# Queuing Theory and Traffic Analysis

CS 552

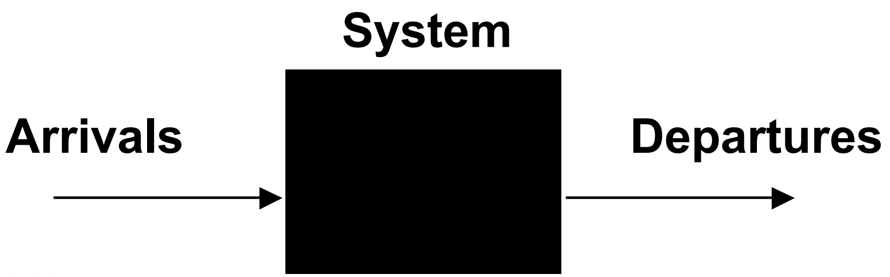
Richard Martin

Rutgers University

# Queuing theory

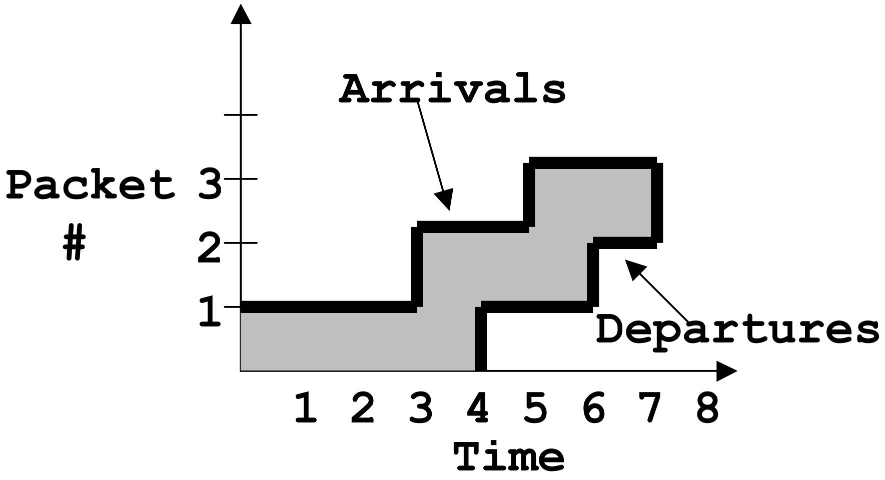
- View network as collections of queues
  - FIFO data-structures
- Queuing theory provides probabilistic analysis of these queues
- Examples:
  - Average length
  - Probability queue is at a certain length
  - Probability a packet will be lost

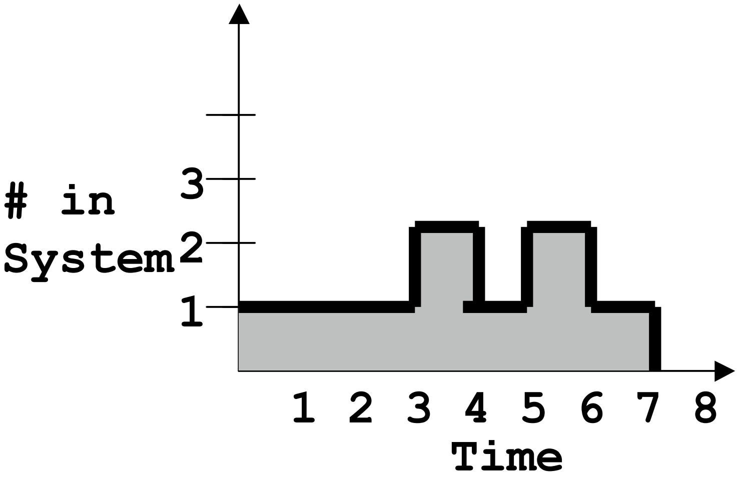
# Little’s Law

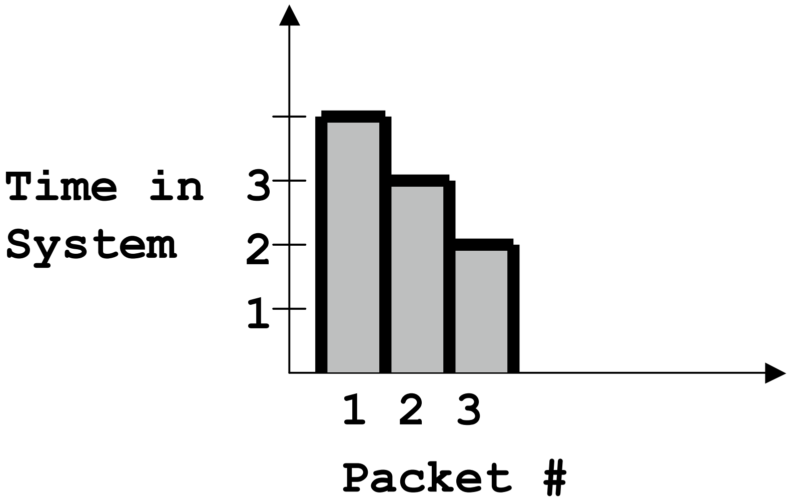


- Little’s Law:
  Mean number tasks in system = arrival rate x mean response time
  - Observed before, Little was first to prove
- Applies to any system in equilibrium,
  as long as nothing in black box
  is creating or destroying tasks

# Proving Little’s Law







J = Shaded area = 9

Same in all cases!

# Definitions

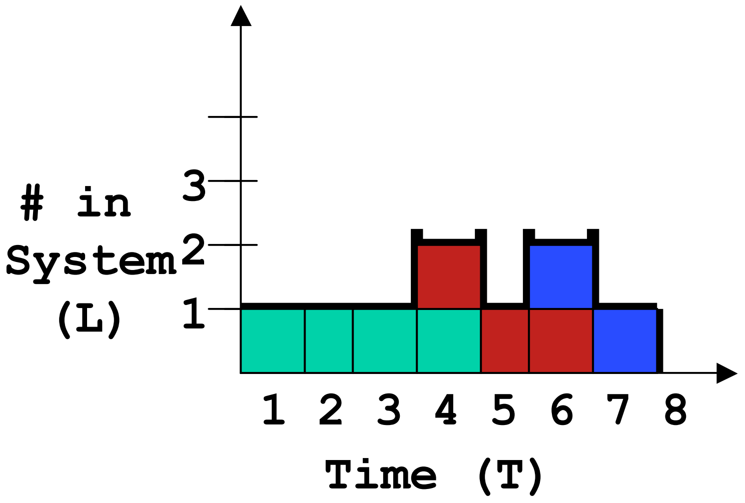
- J: “Area” from previous slide
- N: Number of jobs (packets)
- T: Total time

□ □: Average arrival rate

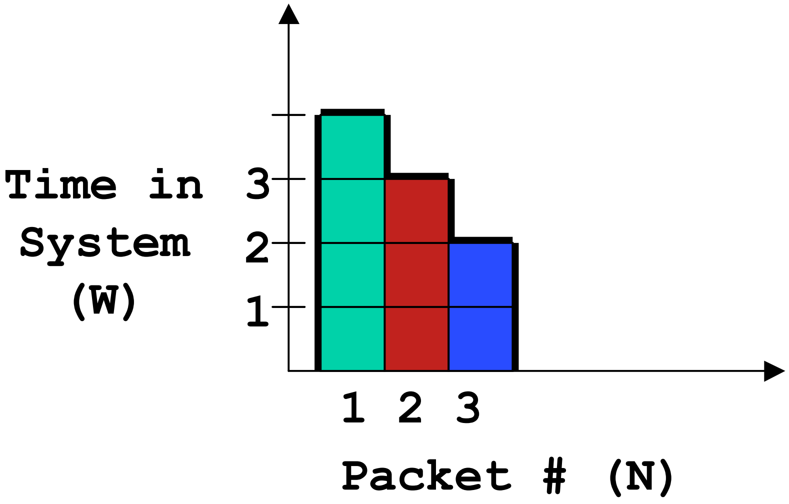
  - N/T

- W: Average time job is in the system
  - = J/N
- L: Average number of jobs in the system
  - = J/T

# Proof: Method 1: Definition



=



$$J = TL = NW$$

$$L = (\tfrac{N}{T})W$$

$$L = (\lambda)W$$

# Proof: Method 2: Substitution

$$L = (\square)W$$

$$L = \left(\tfrac{N}{T}\right)W$$

$$\frac{J}{T} = \left(\frac{N}{T}\right)\left(\frac{J}{N}\right)$$

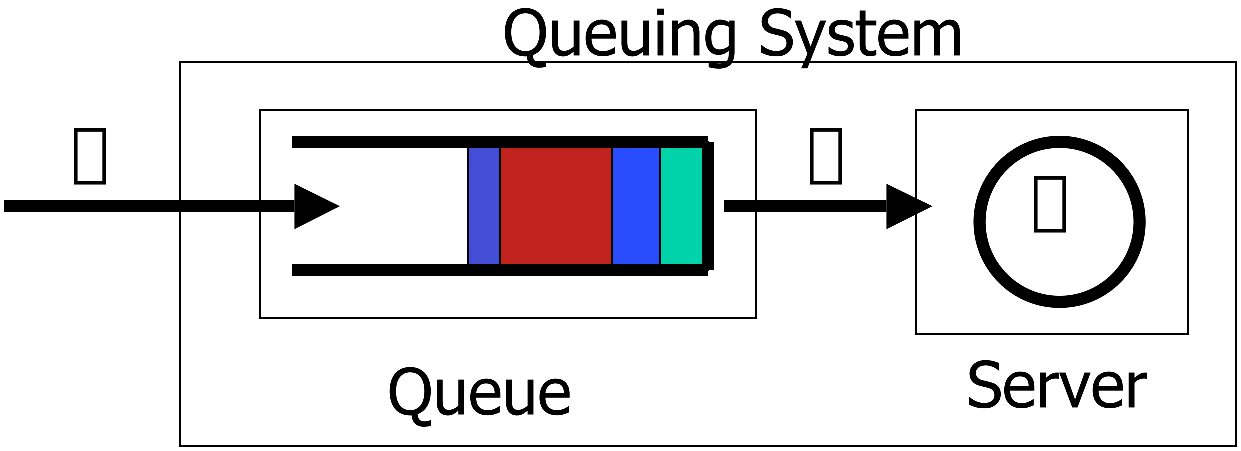
$$\frac{J}{T} = \frac{J}{T}$$ Tautology

# Example using Little's law

- Observe 120 cars in front of the Lincoln Tunnel
    - Observe 32 cars/minute depart over a period where no cars in the tunnel at the start or end (e.g. security checks)
- What is average waiting time before and in the tunnel?

$$W = \tfrac{L}{\lambda} = \left(\tfrac{120}{32}\right) = 3.75\text{min}$$

# Model Queuing System



Queuing System

Server System

Strategy:

Use Little’s law on both the complete system and its parts to reason about average time in the queue

# Kendal Notation

- Six parameters in shorthand
  - First three typically used, unless specified

1. Arrival Distribution
   - Probability of a new packet arrives in time t
2. Service Distribution
   - Probability distribution packet is serviced in time t
3. Number of servers
4. Total Capacity (infinite if not specified)
5. Population Size (infinite)
6. Service Discipline (FCFS/FIFO)

# Distributions

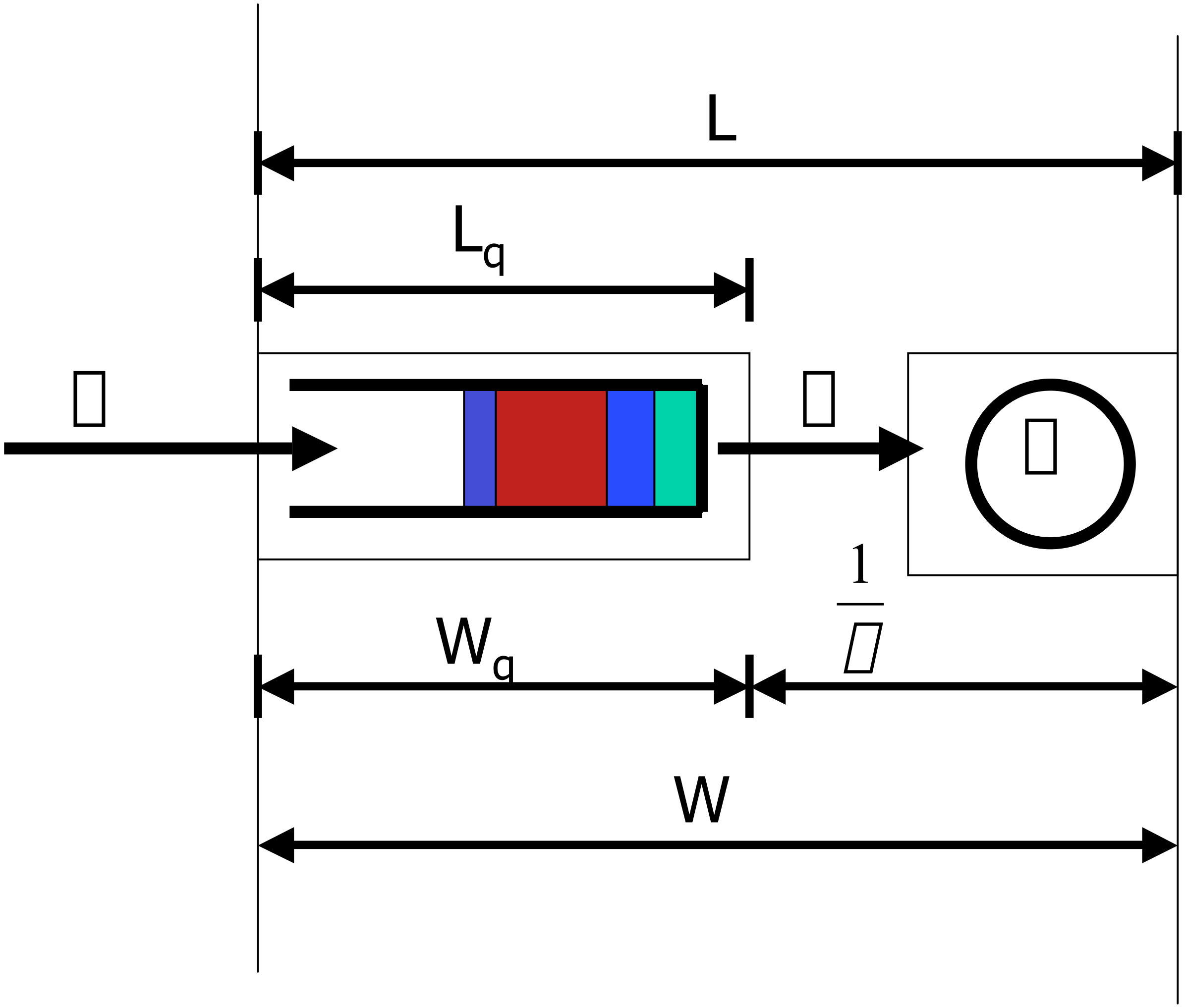
- M: Exponential
- D: Deterministic (e.g. fixed constant)
- $E_k$: Erlang with parameter *k*
- $H_k$: Hyperexponential with param. *k*
- G: General (anything)

- M/M/1 is the simplest ‘realistic’ queue

# Kendal Notation Examples

- M/M/1:
  - Exponential arrivals and service, 1 server, infinite capacity and population, FCFS (FIFO)
- M/M/m
  - Same, but M servers
- G/G/3/20/1500/SPF
  - General arrival and service distributions, 3 servers, 17 queue slots (20-3), 1500 total jobs, Shortest Packet First

# M/M/1 queue model



# Analysis of M/M/1 queue

- Goal: A closed form expression of the probability of the number of jobs in the queue ($P_i$) given only □ and □

# Solving queuing systems

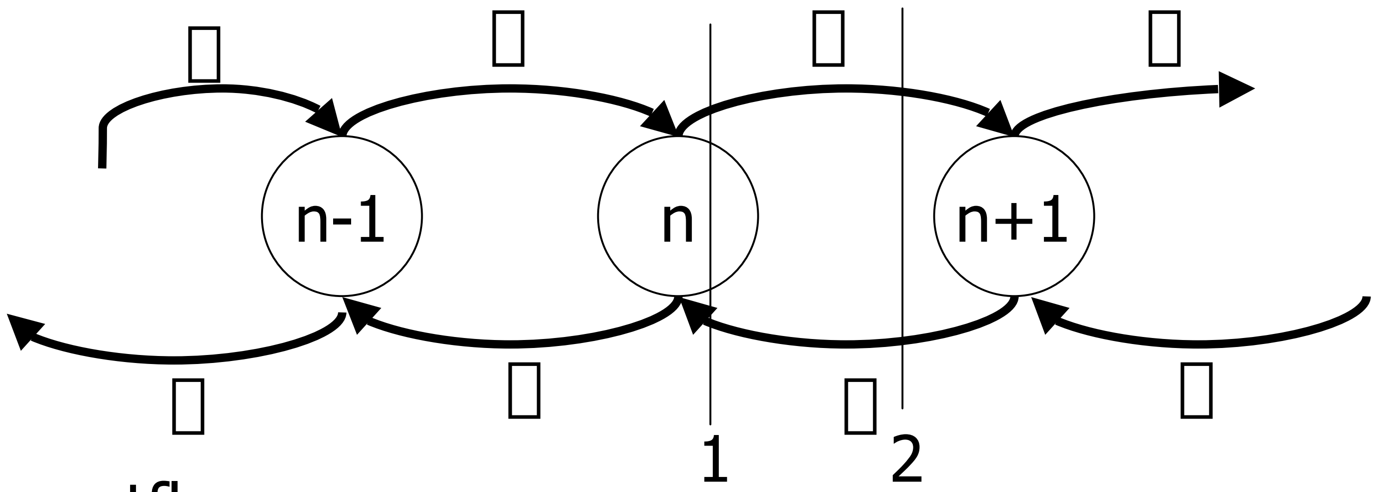
- Given:
  - □: Arrival rate of jobs (packets on input link)
  - □: Service rate of the server (output link)
- Solve:
  - L: average number in queuing system
  - $L_q$ average number in the queue
  - W: average waiting time in whole system
  - $W_q$ average waiting time in the queue
- 4 unknown’s: need 4 equations

# Solving queuing systems

- 4 unknowns: L, $L_q$ W, $W_q$
- Relationships using Little's law:
  - $L = \square W$
  - $L_q = \square W_q$ (steady-state argument)
  - $W = W_q + (1/\square)$
- If we know any 1, can find the others
- Finding L is hard or easy depending on the type of system. In general:

$$L = \underset{n=0}{\square} nP_n$$

# Equilibrium conditions



inflow = outflow

1: $(\lambda + \mu) P_n = \lambda P_{n-1} + \mu P_{n+1}$

2: $\lambda P_n = \mu P_{n+1}$

stability: 3: $\lambda < \mu, \rho = \frac{\lambda}{\mu}, \rho < 1$

# Solving for $P_0$ and $P_n$

1: $P_1 = \rho P_0\ ,\ P_2 = (\rho)^2 P_0\ ,\ P_n = (\rho)^n P_0$ ,

2: $\sum_{n=0} P_n = 1\ ,\ P_0 \sum_{n=0} \rho^n = 1\ ,\ P_0 = \frac{1}{\sum_{n=0} \rho^n}$

3: $\sum_{n=0} \rho^n = \frac{1}{1-\rho}, \rho < 1$ (geometric series)

4: $P_0 = \frac{1}{\sum_{n=0} \rho^n} = \frac{1}{\frac{1}{(1-\rho)}} = 1-\rho$ 5: $P_n = (\rho)^n (1-\rho)$

# Solving for L

$$L = \sum_{n=0} nP_n = \sum_{n=0} n\rho^n(1-\rho) = (1-\rho)\rho\sum_{n=1} n\rho^{n-1}$$

$$(1-\rho)\rho\frac{d}{d\rho}\left[\sum_{n=0}^{1}\rho^n\right] = (1-\rho)\rho\frac{d}{d\rho}\left(\frac{1}{1-\rho}\right)$$

$$(1-\rho)\rho\left(\frac{1}{(1-\rho)^2}\right) = \frac{\rho}{(1-\rho)} = \frac{\lambda}{\mu-\lambda}$$

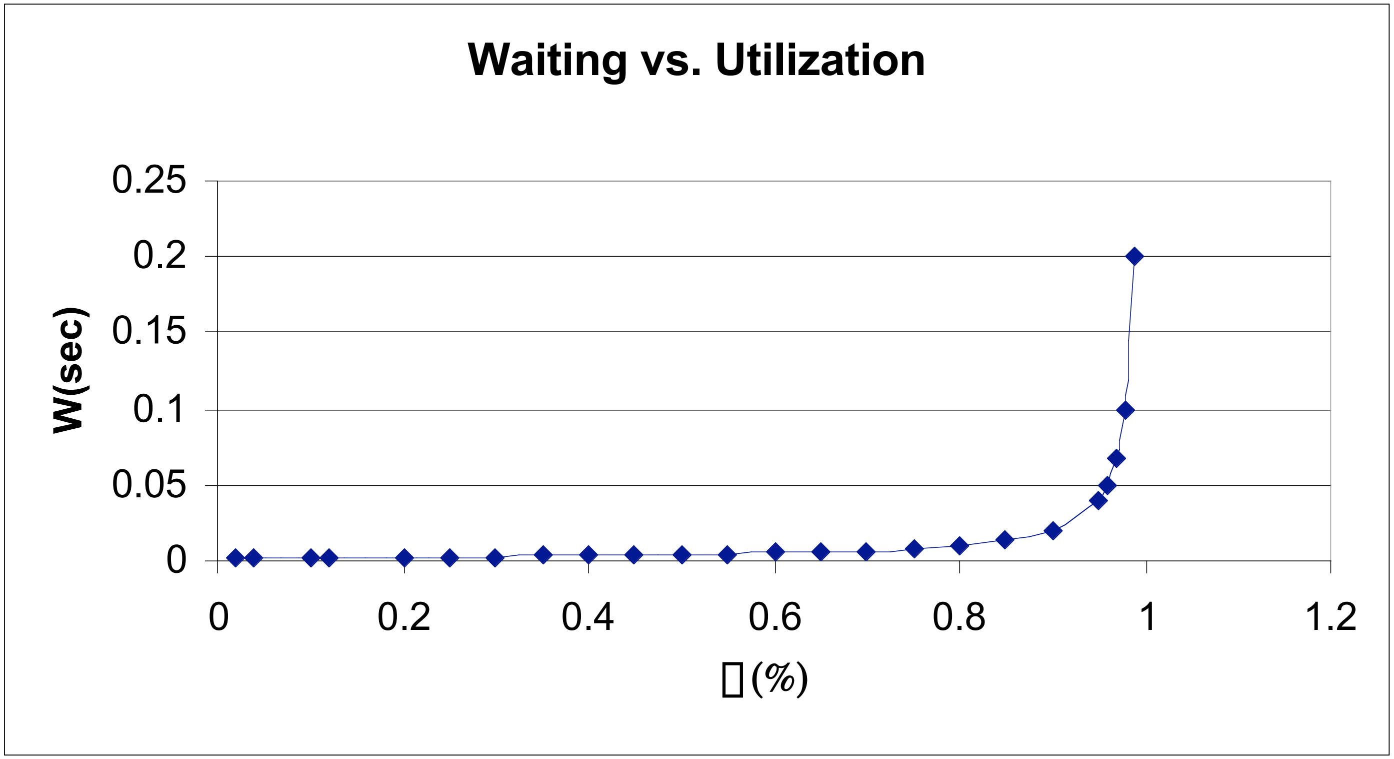
# Solving W, $W_q$ and $L_q$

$$W = \frac{L}{\lambda} = \left(\frac{\lambda}{\mu - \lambda}\right)\left(\frac{1}{\lambda}\right) = \frac{1}{\mu - \lambda}$$

$$W_q = W - \frac{1}{\mu} = \left(\frac{\lambda}{\mu - \lambda}\right) - \left(\frac{1}{\mu}\right) = \frac{\lambda}{\mu(\mu - \lambda)}$$

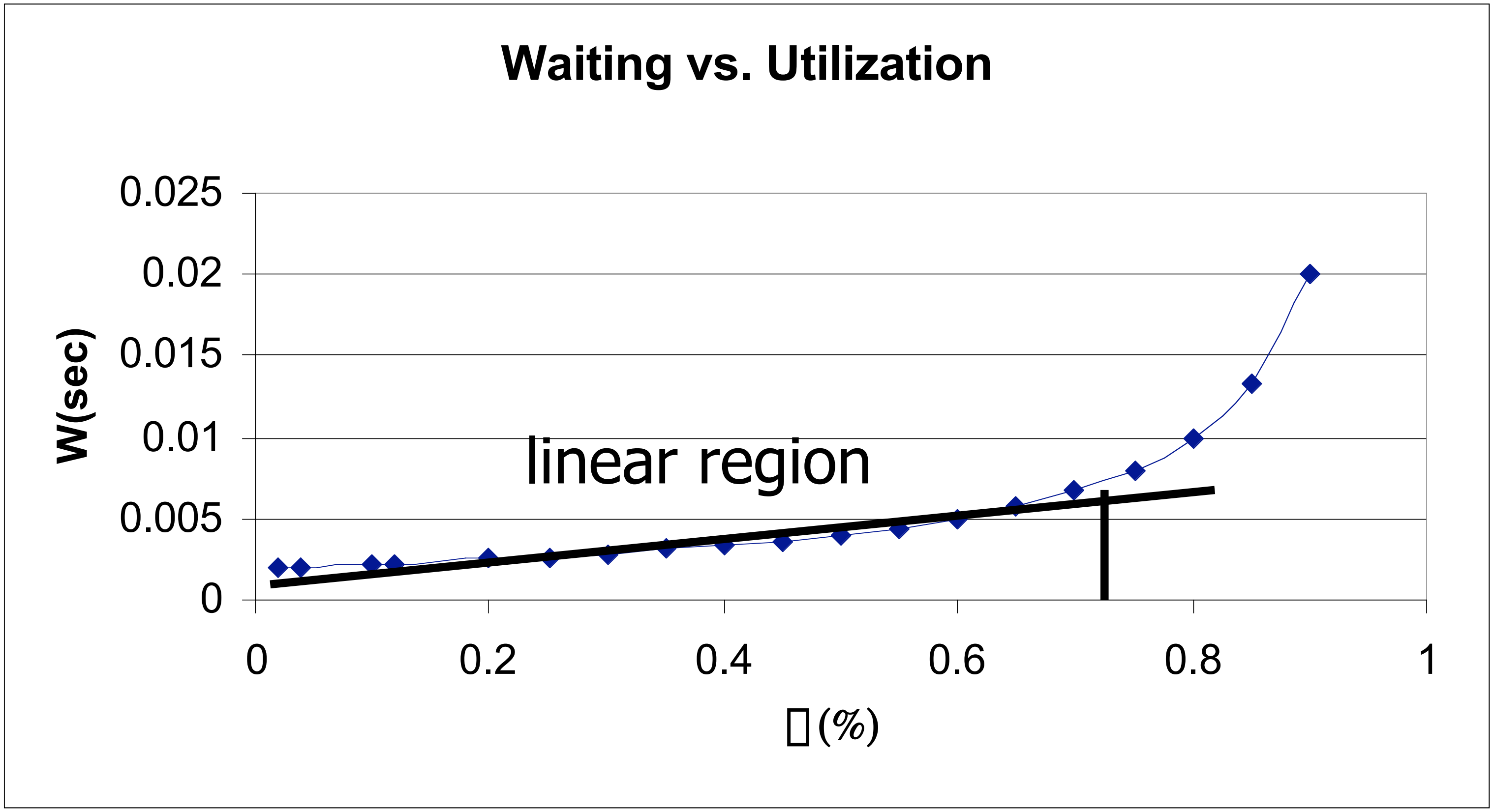
$$L_q = \lambda W_q = \lambda \frac{\lambda}{\mu(\mu - \lambda)} = \frac{\lambda^2}{\mu(\mu - \lambda)}$$

# Response Time vs. Arrivals

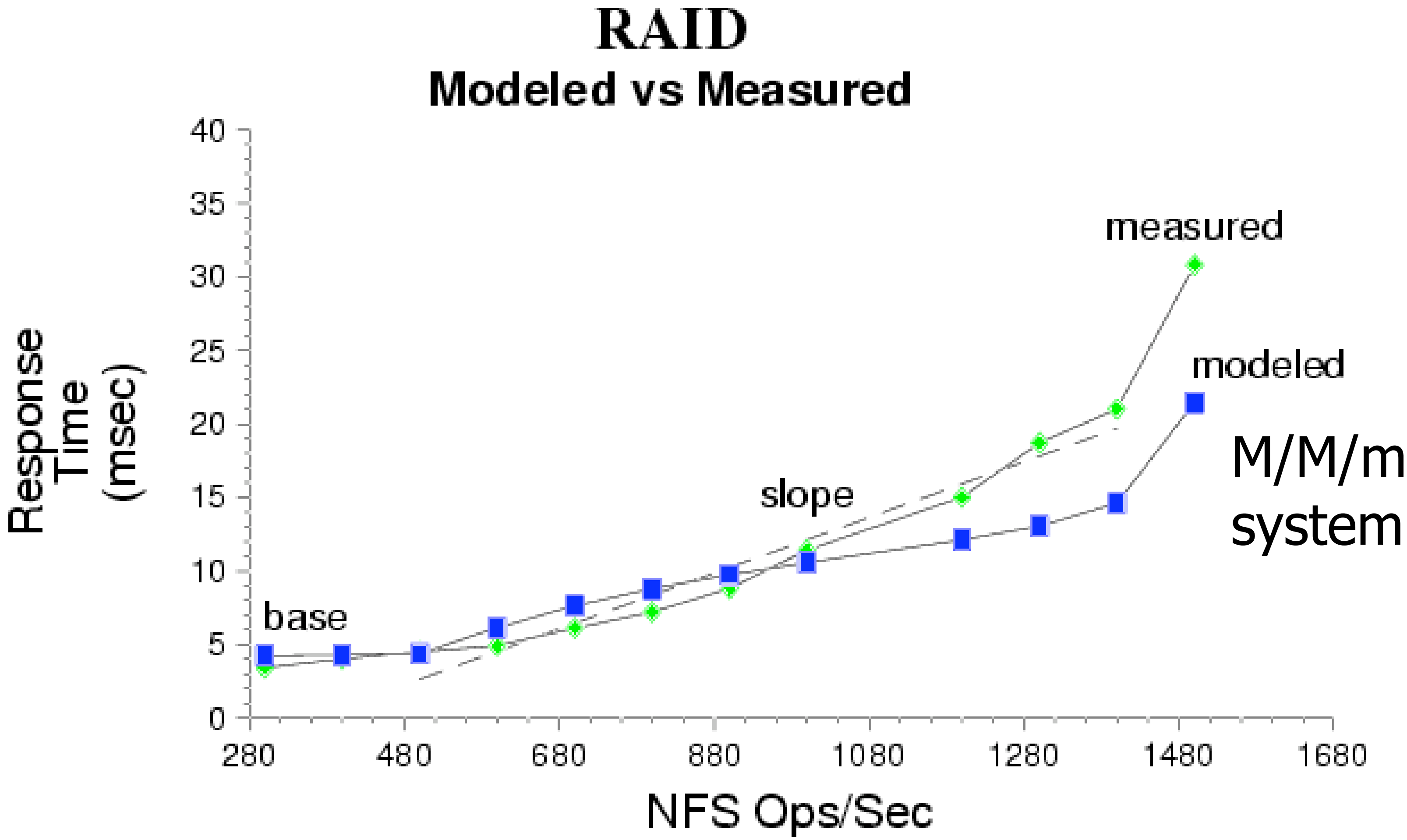


$$W = \frac{1}{\square\square\square}$$

# Stable Region



# Empirical Example



M/M/m
system

# Example

- Measurement of a network gateway:
  - mean arrival rate (□): 125 Packets/s
  - mean response time per packet: 2 ms
- Assuming exponential arrivals & departures:
  - What is the service rate, □ ?
  - What is the gateway’s utilization?
  - What is the probability of n packets in the gateway?
  - mean number of packets in the gateway?
  - The number of buffers so P(overflow) is $<10^{-6}$?

