Investigating Some Estimators of the Fractional Degree of Differencing, In Long Memory Time Series

by

Sebastien D. Kiogou

(Under the direction of William P. McCormick)

Abstract

We investigated three estimators of the fractional parameter d, in long memory time series. Discussed in [4], the first estimator is based on a regression analysis using the periodogram of the long memory series; the second estimator which is discussed in [11, 10], is based on a regression analysis using a lag-window spectral density estimator; the third estimator performs a maximum likelihood estimation (MLE) of d using the fast and accurate method of [8]. To conduct our investigation, we generated synthetic ARFIMA(p, d, q) samples where $d = .25$ and $d = .45$. Computational results showed that in general the MLE method performs better in large samples whereas the [4] proposed estimator performs better in small samples. Yet, the estimator in [11, 10] behaves better than that proposed by [4] in large samples. While the MLE has the smallest standard errors in both small and large samples, the standard errors of the [4] approach are the largest.

Index words: Fractional degree of differencing, Fractional parameter d, Long memory time series models, Spectral density, lag-window, Fast and accurate MLE

Investigating Some Estimators of the Fractional Degree of Differencing, In Long Memory Time Series

by

Sebastien D. Kiogou

(Under the direction of William P. McCormick)

M.Sc., University of Abidjan-Cocody, Ivory Coast, 1996

M.Sc., Kennesaw State University, GA, 2006

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master's of Science

Athens, Georgia

2011

Investigating Some Estimators of the Fractional Degree of Differencing, In Long Memory Time Series

by

Sebastien D. Kiogou

(Under the direction of William P. McCormick)

Approved:

Major Professor:	William P. McCormick
Committee:	Lynne Billard
Committee:	William Lastrapes

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2011

Acknowledgments

I hereby would like to thank my God Jehovah Ezer for walking me through this journey; may his Holy name be blessed for ever.

I also want to seize this opportunity to thank my wife Kiogou Lydie; I think I would have been mad if she was not around. May God bless her. Likewise, I would like to thank my lovely kids Joseph and Miriam. Going back home, I knew that they will always keep me busy, playful, and laughing; they kept my spirit high. I have the conviction that Joseph will be one of the greatest scientist to have walked on planet earth. I also rest assured that Miriam will be a great American judge; her name Ozoua in *Bété* means the one who is always above all. By the way, Joseph's name in *Bété* is Dalli which means the beloved of his mother.

In the same vein, I want to thank Dr. McCormick my good Samaritan and the committee members including Dr. Billard and Dr. Lastrapes for accepting to walk me through this thesis. Their contribution to this thesis is inestimable. I pray that God grant them with very long lives.

To all of them, thank you very much and may God bless you!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A natural extension of the autoregressive and the moving average models is the mixed autoregressive moving average or ARMA(p,q) model. Autoregressive moving average family of models are useful in describing stationary time series. Yet, not all time series processes are stationary; as a result, non-stationary processes need to be differenced in order to achieve "stationarity" . When the degree of differencing used to difference non-stationary realizations is an integer greater or equal to 1, then we have another family of models called autoregressive integrated moving average or ARIMA(p,d,q) models useful in describing various homogeneous non-stationary time series.

Nonetheless, an autoregressive moving average process can be modeled as follows:

$$\phi_p(\beta)Z_t = \theta_q(\beta)a_t \qquad where \tag{1.1}$$

$\theta_q(\beta)$ is an invertible moving average operator

$\phi_p(\beta)$ is a stationary autoregressive operator

$\phi_p(\beta) = 1 - \phi_1\beta - \ldots - \phi_p\beta^p$

$\theta_q(\beta) = 1 - \theta_1\beta - \ldots - \theta_q\beta^q$

a_t is a white noise process with mean 0 and variance 1.

An invertible and stationary ARMA(p,q) technically requires respectively that the roots of $\theta_q(\beta) = 0$ and $\phi_p(\beta) = 0$ lie outside of the unit circle; in addition it is assumed that $\theta_q(\beta) = 0$ and $\phi_p(\beta) = 0$ share no common roots.

On the other hand, an autoregressive integrated moving average process can be presented as follows:

$$\phi_p(\beta)(1-\beta)^d Z_t = \theta_q(\beta)a_t \qquad \textit{where} \tag{1.2}$$

$\theta_q(\beta)$ is an invertible moving average operator

$\phi_p(\beta)$ is a stationary autoregressive operator

β is a backward shift operator such that $\beta X_t = X_{t-1}$ *and*

$d \geq 1$ is an integer.

Thus a homogeneous non-stationary series can be reduced to a stationary series by taking a suitable difference of the series. Clearly, the series $\{Z_t\}$ is non-stationary but its d^{th} differenced series $\left\{(1-\beta)^d Z_t\right\}$ for some integer $d \geq 1$ is stationary.

A more general family of models which use fractional differencing is of interest in particular in the modeling of long memory processes. Yet, [7] and [10] summarized the properties of a long memory process as follows:

- unbounded spectral density at low frequency $\omega = 0$
  i.e. $f(\omega) \to \infty$ as $\omega \to 0$

- hyperbolic decay of the autocorrelation function

  i.e. $|\rho(k)| \approx c(k)^{\alpha}$, as k $\to \infty$ for $\alpha < 0$

- long-term persistence

  i.e. $\sum_{k=-\infty}^{\infty} |\rho(k)| = \infty$, or ACF not absolutely summable.

In the same vein, [4] argued that in standard autoregressive moving average models used for the modeling of stationary time series, the spectral density function is bounded at frequency $\omega = 0$ and the autocorrelation function decays exponentially; as a result these models fail to model adequately long memory processes. They insisted that the failure of standard ARMA models to represent the spectral density at low frequencies suggest that long run forecasts obtained using these models could be less appropriate than those produced by long memory models that permit unbounded spectral densities at frequency $\omega = 0$ and autocorrelation functions that decay hyperbolically. Also, [7] discussed the same class of fractional differenced models and showed that these models have long range forecast abilities. Nonetheless, let us re-consider equation (2). We know that the series $\{Z_t\}$ is non-stationary but its d^{th} differenced series $\left\{(1-\beta)^d Z_t\right\}$ for some integer $d \geq 1$ is stationary; then necessarily for a stationary process we should have $d < 1$. However, it can be shown that when $d < 1/2$ the process $\{Z_t\}$ is causal, i.e. $\{Z_t\}$ can be represented as $\sum_{j=0}^{\infty} \psi_j a_{t-j}$ where $\sum_{j=0}^{\infty} \psi_j^2 < \infty$; see [1] for more details. Hence we can find the exact range of d where $\left\{(1-\beta)^d Z_t\right\}$ is stationary and invertible as follows:

let $\phi_p(\beta)(1-\beta)^d Z_t = \theta_q(\beta) a_t$ *with* $-1/2 < d < 1/2$.

Without lost of generality, let $(1-\beta)^d Z_t = a_t$.

If $\left\{(1-\beta)^d Z_t\right\}$ is stationary, then $Z_t = (1-\beta)^{-d} a_t = \sum_{j=0}^{\infty} \psi_j a_{t-j}$ *where*

$(1-\beta)^{-d} = \sum_{j=0}^{\infty} \psi_j \beta^j$ *and* $\{\psi_j\}$ is square summable.

From Taylor series expansion, we have the general binomial formula:

$(1-\beta)^{-d} = \sum_{j=0}^{\infty} \binom{-d}{j}(-\beta)^j = \sum_{j=0}^{\infty} \psi_j \beta^j$ *where*

$\psi_j = (-1)^j \binom{-d}{j} = (-1)^j \frac{(-d)(-d-1)...(-d-j+1)}{j!} = \frac{(j+d-1)(j+d-2)...(j+d-j)}{j!}$

$= \frac{\Gamma(j+d)}{j!\Gamma(d)}$ *with* $\Gamma(.)$ the gamma function.

Using Stirling's formula, we know that $\Gamma(x) \approx \sqrt{2\pi} e^{-x} x^{x-1/2}$ as x $\to \infty$.

Hence we have:

$\psi_j = \frac{\Gamma(j+d)}{\Gamma(j+1)\Gamma(d)} \approx \frac{1}{\Gamma(d)} \frac{\sqrt{2\pi} e^{-j-d}(j+d)^{j+d-1/2}}{\sqrt{2\pi} e^{-j-1}(j+1)^{j+1-1/2}} \approx \frac{1}{\Gamma(d) j^{1-d}}$

Clearly, $\{\psi_j\}$ is square summable if and only if $2(1-d) > 1$, i.e. $d < .5$. Similarly, the process is invertible if and only if $d > -.5$. See [7] for more details. Hence $(1-\beta)^d Z_t = a_t$ and more generally $\phi_p(\beta)(1-\beta)^d Z_t = \theta_q(\beta) a_t$ is stationary and invertible if and only if $-.5 < d < .5$. The former model is therefore referred to as fractionally integrated noise and the latter is called autoregressive fractionally integrated moving average model.

Yet, the spectrum of the fractionally integrated noise is equal to:

$f(\omega) = \frac{\sigma_a^2}{2\pi} |1 - e^{-i\omega}|^{-2d} = \frac{\sigma_a^2}{2\pi} |\frac{2ie^{-\frac{i\omega}{2}}(e^{\frac{i\omega}{2}} - e^{-\frac{i\omega}{2}})}{2i}|^{-2d}$

$= \frac{\sigma_a^2}{2\pi} [2\sin(\frac{\omega}{2})]^{-2d}$, $\quad -\pi \leq \omega \leq \pi.$

In the same vein, $\rho(k)$, the autocorrelation function at lag k is:
$\rho(k) = \frac{\gamma_k}{\gamma_0}$, where γ_k is the covariance function of $\{Z_t\}$: $\gamma_k = E(Z_t Z_{t-k})$ assuming $EZ_t = 0$.

Note that by [6], $\int_0^{\pi} \cos(k\omega)[\sin(\omega)]^{\alpha-1} d\omega = \frac{\pi \cos(k\pi/2)\Gamma(\alpha+1)2^{1-\alpha}}{\alpha\Gamma[(\alpha+k+1)/2]\Gamma[(\alpha-k+1)/2]}$.

Also by Fourier inversion,

$\gamma_k = \int_{-\pi}^{\pi} e^{ik\omega} f(\omega) d\omega = \frac{\sigma_a^2}{2\pi} \int_0^{\pi} 2\cos(k\omega)[2\sin(\frac{\omega}{2})]^{-2d} d\omega$

$= \frac{(2)^{-2d}\sigma_a^2}{1} \frac{\cos(2k\pi/2)\Gamma(2-2d)(2)^{1+2d-1}}{(1-2d)\Gamma[(1-2d+2k+1)/2]\Gamma[(1-2d-2k+1)/2]} = (-1)^k \sigma_a^2 \frac{\Gamma(1-2d)}{\Gamma(k-d+1)\Gamma(1-d-k)}.$

Hence

$\rho(k) = \frac{\gamma_k}{\gamma_0} = \frac{(-1)^k\Gamma(1-2d)}{\Gamma(k-d+1)\Gamma(1-d-k)} \frac{\Gamma(1-d)\Gamma(1-d)}{\Gamma(1-2d)}$

$= \frac{\Gamma(1-d)}{\Gamma(k-d+1)} \frac{(-1)^k(1-d-1)(1-d-2)...(1-d-k)\Gamma(1-d-k)}{\Gamma(1-d-k)}$

$= \frac{\Gamma(1-d)}{\Gamma(k-d+1)}(-1)^k(-1)^k(d)(d-1)\dots(d+k-1)$

$= \frac{\Gamma(1-d)\Gamma(k+d)}{\Gamma(k-d+1)\Gamma(d)}$

$\rho(k) \approx \frac{(1-d)}{\Gamma(d)} \frac{\sqrt{2\pi}e^{-k-d}(k+d)^{k+d-1/2}}{\sqrt{2\pi}e^{-k+d-1}(k-d+1)^{k-d+1-1/2}}$

$\approx \frac{(1-d)}{\Gamma(d)} e^{-2d+1}(k)^{2d-1}$

$\approx ck^{2d-1}$, where c is a constant.

