ARTFIMA Simulation Experiments

A. I. McLeod, Mark M. Meerschaert, Farzad Sabzikar

A. I. McLeod, Mark M. Meerschaert, Farzad Sabzikar (2016). ARTFIMA model fitting. R package version 1.2. URL http://CRAN.R-project.org/package=artfima

Purpose and summary

The main purpose of the simulations is to demonstrate the correctness of the analytical result obtained for the asymtotic covariance matrix of the parameters. Our simulations show that standard errors predicted by this formula agree well empirical results.

Secondly, the observed vs expected information estimates for the standard errors of the parameters are compared empirically. It was found there is little difference in most cases but in some cases the observed information estimates are more accurate as compared with the empirical estimates.

Thirdly, the reliability of the software and small sample properties of the estimates are also investigated.

The R scripts for producing the simulations and for their analysis including all plots shown below are included in the Appendix.

This report and the associated R scripts are included in the R artfima package and may be accessed by loading this package and typing the following command at the R prompt,

```
> help(package="artfima")
```

Introduction

In the simulation experiments we consider the special case of the ARTFIMA model

$\left(1 - e^{-\lambda} B\right)^{d} Y_t = a_t$

The large-sample information matrix per observation defined by,

$$\mathcal{I}_\beta = \lim_{n \to \infty} \frac{1}{n} \left(-\mathcal{E} \left\{ \frac{\partial^2 \mathcal{L}}{\partial \beta_i \partial \beta_j} \right\} \right) \qquad (1)$$

was derived in our paper (eqns 43, 48, 54). The computation for this result has been implemented in *Mathematica* , symbolically and numerically as well in our R package artfima Version 1.2.

R functions

```
> informationMatrixARTFIMA
function(lambda, d){
  v11 <- 4*pi*exp(-2*lambda)*(1+ 0.5*exp(-2*lambda)) #eqn (43)
  v22 <- (d^2)*exp(-2*lambda)/(1-exp(-2*lambda)) #eqn (48)
  v12 <- d*log(1-exp(-2*lambda)) #eqn (54)
  matrix(c(v11,v12,v12,v22), ncol=2,
         dimnames=list(c("lambda","d"),c("lambda","d")))
}
> seARTFIMA
function(lambda, d, n){
  v <- solve(informationMatrixARTFIMA(lambda, d)*n)
  sqrt(diag(v))
}
```

Mathematica function and some symbolic computations

```
InformationMatrixARTFIMA[λ_, d_] := Module[{v11, v22, v12},
   v11 = 4 π Exp[-2 λ] (1 + Exp[-2 λ] / 2);
   v22  =  d² Exp[-2 λ] / (1 - Exp[-2 λ]);
   v12 = d Log[1 - Exp[-2 λ]];
   {{v11, v12}, {v12, v22}}]

d =.; λ =.; (*in case we used them before*)
Style[InformationMatrixARTFIMA[λ, d] // TraditionalForm // MatrixForm, 20]
```

$$\begin{pmatrix} 4\, e^{-2d} \left(1 + \frac{e^{-2d}}{2}\right) \pi & \lambda \log\left(1 - e^{-2d}\right) \\ \lambda \log\left(1 - e^{-2d}\right) & \frac{e^{-2d}\, \lambda^2}{1 - e^{-2d}} \end{pmatrix}$$

The large sample variance matrix for $\sqrt{n}\left(\left(\hat{\lambda} - \lambda\right), \left(\hat{d} - d\right)\right)$ is complicated,

```
Style[FullSimplify[Inverse[InformationMatrixARTFIMA[λ, d]]] // TraditionalForm //
   MatrixForm, 14]
```

$$\begin{pmatrix} -\frac{e^{4d}}{e^{4d}\left(-1+e^{2d}\right)\log^2\left(1-e^{-2d}\right)-2\left(1+2\,e^{2d}\right)\pi} & -\frac{e^{4d}\left(-1+e^{2d}\right)\log\left(1-e^{-2d}\right)}{\lambda\left(\left(e^{4d}-e^{6d}\right)\log^2\left(1-e^{-2d}\right)+2\left(1+2\,e^{2d}\right)\pi\right)} \\ -\frac{e^{4d}\left(-1+e^{2d}\right)\log\left(1-e^{-2d}\right)}{\lambda\left(\left(e^{4d}-e^{6d}\right)\log^2\left(1-e^{-2d}\right)+2\left(1+2\,e^{2d}\right)\pi\right)} & -\frac{2\left(1+e^{2d}-2\,e^{4d}\right)\pi}{\lambda^2\left(\left(e^{4d}-e^{6d}\right)\log^2\left(1-e^{-2d}\right)+2\left(1+2\,e^{2d}\right)\pi\right)} \end{pmatrix}$$

The following asymptotic formula were obtained for $\sigma_{\lambda}^2 = \sqrt{n}\left(\hat{\lambda} - \lambda\right)$ and $\sigma_{d}^2 = \sqrt{n}\left(\hat{d} - d\right)$,

$$\sigma_{\lambda}^2 = -\frac{e^{4\lambda}}{e^{4\lambda}\left(e^{2\lambda} - 1\right)\log^2\left(1 - e^{-2\lambda}\right) - 2\pi\left(2\,e^{2\lambda} + 1\right)} \tag{2}$$

$$\sigma_{d}^2 = -\frac{2\pi\left(e^{2\lambda} - 2\,e^{4\lambda} + 1\right)}{\mathrm{d}^2\left(2\pi\left(2\,e^{2\lambda} + 1\right) + \left(e^{4\lambda} - e^{6\lambda}\right)\log^2\left(1 - e^{-2\lambda}\right)\right)} \tag{3}$$

Hessian matrix and standard errors estimated in the artfima() function

The negative of the Hessian matrix of the log-likelihood function evaluated at the MLE is defined as the observed Fisher information matrix,

$$\hat{\mathcal{I}}_{\beta} = \left(-\frac{\partial^2 \mathcal{L}}{\partial\, \hat{\beta}_i\, \partial\, \hat{\beta}_j}\right) \tag{4}$$

where $\mathcal{L}$ is the log-likelihood function. When Whittle estimation is used, the log-likelihood function may be written,

$$\mathcal{L} = -\frac{n}{2}\log\sigma_a^2 - \frac{S}{2\,\sigma_a^2} \tag{5}$$

where S is the objective function that is minimized,

$$S = \sum_{j=1}^{m} \frac{I_j}{p_j} \tag{6}$$

and $p_j = \frac{1}{2\pi}\left|\psi\left(e^{-i\,2\pi j/n}\right)\right|^2$.

so the observed Fisher estimation can be written,

$$\hat{\mathcal{I}}_\beta = \frac{1}{2\,\hat{\sigma}_a^2}\,\frac{\partial^2 S}{\partial\,\hat{\beta}_i\,\partial\,\hat{\beta}_j} \tag{7}$$

Eqn (7) is used in our implementation in the artfima package. The standard error estimates are the square root of diagonal elements in $\hat{\mathcal{I}}_\beta^{-1}$.

Simulation Experiments

Design

The ARTFIMA model was simulated in R and fit using our artfima() function. For each parameter setting 10^4 simulations were done with $\lambda = 0.01,\ 0.02,\ 0.03$, $d = 0.4,\ 0.8,\ 1.2$, and series length $n = 200,\ 500,\ 1000,\ 10\,000$. The simulations were done on an i7 PC using the R parallel package and took about 6 hours. The R script used for the simulations is given in Appendix below.

For a particular parameter setting let $\hat{\lambda}_i$ and $\hat{d}_i$, $i = 1, \ldots, 10^4$ be the Whittle MLE estimates.

The standard errors of these MLE estimates were obtained using the Hessian matrix and are denoted by $\hat{\sigma}_{\hat{\lambda},i}$ and $\hat{\sigma}_{\hat{d},i}$, $i = 1, \ldots, 10^4$ respectively and referred to as the **observed standard errors**.

The **empirical standard deviations** obtained directly from the samples of $\hat{\lambda}_i$ and $\hat{d}_i$ are the “gold” standard for the true correct finite-sample standard deviation for the MLEs. Apart from a small sampling error, because 10^4 simulations were used, these are exact small sample estimates. The accuracy of these estimates was assessed by the bootstrap and because the 10^4 simulations were done the accuracy, as measured is the 95% margin-of-error (MOE) is very low, less that the width of the plotting symbols used in the graphs. The MOE is given in tables presented in the Appendix.

Both the observed standard errors and the empirical standard deviations were used to assess asymptotic formula given in eqns. (2) and (3).

Main results

Distribution of the estimates

It was found that, especially when n is not too large, the empirical distributions of $\hat{\lambda}$ and $\hat{d}$ were strongly influenced by outliers and was strongly skewed to the right. Boxplots of these estimates are shown in the Appendix.

Due to outliers, the median was used to estimate the asymptotic and observed standard errors for each parameter setting. The empirical standard deviations were estimated using the median absolute deviation scaled so as to estimate the standard deviation in a normal distribution. The R function mad() was used.

Dotcharts of standard error estimates

The multi-panel dotchart display shown below compares the observed Hessian derived estimates generated by the arffima() function with the other standard error estimates. The observed Hessian estimates are derived by taking the median of the 10^4 samples at each parameter setting are shown by the green square, 🟩. The empirical estimates are robust estimates of the standard deviations of the MLE's produced by artfima() and shown as the stars, ✶. The black dots, ●, show the asymptotic standard error estimates as defined by eqns. (2) and (3).

In each panel, the vertical axis corresponds to parameter settings $n = 200$, 500, 1000, 5000 and the horizontal axis to the standard error. The label at the top of the panel indicates the other parameter settings for λ and d. The corresponding exact numerical values are tabulated in the next section.

The dotcharts below show that for both the asymptotic and observed information matrix approaches the standard error is underestimated in all cases but the accuracy improves as the series length n increases. The observed and asymptotic estimates of the standard error for $\hat{\lambda}$ generally agree quite closely except when in the case $n = 200$, $d = 0.4$. For $\hat{d}$, the observed and asymptotic standard errors converge more slowly to the true value but there is good agreement when $n = 5000$.

The multi-panel dotchart displays shown below are for the estimates for the observed and asymptotic SE compared with the SD estimated empirically. In each panel, the vertical axis corresponds to parameter settings $\lambda = 0.01$, 0.02 and 0.03 and the horizontal axis to the three standard error estimates. The label at the top of the panel indicates the other parameter settings for d and n. The corresponding exact numerical values are tabulated in the next section.

Figure 1. Dotchart asymptotic se compared with empirical and observed $\hat{\lambda}$

The horizontal axes are all on the same scale for the three different estimates for $\sigma(\hat{\lambda})$. As *n* increases, the estimates converge to the asymptotic value and verifies the correctness and usefulness of these asymptotic results.

