

# Predicting Software Defects Through Time Series

---

B. Kenmei, G. Antoniol



# Contents

---

- The aim
- Background on time series
- How we proceed
- Case study and results



# The aim

---

- Identifying relevant characteristics (periods, trends, etc)
- Analyze and understand trends
- Predicting the defects (or other parameters) can be a significant help to manage software evolution



# Time Series

---

- A collection of observations made sequentially in time
  - number of defects, size, people, average age of modification, complexity, ...
- Each observation is a realization of a stochastic process
- Assumptions of stationarity and ergodicity



# Objectives

---

- Understand the underlying stochastic process

(Trend, seasonality, periods, etc.)

- Predict futures values

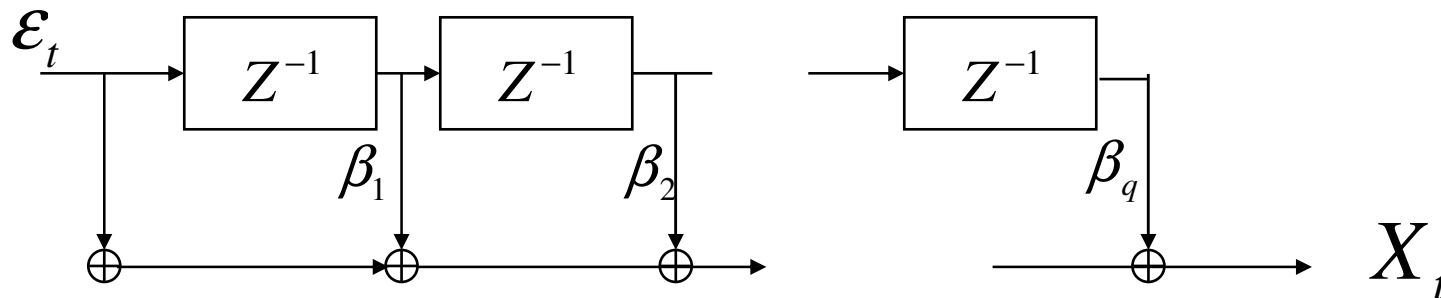
- $X_t = T_t + C_t + \varepsilon_t$

- Identify a model

# Moving Average

Suppose that  $\{\varepsilon_t\}$  is a white noise (discrete purely random process with zero mean and variance  $\sigma^2$ ) then a process  $\{X_t\}$  is said to be a moving average process of order  $q$ , MA( $q$ ), if:

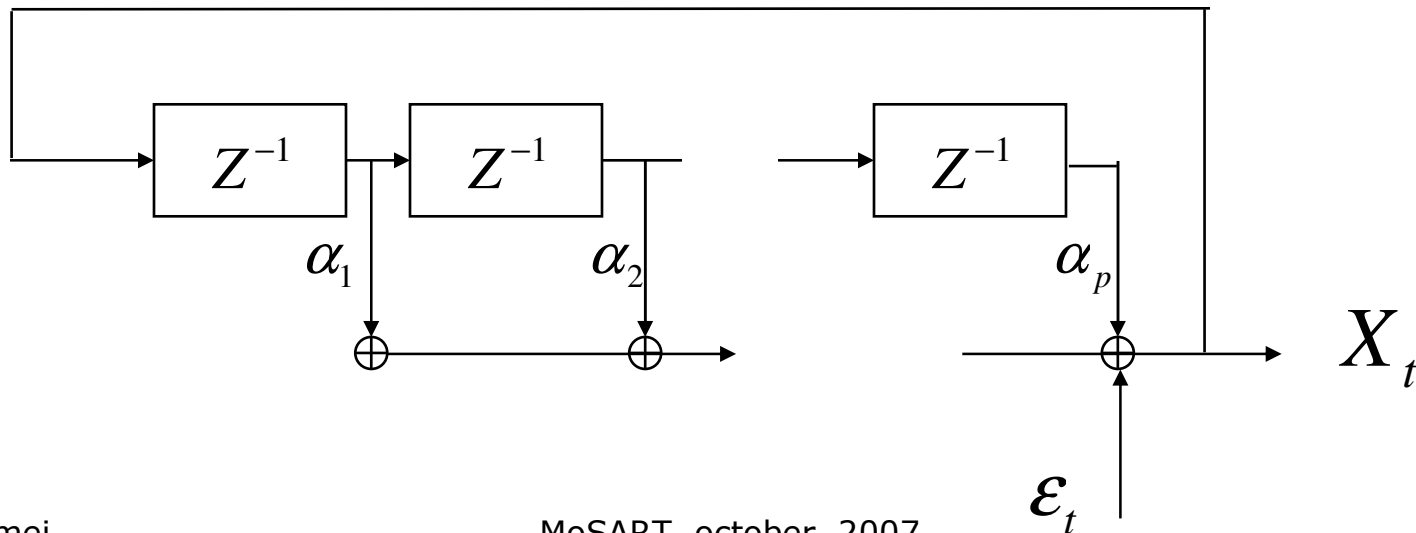
$$X_t = \varepsilon_t + \beta_1 \varepsilon_{t-1} + \dots + \beta_q \varepsilon_{t-q}$$