Furthermore, a stationary process is said to be a long memory process if its ACF follows an asymptotic hyperbolic decay, i.e. $|\rho(k)| \approx c(k)^\alpha$, as k $\to \infty$ for $\alpha < 0$. A long memory process with an ACF which is not absolutely summable, i.e. $\sum_{k=-\infty}^{\infty} |\rho(k)| = \infty$, is further said to be persistent. Since $d < .5$, then the autocorrelation function, $\rho(k) \approx c(k)^{2d-1}$, follows an asymptotic hyperbolic decay. As a result, the fractionally differenced noise process and furthermore the ARFIMA(p,d,q) process, where $-.5 < d < .5$, is a long memory process. In addition, for the ACF $\rho(k)$, $\sum_{k=-\infty}^{\infty} |\rho(k)|$ converges only if $(2d-1) < -1$ or $d < 0$. As such, the fractionally differenced noise process and additionally the ARFIMA(p,d,q) process is in fact persistent when $0 < d < .5$. In the same vein, [7] stated that for suitable values of d, $0 < d < .5$, fractionally differenced models are capable of modeling long memory time series.

According to [7, 10] , the family of fractionally integrated autoregressive moving average models, ARFIMA(p,d,q), has a wide variety of applications in many areas of science including

hydrology (stream flow, rain fall, ...), weather (temperature) and economics (financial time series). In [3] for instance, fractionally integrated models were used to examine if there are significant trends in the global and hemispheric temperature anomaly series. Likewise, [2] used fractionally integrated ARMA processes in order to forecast times series that exhibit long-memory features.

Even though we are using fractionally integrated noise processes ($a_t \sim N(0,1)$), and values of .25 and .45 for d, the graphical illustration below could characterize long memory processes:

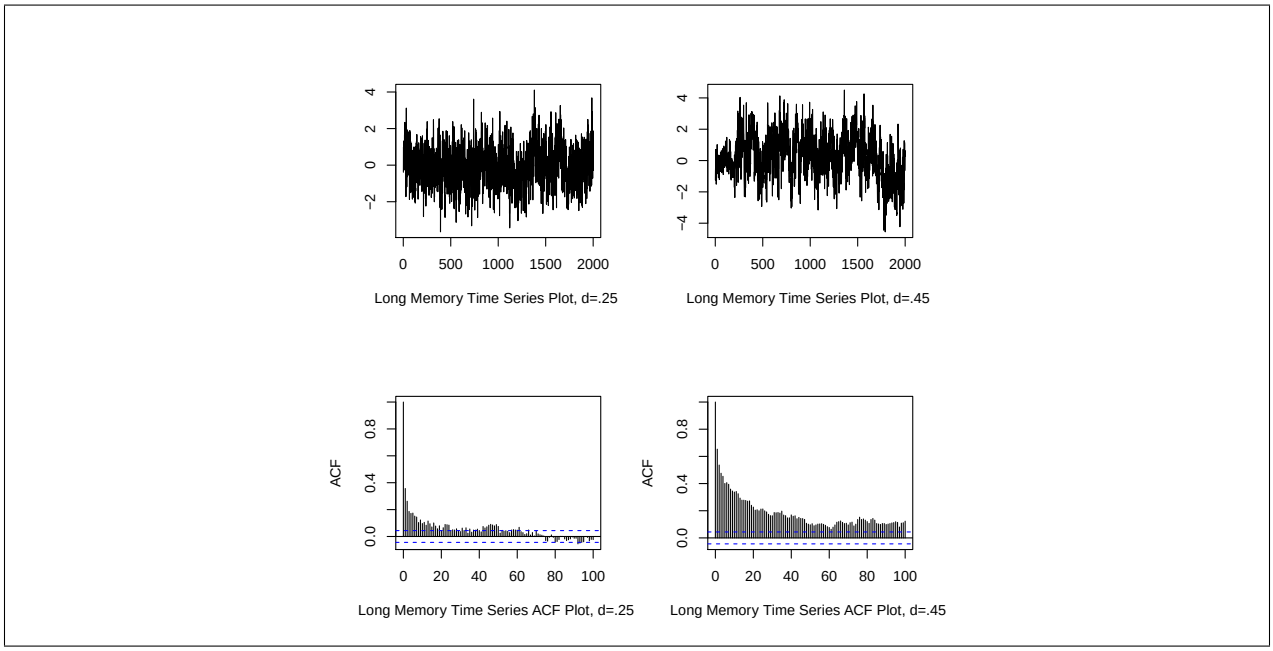

Figure 1.1: Long Memory Time Series Plots Along With ACF Plots where $a_t \sim N(0,1)$

Nonetheless, this thesis will mainly investigate three estimators of the fractional degree of differencing, d. Discussed in [4] , the first estimator is based on a regression analysis using the periodogram of the long memory series; the second estimator, discussed in [11, 10] is based on a regression analysis and uses a lag-window spectral density estimator; finally we will use a maximum likelihood estimator of d where the likelihood is approximated using the fast and accurate method of [8]. Hence, in the first section we will present the three

estimators; then we shall discuss our method of data gathering in the second section; in the end, we will present and analyze the simulation results.

Chapter 2

Some Estimators of the fractional differencing parameter, d

2.1 Estimating d by a Regression Analysis Using the Periodogram

Here is how [4] approached the problem of estimating the parameter *d*. For the ARFIMA model given in (2), let $W_t = (1-\beta)^d Z_t$, and let $f_w(\omega)$ and $f_z(\omega)$ be the spectral density functions of $\{W_t\}$ and $\{Z_t\}$, respectively. Then,

$$f_z(\omega) = |1 - e^{-i\omega}|^{-2d} f_w(\omega)\,,\ 0 \le \omega \le \pi, \tag{2.1}$$

where

$$f_w(\omega) = \frac{\sigma_\omega^2}{2\pi} |\frac{\theta_q(e^{-i\omega})}{\phi_p(e^{-i\omega})}|^2$$

is the spectral density of W_t, a regular ARMA(p,q) model . Most importantly, we should also note that $f_z(\omega) \to \infty$ as $\omega \to 0$.

Taking logarithms on both sides of (3), we get:

$$\ln f_z(\omega) = -d \ln |1 - e^{-i\omega}|^2 + \ln f_w(\omega)$$

$$= \ln f_w(0) - d \ln |1 - e^{-i\omega}|^2 + \ln(\frac{f_w(\omega)}{f_w(0)}). \tag{2.2}$$

Replacing ω by the Fourier frequency $\omega_j = 2\pi j/n\,, j = 1, \ldots, [n/2]$ and adding $\ln I_z(\omega_j)$, the periodogram of $\{Z_t\}$, to both sides of (4), gives:

$$\ln I_z(\omega_j) = \ln f_w(0) - d \ln |1 - e^{-i\omega_j}|^2 + \ln(\frac{f_w(\omega_j)}{f_w(0)}) + \ln(\frac{I_z(\omega_j)}{f_z(\omega_j)}). \tag{2.3}$$

where $I_z(\omega_j) = \frac{1}{2\pi}\frac{1}{n}|\sum_{t=1}^{n} Z_t e^{-itw_j}|^2$ is the periodogram of $\{Z_t\}$.

For ω_j near zero, i.e., for $j = 1, \ldots, m \ll (n/2)$ such that $m/n \to 0\, as\, n \to \infty$, we have $\ln(f_w(\omega_j)/f_w(0)) \approx 0$. Thus,

$$Y_j = c - dX_j + e_j\,, \qquad\qquad j = 1, \ldots, m, \tag{2.4}$$

where $Y_j = \ln I_z(\omega_j)$, $c = \ln f_w(0)$, $X_j = \ln |1 - e^{-i\omega_j}|^2$, $e_j = \ln(I_z(\omega_j)/f_z(\omega_j))$ and the least square estimator of d is then given by

$$\hat{d} = -\frac{\sum_{j=1}^{m}(X_j - \bar{X})(Y_j - \bar{Y})}{\sum_{j=1}^{m}(X_j - \bar{X})^2}.$$

Furthermore, [4] showed that

$$\hat{d} \to N(d, \frac{\pi^2}{6\sum_{j=1}^{m}(X_j - \bar{X})^2})$$

95% C.I. of d is :

$$\hat{d} \pm 1.96\sqrt{\frac{\pi^2}{6\sum_{j=1}^{m}(X_j - \bar{X})^2}}$$

2.2 Estimating d by a Regression Analysis Using the Lag-Window

In [10] it is stated that even though $\hat{d}$ proposed in the precedent section by [4] is asymptotically unbiased , it is not a consistent estimator of d. They also insisted that one cannot estimate d separately from ϕ and θ as stipulated by [4]. As such, [10] then [11] considered the use of a consistent estimator of the spectral density function of $\{Z_t\}$, the Parzen lag-window in place of the periodogram $I_z(\omega_j)$, in order to improve the quality of $\hat{d}$.

They proceeded as follows:

let $f_s(\omega)$ denote the lag-window spectral density estimator of the long memory time series; then

$$f_s(\omega_j) = \frac{1}{2\pi}\sum_{s=-m}^{m} K(\frac{s}{m})R(s)\cos(s\omega_j)$$

where

$$K(u) = \begin{cases} 1 - 6u^2 + 6|u|^3 & |u| \leq 1/2 \\ 2(1-|u|)^3 & 1/2 < u \leq 1 \\ 0 & |u| > 1 \end{cases}$$

is the Parzen lag-window and $R(s)$ is the estimated auto-covariance at lag s;

also $-1 < u < 1$, $K(0) = 1$, and $K(-u) = K(u)$. Clearly, $K(u)$ is a fixed continuous even function; parameter m is chosen such that, as $n \to \infty$, $m \to \infty$, $(m/n) \to 0$.

Adding $\ln f_s(\omega_j)$ to both sides of (4), we obtain:

$$\ln f_s(\omega_j) = \ln f_w(0) - d \ln |1 - e^{-i\omega_j}|^2 + \ln(\frac{f_s(\omega_j)}{f_z(\omega_j)}) + \ln(\frac{f_w(\omega_j)}{f_w(0)}) \tag{2.5}$$

If the upper limit of j, say m, is chosen so that $m/n \to 0$ as $n \to \infty$ and if ω_j is near zero, then $\ln f_w(\omega_j)/f_w(0)$ is negligible and we can re-write (7) as:

$$\ln f_s(\omega_j) \approx \ln f_w(0) - d \ln |1 - e^{-i\omega_j}|^2 + \ln(\frac{f_s(\omega_j)}{f_z(\omega_j)}). \tag{2.6}$$

If $Y_j = \ln f_s(\omega_j)$, $X_j = \ln |1 - e^{-i\omega_j}|^2$,

$\alpha = \ln f_w(0)$, $b = -d$, $e_j = \ln(\frac{f_s(\omega_j)}{f_z(\omega_j)})$,

then by taking $m = [n^\theta]$ we have

$$Y_j = \alpha + bX_j + e_j,\ j = 1, 2, \ldots, [n^\theta],\ 0 < \theta < 1, \tag{2.7}$$

where $[n^\theta]$ denotes the integer part of n^θ.

Hence

$$\hat{b} = \frac{\sum_{j=1}^m (X_j - \bar{X})Y_j}{\sum_{j=1}^m (X_j - \bar{X})^2} \qquad or \qquad \hat{d} = -\frac{\sum_{j=1}^m (X_j - \bar{X})Y_j}{\sum_{j=1}^m (X_j - \bar{X})^2}$$

In addition to the insights above, [10] showed that:

$$\hat{d} \sim AN(d, \frac{km}{n \sum_{j=1}^m (X_j - \bar{X})^2})$$

where k is a constant depending on the lag-window. In this instance we are using the Parzen window; so we used $k = .5393$ and $m = [n^{\theta}]$. Also [10] argued that since the choice of m satisfies $m/n \to 0$ as $n \to \infty$, it follows that $\hat{d}$ is consistent for d. In the same vein, their efficiency claim over the periodogram approach lies in the fact that $km/n < \pi^2/6$ for any $m = [n^{\theta}], \quad 0 < \theta < 1$.

Remember that using the periodogram approach,

$$\hat{d} \to N(d, \frac{\pi^2}{6\sum_{j=1}^{m}(X_j - \bar{X})^2}).$$

2.3 Estimating d by a Maximum Likelihood Estimator

In their conclusion, [10] indicated that it was not reasonable to estimate d separately from ϕ and θ because the behavior of the spectral density near frequency zero could be affected by d, ϕ, and θ . They even insisted that the maximum likelihood approach was more promising to the above mentioned issue. On the other hand, we noticed that most courses in a statistics department discuss the concept of maximum likelihood estimate. Hence, we thought [10] gave us a great opportunity to discuss the same concept in our master's thesis. However, in this thesis we are referring to the likelihood estimator of d where the likelihood is approximated using the fast and accurate method of [8].