# Example (cont)

The service rate, $\mu = \dfrac{1}{0.002} = 500\,pps$

utilization = $\rho = (\frac{\lambda}{\mu}) = 0.25\%$

P(n) packets in the gateway =

$$P_0 P_n = (1 - \rho)(\rho^n) = (0.75)(0.25^n)$$

# Example (cont)

Mean # in gateway (L) =

$$\frac{\square}{1\square\square} = \frac{0.25}{1\square 0.25} = 0.33$$

to limit loss probability to less than 1 in a million:

$$\square^{n} \; \square 10^{\square 6}$$

# Properties of a Poisson processes

- Poisson process = exponential distribution between arrivals/departures/service

$$P(\text{arrival} < t) = 1 \square e^{\square\square t}$$

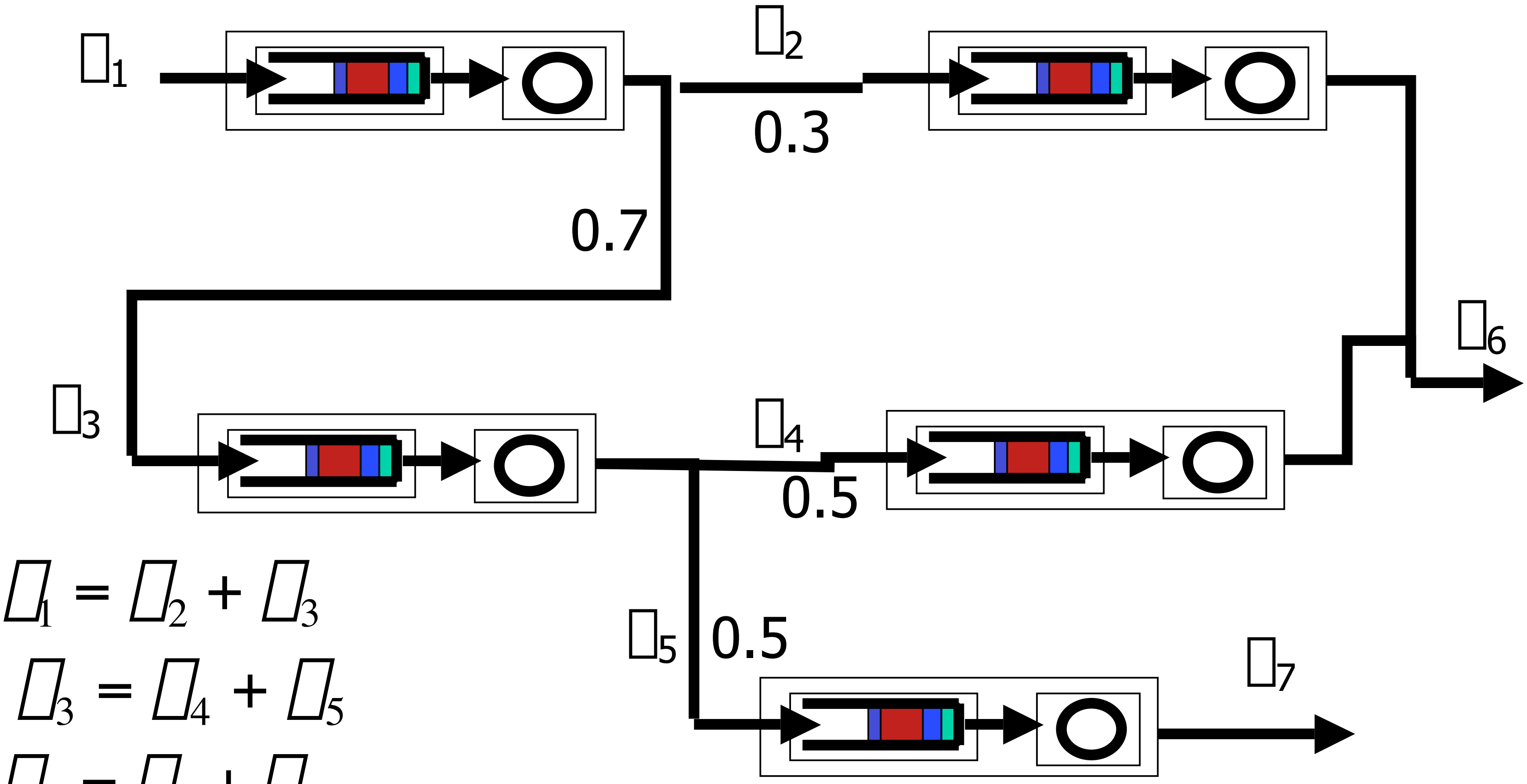
- Key properties:
  - memoryless
    - Past state does not help predict next arrival
  - Closed under:
    - Addition
    - Subtraction

# Addition and Subtraction

- Merge:
  - two poisson streams with arrival rates $\square_1$ and $\square_2$:
    - new poisson stream: $\square_3=\square_1+\square_2$
- Split :
  - If any given item has a probability $P_1$ of “leaving” the stream with rate $\square_{1:}$

    $\square$ $\square_2=(1-P_1)\square_1$

# Queuing Networks



$$\lambda_1 = \lambda_2 + \lambda_3$$

$$\lambda_3 = \lambda_4 + \lambda_5$$

$$\lambda_6 = \lambda_2 + \lambda_4$$

$$\lambda_7 = \lambda_5$$

# Bridging Router Performance and Queuing Theory

Sigmetrics 2004

Slides by N. Hohn*, D. Veitch*, K. Papagiannaki, C. Diot

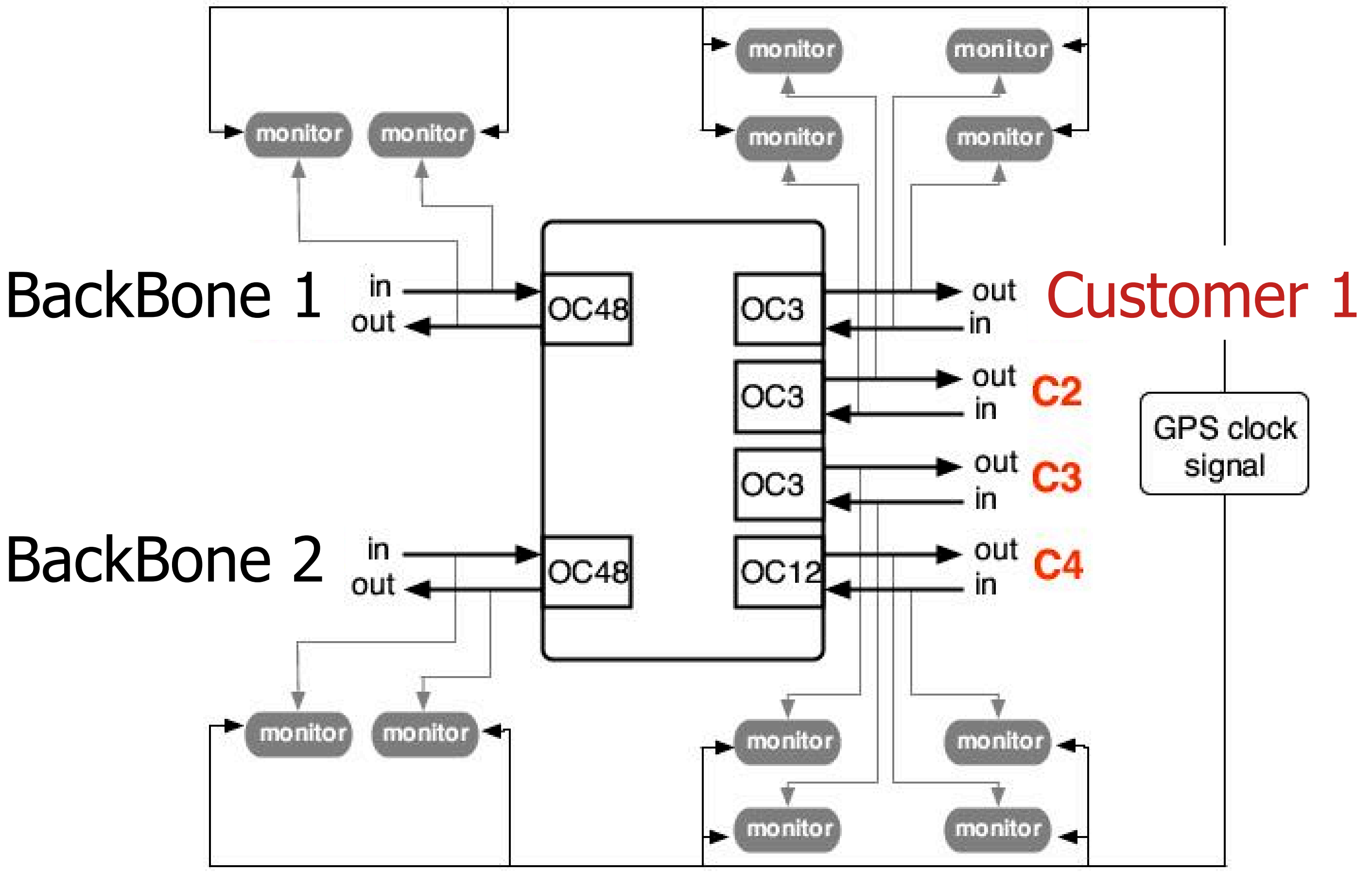
# Motivation

- End-to-end packet delay is an important metric for performance and Service Level Agreements (SLAs)
- Building block of end-to-end delay is through router delay
- Measure the delays incurred by *all* packets crossing a single router

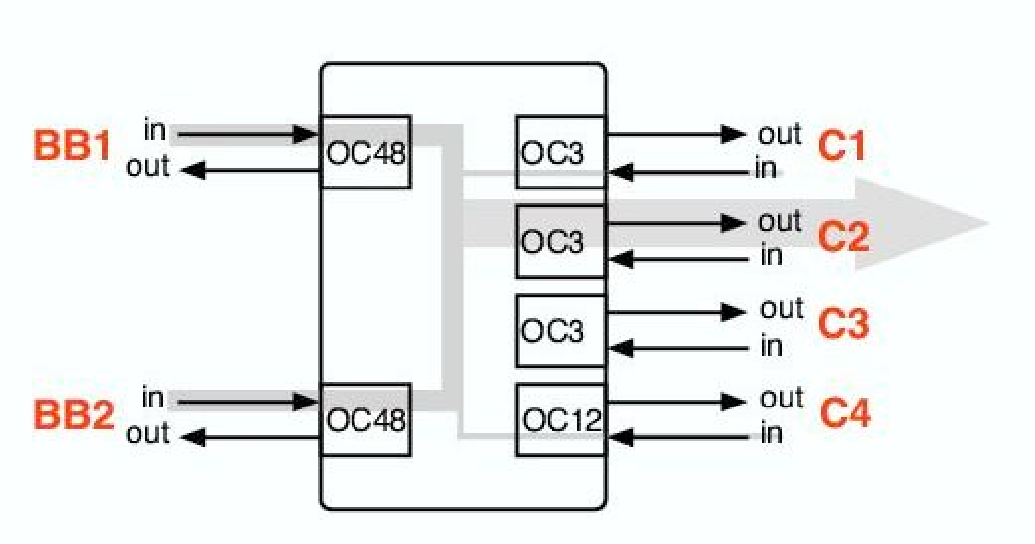
# Overview

- Full Router Monitoring
- Delay Analysis and Modeling
- Delay Performance: Understanding and Reporting

# Measurement Environment



# Packet matching



| Set | Link | Matched pkts | % traffic C2-out |
|---|---|---|---|
| C4 | In | 215987 | 0.03% |
| C1 | In | 70376 | 0.01% |
| BB1 | In | 345796622 | 47.00% |
| BB2 | In | 389153772 | 52.89% |
| C2 | out | 735236757 | **99.93**% |

# Overview

- Full Router Monitoring
- Delay Analysis and Modeling
- Delay Performance: Understanding and Reporting

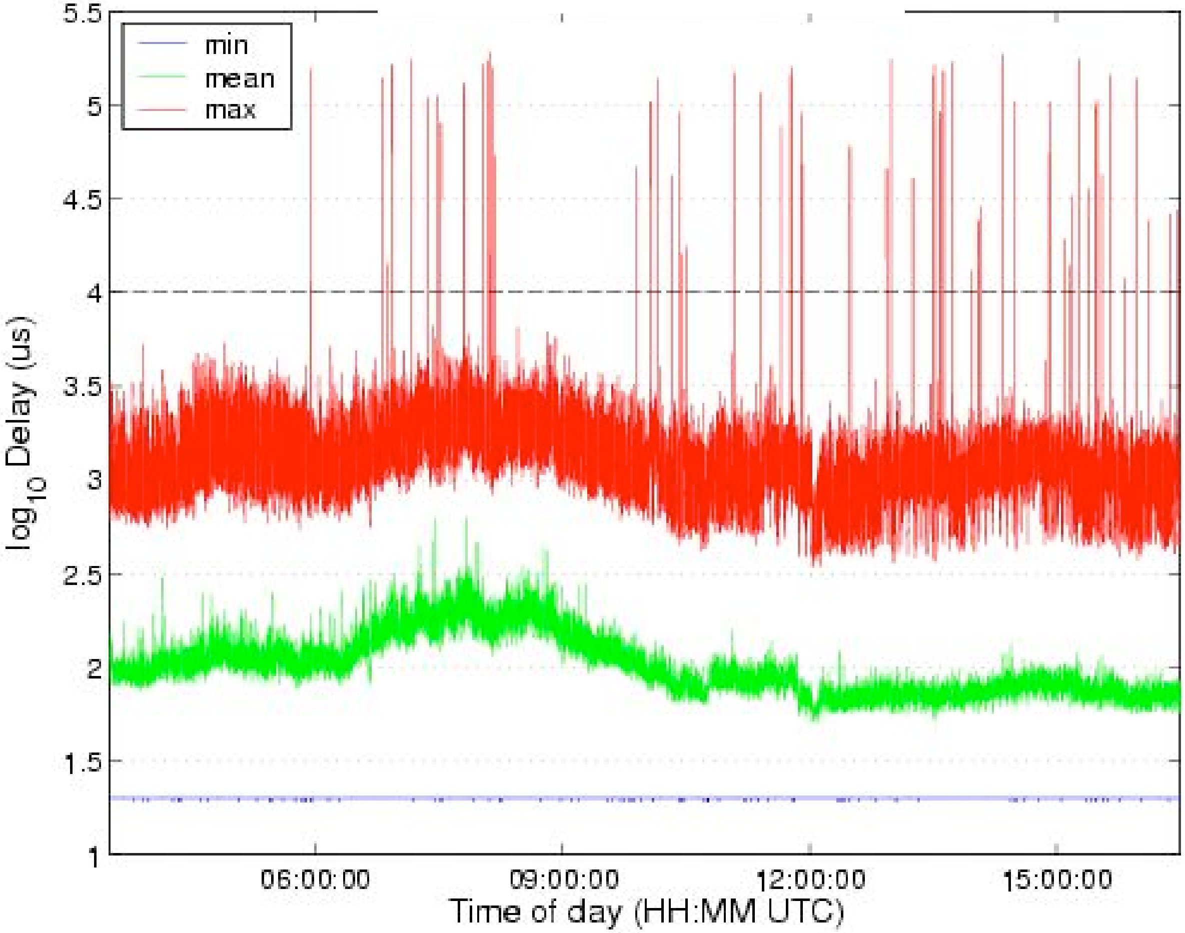
# Definition of delay

# Store & Forward Datapath

- Store: storage in input linecard's memory ← **Not part of the system**
- Forwarding decision
- Storage in dedicated Virtual Output Queue (VOQ)
- Decomposition into fixed-size cells
- Transmission through switch fabric cell by cell
- Packet reconstruction
- Forward: Output link scheduler

# Delays: 1 minute summary

BB1-In to C2-Out



MAX

Mean

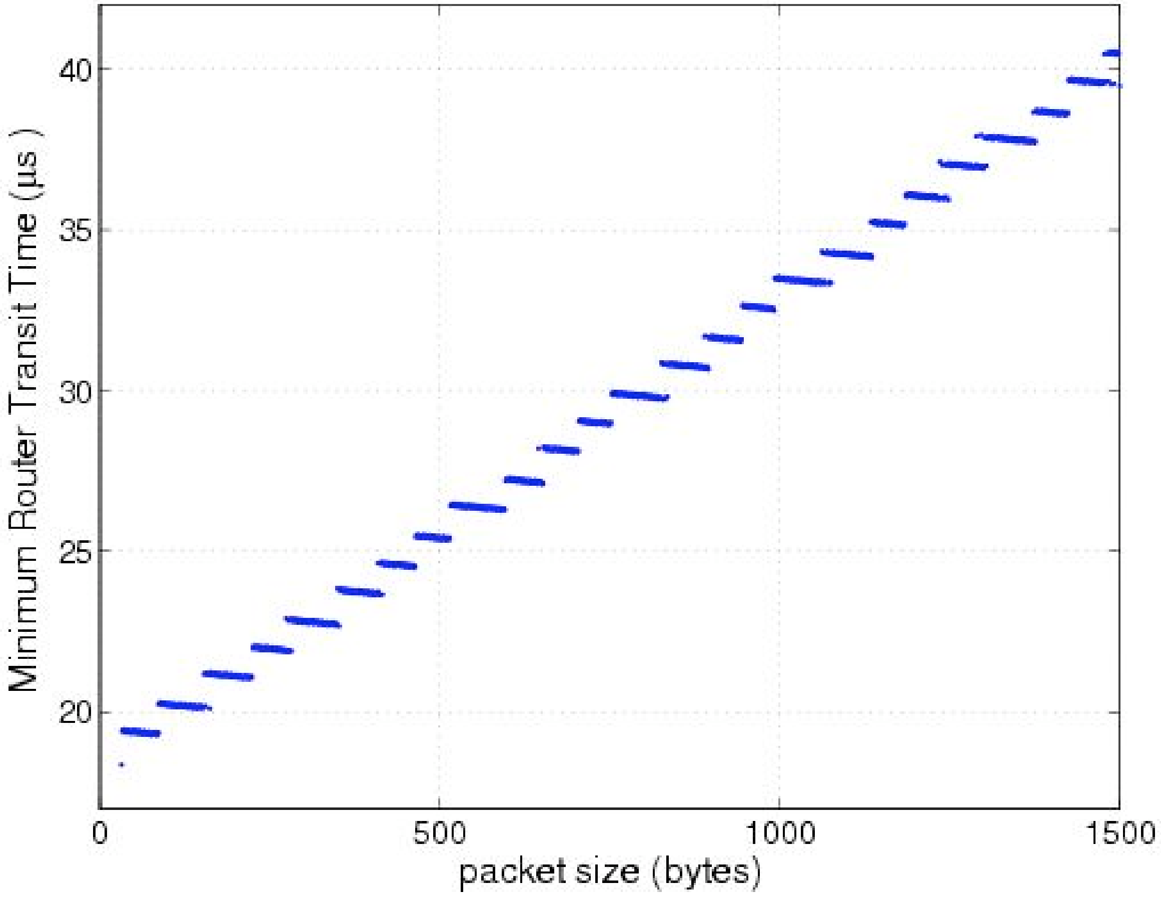
MIN

# Store & Forward Datapath

- Store: storage in input linecard's memory ← **Not part of the system**
- Forwarding decision
- Storage in dedicated Virtual Output Queue (VOQ)
- Decomposition into fixed-size cells
- Transmission through switch fabric cell by cell
- Packet reconstruction
- Forward: Output link scheduler

$$\left.\begin{array}{l}\text{Forwarding decision}\\ \text{Storage in dedicated Virtual Output Queue (VOQ)}\\ \text{Decomposition into fixed-size cells}\\ \text{Transmission through switch fabric cell by cell}\\ \text{Packet reconstruction}\\ \text{Forward: Output link scheduler}\end{array}\right\} \boldsymbol{\square\square_{i}\square_{j}(L)}$$

# Minimum Transit Time



**Packet size dependent minimum delay.**

# Store & Forward Datapath

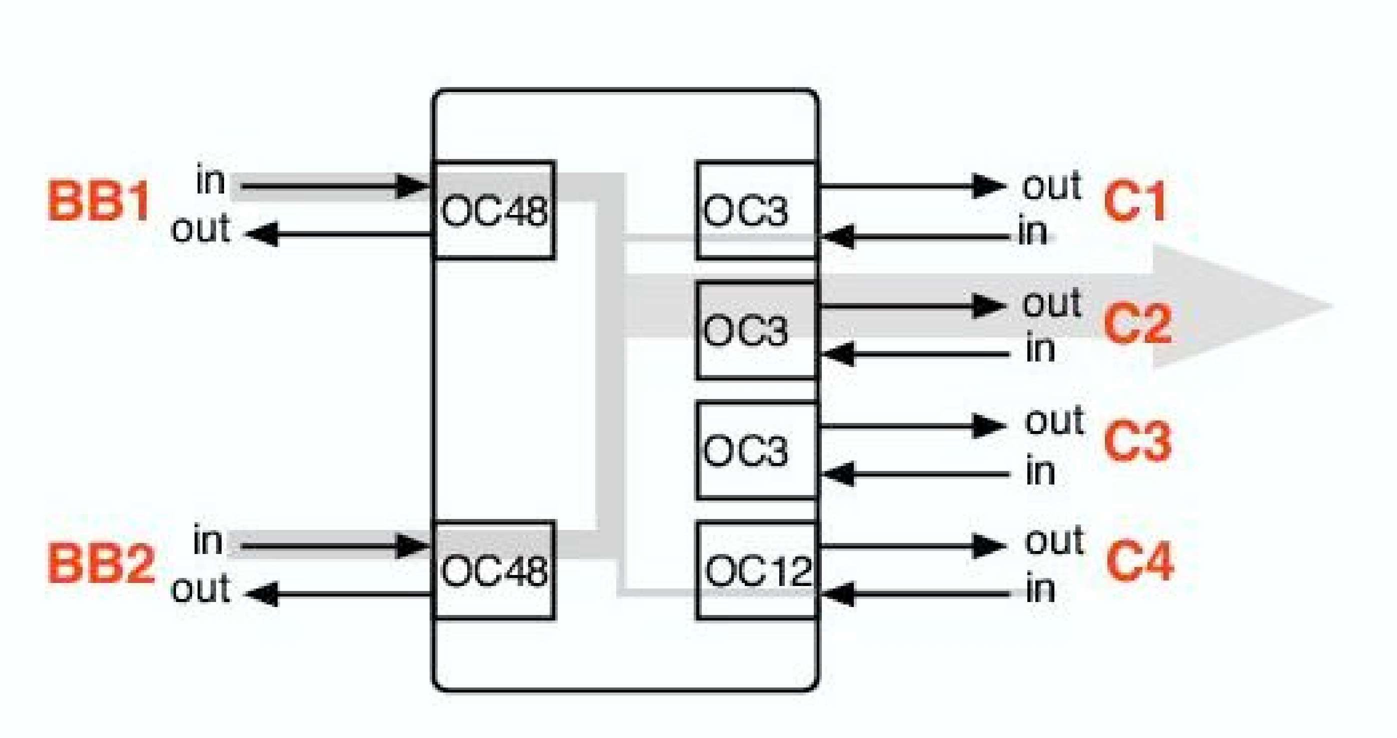
- Store: storage in input linecard's memory
- Forwarding decision
- Storage in dedicated Virtual Output Queue (VOQ)
- Decomposition into fixed-size cells
- Transmission through switch fabric cell by cell
- Packet reconstruction
- Forward: Output link scheduler

← **Not part of the system**

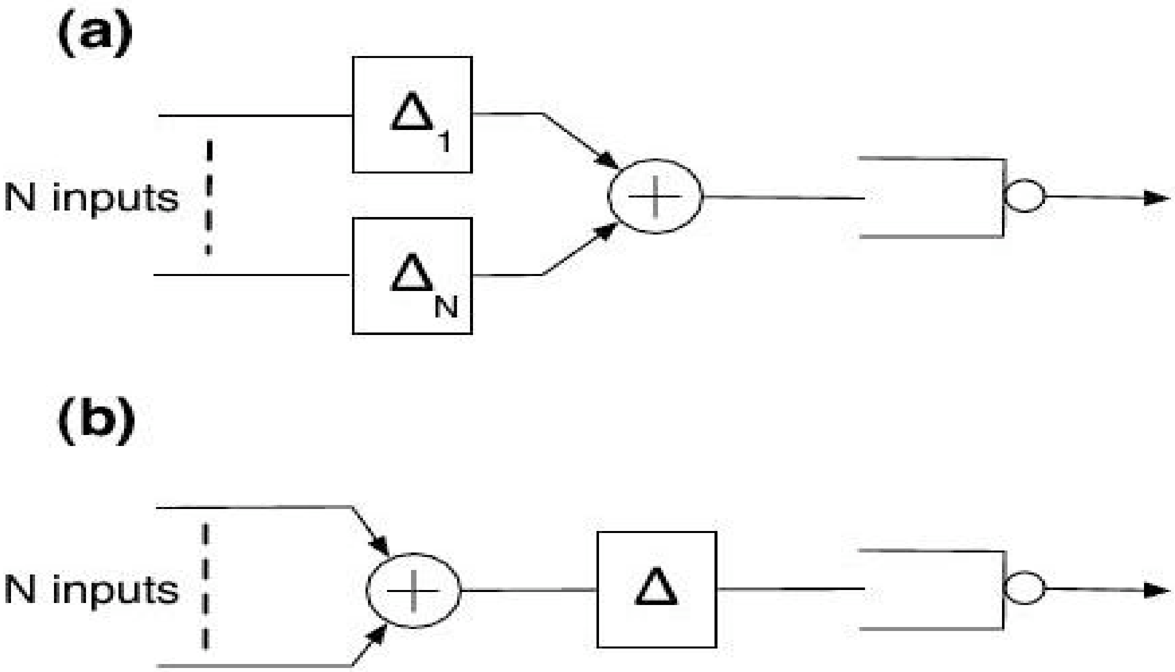
$\mathbf{\square\square_i\square_j(L)}$

← **FIFO queue**

# Modeling

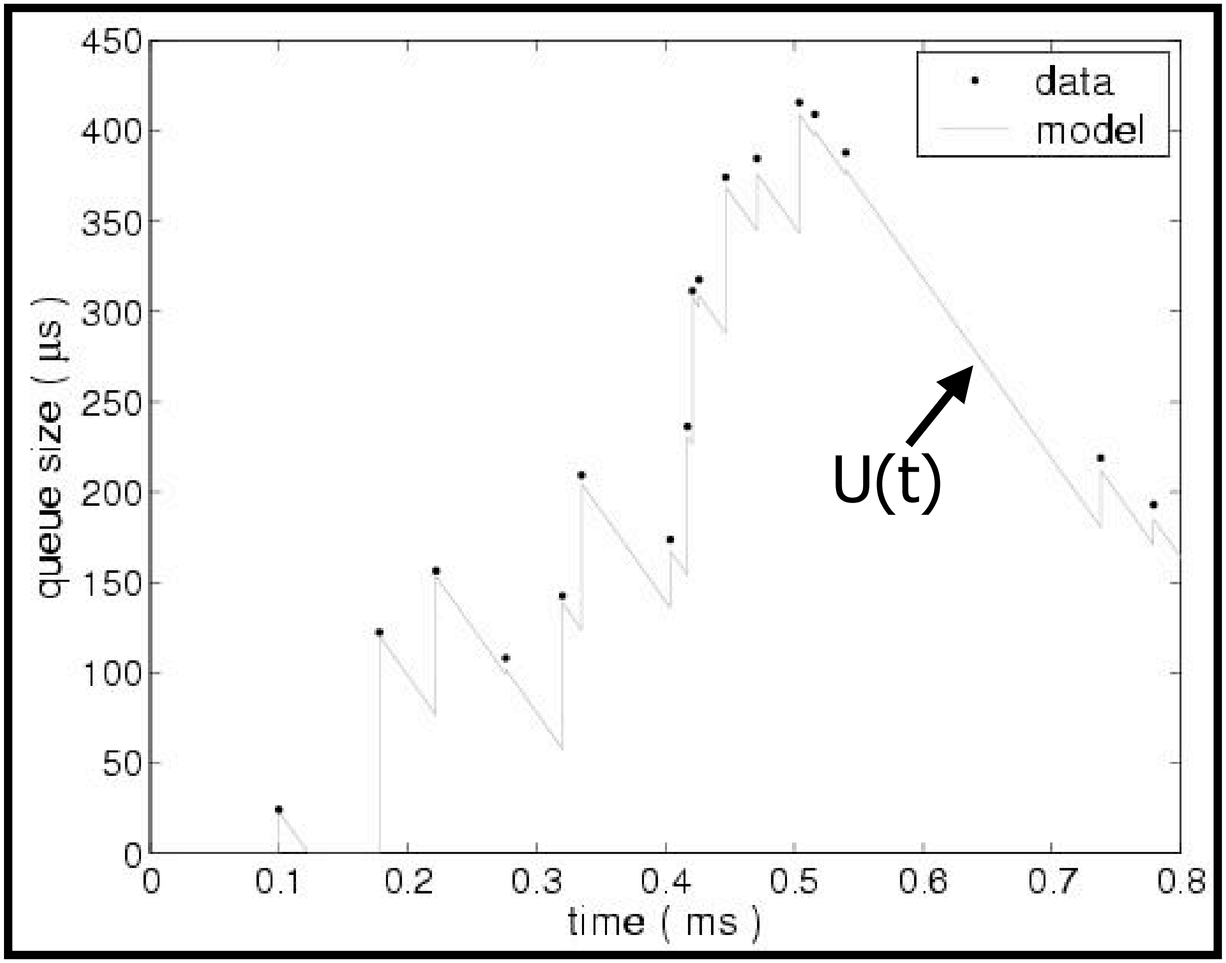

BB1
in
out
OC48
BB2
in
out
OC48
OC3
out
in
C1
OC3
out
in
C2
OC3
out
in
C3
OC12
out
in
C4