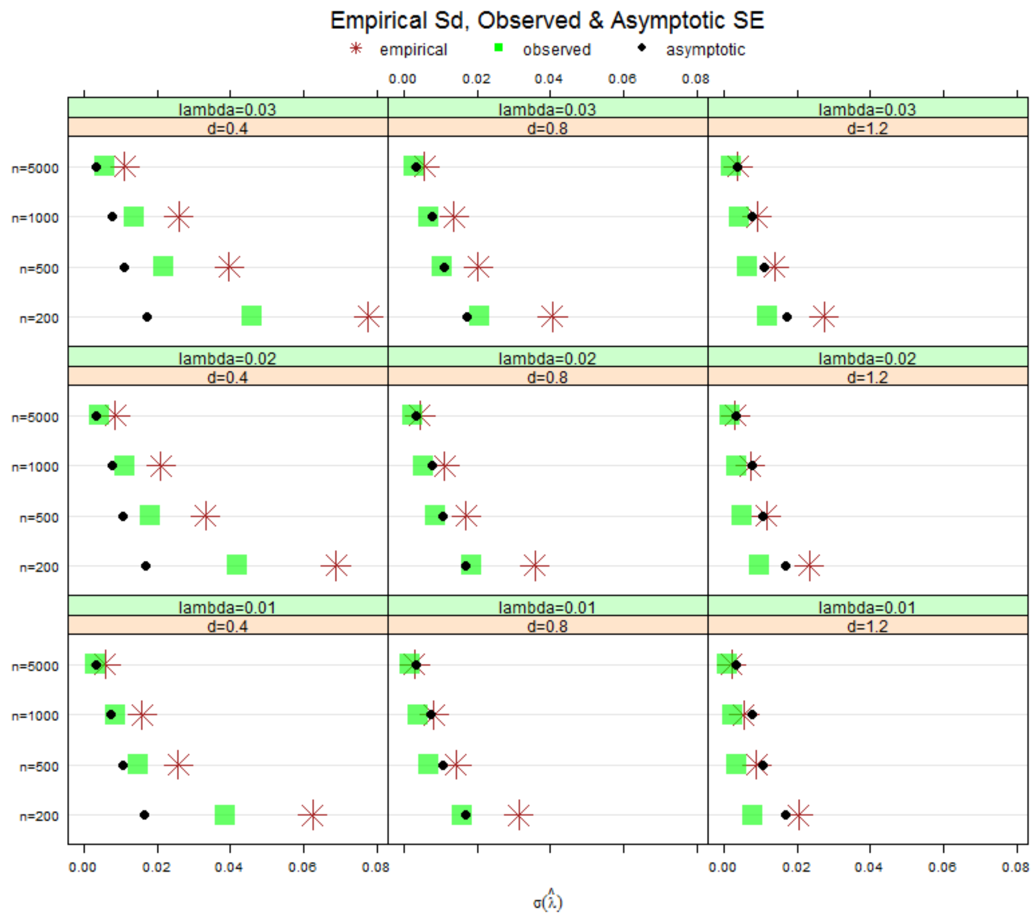
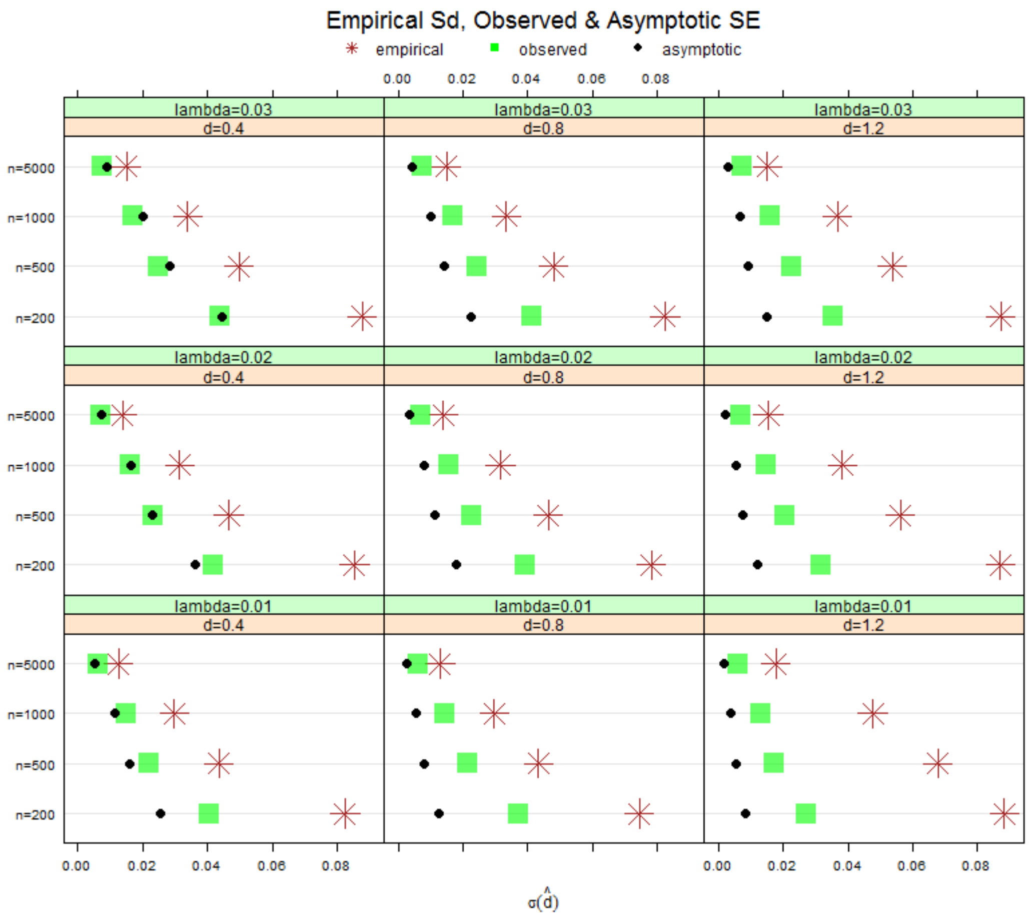


Figure 2. Dotchart asymptotic se compared with empirical and observed $\hat{d}$

The horizontal axes are all on the same scale for the three different estimates for $\sigma(\hat{d})$. As n increases, the estimates get closer together. This suggests the asymptotic formula are correct and provide a valid approximation.



Appendices

Convergence issues

In the artfima() function, the default optimization algorithm is “L-BFGS-B”. This is the most reliable of the nonlinear optimizers for our purpose and was used in the simulations. Lower and upper bounds were placed on the parameters λ and d so $\lambda \in [0.0001,\ 10]$ and $d \in (-10, 10)$.

There were 52 cases where optim() reported full convergence was not obtained but the estimates appeared to be accurate enough for our purposes.

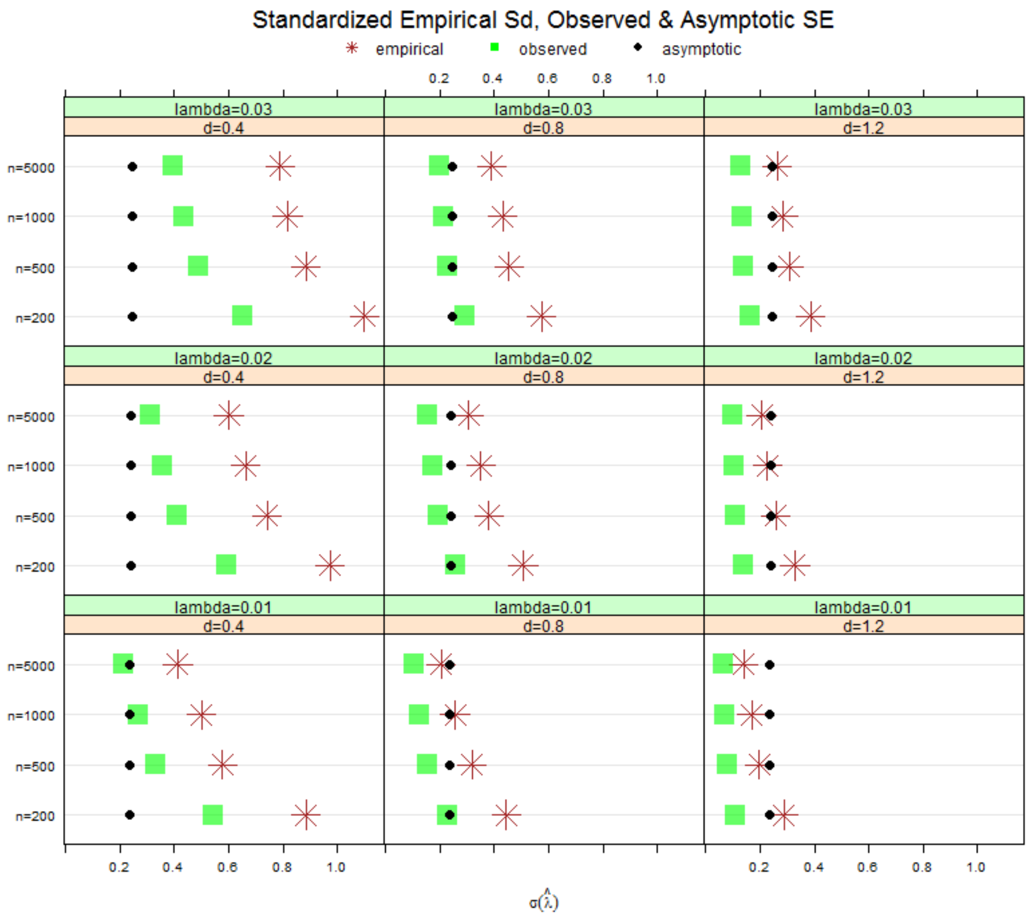
When $d = 0.4$, there were 85 out of 10^4 cases where convergence was on the boundary producing $\hat{d} = 10$. These cases were about evenly spread over the possible λ settings (20, $\lambda = 0.01$, 34, $\lambda = 0.02$, 31, $\lambda = 0.03$).

Dotcharts of standardized standard error estimates

Instead of the raw standard errors, we may standardize them by multiplying by $\sqrt{n}$. As $n \to \infty$, the observed and empirical should converge to the asymptotic.

