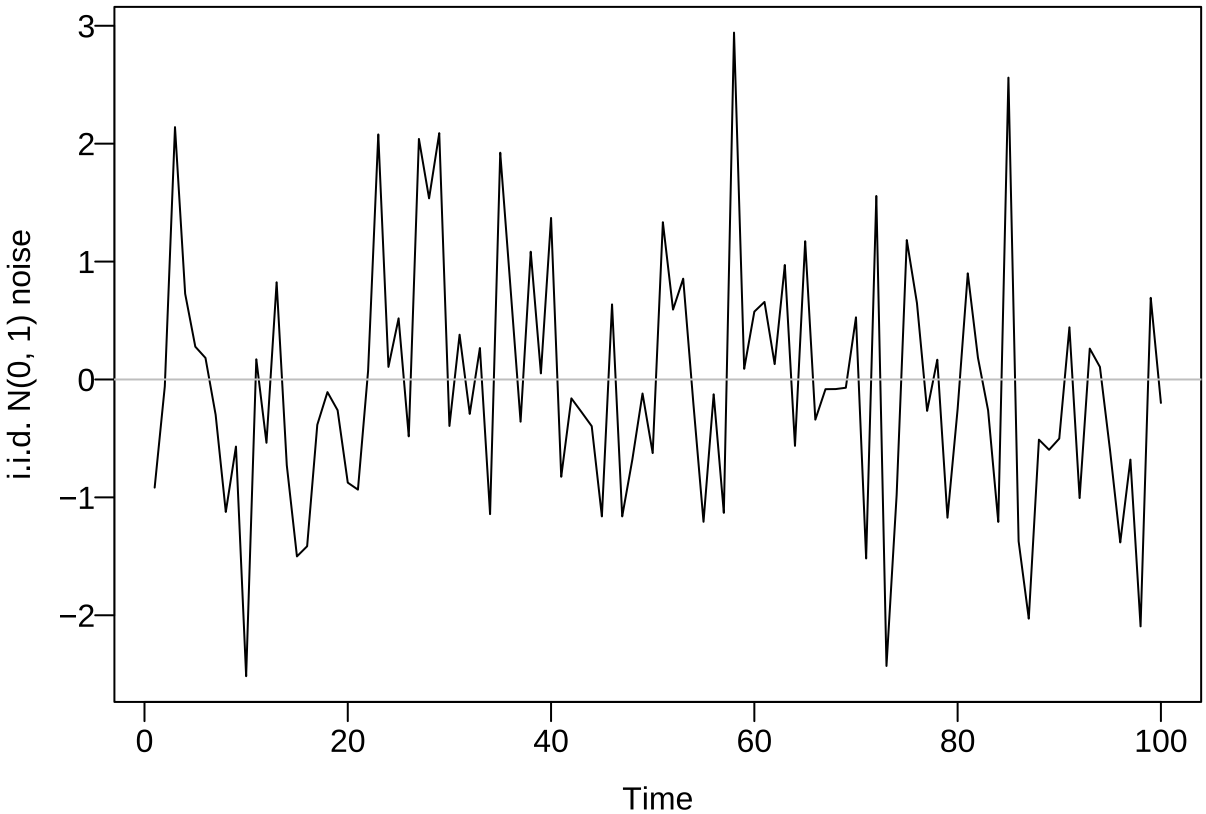
Whitney Huang, Clemson University

9/2-9/4/2025

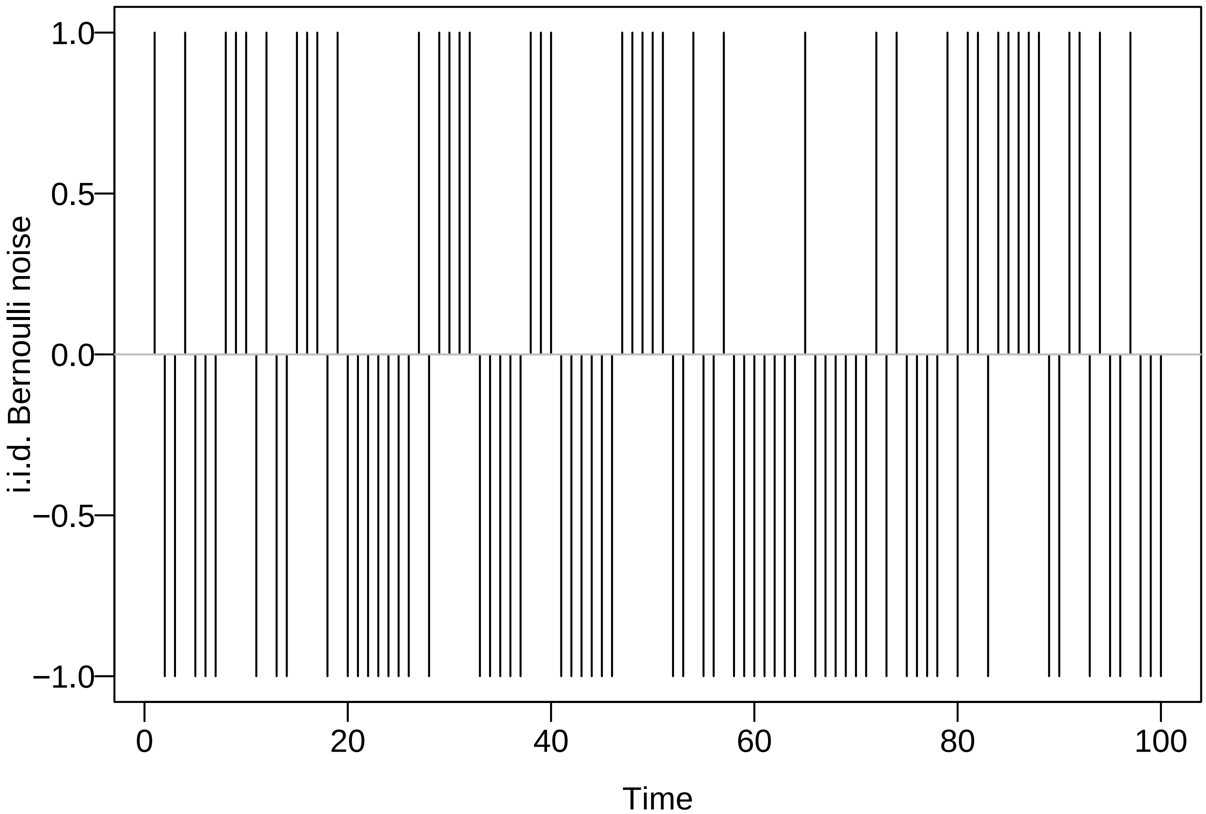
Contents

Examples of i.i.d. Noise

```
T = 100
t <- 1:T

## i.i.d. normal
normal_iid <- rnorm(T)
par(las = 1, mgp = c(2, 0.5, 0), mar = c(3.6, 3.6, 0.8, 0.6))
plot(t, normal_iid, type = "l", las = 1,
     xlab = "Time", ylab = "i.i.d. N(0, 1) noise")
abline(h = 0, col = "gray")
```




```r
## i.i.d. Binary
ber_iid <- replicate(T, rbinom(1, 1, 0.5))
ber_iid <- ifelse(ber_iid == 0, -1, 1)
plot(t, ber_iid, type = "h", las = 1,
     xlab = "Time", ylab = "i.i.d. Bernoulli noise")
abline(h = 0, col = "gray")
```

