# Size-power tradeoff Illustration

## Setting up

```
rm(list=ls());
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(ForeComp)
library(astsa)
```

```
##
## Attaching package: 'astsa'
```

```
## The following object is masked from 'package:forecast':
##
##     gas
```

### Global tuning parameter

```
nlen    = 200;   # length of time-series data
Mchoice = 2;   # bandwidth
nsim    = 1000; # number of simulation to compute size and power
cl      = 0.05; # confidence level
```

### Data generating process

```
T = 1000 + nlen;
dt = matrix(NA, T, 1);
dt0 = 0;
for (t in 1:T){
  dt0 = 0.5 + 0.5*dt0 + rnorm(1);
  dt[t] = dt0;
}
dt = dt[1001:T, , drop=F]; # because we are starting from the arbitrary value, we need to discrard init

mut = mean(dt); # demean
dt_tilde = dt - mut;
```

### Estimate ARIMA and extract information

This info will be used to compute the spectral density (i.e., long-run variance)

```
a = auto.arima(y=dt_tilde, max.p = 12, max.q=12, stationary=T, ic="aic", seasonal=F, allowmean=F);
i_ar = grep("ar", names(a$coef));
i_ma = grep("ma", names(a$coef));
m_sim = list("ar"=a$coef[i_ar], "ma"=a$coef[i_ma]);
```

```
if (length(m_sim$ar)==0){m_sim$ar=0.0}
if (length(m_sim$ma)==0){m_sim$ma=0.0}
```

#### Size computation

```
mat_rej  = matrix(NA, nsim, 1);
mat_stat = matrix(NA, nsim, 1);
mat_dt   = matrix(NA, nlen, nsim); #matrix that stores data (for size calculation)
for (irep in 1:nsim){
  dt_sim = arima.sim(m_sim, n=nlen, innov = rnorm(nlen, 0, sqrt(a$sigma2)));
  rst = dm.test.bt(dt_sim, M=NA, cl=cl);
  mat_rej[irep, ] = rst$rej;
  mat_stat[irep,] = rst$stat;
  mat_dt[, irep] = dt_sim;
}
print("empirical size");
```

```
## [1] "empirical size"
```

```
print(mean(mat_rej));
```

```
## [1] 0.109
```

```
size_distortion = mean(mat_rej) - cl;
```

### Documenting trade-off

The goal is to test $H_0 : \mu = 0$. We consider a local alternative $H_\delta : \mu = \frac{\sqrt{\Omega}}{\sqrt{T}}\delta$