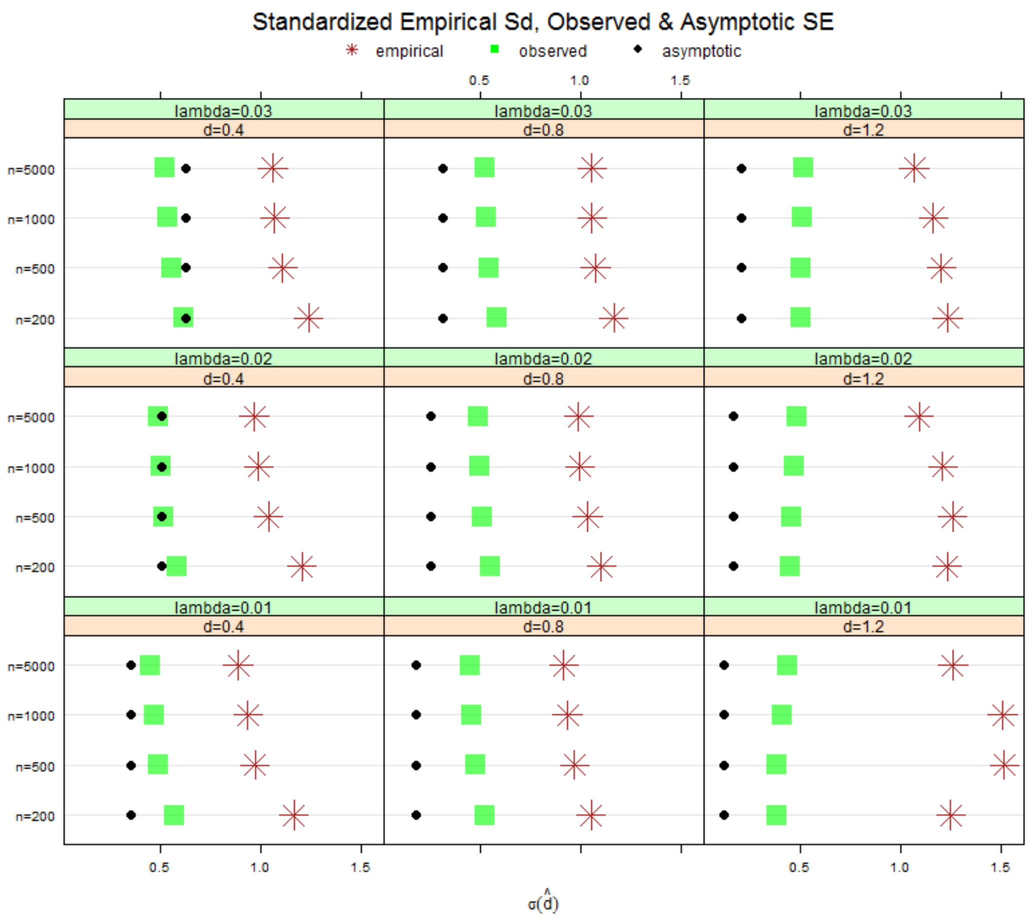
For the se of λ, the accuracy of the asymptotic formula is seen to improve in all cases as n increases. When $d = 1.2$, right column, the empirical and asymptotic agree closely but as we move left in the columns corresponding to $d = 0.8$ and $d = 0.4$ the agreement is less but the asymptotic and observed are in better agreement.

Figure 3. Dotchart standardized asymptotic se compared with empirical and observed $\hat{\lambda}$



For *d* the se improves slightly as *n* increases. The case with $d = 1.2$ and $\lambda = 0.01$ seems odd. This could be due perhaps to some problems with the optimization algorithm. However there is reasonably good agreement between the asymptotic result and the estimate obtained numerically from the Hessian matrix shown by the green square, 🟩.

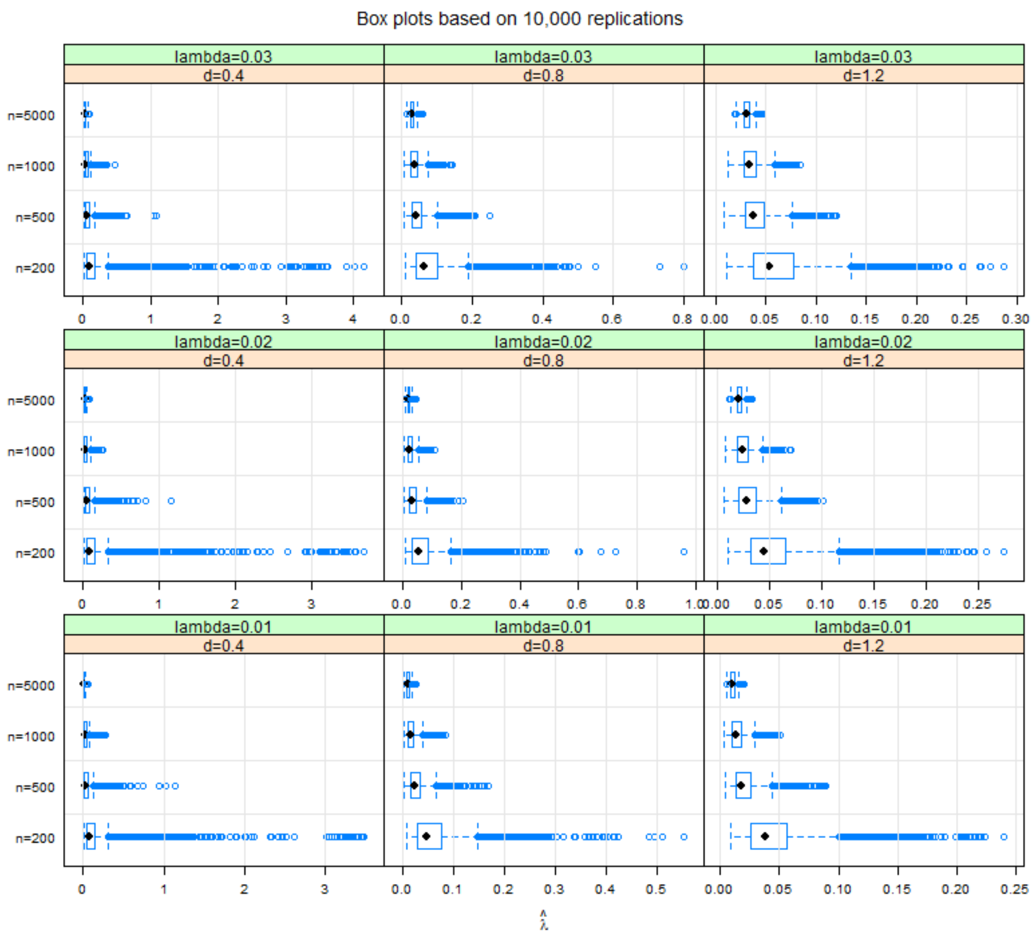
Figure 4. Dotchart standardized asymptotic se compared with empirical and observed $\hat{d}$



Boxplots of the distributions of $\hat{\lambda}$ and $\hat{d}$

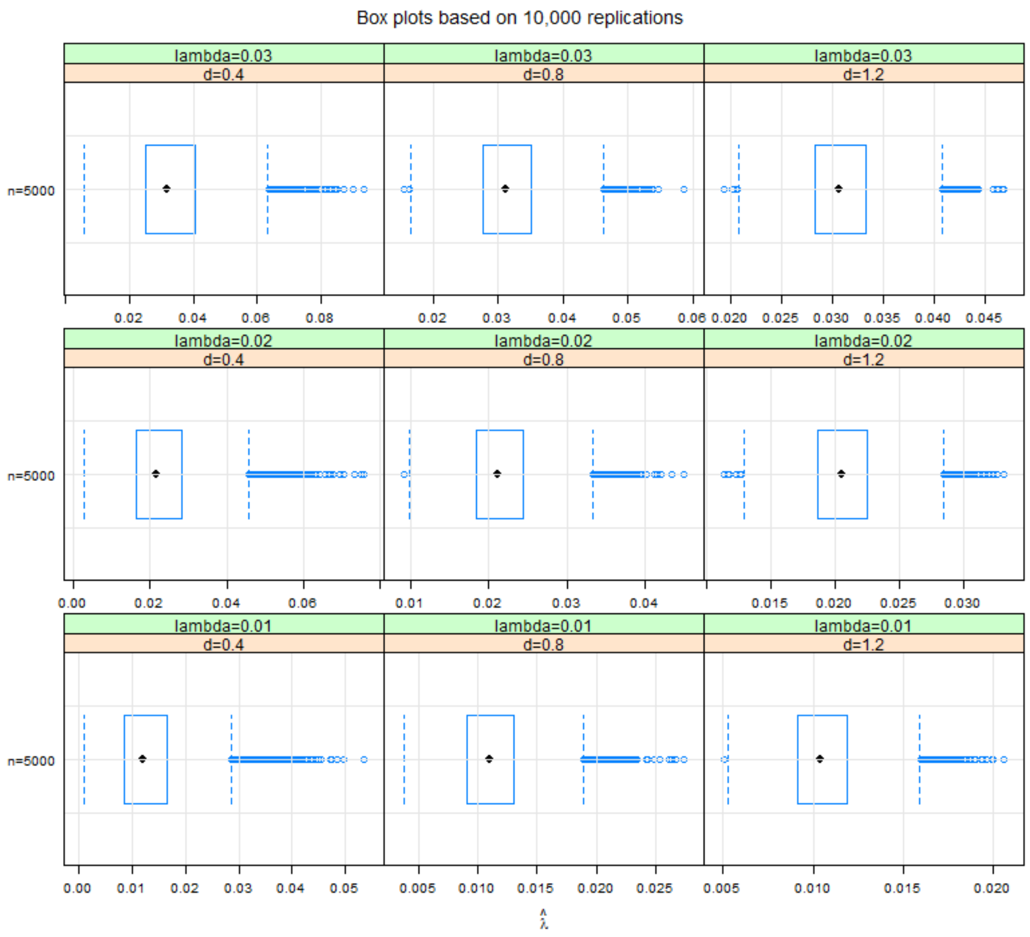
The distribution of $\hat{\lambda}$ is summarized below. The distribution is skewed to the right especially when $n = 200$ and $d = 0.4$ (see left column of panels with $d = 0.4$, note the horizontal scale goes from 0 to 3+ whereas on the other panels the scale is much reduced. Each boxplot summarizies the distribution over 10,000 samples.