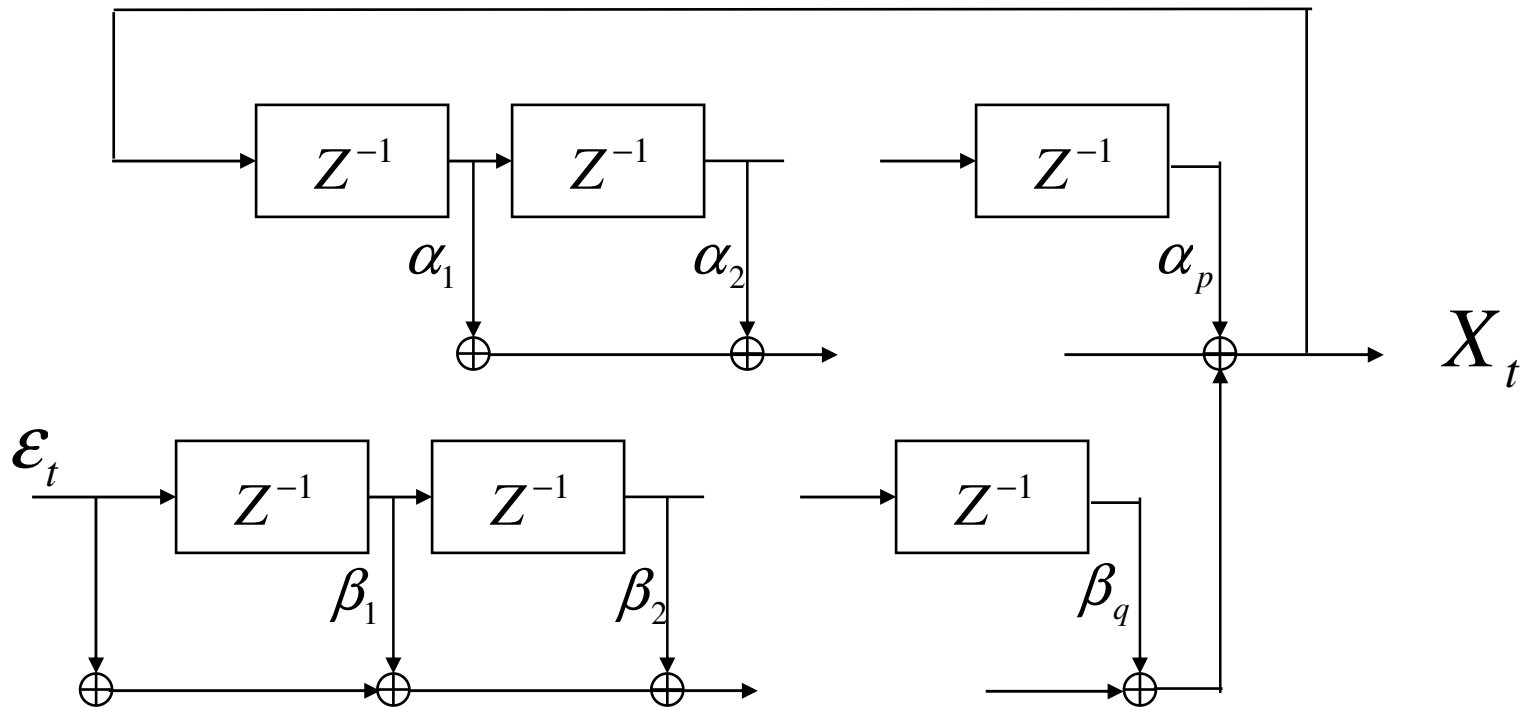
# Auto-regressive

Suppose that  $\{\varepsilon_t\}$  is white noise, then a process  $\{X_t\}$  is said to be an autoregressive process of order  $p$ , AR( $p$ ), if:

$$X_t = \alpha_1 X_{t-1} + \dots + \alpha_p X_{t-p} + \varepsilon_t$$



# ARMA(p,q)



$$X_t = \alpha_1 X_{t-1} + \dots + \alpha_p X_{t-p} + \epsilon_t + \beta_1 \epsilon_{t-1} + \dots + \beta_q \epsilon_{t-q}$$



# Time Series Modeling

---

1. Model identification: ARIMA(p,d,q)  
i.e., determine p,d,q
  - d=0 means ARMA
  - d>0 means a non-stationary process
2. Estimation: given the model order p,d,q, estimate the coefficients  $\{\alpha_i\}$   $\{\beta_j\}$