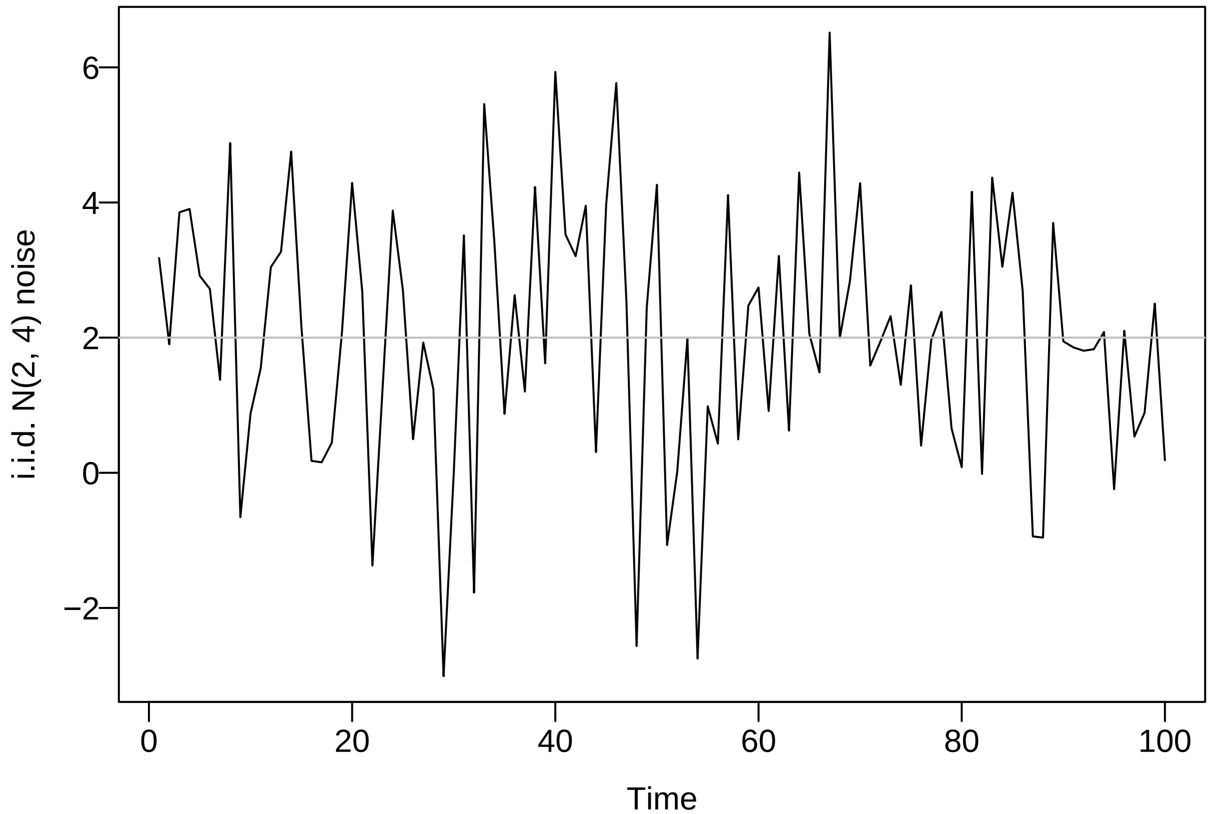


Examples realizations of white noise processes

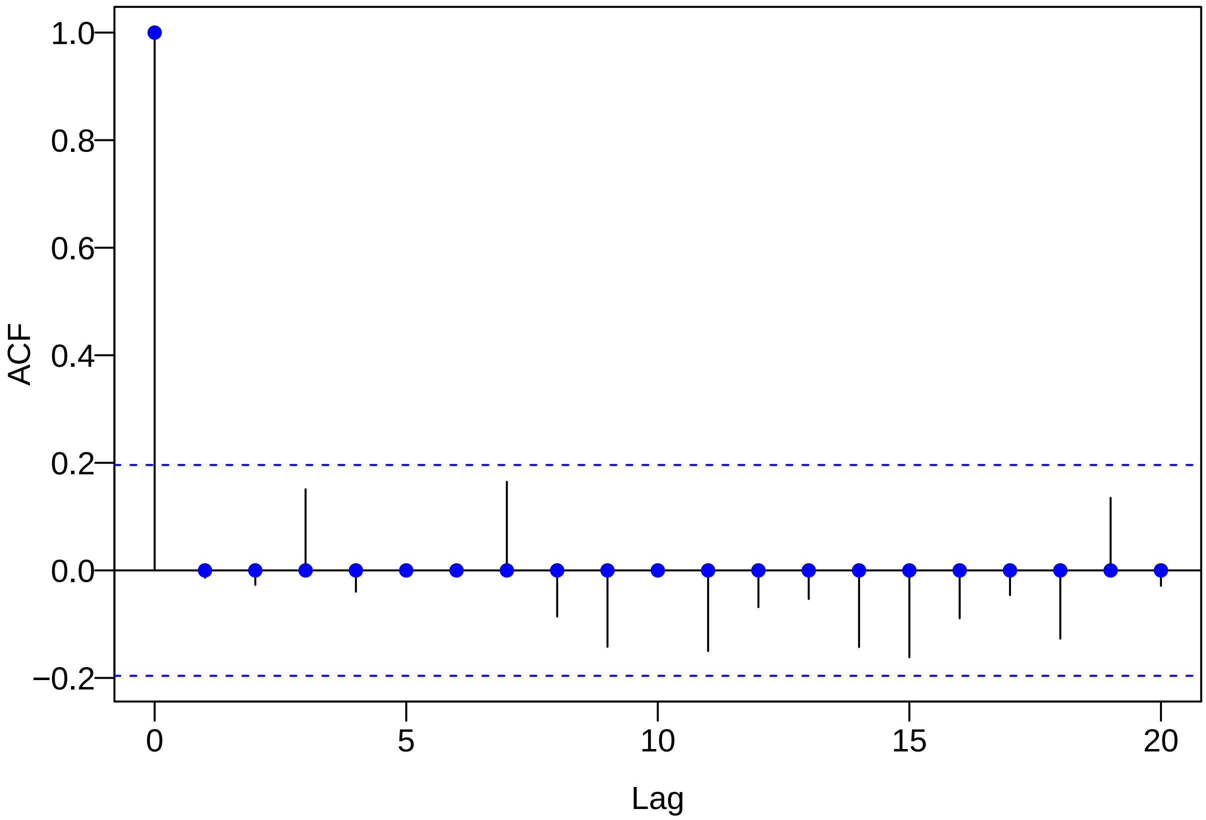
If Z_t is a *white noise* process, then its mean and variance are constants and uncorrelated in time

Note: here we do not require the sequence follow the same distribution.

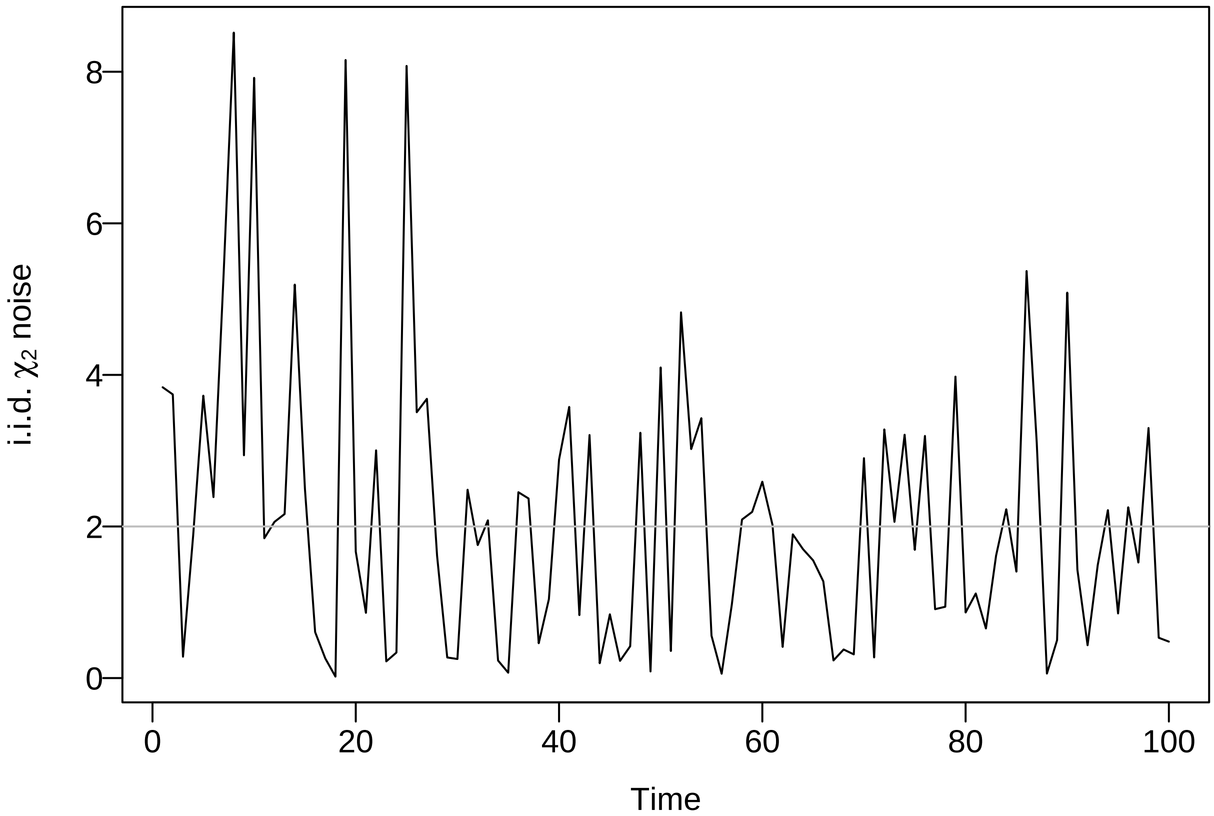
```
T = 100
t <- 1:T
WN1 <- rnorm(n = T, mean = 2, sd = 2)

par(las = 1, mgp = c(2, 0.5, 0), mar = c(3.6, 3.6, 0.8, 0.6))
plot(t, WN1, type = "l", xlab = "Time", ylab = "i.i.d. N(2, 4) noise")
abline(h = 2, col = "gray")
```



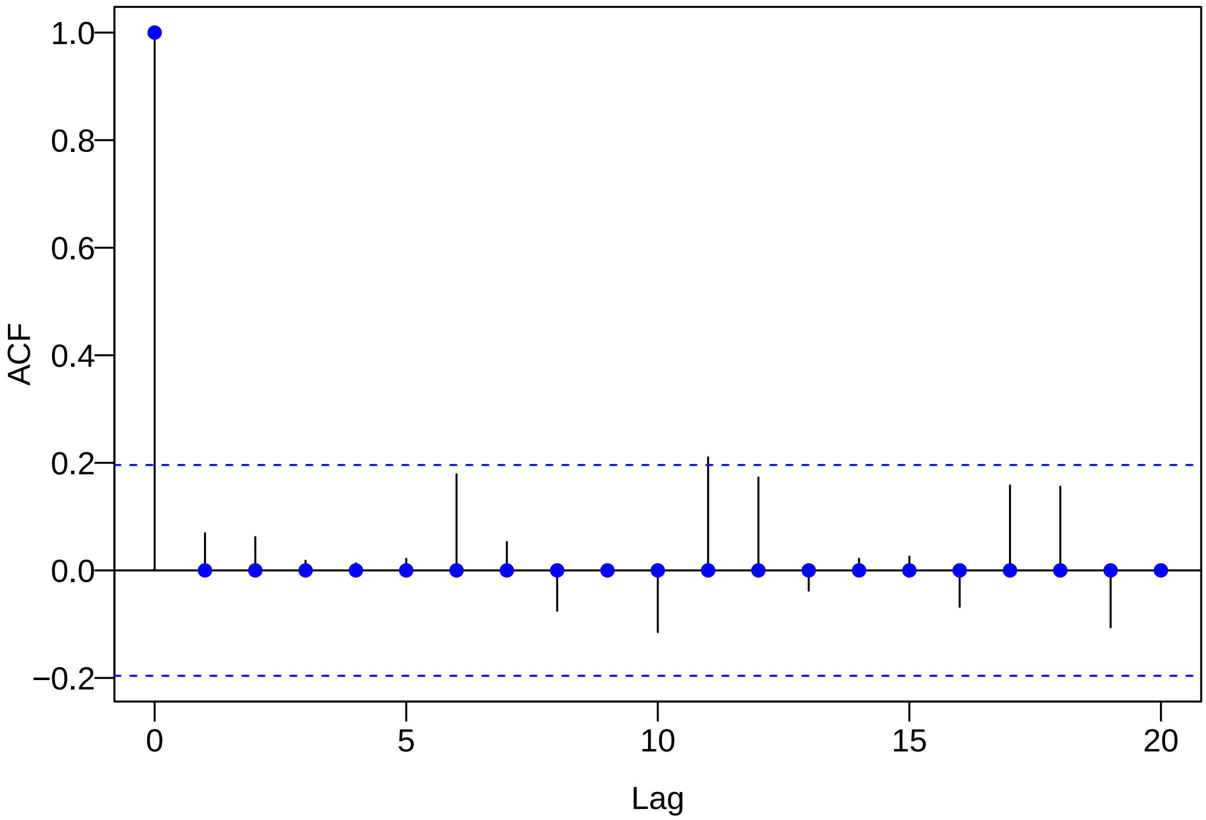

```r
acf(WN1)
points(0:20, c(1, rep(0, 20)), pch = 16, col = "blue")
```