Arguing as in [8], let

$X^t = \{X_1, \ldots, X_t\}$,

$X_i^t = \{X_{i1}, \ldots, X_{it}\}$,

$X_{it} = \mu_i + \bigtriangledown^{-d}\phi(\beta)^{-1}\theta(\beta)\varepsilon_{it}$,

$cov(X_{it}, X_{jt}) = \sigma_x^2 r_{ij}$ where $r_{ij} = 1$ if $i = j$ and $r_{ij} = \alpha$ exp(-βd_{ij}) if $i \neq j$ $0 \leq \alpha \leq 1$,

d_{ij} is the distance between location i and j.

An exact maximum likelihood estimation of α, β, d, $\phi(\beta)$, $\theta(\beta)$ would be computationally time consuming (at least 45 hours of CPU time), cumbersome, and prone to be effective only for short, one-dimensional series of length less than 220. However, a fast and accurate approximation can be found as follows:

1. Approximate the conditional means and variances

$$E[X_{it}|X_i^{t-1}] = u_{it} + \omega_t \mu_i$$

and

$$var[X_{it}|X_i^{t-1}] = \sigma_x^2 k \Pi_{j=1}^{t-1}(1-\phi_j^2) = \nu_t$$

where

$$u_{it} = \phi(\beta)\theta(\beta)^{-1} \sum_{j=1}^{t-1} \phi_{tj} X_{i,t-j}$$

$$\omega_t = 1 - \phi(1)\theta(1)^{-1} \sum_{j=1}^{t-1} \phi_{tj}$$

$$\sum_{j=1}^{t-1} \phi_{tj} X_{i,t-j} \approx \sum_{j=1}^{M} \phi_{tj} X_{i,t-j} - \sum_{j=M+1}^{t-1} \pi_j X_{t-j}$$

ϕ_{tj}'s are the partial linear regression coefficients for the ARIMA $(0, d, 0)$ and k is the ratio of the innovations variance to the process variance for the ARMA (p, q) process with parameters $\phi(\beta)$ and $\theta(\beta)$.

2. Given β, d, $\phi(\beta)$, $\theta(\beta)$ approximate MLE of μ and σ_ε^2 by

$$\hat{\mu}_i = \sum_{t=1}^{N} \omega_t (X_{it} - u_{it}) \nu_t^{-1/2} / \sum_{t=1}^{N} \omega_t^2$$

$$\hat{\sigma}_\varepsilon^2 = (Nm)^{-1} \sum_{t=1}^{N} (X_t - u_t - \omega_t \hat{\mu})^T A (X_t - u_t - \omega_t \hat{\mu}) \nu_t^{-1}$$

where $u_t = (u_{1t}, \ldots, u_{mt})^T$ and $\hat{\mu} = (\hat{\mu}_1, \ldots, \hat{\mu}_m)^T$.

3. Construct and maximize the log likelihood function to obtain $\hat{\beta}$, $\hat{d}$, $\phi(\hat{\beta})$, and $\theta(\hat{\beta})$

$$l(\alpha,\, \beta,\, d,\, \phi(\beta),\, \theta(\beta) = constant\, -\, \frac{1}{2}Nm\, \log \hat{\sigma}_{\varepsilon}^{2}\, -\, \frac{1}{2}N|\mathbf{R}|,$$

where $\mathbf{R}$ is a m x m matrix of r_{ij}'s.

Chapter 3

Simulation and Data Gathering

To conduct our investigation on the above mentioned fractional parameter estimators, we generated synthetic ARFIMA$(0, d, 0)$ and ARFIMA$(0, d, 1)$ samples where d took values of .25 and .45. Here is how these data were generated.

3.1 Simulation of ARFIMA(0,d,0)

Consider the model $\bigtriangledown^d X_t = Z_t$. Its autocorrelation function is given by

$$\rho(h) = Corr(X_{t+h}, X_t) = \frac{\Gamma(1-d)\Gamma(h+d)}{\Gamma(d)\Gamma(h+1-d)}$$

$$= \frac{\Gamma(1-d)\Pi_{i=0}^{h-1}(i+d)\Gamma(d)}{\Gamma(d)\Pi_{i=0}^{h-1}(i+1-d)\Gamma(1-d)}$$

$$= \Pi_{i=0}^{h-1}\frac{i+d}{i+1-d} = \Pi_{i=1}^{h}\frac{i-1+d}{i-d}$$

Now let $\{X_0, \ldots, X_{M-1}\}$ be a series of length M from ARFIMA $(0, d, 0)$ based on $N(0, 1)$ innovations such that $\{X_0, \ldots, X_{M-1}\}$ has a multivariate normal distribution with mean

$\underline{0} = (0, \ldots, 0)$ and variance-covariance matrix

$$\sum = \sigma^2 \begin{pmatrix} 1 & \gamma(1) & \gamma(2) & \ldots & \gamma(M-1) \\ \gamma(1) & 1 & \gamma(1) & \ldots & \gamma(M-2) \\ \vdots & & & & \vdots \\ \gamma(M-1) & \gamma(M-2) & \gamma(M-3) & \ldots & 1 \end{pmatrix}$$

where $\gamma(i)$ is the autocorrelation at lag i.

In order to generate the sequence $\{X_0, \ldots, X_{M-1}\}$ we obtained a Cholesky decomposition of $\sum = L'L$ where L is a lower triangular matrix; observe that if the sequence $\underline{e} = \{e_0, \ldots, e_{M-1}\}$ is a sequence of $iid \sim N(0,1)$ variables, then $L'\underline{e}'$ has a multivariate normal distribution with mean $\underline{0}$ and variance-covariance matrix $\sum$ since $E(L'\underline{e}')(\underline{e}L) = L'IL = L'L = \sum$. Hence, the matrix product $L'\underline{e}'$ generates the $\{X_0, \ldots, X_{M-1}\}$ sequence. Please see code portion in the appendix for details about generating sequence $\{X_0, \ldots, X_{M-1}\}$. Note also that this approach was discussed in [4, 5]. If we have the sequence $\{X_0, \ldots, X_{M-1}\}$ generated according to this joint distribution, then observe that we can generate the next variables in the sequence as follows:

$$Z_M = \sum_{j=0}^{\infty} a_j X_{M-j} \approx \sum_{j=0}^{M-1} a_j X_{M-j}$$

$$= X_M + \sum_{j=1}^{M-1} a_j X_{M-j}$$

and

$$X_M = Z_M - \sum_{j=1}^{M-1} a_j X_{M-j}$$

$$X_{M+1} = Z_{M+1} - \sum_{j=1}^{M} a_j X_{M+1-j}$$

$$\vdots$$

where

$$a_j = \frac{\Gamma(j-d)}{\Gamma(-d)\Gamma(j+1)} = \frac{\Pi_{i=0}^{j-1}(i-d)\Gamma(-d)}{\Gamma(-d)\Gamma(j+1)}$$

$$= \frac{1}{j!}\Pi_{i=1}^{j}(i-1-d), \qquad j \geq 1\ ,\ a_0 = 1.$$

3.2 Simulation of ARFIMA(0,d,1)

The method used to generate the ARFIMA$(0, d, 1)$ processes is also based on the fast and accurate method of [8]. This data simulation process was actually set forth to test the accuracy of the fast and accurate method in estimating α, β, d, $\phi(\beta)$, $\theta(\beta)$. As such, the mathematical approach stays the same. Furthermore, this method is perfectly implemented by a package, *fracdiff*, in the statistical software R library. Precisely, a function called *fracdiff.sim* generates simulated long-memory time series data from the fractional ARIMA(p, d, q). Details about this function can be found in the appendix.

Chapter 4

Simulation Results

In order to abide with the small sample / large sample methodology, we actually simulated samples of sizes 50 and 2000 to be used throughout our analysis. We generated sample data for $ARFIMA(0, d, 0)$ and $ARFIMA(0, d, 1)$ keeping values of only .25 and .45 for the fractional parameter d.

At first, we looked at statistics in ARFIMA$(0, d, 0)$ tables containing both large and small samples. In large samples, as in the table below, the maximum likelihood estimator has estimated values closer to the real values of d; its sample median values of .25 and .44 as well as its sample means values of .25 and .44 are the closest possible to real values of d, i.e. .25 and .45 ; likewise, its standard error of .02 is the smallest. In addition, we conjecture that the lag-window spectral density estimator approach performs better in large samples than the periodogram based estimator. Looking at Table 1 below, we see that the sample means and medians of the [11] proposed estimator with values .25 and .24 for $d = .25$, and values of .43 and .43 for $d = .45$ are closer to the real values of $d$when compared to those of [4] ; this is mostly true because the standard errors of the [11] estimator are smaller in general in large samples.

In small samples, the most notable fact is that the maximum likelihood estimator of d is

Table 4.1: Statistics for ARFIMA(0,d,0), samples of 2000, d = .25 / d = .45

	d=.25			d=.45		
Statistics	Gwk-P.Hdk	Reisen	MaxLikhood	Gwk-P.Hdk	Reisen	MaxLikhood
Minimum	-0.02	-0.01	0.20	0.19	0.18	0.40
1st Quartile	0.16	0.17	0.24	0.38	0.36	0.43
Median	0.27	0.24	0.25	0.46	0.43	0.44
Mean	0.26	0.25	0.25	0.46	0.43	0.44
3rd Quartile	0.35	0.32	0.26	0.53	0.49	0.45
Maximum	0.56	0.50	0.28	0.71	0.69	0.48
Standard Error	0.12	0.11	0.02	0.11	0.11	0.02

not even close to the real values of d; even if its standard errors are the smallest, its means and medians values are below the true values $d = .25$ and $d = .45$. On the other hand, the periodogram approach in [4] behaves pretty well in small samples; its sample medians and means values of .26 and .23 for $d = 25$, and .53 and .50 for $d = .45$, are in general the closest possible to the real values of d; the only downside is the standard errors which are the largest. Please see table below for more details:

Table 4.2: Statistics for ARFIMA(0,d,0), Samples of 50, d = .25 / d = .45

	d=.25			d=.45		
Statistics	Gwk-P.Hdk	Reisen	MaxLikhood	Gwk-P.Hdk	Reisen	MaxLikhood
Minimum	-0.89	-0.49	0.00	-0.48	-0.52	0.00
1st Quartile	0.01	-0.06	0.08	0.33	0.21	0.25
Median	0.26	0.17	0.16	0.53	0.39	0.34
Mean	0.23	0.14	0.16	0.50	0.34	0.32
3rd Quartile	0.47	0.31	0.24	0.75	0.52	0.40
Maximum	1.19	0.78	0.41	1.19	0.98	0.48
Standard Error	0.38	0.28	0.11	0.34	0.28	0.11

We also thought that a graphical illustration of the above findings could help visualize differences between estimators. As such, we plotted estimated values of each estimator and

drew horizontal lines at $d = .25$ and $d = .45$. The goal was to show how estimated values of a particular estimator were close to the real values of d. When values of an estimator were concentrated along the horizontal lines, we concluded that the particular estimator behaved well in the defined context. Figures 2 and 3 below show comparison of estimators in large and small samples. These graphics confirm the observations made on the tables; estimated values of d in large samples are closer to the horizontal lines for maximum likelihood estimator whereas estimated values of d in small samples are way off the horizontal lines for the same estimator. In addition, estimated values of d are closer to the horizontal line for Geweke and Porter-Hudak in small samples.