# Modeling

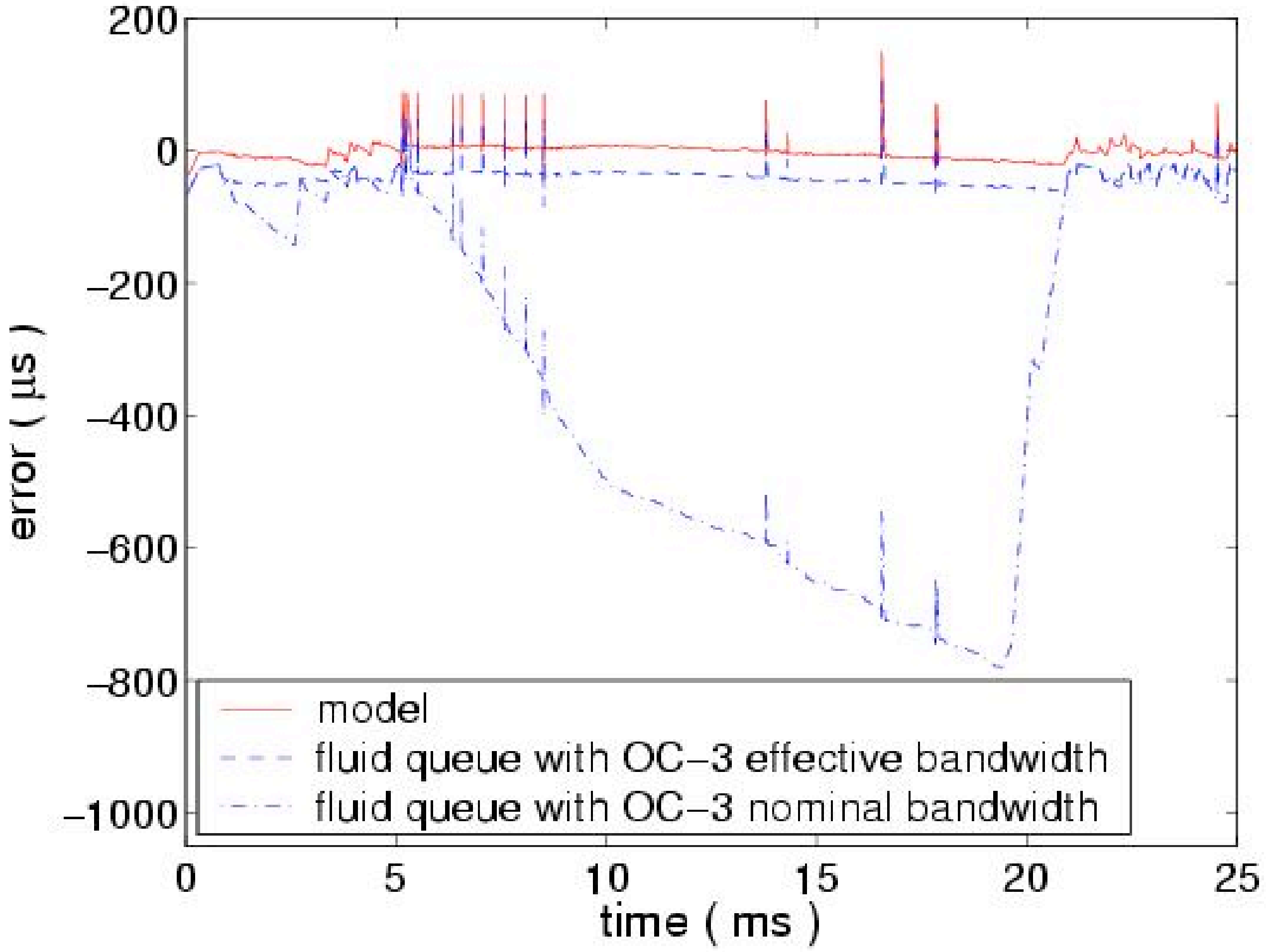


**Fluid queue with a delay element at the front**

# Model Validation



# Error as a function of time



# Modeling results

- A crude model performs well!
  - As simpler/simpler than an M/M/1 queue
- Use effective link bandwidth
  - account for encapsulation
- Small gap between router performance and queuing theory!
- The model defines Busy Periods: time between the arrival of a packet to the empty system and the time when the system becomes empty again.

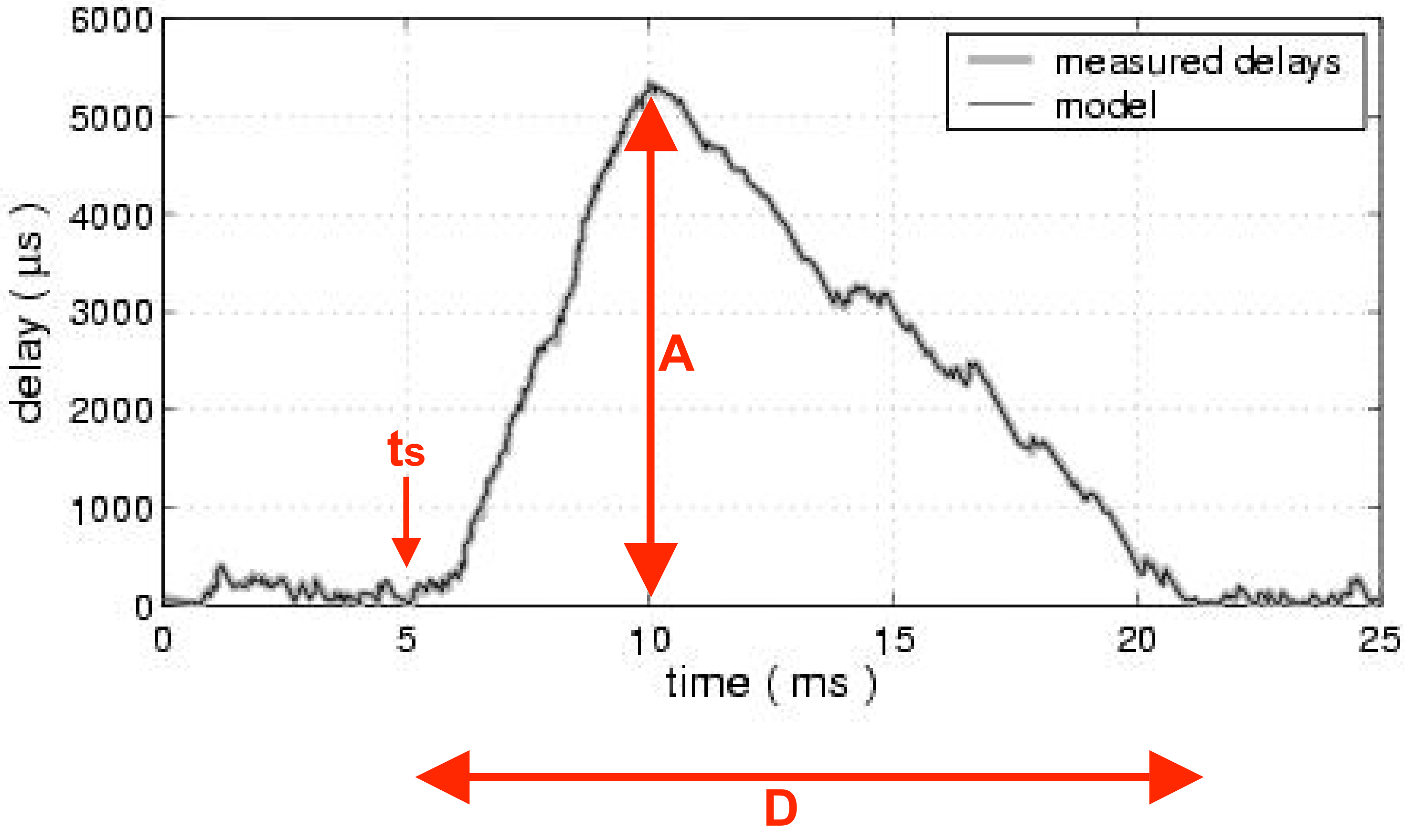
# Overview

- Full Router Monitoring
- Delay Analysis and Modeling
- Delay Performance: Understanding and Reporting

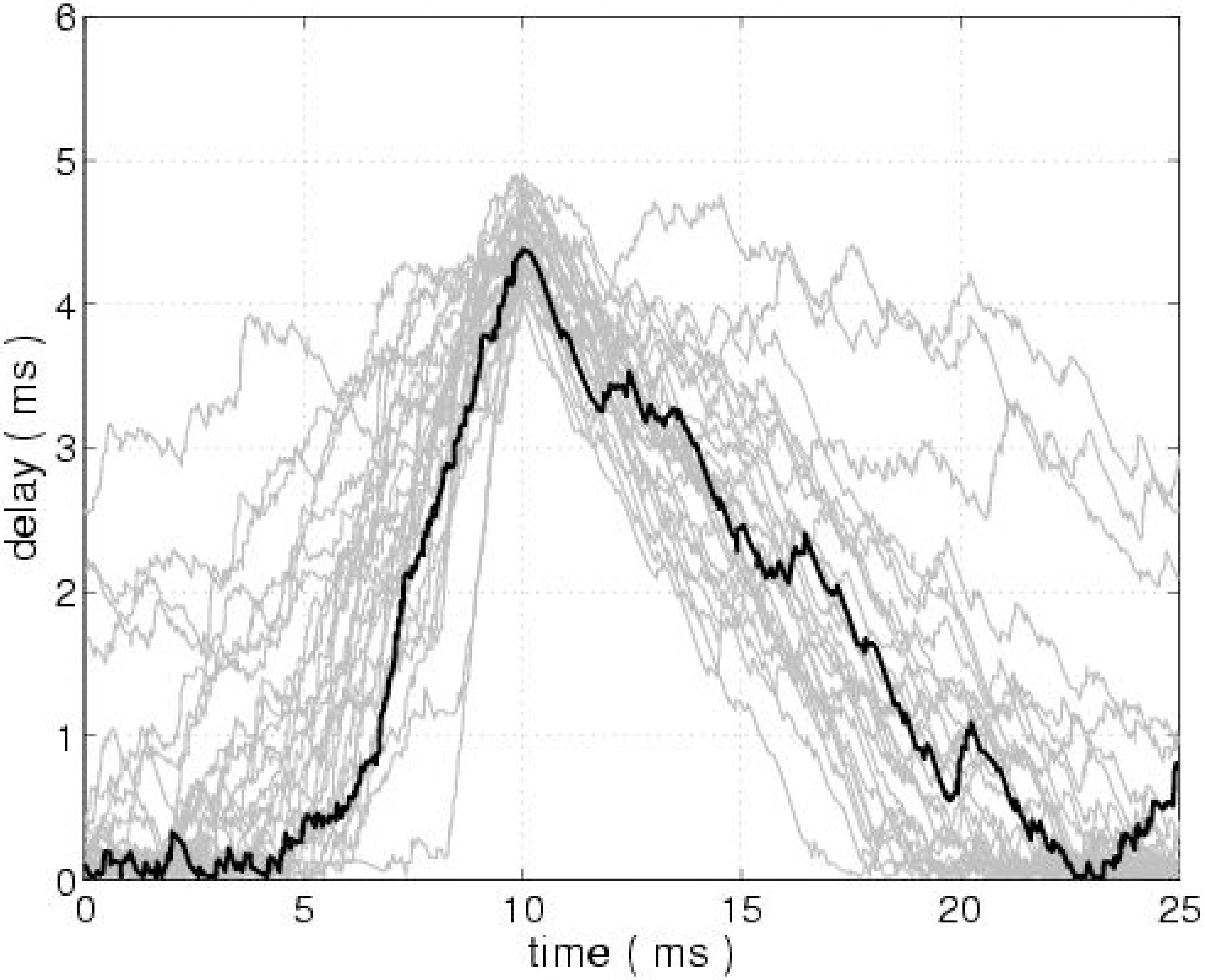
# On the Delay Performance

- Model allows for router performance evaluation when arrival patterns are known
- Goal: metrics that
  - Capture operational-router performance
  - Can answer performance questions directly
- Busy Period structures contain *all delay* information
  - BP better than utilization or delay reporting

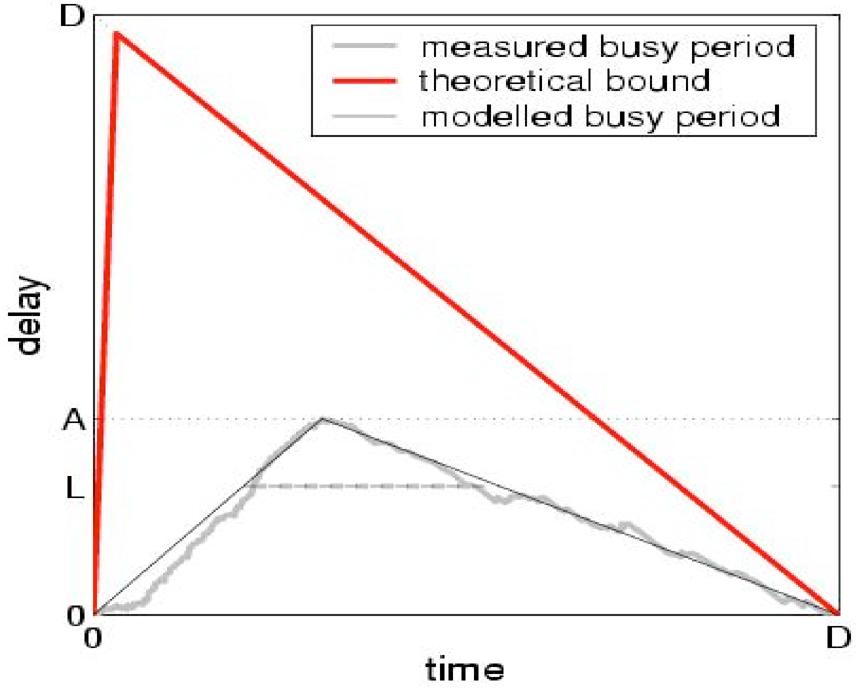
# Busy periods metrics



# Property of significant BPs



# Triangular Model

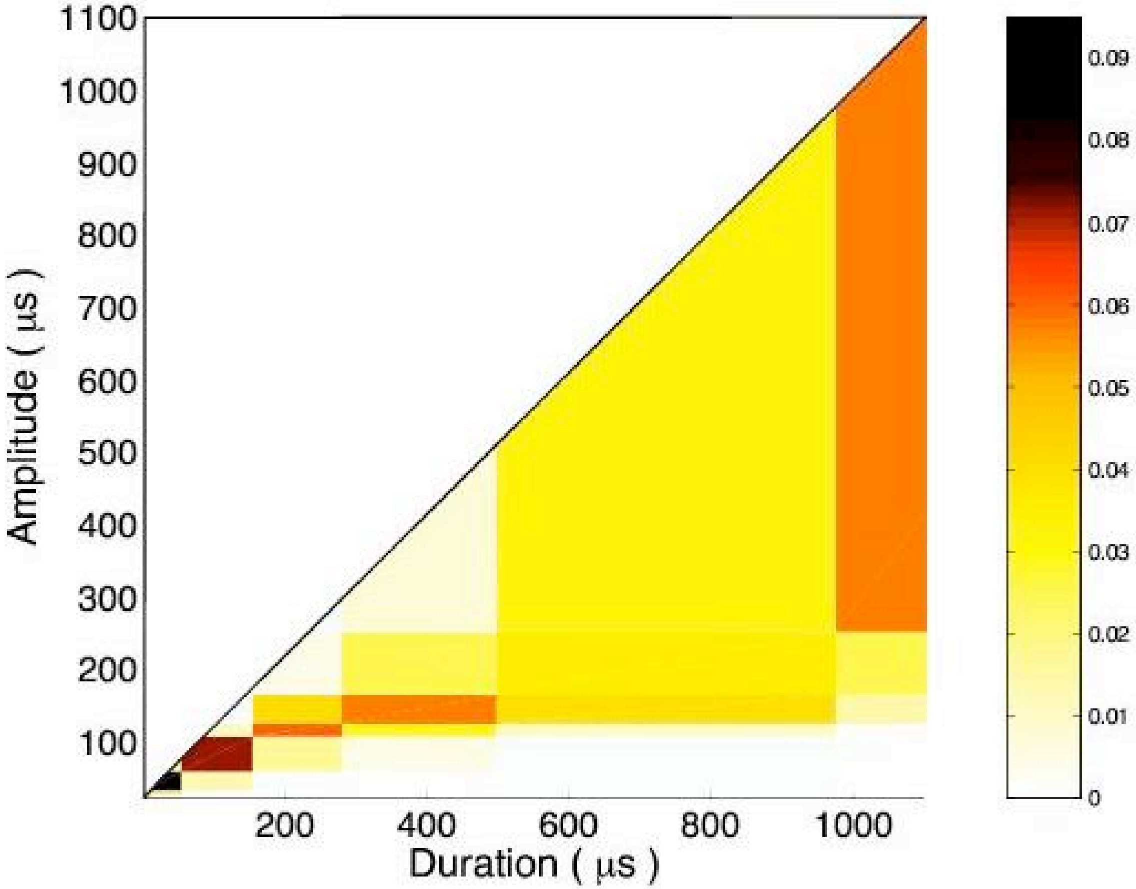


$$d_{L,A,D}^{(T)} = D(1 \square \frac{L}{A}), if \quad A \geq L$$

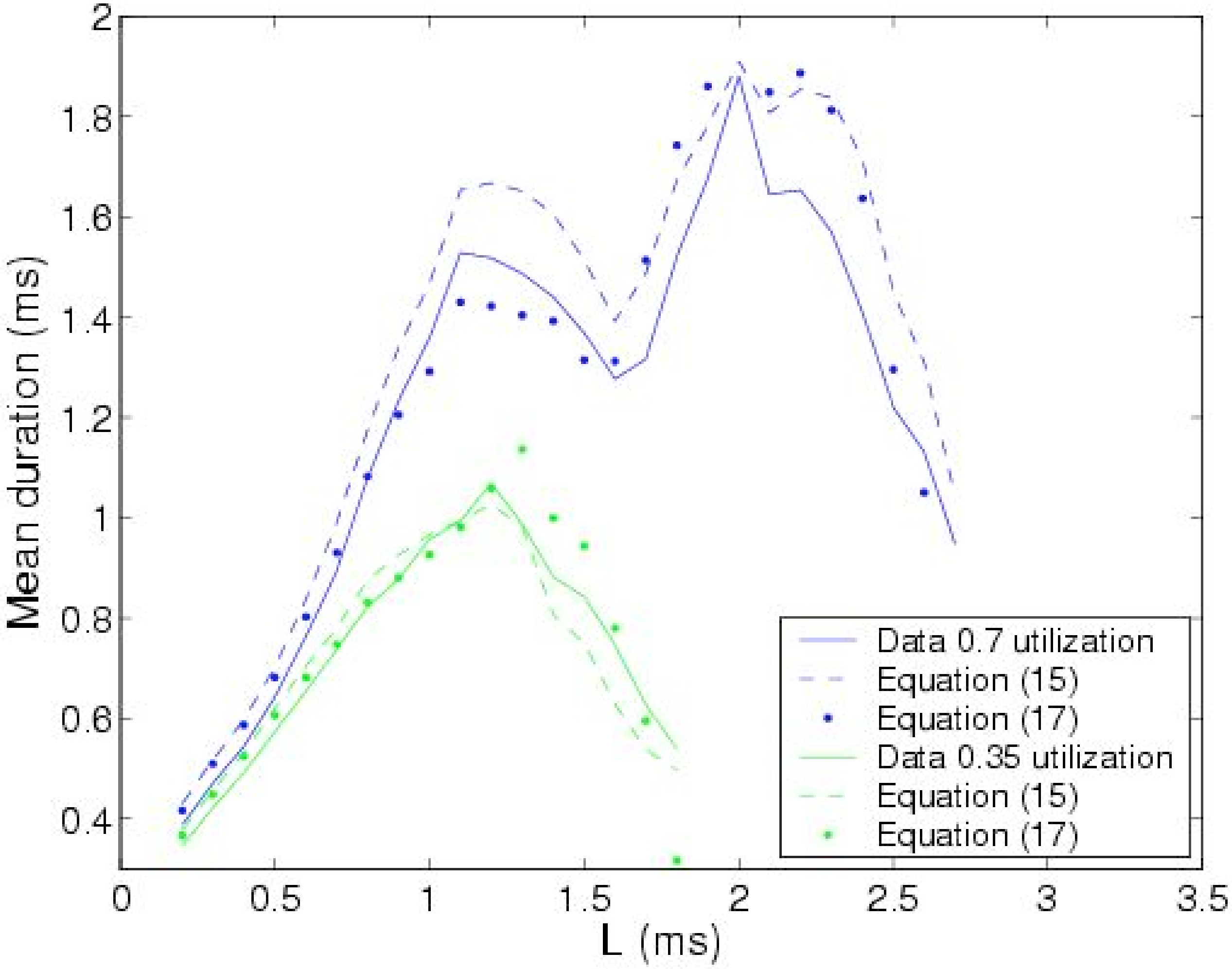
# Issues

- Report (A,D) measurements
- There are millions of busy periods even on a lightly utilized router
- Interesting episodes are rare and last for a very small amount of time

# Report BP joint distribution



# Duration of Congestion Level-L



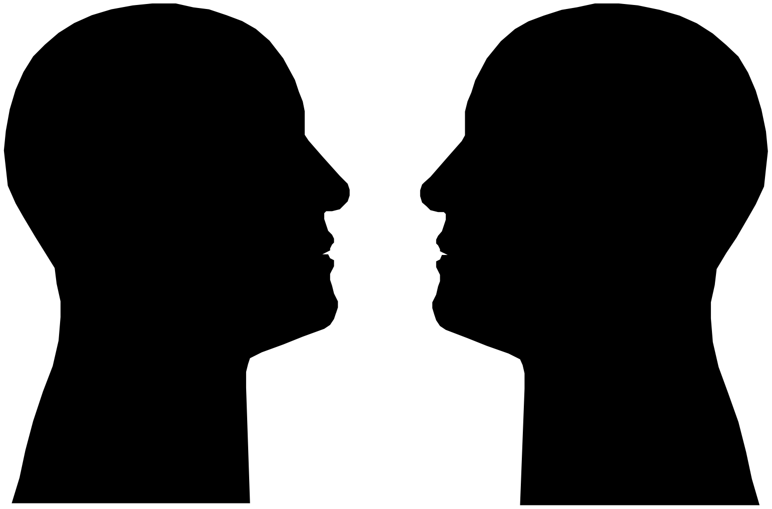
# Conclusions

- Results
  - Full router empirical study
  - Delay modeling
  - Reporting performance metrics
- Future work
  - Fine tune reporting scheme
  - Empirical evidence of large deviations theory

# Network Traffic Self-Similarity

Slides by Carey Williamson

Department of Computer Science
University of Saskatchewan



# Introduction

- A classic measurement study has shown that aggregate Ethernet LAN traffic is <u>self-similar</u> [Leland et al 1993]
- A statistical property that is very different from the traditional Poisson-based models
- This presentation: definition of network traffic self-similarity, Bellcore Ethernet LAN data, implications of self-similarity

# Measurement Methodology

- Collected lengthy traces of Ethernet LAN traffic on Ethernet LAN(s) at Bellcore
- High resolution time stamps
- Analyzed statistical properties of the resulting time series data
- Each observation represents the number of packets (or bytes) observed per time interval (e.g., 10 4 8 12 7 2 0 5 17 9 8 8 2...)