## setting

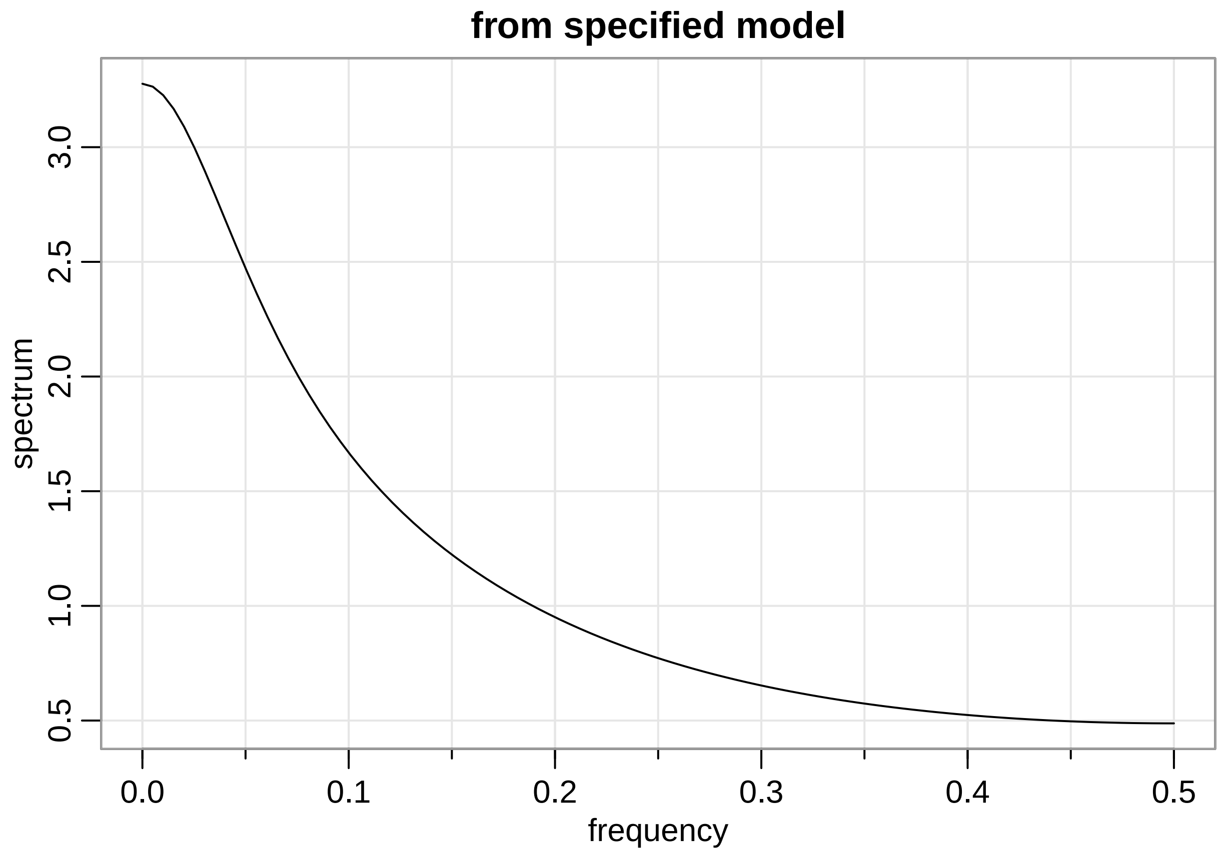
```
ndel = 50; #number of deltas
del_tilde = 10; #largest delta
del_grid = seq(from=-del_tilde, to=del_tilde, length.out=ndel);
```

## 1: Power of oracle test

A oracle test knows the long-run variance

Note that the long-run variance equals to $2pi$f(0) where f() is a spectral density

```
# long-run variance
ss = arma.spec(ar = m_sim$ar, ma = m_sim$ma, var.noise = a$sigma2, n.freq = 100);
```




```
Om = ss$spec[1]; #this is 2*pi*f(0), spectrum at zero rather than a spectral density at zero

# Oracle test
mat_stat_o = matrix(NA, nsim, 1);
mat_rej_o = matrix(NA, nsim, 1);
for (irep in 1:nsim){
  dt_sim = mat_dt[, irep,drop=F];
  dmstat = mean(dt_sim) / sqrt(Om / nlen); #Oracle's statistic

  pval = 2 * stats::pnorm(-abs(dmstat), mean=0, sd=1); #p-val based on normal approximation

  rej = pval < cl; #reject decision

  mat_stat_o[irep, 1] = dmstat;
  mat_rej_o[irep,1] = rej;
}
print("empirical size of oracle");
```

```
## [1] "empirical size of oracle"
```

```
print(mean(mat_rej_o));
```

```
## [1] 0.054
```