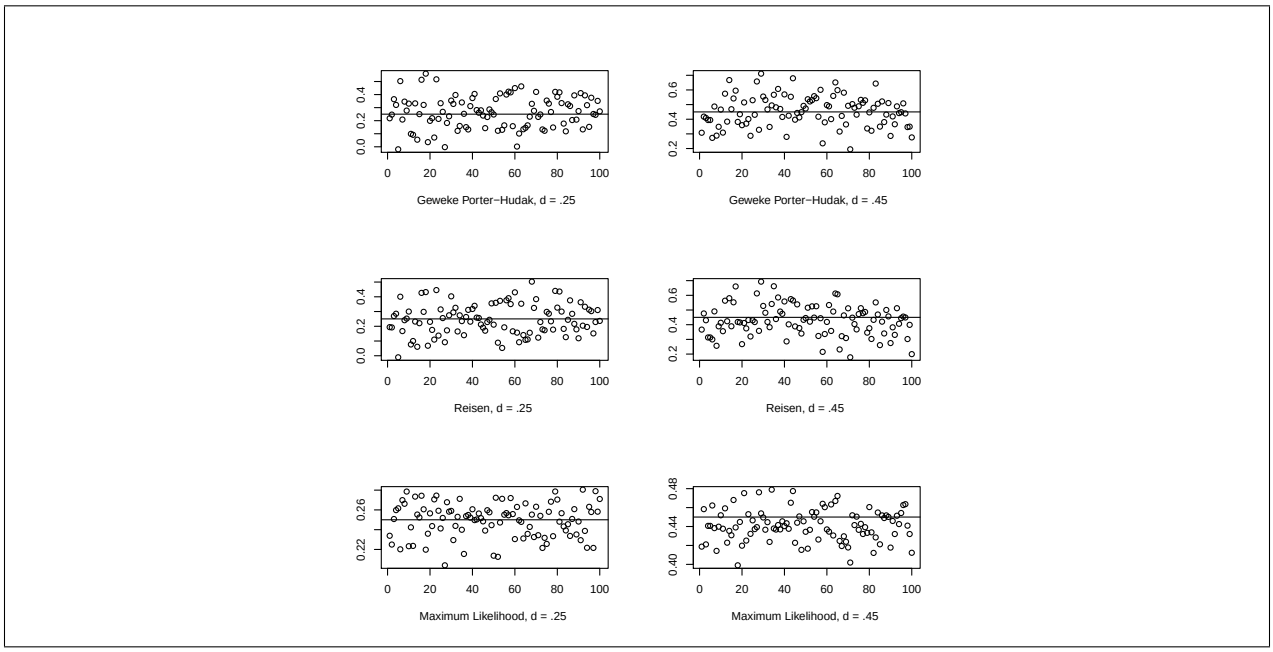

Figure 4.1: Estimators' Goodness of Fit in large ARFIMA(0,d,0) samples

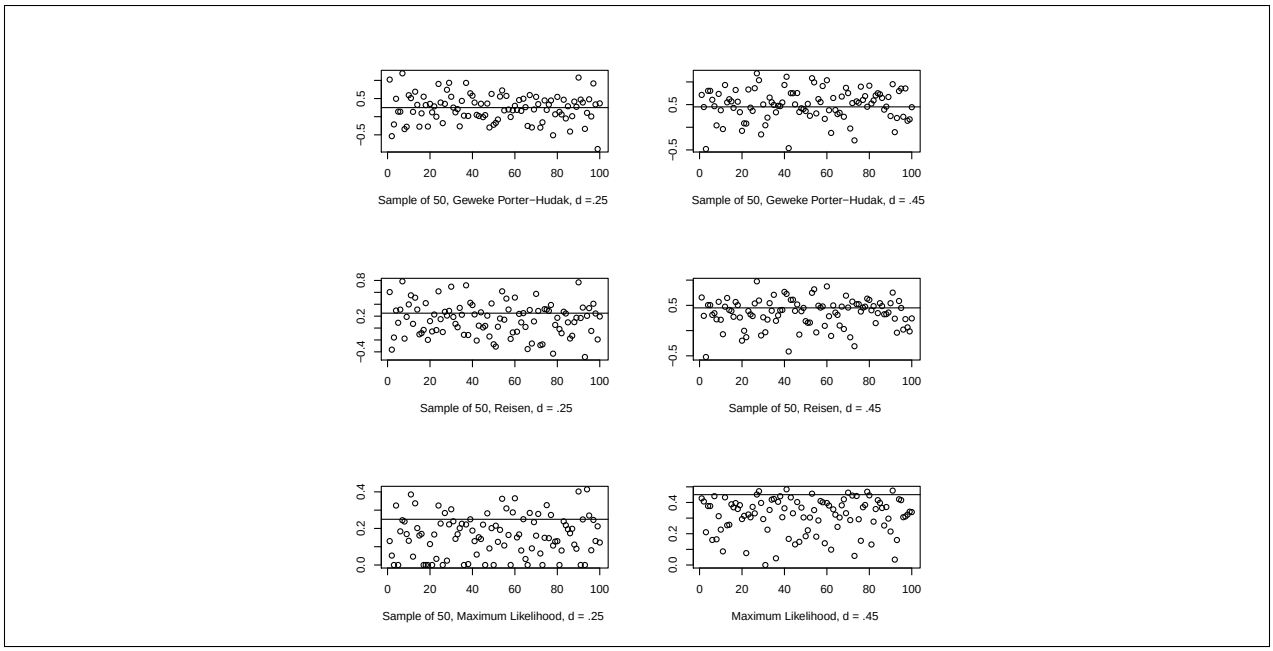

Figure 4.2: Estimators' Goodness of Fit in small ARFIMA(0,d,0) samples

Next, we are going to look at ARFIMA$(0, d, 1)$ tables for large and small samples. However, we first want to address the issue raised by [10] in section 2.3. Hence we created tables for different values of θ in estimating d; we actually kept $|\theta| < 1$.
Looking at the tables below:

Table 4.3: Comparing Models For Best Theta, ARFIMA(0,d,1), d = .25

	Geweke Porter-Hudak	Reisen	Max.Likhood
θ = .9	0.02	0.02	0.00
θ = -.8	0.16	0.27	0.50
θ = .7	0.20	0.20	0.00
θ = -.08	0.16	0.07	0.27
θ = -.02	0.18	0.22	0.26
θ = .01	0.31	0.31	0.24

Table 4.4: Comparing Models For Best Theta, ARFIMA(0,d,1), d = .45

	Geweke Porter-Hudak	Reisen	Max.Likhood
θ = -.9	0.49	0.43	0.50
θ = .8	0.33	0.31	0.00
θ = -.7	0.46	0.43	0.50
θ = .08	0.53	0.56	0.41
θ = .02	0.57	0.49	0.41
θ = -.01	0.45	0.47	0.45

we noticed that the estimated values of d changed when different values of θ were used. In general when we used $|\theta|$ close to 0, for instance .01, the estimated values were closer to the real values of d; whereas when we used $|\theta|$ close to 1, for instance $-.9$ the values of the estimators were off; this issue was more pronounced with the maximum likelihood estimator. Hence we thought that the observations made in [10] about not estimating d separately from θ and ϕ was justified by our data. Meanwhile, based on our observation above, we decided to continue our investigation of the three estimators by using $\theta = .001$ in an ARFIMA$(0, d, 1)$ setting. Yet, looking at the $ARFIMA(0, d, 1)$ tables below, we noted no major difference

with the $ARFIMA(0, d, 0)$ simulation results; the means, medians, and standard errors in both tables, show that in general the maximum likelihood estimator has estimated values closer to the real value of d in large samples whereas the [4] proposed estimator behaves the best in small samples. In addition, the standard errors of the latter are the largest in both large and small samples. The [11] proposed estimator behaves better than that of [4] in large samples; in addition its standard errors are smaller in both large and small samples:

Table 4.5: ARFIMA(0,d,1) stats, samples of 2000, d = .25 / d = .45, θ = .001

	d=.25			d=.45		
Statistics	Gwk-P.Hdk	Reisen	MaxLikhood	Gwk-P.Hdk	Reisen	MaxLikhood
Minimum	-0.03	-0.00	0.21	0.08	0.12	0.38
1st Quartile	0.17	0.17	0.23	0.39	0.36	0.43
Median	0.25	0.24	0.25	0.47	0.44	0.44
Mean	0.26	0.23	0.25	0.46	0.43	0.44
3rd Quartile	0.34	0.31	0.26	0.55	0.50	0.45
Maximum	0.50	0.42	0.29	0.69	0.67	0.48
Standard Error	0.12	0.09	0.02	0.12	0.10	0.02

Table 4.6: ARFIMA(0,d,1) stats, samples of 50, d = .25 / d = .45, θ = .001

	d=.25			d=.45		
Statistics	Gwk-P.Hdk	Reisen	MaxLikhood	Gwk-P.Hdk	Reisen	MaxLikhood
Minimum	-0.72	-0.49	0.00	-1.21	-0.77	0.02
1st Quartile	0.04	-0.00	0.06	0.22	0.17	0.25
Median	0.25	0.16	0.15	0.47	0.36	0.33
Mean	0.29	0.15	0.15	0.39	0.32	0.31
3rd Quartile	0.57	0.29	0.23	0.66	0.52	0.39
Maximum	1.09	0.77	0.40	1.43	0.91	0.46
Standard Error	0.37	0.27	0.11	0.46	0.30	0.11

A Graphical illustration leading to the same conclusion is shown below:

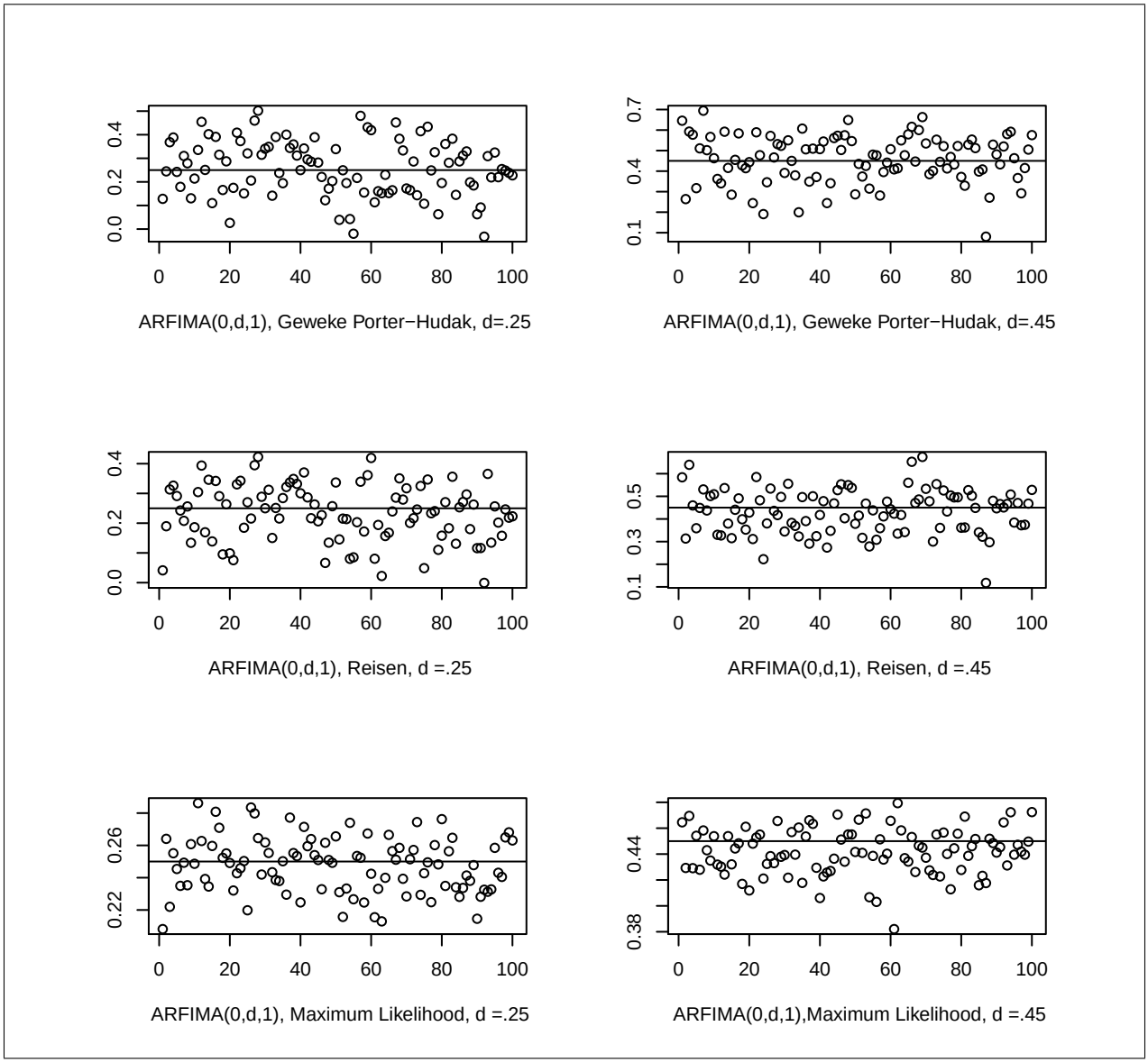

Figure 4.3: Estimators' Goodness of Fit in large ARFIMA(0,d,1) samples

These graphics also confirm that the maximum likelihood estimator behaves well in large samples; for instance, estimated values of d in the maximum likelihood diagram are concentrated along the horizontal lines. On the other hand, the Geweke and Porter-Hudak graphic shows spread out estimated values.