3. Diagnostic checking: verify if the identified model is adequate  
(white noise test of the residuals, normality)



## How we proceed

---

- We use different parameters from software databases
  - CVS and bug tracking
- Not only bugs, but also
  - Patches
  - People
  - System size
  - Average age of the system, etc.



# Data extraction

---

- o Software : Mozilla, Jboss, Eclipse
- o Every 2 weeks
- o From 2002 to 2005
- o Temporal features are being considered



# Case Study

---

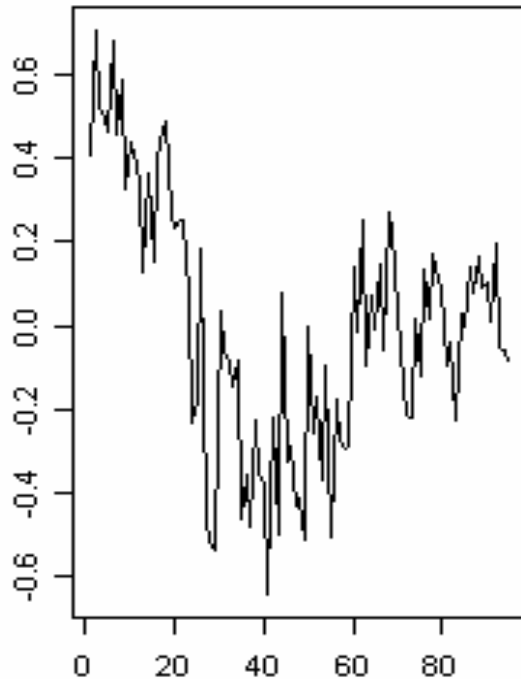
- Prediction error measured as:

$$100 \frac{|x_t - \hat{x}_t|}{x_t}$$

- Statistical comparisons with naive models (ARIMA(0,1,0), ...)