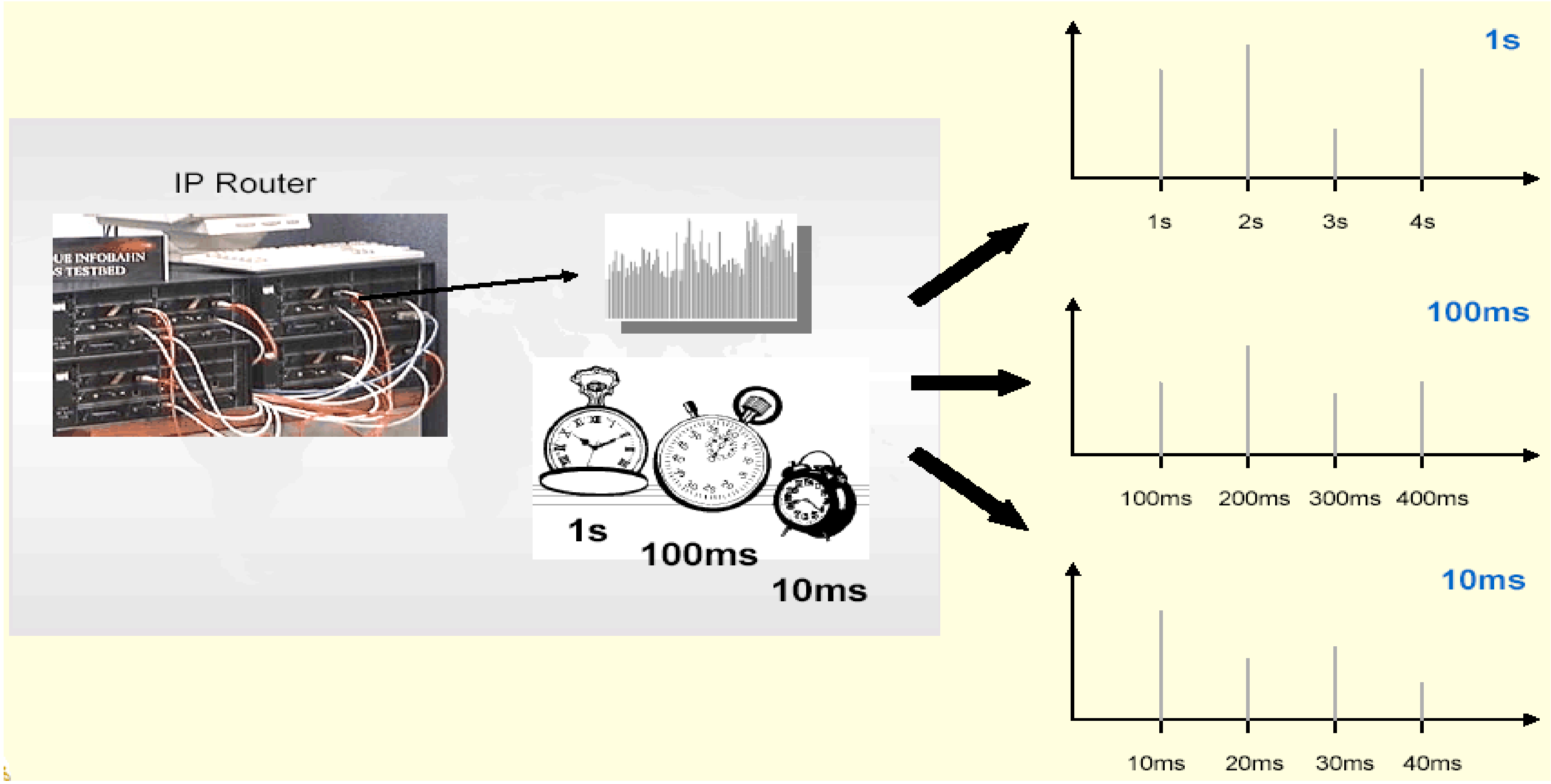
# Self-Similarity: The intuition

- If you plot the number of packets observed per time interval as a function of time, then the plot looks “the same” regardless of what interval size you choose
- E.g., 10 msec, 100 msec, 1 sec, 10 sec,...
- Same applies if you plot number of bytes observed per interval of time

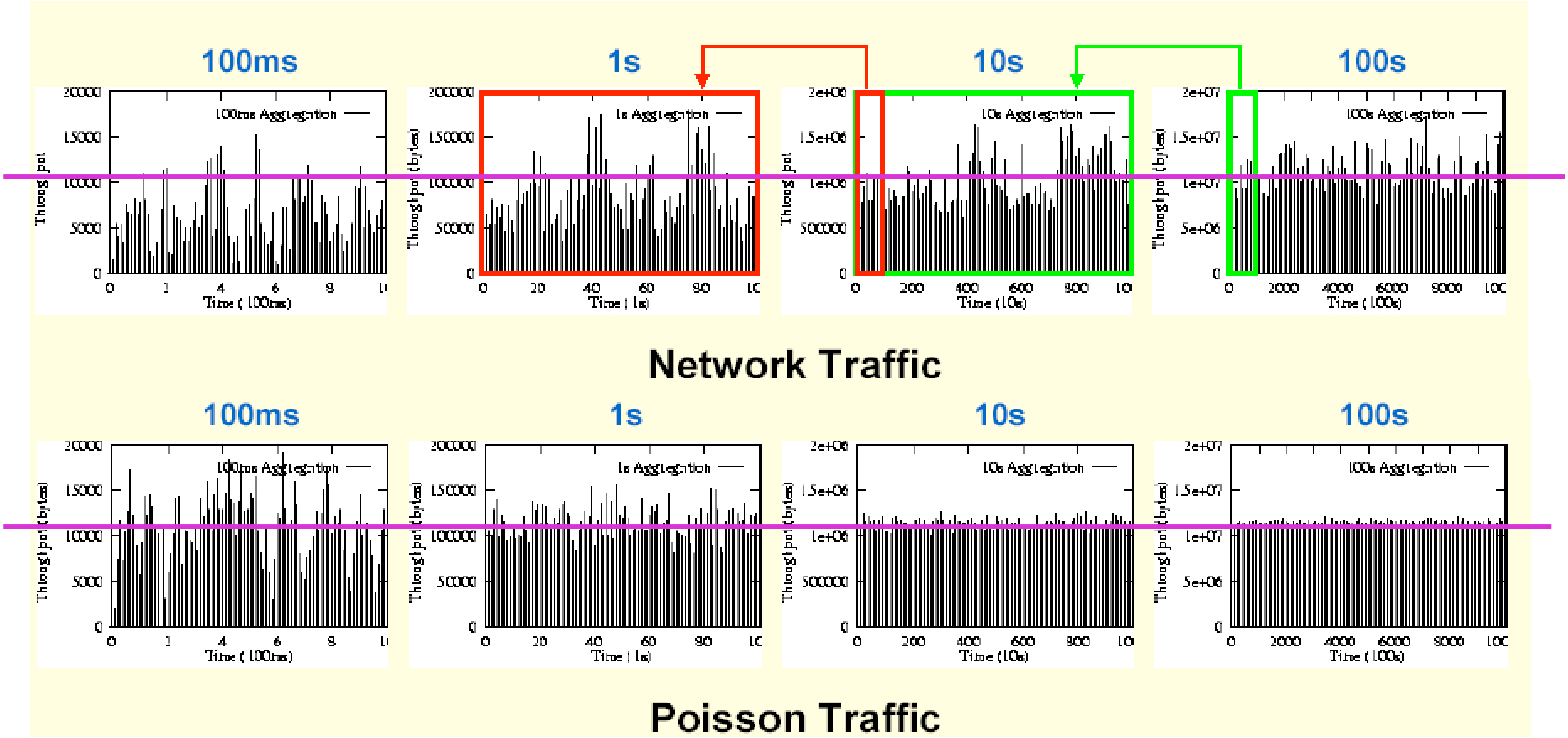
# Self-Similarity: The Intuition

- In other words, self-similarity implies a “fractal-like” behavior: no matter what time scale you use to examine the data, you see similar patterns
- Implications:
  - Burstiness exists across many time scales
  - No natural length of a burst
  - Key: Traffic does not necessarily get “smoother” when you aggregate it (unlike Poisson traffic)

# Self-Similarity Traffic Intuition (I)



# Self-Similarity in Traffic Measurement II



Network Traffic

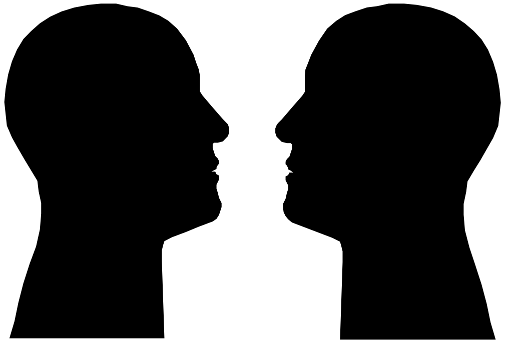
Poisson Traffic

# Self-Similarity: The Math

- Self-similarity is a rigorous statistical property
  - (i.e., a lot more to it than just the pretty “fractal-like” pictures)
- Assumes you have time series data with finite mean and variance
  - i.e., covariance stationary stochastic process
- Must be a very long time series
  - infinite is best!
- Can test for presence of self-similarity

# Self-Similarity: The Math

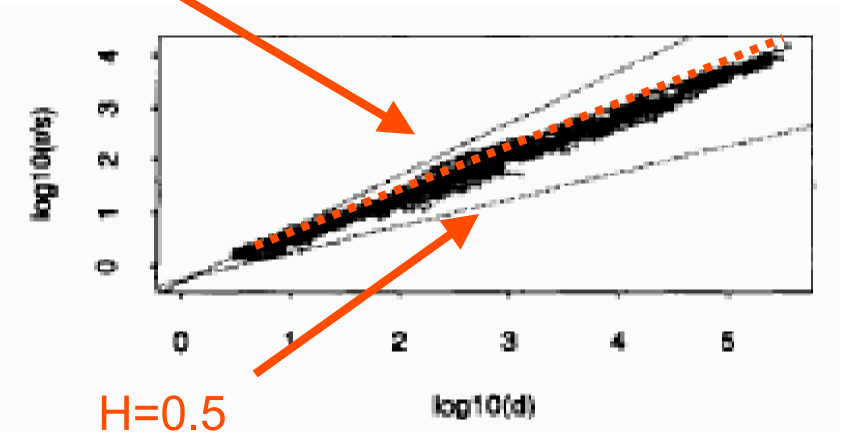
- Self-similarity manifests itself in several equivalent fashions:
- Slowly decaying variance
- Long range dependence
- Non-degenerate autocorrelations
- Hurst effect



# Methods of showing Self-Similarity

Estimate H  0.8

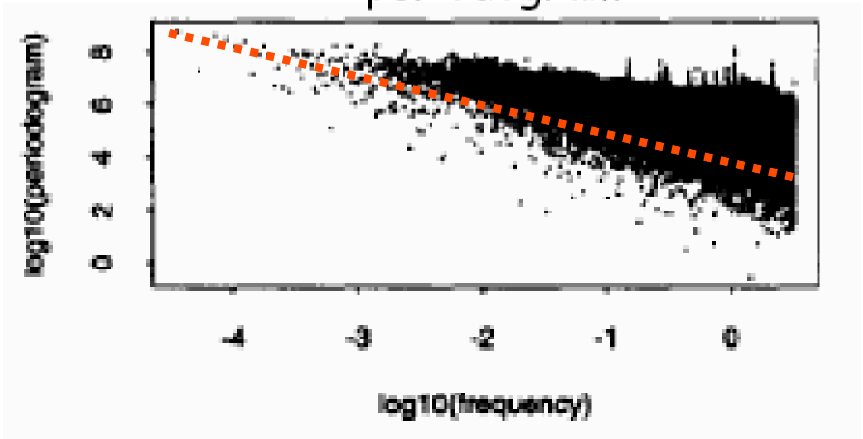
R/S method



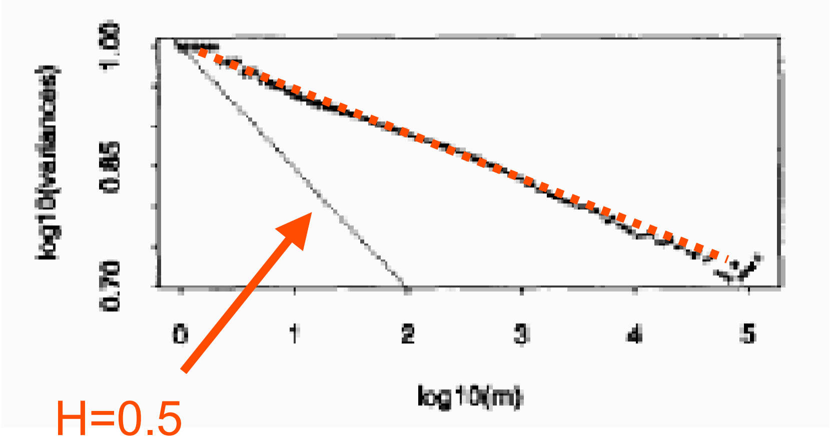
H=1

H=0.5

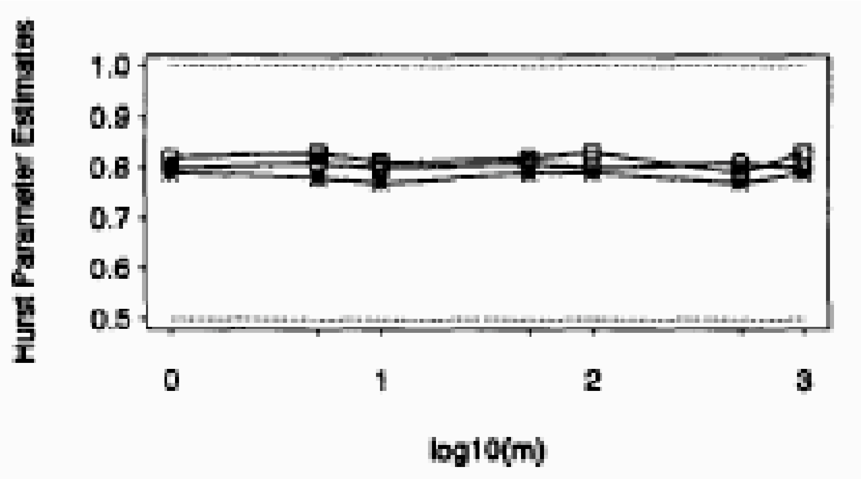
periodogram



variance time method



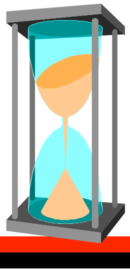
H=0.5



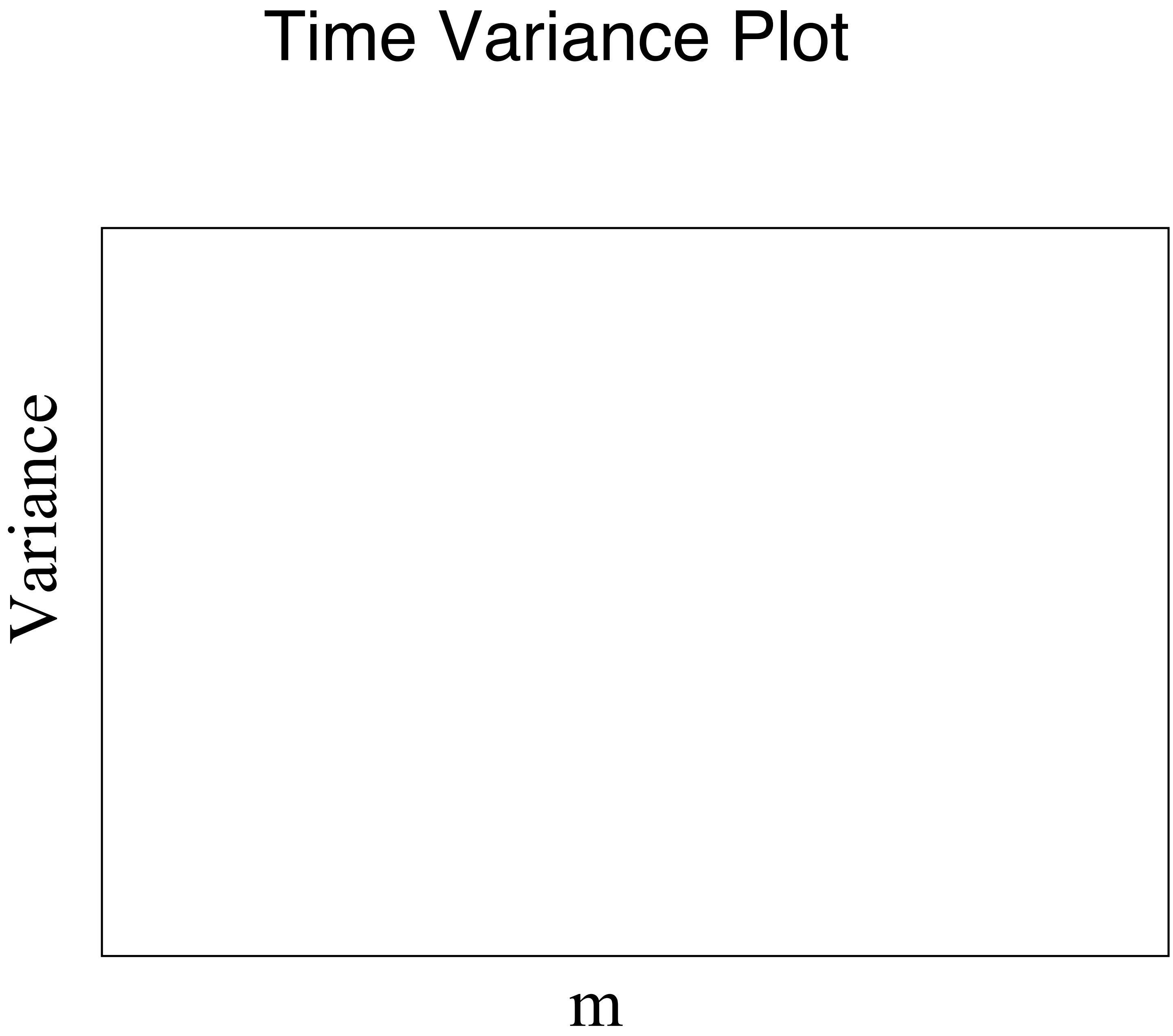
# Slowly Decaying Variance

- The variance of the sample decreases more slowly than the reciprocal of the sample size
- For most processes, the variance of a sample diminishes quite rapidly as the sample size is increased, and stabilizes soon
- For self-similar processes, the variance decreases very slowly, even when the sample size grows quite large

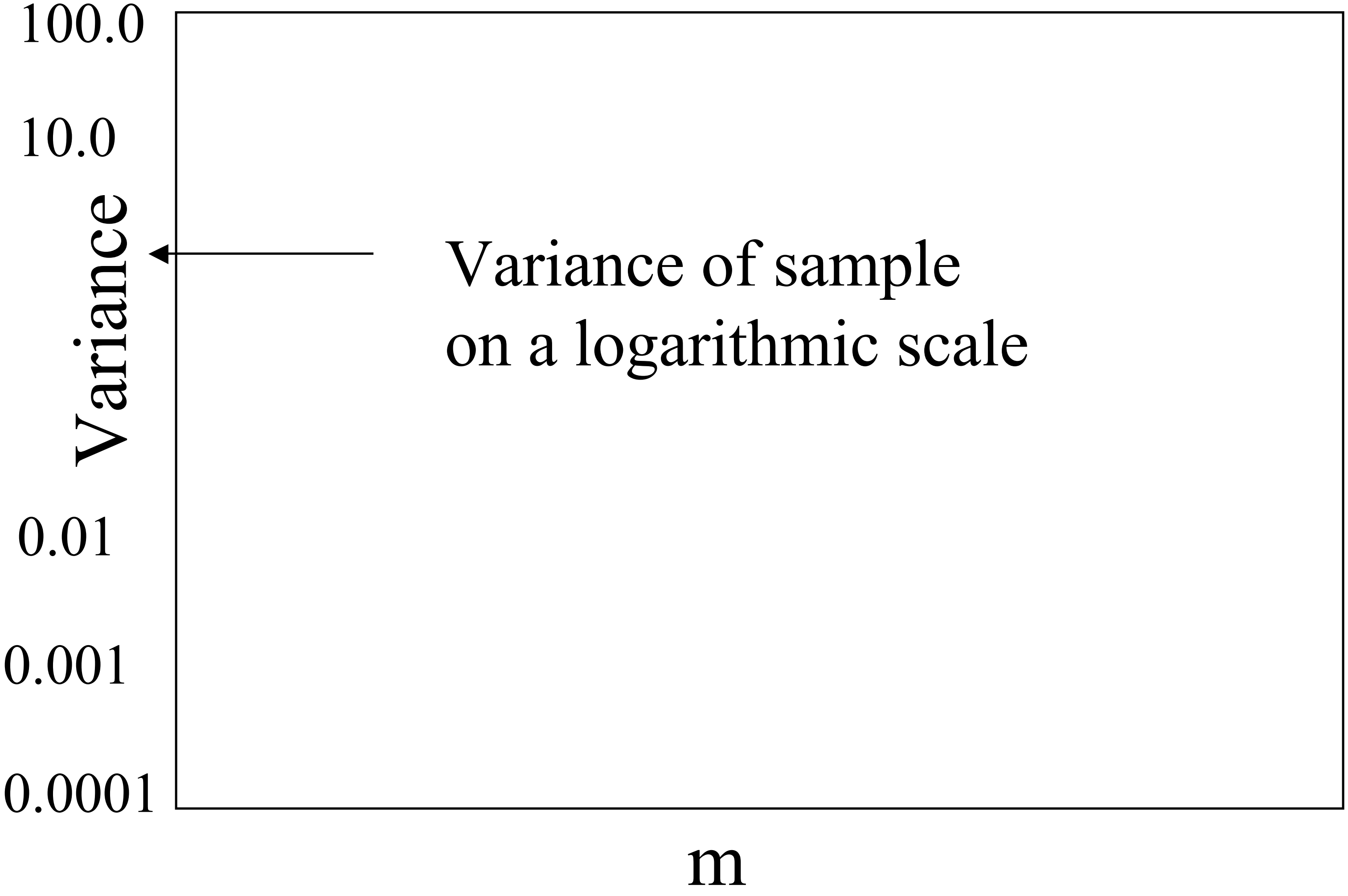
# Time-Variance Plot



- The “variance-time plot” is one means to test for the slowly decaying variance property
- Plots the variance of the sample versus the sample size, on a log-log plot
- For most processes, the result is a straight line with slope -1
- For self-similar, the line is much flatter