Chapter 5

Conclusion

In investigating three fractional parameter estimators in long memory processes, we used fractional noise or ARFIMA$(0, d, 0)$ and some fractionally integrated moving average or ARFIMA$(0, d, 1)$ models. Simulation via these long memory models produced long memory time series in small and large samples. Computational results have shown that in general the maximum likelihood estimator of d is well behaved in large samples whereas the [4] proposed estimator has estimates closer to the real values of d in small samples. Yet, the [11] proposed estimator behaves better than that of [4] in large sample as well; in addition its standard errors are smaller. Equally important, the standard errors of the periodogram based estimator discussed in [4] are the largest in both large and small samples. Hence, choosing the [4] proposed estimator could lead to lack of precision in estimated values since standard errors of the same are notably the largest.

References

[1] Peter J. Brockwell and Richard A. Davis, *Time Series: Theory and Methods*. Springer, New York NY, pages 466, 1987

[2] Saverio Bivona, Giovanni Bonanno, Riccardo Burlon, Davide Gurrera, Claudio Leone *Stochastic Models For Wind Speed Time Series: A Case Study*. Acta Physica Polonica B, volume 41, pages 1083–1092, 2010.

[3] Luis A. Gil-Alana, *Warming Break Trends and Fractional Integration in the Northern, Southern, and Global Temperature Anomaly Series*. Journal of Atmospheric and Oceanic Technology, volume 25, pages 570–578, 2007.

[4] John Geweke and Susan Porter-Hudak, *The Estimation and Application of Long Memory Time Series Models*. Journal of Time Series Analysis, volume 4, pages 221–237, 1983.

[5] C. W. J. Granger and Roselyne Joyeux, *An Introduction to Long Memory Time Series Models and Fractional Differencing*. Journal of Time Series Analysis, volume 1, pages 15–29, 1980.

[6] I. S. Gradshteyn and I. M Ryzhik, *Tables of Integral, Series, and Products*. Academic Press, New York, 1965.

[7] J. R. M. Hosking, *Fractional Differencing*. Biometrika, volume 68, pages 165–176, 1981.

[8] John Haslett and Adrian E. Raftery, *Space-Time Modeling With Long-Memory Dependence: Assessing Ireland's Wind Power Resource*. Journal of Applied Statistics, volume 38, pages 1–50, 1989.

[9] Christian Kleiber and Achim Zeileis, *Applied Econometrics With R*. Springer, New York NY, 2008.

[10] M. S. Peiris and J. R. Court, *A Note On The Estimation of Degree of Differencing in Long Memory TimeSeries Analysis*. Probability and Mathematical Statistics, volume 14, pages 223–229, 1993.

[11] Valderio A. Reisen, *Estimation of The Fractional Difference Parameter in ARIMA(p, d, q) Models Using The Smoothed Periodogram*. Journal of Time Series Analysis, volume 15, pages 335–350, 1994.

[12] Wei, William W.S, *Time Series Analysis: Univariate and Multivariate Methods*. Addison Wesley, Readings Massachusetts, 2nd Edition, 2006.