# Mozilla

95 observations open bugs bi-weekly,  
stationary



$$X_t : ARIMA(2,0,3)$$

Let  $m_1$  be the errors of ARIMA(0,1,0) model  
and  $m_2$  the errors from our model

$$H_0 : m_1 = m_2$$

$$H_1 : m_1 > m_2$$



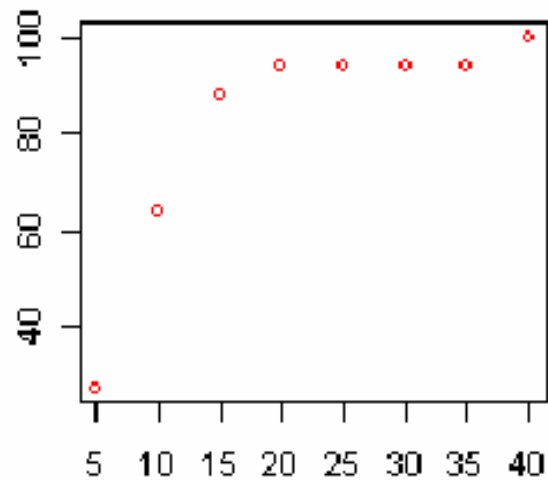
## Mozilla : models comparison

|      | h=1     | h=2     | h=3     |
|------|---------|---------|---------|
| S=20 | T=0.006 | T=0.090 | T=0.000 |
| S=40 | T=0.000 | T=0.034 | T=0.000 |
| S=60 | T=0.019 | T=0.206 | T=0.005 |
| S=70 | T=0.025 | T=0.507 | T=0.038 |