Figure 5. Boxplots $\hat{\lambda}$



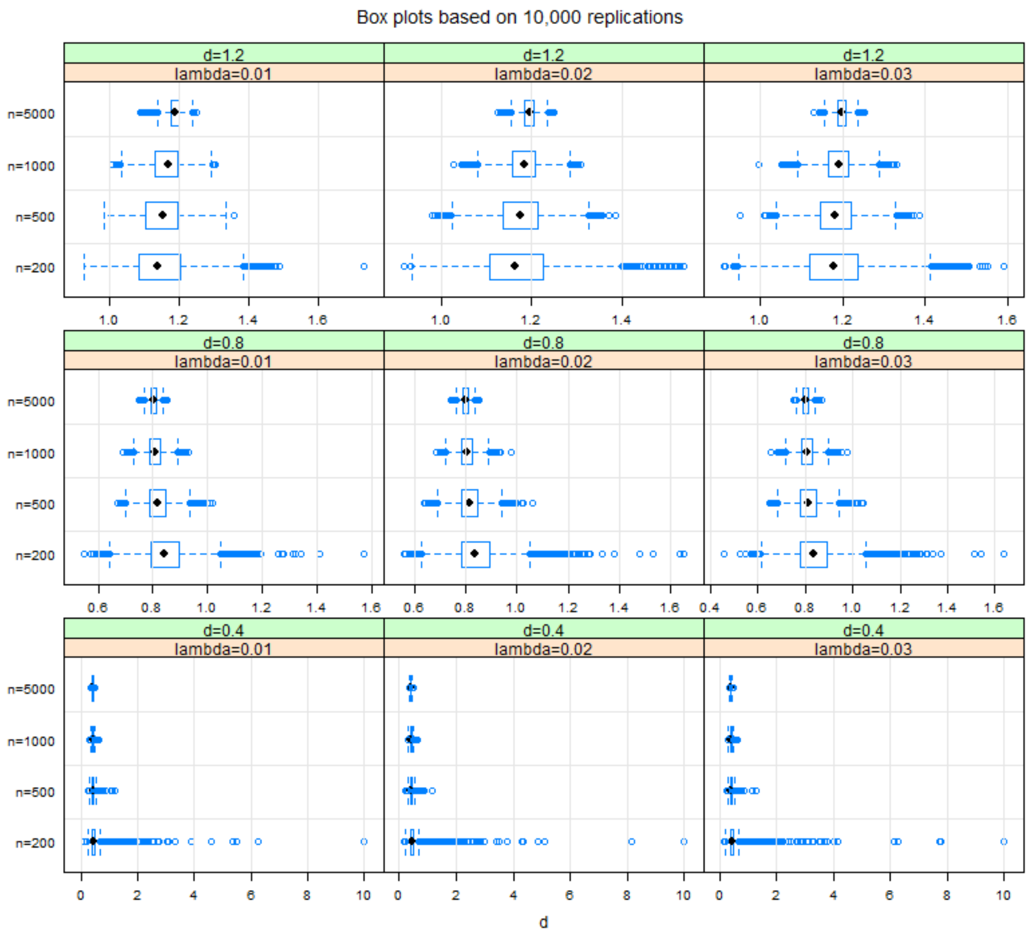
Drilling down with series length $n = 5000$, we see the distribution of $\hat{\lambda}$ is still skewed.

Figure 6. Boxplots $\hat{\lambda}$, $n = 5000$



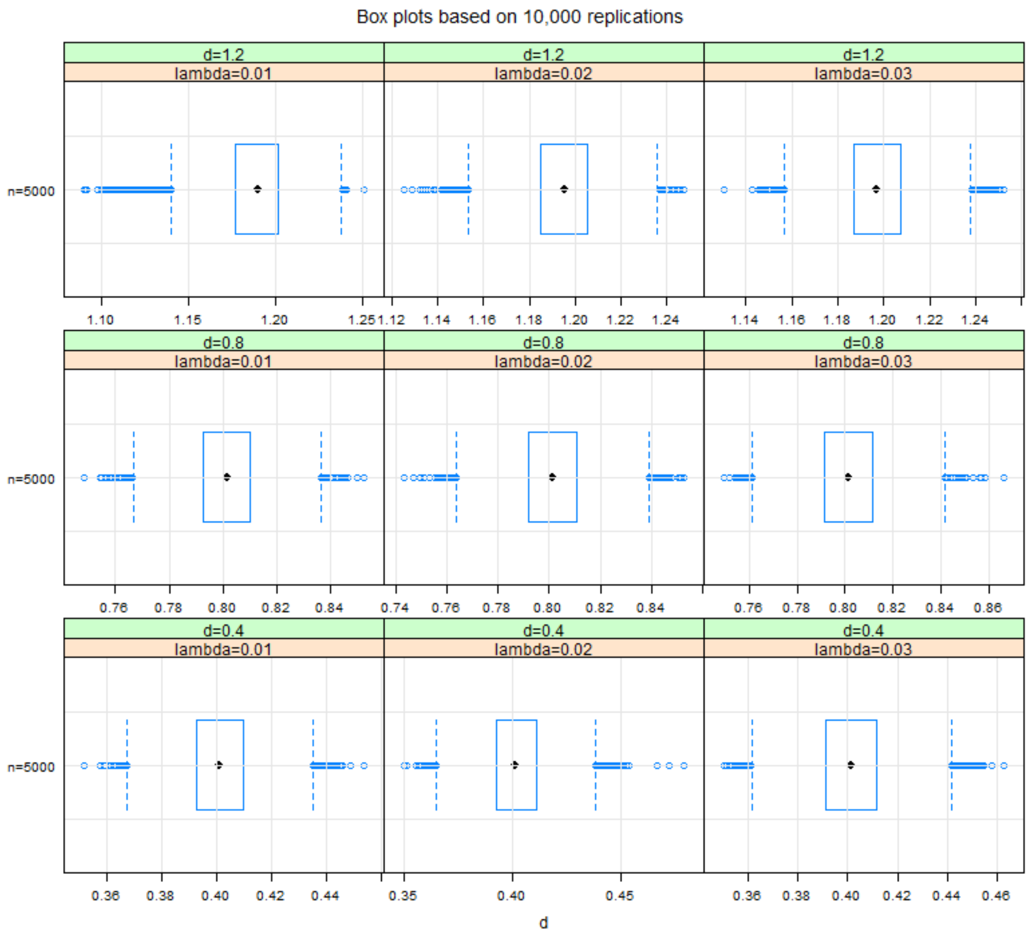
The distribution of $\hat{d}$ is summarized below. The distribution is skewed to the right but symmetry improves as n increases. Note the boundary cases where $d = 10$ and $\lambda = 0.4$.

Figure 7. Boxplots $\hat{d}$



Focusing on $n = 5000$, symmetry is much improved in most cases the most noteable exception being when $d = 1.2$, $\lambda = 0.01$.

Figure 8. Boxplots $\hat{d}$, $n = 5000$



Observed and Expected Information Matrix

The negative of the Hessian matrix of the log-likelihood function evaluated at the MLE is defined as the observed Fisher information matrix,

$$\hat{\mathcal{I}}_\beta = \left(- \frac{\partial^2 \mathcal{L}}{\partial \hat{\beta}_i \partial \hat{\beta}_j} \right) \tag{8}$$

where $\mathcal{L}$ is the log-likelihood function. The standard errors are the square root of the diagonal elements of $\hat{\mathcal{I}}_\beta^{-1}$.

The theoretical or expected large sample information matrix per observation may be written,

$$\mathcal{I}_\beta = \lim_{n \to \infty} \in \{ \mathcal{I}_\beta / n \} \tag{9}$$

An explicit expression for this is given in our paper and provide another method of estimating the standard error of the estimates. Evaluating $\mathcal{I}_\beta$ at $\beta = \hat{\beta}$ and denoting this matrix by $\mathcal{I}_{\hat{\beta}}$ then the estimated standard errors for the parameters are given by the square root of the diagonal elements in $(n\ \mathcal{I}_{\hat{\beta}})^{-1}$.

Given the MLE, the standard errors for the estimates may be estimated using either the expected information matrix or the observed information matrix. Usually when non-linear optimization is used, as in the R function optim() the Hessian matrix is available so it can be used.

A statistical inference argument, based on the principle of conditionality, is presented in a famous paper by Efron & Hinkley (Biometrika, 1978) suggesting that the observed information is preferable for the purpose of estimating the standard errors and this is endorsed as a general principle by Cox (2006 §6.6, *Principles of Statistical Inference.* Cambridge University Press).

The bootstrap or jackknife computationally intensive methods that may also be used to obtain estimates of the standard errors as well.

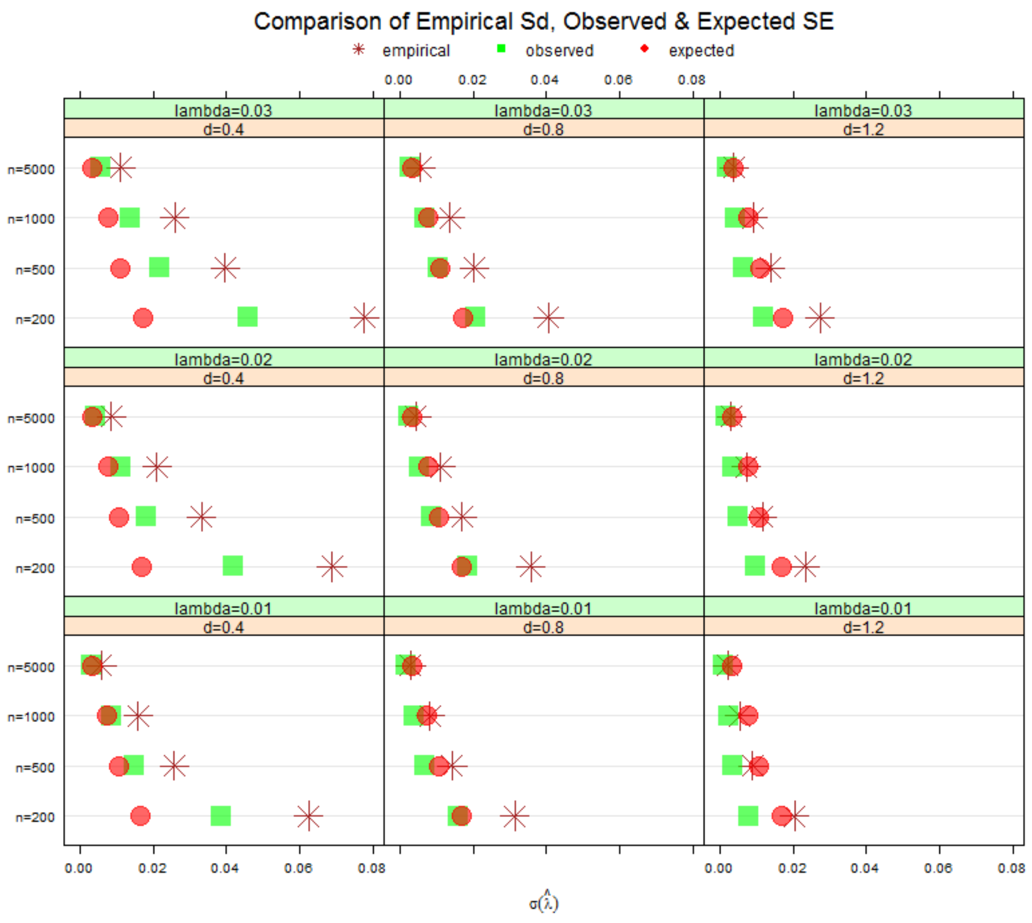
Now we investigate the question of which estimate to use for our standard error in the artfima: the observed or expected information estimate? The estimates are compared to the gold standard, the empirical estimates. In general there is little difference in accuracy between the observed vs expected estimates but when there is a noticeable difference, the observed se performs better than the expected se estimate.