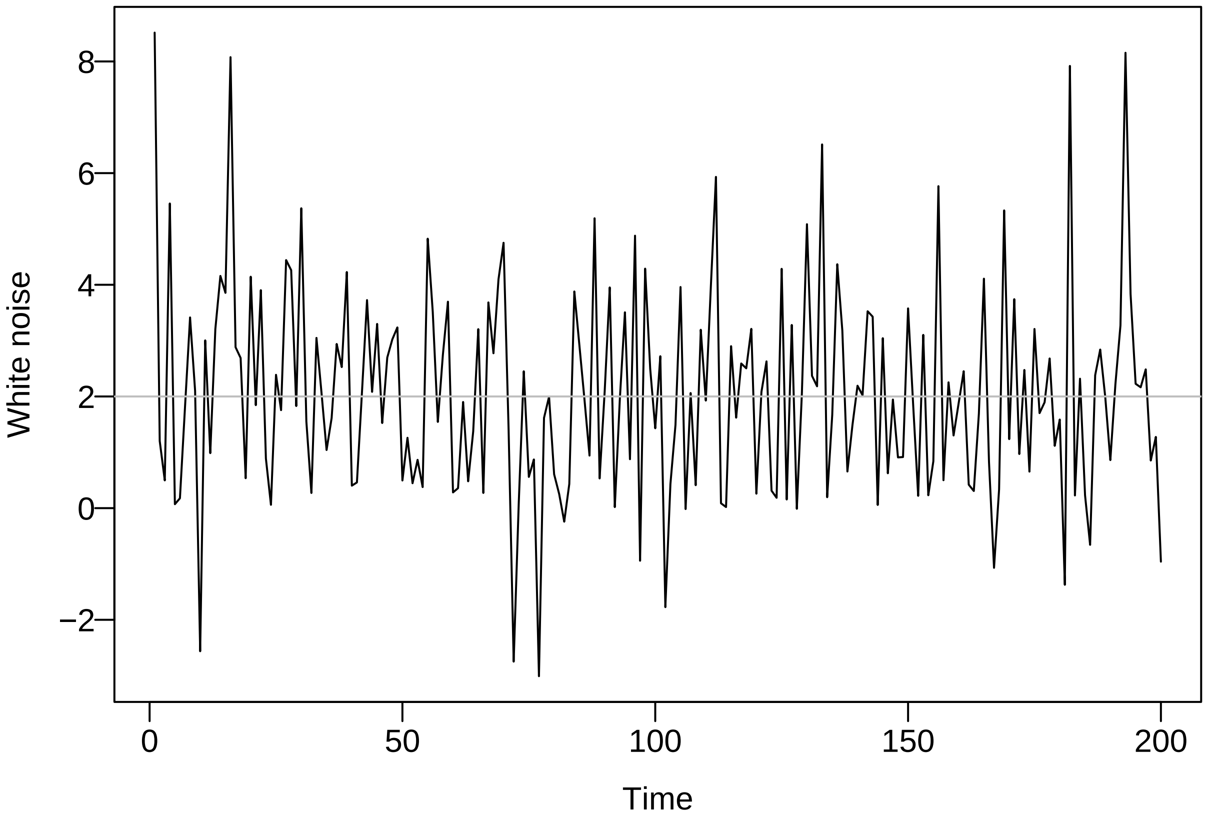

```
WN2 <- rchisq(n = T, df = 2)

plot(t, WN2, type = "l", xlab = "Time", ylab = expression(paste("i.i.d. ", chi[2], " noise")))
abline(h = 2, col = "gray")
```



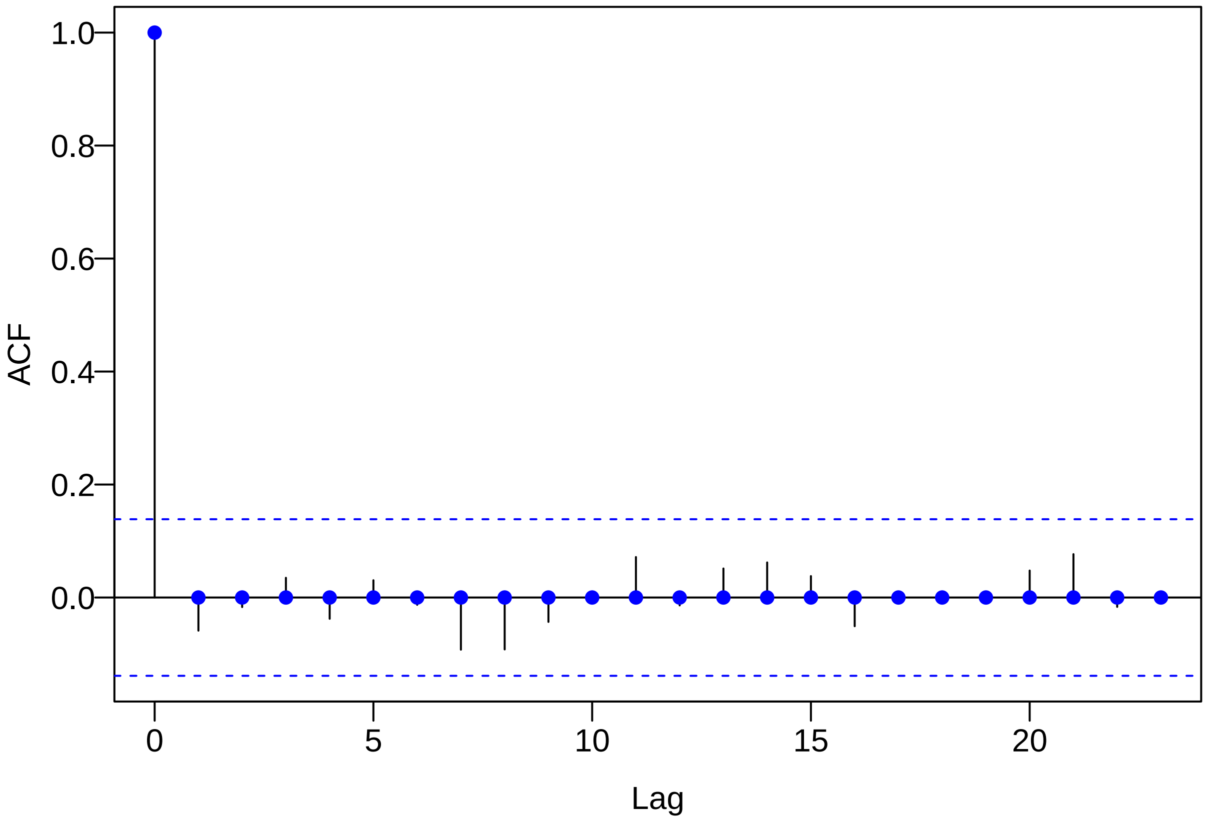

```
acf(WN2)
points(0:20, c(1, rep(0, 20)), pch = 16, col = "blue")
```




```
WN3 <- c(WN1, WN2)[sample(1:200)]
plot(1:200, WN3, type = "l", xlab = "Time", ylab = expression(paste("White noise")))
abline(h = 2, col = "gray")
```




```r
acf(WN3)
points(0:23, c(1, rep(0, 23)), pch = 16, col = "blue")
```