With a threshold of 95%,  $m_1 > m_2$  in all cases  
when  $h=1,3$

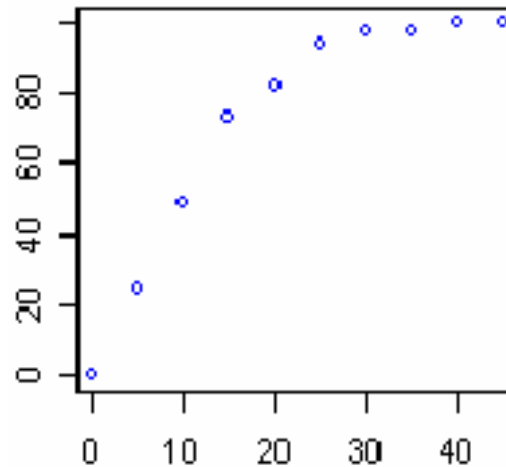
# Mozilla : prediction performance

learning size : 60



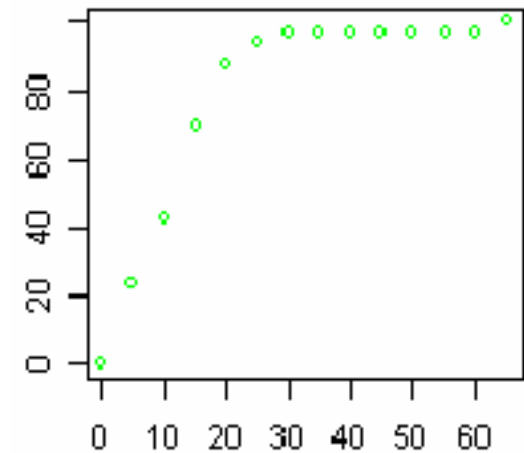
H=1

90% < 15%



H=2

90% < 25%

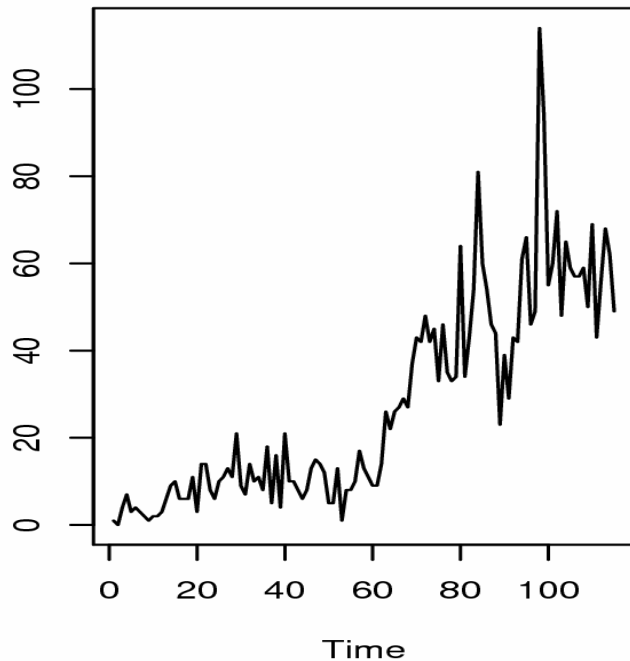


H=3

90% < 25%

# Jboss

115 observations, trend stationary (TS)  
– bugs open bi-weekly



$$X_t = 0.03811 * t + 1.60740 + Z_t$$

$$Z_t : AR(2)$$

Let  $m_1$  be the errors of ARIMA(0,1,0) model  
and  $m_2$  the errors from our model

$$H_0 : m_1 = m_2$$

$$H_1 : m_1 > m_2$$



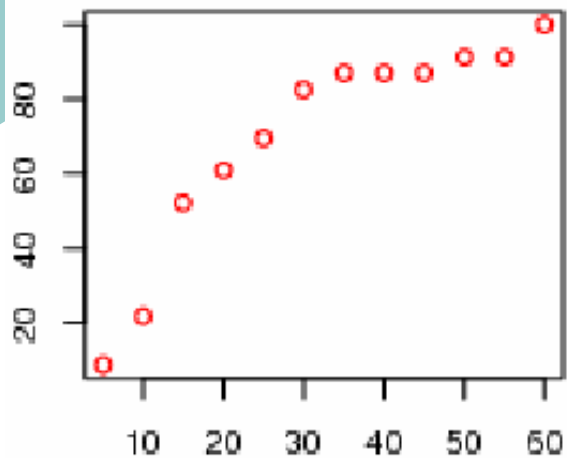
## Jboss : models comparison

|      | h=1     | h=2     | h=3     |
|------|---------|---------|---------|
| S=20 | T=0.047 | T=0.093 | T=0.008 |
| S=70 | T=0.040 | T=0.015 | T=0.053 |
| S=80 | T=0.053 | T=0.002 | T=0.052 |
| S=90 | T=0.078 | T=0.010 | T=0.129 |