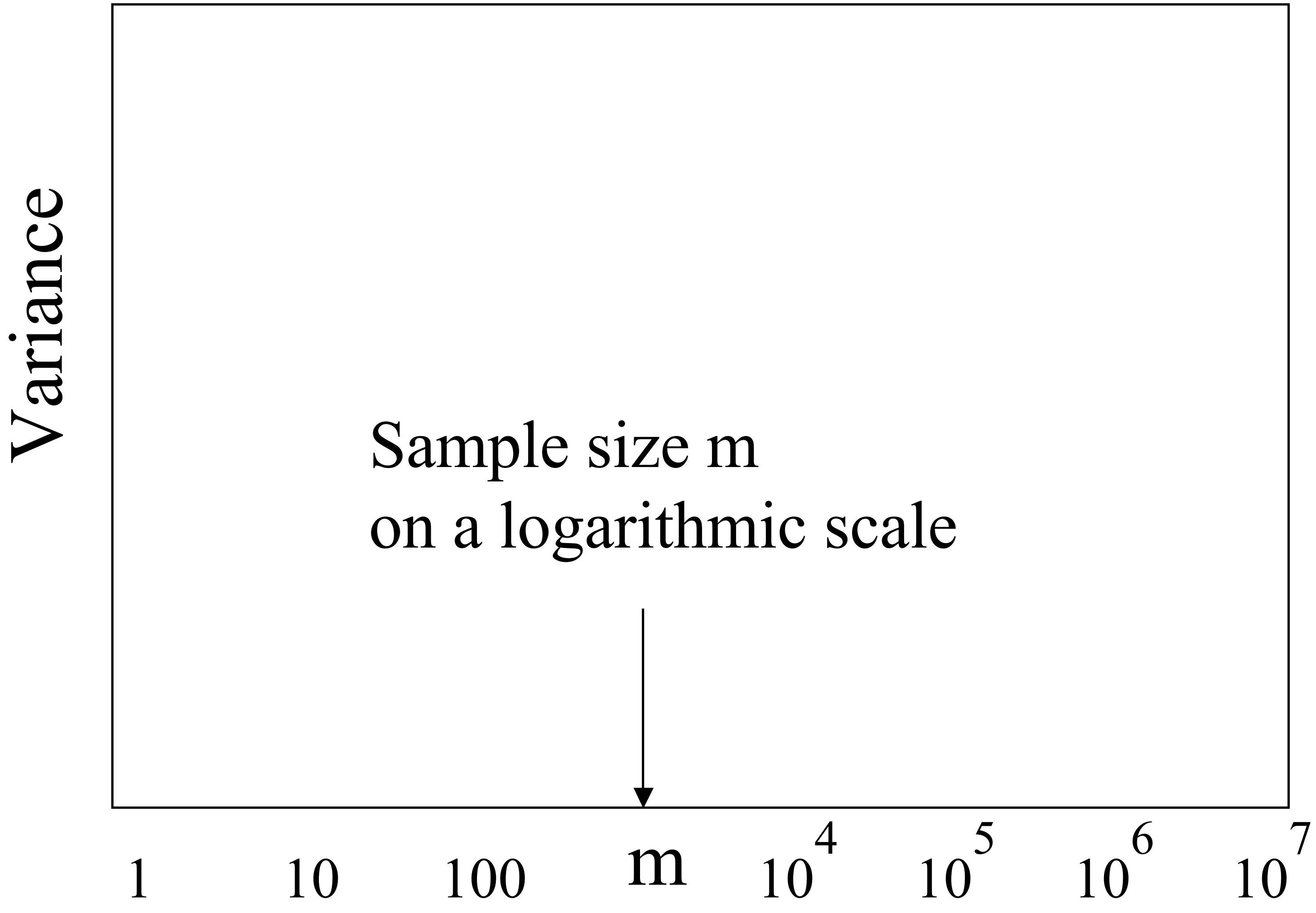

Time Variance Plot
Variance
m

# Variance-Time Plot

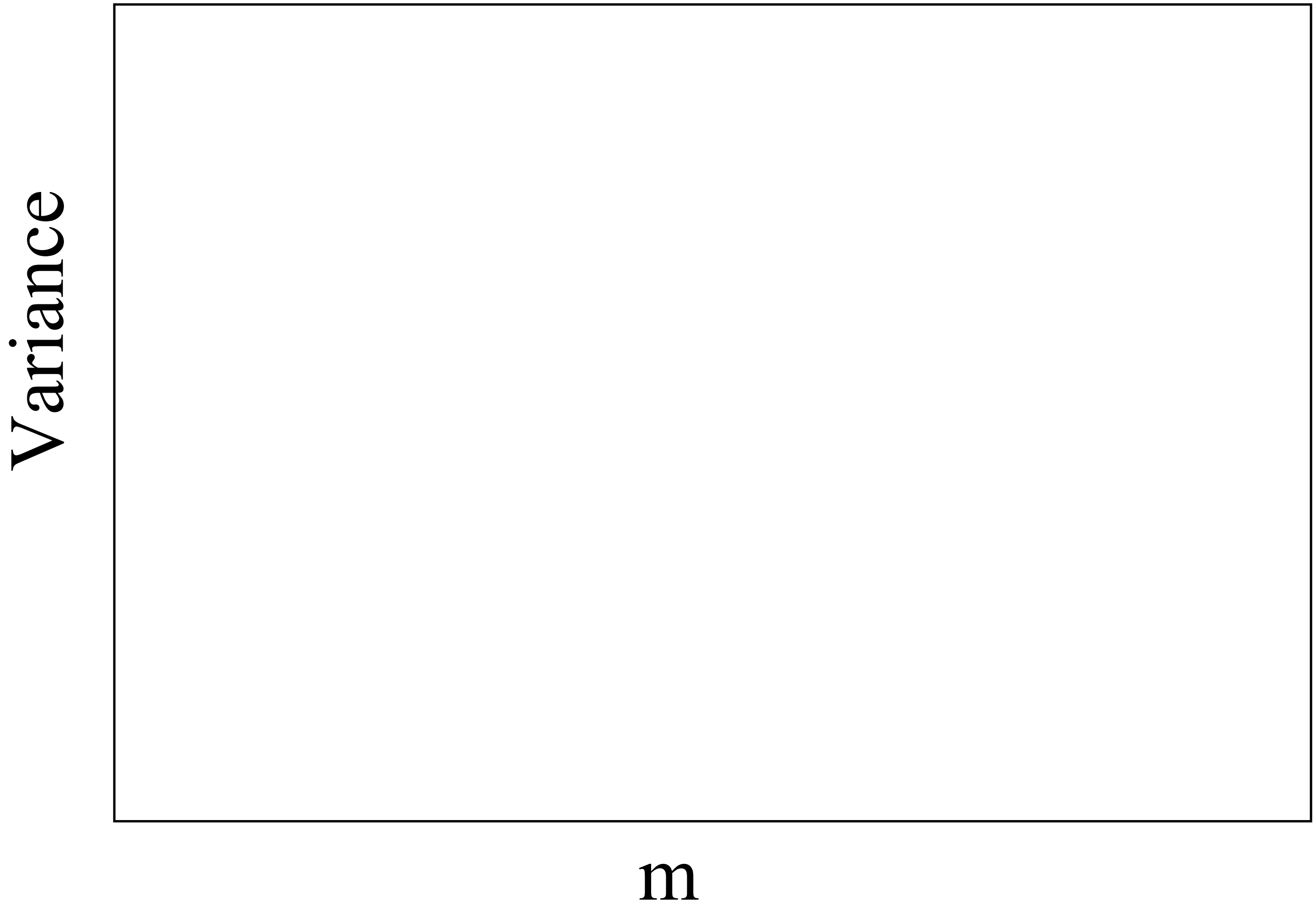


Variance of sample on a logarithmic scale

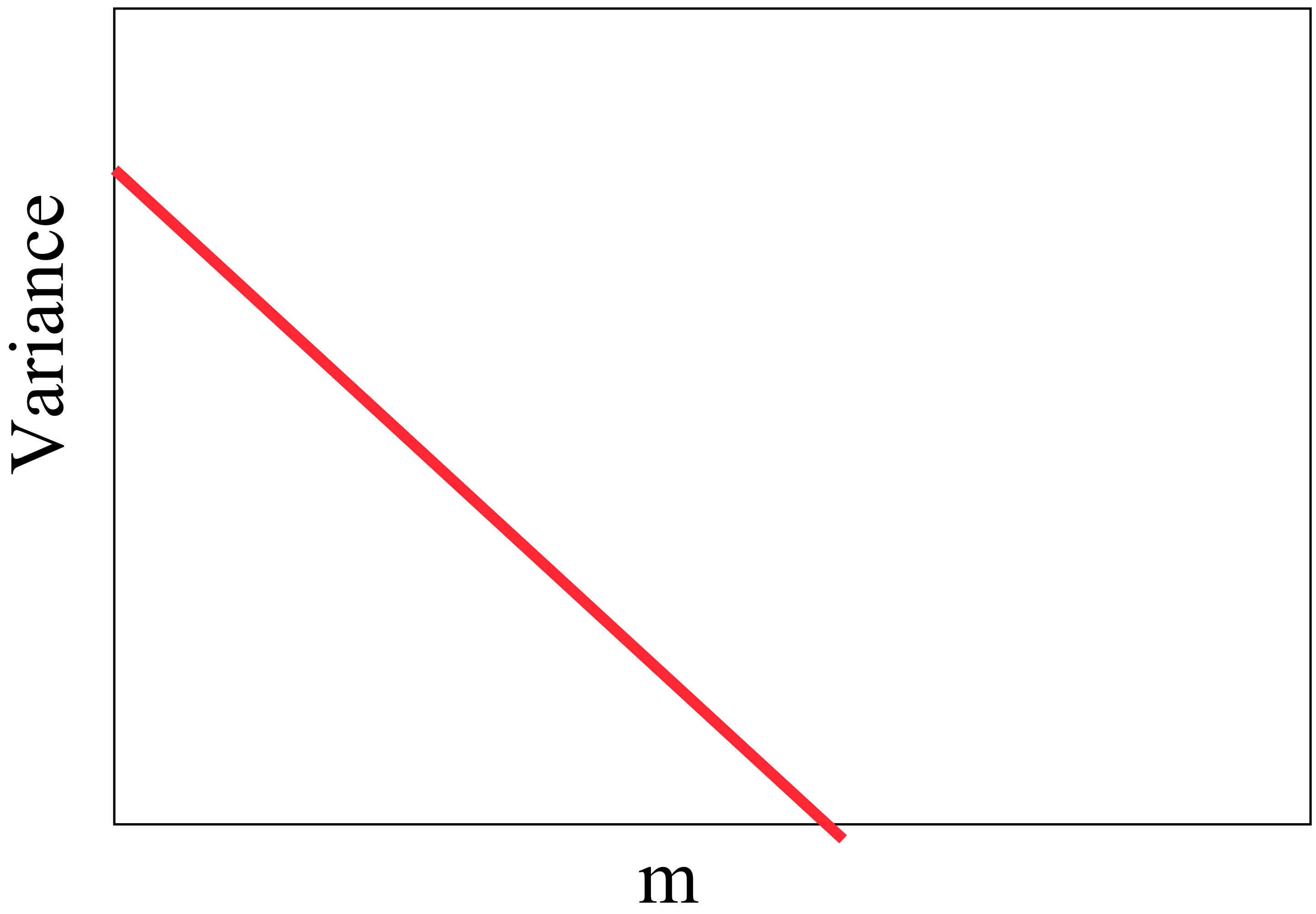
# Variance-Time Plot



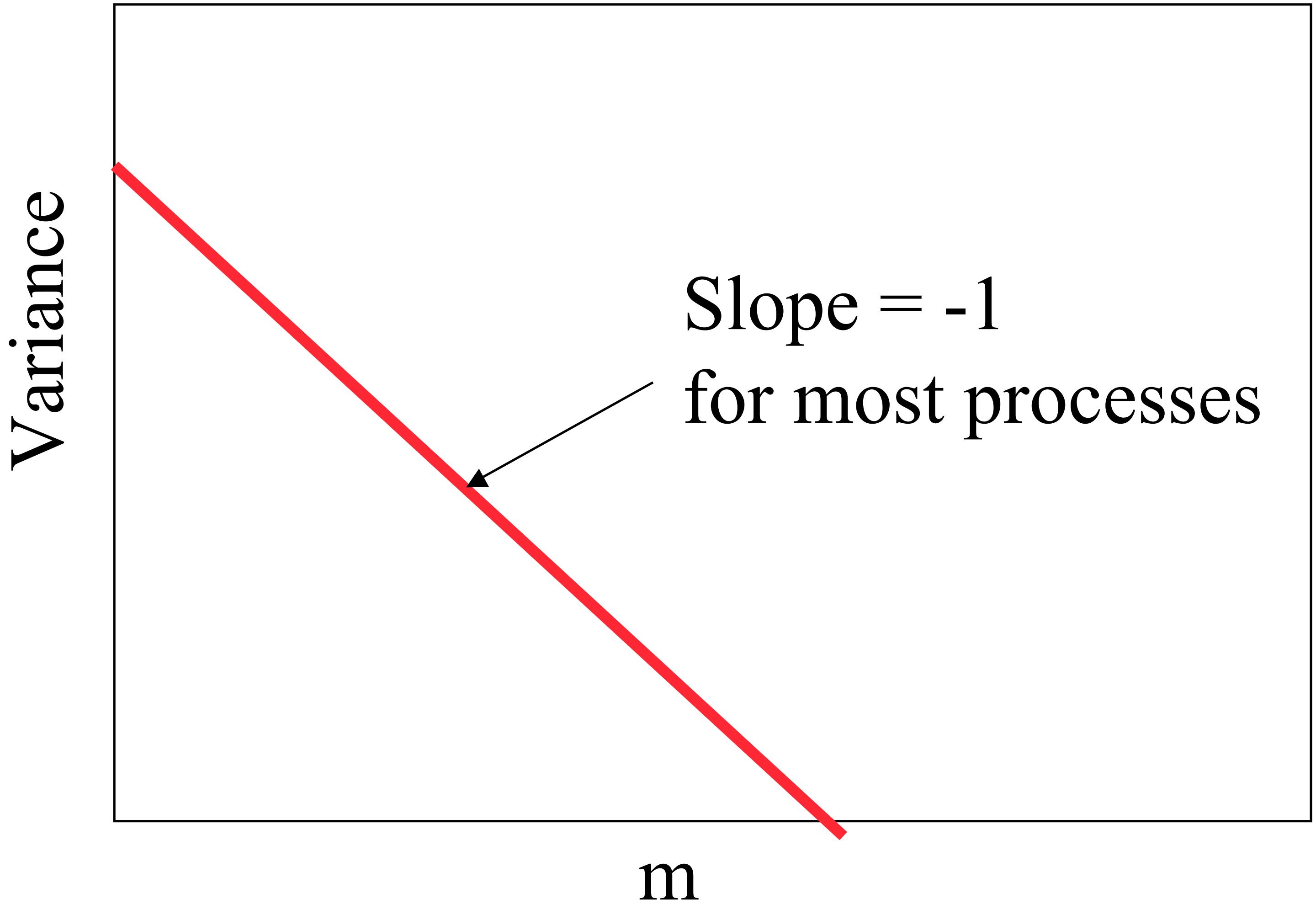
# Variance-Time Plot



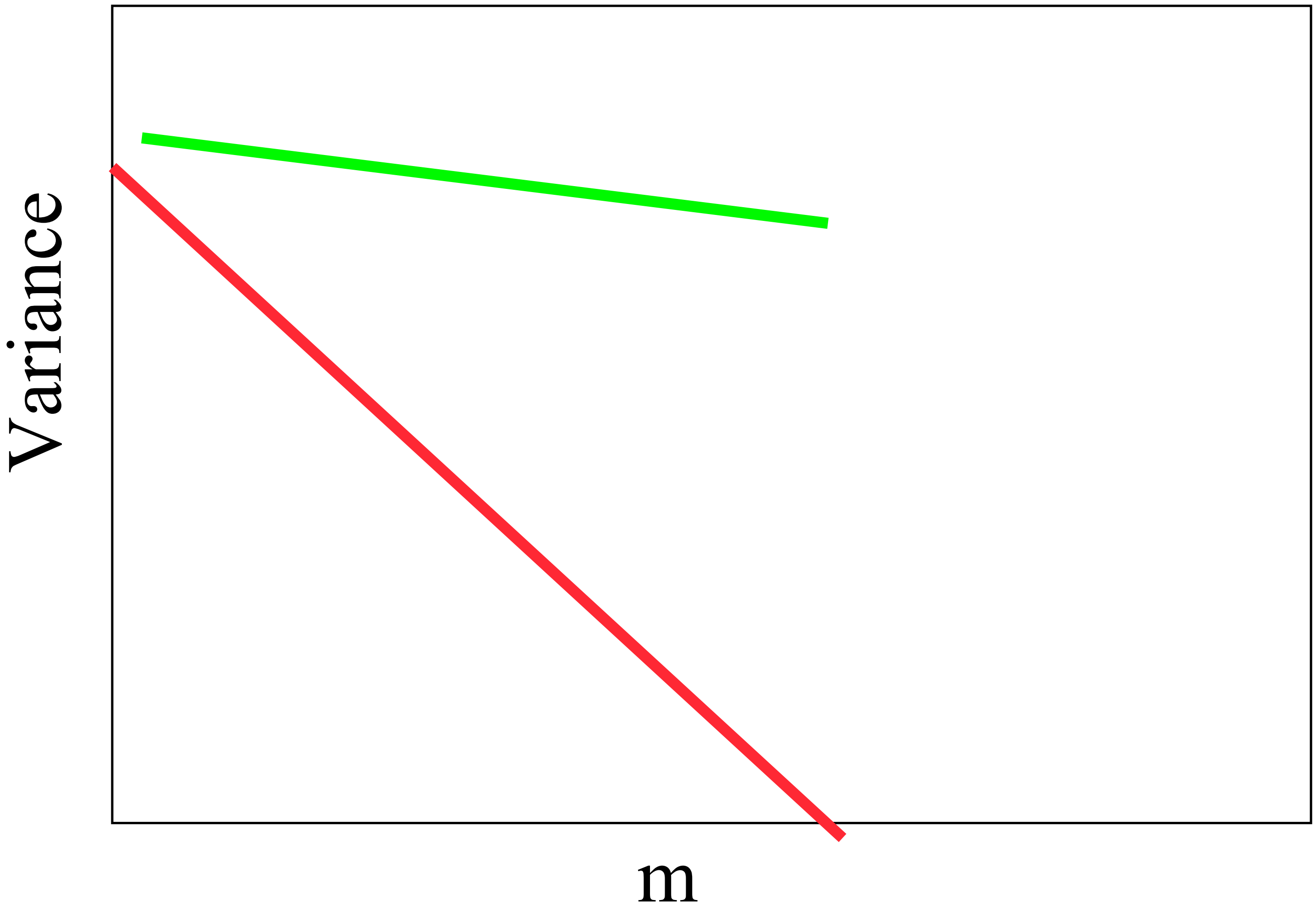
# Variance-Time Plot



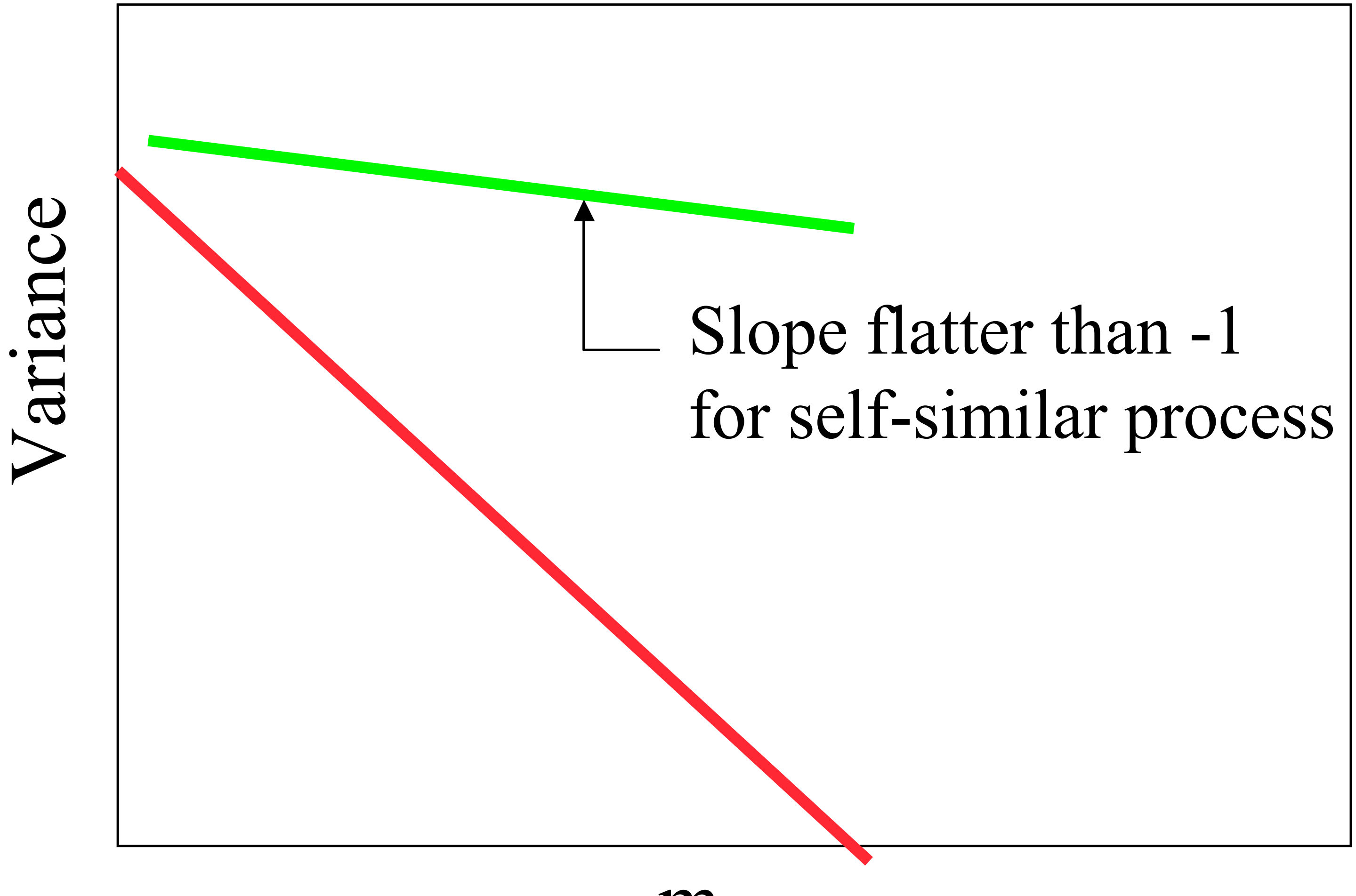
# Variance-Time Plot



# Variance-Time Plot



# Variance-Time Plot



# Long Range Dependence

- Correlation is a statistical measure of the relationship, if any, between two random variables
- Positive correlation: both behave similarly
- Negative correlation: behave as opposites
- No correlation: behavior of one is unrelated to behavior of other

# Long Range Dependence

- Autocorrelation is a statistical measure of the relationship, if any, between a random variable and itself, at different time lags
- Positive correlation: big observation usually followed by another big, or small by small
- Negative correlation: big observation usually followed by small, or small by big
- No correlation: observations unrelated

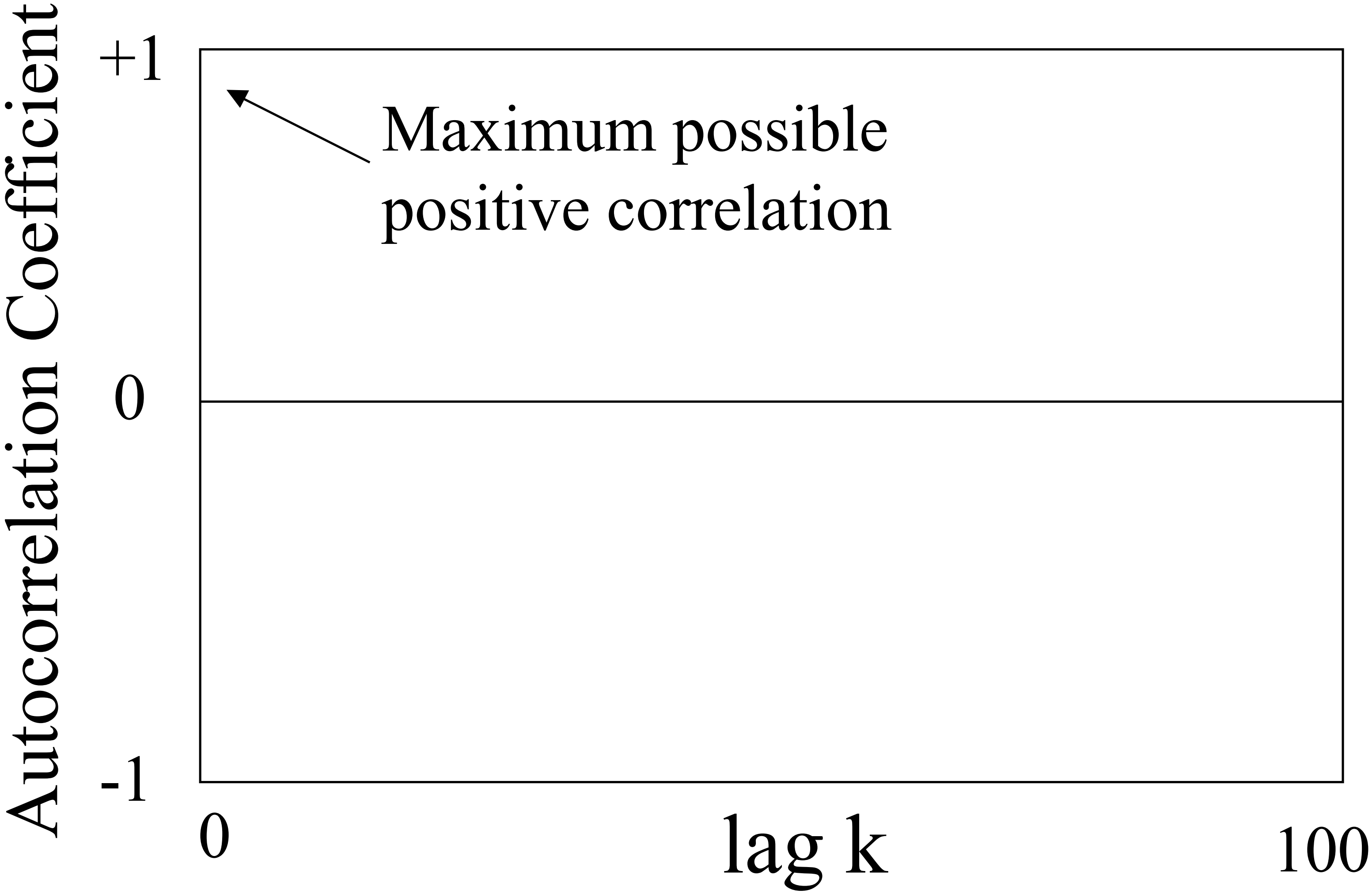
# Long Range Dependence

- Autocorrelation coefficient can range between:
  - +1 (very high positive correlation)
  - -1 (very high negative correlation)
- Zero means no correlation
- Autocorrelation function shows the value of the autocorrelation coefficient for different time lags k