Appendix

R Code

```
####################################
## code used for Master's Thesis  ##
####################################

## Create initial M=200 X's

# Some library to be used

library(Rmpfr)
library(fractal)
library(sapa)
library(fracdiff)
library(longmemo)

# Initialization of values

d = .25
rho_hat = p = w = Z = B = b = x = 0

# Generate 200 std normal

z = rnorm(200,mean=0, sd=1)

# generate correlation at lag h
for ( h in 1:199) {
 w[h] = ((h-1+d)/(h-d))
  for (i in 2:199){
    p[1] = w[1]
    p[i] = p[i-1]*w[i]
   }
}
 p # correlation at lag h=1,2,...

rho = c(1,p)
l = toeplitz(rho) # create a toeplitz data frame
```

```
#  create initial 200 X's

Ml = as.matrix (l)
W = chol(Ml)
L = t(chol(Ml))
X = W%*%z # L'e'

X # initial 200 X's


# Generate j! could have used factorial(mpfr(200, prec=100))

v <- j <- mpfr(1, prec = 100)
Const(name = c("gamma"), 128L)
for (i in 1:200){
v[i] = i
for (i in 2:200){
j[1]=1
j[i] = j[i-1]*v[i]
}
}
j

# Generate aj

l <- A <- a <- mpfr(0,prec = 100)
Const(name = c("gamma"), 128L)
for (k in 1:200){
l[k] = (k-1-d)
  for (i in 2:200){
    A[1] = l[1]
    A[i] = A[i-1]*l[i]

    }
}

for ( i in 1:200){
  a[i]= (1/j[i])*A[i]
}
a # aj's

# transform a-j-A to as.numeric
```

```
a <- as.numeric(a)
j <- as.numeric(j)
A <- as.numeric(A)

## Generate the rest of 2000 X's

Z = rnorm(2000,mean=0, sd=1)

summ=numeric(0)
for (i in 201:2000){
g = 0
for (j in 1:200){
g = g + a[j]*X[i-j]
}

summ[i] = g

X[i] = Z[i]- summ[i]

}
X # 2000 X's

# Saving datasets
setwd("C:/Documents and Settings/User/My Documents/Thesis/data")
save(X, file="X")
# load("X") brings X back in console

X <- ts(X) # transform X as time series

## ACF of X's

ts1 <- X
ts2 =ts(ts1)
ts3 = 0
 for (i in 1:2000){
g1 = 0
g1 = g1 + (ts1[i]- mean(ts1))**2
      ts3[i] = g1
}
A = sum(ts3)

x=0
```

```
for (i in 1:1999){
g2 = 0
for (j in 1:(2000-i)){
g2 = g2 + (ts1[j]- mean(ts1))*(ts1[j+i]-mean(ts1))
}
      x[i] = g2
}

RHO_HAT = 0
for ( i in 1:1999){
RHO_HAT[i] = x[i]/A
}
RHO_HAT # ACF of X's; could have used just acf(X)

## Create ARFIMA(0,d,0) samples of size 2000 with d=.25/.45. (Xi for d=.25) &
(Xi.c for d=.45)

setwd("C:/Documents and Settings/User/My Documents/Thesis/data")

# should use load("Xi") to bring Xi in console

data <- data.frame(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16,X17,
X18,X19,X20,X21,X22,X23,X24,X25,X26,X27,X28,X29,X30,X31,X32,
X33,X34,X35,X36,X37,X38,X39,X40,X41,X42,X43,X44,X45,X46,X47,
X48,X49,X50,X51,X52,X53,X54,X55,X56,X57,X58,X59,X60,X61,X62,
X63,X64,X65,X66,X67,X68,X69,X70,X71,X72,X73,X74,X75,X76,X77,
X78,X79,X80,X81,X82,X83,X84,X85,X86,X87,X88,X89,X90,X91,X92,
X93,X94,X95,X96,X97,X98,X99,X100)
save(data,file=("data"))

# load("Xi.c") for d=.45

data.c <- data.frame(X1.c,X2.c,X3.c,X4.c,X5.c,X6.c,X7.c,X8.c,X9.c,X10.c,X11.c,X12.c,X13.
 X18.c,X19.c,X20.c,X21.c,X22.c,X23.c,X24.c,X25.c,X26.c,X27.c,X28.c,X29.c,X30.c,X31.c,X32
 X33.c,X34.c,X35.c,X36.c,X37.c,X38.c,X39.c,X40.c,X41.c,X42.c,X43.c,X44.c,X45.c,X46.c,X47
 X48.c,X49.c,X50.c,X51.c,X52.c,X53.c,X54.c,X55.c,X56.c,X57.c,X58.c,X59.c,X60.c,X61.c,X62
 X63.c,X64.c,X65.c,X66.c,X67.c,X68.c,X69.c,X70.c,X71.c,X72.c,X73.c,X74.c,X75.c,X76.c,X77
 X78.c,X79.c,X80.c,X81.c,X82.c,X83.c,X84.c,X85.c,X86.c,X87.c,X88.c,X89.c,X90.c,X91.c,X92
X93.c,X94.c,X95.c,X96.c,X97.c,X98.c,X99.c,X100.c)
save(data.c,file=("data.c"))
```

```
## Create ARFIMA(0,d,0) samples of size 50 with d=.25/.45

X100.50 <- X100[1951:2000]
X100.50c <- X100.c[1951:2000]

data.50c <- data.frame(X1.50c,X2.50c,X3.50c,X4.50c,X5.50c,X6.50c,X7.50c,X8.50c,X9.50c,X1
 X18.50c,X19.50c,X20.50c,X21.50c,X22.50c,X23.50c,X24.50c,X25.50c,X26.50c,X27.50c,X28.50c
 X33.50c,X34.50c,X35.50c,X36.50c,X37.50c,X38.50c,X39.50c,X40.50c,X41.50c,X42.50c,X43.50c
 X48.50c,X49.50c,X50.50c,X51.50c,X52.50c,X53.50c,X54.50c,X55.50c,X56.50c,X57.50c,X58.50c
 X63.50c,X64.50c,X65.50c,X66.50c,X67.50c,X68.50c,X69.50c,X70.50c,X71.50c,X72.50c,X73.50c
 X78.50c,X79.50c,X80.50c,X81.50c,X82.50c,X83.50c,X84.50c,X85.50c,X86.50c,X87.50c,X88.50c
X93.50c,X94.50c,X95.50c,X96.50c,X97.50c,X98.50c,X99.50c,X100.50c)

data.50 <- data.frame(X1.50,X2.50,X3.50,X4.50,X5.50,X6.50,X7.50,X8.50,X9.50,X10.50,X11.5
 X18.50,X19.50,X20.50,X21.50,X22.50,X23.50,X24.50,X25.50,X26.50,X27.50,X28.50,X29.50,X30
 X33.50,X34.50,X35.50,X36.50,X37.50,X38.50,X39.50,X40.50,X41.50,X42.50,X43.50,X44.50,X45
 X48.50,X49.50,X50.50,X51.50,X52.50,X53.50,X54.50,X55.50,X56.50,X57.50,X58.50,X59.50,X60
 X63.50,X64.50,X65.50,X66.50,X67.50,X68.50,X69.50,X70.50,X71.50,X72.50,X73.50,X74.50,X75
 X78.50,X79.50,X80.50,X81.50,X82.50,X83.50,X84.50,X85.50,X86.50,X87.50,X88.50,X89.50,X90
X93.50,X94.50,X95.50,X96.50,X97.50,X98.50,X99.50,X100.50)

save(data.50, file=("data.50"))
save(data.50c, file=("data.50c"))

## Create ARFIMA(0,d,1) datasets with different \theta using function fracdiff.sim()

ARFIMA.0d1.45.v <- fracdiff.sim(n=2000, ma=c(-.8),d=.45)
ARFIMA.0d1.45.w <- fracdiff.sim(n=2000, ma=c(.4),d=.45)
ARFIMA.0d1.45.x <- fracdiff.sim(n=2000, ma=c(-.04),d=.45)
ARFIMA.0d1.45.y <- fracdiff.sim(n=2000, ma=c(.08),d=.45)
ARFIMA.0d1.45.z <- fracdiff.sim(n=2000, ma=c(.01),d=.45)

ARFIMA.0d1.45xy <- data.frame(ARFIMA.0d1.45.v$series,ARFIMA.0d1.45.w$series,ARFIMA.0d1.4
                                ARFIMA.0d1.45.y$series,ARFIMA.0d1.45.z$series)
save(ARFIMA.0d1.45xy, file=("data.ARFIMA.0d1.45xy"))


ARFIMA.0d1.25.v <- fracdiff.sim(n=2000, ma=c(.8),d=.25)
ARFIMA.0d1.25.w <- fracdiff.sim(n=2000, ma=c(-.4),d=.25)
ARFIMA.0d1.25.x <- fracdiff.sim(n=2000, ma=c(.04),d=.25)
ARFIMA.0d1.25.y <- fracdiff.sim(n=2000, ma=c(-.08),d=.25)
```

```
ARFIMA.0d1.25.z <- fracdiff.sim(n=2000, ma=c(-.01),d=.25)

ARFIMA.0d1.25xy <- data.frame(ARFIMA.0d1.25.v$series,ARFIMA.0d1.25.w$series,ARFIMA.0d1.2
                                ARFIMA.0d1.25.y$series,ARFIMA.0d1.25.z$series)
save(ARFIMA.0d1.25xy, file=("ARFIMA.0d1.25xy"))


## Create ARFIMA(0,d,1) datasets with  \theta = .001

ARFIMA.0d1.45.100 <- fracdiff.sim(n=2000, ma=c(.001),d=.45)
ARFIMA.0d1.25.100 <- fracdiff.sim(n=2000, ma=c(.001),d=.25)
v100 <- ARFIMA.0d1.25.100$series
v100.c <- ARFIMA.0d1.45.100$series

ARFIMA.0d1.25 <- data.frame(v1,v2,v3,v4,v5,v6,v7,v8,v9,
                            v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,
                            v20,v21,v22,v23,v24,v25,v26,v27,v28,v29,
                            v30,v31,v32,v33,v34,v35,v36,v37,v38,v39,
                            v40,v41,v42,v43,v44,v45,v46,v47,v48,v49,
                            v50,v51,v52,v53,v54,v55,v56,v57,v58,v59,
                            v60,v61,v62,v63,v64,v65,v66,v67,v68,v69,
                            v70,v71,v72,v73,v74,v75,v76,v77,v78,v79,
                            v80,v81,v82,v83,v84,v85,v86,v87,v88,v89,
                       v90,v91,v92,v93,v94,v95,v96,v97,v98,v99,v100)

save(ARFIMA.0d1.25, file=("ARFIMA.0d1.25"))


ARFIMA.0d1.45 <- data.frame(v1.c,v2.c,v3.c,v4.c,v5.c,v6.c,v7.c,v8.c,v9.c,
                            v10.c,v11.c,v12.c,v13.c,v14.c,v15.c,v16.c,v17.c,v18.c,v19.c,
                            v20.c,v21.c,v22.c,v23.c,v24.c,v25.c,v26.c,v27.c,v28.c,v29.c,
                            v30.c,v31.c,v32.c,v33.c,v34.c,v35.c,v36.c,v37.c,v38.c,v39.c,
                            v40.c,v41.c,v42.c,v43.c,v44.c,v45.c,v46.c,v47.c,v48.c,v49.c,
                            v50.c,v51.c,v52.c,v53.c,v54.c,v55.c,v56.c,v57.c,v58.c,v59.c,
                            v60.c,v61.c,v62.c,v63.c,v64.c,v65.c,v66.c,v67.c,v68.c,v69.c,
                            v70.c,v71.c,v72.c,v73.c,v74.c,v75.c,v76.c,v77.c,v78.c,v79.c,
                            v80.c,v81.c,v82.c,v83.c,v84.c,v85.c,v86.c,v87.c,v88.c,v89.c,
                       v90.c,v91.c,v92.c,v93.c,v94.c,v95.c,v96.c,v97.c,v98.c,v99.c,v100.c

save(ARFIMA.0d1.45, file=("ARFIMA.0d1.45"))
```

```
## Create samples of size 50 for ARFIMA(0,d,1) datasets with  \theta = .001

w.25.50.100 <- v100[1951:2000]
w.45.50.100 <- v100.c[1951:2000]

ARFIMA.0d1.25.50 <- data.frame(w.25.50.1,w.25.50.2,w.25.50.3,w.25.50.4,w.25.50.5,w.25.50
                               w.25.50.10,w.25.50.11,w.25.50.12,w.25.50.13,w.25.50.14,w.
                               w.25.50.20,w.25.50.21,w.25.50.22,w.25.50.23,w.25.50.24,w.
                               w.25.50.30,w.25.50.31,w.25.50.32,w.25.50.33,w.25.50.34,w.
                               w.25.50.40,w.25.50.41,w.25.50.42,w.25.50.43,w.25.50.44,w.
                               w.25.50.50,w.25.50.51,w.25.50.52,w.25.50.53,w.25.50.54,w.
                               w.25.50.60,w.25.50.61,w.25.50.62,w.25.50.63,w.25.50.64,w.
                               w.25.50.70,w.25.50.71,w.25.50.72,w.25.50.73,w.25.50.74,w.
                               w.25.50.80,w.25.50.81,w.25.50.82,w.25.50.83,w.25.50.84,w.
                               w.25.50.90,w.25.50.91,w.25.50.92,w.25.50.93,w.25.50.94,w.

save(ARFIMA.0d1.25.50, file=("ARFIMA.0d1.25.50"))


ARFIMA.0d1.45.50 <- data.frame(w.45.50.1,w.45.50.2,w.45.50.3,w.45.50.4,w.45.50.5,w.45.50
                               w.45.50.10,w.45.50.11,w.45.50.12,w.45.50.13,w.45.50.14,w.
                               w.45.50.20,w.45.50.21,w.45.50.22,w.45.50.23,w.45.50.24,w.
                               w.45.50.30,w.45.50.31,w.45.50.32,w.45.50.33,w.45.50.34,w.
                               w.45.50.40,w.45.50.41,w.45.50.42,w.45.50.43,w.45.50.44,w.
                               w.45.50.50,w.45.50.51,w.45.50.52,w.45.50.53,w.45.50.54,w.
                               w.45.50.60,w.45.50.61,w.45.50.62,w.45.50.63,w.45.50.64,w.
                               w.45.50.70,w.45.50.71,w.45.50.72,w.45.50.73,w.45.50.74,w.
                               w.45.50.80,w.45.50.81,w.45.50.82,w.45.50.83,w.45.50.84,w.
                               w.45.50.90,w.45.50.91,w.45.50.92,w.45.50.93,w.45.50.94,w.

save(ARFIMA.0d1.45.50, file=("ARFIMA.0d1.45.50"))

## Formula for estimation of d



# Geweke and Porter-Hudak (GPH)

bandw.exp = 0.5
n <- length(X)
    g <- trunc(n^bandw.exp)
    j <- 1:g
```

```
    kk <- 1:(n - 1)
    w <- 2 * pi * j/n
    mx <- mean(X)
    var.x <- sum((X - mx)^2)/n
    cov.x <- numeric(n - 1)
for (k in kk) cov.x[k] <- sum((X[1:(n - k)] - mx) * (X[(1 + k):n] - mx))/n
    periodogram <- numeric(g)
periodogram <- numeric(g)
for (i in 1:g) periodogram[i] <- var.x + 2 * sum(cov.x * cos(w[i] * kk))
pos <- j[periodogram > 0]
xGPH.reg <- 2 * log(2 * sin(w[pos]/2))
YGPH.reg <- log(periodogram[pos]/(2 * pi))
lm(YGPH.reg~xGPH.reg)



# Reisen
bandw.exp = 0.5
 beta = 0.9
n <- length(X)
    g <- trunc(n^bandw.exp)
    j <- 1:g
    kk <- 1:(n - 1)
    w <- 2 * pi * j/n
    mx <- mean(X)
    var.x <- sum((X - mx)^2)/n
    cov.x <- numeric(n - 1)
    for (k in kk) cov.x[k] <- sum((X[1:(n - k)] - mx) * (X[(1 +
        k):n] - mx))/n
    M <- trunc(n^beta)
    M2 <- M%/%2
    pw <- numeric(n - 1)
    for (k in kk) {
        A_k <- k/M
        pw[k] <- if (k <= M2)
            1 - 6 * A_k^2 * (1 - A_k)
        else if (k <= M)
            2 * (1 - A_k)^3