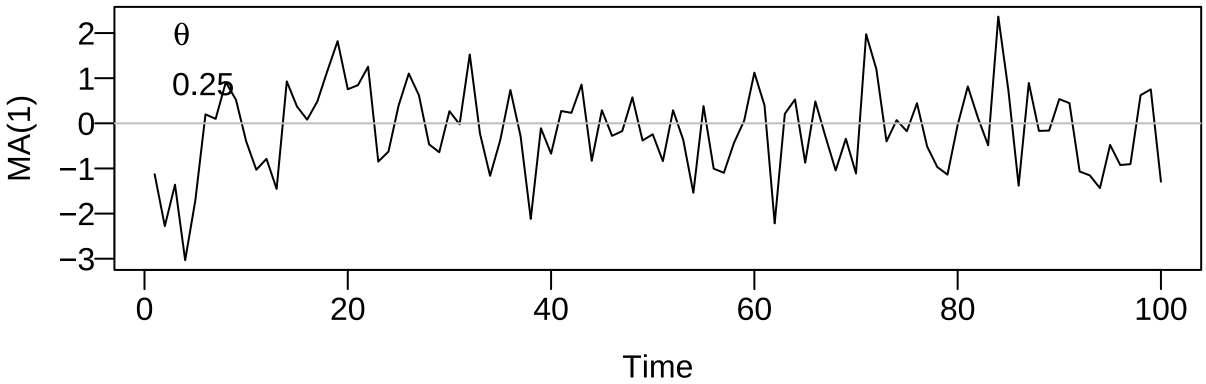


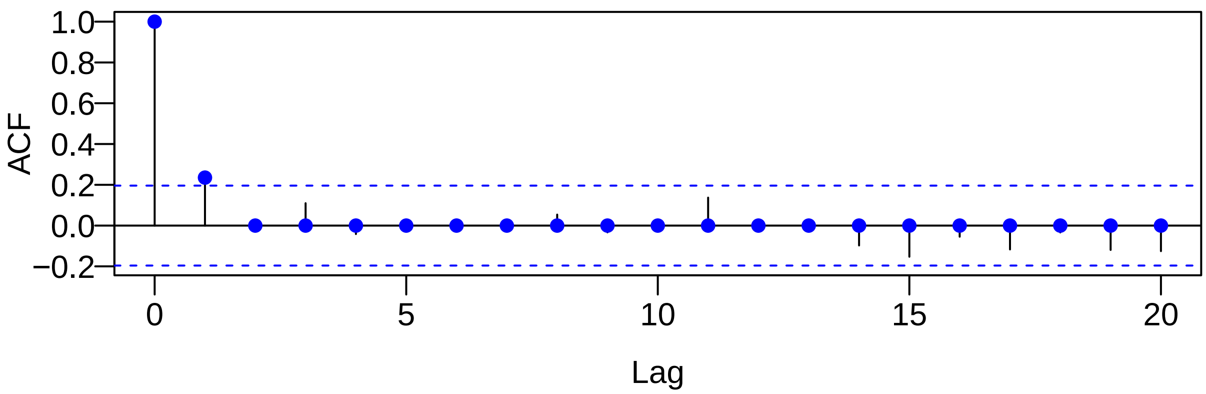
MA(1) processes

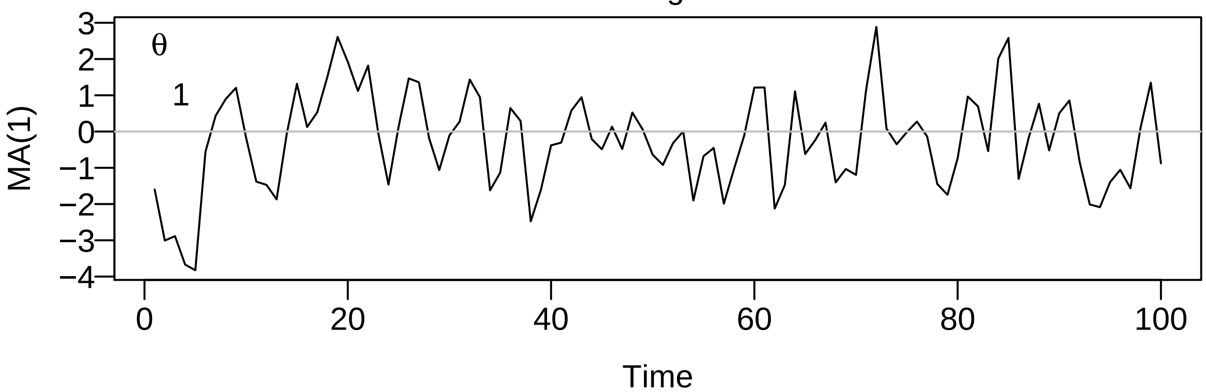
$$\eta_t = Z_t + \theta Z_{t-1},$$

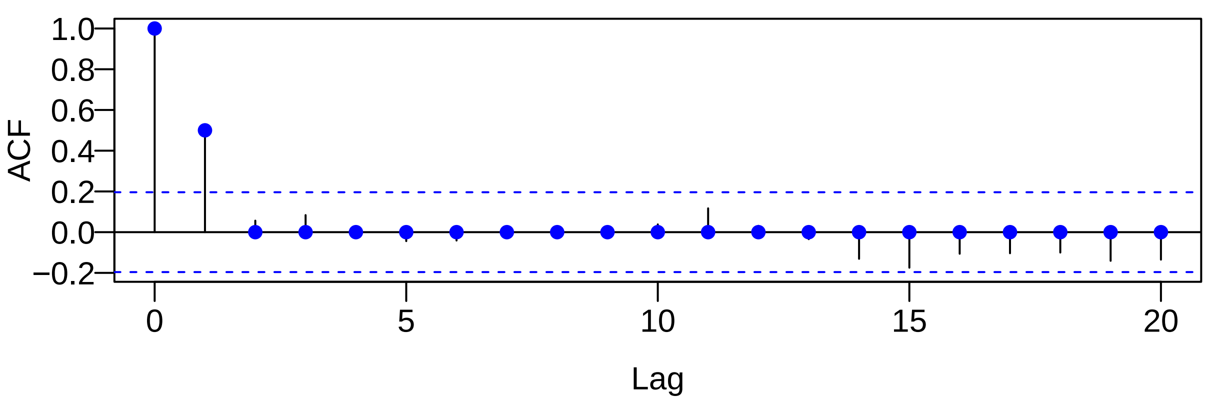
where $Z \sim \mathrm{WN}(0, \sigma^2)$.

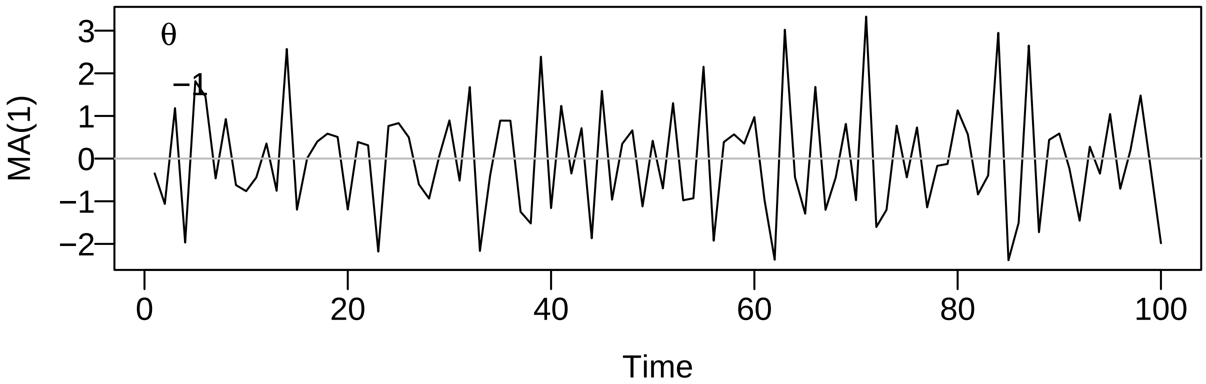
```
T = 100
t <- 1:T
z <- rnorm(110)
theta <- c(0.25, 1, -1, -0.25)

par(las = 1, mgp = c(2, 0.5, 0), mar = c(3.6, 3.6, 0.8, 0.6), mfrow = c(2, 1))
for (i in 1:4){
  MA1 <- filter(z, sides = 1, c(1, theta[i]))[-(1:10)]
  plot(t, MA1, type = "l", xlab = "Time", ylab = "MA(1)")
  abline(h = 0, col = "gray")
  legend("topleft", legend = theta[i], title = expression(theta),
         bty = "n")
  acf(MA1)
  points(0:20, c(1, theta[i] / (1 + theta[i]^2), rep(0, 19)),
         pch = 16, col = "blue")
}
```