In the following multi-panel dotcharts, the standard error estimates obtained by the observed or Hessian matrix, denoted by, the green square, ■, are compared with estimates of standard error using the expected information, red round disks, ●. Both of these estimates are based on the medians of 10^4 estimates. The empirical estimates, brown stars, ✶, are obtained using the median absolute deviation estimator for the standard deviation and ideally estimate the true finite sample standard deviations.

Dotchart observed vs expected se's compared for $\hat{\lambda}$

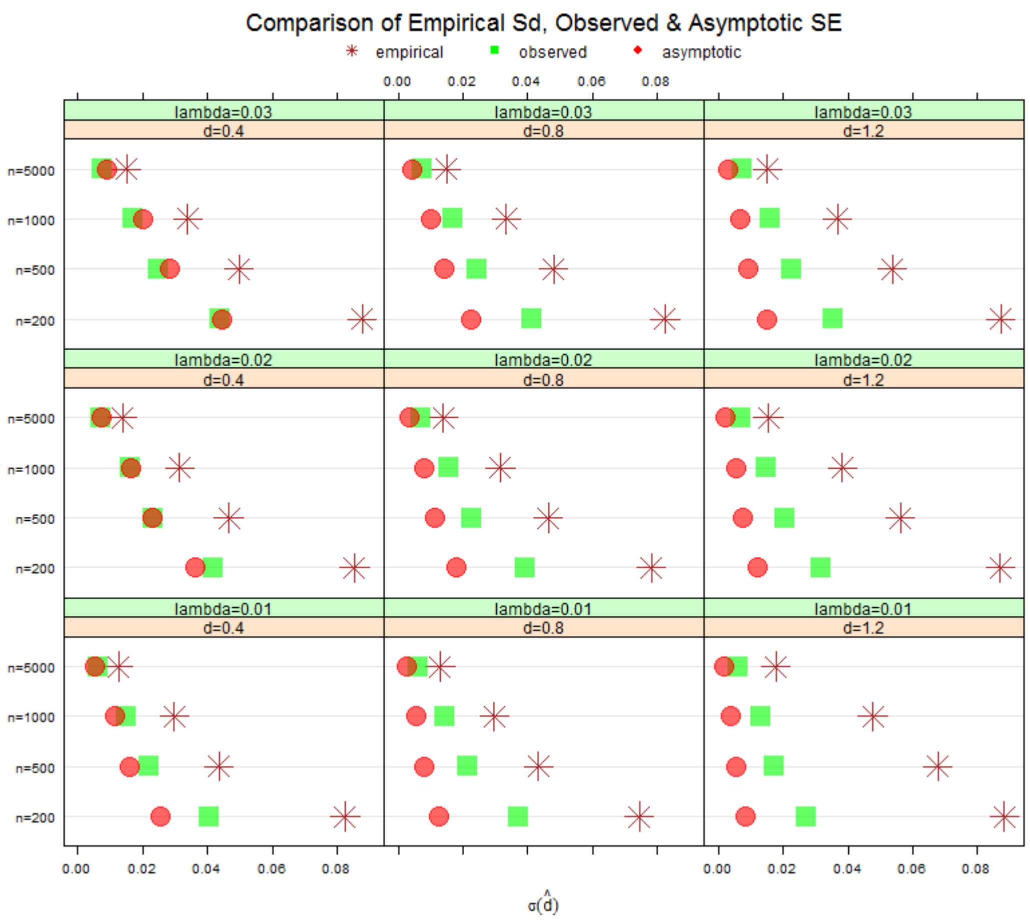
As might be expected the results are very similar to the previous dotchart, Figure (1). The only difference between the ● and • being that ● is the median of the estimates obtained by plugging the estimates into the asymptotic formula where as • represents the standard error evaluated using the true parameter settings.

Figure 9. Dotchart observed vs. expected compared with empirical, $\hat{\lambda}$



This chart is again very similar to Figure 2.

Figure 10. Dotchart observed vs. expected compared with empirical, $\hat{d}$



95% confidence intervals for the empirical estimates

The empirical estimates for the se’s are obtained from the robust median absolute deviation estimate for the standard deviation based on the MLE estimates produced by artfima(). These estimates are obtained by bootstrapping using 1000 iterations.

In all cases the 95% confidence intervals for the empirical estimates are very narrow as might be expected since 10^4 simulations were used. The confidence intervals are indicated by the vertical line. For $\hat{\lambda}$ the widest 95% confidence intervals correspond to $d = 0.4$ and $n = 200$. For $\hat{\lambda}$, the widest confidence are when $n = 200$. In all cases, the width of the 95% confidence interval is less than the size of plotting symbol used in Figure 1-4 and 9-10.

Figure 11. Dotchart showing the empirical estimates and their 95% confidence interval for the empirical standard deviation of the MLE, $\hat{\lambda}$

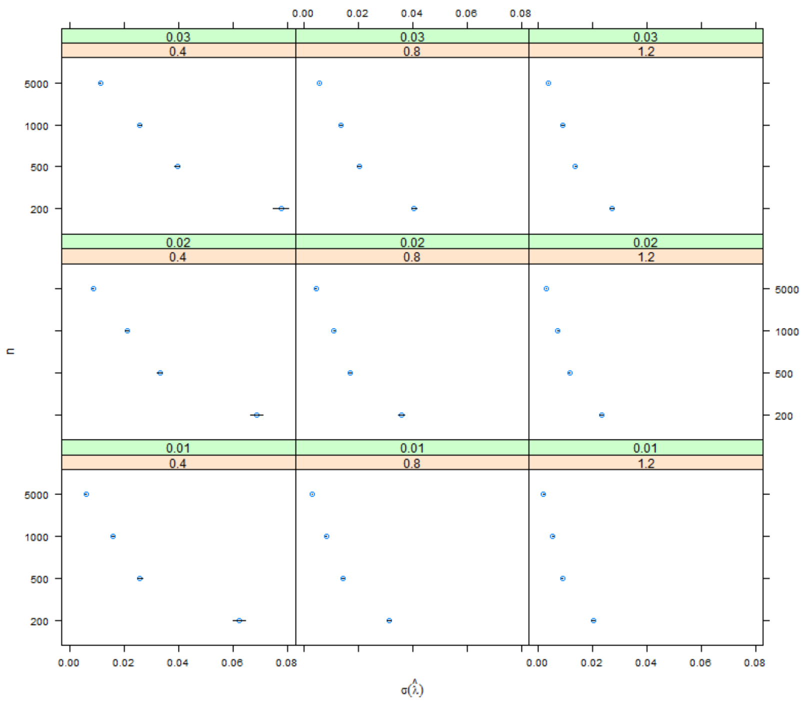
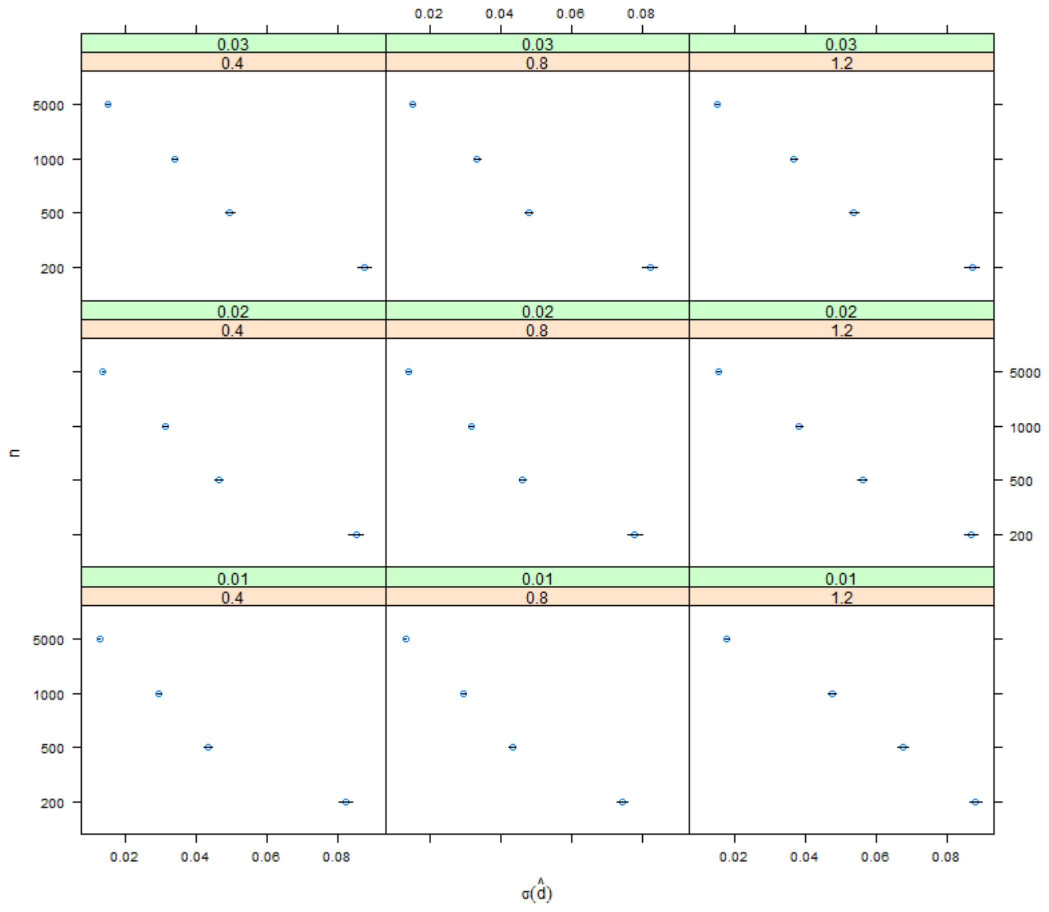


Figure 12. Dotchart showing the empirical estimates and their 95% confidence interval for the empirical standard deviation of the MLE, $\hat{d}$



R scripts and numerical tabulations

The R scripts are available in text form in the files: simCompareSE-ll-Jan17.R and SimAnalyze-Jan19.R.

