Time-Series Model

Introduction

A time series model specifies how the random variables $\{X_t\}$ behave jointly over time. This can be done in two main ways:

1. **Full Joint Distribution:** The model might completely specify the joint probability distribution of $\{X_t\}$.
2. **Moments (Means and Covariances):** In many practical situations, especially when dealing with linear processes or Gaussian assumption: it is sufficient to specify just the means and covariances (or autocovariances) of the process. This is because for Gaussian processes, the mean and covariance completely determine the joint distribution.

! Definition

A **time series model** for the observed data $\{x_t\}$ is a specification of the joint distributions (or possibly only the means and covariances) of a sequence of random variables $\{X_t\}$ of which $\{x_t\}$ is postulated to be a realization. Here, x_t is the single outcome of stochastic process X_t.

Zero-mean models

IID noise:

No trend or seasonal component, and the observations are simply independent and identically distributed (iid) random variables with zero mean.

$$P[X \leq x_1, \cdots, X_n \leq x_n] = P[X_1 \leq x_1] \cdots P[X_n \leq x_n] = F(x_1) \cdots F(x_n)$$

F is the cumulative distribution function of each of the iids.

$$P[X_{n+h} \leq x | X_1 = x_1, \cdots, X_n = x_n] = P[X_{n+h} \leq x]$$

Simple Symmetric Random Walk:

A **simple symmetric random walk** is a foundational stochastic process that models a path formed by successive random steps. Assume, we start at $S_0 = 0$. For $t \geq 1$, define

$$S_t = X_1 + X_2 + \cdots + X_t$$

where $\{X_t\}$ are independent and identically distributed (iid) random variables. Here, each binary step is defined as:

$$X_t = \begin{cases} +1 & \text{with probability 0.5 ``heads''}, \\ -1 & \text{with probability 0.5 ``tails''}. \end{cases}$$

So, we start at position 0 and move ± 1 at each time t based on a fair coin toss.

```r
# For reproducibility we set the seed

set.seed(123)
steps <- sample(c(-1, 1), 200, replace = TRUE)
s <- c(0, cumsum(steps))
plot(0:200, s, type = "l",
     xlab = "Time (t)", ylab = expression(S[t]),
     main = "Simple Symmetric Random Walk")
```

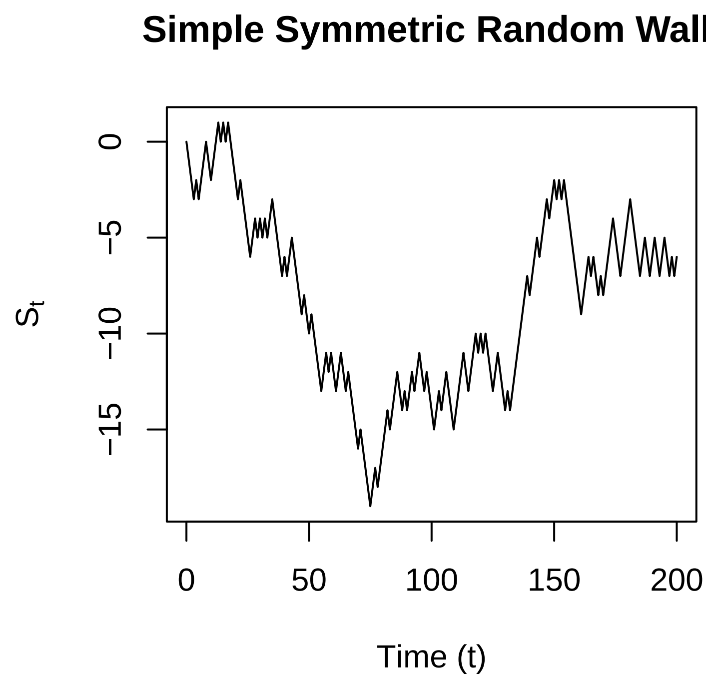


Figure 1: Simulation of simple symmetric RW

Models with trends and seasonality

Using a zero-mean model for data is inappropriate. When analyzing time series data with trends or seasonality, a zero-mean model is often inadequate because such data inherently violates the assumption of a constant mean over time. Instead, the series should be decomposed into distinct components: a **trend** (m_t), capturing slow, long-term changes; **seasonality** (s_t), representing periodic fluctuations; and **stationary residuals** (Y_t), which account for random, zero-mean noise. To model this structure, we can look for parametric methods like **linear regression** or nonparametric techniques (e.g., moving averages).

Assume there is a clear increasing trend in the population of USA. So, we can suggest a model of the form:

$$X_t = m_t + Y_t$$

- m_t (trend component): A slowly changing deterministic function (e.g., linear, quadratic, or exponential).

- Y_t (residuals): A zero-mean stochastic process, ideally stationary.

We can estimate m_t by minimizing the sum of squared residuals:

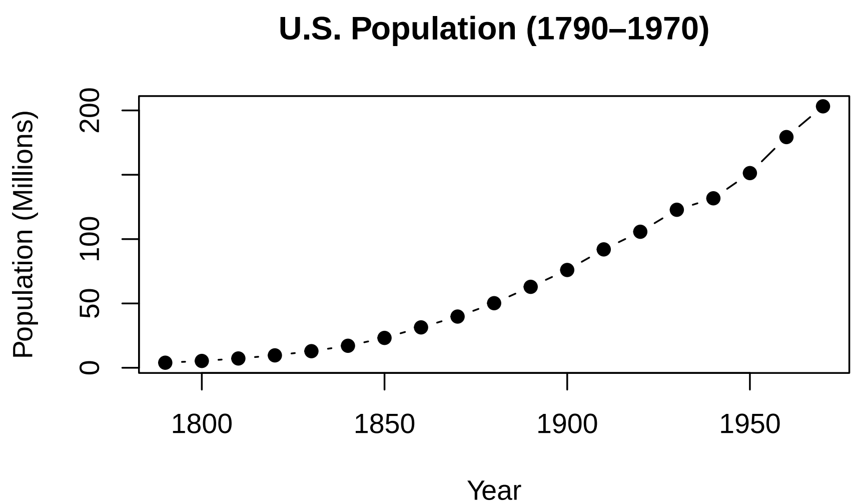
$$\sum_{i=1}^{n}(X_t - m_t)^2$$

We can choose a functional form for m_t such as:

- **Linear**: $m_t = a + bt$
- **Quadratic**: $m_t = a + bt + ct^2$
- **Exponential**: $m_t = ae^{bt}$ (requires linearization via log transforms).

Population of the USA, 1790-1990

```
data <- read.csv("uspop.csv")
ts_data <- ts(data$value, start = 1790, deltat = 10)

# Plot the time series with points
plot(ts_data,
     main = "U.S. Population (1790-1970)",
     xlab = "Year",
     ylab = "Population (Millions)",
     type = "b",  # "b" for both points and lines
     pch = 19)    # Use solid circles as points
```




```r
# Load necessary packages
library(ggplot2)
```

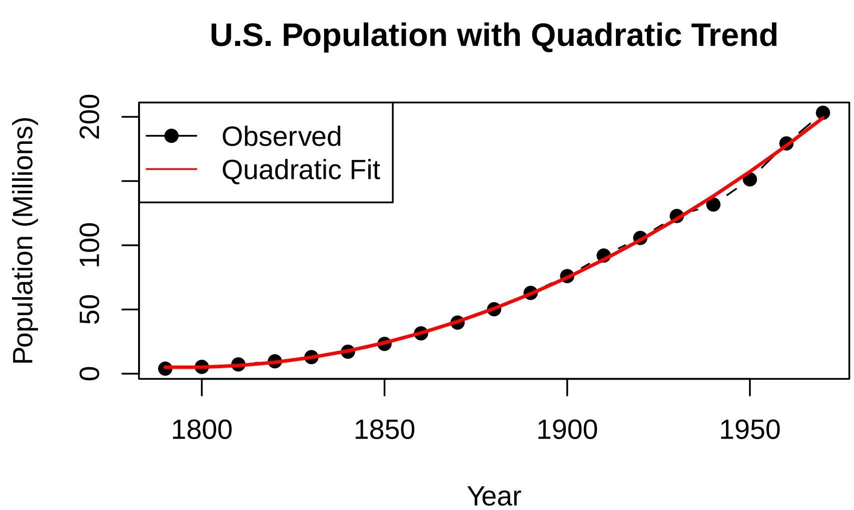
```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```r
# Read and prepare data
data <- read.csv("uspop.csv")
data$t <- data$time - 1790  # Create time variable starting at 0 for 1790

# Fit quadratic model
quad_model <- lm(value ~ t + I(t^2), data = data)

# Add fitted values to dataframe
data$fitted <- predict(quad_model)

# Show model coefficients
summary(quad_model)
```