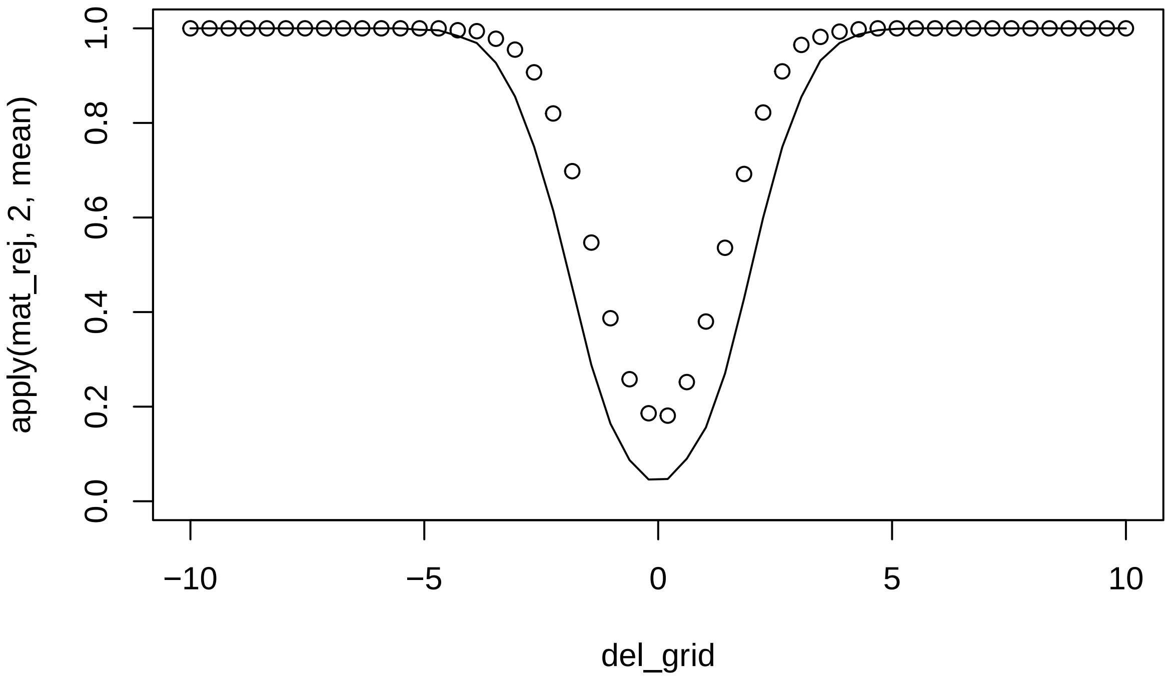
```
# find a cut-off (c*) to get size-corrected critical value
c05_star_o = quantile(abs(mat_stat_o), (1-cl));
```

```r
# size corrected power
mat_rej_o = matrix(NA, nsim, ndel);
mat_rej2_o = matrix(NA, nsim, ndel);
for (idel in 1:ndel){
  for (irep in 1:nsim){
    dt_sim = mat_dt[, irep, drop=F] + (1/sqrt(nlen)) *sqrt(Om)*del_grid[idel];
    dmstat = mean(dt_sim) / sqrt(Om / nlen); #statistic
    pval = 2*stats::pnorm(-abs(dmstat), mean=0, sd=1); # p-val from normal approximation
    rej = pval < cl; # reject decision
    mat_rej_o[irep, idel] = rej; # this is for raw power
    mat_rej2_o[irep, idel] = abs(dmstat) > c05_star_o; # for size-corrected power
  }
}
```

## 2: Power of a standard test

With a standard test we have to estimate the denominator.