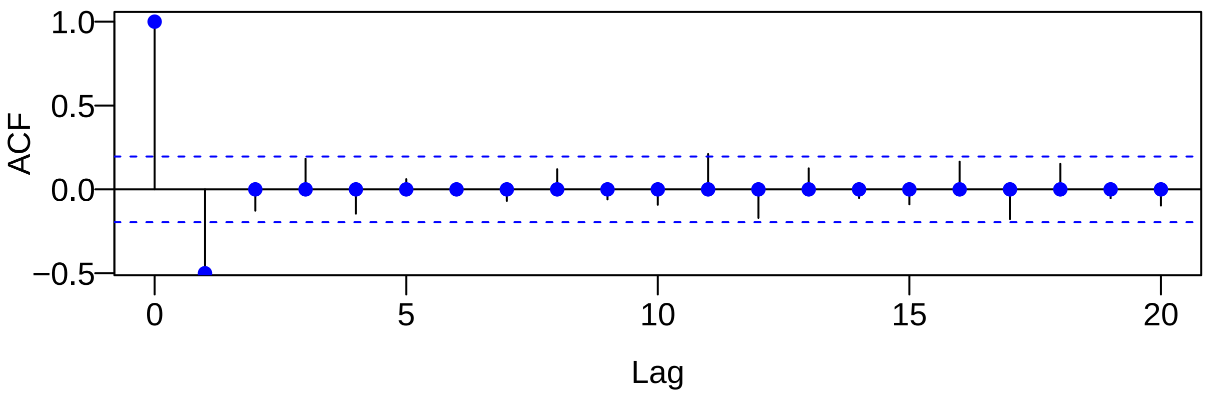

θ
0.25
MA(1)
Time

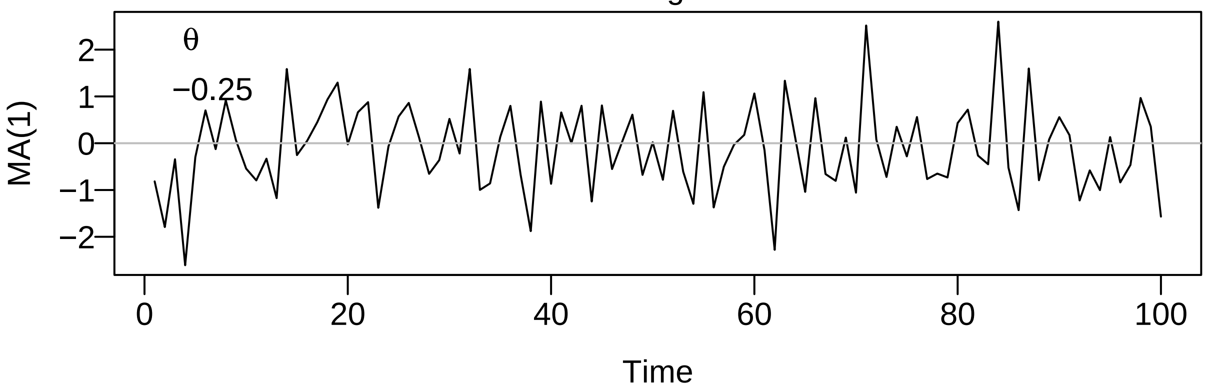

ACF
Lag

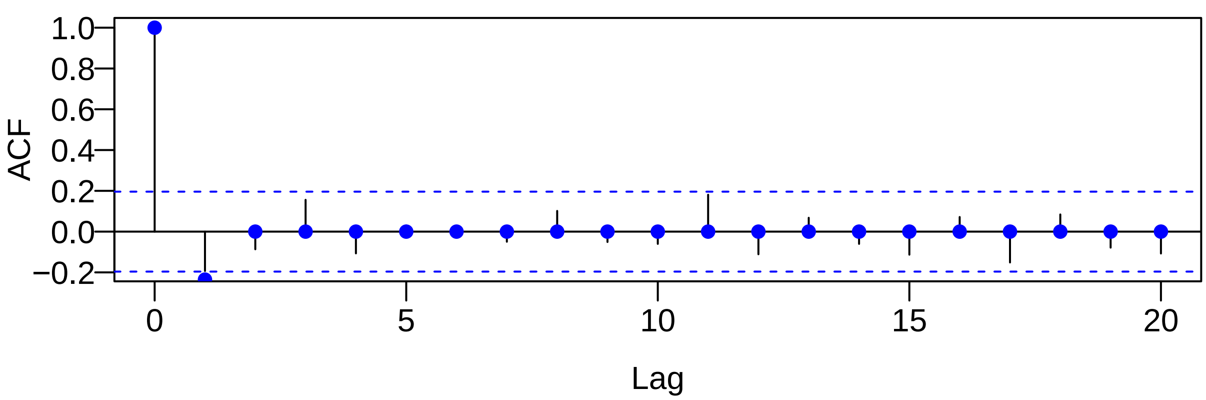

θ
1
MA(1)
Time


ACF
Lag

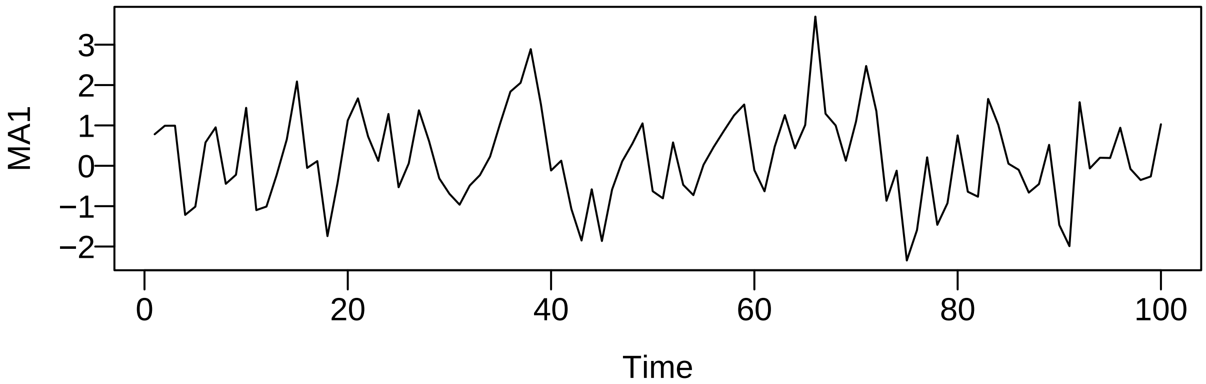

θ
−1
MA(1)
3
2
1
0
−1
−2
0
20
40
60
80
100
Time

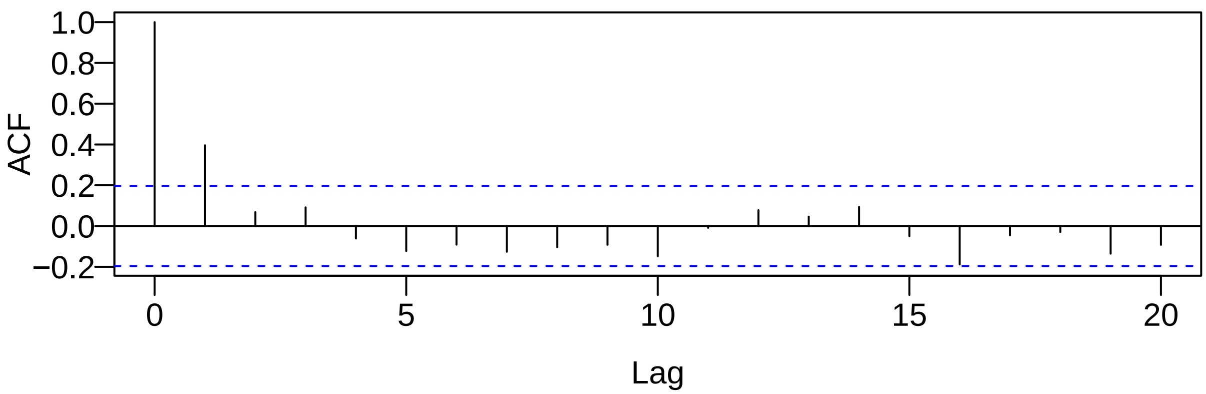

ACF
1.0
0.5
0.0
−0.5
0
5
10
15
20
Lag


θ
−0.25
MA(1)
2
1
0
−1
−2
0
20
40
60
80
100
Time


ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
5
10
15
20
Lag

```r
##another way to simulate MA(1)
MA1 <- arima.sim(n = 100, list(ma = c(0.5)))
plot(MA1)
acf(MA1)
```



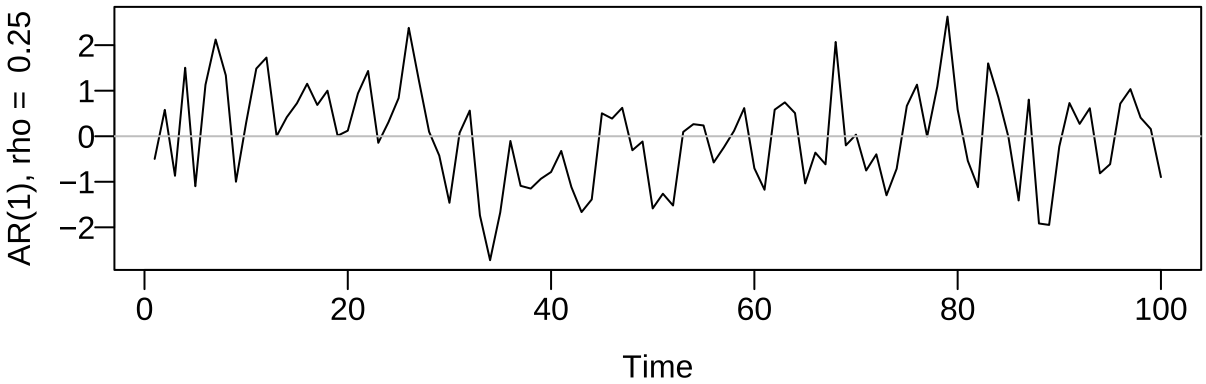


AR(1) processes

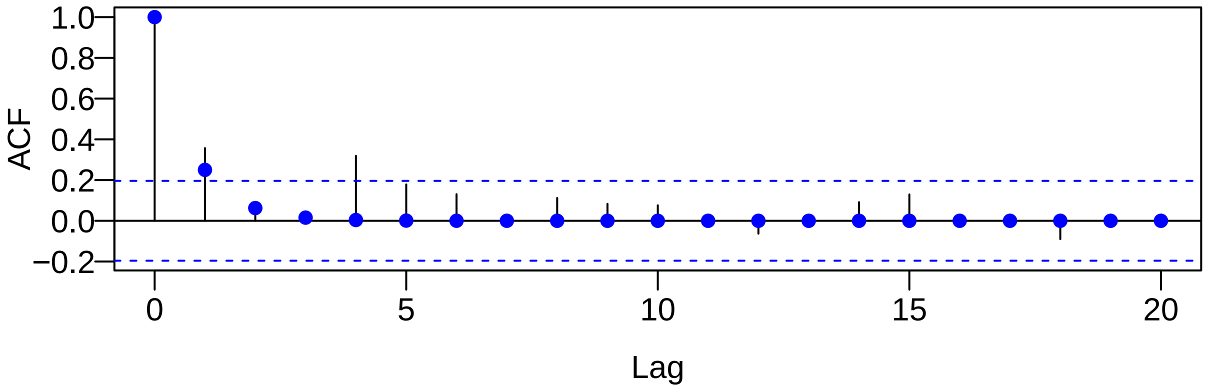
$$\eta_t = \phi \eta_{t-1} + Z_t,$$

where $|\rho| < 1$ is a constant and η_s and Z_t are uncorrelated for all $s < t \Rightarrow$ future noise is uncorrelated with the current value.