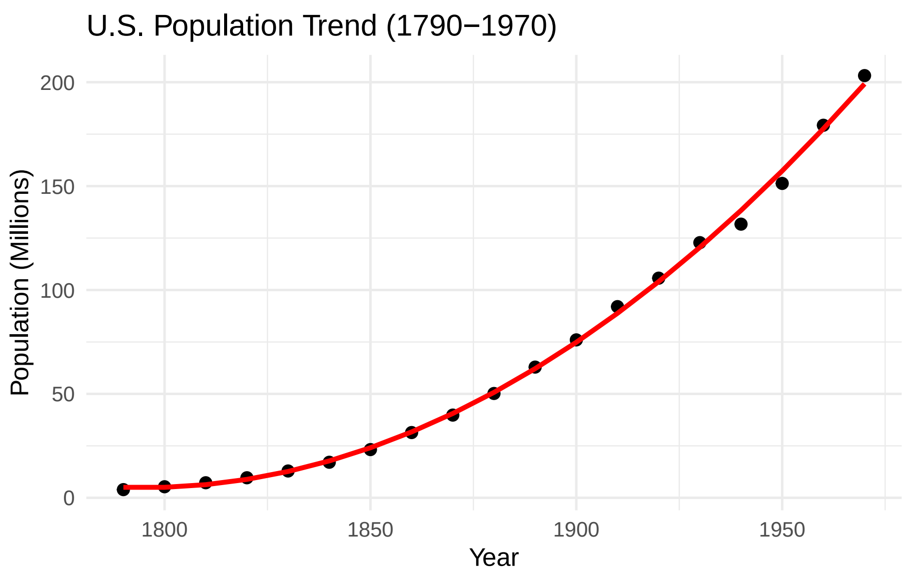
```
Call:
lm(formula = value ~ t + I(t^2), data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-6.5997 -0.7105  0.2669  1.4065  3.9879
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.0416692  1.7280922   2.917   0.0101 *
t           -0.0633015  0.0445092  -1.422   0.1742
I(t^2)       0.0063446  0.0002387  26.584 1.14e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.78 on 16 degrees of freedom
Multiple R-squared:  0.9983,    Adjusted R-squared:  0.9981
F-statistic:  4645 on 2 and 16 DF,  p-value: < 2.2e-16
```

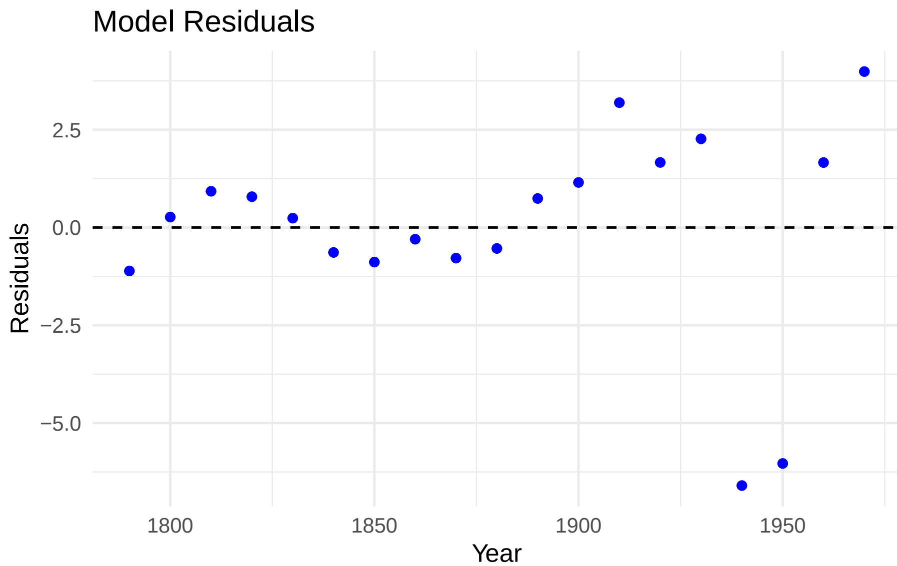
```r
# Base R plot version
plot(ts_data,
     main = "U.S. Population with Quadratic Trend",
     xlab = "Year",
     ylab = "Population (Millions)",
     type = "b", pch = 19)
lines(data$time, data$fitted, col = "red", lwd = 2)
legend("topleft",
       legend = c("Observed", "Quadratic Fit"),
       col = c("black", "red"),
       lty = 1, pch = c(19, NA))
```



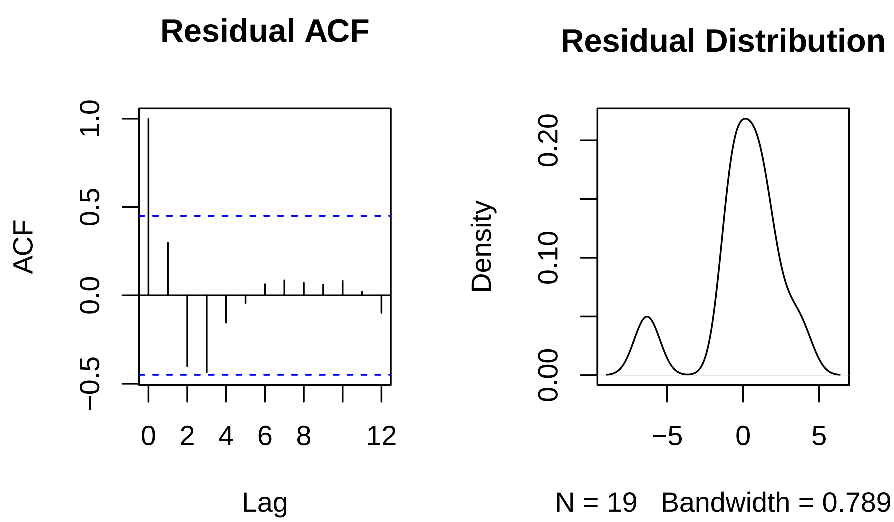
```r
# ggplot2 version
ggplot(data, aes(x = time)) +
  geom_point(aes(y = value), size = 2) +
  geom_line(aes(y = fitted), color = "red", linewidth = 1) +
  labs(title = "U.S. Population Trend (1790-1970)",
       x = "Year",
       y = "Population (Millions)") +
  theme_minimal()
```




```r
# Add residuals
data$residuals <- residuals(quad_model)

# Residual plot
ggplot(data, aes(x = time, y = residuals)) +
  geom_point(color = "blue") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(title = "Model Residuals",
       x = "Year",
       y = "Residuals") +
  theme_minimal()
```




```r
# Residual diagnostics
par(mfrow = c(1,2))
acf(data$residuals, main = "Residual ACF")
plot(density(data$residuals), main = "Residual Distribution")
```



(a) Trend
The U.S. population data exhibits a **strong upward non-linear trend**. The

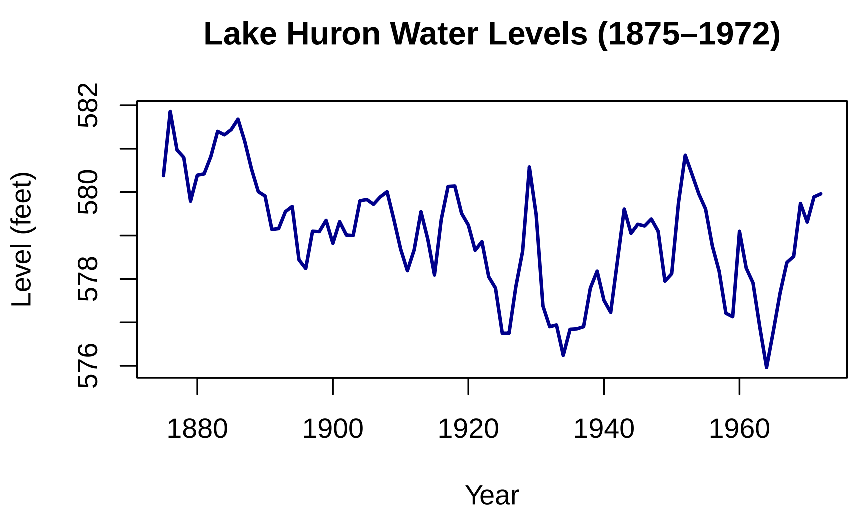
quadratic model (with a significant quadratic term, $p < 0.001$) captures accelerating growth, explaining 99.83% of the variance. The curve reflects increasing growth rates over time, consistent with historical population expansion.