With a threshold of 90%,  $m1 > m2$  in 11 cases/12

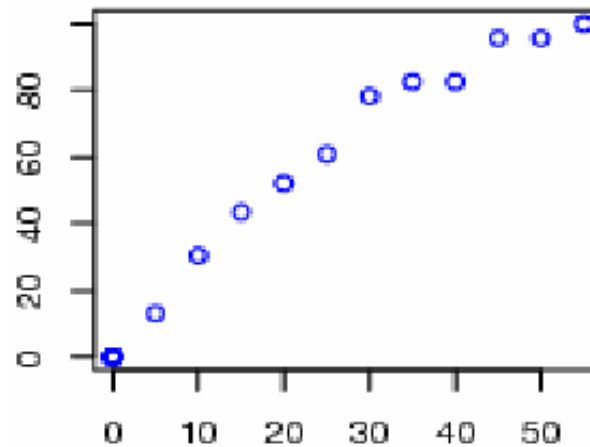
# Jboss : prediction performance

learning size : 90



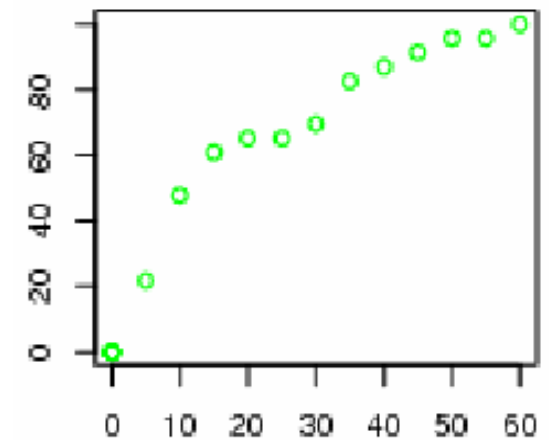
H=1

80% < 30%



H=2

80% < 30%



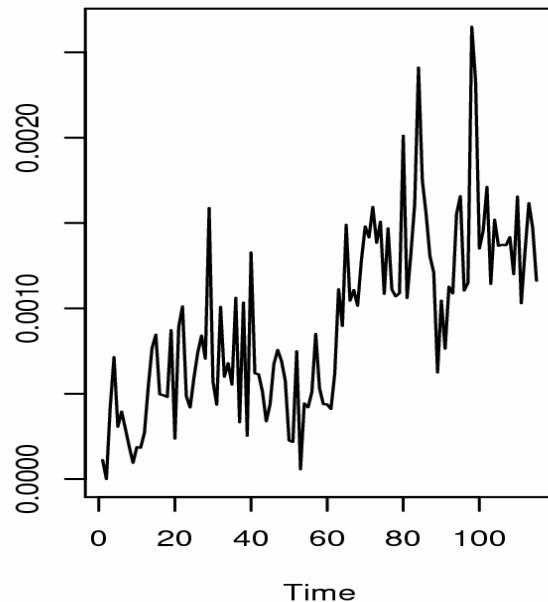
H=3

70% < 30%

# Jboss : normalised series

---

115 observations, trend stationary (TS) bi-weekly bug open normalized over size



$$X_t = 0.001178 * t - 1.896278 + Z_t$$

$$Z_t : ARMA(4,3)$$

70% of prediction errors <25%

With a threshold of 95%,  $m1 > m2$  in 10 cases/12

# Eclipse

119 observations, trend stationary (TS)  
– bug open bi-weekly non normalized

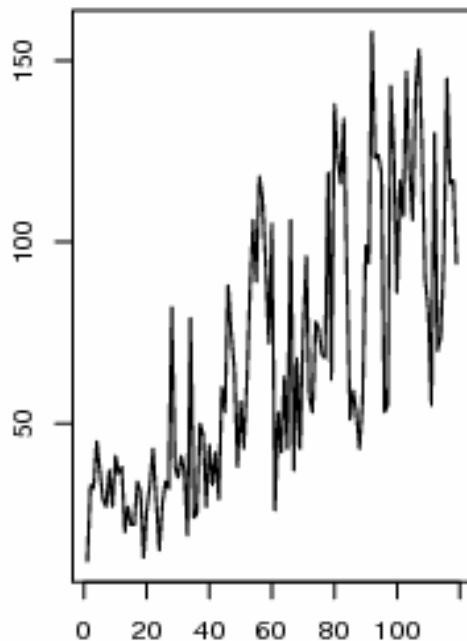
$$X_t = 0.03419 * t + 4.75902 + Z_t$$

$$Z_t : ARMA(5,4)$$

Let  $m_1$  be the errors of ARIMA(0,1,0) model  
and  $m_2$  the errors from our model

$$H_0 : m_1 = m_2$$

$$H_1 : m_1 > m_2$$





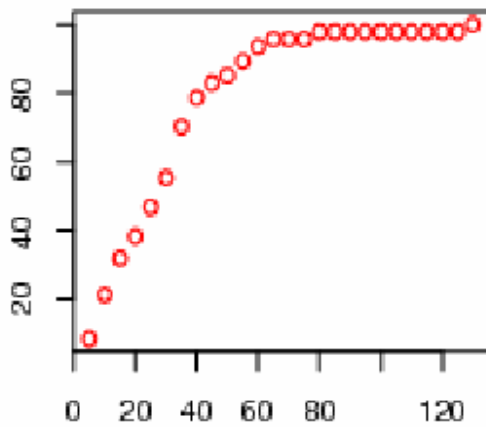
## Eclipse : models comparison

|      | h=1     | h=2     | h=3     |
|------|---------|---------|---------|
| S=60 | T=0.083 | T=0.015 | T=0.000 |
| S=70 | T=0.574 | T=0.021 | T=0.003 |
| S=80 | T=0.540 | T=0.034 | T=0.007 |
| S=90 | T=0.481 | T=0.005 | T=0.013 |