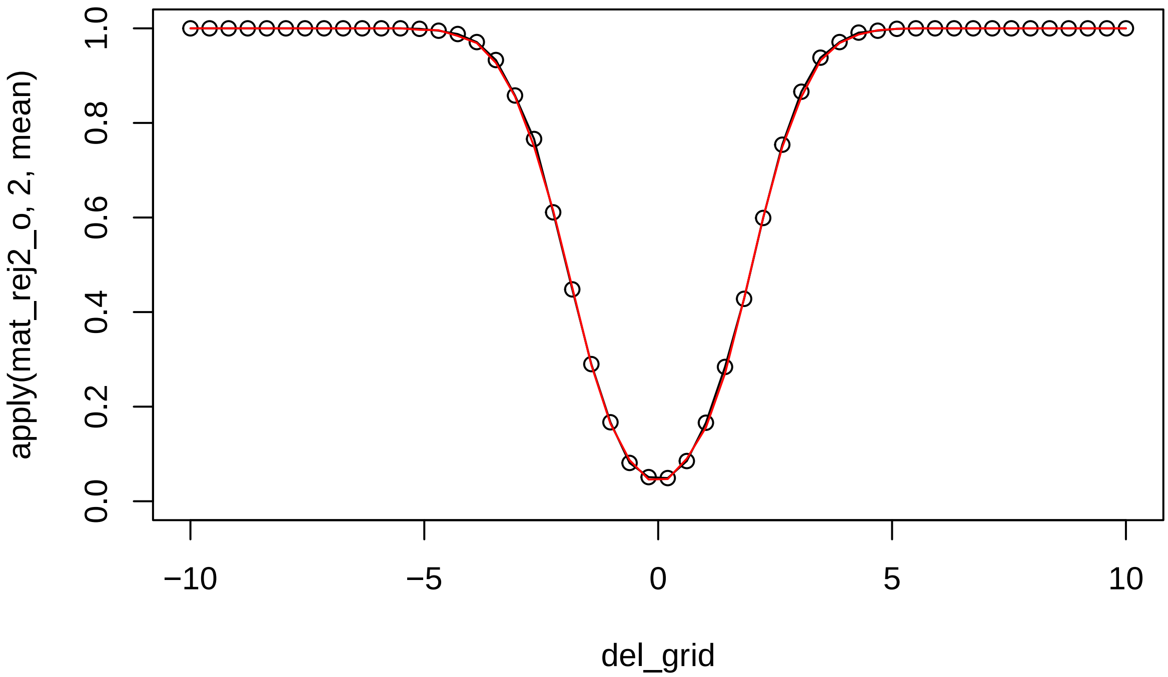
```r
# --- size to compute size-corrected critical value
mat_stat = matrix(NA, nsim, 1);
for (irep in 1:nsim){
  dt_sim = mat_dt[, irep, drop=F];
  rst = dm.test.bt(dt_sim, M= Mchoice, cl = cl);
  mat_stat[irep,1] = rst$stat;
}

# --- size-corrected crit val
c05_star = quantile(abs(mat_stat), (1-cl));

# --- size-corrected power
mat_stat = matrix(NA, nsim, ndel);
mat_rej  = matrix(NA, nsim, ndel);
mat_rej2 = matrix(NA, nsim, ndel);
for (idel in 1:ndel){
  for (irep in 1:nsim){
    dt_sim = mat_dt[, irep, drop=F] + (1/sqrt(nlen)) * sqrt(Om) * del_grid[idel];
    rst = dm.test.bt(dt_sim, M = Mchoice, cl = cl);
    mat_rej[irep, idel] = rst$rej; #for raw power
    mat_stat[irep, idel] = rst$stat;
    mat_rej2[irep, idel] = abs(rst$stat) > c05_star; # for size-corrected power
  }
}

# --- plotting (two powers: raw versus size-adjusted power)
plot(del_grid, apply(mat_rej, 2, mean), ylim=range(0, 1.0))
lines(del_grid, apply(mat_rej2, 2, mean))
```




## Trade-off

Two power curves.

```
plot(del_grid, apply(mat_rej2_o, 2, mean), ylim=range(0,1));
lines(del_grid, apply(mat_rej2_o, 2, mean));
lines(del_grid, apply(mat_rej2, 2, mean), col = "red");
```




Maximum power loss

```
max_power_loss = max( apply(mat_rej2_o, 2, mean) - apply(mat_rej2, 2, mean) );
```

Final numbers

```
print(paste0("At M = ", Mchoice));
```

```
## [1] "At M = 2"
```

```
print(paste0("size distortion = ", size_distortion));
```

```
## [1] "size distortion = 0.059"
```

```
print(paste0("maximum power loss = ", max_power_loss));
```

```
## [1] "maximum power loss = 0.016"
```