(b) Seasonal Component
There is **no seasonal component** in the data. Measurements are decennial (10-year intervals), and residuals show no cyclical patterns. Seasonality, as seen in monthly or quarterly data, does not apply here.

(c) Sharp Changes in Behavior
The series displays **no abrupt shifts or discontinuities**. Population growth follows a smooth, accelerating trajectory aligned with the quadratic trend. No sudden spikes, drops, or structural breaks are evident.

Level of Lake Huron 1875–1972

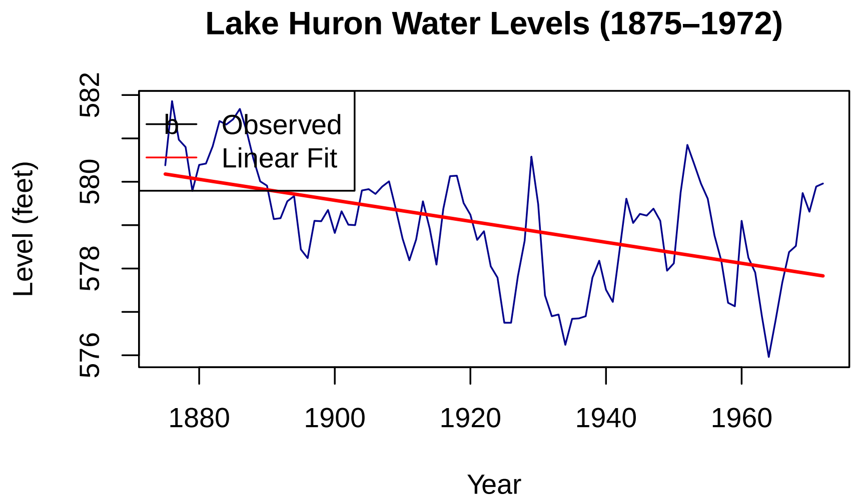
```r
lake_data <- read.csv("LakeHuron.csv")
lake_ts <- ts(lake_data$level,
             start = 1875,
             frequency = 1)  # Annual data
plot(lake_ts,
     main = "Lake Huron Water Levels (1875-1972)",
     xlab = "Year",
     ylab = "Level (feet)",
     col = "darkblue",
     lwd = 2)
```



```r
# Load necessary packages
library(ggplot2)
lake_data <- read.csv("LakeHuron.csv")
lake_data$t <- lake_data$year - 1875  # Create time variable starting at 0

# Fit linear model
model <- lm(level ~ t, data = lake_data)
lake_data$fitted <- predict(model)
# Show model summary
summary(model)
```

```

Call:
lm(formula = level ~ t, data = lake_data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.50997 -0.72726  0.00083  0.74402  2.53565

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept) 580.177835   0.226616 2560.182  < 2e-16 ***
t            -0.024201   0.004036   -5.996 3.55e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.13 on 96 degrees of freedom
Multiple R-squared:  0.2725,	Adjusted R-squared:  0.2649
F-statistic: 35.95 on 1 and 96 DF,  p-value: 3.545e-08
```

```r
# Base R plot version
plot(lake_ts,
     main = "Lake Huron Water Levels (1875-1972)",
     xlab = "Year",
     ylab = "Level (feet)",
     col = "darkblue",pch = 98)
lines(lake_data$year, lake_data$fitted, col = "red", lwd = 2)
legend("topleft",
       legend = c("Observed", "Linear Fit"),
       col = c("black", "red"),
       lty = 1, pch = c(98, NA))
```



(a) Trend
The data exhibits a **significant linear downward trend** ($\hat{a}_1 = -0.0242$), p-value (3.55×10^{-8}). The fitted model $\hat{m}_t = 580.178 - 0.0242t$ (where t = Year − 1875) indicates a gradual decline in water levels over time. The trend explains **27.25%** of the variance ($R^2 = 0.2725$), reflecting moderate but meaningful linearity.

(b) Seasonal Component
There is **no seasonal component** in the series. The data is annual, and residuals show no cyclical patterns. Seasonality is irrelevant for yearly measurements.

(c) Sharp Changes in Behavior
The series displays **no abrupt changes**. The decline is smooth and consistent, with no sudden drops or spikes. The residuals also lack discontinuities, supporting a stable linear trend.

Stationary Models

A time series is **stationary** if its statistical properties (mean, variance, covariance) do not change over time. This simplifies modeling, as patterns remain consistent.

Mean Function:

Let $\{X_t\}$ be a time series with $E(X_t^2) < \infty$. The **mean function** of $\{X_t\}$ is

$$\mu_X(t) = E(X_t)$$

Covariance Function:

The **covariance function** of $\{X_t\}$ is

$$\gamma_X(r, s) = Cov(X_r, X_s) = E[(X_r - \mu_X(r))(X_s - \mu_X(s))]$$

for all integers **r** and **s**.

Weakly Stationary:

$\{X_t\}$ is **weakly stationary** if

(i) $\mu_X(t)$ is independent of t,

and

(ii) $\gamma_X(t + h, t)$ is independent of t for each h.

Strictly Stationary:

$\{X_t\}$ is said to be **strictly stationary** if the joint distribution of any set of observations $(X_{t_1}, X_{t_2}, \cdots, X_{t_n})$ is identical to the shifted set $(X_{t_1+h}, X_{t_2+h}, \cdots, X_{t_n+h})$ for all h.

Note:

Whenever we use the term stationary we shall mean **weakly stationary**, unless we specifically indicate otherwise.

Autocovariance Function (ACVF)

For a stationary time series $\{X_t\}$, the **autocovariance function (ACVF)** at lag h is defined as:

$$\gamma_X(h) = Cov(X_{t+h}, X_t)$$

It depends only on the *lag* h, not on the specific time t (due to stationarity). It measures the linear dependence between observations separated by h time units. At lag $h = 0$, $\gamma_X(0) = Var(X_t)$, which is the variance of the series.

Autocorrelation Function (ACF)

The **autocorrelation function (ACF)** normalizes the ACVF by the variance:

$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = Cor(X_{t+h}, X_t)$$

$\rho_X(h)$ is the correlation coefficient between X_{t+h} and X_t, ranging between -1 and 1. For example, If $\rho_X(1) = 0.8$ then observations one lag apart are strongly positively correlated.

Linearity Property of Covariances

For random variables X, Y, Z with finite variance and constants a, b, c :

$$Cov(aX + bY + c, Z) = a.Cov(X, Y) + b.Cov(Y, Z)$$

Examples:

IID Noise:

Given a sequence $\{X_t\}$ is **independent and identically distributed (IID) noise**, so X_t and X_s are independent for $t \neq s$ and all $\{X_t\}$ have the same probability distribution with **zero mean** and **finite variance**. Therefore,

$$E[X_t] = 0; \quad Var(X_t) = E[X_t^2] = \sigma^2 < \infty$$

So,

$$\mu_X(t) = E[X_t] = 0$$

and

$$\gamma_X(h) = Cov(X_{t+h}, X_t) = E[(X_{t+h} - \mu_X(t+h))(X_t - \mu_X(t))]$$

$$\gamma_X(h) = E[X_{t+h}.X_t]$$

If $h = 0$, then

$$\gamma_X(0) = E[X_t.X_t] = \sigma^2$$

If $h \neq 0$, then X_{t+h} and X_t are independent, hence

$$\gamma_X(h) = E[X_{t+h}].E[X_t] = 0.0 = 0$$

Therefore,

$$\gamma_X(h) = \begin{cases} \sigma^2 & \textbf{if } h = 0 \\ 0 & \textbf{if } h \neq 0 \end{cases}$$

So, $\gamma_X(h)$ does not depend on t, hence **IID Noise** is stationary. Therefore,

$$\rho_X(h) = \begin{cases} 1 & \textbf{if } h = 0 \\ 0 & \textbf{if } h \neq 0 \end{cases}$$

No autocorrelation (except at lag 0) makes it a "memoryless" process. This means IID noise has **no memory**: past values do not influence future values. Few examples of IID Noise are **White Noise (e.g., Gaussian white noise)** and **Fair coin tosses (heads = +1, tails = -1)**.