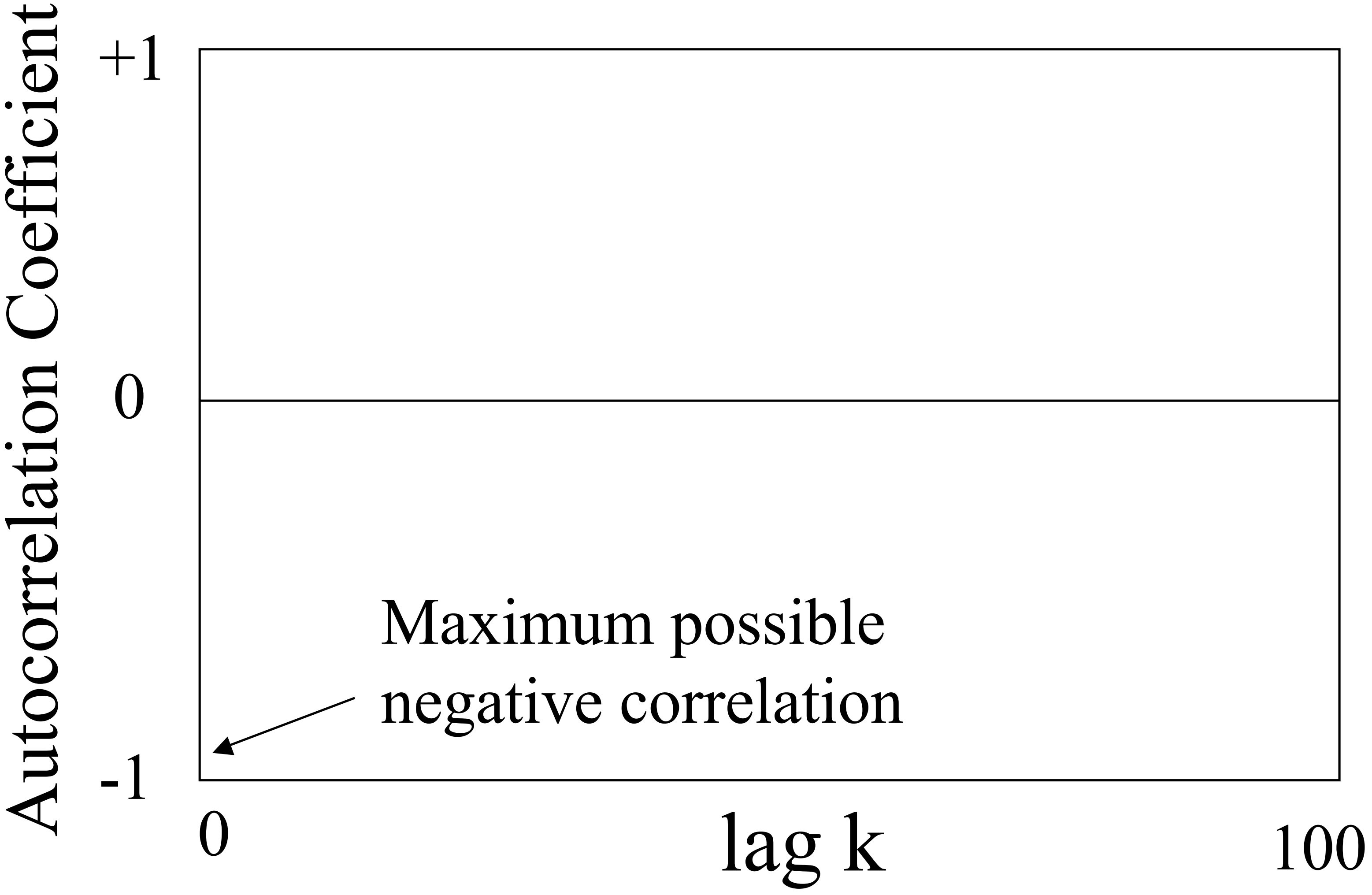
# Autocorrelation Function

Autocorrelation Coefficient
+1
0
-1
0
lag k
100

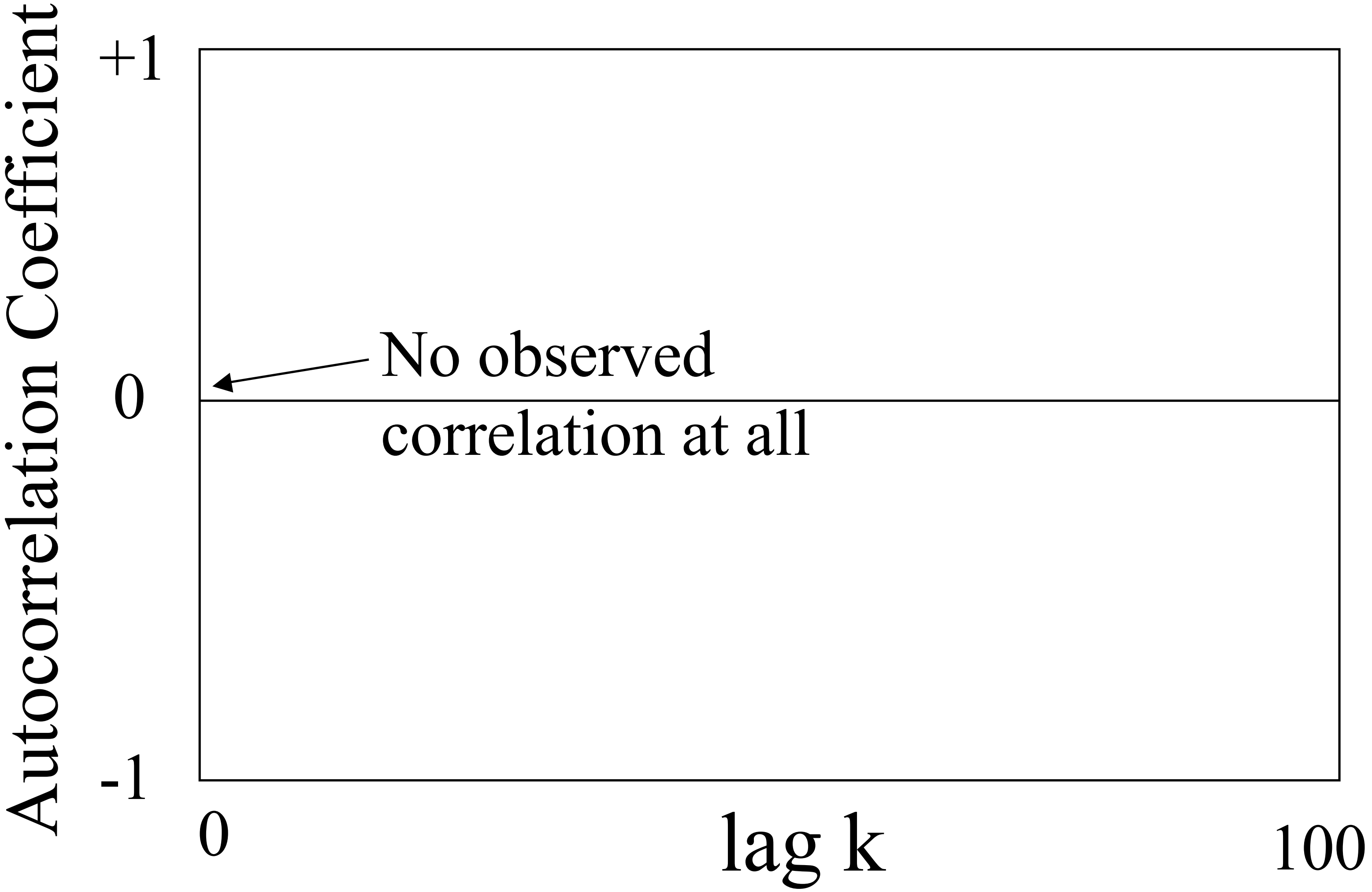
# Autocorrelation Function



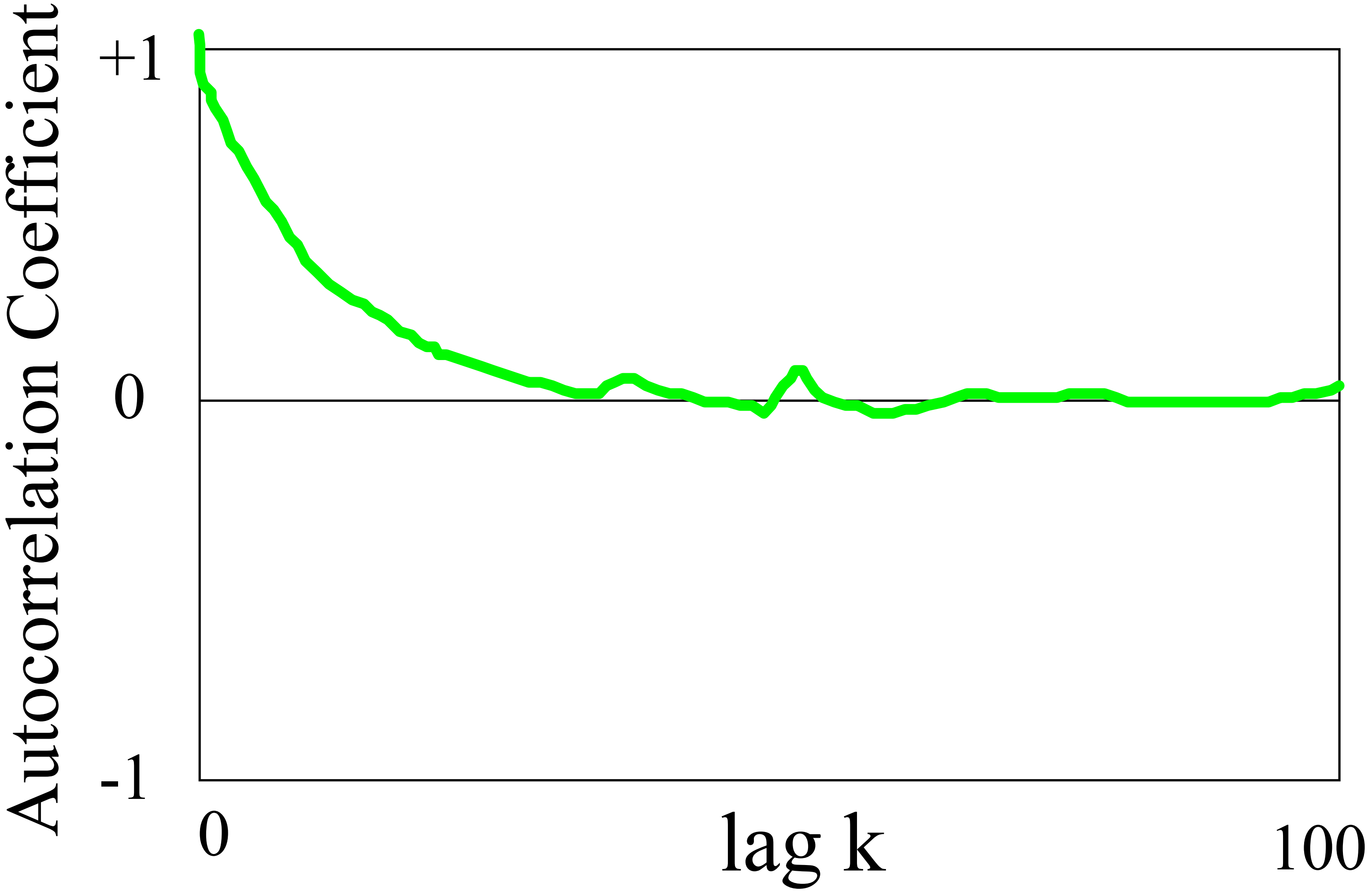
# Autocorrelation Function



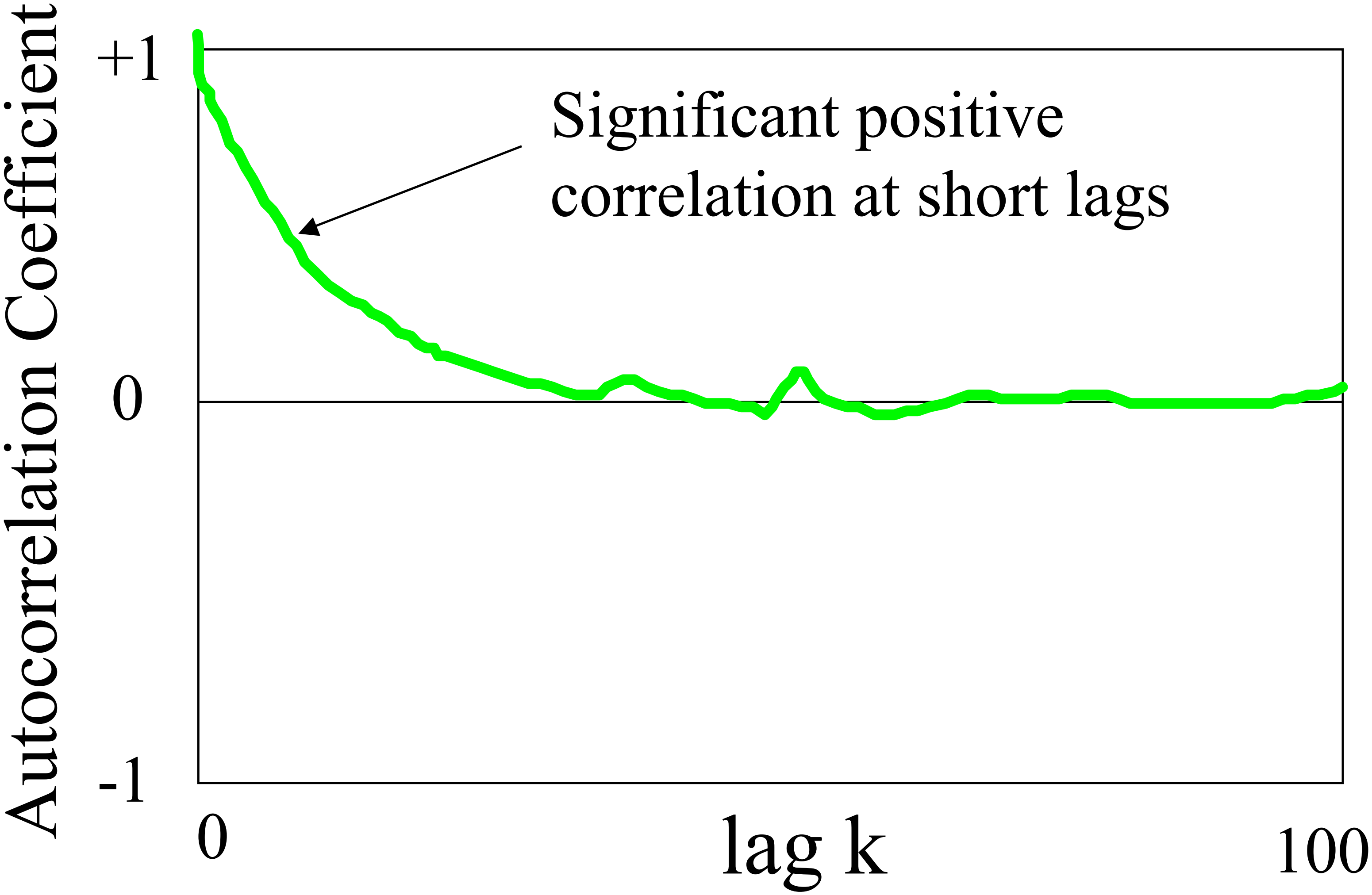
# Autocorrelation Function



# Autocorrelation Function

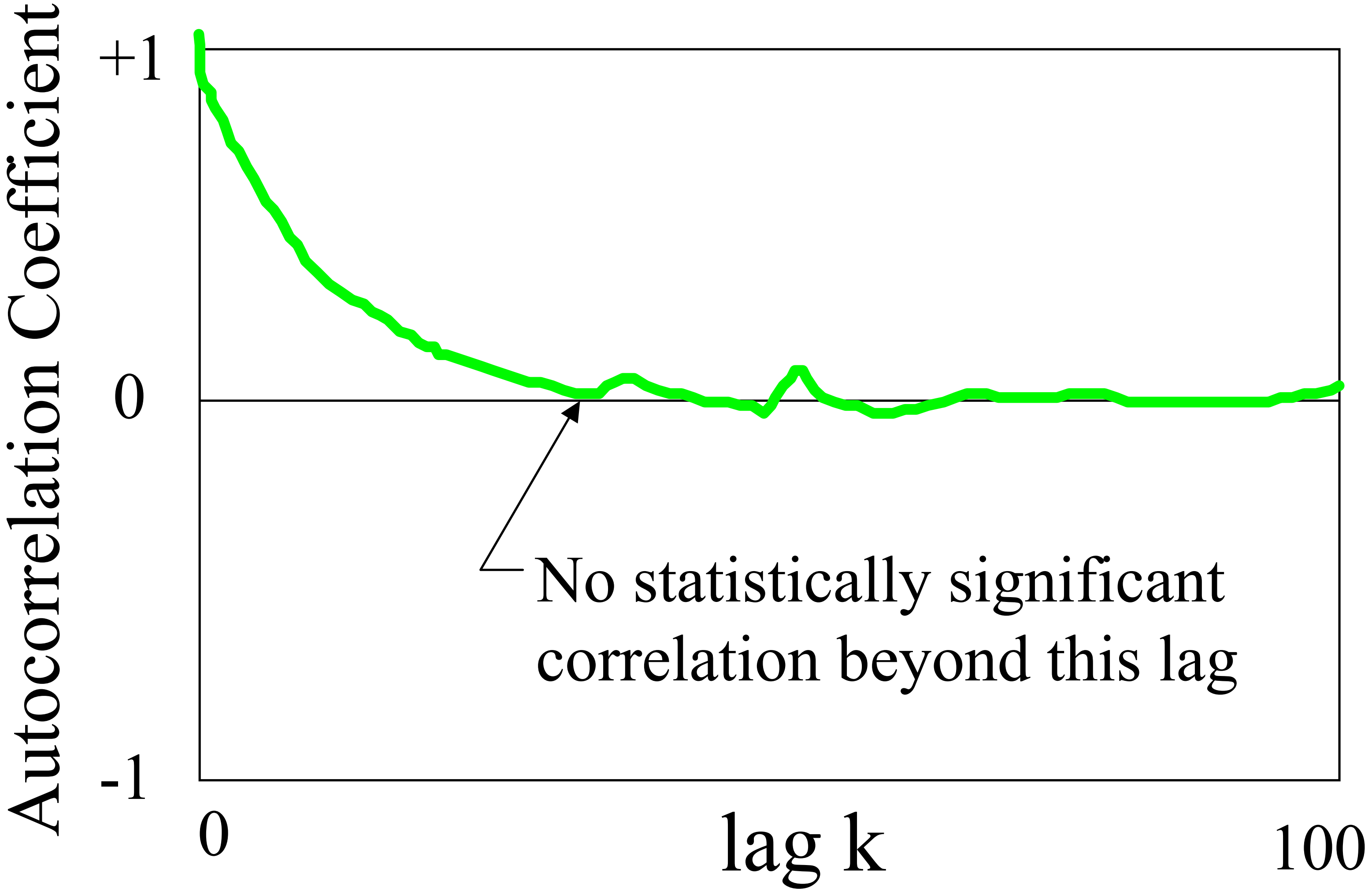


# Autocorrelation Function



Significant positive correlation at short lags

# Autocorrelation Function

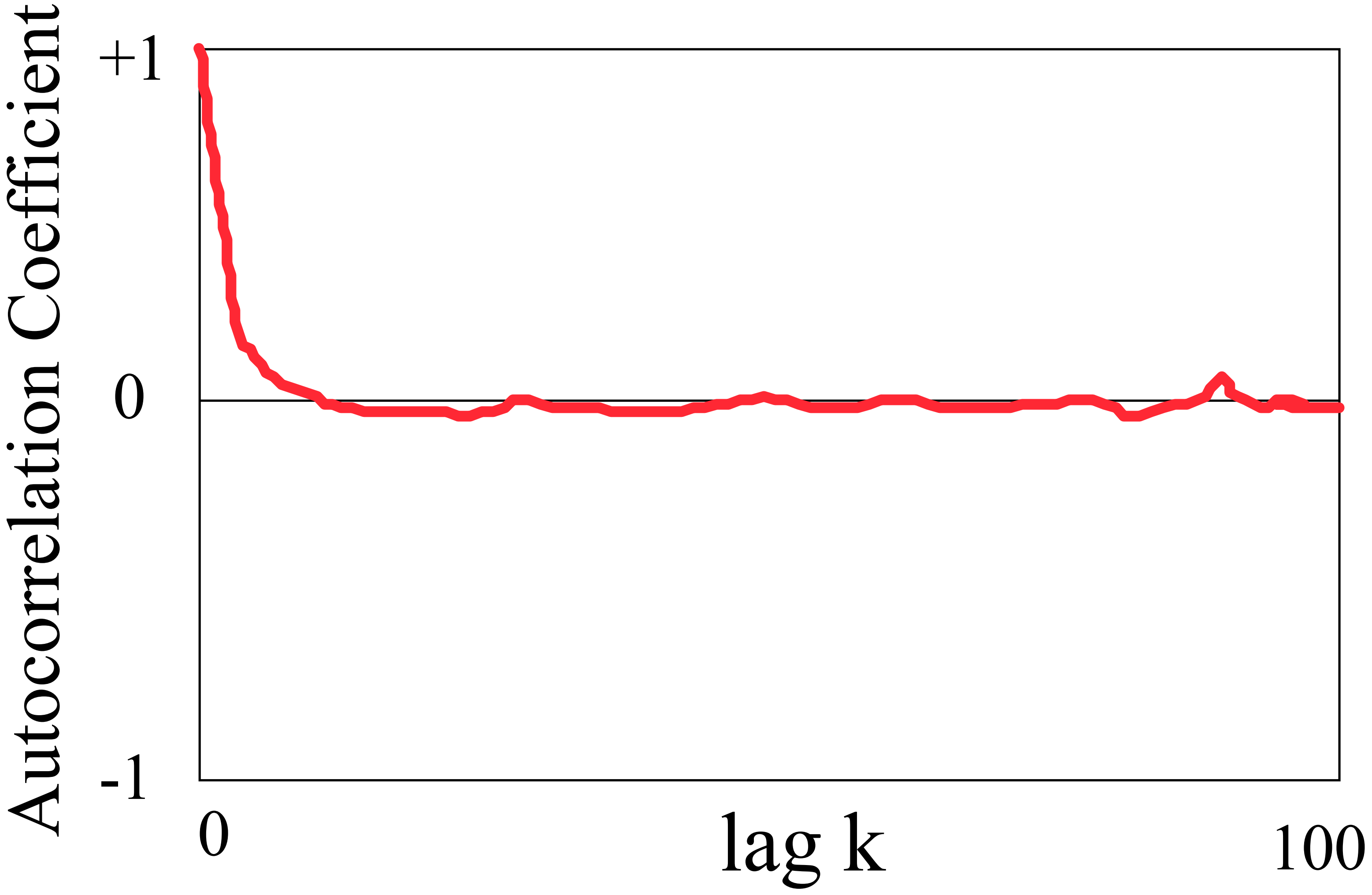


No statistically significant correlation beyond this lag

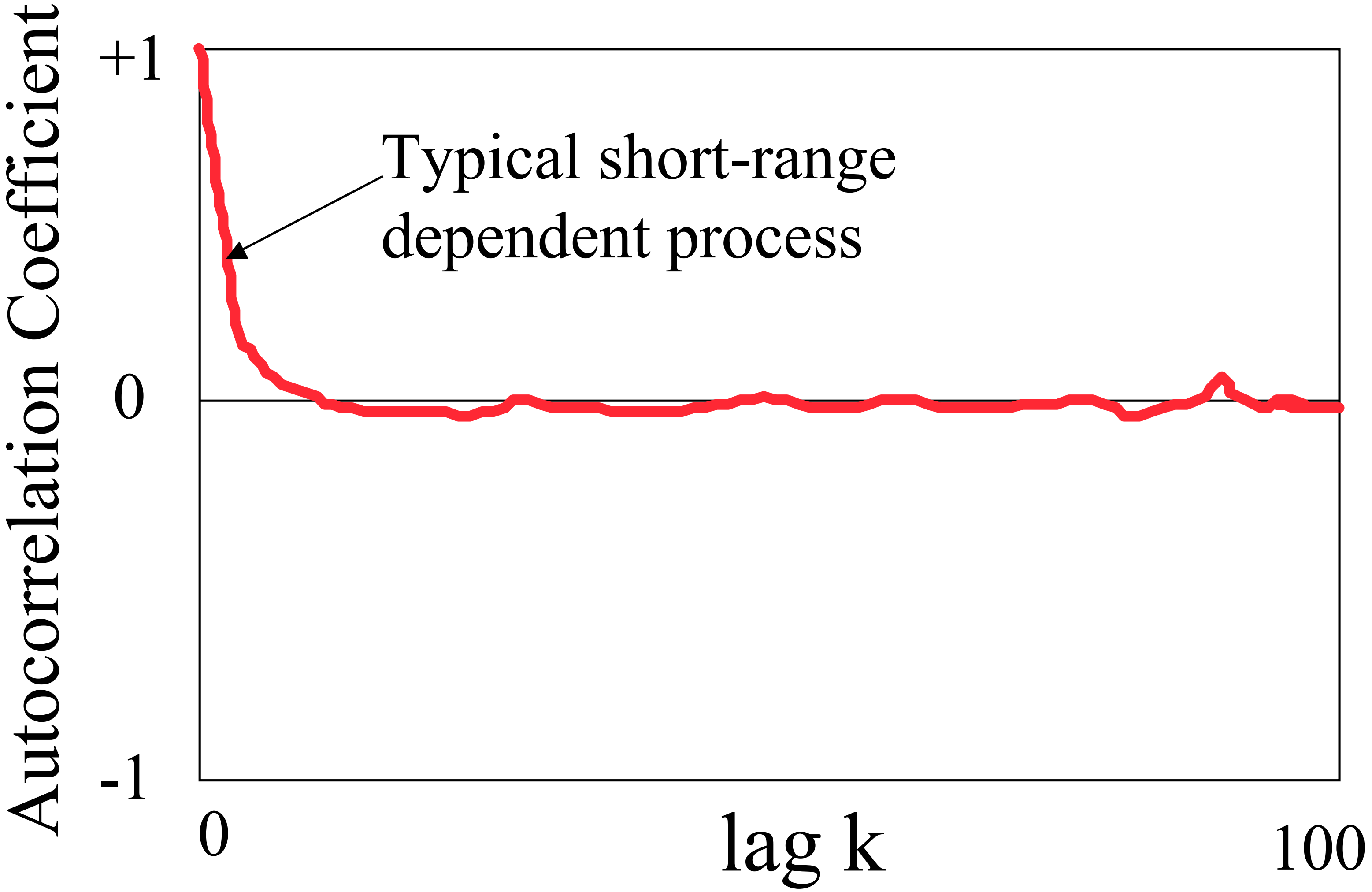
# Long Range Dependence

- For most processes (e.g., Poisson, or compound Poisson), the autocorrelation function drops to zero very quickly
  - usually immediately, or exponentially fast
- For self-similar processes, the autocorrelation function drops very slowly
  - i.e., hyperbolically, toward zero, but may never reach zero
- Non-summable autocorrelation function

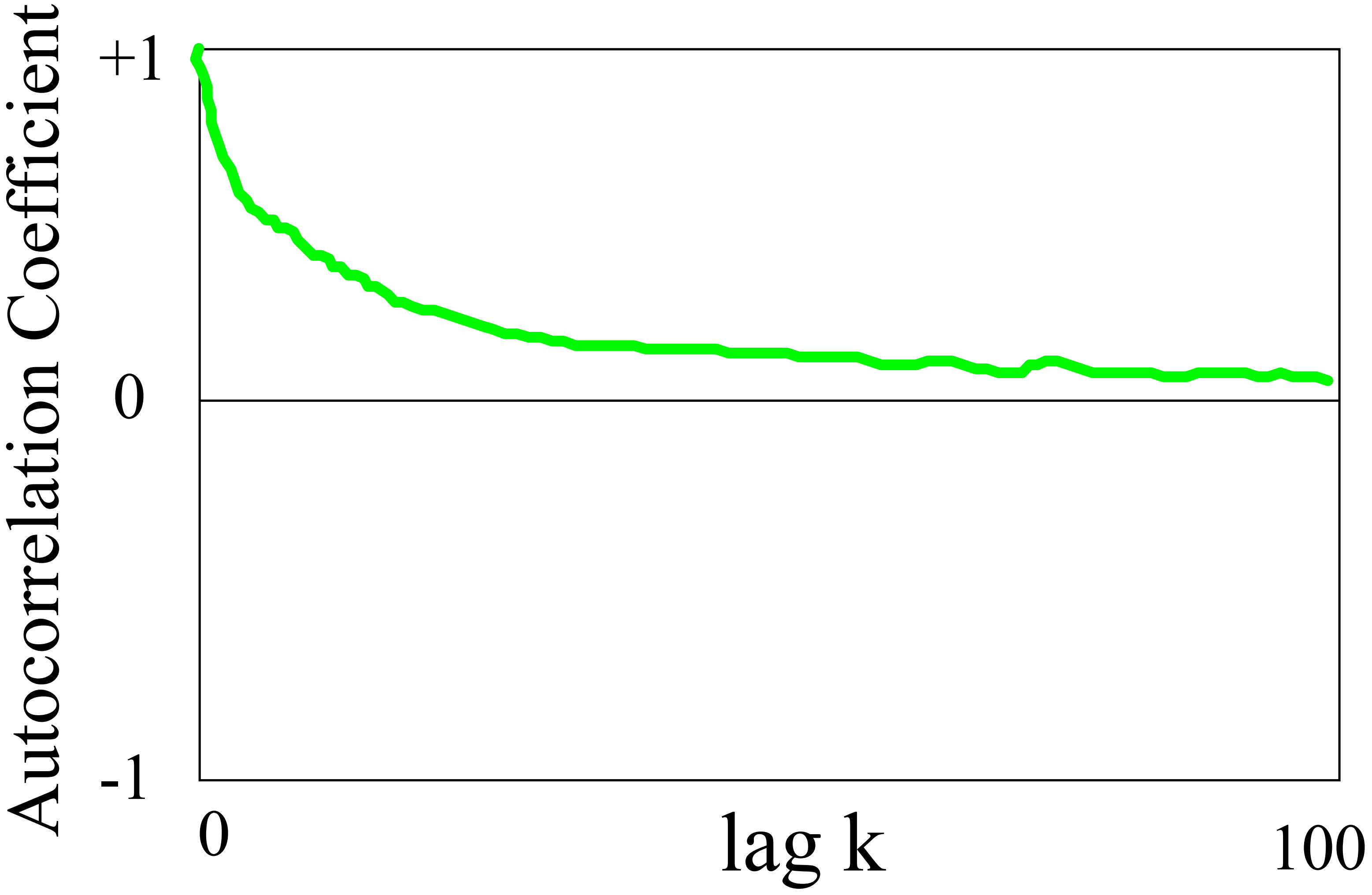
# Autocorrelation Function



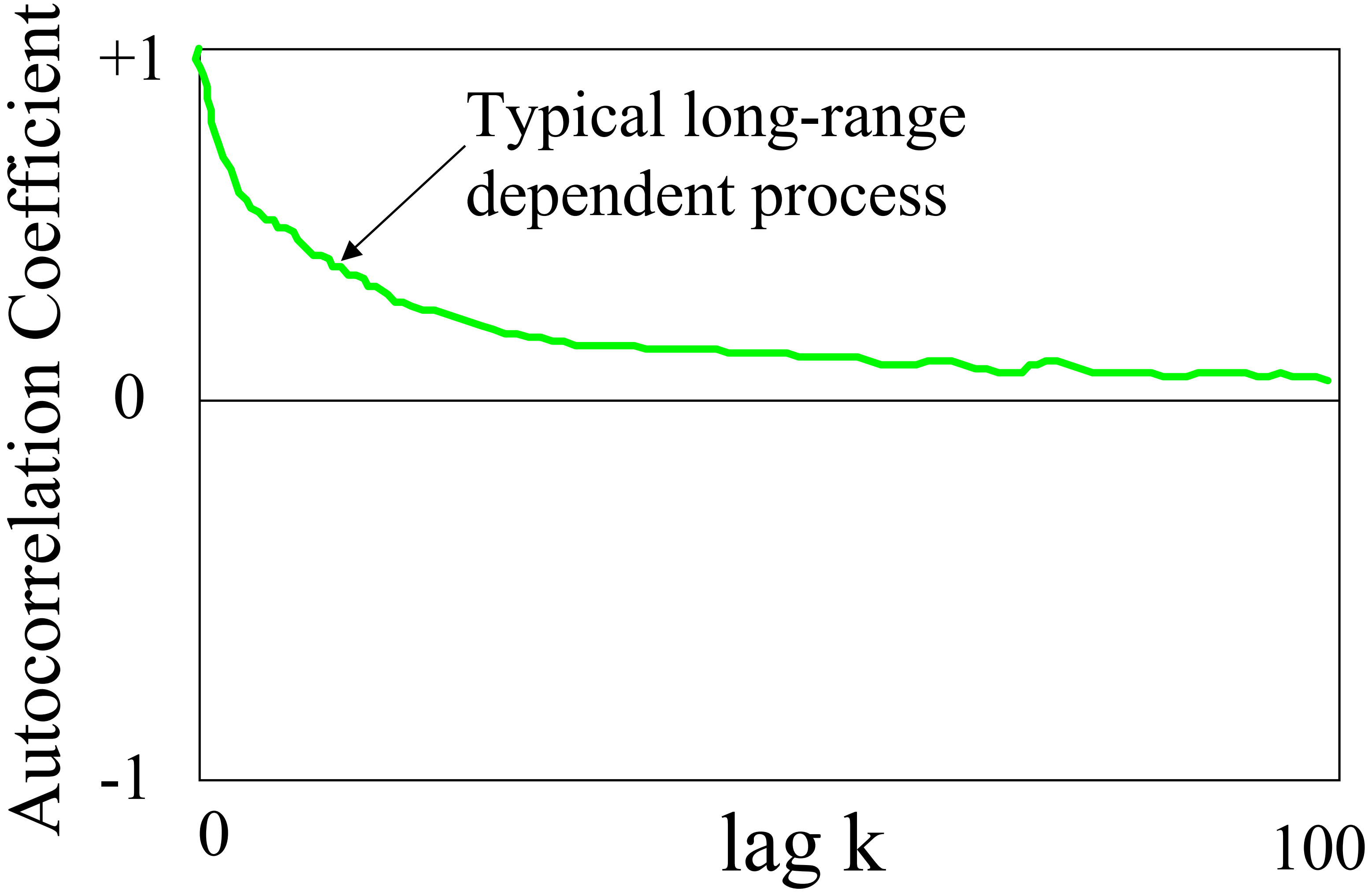
# Autocorrelation Function



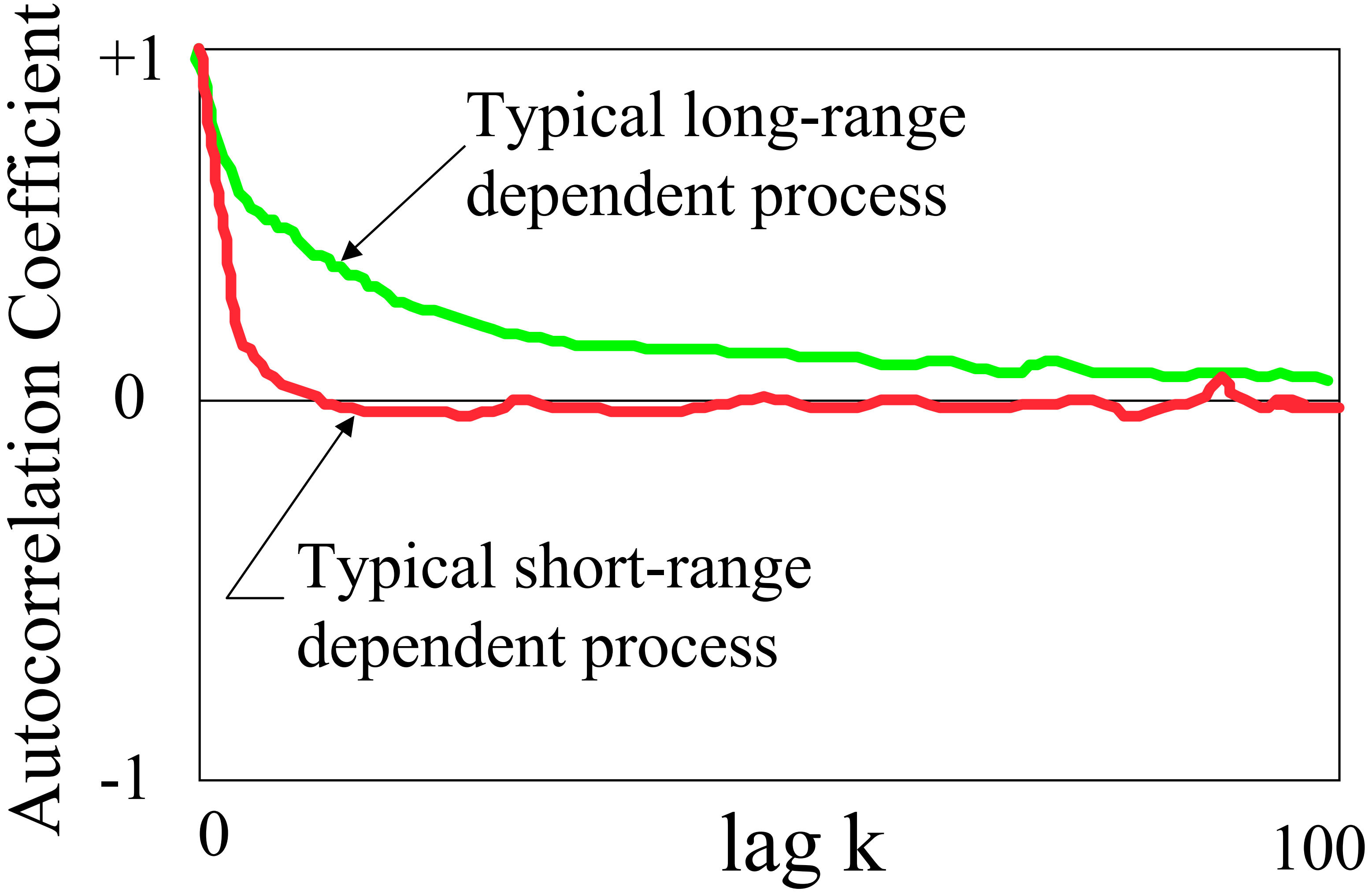
# Autocorrelation Function



# Autocorrelation Function



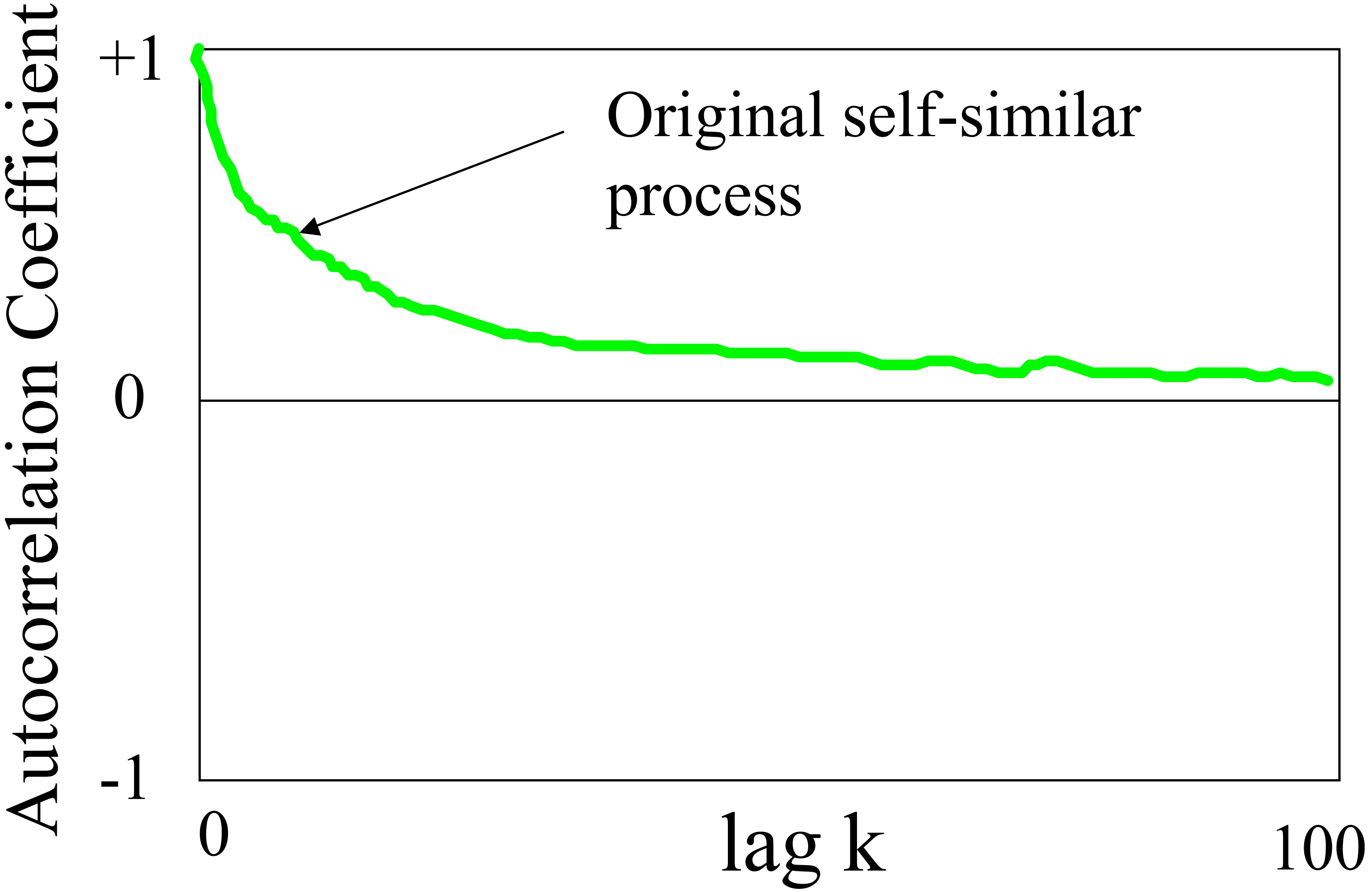
# Autocorrelation Function



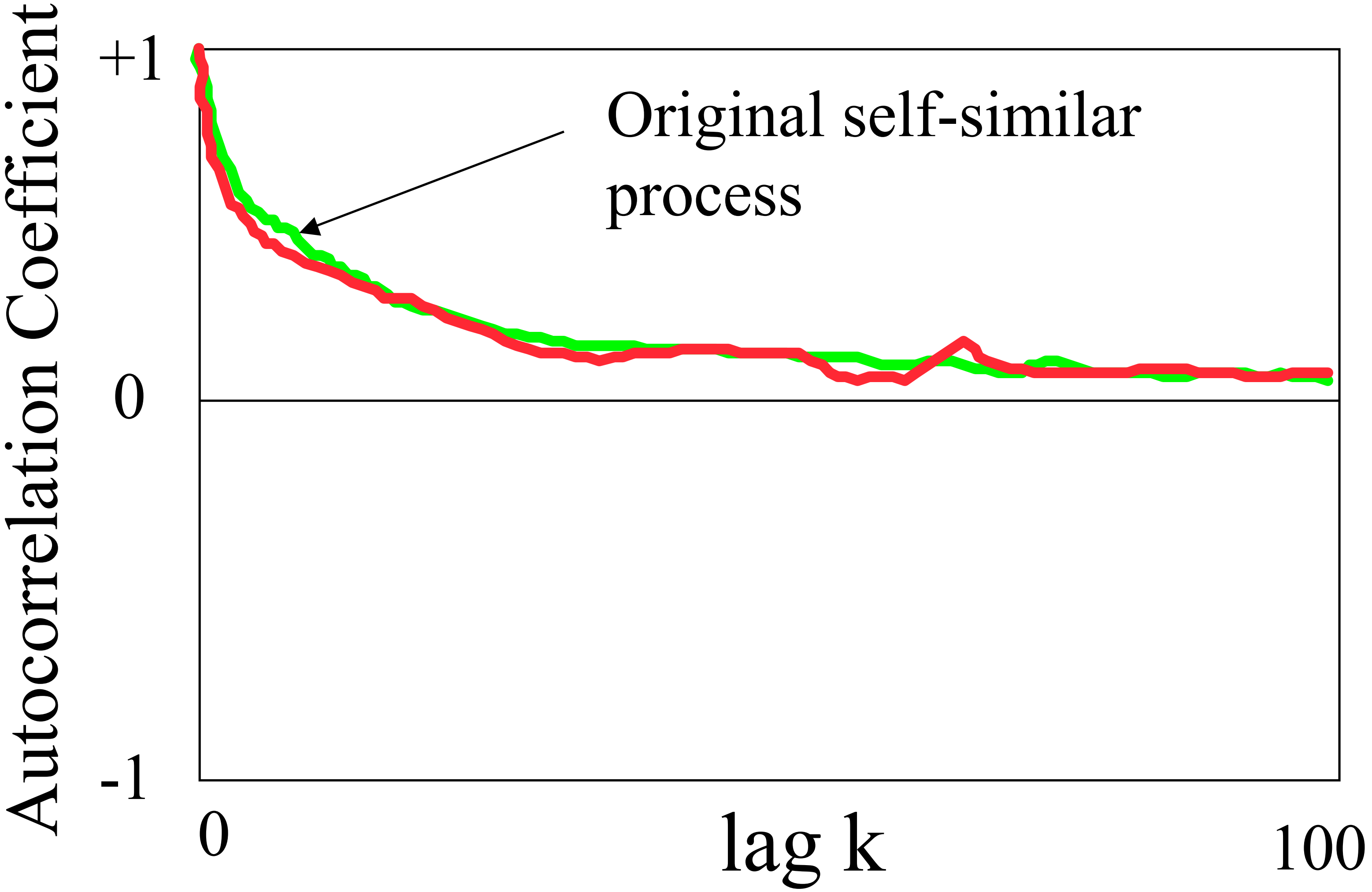
# Non-Degenerate Autocorrelations

- For self-similar processes, the autocorrelation function for the aggregated process is indistinguishable from that of the original process
- If autocorrelation coefficients match for all lags k, then called exactly self-similar
- If autocorrelation coefficients match only for large lags k, then called asymptotically self-similar