Simulation script

```
#Source: simCompareSE-ll-Jan17.R
#Purpose: artfima parameter estimates & observed se
#requires: arfima, ltsa, gsl
#
#Set directory and output file here
#setwd("D:/R/2015/artfima/simulations/Dec14") #home
setwd("E:/dropbox/R/2015/artfima/simulations/Jan19") #office
#Set NSIM and file output name
#NSIM <- 100 #RStudio: ; R 231 sec
NSIM <- 10^4 #R  24758 sec, 6.9 hr

#R workspace saved here with unique FILEOUT
FILEOUT <- paste0("WS-Jan17-ll",NSIM,".RData")
#
require("artfima")
require("parallel")
NCORES <- 8
#
#OneIt - modified RGN for parallel
OneIt <- function(x){
  Ns <- c(200, 500, 1000, 5000)
  out <- numeric(0)
  LAM0 <- x$LAM
  D0 <- x$D
  for (jn in 1:length(Ns)) {
    n <- Ns[jn]
    z <- artSim(n=n, lambda=LAM0, d=D0)
    answ <- artfima(z, likAlg="Whittle")
    outj <- c(n, LAM0, D0, answ$lambdaHat, answ$dHat, answ$se, answ$convergence,
              as.numeric(answ$onBoundary)) #has length 9 since se has 2 components
    out <- c(out, outj)
  }
  out
}
#set define parameters for the simulation
Ns <- c(200,500,1000,5000)#copy defined in OneIt. Needed for parallel.
LAMs <- c(0.01, 0.02, 0.03)
Ds <- c(0.4, 0.8, 1.2)
NumPars <- length(LAMs)*length(Ds)
#set w matrix
w <- vector(mode="list", length=NumPars*NSIM)
it <- 0
for (k in 1:length(LAMs)) {
  for (i in 1:length(Ds)) {
    for (jsim in 1:NSIM) {
      it <- it+1
      w[[it]] <- list(LAM=LAMs[k], D=Ds[i])
    }
  }
}
#using parLapply
date()
startTime <- proc.time()[3]
cl <- makeCluster(spec=NCORES, type="PSOCK")
clusterSetRNGStream(cl, iseed=775123)
#initialized seed to a fixed default
#Export variables
clusterExport(cl, list("OneIt"))
#Export library
clusterEvalQ(cl, require("artfima"))
#parallel lapply
OUT<-parLapply(cl, w, fun=OneIt)
stopCluster(cl) #stop cluster
date()
totalTime <- proc.time()[3]-startTime
totalTime
save.image(file=FILEOUT)
```

Dotcharts and boxplots

```
#Source: SimAnalyze-Jan19.R
require("lattice")
require("plyr") #need for maply()
informationMatrixARTFIMA <- function(lambda, d){
  v11 <- 4*pi*exp(-2*lambda)*(1+ 0.5*exp(-2*lambda)) #eqn (43)
  v22 <- (d^2)*exp(-2*lambda)/(1-exp(-2*lambda)) #eqn (48)
  v12 <- d*log(1-exp(-2*lambda)) #eqn (54)
  matrix(c(v11,v12,v12,v22), ncol=2,
         dimnames=list(c("lambda","d"),c("lambda","d")))
}
seARTFIMA <- function(lambda, d, n){
  v <- solve(informationMatrixARTFIMA(lambda, d)*n)
  sqrt(diag(v))
}
#
#analysis of simulation results
setwd("D:/DropBox/R/2015/artfima/simulations/Jan17")
FileSimulation<-"WS-Jan17-ll10000.RData"
attach(what=FileSimulation)
#check
objects(2)
NSIM
LAMs
Ds
Ns
length(OUT)
OUT[[1]][1:9]
OUT[[1]]
length(OUT[[1]]/9)
#
#OUT components: rep(
#{n, lamda, d, lambdaHatWhittle, dHatWhittle, seWhittle}
# {n=c(200, 500, 1000,5000)})
#Note: se contains 2 elements, so there are 7 elements
#
#check converge/boundary issues
me <- matrix(0, ncol=9, nrow=NSIM*36)
i1 <- 1
for (i in 1:length(OUT)) {
  i2 <- i1+3
  me[i1:i2,] <- matrix(OUT[[i]], ncol=9, byrow=TRUE)
  i1 <- i2+1
}
dimnames(me)[[2]] <- names(me) <- c("n", "lambda", "d", "lambdaHat", "dHat",
                      "seLambdaHat", "seDHat", "convQ", "boundQ")
#
#check for convergence issues
table(me[,"convQ"])
table(me[,"boundQ"])
indBound <- me[,"boundQ"]>0
meB <- me[indBound,]
meB <- data.frame(meB)
with(meB, tapply(boundQ, list(n,lambda, d),  sum))
#
me <- me[,1:7]
medf <- as.data.frame(me) #large 360,000 rows
medf$n <- ordered(medf$n, levels=Ns, labels=paste0("n=",Ns))
medf$lambda <- ordered(medf$lambda, levels=LAMs, labels=paste0("lambda=",LAMs))
medf$d <- ordered(medf$d, levels=Ds, labels=paste0("d=",Ds))
#
#distribution of lambda-hat is highly skewed with outliers when n=200 improves
# as n increases so when n=5000 much better.
bwplot(n~lambdaHat|d*lambda, data=medf,layout=c(3,3), xlab=expression(hat(lambda)),
       main=expression("Box plots based on 10,000 replications"),
       panel=function(x,y){
         panel.bwplot(x,y)
         panel.grid(v=-1)
       }, scales=list(x=list(relation="free")))
#drilling down to n=5000
bwplot(n~lambdaHat|d*lambda, data=medf,layout=c(3,3), xlab=expression(hat(lambda)),
       main=expression("Box plots based on 10,000 replications"),
       panel=function(x,y){
         panel.bwplot(x,y)
         panel.grid(v=-1)
       }, scales=list(x=list(relation="free")),
       subset = n=="n=5000")
#
#distribution of d-hat
bwplot(n~dHat|lambda*d, data=medf,layout=c(3,3), xlab="d",
       main=expression("Box plots based on 10,000 replications"),
       panel=function(x,y){
         panel.bwplot(x,y)
         panel.grid(v=-1)
       }, scales=list(x=list(relation="free")))
bwplot(n~dHat|lambda*d, data=medf,layout=c(3,3), xlab="d",
       main=expression("Box plots based on 10,000 replications"),
       subset = n=="n=5000",
       panel=function(x,y){
         panel.bwplot(x,y)
         panel.grid(v=-1)
       }, scales=list(x=list(relation="free")))
#robust estimate of empirical sd's (use mad())
aLamHat <- with(medf, tapply(lambdaHat, list(n=n, lambda=lambda, d=d),mad))
adHat <- with(medf, tapply(dHat, list(n=n, lambda=lambda, d=d), mad))
#robust estimate of observed sd's (use median())
obseLamHat <- with(medf, tapply(seLambdaHat, list(n=n, lambda=lambda, d=d),median))
obsedHat <- with(medf, tapply(seDHat, list(n=n, lambda=lambda, d=d), median))
#vectorize result and make data.frame, dfLam
vLamHat <- c(aLamHat)
vdHat <- c(adHat)
vobseLamHat <- c(obseLamHat)
vobsedHat <- c(obsedHat)
vn <- ordered(rep(dimnames(aLamHat)[[1]],9), levels=dimnames(aLamHat)[[1]])
vLam <- ordered(rep(rep(dimnames(aLamHat)[[2]], rep(4, 3)),3),
                levels=dimnames(aLamHat)[[2]])
```

```
vd <- ordered(rep(dimnames(aLamHat)[[3]], rep(12, 3)),
              levels=dimnames(aLamHat)[[3]])
dfLam <- data.frame(emsd.lamHat=vLamHat, emsd.d=vdHat, n=vn, lam=vLam, d=vd,
                    obse.lamHat=vobseLamHat, obse.dHat=vobsedHat)
#compute se with seARTFIMA
ind <- c(t(outer(seq(0, 320000, by=4*10^4), 1:4, "+")))
out <- maply(me[ind,c(2,3,1)], seARTFIMA)
seLam <- c(aperm(out[,,,1], c(3,1,2)))
sed <- c(aperm(out[,,,2], c(3,1,2)))
dfLam2 <- cbind(dfLam, thse.lam=seLam, thse.d=sed)
#barchart and dotchart comparison
barchart(d~emsd.lamHat|n*lam, data=dfLam2)
dotplot(d~emsd.lamHat|n*lam, data=dfLam2)
#dotplot with groups
#first make new data frame. combine emsd & thse
outLam <- make.groups(dfLam2$emsd.lamHat, dfLam2$thse.lam)
levels(outLam$which)<-c("emse","asyse")
outD <- make.groups(dfLam2$emsd.d, dfLam2$thse.d)
levels(outD$which)<-c("emse","asyse")
names(outLam)[1] <- "lambda"
names(outD)[1] <- "d"
out3 <- cbind(outLam, outD) #df with c(em,th) for lam & d
dimnames(out3)[[1]] <- 1:nrow(out3)
out3 <- out3[,-4] #out3 is 72-by-3
#add ob both lam & d, creating 108-by-3 out4
out4lambda <- c(out3$lambda, dfLam2$obse.lamHat)
out4d <- c(out3$d, dfLam2$obse.dHat)
out4which <- ordered(c(as.character(out3$which), rep("obse", 36)))
out4 <- data.frame(lambda=out4lambda, d=out4d, which=out4which)
fn <- rep(dfLam2$n, 3)
flam <- rep(dfLam2$lam, 3)
fd <- rep(dfLam2$d, 3)
dfLam3 <- cbind(out4, fn=fn, flam=flam, fd=fd)
test <- ordered(dfLam3$which, levels=c("emse", "obse", "asyse"))
dfLam3$which <- test
#
#now dotplots - comparing observed and expected se methods
#lambda
dotplot(fn~lambda | fd*flam, groups=which, data=dfLam3, pch=c(8,15,19),
        cex=2.5, col=c("brown",rgb(0,1,0,0.6),rgb(1,0,0,0.6)),
        xlab=expression(sigma(hat(lambda))),
        key=list(points=list(pch=c(8,15,19),
col=c("brown",rgb(0,1,0,1),rgb(1,0,0,1))),
                 text=list(lab=c("empirical","observed","expected")),
                 columns=3,
                 title="Comparison of Empirical Sd, Observed & Expected SE"),
        )
)
#d
dotplot(fn~d | fd*flam, groups=which, data=dfLam3, pch=c(8,15,19),
        cex=2.5, col=c("brown",rgb(0,1,0,0.6),rgb(1,0,0,0.6)),
        xlab=expression(sigma(hat(d))),
        key=list(points=list(pch=c(8,15,19),
col=c("brown",rgb(0,1,0,1),rgb(1,0,0,1))),
                 text=list(lab=c("empirical","observed","asymptotic")),
                 columns=3,
                 title="Comparison of Empirical Sd, Observed & Asymptotic SE")
)
#
#dotcharts with true asymptotic rather than estimated
n <- as.integer(gsub(pattern="n=", replacement="", x=as.character(dfLam$n)))
lam <- as.numeric(gsub(pattern="lambda=", replacement="", x=as.character(dfLam$lam)))
d <- as.numeric(gsub(pattern="d=", replacement="", x=as.character(dfLam$d)))
asySDs <- maply(cbind(lambda=lam[1:36], d=d[1:36], n=n[1:36]), seARTFIMA,
.expand=FALSE)
dfLam4 <- dfLam3
dfLam4[37:72,1:2] <- asySDs
dfLam4$alambda <- sqrt(n)*dfLam4$lambda
dfLam4$ad <- sqrt(n)*dfLam4$d
#un-adjusted se
dotplot(flam~lambda | fd*fn, groups=which, data=dfLam4, pch=c(8,15,19),
        cex=2.5, col=c("brown",rgb(0,1,0,0.6),rgb(1,0,0,0.6)),
        xlab=expression(sigma(hat(lambda))),
        key=list(points=list(pch=c(8,15,19),
col=c("brown",rgb(0,1,0,1),rgb(1,0,0,1))),
                 text=list(lab=c("empirical","observed","asymptotic")),
                 columns=3,
                 title="Comparison of Empirical Sd, Observed & Asymptotic SE"),
        scales=list(x=list(relation="free", tick.number=3))
)
#Revised plots##################################################################
#se's have been standardized as well
#
#lambda: empirical, observed, asymptotic
#1: raw. Better reflects the convergence with n perhaps.
dotplot(fn~lambda | fd*flam, groups=which, data=dfLam4,
        xlab=expression(sigma(hat(lambda))), pch=c(8,15,19),cex=c(2.5,2.5,1.2),
        col=c("brown",rgb(0,1,0,0.6),rgb(0,0,0,1)),
        key=list(points=list(pch=c(8,15,19),
col=c("brown",rgb(0,1,0,1),rgb(0,0,0,1))),
                 text=list(lab=c("empirical","observed","asymptotic")),columns=3,
                 title="Empirical Sd, Observed & Asymptotic SE")
)

#2: standardized. In some cases, convergence is apparent but less so in others
dotplot(fn~alambda | fd*flam, groups=which, data=dfLam4,
        xlab=expression(sigma(hat(lambda))), pch=c(8,15,19),cex=c(2.5,2.5,1.2),
        col=c("brown",rgb(0,1,0,0.6),rgb(0,0,0,1)),
        key=list(points=list(pch=c(8,15,19),
col=c("brown",rgb(0,1,0,1),rgb(0,0,0,1))),
                 text=list(lab=c("empirical","observed","asymptotic")),columns=3,
                 title="Standardized Empirical Sd, Observed & Asymptotic SE")
)

#d: empirical, observed, asymptotic
#1: raw. Better reflects the convergence with n perhaps.
dotplot(fn~d | fd*flam, groups=which, data=dfLam4,
        xlab=expression(sigma(hat(d))), pch=c(8,15,19),cex=c(2.5,2.5,1.2),
        col=c("brown",rgb(0,1,0,0.6),rgb(0,0,0,1)),
        key=list(points=list(pch=c(8,15,19),
col=c("brown",rgb(0,1,0,1),rgb(0,0,0,1))),
                 text=list(lab=c("empirical","observed","asymptotic")),columns=3,
                 title="Empirical Sd, Observed & Asymptotic SE")
)

#2: standardized. In some cases, convergence is apparent but less so in others
dotplot(fn~ad | fd*flam, groups=which, data=dfLam4,
        xlab=expression(sigma(hat(d))), pch=c(8,15,19),cex=c(2.5,2.5,1.2),
        col=c("brown",rgb(0,1,0,0.6),rgb(0,0,0,1)),
        key=list(points=list(pch=c(8,15,19),
col=c("brown",rgb(0,1,0,1),rgb(0,0,0,1))),
                 text=list(lab=c("empirical","observed","asymptotic")),columns=3,
                 title="Standardized Empirical Sd, Observed & Asymptotic SE")
)

#generate se.csv file, uncomment code below
#tb <- dfLam2[,c(1,2,6,7,8,9)]
tb <- dfLam2
tb$n <- as.integer(gsub(pattern="n=", replacement="", x=as.character(dfLam$n)))
tb$lam <- as.numeric(gsub(pattern="lambda=", replacement="",
x=as.character(dfLam$lam)))
tb$d <- as.numeric(gsub(pattern="d=", replacement="", x=as.character(dfLam$d)))
tb <- tb[,c(3,4,5, 1, 2, 6, 7, 8, 9)]
names(tb) <- gsub(pattern="th", replacement="ex", names(tb))
names(tb) <- gsub(pattern="Hat", replacement="", names(tb))
write.table(x=tb, file="se.csv", row.names=FALSE)
tbasy <- cbind(tb[1:36,1:3], asySDs)
write.table(x=tb, file="asyse.csv", row.names=FALSE)
#
#raw R values
print(tb, digits=4)
print(tbasy, digits=4)
```