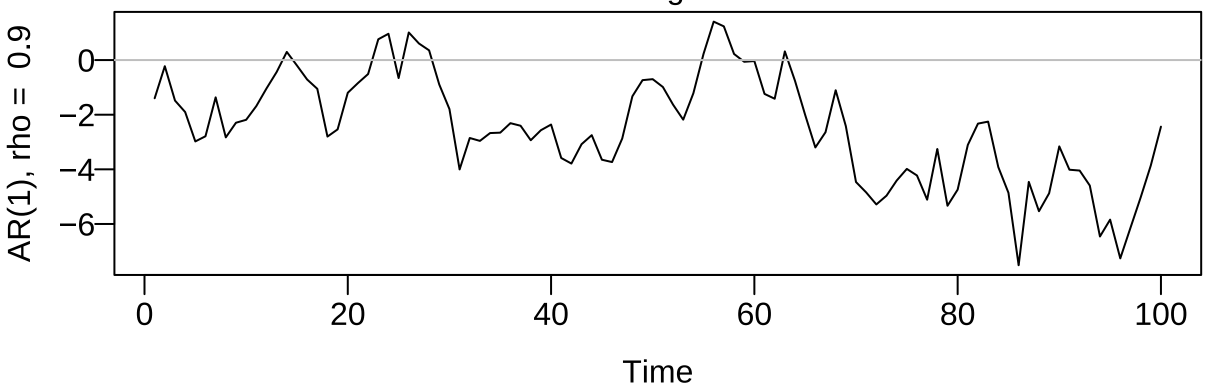
```r
phi <- c(0.25, 0.9, -0.5)

par(las = 1, mgp = c(2, 0.5, 0), mar = c(3.6, 3.6, 0.8, 0.6), mfrow = c(2, 1))
for (i in 1:3){
  AR1 <- arima.sim(n = 100, list(ar = c(phi[i])))
  plot(t, AR1, type = "l", xlab = "Time",
       ylab = paste("AR(1), rho = ", phi[i]))
  abline(h = 0, col = "gray")
  acf(AR1)
  points(0:20, phi[i]^(0:20), pch = 16, col = "blue")
}
```



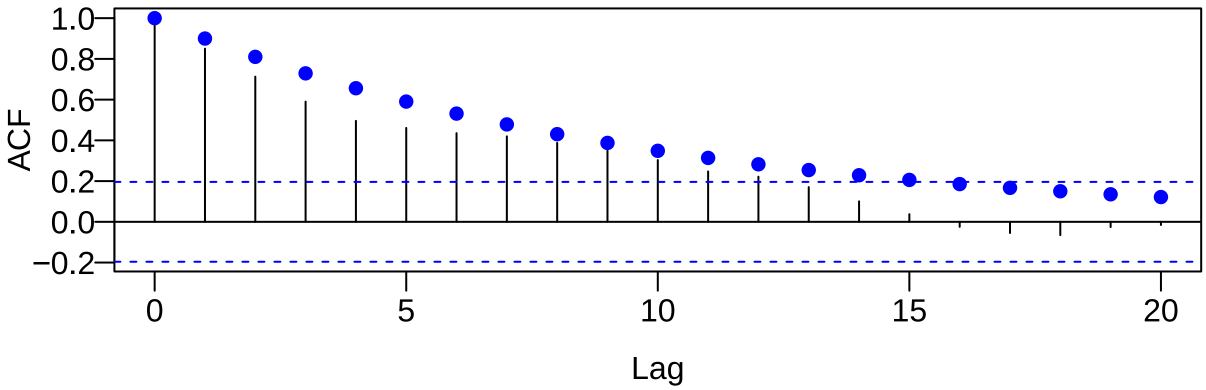
AR(1), rho = 0.25
2
1
0
−1
−2
0
20
40
60
80
100
Time



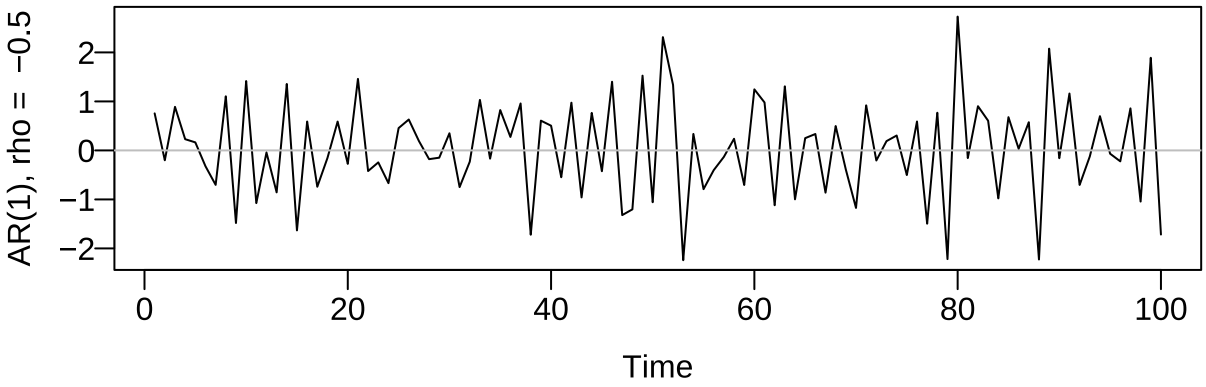
ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
5
10
15
20
Lag