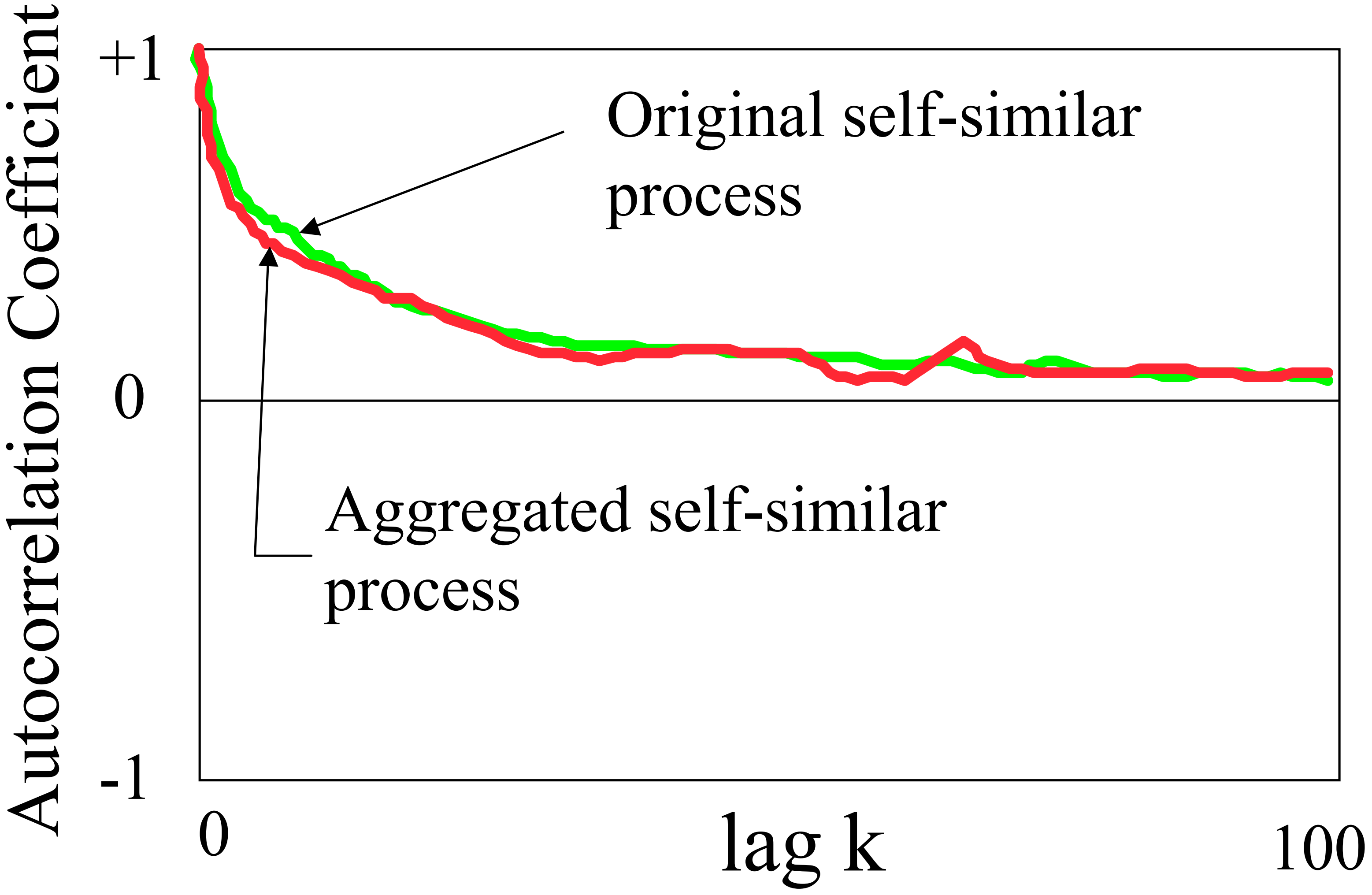
# Autocorrelation Function



# Autocorrelation Function



# Autocorrelation Function



# Aggregation

- Aggregation of a time series X(t) means smoothing the time series by averaging the observations over non-overlapping blocks of size m to get a new time series $X_m(t)$

# Aggregation Example

- Suppose the original time series X(t) contains the following (made up) values

```
2  7  4  12  5  0  8  2  8  4  6  9  11  3  3  5  7  2  9  1...
```

- Then the aggregated series for m = 2 is:

# Aggregation Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

- Then the aggregated series for m = 2 is:

# Aggregation Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

- Then the aggregated series for m = 2 is:

4.5

# Aggregation example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

- Then the aggregated series for m = 2 is:

4.5 8.0

# Aggregation Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

- Then the aggregated series for m = 2 is:

4.5 8.0 2.5

# Aggregation Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

- Then the aggregated series for m = 2 is:

4.5 8.0 2.5 5.0

# Aggregation Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

- Then the aggregated series for m = 2 is:

4.5 8.0 2.5 5.0 6.0 7.5 7.0 4.0 4.5 5.0...

# Aggregation Example

- Suppose the original time series X(t) contains the following (made up) values:

```
2  7  4  12  5  0  8  2  8  4  6  9  11  3  3  5  7  2  9  1...
```

Then the aggregated time series for m = 5 is:

# Aggregation: An Example

- Suppose the original time series X(t) contains the following (made up) values:

```
2  7  4  12  5  0  8  2  8  4  6  9  11  3  3  5  7  2  9  1...
```

Then the aggregated time series for m = 5 is:

# Aggregation: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

Then the aggregated time series for m = 5 is:

6.0

# Aggregation: An Example

- Suppose the original time series X(t) contains the following (made up) values:

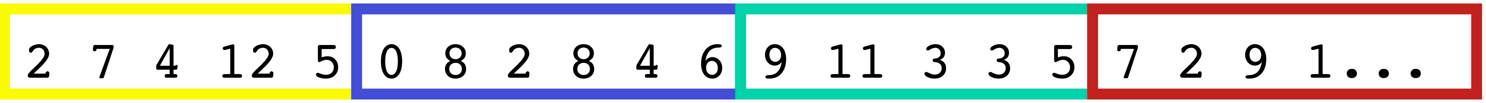
2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...

Then the aggregated time series for m = 5 is:

6.0 4.4

# Aggregation: An Example

- Suppose the original time series X(t) contains the following (made up) values:



```
2  7  4  12  5 | 0  8  2  8  4  6 | 9  11  3  3  5 | 7  2  9  1...
```

Then the aggregated time series for m = 5 is:

6.0      4.4      6.4      4.8 ...

# Aggregation: An Example

- Suppose the original time series X(t) contains the following (made up) values:

```
2  7  4  12  5  0  8  2  8  4  6  9  11  3  3  5  7  2  9  1...
```

Then the aggregated time series for m = 10 is:

# Aggregation: An Example

- Suppose the original time series X(t) contains the following (made up) values:

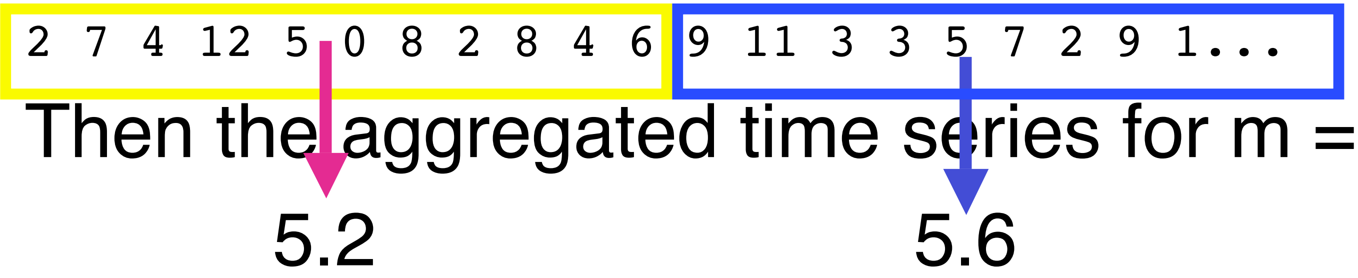
```
2  7  4  12  5  0  8  2  8  4  6  9  11  3  3  5  7  2  9  1...
```

Then the aggregated time series for m = 10 is:

5.2

# Aggregation: An Example

- Suppose the original time series X(t) contains the following (made up) values:

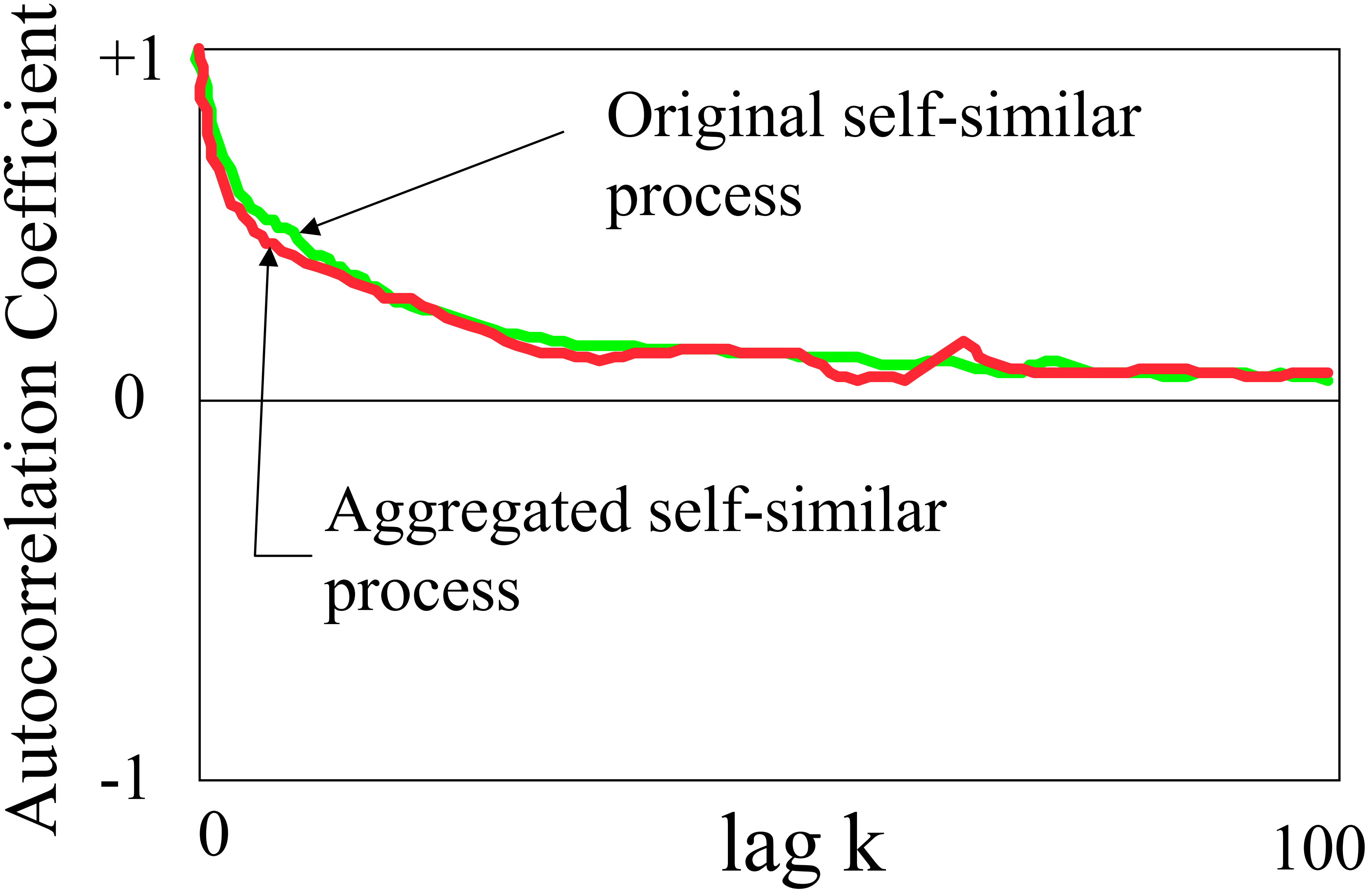


```
2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1...
```

Then the aggregated time series for m = 10 is:

5.2 5.6

# Autocorrelation Function



# Hurst Effect

- For almost all naturally occurring time series, the rescaled adjusted range statistic (also called the R/S statistic) for sample size n obeys the relationship

$$E[R(n)/S(n)] = c\, n^{H}$$

where:

$R(n) = \max(0, W_1, \ldots W_n) - \min(0, W_1, \ldots W_n)$

$S^2(n)$ is the sample variance, and

$$W_K = \sum_{i=1}^{n} (X_i) - k\overline{X}_n \qquad \text{for } k = 1, 2, \ldots n$$

# Hurst Effect

- For models with only short range dependence, H is almost always 0.5
- For self-similar processes, $0.5 < H < 1.0$
- This discrepancy is called the Hurst Effect, and H is called the Hurst parameter
- Single parameter to characterize self-similar processes

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- There are 20 data points in this example

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- There are 20 data points in this example
- For R/S analysis with n = 1, you get 20 samples, each of size 1:

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- There are 20 data points in this example
- For R/S analysis with n = 1, you get 20 samples, each of size 1:

Block 1: X  = 2, W = 0, R(n) = 0, S(n) = 0

—

n 1

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- There are 20 data points in this example
- For R/S analysis with n = 1, you get 20 samples, each of size 1:

Block 2: X = 7, W = 0, R(n) = 0, S(n) = 0

—

n 1

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 2, you get 10 samples, each of size 2:

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 2, you get 10 samples, each of size 2:

Block 1: $\overline{X}_n$ = 4.5, $W_1$ = -2.5, $W_2$ = 0,

R(n) = 0 - (-2.5) = 2.5, S(n) = 2.5,

R(n)/S(n) = 1.0

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 2, you get 10 samples, each of size 2:

Block 2: $\bar{X}_n = 8.0$, $W_1 = -4.0$, $W_2 = 0$,

R(n) = 0 - (-4.0) = 4.0, S(n) = 4.0,

R(n)/S(n) = 1.0

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 3, you get 6 samples, each of size 3:

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 3, you get 6 samples, each of size 3:

Block 1: $\bar{X}_n = 4.3$, $W_1 = -2.3$, $W_2 = 0.3$, $W_3 = 0$

R(n) = 0.3 - (-2.3) = 2.6, S(n) = 2.05,

R(n)/S(n) = 1.30

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 3, you get 6 samples, each of size 3:

Block 2: $X_n = 5.7$, $W_1 = 6.3$, $W_2 = 5.7$, $W_3 = 0$

R(n) = 6.3 - (0) = 6.3, S(n) = 4.92,

R(n)/S(n) = 1.28

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 5, you get 4 samples, each of size 5:

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 5, you get 4 samples, each of size 4:

Block 1: $\bar{X}_n$ = 6.0, $W_1$ = -4.0, $W_2$ = -3.0,

$W_3$ = -5.0 , $W_4$ = 1.0 , $W_5$ = 0, S(n) = 3.41,

R(n) = 1.0 - (-5.0) = 6.0, R(n)/S(n) = 1.76

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 5, you get 4 samples, each of size 4:

Block 2: $\bar{X}_n$ = 4.4, $W_1$ = -4.4, $W_2$ = -0.8,

$W_3$ = -3.2 , $W_4$ = 0.4 , $W_5$ = 0, S(n) = 3.2,

R(n) = 0.4 - (-4.4) = 4.8, R(n)/S(n) = 1.5

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 10, you get 2 samples, each of size 10:

# R/S Statistic: An Example

- Suppose the original time series X(t) contains the following (made up) values:

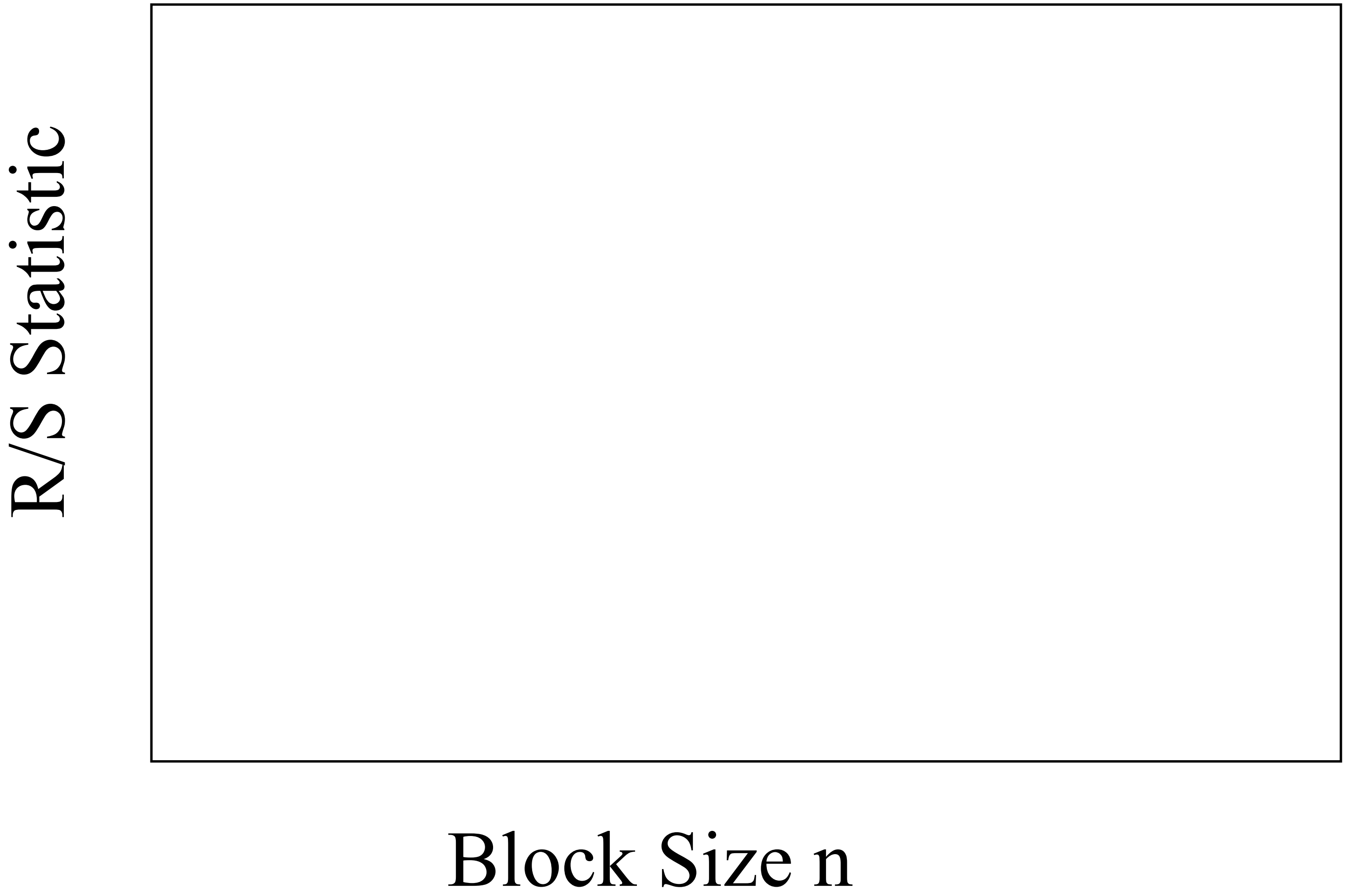
2 7 4 12 5 0 8 2 8 4 6 9 11 3 3 5 7 2 9 1

- For R/S analysis with n = 20, you get 1 sample of size 20:

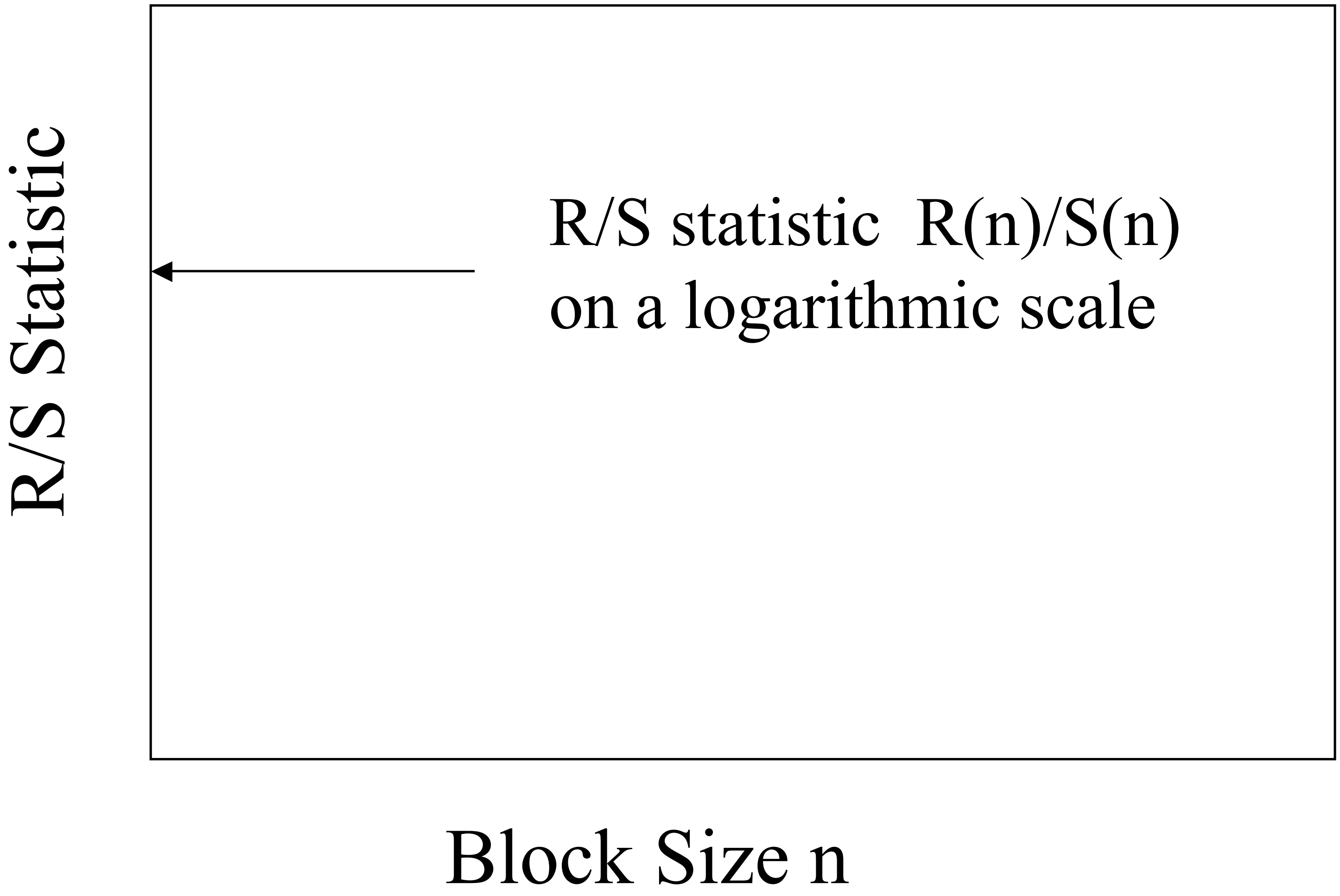
# R/S Plot

- Another way of testing for self-similarity, and estimating the Hurst parameter
- Plot the R/S statistic for different values of n, with a log scale on each axis
- If time series is self-similar, the resulting plot will have a straight line shape with a slope H that is greater than 0.5
- Called an R/S plot, or R/S pox diagram