With a threshold of 95%,  $m_1 > m_2$  when  $h=2,3$

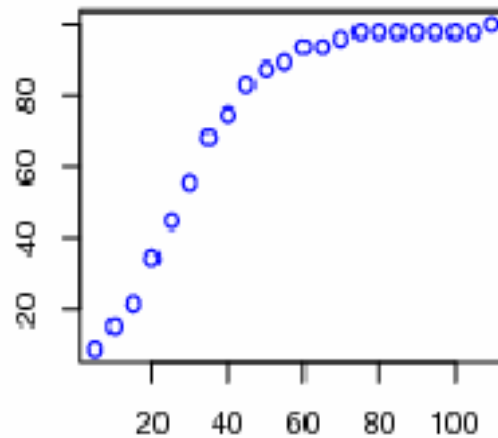
# Eclipse : prediction performance

learning size : 70



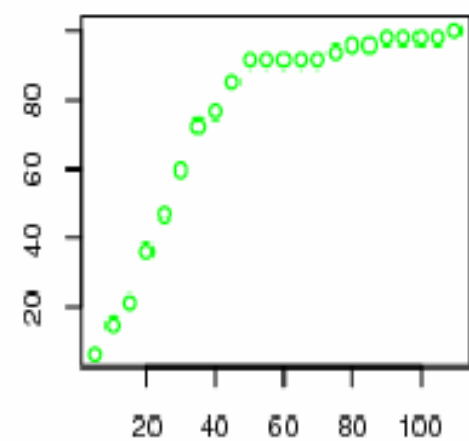
H=1

70% < 35%



H=2

70% < 35%



H=3

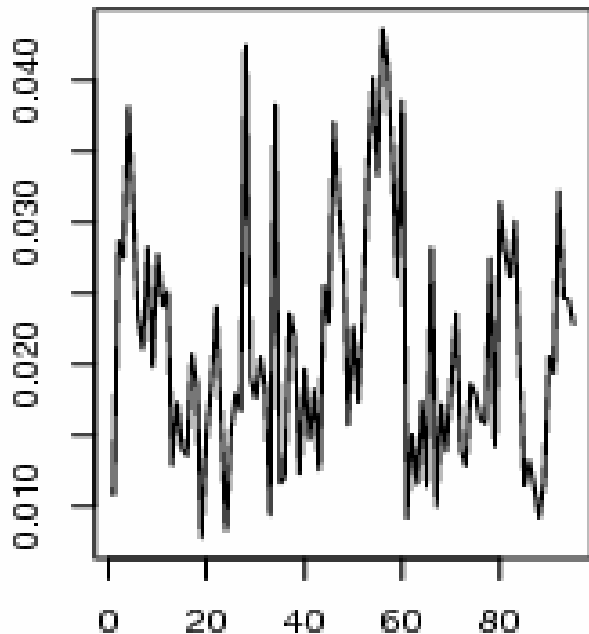
75% < 35%

# Eclipse : normalised series

---

119 observations, stationary, bug open  
malized time series

$$X_t : ARIMA(5,0,5)$$



80% of prediction errors <40%

With a threshold of 95%,  $m_1 > m_2$  when  $h=2,3$



# Conclusion

---

## Difficulties

- The non automatic steps of the analysis
- Dynamic prediction with a single model

## On going work

- Extraction of temporal features at the system level
- Perform a multivariate analysis
- Prospect non linear methods