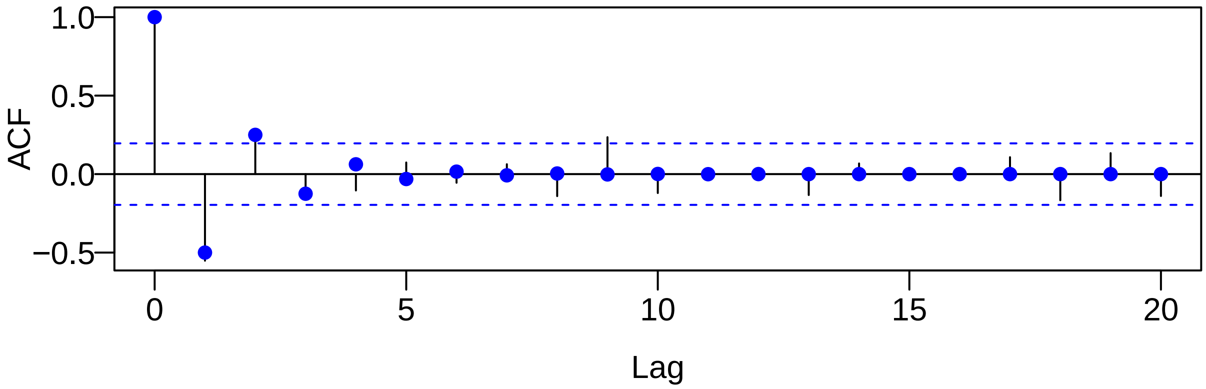


AR(1), rho = 0.9
0
−2
−4
−6
0
20
40
60
80
100
Time



ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
5
10
15
20
Lag

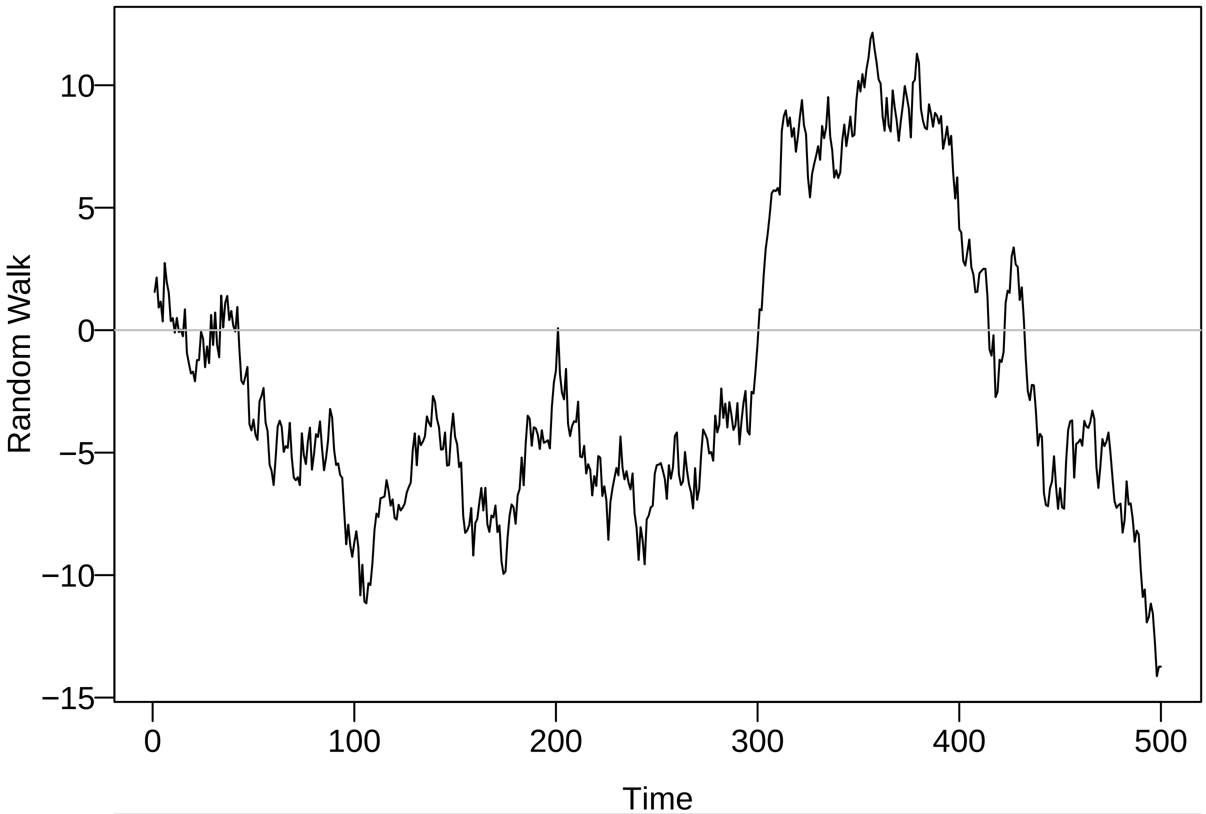




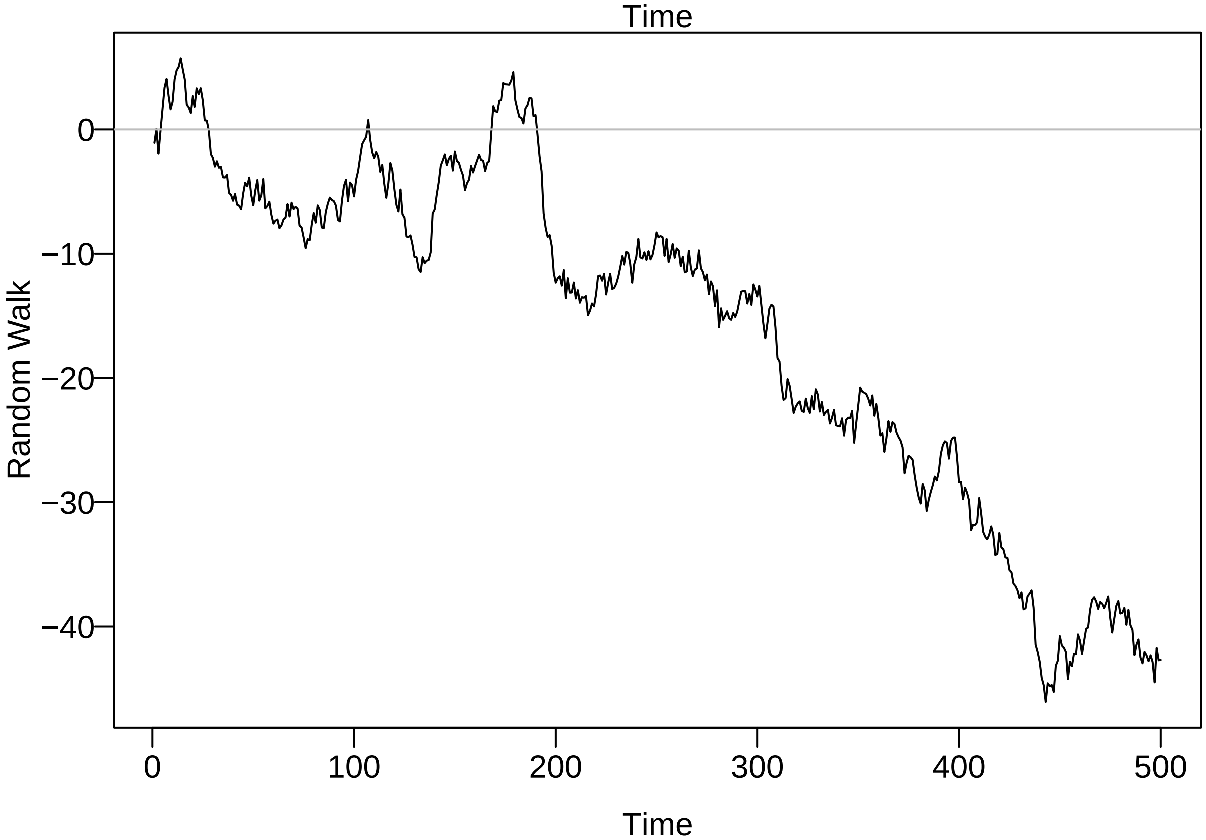
Random walk

$$\eta_t = \sum_{s=1}^{t} Z_s.$$

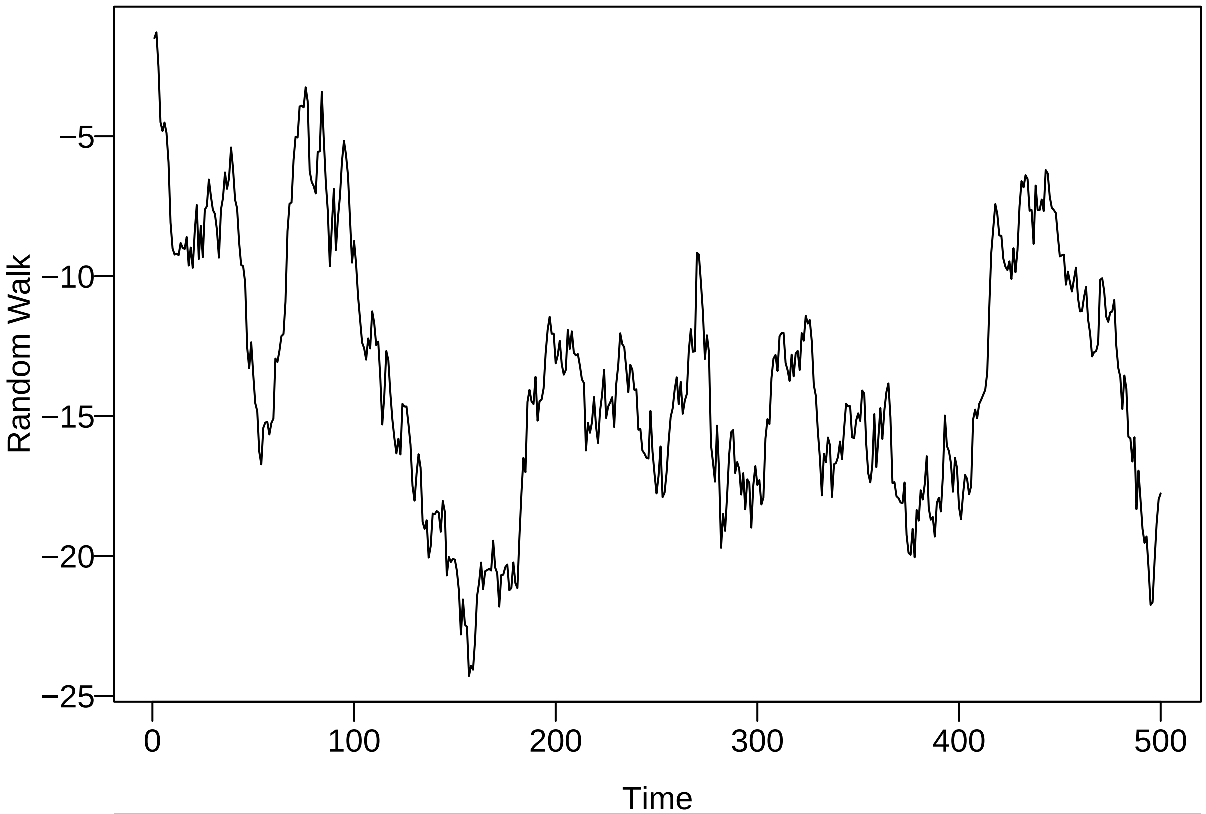
```
par(las = 1, mgp = c(2, 0.5, 0), mar = c(3.6, 3.6, 0.8, 0.6))
for (i in 1:5){
  z <- rnorm(500)
  plot(1:500, cumsum(z), type = "l", xlab = "Time",
       ylab = "Random Walk")
  abline(h = 0, col = "gray")
}
```