Asymptotic values for se for parameters used

The asymptotic se are determined by using the R function seARTFIMA with the various parameter settings used in the simulation. The asymptotic se are represented by ● in Figures 1 and 2.

```
> print(tbasy, digits=4)
      n  lam   d   lambda         d
1   200 0.01 0.4 0.016647 0.025341
2   500 0.01 0.4 0.010528 0.016027
3  1000 0.01 0.4 0.007445 0.011333
4  5000 0.01 0.4 0.003329 0.005068
5   200 0.02 0.4 0.016929 0.036147
6   500 0.02 0.4 0.010707 0.022862
7  1000 0.02 0.4 0.007571 0.016166
8  5000 0.02 0.4 0.003386 0.007229
9   200 0.03 0.4 0.017197 0.044604
10  500 0.03 0.4 0.010876 0.028210
11 1000 0.03 0.4 0.007691 0.019948
12 5000 0.03 0.4 0.003439 0.008921
13  200 0.01 0.8 0.016647 0.012670
14  500 0.01 0.8 0.010528 0.008014
15 1000 0.01 0.8 0.007445 0.005666
16 5000 0.01 0.8 0.003329 0.002534
17  200 0.02 0.8 0.016929 0.018074
18  500 0.02 0.8 0.010707 0.011431
19 1000 0.02 0.8 0.007571 0.008083
20 5000 0.02 0.8 0.003386 0.003615
21  200 0.03 0.8 0.017197 0.022302
22  500 0.03 0.8 0.010876 0.014105
23 1000 0.03 0.8 0.007691 0.009974
24 5000 0.03 0.8 0.003439 0.004460
25  200 0.01 1.2 0.016647 0.008447
26  500 0.01 1.2 0.010528 0.005342
27 1000 0.01 1.2 0.007445 0.003778
28 5000 0.01 1.2 0.003329 0.001689
29  200 0.02 1.2 0.016929 0.012049
30  500 0.02 1.2 0.010707 0.007621
31 1000 0.02 1.2 0.007571 0.005389
32 5000 0.02 1.2 0.003386 0.002410
33  200 0.03 1.2 0.017197 0.014868
34  500 0.03 1.2 0.010876 0.009403
35 1000 0.03 1.2 0.007691 0.006649
36 5000 0.03 1.2 0.003439 0.002974
```

Various se estimates

Columns 3 and 4, emsd.lam and emsd.d, are the empirical sd are computed using the median absolute deviation estimator applied to the estimated parameters and are represented by the stars, ✶. These are the gold standard since they should estimate the true finite sample expected values.

Columns 5 and 6 are the observed se’s are estimated using the median of the Hessian derived se estimates and are shown in Figures 1, 2, 3 and 4 with a green square, ■.

Columns 7 and 8 are the expected se’s obtained by plugging the MLE estimates into the asymptotic formula and are denoted by ● in Figures 9 and 10.

```
> print(tb, digits=4)
      n  lam   d emsd.lam  emsd.d  obse.lam   obse.d exse.lam   exse.d
1   200 0.01 0.4 0.062516 0.08236 0.0384778 0.040393 0.016647 0.025341
2   500 0.01 0.4 0.025786 0.04352 0.0148834 0.021923 0.010528 0.016027
3  1000 0.01 0.4 0.015833 0.02959 0.0084478 0.014856 0.007445 0.011333
4  5000 0.01 0.4 0.005837 0.01256 0.0030065 0.006411 0.003329 0.005068
5   200 0.02 0.4 0.068833 0.08524 0.0418114 0.041613 0.016929 0.036147
6   500 0.02 0.4 0.033148 0.04649 0.0183046 0.023334 0.010707 0.022862
7  1000 0.02 0.4 0.020948 0.03133 0.0112048 0.015935 0.007571 0.016166
8  5000 0.02 0.4 0.008493 0.01370 0.0043797 0.006945 0.003386 0.007229
9   200 0.03 0.4 0.077692 0.08763 0.0459885 0.043641 0.017197 0.044604
10  500 0.03 0.4 0.039577 0.04964 0.0218646 0.024874 0.010876 0.028210
11 1000 0.03 0.4 0.025817 0.03379 0.0138028 0.017029 0.007691 0.019948
12 5000 0.03 0.4 0.011167 0.01501 0.0055825 0.007422 0.003439 0.008921
13  200 0.01 0.8 0.031352 0.07444 0.0160286 0.036986 0.016647 0.012670
14  500 0.01 0.8 0.014221 0.04331 0.0068268 0.021268 0.010528 0.008014
15 1000 0.01 0.8 0.008074 0.02955 0.0039325 0.014565 0.007445 0.005666
16 5000 0.01 0.8 0.002893 0.01296 0.0014685 0.006381 0.003329 0.002534
17  200 0.02 0.8 0.035770 0.07802 0.0183094 0.039066 0.016929 0.018074
18  500 0.02 0.8 0.016957 0.04624 0.0085412 0.022827 0.010707 0.011431
19 1000 0.02 0.8 0.011049 0.03153 0.0054183 0.015799 0.007571 0.008083
20 5000 0.02 0.8 0.004334 0.01399 0.0021789 0.006937 0.003386 0.003615
21  200 0.03 0.8 0.040577 0.08231 0.0206525 0.041121 0.017197 0.022302
22  500 0.03 0.8 0.020233 0.04801 0.0102932 0.024326 0.010876 0.014105
23 1000 0.03 0.8 0.013591 0.03336 0.0067077 0.016848 0.007691 0.009974
24 5000 0.03 0.8 0.005519 0.01492 0.0027912 0.007424 0.003439 0.004460
25  200 0.01 1.2 0.020287 0.08826 0.0078072 0.027066 0.016647 0.008447
26  500 0.01 1.2 0.008824 0.06778 0.0034784 0.017300 0.010528 0.005342
27 1000 0.01 1.2 0.005382 0.04764 0.0022151 0.012944 0.007445 0.003778
28 5000 0.01 1.2 0.001979 0.01781 0.0009325 0.006183 0.003329 0.001689
29  200 0.02 1.2 0.023193 0.08697 0.0097300 0.031721 0.016929 0.012049
30  500 0.02 1.2 0.011508 0.05620 0.0049568 0.020509 0.010707 0.007621
31 1000 0.02 1.2 0.007140 0.03817 0.0032820 0.014812 0.007571 0.005389
32 5000 0.02 1.2 0.002904 0.01544 0.0014178 0.006821 0.003386 0.002410
33  200 0.03 1.2 0.027244 0.08718 0.0116656 0.035485 0.017197 0.014868
34  500 0.03 1.2 0.013693 0.05374 0.0062339 0.022509 0.010876 0.009403
35 1000 0.03 1.2 0.008973 0.03674 0.0042228 0.016128 0.007691 0.006649
36 5000 0.03 1.2 0.003728 0.01508 0.0018305 0.007342 0.003439 0.002974
```