# R/S Pox Diagram



# R/S Pox Diagram

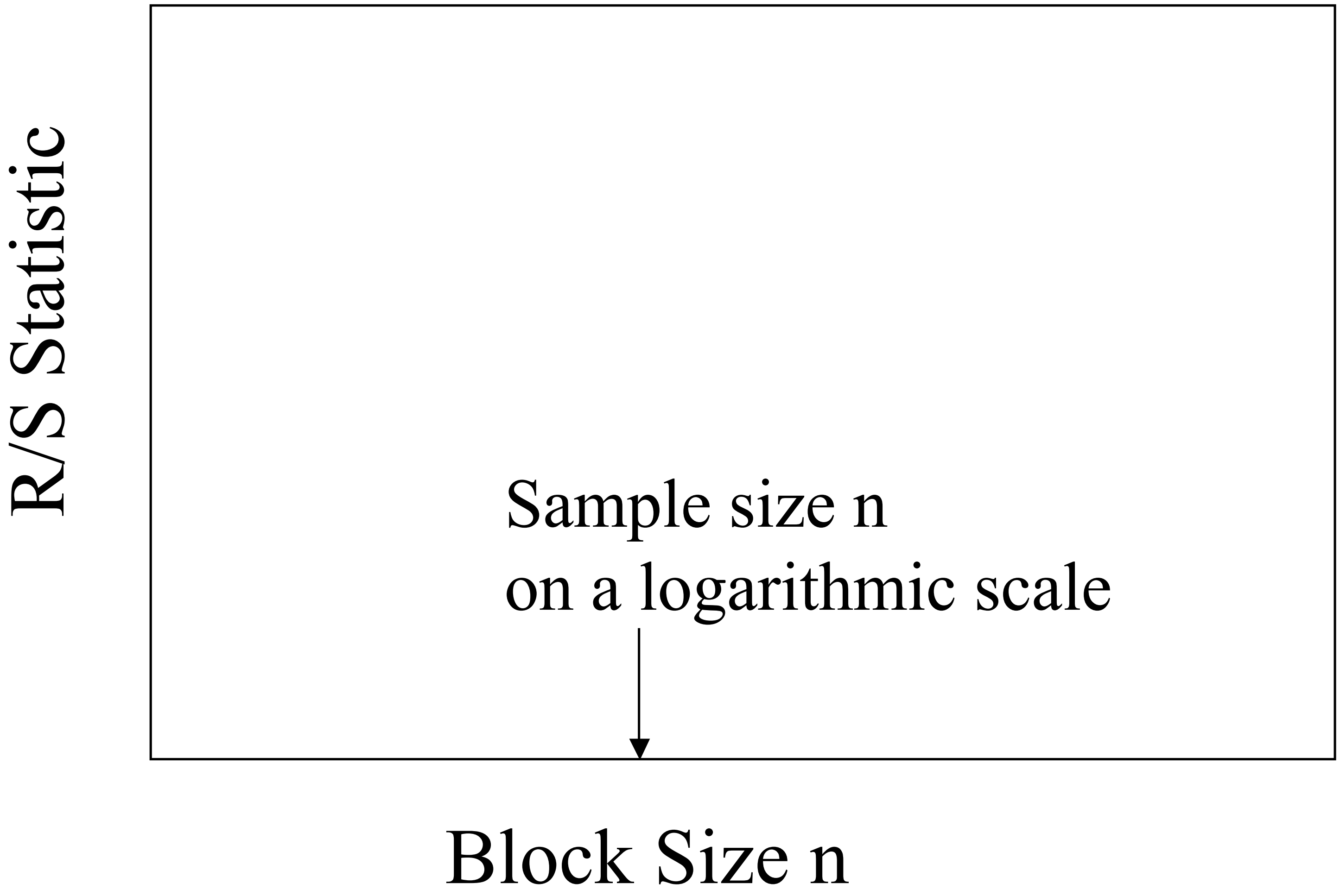


R/S Statistic

R/S statistic R(n)/S(n)
on a logarithmic scale

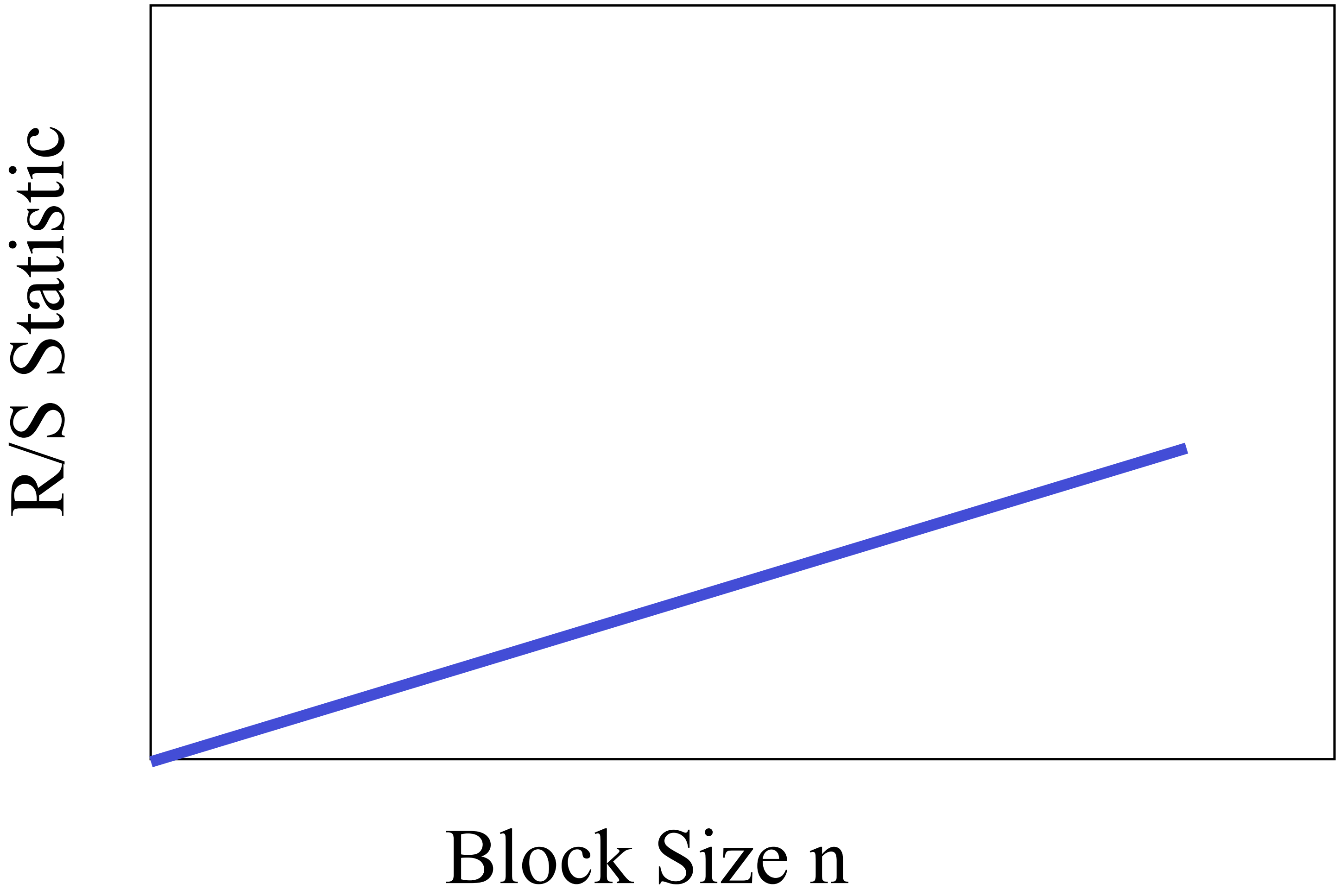
Block Size n

# R/S Pox Diagram

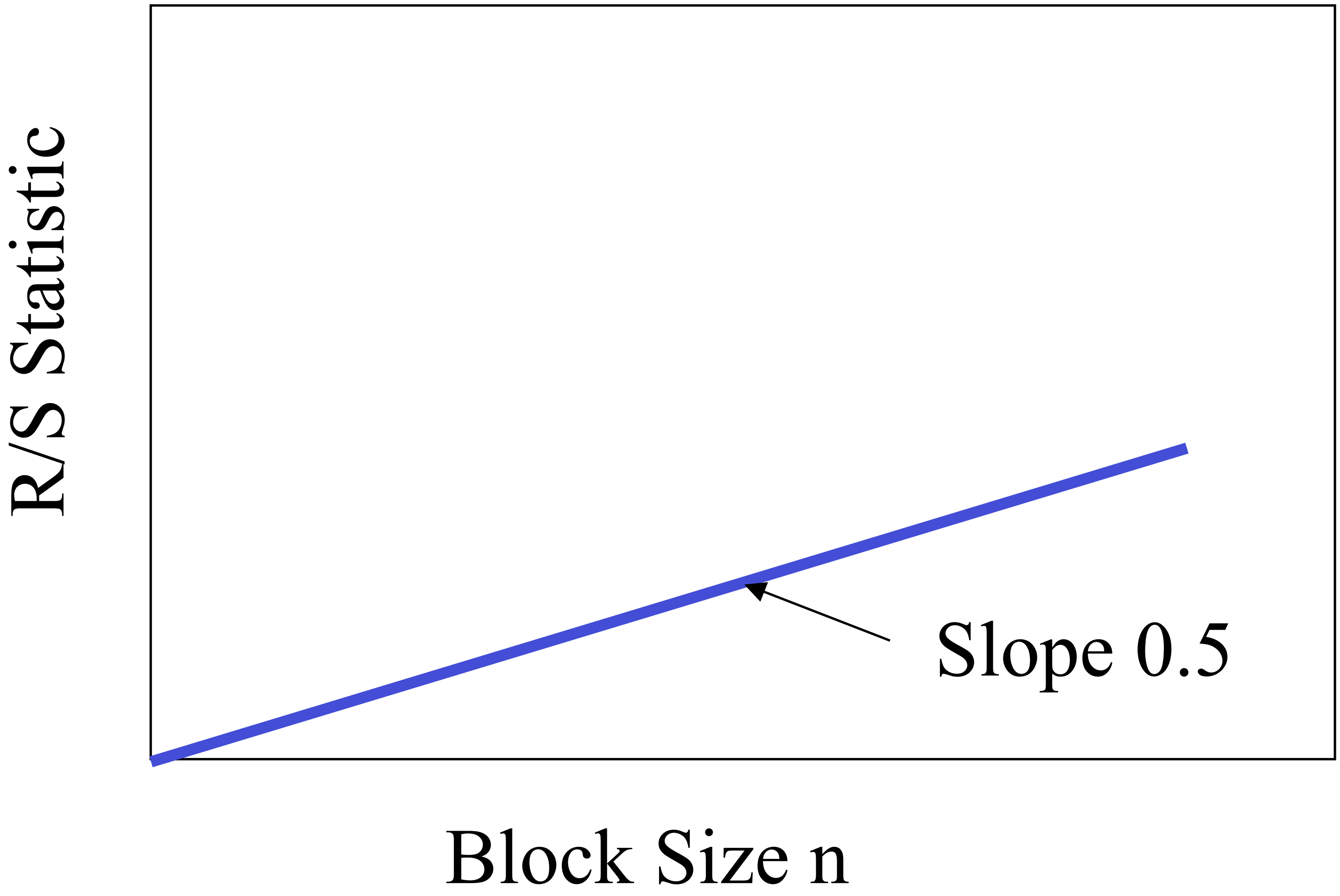


Sample size n
on a logarithmic scale

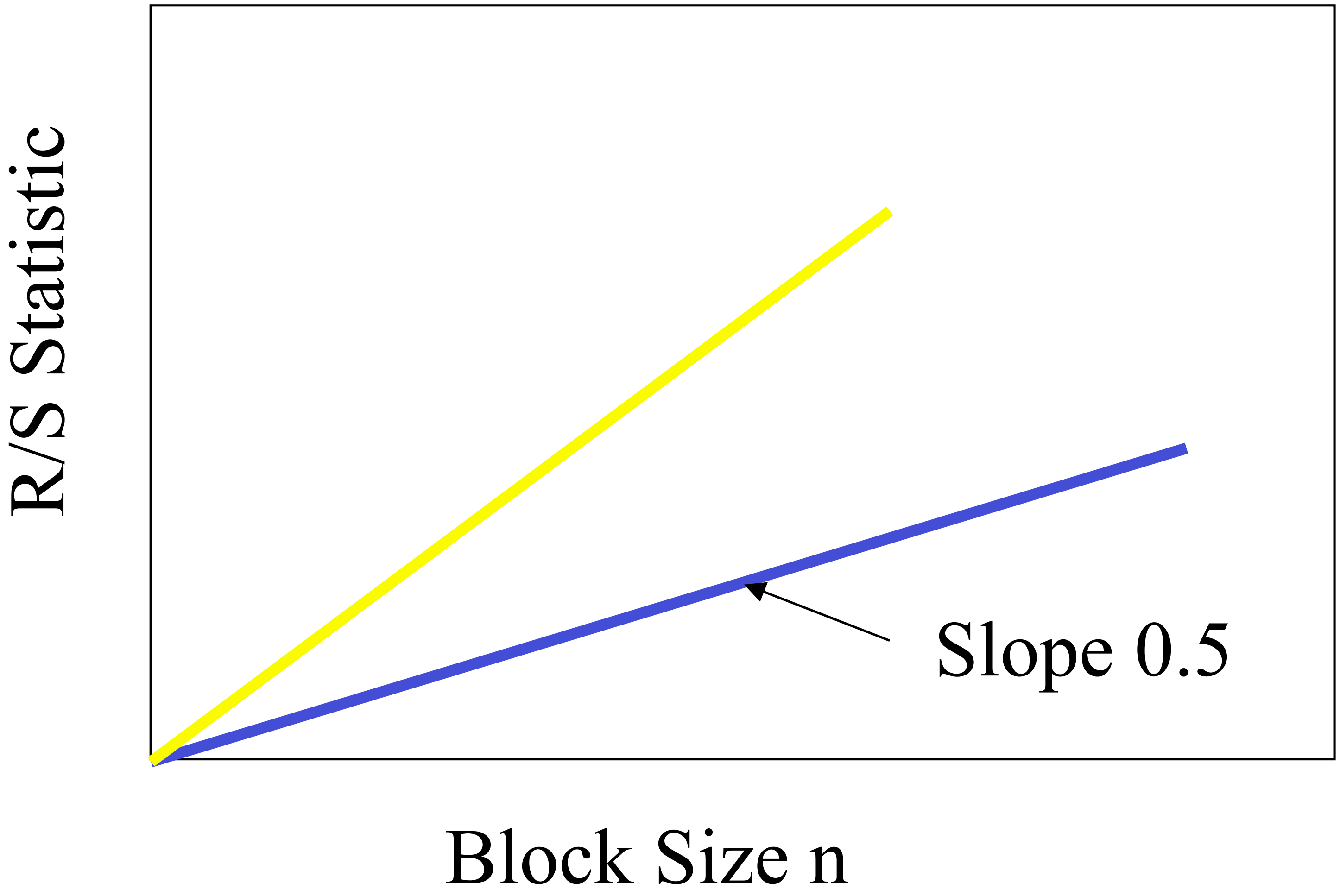
# R/S Pox Diagram



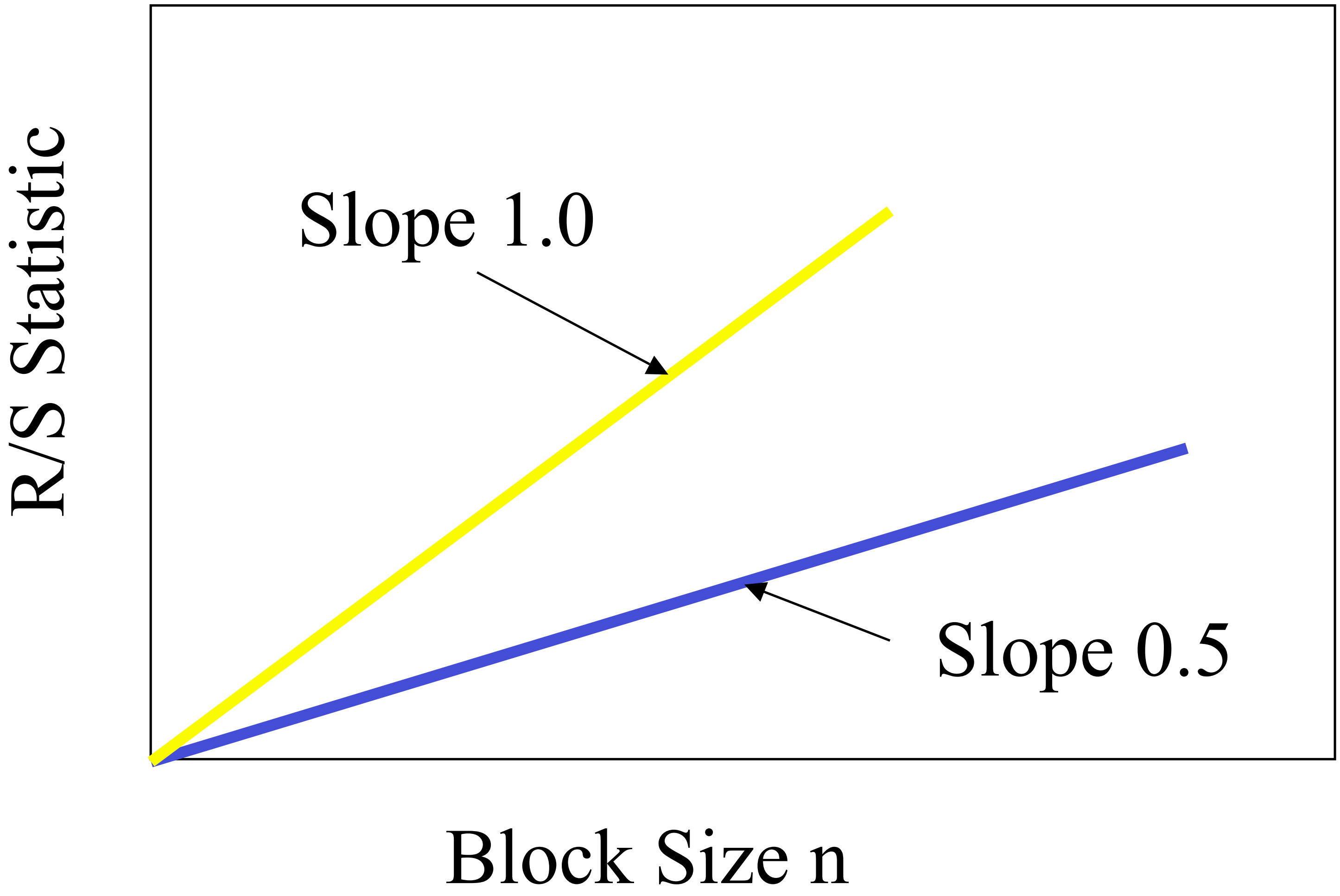
# R/S Pox Diagram



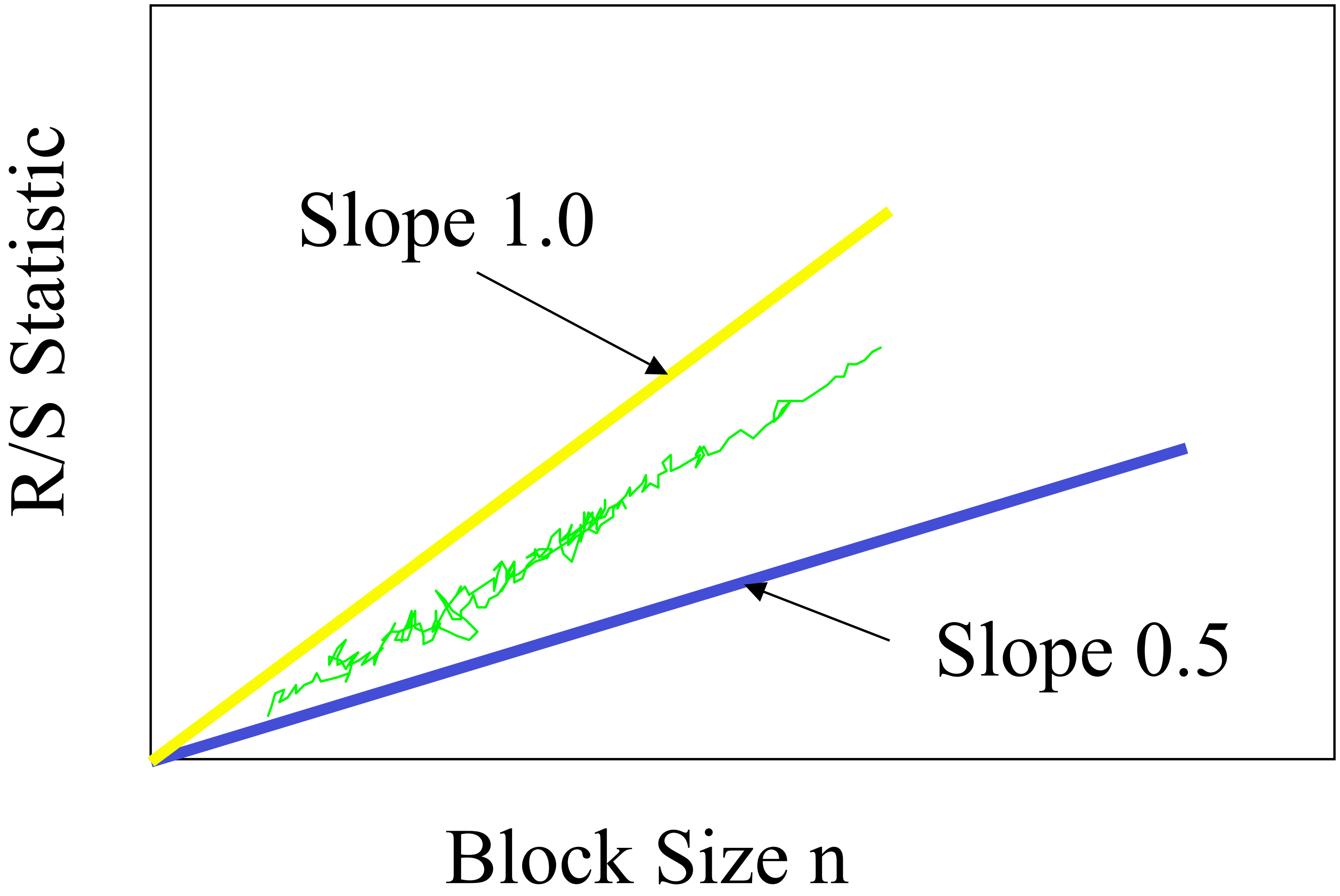
# R/S Pox Diagram



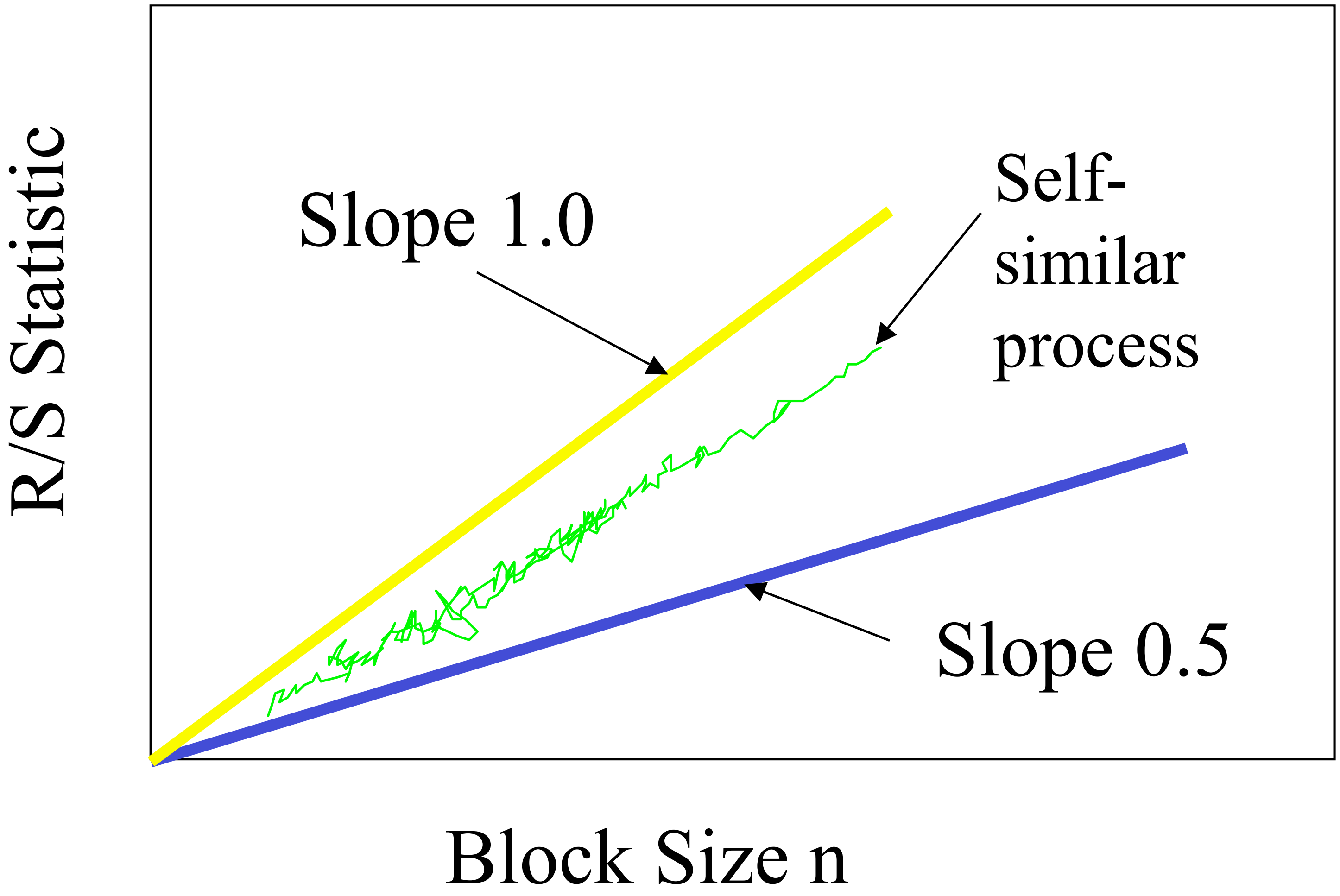
# R/S Pox Diagram



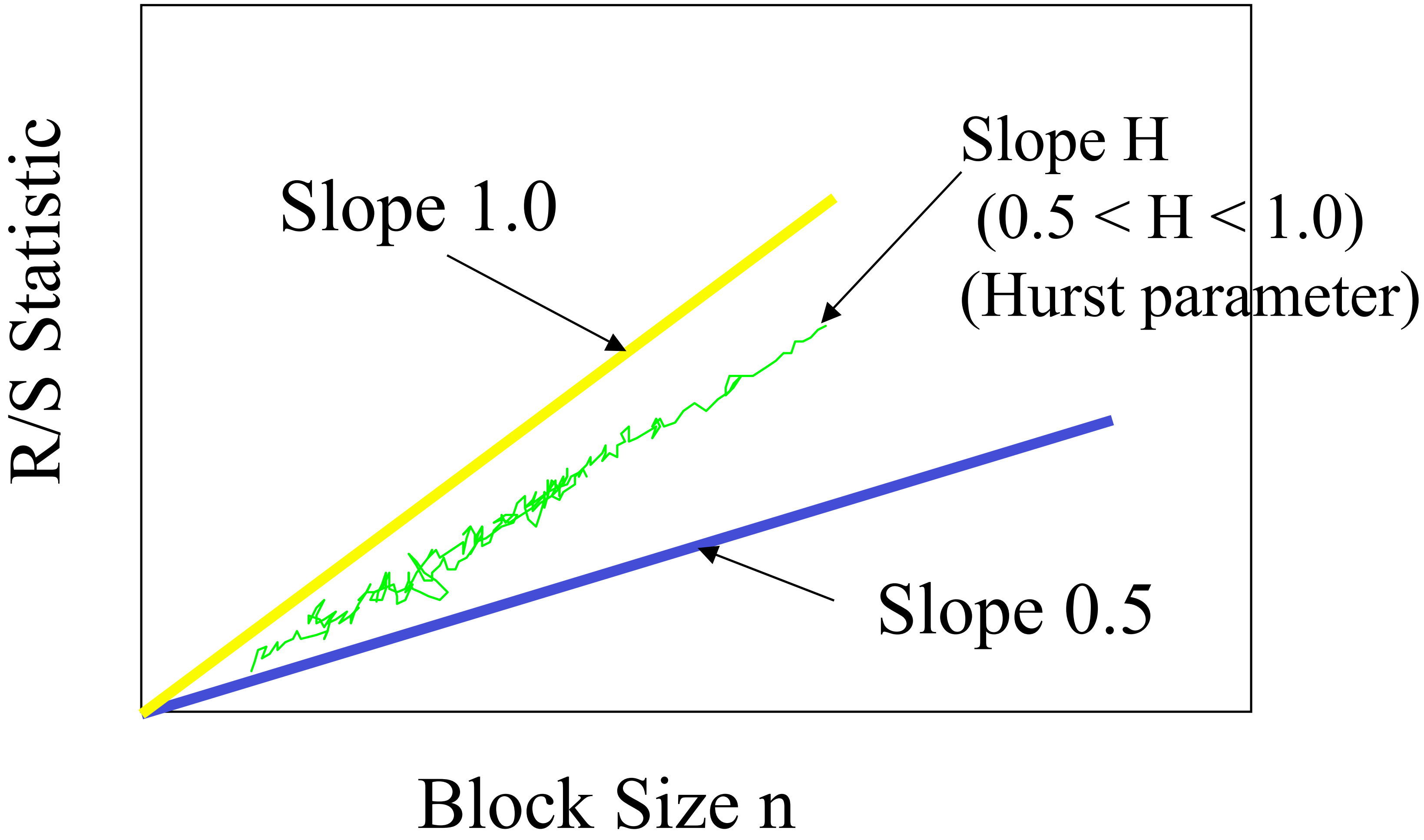
# R/S Pox Diagram



# R/S Pox Diagram



# R/S Pox Diagram



# Self-Similarity Summary

- Self-similarity is an important mathematical property that has recently been identified as present in network traffic measurements
- Important property: burstiness across many time scales, traffic does not aggregate well
- There exist several mathematical methods to test for the presence of self-similarity, and to estimate the Hurst parameter H
- There exist models for self-similar traffic

# Newer Results

V. Paxson, S. Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling, IEEE/ACM* Transaction on Networking, 1995.

- TCP *session* arrivals are well modeled by a Poisson process
- A number of WAN characteristics were well modeled by *heavy tailed* distributions
- *Packet* arrival process for two typical applications (TELNET, FTP) as well as aggregate traffic is *self-similar*

# Another Study

M. Crovella, A. Bestavros, *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes, IEEE/ACM* Transactions on Networking, 1997

- Analyzed WWW logs collected at clients over a 1.5 month period
  - First WWW client study
  - Instrumented MOSAIC
    - ~600 students
    - ~130K files transferred
    - ~2.7GB data transferred

# Self-Similar Aspects of Web traffic

- One difficulty in the analysis was finding stationary, busy periods
  - A number of candidate hours were found
- All four tests for self-similarity were employed
  - $0.7 < H < 0.8$

# Explaining Self-Similarity

- Consider a set of processes which are either ON or OFF
  - The distribution of ON and OFF times are heavy tailed
  - The aggregation of these processes leads to a self-similar process
- So, how do we get heavy tailed ON or OFF times?

# Impact of File Sizes

- Analysis of client logs showed that ON times were, in fact, heavy tailed
  - Over about 3 orders of magnitude
- This lead to the analysis of underlying file sizes
  - Over about 4 orders of magnitude
  - Similar to FTP traffic
- Files available from UNIX file systems are typically heavy tailed

# Heavy Tailed OFF times

- Analysis of OFF times showed that they are also heavy tailed
- Distinction between Active and Passive OFF times
  - Inter vs. Intra click OFF times
- Thus, ON times are more likely to be cause of self-similarity

# Major Results from CB97

- Established that WWW traffic was self-similar
- Modeled a number of different WWW characteristics (focus on the tail)
- Provide an explanation for self-similarity of WWW traffic based on underlying file size distribution

# Where are we now?

- There is no mechanistic model for Internet traffic
  - Topology?
  - Routing?
- People want to blame the protocols for observed behavior
- Multiresolution analysis may provide a means for better models
- Many people (vendors) chose to ignore self-similarity
  - Does it matter????
  - Critical opportunity for answering this question.