IID noise is the simplest stationary process. Many time series models (e.g., regression errors) assume residuals behave like IID noise. Real-world data often violates IID assumptions (e.g., trends, autocorrelation).

The Random Walk:

A **random walk** $\{S_t\}$ is constructed by cumulatively summing iid noise $\{X_t\}$ with $E[X_t] = 0$ and $Var(X_t) = \sigma^2$. For example, if X_t represents steps of $+1$ or -1 (as in a simple symmetric random walk), the position S_t at time t is the sum of all steps up to t:

$$S_t = X_1 + X_2 + \cdots + X_t, \quad S_0 = 0$$
$$E[S_t] = 0; \quad Var(S_t) = Var(X_1 + X_2 + \cdots + X_t) = t.\sigma^2 < \infty$$

So,

$$\mu_S(t) = E[S_t] = 0$$

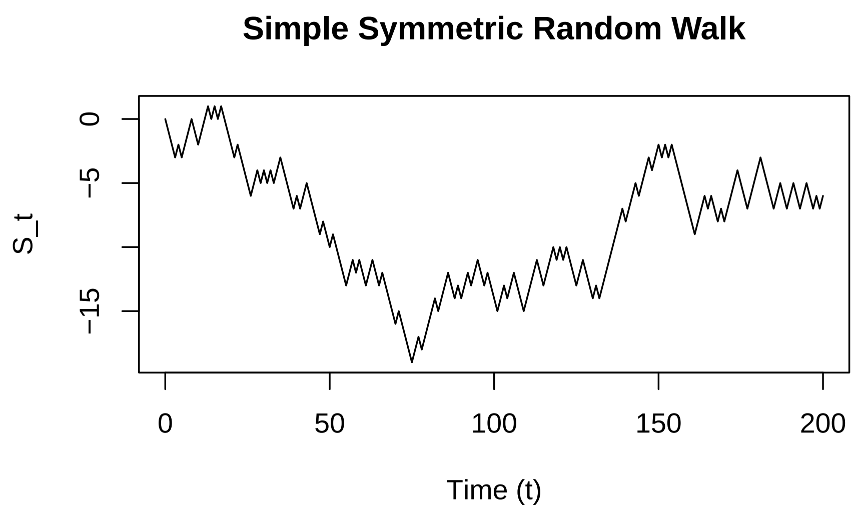
and

$$\gamma_S(h) = Cov(S_{t+h}, S_t) = Cov(S_t + X_{t+1} + X_{t+2} + \cdots + X_{t+h}, S_t)$$

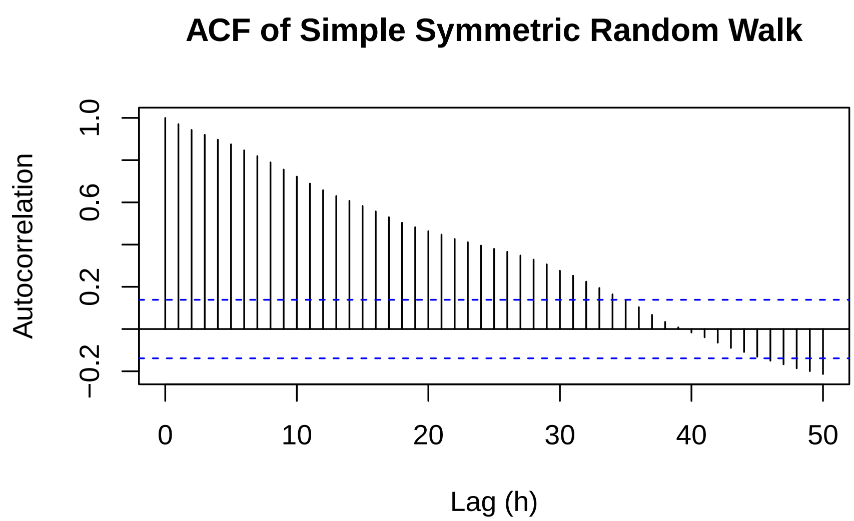
$$\gamma_S(h) = Cov(S_t, S_t) = Var(S_t) = t\sigma^2$$

Since ACVF depends on t, so a simple random walk is **non-stationary**.

```
# For reproducibility
set.seed(123)
steps <- sample(c(-1, 1), 200, replace = TRUE)
s <- c(0, cumsum(steps))  # s[1:201] contains S =0 to S
plot(0:200, s, type = "l",
     xlab = "Time (t)", ylab = "S_t",
     main = "Simple Symmetric Random Walk")
```




```r
# Compute ACF for the realization (excluding initial S =0)
acf(s[-1],
    main = "ACF of Simple Symmetric Random Walk",
    lag.max = 50,  # Show up to lag 50
    ylab = "Autocorrelation",
    xlab = "Lag (h)")
```



The autocorrelation remains significantly positive for many lags, reflecting the non-stationarity of the random walk. Each value S_t depends heavily on all prior values S_{t-1}, S_{t-2}, $\cdots$ leading to high autocorrelation at all lags.

First-order moving average MA(1) process:

The **first-order moving average (MA(1))** process is defined as:

$$X_t = Z_t + \theta.Z_{t-1}, \quad \{Z_t\} \sim WN(0, \sigma^2)$$

where $\{Z_t\}$ is white noise (uncorrelated, zero mean, variance σ^2) and θ is a constant.

$$\mu_X(t) = E[X_t] = E[Z_t] + \theta.E[Z_{t-1}] = 0$$

$$Var(X_t) = Var(Z_t) + \theta^2.Var(Z_{t-1}) = \sigma^2(1 + \theta^2)$$

So, the ACVF:

$$\gamma_X(h) = \begin{cases} \sigma^2(1+\theta^2) & \textbf{if } h = 0 \\ \sigma^2\theta & \textbf{if } h = \pm 1 \\ 0 & \textbf{if } |h| > 1 \end{cases}$$

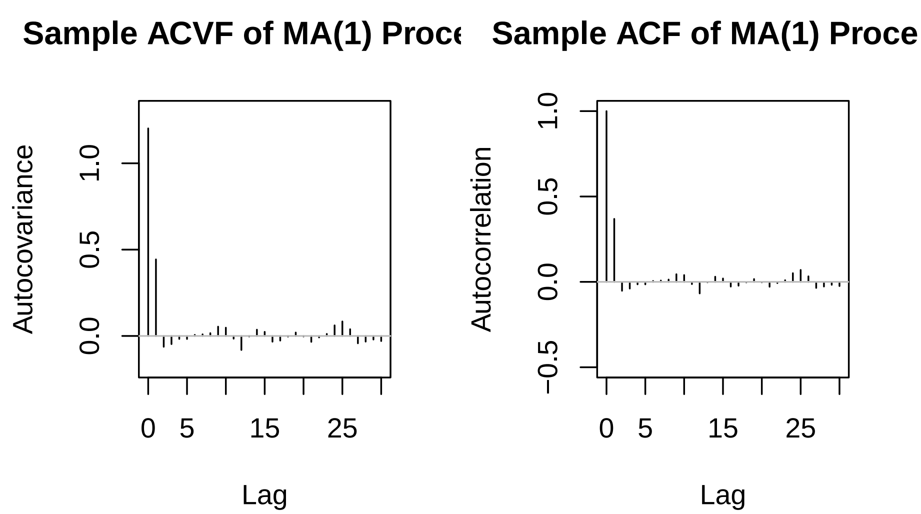
and, the ACF:

$$\rho_X(h) = \begin{cases} 1 & \textbf{if } h = 0 \\ \frac{\theta}{1+\theta^2} & \textbf{if } h = \pm 1 \\ 0 & \textbf{if } |h| > 1 \end{cases}$$