Boostrap estimates for the 95% MOE

R script dotcharts in Figures 11 and 12

```
#Source: MOE-Jan21.R
#boostrap estimation of the MOE for the empirical standard devations
#
setwd("D:/DropBox/R/2015/artfima/simulations/Jan17")
FileSimulation<-"WS-Jan17-ll10000.RData"
attach(what=FileSimulation)
objects(2)


#simulation results#WS-Dec27-ll10000.RData
setwd("D:/DropBox/R/2015/artfima/simulations/Dec27")
FileSimulation<-"WS-Dec27-ll10000.RData"
attach(what=FileSimulation)
#check
objects(2)
#
#OUT components: rep(
#{n, lamda, d, lambdaHatWhittle, dHatWhittle, seWhittle}
# {n=c(200, 500, 1000,5000)})
#Note: se contains 2 elements, so there are 7 elements
#
me <- matrix(0, ncol=9, nrow=NSIM*36)
i1 <- 1
for (i in 1:length(OUT)) {
  i2 <- i1+3
  me[i1:i2,] <- matrix(OUT[[i]], ncol=9, byrow=TRUE)
  i1 <- i2+1
}
me <- me[,-(8:9)]
dimnames(me)[[2]] <- names(me) <- c("n", "lambda", "d", "lambdaHat", "dHat",
                                    "seLambdaHat", "seDHat")
medf <- as.data.frame(me) #large 360,000 rows
#
medf$n <- ordered(medf$n, levels=Ns, labels=paste0("n=",Ns))
medf$lambda <- ordered(medf$lambda, levels=LAMs, labels=paste0("lambda=",LAMs))
medf$d <- ordered(medf$d, levels=Ds, labels=paste0("d=",Ds))
#
#robust estimate of empirical sd's (use mad())
aLamHat <- with(medf, tapply(lambdaHat, list(n=n, lambda=lambda, d=d),mad))
adHat <- with(medf, tapply(dHat, list(n=n, lambda=lambda, d=d), mad))
vLamHat <- c(aLamHat)
vdHat <- c(adHat)
#
#Bootstrap
NBoot <- 1000 #155 sec
bmad <- function(x) {
  mad(sample(x, size=length(x), replace=TRUE))
}
```

```
startTime <- proc.time()[3]
set.seed(227771)
bdHat <- bLamHat <- matrix(numeric(0), nrow=NBoot, ncol=36)
for (iBoot in 1:NBoot) {
  bLamHat[iBoot,] <- c(with(medf,tapply(lambdaHat, list(n=n, lambda=lambda, d=d),
                                     bmad)))
  bdHat[iBoot,] <- c(with(medf, tapply(dHat, list(n=n, lambda=lambda, d=d),
                                    bmad)))
}
proc.time()[3]-startTime
#
boot.lam<-apply(bLamHat, 2, mad)
boot.d<-apply(bdHat, 2, mad)
layout(matrix(1:2, ncol=2))
boxplot(boot.lam, ylab=expression(paste("Bootstrap sd of empirical sd ",lambda)))
boxplot(boot.d, ylab="Bootstrap sd of empirical sd d")
layout(1)
summary(boot.lam)
summary(boot.d)
#
#table with empirical sd and their bootstrap 95% MOE
moe.lam <- 1.96*boot.lam
moe.d <- 1.96*boot.d
vLamHat <- c(aLamHat)
vdHat <- c(adHat)
vn <- ordered(rep(dimnames(aLamHat)[[1]],9), levels=dimnames(aLamHat)[[1]])
vLam <- ordered(rep(rep(dimnames(aLamHat)[[2]], rep(4, 3)),3),
                levels=dimnames(aLamHat)[[2]])
vd <- ordered(rep(dimnames(aLamHat)[[3]], rep(12, 3)),
              levels=dimnames(aLamHat)[[3]])
dfLam <- data.frame(n=vn, lam=vLam, d=vd, emsd.lamHat=vLamHat, emsd.d=vdHat,
                    moe.lamHat=moe.lam, moe.d=moe.d)
#covert dfLam to nicer table
moe.tb <- dfLam
moe.tb$n <- as.integer(gsub(pattern="n=", replacement="", x=as.character(dfLam$n)))
moe.tb$lam <- as.numeric(gsub(pattern="lambda=", replacement="",
x=as.character(dfLam$lam)))
moe.tb$d <- as.numeric(gsub(pattern="d=", replacement="", x=as.character(dfLam$d)))
moe.tb <- moe.tb[,c(1:4, 6, 5, 7)]
write.table(x=moe.tb, file="MOE.csv", row.names=FALSE)
#
#dotcharts with MOE
#lambda
MOE <- moe.tb$moe.lamHat
xyplot(factor(n)~emsd.lamHat | factor(d)*factor(lam), data=moe.tb,
       panel=function(x,y,subscripts=TRUE){
         panel.xyplot(x,y)
         panel.segments(x-MOE[subscripts], y, x+MOE[subscripts], y)
       },
       xlab=expression(sigma(hat(lambda))), ylab="n"
)
#d
MOE <- moe.tb$moe.d
xyplot(factor(n)~emsd.d | factor(d)*factor(lam), data=moe.tb,
       panel=function(x,y,subscripts=TRUE){
         panel.xyplot(x,y)
         panel.segments(x-MOE[subscripts], y, x+MOE[subscripts], y)
       },
       xlab=expression(sigma(hat(d))), ylab="n"
)
#
moe.tb
```

95% margin of error for the estimates

```
> moe.tb
      n  lam   d emsd.lamHat   moe.lamHat     emsd.d        moe.d
1   200 0.01 0.4 0.062516173 2.289023e-03 0.08235745 0.0018756597
2   500 0.01 0.4 0.025785883 9.540830e-04 0.04352078 0.0011392786
3  1000 0.01 0.4 0.015832621 4.601704e-04 0.02958979 0.0006472371
4  5000 0.01 0.4 0.005836959 1.436051e-04 0.01256399 0.0003223247
5   200 0.02 0.4 0.068833294 2.303403e-03 0.08523732 0.0021775851
6   500 0.02 0.4 0.033148085 9.630217e-04 0.04648802 0.0011779264
7  1000 0.02 0.4 0.020948204 4.891804e-04 0.03132501 0.0007047637
8  5000 0.02 0.4 0.008493458 2.010276e-04 0.01369642 0.0002954322
9   200 0.03 0.4 0.077691960 2.663036e-03 0.08762964 0.0019772438
10  500 0.03 0.4 0.039576732 9.851970e-04 0.04964446 0.0012852686
11 1000 0.03 0.4 0.025816616 6.702478e-04 0.03379331 0.0007752365
12 5000 0.03 0.4 0.011167068 2.459910e-04 0.01500652 0.0003679581
13  200 0.01 0.8 0.031352437 7.384572e-04 0.07444004 0.0016488212
14  500 0.01 0.8 0.014221089 3.565200e-04 0.04331397 0.0009662988
15 1000 0.01 0.8 0.008074007 2.354249e-04 0.02955021 0.0006558423
16 5000 0.01 0.8 0.002893437 6.309865e-05 0.01295638 0.0002999702
17  200 0.02 0.8 0.035769980 1.044259e-03 0.07802284 0.0019700338
18  500 0.02 0.8 0.016957458 4.047513e-04 0.04624209 0.0011062788
19 1000 0.02 0.8 0.011049462 2.708791e-04 0.03153470 0.0006793870
20 5000 0.02 0.8 0.004334229 1.042006e-04 0.01398576 0.0003358495
21  200 0.03 0.8 0.040576811 1.075932e-03 0.08230583 0.0020021339
22  500 0.03 0.8 0.020233163 5.458125e-04 0.04801224 0.0010953302
23 1000 0.03 0.8 0.013590793 3.068753e-04 0.03336048 0.0008432904
24 5000 0.03 0.8 0.005518904 1.200417e-04 0.01492328 0.0003521849
25  200 0.01 1.2 0.020287085 5.783938e-04 0.08825988 0.0018513042
26  500 0.01 1.2 0.008824094 2.046651e-04 0.06778286 0.0015551071
27 1000 0.01 1.2 0.005381921 1.247503e-04 0.04763510 0.0011548606
28 5000 0.01 1.2 0.001979208 4.576788e-05 0.01780587 0.0004388657
29  200 0.02 1.2 0.023192657 6.092607e-04 0.08696854 0.0019702585
30  500 0.02 1.2 0.011508361 2.874117e-04 0.05619615 0.0013068597
31 1000 0.02 1.2 0.007140399 1.666473e-04 0.03817126 0.0009524191
32 5000 0.02 1.2 0.002903638 7.001539e-05 0.01543805 0.0003506008
33  200 0.03 1.2 0.027243964 7.259656e-04 0.08718325 0.0020681721
34  500 0.03 1.2 0.013693488 3.520079e-04 0.05373521 0.0012459406
35 1000 0.03 1.2 0.008972532 2.188308e-04 0.03673925 0.0009203786
36 5000 0.03 1.2 0.003728092 8.465782e-05 0.01508326 0.0003698350
```