        else 0
    }
    periodogram <- numeric(g)
    for (i in 1:g) periodogram[i] <- var.x + 2 * sum(cov.x *
```

```
        pw * cos(w[i] * kk))
    pos <- j[periodogram > 0]
    YRsen.reg <- log(periodogram[pos]/(2 * pi))
    xRsen.reg <- 2 * log(2 * sin(w[pos]/2))
    lm(YRsen.reg ~ xRsen.reg)

# For Maximum Likelihood Estimator, fracdiff(X, nar = 0 , nma = 0) was used
# See R console for fracdiff() code:

> library(fracdiff)
> fracdiff
function (x, nar = 0, nma = 0, ar = rep(NA, max(nar, 1)), ma = rep(NA,
    max(nma, 1)), dtol = NULL, drange = c(0, 0.5), h, M = 100)
{
    cl <- match.call()
    if (any(is.na(x)))
        stop("missing values not allowed in time series")
    if (is.matrix(x) && ncol(x) > 2)
        stop("multivariate time series not allowed")
    n <- length(x)
    if (round(nar) != nar || nar < 0 || round(nma) != nma ||
        nma < 0)
        stop("'nar' and 'nma' must be non-negative integer numbers")
    npq <- as.integer(nar + nma)
    npq1 <- npq + 1L
    lenw <- max(npq + 2 * (n + M), 3 * n + (n + 6) * npq + npq%/%2 +
        1, 31 * 12, (3 + 2 * npq1) * npq1 + 1)
    lenw <- as.integer(lenw)
    ar[is.na(ar)] <- 0
    ma[is.na(ma)] <- 0
    if (is.null(dtol))
        dtol <- .Machine$double.eps^0.25
    x <- as.double(x)
    fdf <- .C("fracdf", x, n, as.integer(M), as.integer(nar),
        as.integer(nma), dtol = as.double(dtol), drange = as.double(drange),
        hood = double(1), d = double(1), ar = as.double(ar),
        ma = as.double(ma), w = double(lenw), lenw = lenw, iw = integer(npq),
        info = integer(1), .Machine$double.xmin, .Machine$double.xmax,
        .Machine$double.neg.eps, .Machine$double.eps, PACKAGE = "fracdiff")[c("dtol",
        "drange", "hood", "d", "ar", "ma", "w", "lenw", "info")]
    fd.msg <- if (fdf$info) {
        msg <- switch(fdf$info, stop("insufficient workspace; need ",
```

```
            fdf$lenw, " instead of just ", lenw), stop("error in gamma function"),
            stop("invalid MINPACK input"), "warning in gamma function",
            "C fracdf() optimization failure", "C fracdf() optimization limit reached")
        warning(msg, call. = FALSE, immediate. = TRUE)
        msg
    }
    else "ok"
    if (nar == 0)
        fdf$ar <- numeric(0)
    if (nma == 0)
        fdf$ma <- numeric(0)
    hess <- .C("fdhpq", hess = matrix(double(1), npq1, npq1),
        npq1, fdf$w, PACKAGE = "fracdiff")$hess
    fdc <- .fdcov(x, fdf$d, h, nar = nar, nma = nma, hess = hess,
        fdf.work = fdf$w)
    dimnames(hess) <- dimnames(fdc$covariance.dpq)
    hess[1, ] <- fdc$hd
    hess[row(hess) > col(hess)] <- hess[row(hess) < col(hess)]
    structure(list(log.likelihood = fdf$hood, n = n, msg = c(fracdf = fd.msg,
        fdcov = fdc$msg), d = fdf$d, ar = fdf$ar, ma = fdf$ma,
        covariance.dpq = fdc$covariance.dpq, stderror.dpq = if (fdc$se.ok) fdc$stderror.
        correlation.dpq = if (fdc$se.ok) fdc$correlation.dpq,
        h = fdc$h, d.tol = fdf$dtol, M = M, hessian.dpq = hess,
        length.w = lenw, call = cl), class = "fracdiff")
}
<environment: namespace:fracdiff>

##############################################################################

## Tables creation

# Create statistics for X's, d=.25

setwd("C:/Users/Owner/Documents/Thesis/data")
Geweke.Porter.Hudak <- Risen <- Max.Likhood <- 0
load("data")

for ( i in 1:100){
{
gph <- fdGPH(data[,i])
Geweke.Porter.Hudak[i] = gph$d
}
```

```
{
risen <- fdSperio(data[,i])
Risen[i] = risen$d

}

{
mle <- fracdiff(data[,i], nar = 0 , nma = 0)
Max.Likhood[i] = mle$d
}

}

data.d <- data.frame(Geweke.Porter.Hudak,Risen,Max.Likhood)

save(data.d,file=("data.d"))
save(Geweke.Porter.Hudak,file=("Geweke.Porter.Hudak"))
save(Risen,file=("Risen"))
save(Max.Likhood,file=("Max.Likhood"))

load("data.d")
load("Geweke.Porter.Hudak")
load("Risen")
load("Max.Likhood")
means <- c(mean(Geweke.Porter.Hudak),mean(Risen),mean(Max.Likhood))
sds <- c(sd(Geweke.Porter.Hudak),sd(Risen),sd(Max.Likhood))
mins <- c(min(Geweke.Porter.Hudak),min(Risen),min(Max.Likhood))
maxs <- c(max(Geweke.Porter.Hudak),max(Risen),max(Max.Likhood))
stats <- data.frame(means,sds,mins,maxs,row.names=c("Geweke.Porter.Hudak","Risen","Max.L
names(stats) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table <- xtable(stats,caption=" Statistics for d = .25 (1)",caption.placement="top
print(stats.table,include.rownmames=True)

Sum.stats <- c(summary(Geweke.Porter.Hudak),summary(Risen),summary(Max.Likhood))
Sum.stats <- matrix(Sum.stats,nrow=6)
Sum.stats <- data.frame(Sum.stats,row.names=c("Minimum","1rst Quartile","Median","Mean",
names(Sum.stats) <- c("Geweke.Porter.Hudak","Reisen","Max.Likhood")
library(xtable)
Sum.stats.table <- xtable(Sum.stats,caption="Statistics for d = .25 ",caption.placement=
print(Sum.stats.table,include.rownmames=True)
```

```
# Create statistics for X's, d=.45

load("data.c")
Geweke.Porter.Hudak.c <- Risen.c <- Max.Likhood.c <- 0

for ( i in 1:100){
{
gph <- fdGPH(data.c[,i])
Geweke.Porter.Hudak.c[i] = gph$d

}


{
risen <- fdSperio(data.c[,i])
Risen.c[i] = risen$d

}

{
mle <- fracdiff(data.c[,i], nar = 0 , nma = 0)
Max.Likhood.c[i] = mle$d
}

}

data.d.c <- data.frame(Geweke.Porter.Hudak.c,Risen.c,Max.Likhood.c)

save(data.d.c,file=("data.d.c"))
save(Geweke.Porter.Hudak.c,file=("Geweke.Porter.Hudak.c"))
save(Risen.c,file=("Risen.c"))
save(Max.Likhood.c,file=("Max.Likhood.c"))

load("data.d.c")
load("Geweke.Porter.Hudak.c")
load("Risen.c")
load("Max.Likhood.c")

means.c <- c(mean(Geweke.Porter.Hudak.c),mean(Risen.c),mean(Max.Likhood.c))
sds.c <- c(sd(Geweke.Porter.Hudak.c),sd(Risen.c),sd(Max.Likhood.c))
```

```
mins.c <- c(min(Geweke.Porter.Hudak.c),min(Risen.c),min(Max.Likhood.c))
maxs.c <- c(max(Geweke.Porter.Hudak.c),max(Risen.c),max(Max.Likhood.c))
stats.c <- data.frame(means.c,sds.c,mins.c,maxs.c,row.names=c("Geweke.Porter.Hudak","Ris
names(stats.c) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table.c <- xtable(stats.c,caption=" Statistics for d = .45  (2)",label="tab: Three
print(stats.table.c,include.rownmames=True)

Sum.stats.c <- c(summary(Geweke.Porter.Hudak.c),summary(Risen.c),summary(Max.Likhood.c))
Sum.stats.c <- matrix(Sum.stats.c,nrow=6)
Sum.stats.c <- data.frame(Sum.stats.c,row.names=c("Minimum","1rst Quartile","Median","Me
names(Sum.stats.c) <- c("Geweke.Porter.Hudak","Reisen","Max.Likhood")
library(xtable)
Sum.stats.table.c <- xtable(Sum.stats.c,caption="Statistics for d = .45 ", caption.place
print(Sum.stats.table.c,include.rownmames=True)




# Create statistics for X's, d=.25,  Samples of 50

Geweke.Porter.Hudak50 <- Risen50 <- Max.Likhood50 <- 0
load("data.50")

for ( i in 1:100){
{
gph <- fdGPH(data.50[,i])
Geweke.Porter.Hudak50[i] = gph$d

}


{
risen <- fdSperio(data.50[,i])
Risen50[i] = risen$d

}

{
mle <- fracdiff(data.50[,i], nar = 0 , nma = 0)
Max.Likhood50[i] = mle$d
}
```

```
}

data.d.50 <- data.frame(Geweke.Porter.Hudak50,Risen50,Max.Likhood50) # estimator dataset

save(data.d.50, file=("data.d.50"))
save(Geweke.Porter.Hudak50, file=("Geweke.Porter.Hudak50"))
save(Risen50, file=("Risen50"))
save(Max.Likhood50, file=("Max.Likhood50"))

means.50 <- c(mean(Geweke.Porter.Hudak50),mean(Risen50),mean(Max.Likhood50))
sds.50 <- c(sd(Geweke.Porter.Hudak50),sd(Risen50),sd(Max.Likhood50))
mins.50 <- c(min(Geweke.Porter.Hudak50),min(Risen50),min(Max.Likhood50))
maxs.50 <- c(max(Geweke.Porter.Hudak50),max(Risen50),max(Max.Likhood50))
stats.50 <- data.frame(means.50,sds.50,mins.50,maxs.50,row.names=c("Geweke.Porter.Hudak"
names(stats.50) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table.50 <- xtable(stats.50,caption=" Statistics for Samples of 50, d = .25 ",labe
print(stats.table.50,include.rownmames=True)

Sum.stats.50 <- c(summary(Geweke.Porter.Hudak50),summary(Risen50),summary(Max.Likhood50)
Sum.stats.50 <- matrix(Sum.stats.50,nrow=6)
Sum.stats.50 <- data.frame(Sum.stats.50,row.names=c("Minimum","1rst Quartile","Median","
names(Sum.stats.50) <- c("Geweke.Porter.Hudak","Reisen","Max.Likhood")
library(xtable)
Sum.stats.table.50 <- xtable(Sum.stats.50,caption="Statistics for Samples of 50, d = .25
print(Sum.stats.table.50,include.rownmames=True)



# Create statistics for X's, d=.45,  Samples of 50

Geweke.Porter.Hudak50c <- Risen50c <- Max.Likhood50c <- 0
load("data.50c")

for ( i in 1:100){
{
gph <- fdGPH(data.50c[,i])
Geweke.Porter.Hudak50c[i] = gph$d

}
```

```
{
risen <- fdSperio(data.50c[,i])
Risen50c[i] = risen$d

}

{
mle <- fracdiff(data.50c[,i], nar = 0 , nma = 0)
Max.Likhood50c[i] = mle$d
}

}

data.d.50c <- data.frame(Geweke.Porter.Hudak50c,Risen50c,Max.Likhood50c)

save(data.d.50c, file=("data.d.50c"))
save(Geweke.Porter.Hudak50c, file=("Geweke.Porter.Hudak50c"))
save(Risen50c, file=("Risen50c"))
save(Max.Likhood50c, file=("Max.Likhood50c"))

means.50c <- c(mean(Geweke.Porter.Hudak50c),mean(Risen50c),mean(Max.Likhood50c))
sds.50c <- c(sd(Geweke.Porter.Hudak50c),sd(Risen50c),sd(Max.Likhood50c))
mins.50c <- c(min(Geweke.Porter.Hudak50c),min(Risen50c),min(Max.Likhood50c))
maxs.50c <- c(max(Geweke.Porter.Hudak50c),max(Risen50c),max(Max.Likhood50c))
stats.50c <- data.frame(means.50c,sds.50c,mins.50c,maxs.50c,row.names=c("Geweke.Porter.H
names(stats.50c) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table.50c <- xtable(stats.50c,caption=" Statistics for Samples of 50, d = .45 ",la
print(stats.table.50c,include.rownmames=True)

Sum.stats.50c <- c(summary(Geweke.Porter.Hudak50c),summary(Risen50c),summary(Max.Likhood
Sum.stats.50c <- matrix(Sum.stats.50c,nrow=6)
Sum.stats.50c <- data.frame(Sum.stats.50c,row.names=c("Minimum","1rst Quartile","Median"
names(Sum.stats.50c) <- c("Geweke.Porter.Hudak","Risen","Max.Likhood")
library(xtable)
Sum.stats.table.50c <- xtable(Sum.stats.50c,caption="Statistics for Samples of 50, d = .
print(Sum.stats.table.50c,include.rownmames=True)

# Create statistics for ARFIMA(0,d,1), d=.25

Geweke.Porter.Hudaks25 <- Risens25 <- Max.Likhoods25 <- 0
```

```
load("ARFIMA.0d1.25")
data.s25 <- ARFIMA.0d1.25


for ( i in 1:100){
{
gph <- fdGPH(data.s25[,i])
Geweke.Porter.Hudaks25[i] = gph$d

}


{
risen <- fdSperio(data.s25[,i])
Risens25[i] = risen$d

}

{
mle <- fracdiff(data.s25[,i], nar = 0 , nma = 0)
Max.Likhoods25[i] = mle$d
}

}

data.d.s25 <- data.frame(Geweke.Porter.Hudaks25,Risens25,Max.Likhoods25) # estimator dat

save(data.d.s25, file=("data.d.s25"))
save(Geweke.Porter.Hudaks25, file=("Geweke.Porter.Hudaks25"))
save(Risens25, file=("Risens25"))
save(Max.Likhoods25, file=("Max.Likhoods25"))

means.s25 <- c(mean(Geweke.Porter.Hudaks25),mean(Risens25),mean(Max.Likhoods25))
sds.s25 <- c(sd(Geweke.Porter.Hudaks25),sd(Risens25),sd(Max.Likhoods25))
mins.s25 <- c(min(Geweke.Porter.Hudaks25),min(Risens25),min(Max.Likhoods25))
maxs.s25 <- c(max(Geweke.Porter.Hudaks25),max(Risens25),max(Max.Likhoods25))
stats.s25 <- data.frame(means.s25,sds.s25,mins.s25,maxs.s25,row.names=c("Geweke.Porter.H