Hence, MA(1) is stationary.

```
# Load required package
library(stats)

# Simulate MA(1) process with   = 0.5
set.seed(123)
n <- 1000  # Number of observations
theta <- 0.5
sigma <- 1
ma1_process <- arima.sim(model = list(ma = theta), n = n, sd = sigma)

# Compute ACVF and ACF
acvf <- acf(ma1_process, type = "covariance", plot = FALSE)
```

```r
acf_values <- acf(ma1_process, plot = FALSE)

# Plot ACVF and ACF side by side
par(mfrow = c(1, 2))  # 1 row, 2 columns

# Plot ACVF
plot(acvf$lag, acvf$acf, type = "h",
     main = "Sample ACVF of MA(1) Process",
     xlab = "Lag", ylab = "Autocovariance",
     ylim = c(min(acvf$acf)-0.1, max(acvf$acf)+0.1))
abline(h = 0, col = "gray")

# Plot ACF
plot(acf_values$lag, acf_values$acf, type = "h",
     main = "Sample ACF of MA(1) Process",
     xlab = "Lag", ylab = "Autocorrelation",
     ylim = c(-0.5, 1))
abline(h = 0, col = "gray")
```



First-order autoregression or AR(1) process

The **AR(1) process** is a foundational time series model where each observation depends linearly on its immediate past value plus random noise.

$$X_t = \phi X_{t-1} + Z_t, \quad t = 0, \pm 1, \cdots,$$

$\{Z_t\} \sim WN(0, \sigma^2)$

where:

- ϕ is the autoregressive coefficient ($|\phi| < 1$ for stationarity).
- $\{Z_t\}$ is white noise (uncorrelated, zero mean, variance σ^2).

Computing expectation:

$$E[X_t] = E[\phi X_{t-1} + Z_t]$$

$$E[X_t] = \phi E[X_{t-1}] + E[Z_t]$$

Since $E[Z_t] = 0$,

$$E[X_t] = \phi E[X_{t-1}]$$

Let $E[X_t] = \mu$ and the process $\{X_t\}$ is stationary, so $E[X_t] = E[X_{t-1}]$:

$$\mu = \phi \mu$$

$$\mu(1 - \phi) = 0$$

Since $|\phi| < 1$:

$$\mu = 0$$

So, $E[X_t] = 0$.

Computing ACVF:

$$\gamma_X(h) = Cov(X_t, X_{t-h}) = Cov(\phi X_{t-1}, X_{t-h}) + Cov(Z_t, X_{t-h})$$

$$\gamma_X(h) = \phi \gamma_X(h-1) + 0 = \cdots = \phi^h \gamma_X(0)$$

$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = \phi^{|h|}, \quad h = 0, \pm 1, \cdots$$

Computing $\gamma_X(0)$:

$$\gamma_X(0) = Cov(X_t, X_t) = Cov(\phi X_{t-1} + Z_t, \phi X_{t-1} + Z_t)$$

$$\gamma_X(0) = \phi.Cov(X_{t-1}, \phi X_{t-1} + Z_t) + Cov(Z_t, \phi X_{t-1} + Z_t) = \phi^2.Cov(X_{t-1}, X_{t-1}) + Cov(Z_t, Z_t)$$

$$\gamma_X(0) = \phi^2 \gamma_X(0) + \sigma^2$$

$$\gamma_X(0) = \frac{\sigma^2}{1 - \phi^2}$$

Hence,

$$\gamma_X(h) = \frac{\sigma^2 \phi^{|h|}}{1 - \phi^2}$$

Since, ACVF is independent of t, hence $\{X_t\}$ is stationary.

Sample Autocorrelation Function

When analyzing a time series $\{x_1, x_2, \cdots, x_n\}$, one of the key tools is the **sample autocorrelation function (sample ACF)**. It provides insights into the correlation of the time series with itself at different time lags.

Sample Mean:

Given observations $\{x_1, x_2, \cdots, x_n\}$, the **sample mean** is:

$$\overline{x} = \frac{1}{n} \sum_{t=1}^{n} x_t$$

This is just the average of all observed values and is used when centering the data (subtracting the mean) before computing the sample autocovariance and autocorrelation.

Sample Autocovariance Function:

The **sample autocovariance** at lag h (denoted $\hat{\gamma}(h)$) is defined by:

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-|h|} (x_{t+|h|} - \overline{x})(x_t - \overline{x}), \quad -n < h < n$$

and $\hat{\gamma}(h) = \hat{\gamma}(-h)$ for negative h.

Autocovariance measures how the time series covaries with itself at different lags. For example, $\hat{\gamma}(1)$ roughly measures how x_t relates to x_{t+1}, $\hat{\gamma}(2)$ relates x_t to x_{t+2}, etc.

We define **sample covariance matrix** as $\hat{\Gamma}n := [\hat{\gamma}(i-j)]_{i,j=1}^n$ and using n as divisor ensures that this matrix is **non-negative definite**.

Sample Autocorrelation Function:

The **sample autocorrelation function** at lag h (denoted $\hat{\rho}(h)$) is:

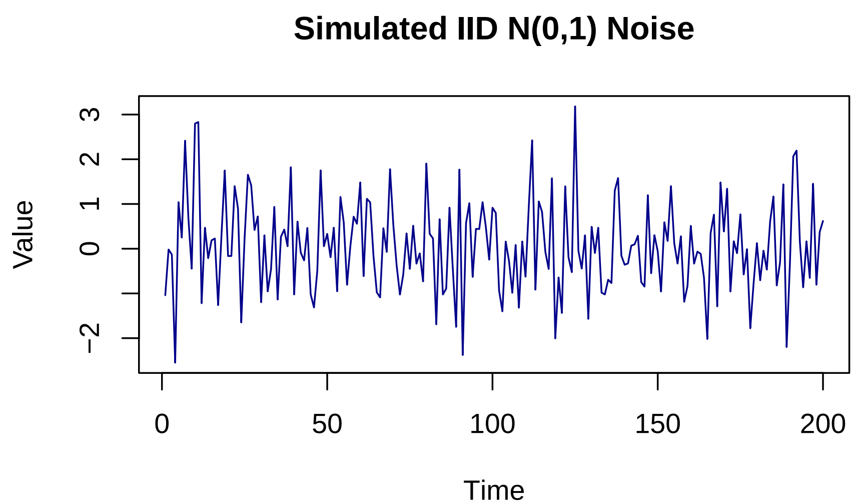
$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}$$

We define **sample correlation matrix** as $\hat{R}n := [\hat{\rho}(i-j)]_{i,j=1}^n$ and it is also **non-negative definite**. Each of its diagonal elements is **1**.

Illustration of Sample ACF through IID N(0,1) noise

We assume $X_t \sim N(0,1)$, i.e., Gaussian white noise with mean 0 and variance 1. We first simulate 200 values of $N(0,1)$ noise.

```r
# Set seed for reproducibility

# Generate 200 observations of IID N(0,1) noise
n <- 200
iid_noise <- rnorm(n, mean = 0, sd = 1)

# Plot the simulated series (Figure 1.12)
plot.ts(iid_noise,
        main = "Simulated IID N(0,1) Noise",
        ylab = "Value",
        col = "darkblue")
```



Computing **sample mean**, **sample ACF** using above definitions.

$$\bar{X} = \frac{\sum_{t=1}^{200} X_t}{200}$$

$$\hat{\rho}(h) = \frac{\sum_{t=1}^{200-|h|}(X_{t+|h|} - \overline{X})(X_t - \overline{X})}{\sum_{t=1}^{200}(X_t - \bar{X})^2}$$

```
# Sample mean
n<-200
x_bar <- sum(iid_noise) / n
cat("Sample Mean =",x_bar, "\n")
```

```
Sample Mean = 0.05846075
```