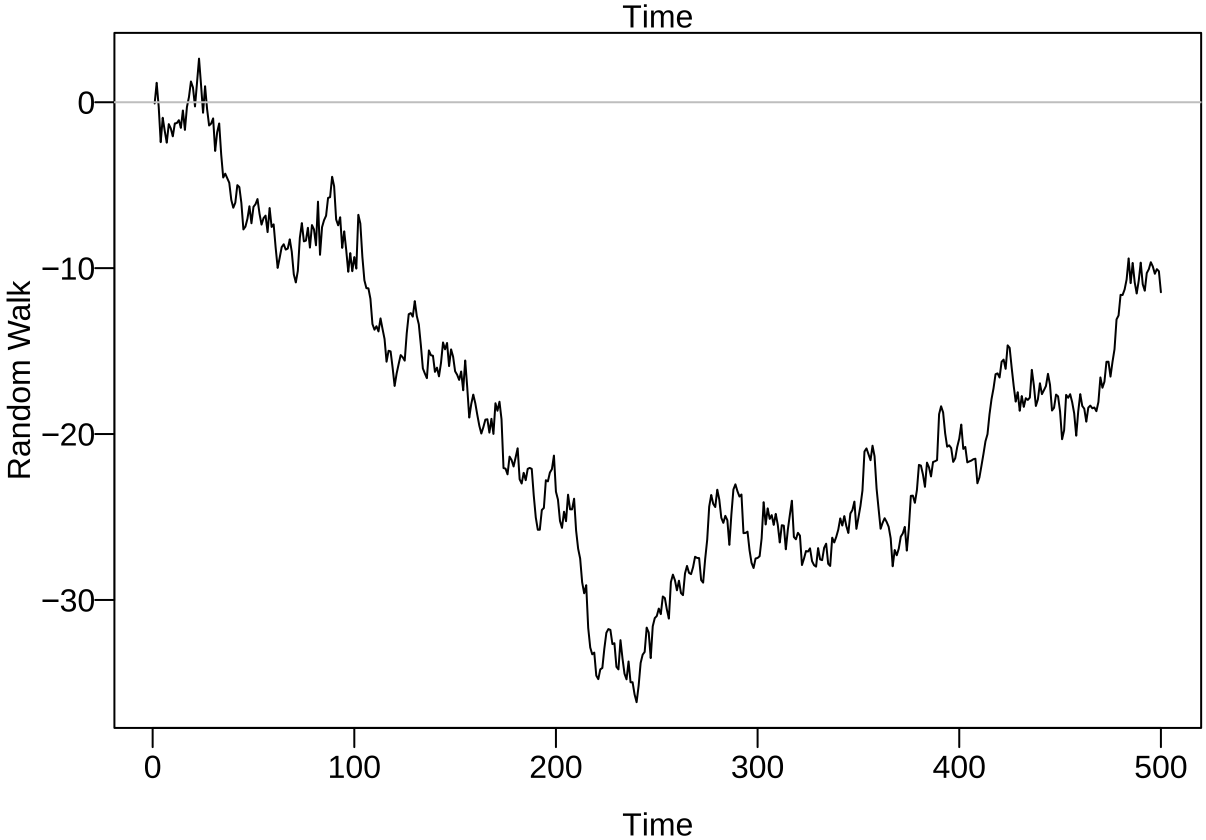


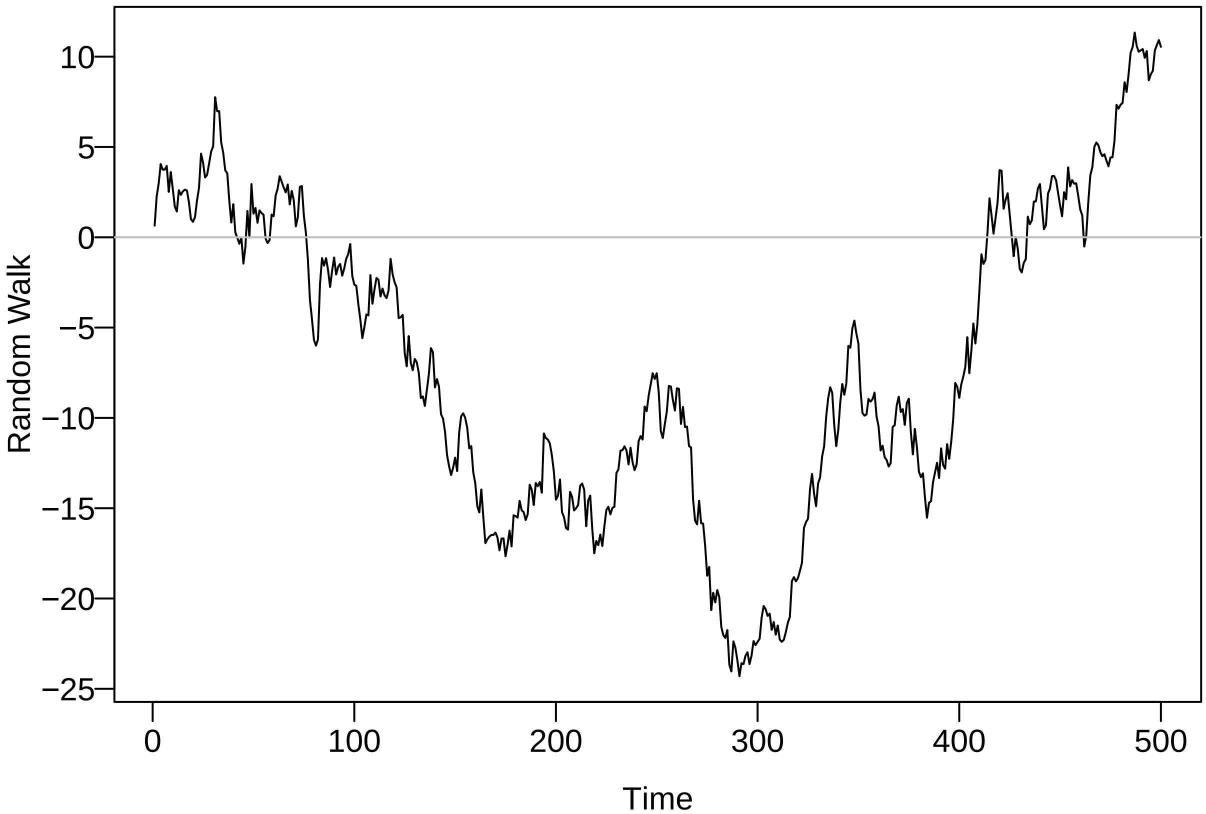
10
5
0
−5
−10
−15
Random Walk
0
100
200
300
400
500
Time



0
−10
−20
−30
−40
Random Walk
0
100
200
300
400
500
Time


Random Walk
Time
−5
−10
−15
−20
−25
0
100
200
300
400
500

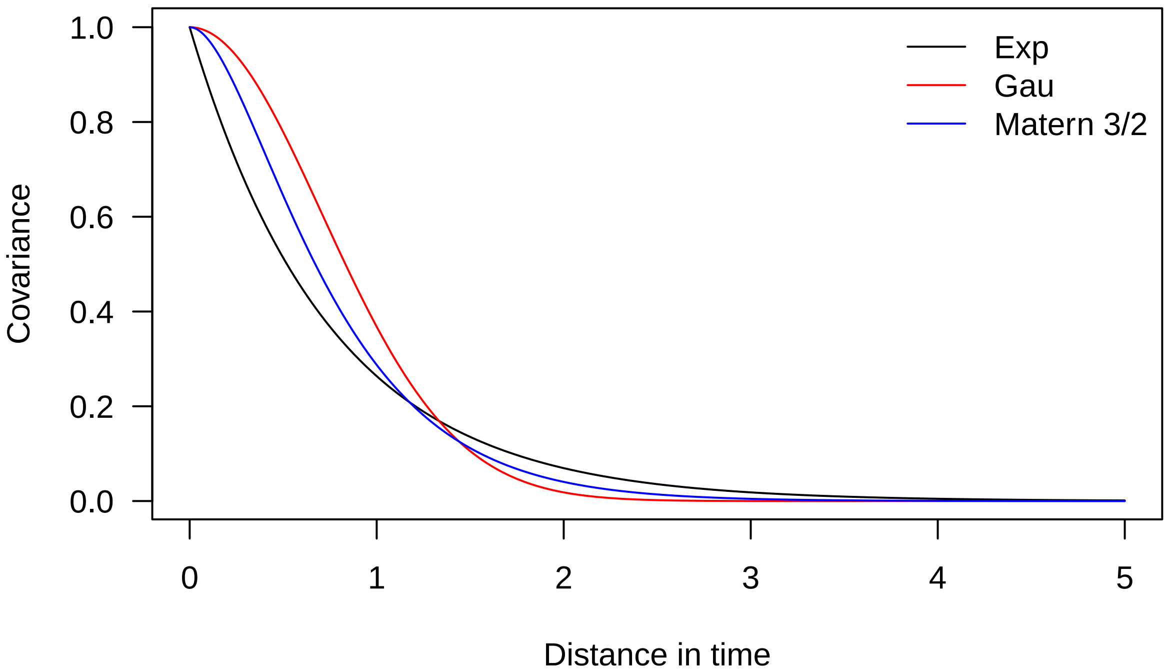

Random Walk
Time
0
−10
−20
−30
0
100
200
300
400
500



Gaussian process

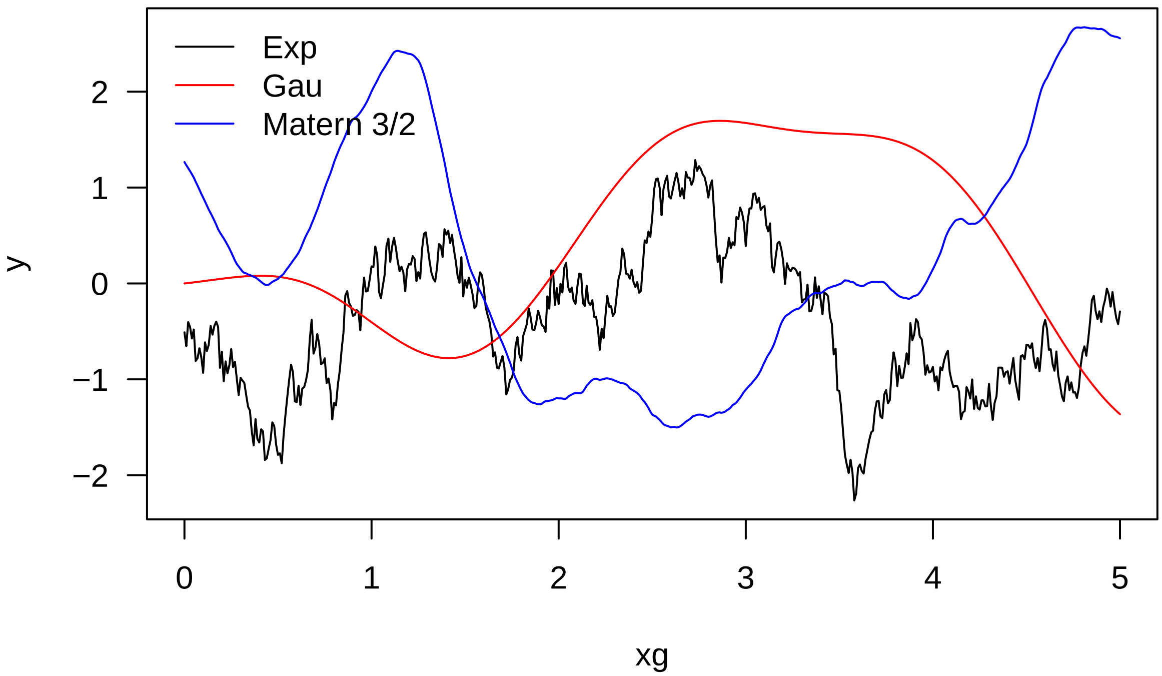
Different covaraince functions (kernels)

```
library(fields)
# Commonly used covariance functions
cov.exp <- function(h, pars) pars[1] * exp(-h / pars[2])
cov.doubleExp <- function(h, pars) pars[1] * exp(-(h / pars[2])^2)
cov.Matern <- function(h, pars) Matern(h, phi = pars[1], range = pars[2], smoothness = pars[3])

xg <- seq(0, 5, 0.01)
c_exp <- cov.exp(xg, c(1, 0.75))
c_doubleExp <- cov.doubleExp(xg, c(1, 1))
c_Matern <- cov.Matern(xg, c(1, 0.4, 1.5))

plot(xg, c_exp, type = "l", ylab = "Covariance", xlab = "Distance in time", las = 1)
lines(xg, c_doubleExp, col = "red")
lines(xg, c_Matern, col = "blue")
legend("topright", legend = c("Exp", "Gau", "Matern 3/2"),
       col = c("black", "red", "blue"), lty = 1, bty = "n")
```



Generate one sample from each Gaussian Process with different kernels

```
Sigma_exp <- cov.exp(rdist(xg), c(1, 0.75))
Sigma_doubleExp <- cov.doubleExp(rdist(xg), c(1, 1))
Sigma_Matern <- cov.Matern(rdist(xg), c(1, 0.4, 1.5))
library(MASS)
set.seed(123)
sim_exp_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_exp)
sim_doubleExp_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_doubleExp)
sim_Matern_1d <- mvrnorm(n = 1, rep(0, 501), Sigma_Matern)

plot(xg, sim_exp_1d, type = "l", ylim = range(sim_exp_1d, sim_doubleExp_1d,
                                              sim_Matern_1d), ylab = "y", las = 1)
lines(xg, sim_doubleExp_1d, col = "red")
lines(xg, sim_Matern_1d, col = "blue")
legend("topleft", legend = c("Exp", "Gau", "Matern 3/2"),
       col = c("black", "red", "blue"), lty = 1, bty = "n")
```


Exp
Gau
Matern 3/2
y
xg
2
1
0
−1
−2
0
1
2
3
4
5