names(stats.s25) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table.s25 <- xtable(stats.s25,caption=" Statistics for ARFIMA(0,d,1), d = .25 (2)
print(stats.table.s25,include.rownmames=True)
```

```
Sum.stats.s25 <- c(summary(Geweke.Porter.Hudaks25),summary(Risens25),summary(Max.Likhood
Sum.stats.s25 <- matrix(Sum.stats.s25,nrow=6)
Sum.stats.s25 <- data.frame(Sum.stats.s25,row.names=c("Minimum","1rst Quartile","Median"
names(Sum.stats.s25) <- c("Geweke.Porter.Hudak","Reisen","Max.Likhood")
library(xtable)
Sum.stats.table.s25 <- xtable(Sum.stats.s25,caption="Statistics for ARFIMA(0,d,1), d = .
print(Sum.stats.table.s25,include.rownmames=True)

# Create statistics for ARFIMA(0,d,1), d=.45

Geweke.Porter.Hudaks45 <- Risens45 <- Max.Likhoods45 <- 0
load("ARFIMA.0d1.45")
data.s45 <- ARFIMA.0d1.45


for ( i in 1:100){
{
gph <- fdGPH(data.s45[,i])
Geweke.Porter.Hudaks45[i] = gph$d

}


{
risen <- fdSperio(data.s45[,i])
Risens45[i] = risen$d

}

{
mle <- fracdiff(data.s45[,i], nar = 0 , nma = 0)
Max.Likhoods45[i] = mle$d
}

}

data.d.s45 <- data.frame(Geweke.Porter.Hudaks45,Risens45,Max.Likhoods45) # estimator dat

save(data.d.s45, file=("data.d.s45"))
save(Geweke.Porter.Hudaks45, file=("Geweke.Porter.Hudaks45"))
save(Risens45, file=("Risens45"))
save(Max.Likhoods45, file=("Max.Likhoods45"))
```

```
means.s45 <- c(mean(Geweke.Porter.Hudaks45),mean(Risens45),mean(Max.Likhoods45))
sds.s45 <- c(sd(Geweke.Porter.Hudaks45),sd(Risens45),sd(Max.Likhoods45))
mins.s45 <- c(min(Geweke.Porter.Hudaks45),min(Risens45),min(Max.Likhoods45))
maxs.s45 <- c(max(Geweke.Porter.Hudaks45),max(Risens45),max(Max.Likhoods45))
stats.s45 <- data.frame(means.s45,sds.s45,mins.s45,maxs.s45,row.names=c("Geweke.Porter.H
names(stats.s45) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table.s45 <- xtable(stats.s45,caption=" Statistics for ARFIMA(0,d,1), d = .45 (2)
print(stats.table.s45,include.rownmames=True)

Sum.stats.s45 <- c(summary(Geweke.Porter.Hudaks45),summary(Risens45),summary(Max.Likhood
Sum.stats.s45 <- matrix(Sum.stats.s45,nrow=6)
Sum.stats.s45 <- data.frame(Sum.stats.s45,row.names=c("Minimum","1rst Quartile","Median"
names(Sum.stats.s45) <- c("Geweke.Porter.Hudak","Reisen","Max.Likhood")
library(xtable)
Sum.stats.table.s45 <- xtable(Sum.stats.s45,caption="Statistics for ARFIMA(0,d,1), d = .
print(Sum.stats.table.s45,include.rownmames=True)

# Create statistics for ARFIMA(0,d,1), d=.25, sample of 50

Geweke.Porter.Hudakss25 <- Risenss25 <- Max.Likhoodss25 <- 0
load("ARFIMA.0d1.25.50")
data.s25.50 <- ARFIMA.0d1.25.50


for ( i in 1:100){
{
gph <- fdGPH(data.s25.50[,i])
Geweke.Porter.Hudakss25[i] = gph$d

}


{
risen <- fdSperio(data.s25.50[,i])
Risenss25[i] = risen$d

}

{
mle <- fracdiff(data.s25.50[,i], nar = 0 , nma = 0)
```

```
Max.Likhoodss25[i] = mle$d
}

}

data.d.s25.50 <- data.frame(Geweke.Porter.Hudakss25,Risenss25,Max.Likhoodss25) # estimat

save(data.d.s25.50, file=("data.d.s25.50"))
save(Geweke.Porter.Hudakss25, file=("Geweke.Porter.Hudakss25"))
save(Risenss25, file=("Risenss25"))
save(Max.Likhoodss25, file=("Max.Likhoodss25"))

means.s25.50 <- c(mean(Geweke.Porter.Hudakss25),mean(Risenss25),mean(Max.Likhoodss25))
sds.s25.50 <- c(sd(Geweke.Porter.Hudakss25),sd(Risenss25),sd(Max.Likhoodss25))
mins.s25.50 <- c(min(Geweke.Porter.Hudakss25),min(Risenss25),min(Max.Likhoodss25))
maxs.s25.50 <- c(max(Geweke.Porter.Hudakss25),max(Risenss25),max(Max.Likhoodss25))
stats.s25.50 <- data.frame(means.s25.50,sds.s25.50,mins.s25.50,maxs.s25.50,row.names=c("
names(stats.s25.50) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table.s25.50 <- xtable(stats.s25.50,caption=" Statistics for samples of 50,  ARFIM
print(stats.table.s25.50,include.rownmames=True)

Sum.stats.s25.50 <- c(summary(Geweke.Porter.Hudakss25),summary(Risenss25),summary(Max.Li
Sum.stats.s25.50 <- matrix(Sum.stats.s25.50,nrow=6)
Sum.stats.s25.50 <- data.frame(Sum.stats.s25.50,row.names=c("Minimum","1rst Quartile","M
names(Sum.stats.s25.50) <- c("Geweke.Porter.Hudak","Reisen","Max.Likhood")
library(xtable)
Sum.stats.table.s25.50 <- xtable(Sum.stats.s25.50,caption="Statistics for samples of 50,
print(Sum.stats.table.s25.50,include.rownmames=True)

# Create statistics for ARFIMA(0,d,1), d=.45, sample of 50

Geweke.Porter.Hudakss45 <- Risenss45 <- Max.Likhoodss45 <- 0
load("ARFIMA.0d1.45.50")
data.s45.50 <- ARFIMA.0d1.45.50


for ( i in 1:100){
{
gph <- fdGPH(data.s45.50[,i])
Geweke.Porter.Hudakss45[i] = gph$d
```

```
}


{
risen <- fdSperio(data.s45.50[,i])
Risenss45[i] = risen$d

}

{
mle <- fracdiff(data.s45.50[,i], nar = 0 , nma = 0)
Max.Likhoodss45[i] = mle$d
}

}

data.d.s45.50 <- data.frame(Geweke.Porter.Hudakss45,Risenss45,Max.Likhoodss45) # estimat

save(data.d.s45.50, file=("data.d.s45.50"))
save(Geweke.Porter.Hudakss45, file=("Geweke.Porter.Hudakss45"))
save(Risenss45, file=("Risenss45"))
save(Max.Likhoodss45, file=("Max.Likhoodss45"))

means.s45.50 <- c(mean(Geweke.Porter.Hudakss45),mean(Risenss45),mean(Max.Likhoodss45))
sds.s45.50 <- c(sd(Geweke.Porter.Hudakss45),sd(Risenss45),sd(Max.Likhoodss45))
mins.s45.50 <- c(min(Geweke.Porter.Hudakss45),min(Risenss45),min(Max.Likhoodss45))
maxs.s45.50 <- c(max(Geweke.Porter.Hudakss45),max(Risenss45),max(Max.Likhoodss45))
stats.s45.50 <- data.frame(means.s45.50,sds.s45.50,mins.s45.50,maxs.s45.50,row.names=c("
names(stats.s45.50) <- c("Mean","SD","Min","Max")
library(xtable)
stats.table.s45.50 <- xtable(stats.s45.50,caption=" Statistics for samples of 50,  ARFIM
print(stats.table.s45.50,include.rownmames=True)

Sum.stats.s45.50 <- c(summary(Geweke.Porter.Hudakss45),summary(Risenss45),summary(Max.Li
Sum.stats.s45.50 <- matrix(Sum.stats.s45.50,nrow=6)
Sum.stats.s45.50 <- data.frame(Sum.stats.s45.50,row.names=c("Minimum","1rst Quartile","M
names(Sum.stats.s45.50) <- c("Geweke.Porter.Hudak","Reisen","Max.Likhood")
library(xtable)
Sum.stats.table.s45.50 <- xtable(Sum.stats.s45.50,caption="Statistics for samples of 50,
print(Sum.stats.table.s45.50,include.rownmames=True)

# Create statistics for ARFIMA(0,d,1), d=.25, multiple \theta
```

```
Geweke.Porter.Hudakxy25 <- Risenxy25 <- Max.Likhoodxy25 <- 0
load("ARFIMA.0d1.25.xy")
data.s25.xy <- ARFIMA.0d1.25.xy


for ( i in 1:6){
{
gph <- fdGPH(data.s25.xy[,i])
Geweke.Porter.Hudakxy25[i] = gph$d

}


{
risen <- fdSperio(data.s25.xy[,i])
Risenxy25[i] = risen$d

}

{
mle <- fracdiff(data.s25.xy[,i], nar = 0 , nma = 0)
Max.Likhoodxy25[i] = mle$d
}

}

data.d.s25.xy <- data.frame(Geweke.Porter.Hudakxy25,Risenxy25,Max.Likhoodxy25,row.names=
names(data.d.s25.xy) <- c("Geweke Porter-Hudak","Reisen","Max.Likhood")

library(xtable)
data.d.s25.xy <- xtable(data.d.s25.xy, caption="Comparing Model For Best Theta,  ARFIMA(
print(data.d.s25.xy,include.rownmames=True)

# then for d=.45

load("ARFIMA.0d1.45.xy")
setwd("C:/Users/Owner/Documents/Thesis/data/Datacreation")
Geweke.Porter.Hudakxy45 <- Risenxy45 <- Max.Likhoodxy45 <- 0
data.s45.xy <- ARFIMA.0d1.45.xy
```

```
for ( i in 1:6){
{
gph <- fdGPH(data.s45.xy[,i])
Geweke.Porter.Hudakxy45[i] = gph$d

}


{
risen <- fdSperio(data.s45.xy[,i])
Risenxy45[i] = risen$d

}

{
mle <- fracdiff(data.s45.xy[,i], nar = 0 , nma = 0)
Max.Likhoodxy45[i] = mle$d
}

}

data.d.s45.xy <- data.frame(Geweke.Porter.Hudakxy45,Risenxy45,Max.Likhoodxy45,row.names=
names(data.d.s45.xy) <- c("Geweke Porter-Hudak","Reisen","Max.Likhood")
library(xtable)
data.d.s45.xy <- xtable(data.d.s45.xy, caption="Comparing Model For Best Theta,  ARFIMA(
print(data.d.s45.xy,include.rownmames=True)


## Graphics Creation

# Long memory time series

setwd("C:/Users/Owner/Documents/Thesis/data") # where data is located
load("X25.c")
setwd("C:/Users/Owner/Documents/Thesis/data/DataX") # where data is located
load("X100")
setwd("C:/Users/Owner/Documents/Thesis/data/Datacreation") # # where Sweave folder is lo
par(mfrow=c(2,2))
plot(X100, main=" Long Memory Time Series Plot, d=.25", xlab=" ",ylab=" ", font.main=6)
plot(X25.c, main=" Long Memory Time Series Plot, d=.45", xlab=" ",ylab=" ",font.main=6)
acf(X100,100,main=" ",xlab="Long Memory Time Series ACF Plot, d=.25 ")
acf(X25.c,100,main=" ",xlab="Long Memory Time Series ACF Plot, d=.45 ")
```

```
par(mfrow=c(1,1))


# compare Estimators Geweke & Porter-Hudak, Reisen, and Maximum likelihood for d=.45/.25

setwd("C:/Users/Owner/Documents/Thesis/data/Dataset")
load("Geweke.Porter.Hudak")
load("Risen")
load("Max.Likhood")
load("Geweke.Porter.Hudak.c")
load("Risen.c")
load("Max.Likhood.c")
setwd("C:/Users/Owner/Documents/Thesis/data/Datacreation")
par(mfrow=c(3,2))
plot(Geweke.Porter.Hudak,main="Geweke Porter-Hudak, d = .25", xlab=" ",ylab=" ")
abline(h=.25)
plot(Geweke.Porter.Hudak.c,main="Geweke Porter-Hudak, d = .45", xlab=" ",ylab=" ")
abline(h=.45)
plot(Risen,main="Reisen, d = .25", xlab=" ",ylab=" ")
abline(h=.25)
plot(Risen.c,main="Reisen, d = .45", xlab=" ",ylab=" ")
abline(h=.45)
plot(Max.Likhood,main="Maximum Likelihood, d = .25", xlab=" ",ylab=" ")
abline(h=.25)
plot(Max.Likhood.c,main="Maximum Likelihood, d = .45", xlab=" ",ylab=" ")
abline(h=.45)
par(mfrow=c(1,1))


# compare Estimators Geweke & Porter-Hudak, Reisen, and Maximum likelihood for d=.45/.25

setwd("C:/Users/Owner/Documents/Thesis/data/Dataset")
load("Geweke.Porter.Hudak50")
load("Risen50")
load("Max.Likhood50")
load("Geweke.Porter.Hudak50c")
load("Risen50c")
load("Max.Likhood50c")
setwd("C:/Users/Owner/Documents/Thesis/data/Datacreation")
par(mfrow=c(3,2))
plot(Geweke.Porter.Hudak50,main="Sample of 50, Geweke Porter-Hudak, d = .25", xlab=" ",y
abline(h=.25)
```

```
plot(Geweke.Porter.Hudak50c,main="Sample of 50, Geweke Porter-Hudak, d = .45", xlab=" ",
abline(h=.45)
plot(Risen50,main="Sample of 50, Reisen, d = .25", xlab=" ",ylab=" ")
abline(h=.25)
plot(Risen50c,main="Sample of 50, Reisen, d = .45", xlab=" ",ylab=" ")
abline(h=.45)
plot(Max.Likhood50,main="Sample of 50, Maximum Likelihood, d = .25", xlab=" ",ylab=" ")
abline(h=.25)
plot(Max.Likhood50c,main="Maximum Likelihood, d = .45", xlab=" ",ylab=" ")
abline(h=.45)
par(mfrow=c(1,1))


# compare Estimators Geweke & Porter-Hudak, Reisen, and Maximum likelihood for ARFIMA(0,

setwd("C:/Users/Owner/Documents/Thesis/data/Dataset")
load("Geweke.Porter.Hudaks25")
load("Risens25")
load("Max.Likhoods25")
load("Geweke.Porter.Hudaks45")
load("Risens45")
load("Max.Likhoods45")
setwd("C:/Users/Owner/Documents/Thesis/data/Datacreation")
par(mfrow=c(3,2))
plot(Geweke.Porter.Hudaks25, main="ARFIMA(0,d,1), Geweke Porter-Hudak, d =.25", xlab=" "
abline(h=.25)
plot(Geweke.Porter.Hudaks45,main="ARFIMA(0,d,1), Geweke Porter-Hudak, d =.45", xlab=" ",
abline(h=.45)
plot(Risens25,main="ARFIMA(0,d,1), Reisen, d =.25", xlab=" ",ylab=" ")
abline(h=.25)
plot(Risens45,main="ARFIMA(0,d,1), Reisen, d =.45", xlab=" ",ylab=" ")
abline(h=.45)
plot(Max.Likhoods25,main="ARFIMA(0,d,1), Maximum Likelihood, d =.25", xlab=" ",ylab=" ")
abline(h=.25)
plot(Max.Likhoods45,main="ARFIMA(0,d,1),Maximum Likelihood, d =.45", xlab=" ",ylab=" ")
abline(h=.45)
par(mfrow=c(1,1))
```