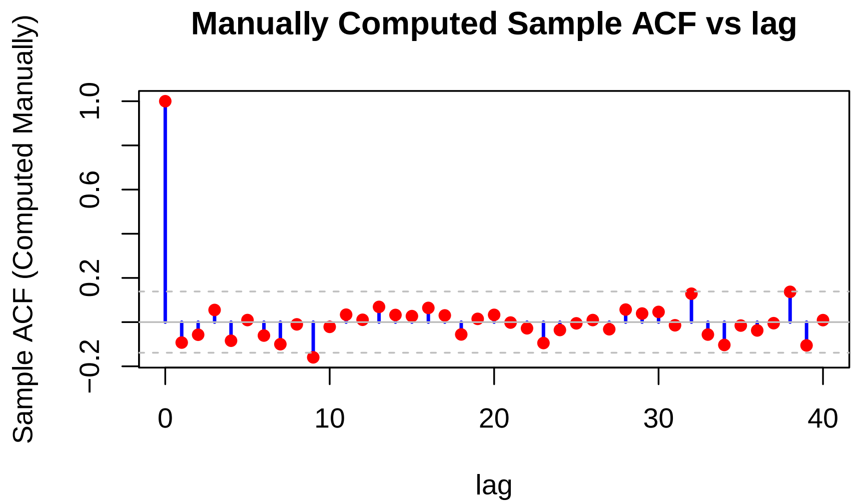
```
acvf <- function(h) {
  sum((iid_noise[(1 + h) : (n - h) ] - x_bar)*(iid_noise[1: (n - h) ]-x_bar)) / n
}
lag <- 0:40
acfs_values <- sapply(lag, acvf)
```

```
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
```

```
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
```

```
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
Warning in (iid_noise[(1 + h):(n - h)] - x_bar) * (iid_noise[1:(n - h)] - :
longer object length is not a multiple of shorter object length
```

```r
acfs_manual <- acfs_values/acfs_values[1]

# Create the plot using type "h" for vertical lines.
plot(lag, acfs_manual, type = "h", lwd = 2, col = "blue",
     xlab = "lag", ylab = "Sample ACF (Computed Manually)",
     main = "Manually Computed Sample ACF vs lag")
points(lag, acfs_manual, pch = 16, col = "red")
abline(h = 0, col = "gray")

# Add Confidence Bounds AFTER the plot is created.
conf_bound <- 1.96 / sqrt(n)
abline(h = c(-conf_bound, conf_bound), col = "gray", lty = 2)
```



For large number of simulations n, $X \approx 0$. Hence,

$$\hat{\rho}(h) \approx \frac{\sum_{t=1}^{n-|h|} X_{t+|h|}X_t}{\sum_{t=1}^{n}(X_t - \bar{X})^2}$$

$$\hat{\rho}(h) \approx \frac{1}{n\sigma^2}\sum_{t=1}^{n-|h|} X_{t+|h|}X_t$$

Now, we will look at the $E[\hat{\rho}(h)]$ and $Var(\hat{\rho}(h))$:

We will use the fact that X_{t+h} and X_t are independent.

$$E[\hat{\rho}(h)] \approx \frac{1}{n\sigma^2}\sum_{t=1}^{n-|h|} E[X_{t+h}X_t] = 0$$

$$Var(\hat{\rho}(h)) \approx \frac{1}{n^2\sigma^4}\sum_{t=1}^{n-|h|} Var(X_{t+h}X_t)$$

$$Var(X_{t+h}X_t) = E[(X_{t+h}X_t)^2] - (E[X_{t+h}X_t])^2 = E[X_{t+h}^2 X_t^2] - 0$$

$$Var(X_{t+h}X_t) = E[X_{t+h}^2]E[X_t^2] = (Var(X_{t+h}){+}(E[X_{t+h}])^2).(Var(X_t){+}(E[X_t])^2)$$

$$Var(X_{t+h}, X_t) = \sigma^4$$

Hence,

$$Var(\hat{\rho}(h)) \approx \frac{1}{n^2\sigma^4} n\sigma^4 \approx \frac{1}{n}$$

By the **Central Limit Theorem (CLT)**, the $\hat{\rho}(h)$ converges to a normal distribution for large n. Thus, for IID noise, the sample ACF at any *lag* $h \neq 0$ is approximately:

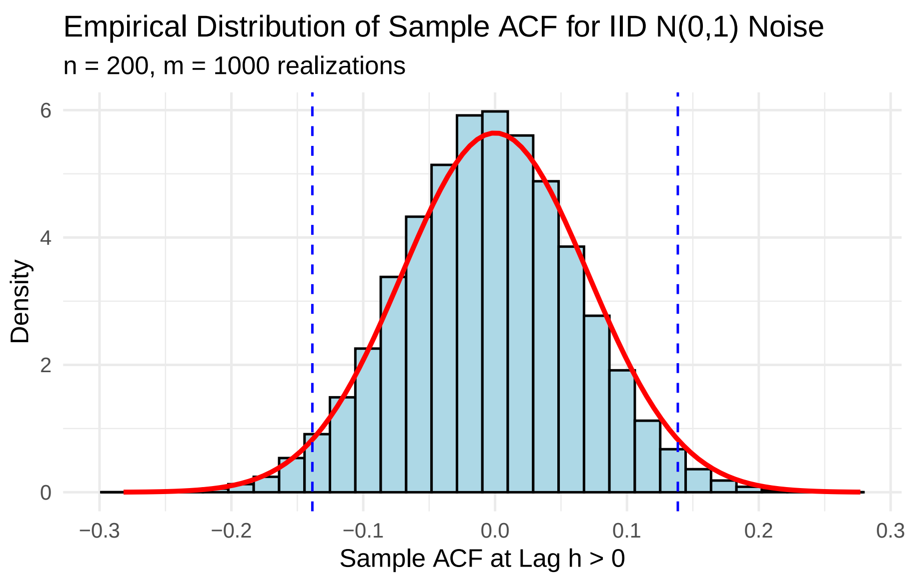
$$\hat{\rho}(h) \sim N(0, \frac{1}{n})$$

Verification: We first simulate m independent datasets of length n from an IID $N(0,1)$ process. Large m ensures the Central Limit Theorem (CLT) applies. For each dataset, calculate the sample ACF $\hat{\rho}(h)$ at lags $h = 1, 2, \cdots, H$. Aggregate all $\hat{\rho}(h)$ values across lags $(h > 0)$ and datasets. This creates an empirical distribution of sample ACFs. Now, we can plot a histogram of the pooled ACF values. This should overlay the theoretical normal density $N(0, \frac{1}{n})$ and 95% confidence bounds $\pm 1.96/\sqrt{n}$.

```r
# Load required package
library(ggplot2)

# Set parameters
set.seed(123)     # Reproducibility
n <- 200          # Length of each time series
m <- 1000         # Number of realizations (large for CLT)
H <- 40           # Maximum lag
conf_level <- 0.95

# Simulate m realizations of IID N(0,1) noise
acf_values <- matrix(NA, nrow = m, ncol = H)

for (k in 1:m) {
  # Generate IID N(0,1) noise
  noise <- rnorm(n, mean = 0, sd = 1)

  # Compute sample ACF up to lag H
  acf_result <- acf(noise, lag.max = H, plot = FALSE)$acf

  # Store ACF values at lags 1 to H
  acf_values[k, ] <- acf_result[2:(H + 1)]
}
```

```r
# Pool all ACF values (across lags and realizations)
pooled_acf <- as.vector(acf_values)

# Theoretical parameters under H0 (IID noise)
mu <- 0
sigma <- 1 / sqrt(n)

# Compute proportion of values outside confidence bounds
conf_bound <- qnorm((1 + conf_level)/2) * sigma
out_of_bounds <- mean(abs(pooled_acf) > conf_bound)

# Create histogram with theoretical normal curve
ggplot(data.frame(x = pooled_acf), aes(x = x)) +
  geom_histogram(aes(y = ..density..),
                 bins = 30,
                 fill = "lightblue",
                 color = "black") +
  stat_function(fun = dnorm,
                args = list(mean = mu, sd = sigma),
                color = "red",
                linewidth = 1) +
  geom_vline(xintercept = c(-conf_bound, conf_bound),
             color = "blue",
             linetype = "dashed") +
  labs(title = "Empirical Distribution of Sample ACF for IID N(0,1) Noise",
       subtitle = sprintf("n = %d, m = %d realizations", n, m),
       x = "Sample ACF at Lag h > 0",
       y = "Density") +
  theme_minimal()
```

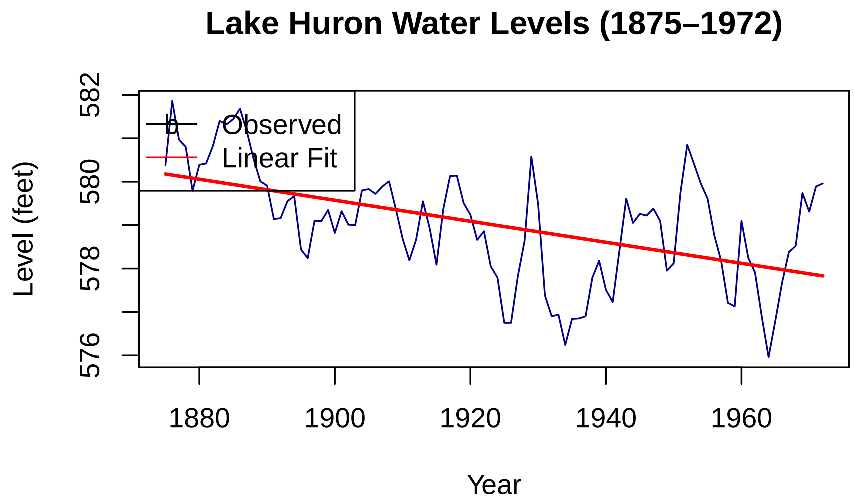
```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```



Lake Huron Residuals Modeling Process

Earlier, the Lake Huron water level data (1875–1972) was analyzed to identify a suitable time series model:

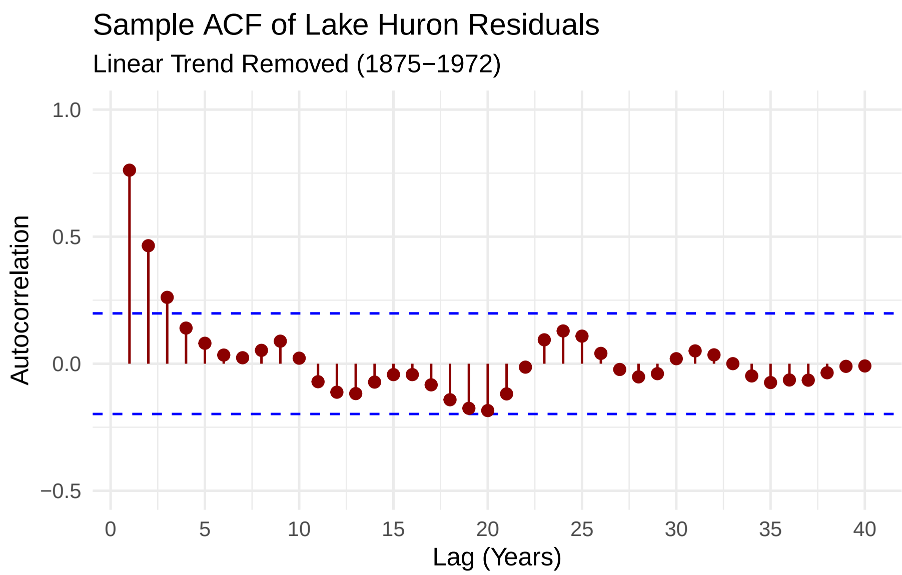
```r
# Load necessary packages
library(ggplot2)
lake_data <- read.csv("LakeHuron.csv")
lake_data$t <- lake_data$year - 1875  # Create time variable starting at 0

# Fit linear model
model <- lm(level ~ t, data = lake_data)
lake_data$fitted <- predict(model)
# Base R plot version
plot(lake_ts,
     main = "Lake Huron Water Levels (1875-1972)",
     xlab = "Year",
     ylab = "Level (feet)",
     col = "darkblue",pch = 98)
lines(lake_data$year, lake_data$fitted, col = "red", lwd = 2)
legend("topleft",
       legend = c("Observed", "Linear Fit"),
       col = c("black", "red"),
       lty = 1, pch = c(98, NA))
```



We can see that our initial model constitutes of a straight line fitted to the data (linear trend: $\hat{m}_t = a + bt$). We then analyzed the residuals $\{y_1, y_2, \cdots, y_{98}\}$. We now compute Sample ACF for the Lake Huron Residuals:

```
lake_data$residuals <- residuals(model)  # Store residuals

# Compute sample ACF of residuals
acf_res <- acf(lake_data$residuals, plot = FALSE, lag.max = 40)

# Convert ACF results to dataframe for ggplot
acf_df <- with(acf_res, data.frame(lag, acf))

# Create confidence bounds (95% CI)
n <- nrow(lake_data)
conf_level <- 1.96/sqrt(n)

# Plot ACF with ggplot
ggplot(acf_df[-1,], aes(x = lag, y = acf)) +  # [-1,] removes lag 0
  geom_hline(yintercept = c(-conf_level, conf_level),
             color = "blue", linetype = "dashed") +
  geom_segment(aes(xend = lag, yend = 0), color = "darkred") +
  geom_point(color = "darkred", size = 2) +
  labs(title = "Sample ACF of Lake Huron Residuals",
       subtitle = "Linear Trend Removed (1875-1972)",
       x = "Lag (Years)",
       y = "Autocorrelation") +
  theme_minimal() +
```

```r
  scale_x_continuous(breaks = seq(0, 40, 5)) +
  ylim(c(-0.5, 1))
```




```r
r1 <- acf_df[1,2]
r2 <- acf_df[2,2]
r3 <- acf_df[3,2]
r4 <- acf_df[4,2]
gradual_Decay <- list()
gradual_Decay[1] <- r2 / r1
gradual_Decay[2] <- r3 / r2
gradual_Decay[3] <- r4 / r3
gradual_Decay
```

```
[[1]]
[1] 0.7615963

[[2]]
[1] 0.6097112

[[3]]
[1] 0.5622723
```

```r
r1
```

```
[1] 1
```

We can see, after fitting a linear trend, the residuals $\{y_t\}$ violated the IID noise assumption. The ACF showed a gradual decay, with $\frac{\hat{\rho}(h+1)}{\hat{\rho}(h)} \approx 0.7$, characteristic of autoregressive (AR) processes. The geometric decay $\hat{\rho}(h) \approx \phi^{|h|}$ suggested an **AR(1) process**. The decay rate $\phi \approx 0.7$ was inferred from $\hat{\rho}(1) \approx 0.7$.

```r
# Fit AR(1) to residuals
ar1_model <- arima(lake_data$residuals, order = c(1,0,0), include.mean = FALSE)
# Model summary
cat("AR(1) Model Summary:\n")
```

```
AR(1) Model Summary:
```

```r
print(ar1_model)
```

```
Call:
arima(x = lake_data$residuals, order = c(1, 0, 0), include.mean = FALSE)

Coefficients:
         ar1
      0.7826
s.e.  0.0635

sigma^2 estimated as 0.4975:  log likelihood = -105.32,  aic = 214.65
```

```r
cat("\nPhi estimate:", ar1_model$coef, "\n")
```

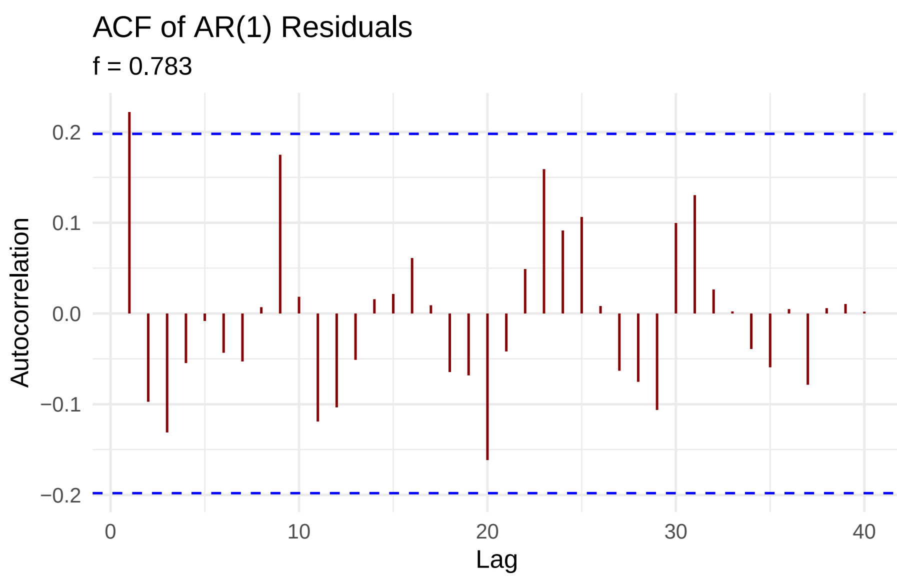
```
Phi estimate: 0.7826118
```

```r
cat("Noise variance:", ar1_model$sigma2, "\n")
```

```
Noise variance: 0.4975348
```

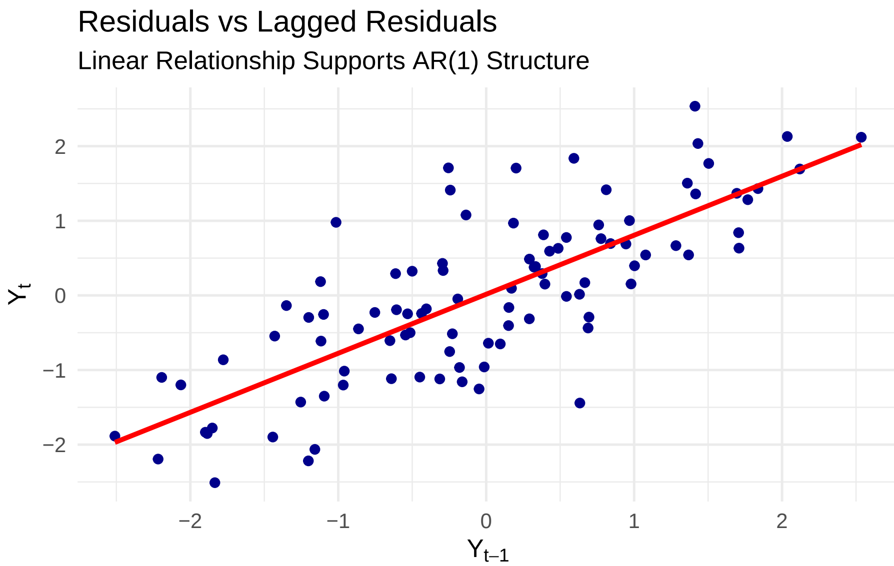
```r
# Get AR(1) residuals
ar1_residuals <- residuals(ar1_model)

# Plot 1: ACF of AR(1) residuals
acf_ar1 <- acf(ar1_residuals, plot = FALSE, lag.max = 40)
conf_bound <- 1.96/sqrt(length(ar1_residuals))

ggplot(data.frame(lag = acf_ar1$lag, acf = acf_ar1$acf)[-1,],
```

```
      aes(x = lag, y = acf)) +
  geom_hline(yintercept = c(-conf_bound, conf_bound),
             color = "blue", linetype = "dashed") +
  geom_segment(aes(xend = lag, yend = 0), color = "darkred") +
  labs(title = "ACF of AR(1) Residuals",
       subtitle = paste("  =", round(ar1_model$coef, 3)),
       x = "Lag", y = "Autocorrelation") +
  theme_minimal()
```




```
# Plot 2: Scatterplot of Y_t vs Y_{t-1}
lake_data$lag1_residual <- c(NA, lake_data$residuals[-nrow(lake_data)])

ggplot(lake_data, aes(x = lag1_residual, y = residuals)) +
  geom_point(color = "darkblue") +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red") +
  labs(title = "Residuals vs Lagged Residuals",
       subtitle = "Linear Relationship Supports AR(1) Structure",
       x = expression(Y[t-1]),
       y = expression(Y[t])) +
  theme_minimal()
```

```
Warning: Removed 1 row containing non-finite outside the scale range
(`stat_smooth()`).

Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```



The linear relationship between Y_t and Y_{t-1} in the above figure supported the AR(1) structure $Y_t = \phi Y_{t-1} + Z_t$. Least squares regression of Y_t on Y_{t-1} gave $\hat{\phi} = 0.7826$. Residual variance ($\sigma^2 = 0.4975$) was computed, but the residuals still showed slight autocorrelation.

Hence, the fitted model:

$$Y_t = 0.78Y_{t-1} + Z_t, \quad Z_t \sim IID(0, 0.50)$$

Residual ACF of **AR(1)** showed slight excess correlation at lag 1 ($\hat{\rho}(1)$ is outside the bound only). This hinted that the AR(1) model might not fully capture dependencies. So, we can go for another model.

A better fit could be done by using **AR(2) process**. The **AR(1)** residuals had a sample ACF value outside bounds at lag 1, indicating unresolved dependence. This suggested the need for a higher-order model.

```
# Fit AR(2) to residuals
ar2_model <- arima(lake_data$residuals, order = c(2,0,0), include.mean = FALSE)
# Model summary
cat("AR(2) Model Summary:\n")
```

```
AR(2) Model Summary:
```

```
print(ar2_model)
```

```
Call:
arima(x = lake_data$residuals, order = c(2, 0, 0), include.mean = FALSE)

Coefficients:
         ar1      ar2
      1.0050  -0.2925
s.e.  0.0976   0.1002

sigma^2 estimated as 0.4572:  log likelihood = -101.26,  aic = 208.51
```

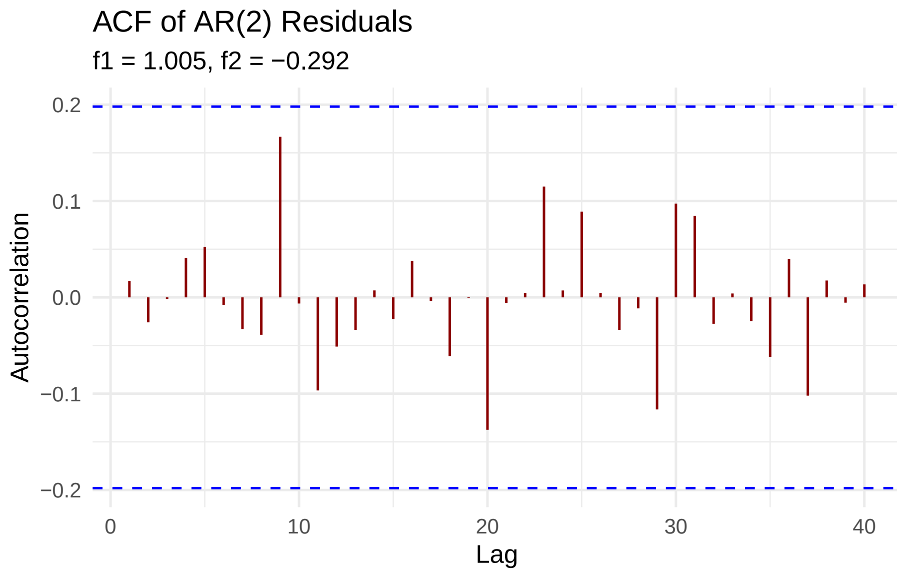
```
cat("\nPhi estimate:", ar2_model$coef, "\n")
```

```
Phi estimate: 1.005013 -0.2924751
```

```
cat("Noise variance:", ar2_model$sigma2, "\n")
```

```
Noise variance: 0.4571513
```

```
# Get AR(1) residuals
ar2_residuals <- residuals(ar2_model)

# Plot 1: ACF of AR(1) residuals
acf_ar2 <- acf(ar2_residuals, plot = FALSE, lag.max = 40)
conf_bound <- 1.96/sqrt(length(ar2_residuals))

ggplot(data.frame(lag = acf_ar2$lag, acf = acf_ar2$acf)[-1,],
       aes(x = lag, y = acf)) +
  geom_hline(yintercept = c(-conf_bound, conf_bound),
             color = "blue", linetype = "dashed") +
  geom_segment(aes(xend = lag, yend = 0), color = "darkred") +
  labs(title = "ACF of AR(2) Residuals",
       subtitle = sprintf(" 1 = %.3f,  2 = %.3f",
                          ar2_model$coef[1],
                          ar2_model$coef[2]),
       x = "Lag", y = "Autocorrelation") +
  theme_minimal()
```



We found residual variance ($\sigma^2 = 0.4571$) to be lesser than before and also from the ACF, we can see no autocorrelation. The fitted model:

$$Y_t = 1.005Y_{t-1} - 0.292Y_{t-1} + Z_t, \quad Z_t \sim IID(0, 0.457)$$

$\phi_1 = 1.005$ indicates strong persistence from the immediate past and $\phi_2 = -0.292$ indicates negative correction for over-persistence, introducing mean reversion. All lags within confidence bounds confirmed noise independence.

The Lake Huron residuals were best modeled by an AR(2) process, which reduced residual variance by 11.8% compared to AR(1) and also eliminated significant autocorrelation in residuals.

Estimation and Elimination of Trend and Seasonal Components

The first step in time series analysis is **visual inspection** of the data to identify structural breaks, outliers, or patterns. If trends or seasonality are present, the **classical decomposition model** is often applied:

$$X_t = m_t + s_t + Y_t$$

where m_t is a slowly varying **trend component**, s_t is a periodic **seasonal component** with known period d, and Y_t represents stationary **noise**. This