Periodogram weiterentwicklung

Carsten Stahl

2025-04-03

Einführung

Die erste Implementierung warf Fragen zu dem implementierung des Periodogramm auf. Es wird also hier versucht, die Funktion nochmal zu implementieren.

Hier sind die vorherigen

```
gridMA <- function(N, M, K, padding = nrow(K) %/% 2) {
  N_tilde <- N + 2 * padding
  M_tilde <- M + 2 * padding
  n <- N_tilde * M_tilde

  eps <- matrix(rnorm(n),
                nrow = M_tilde,
                ncol = N_tilde)

  x <- convolve_image(eps, K)
  # cut padding
  return(x[(padding+1):(M+padding),(padding+1):(N+padding)])
}


I <- function(x){
  N <- nrow(x)
  M <- ncol(x)

  # apply scaling ((2*pi)^2 is already in the fft)
  res <- (1/(N * M))*abs(fft(x))^2

  # fft shift such that 0 is at the center and nyquist is at the borders
  return(res[
    c(
      ((N %/% 2)+1):N,
      1:(N %/% 2)),
    c(
      ((M %/% 2)+1):M,
      1:(M %/% 2))]
    )
}

I_smooth <- function(I, kernel, hr = 1, hc = 1) {
  N <- nrow(I)
```

```r
  M <- ncol(I)

  omega_M <- 2*pi*((-((M - 2) %/% 2)):(M %/% 2))/M
  omega_N <- 2*pi*((-((N - 2) %/% 2)):(N %/% 2))/N
  I_s <- function (omega_1, omega_2) {
    K_M <- kernel(omega_M - omega_1, hr)
    K_N <- kernel(omega_N - omega_2, hc)

    weights <- K_N %*% t(K_M)
    weighted <- sum(weights * I)

    normalized <- (1/(N*M*hr*hc))*weighted
    return(normalized)
  }
  class(I_s) <- c("I_smooth", "grid_function")
  I_s
}

# instanciate generic
evaluate <- function(f) {
  UseMethod("evaluate")
}

evaluate.grid_function <- function(I_s, N = 30, M = 30) {
  x <- seq(-pi+1e-5, pi-1e-5, length.out = N)
  y <- seq(-pi+1e-5, pi-1e-5, length.out = M)


  z <- matrix(0, N, M)
  for (i in 1:length(x)) {
    for (j in 1:length(y)) {
      z[i, j] <- I_s(x[i], y[j])
    }
  }
  return(list(x = x, y = y, z = z))
}

# implementing plot generic for quick and easy use
plot.I_smooth <- function(I_s, ...) {
  vals = evaluate(I_s)
  persp(vals$x,
        vals$y,
        vals$z,
        ticktype = "detailed",
        zlab = "",
        xlab = "",
        ylab = "",
        main = "plot of smoothed Periodogram")
}

tr_k <- function(u, h = 1) ifelse(u/h <= 1 & u / h >= -1, (35 / 32) * (1-(u/h)^2)^3, 0)
```

Verifizierung

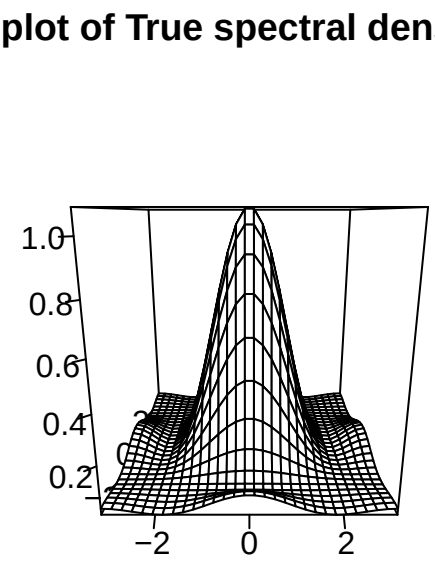
Wir werden checken, ob die Schätzung der Spektraldichte gegen den wahren Wert konvergiert. Dafür brauchen wir aber erstmal die wahre Spektraldichte. Diese bekommen wir aus der Fourier Reihe der ACF (für den räumlichen MA-Prozess):

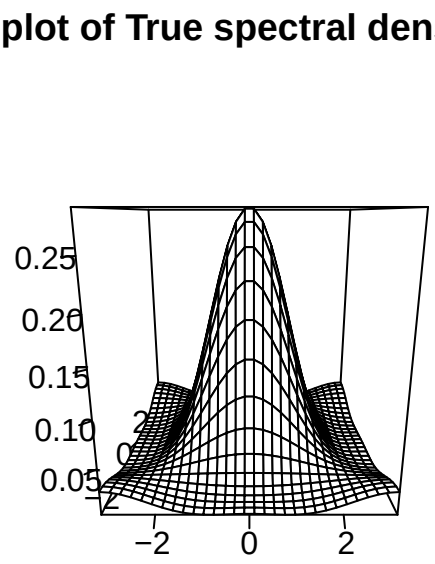
Die wahre spektraldichte ergibt sich durch die diskrete Fourier transformation der Kovarianzfunktion $C_K(k,l)$ wobei K die Matrix aus dem MA-Teil ist.:

$$f(\omega,\tilde{\omega}) = \frac{1}{(2\pi)^2}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} C_K(k,l)\exp(-i(k\omega+l\tilde{\omega})) = \frac{1}{(2\pi)^2}\sum_{k=-3}^{3}\sum_{l=-3}^{3} C_K(k,l)\exp(-i(k\omega+l\tilde{\omega}))$$

```
C <- function(k,l, K) {
  # assuming K is a square matrix
  n <- nrow(K)
  # compute overlap in weights
  # but only if there is an overlap
  # being a bit more generous here to avoid
  # making the code too complicated
  if (abs(k) + abs(l) >= 2*n) {
    return(0)
  }

  first <- matrix(0, nrow = n + abs(k), ncol = n + abs(l))
  second <- matrix(0, nrow = n + abs(k), ncol = n + abs(l))

  first[1:n, 1:n] <- K
  second[(abs(k)+1):(n+abs(k)), (abs(l)+1):(n+abs(l))] <- K

  return(sum(first * second))
}

f_true_MA <- function(K) {
  K <- K
  n <- ncol(K)
  f <- function(omega, omega_tilde) {
    summation <- 0
    for (k in -n:n) {
      for (l in -n:n) {
        summation <- summation + C(k, l, K) * exp(-1i*(k * omega + l*omega_tilde))
      }
    }
    return(Re((1/(2*pi)^2)*summation))
  }
  class(f) <- c("spectral_density", "grid_function")
  return(f)
}

plot.spectral_density <- function(f) {
  vals = evaluate(f)
  persp(vals$x, vals$y, vals$z,
        xlab = "",
        ylab = "",
        zlab = "",
```

```
        ticktype = "detailed",
        main = "plot of True spectral density")
}

K <- matrix(0.7, 3, 3)
K[2,2] <- 1

K_tilde <- matrix(0.3, 3, 3)
K_tilde[2,2] <- 1


f_K <- f_true_MA(K)
f_K_tilde <- f_true_MA(K_tilde)


par(mfrow = c(1,2))
plot(f_K)
plot(f_K_tilde)
```

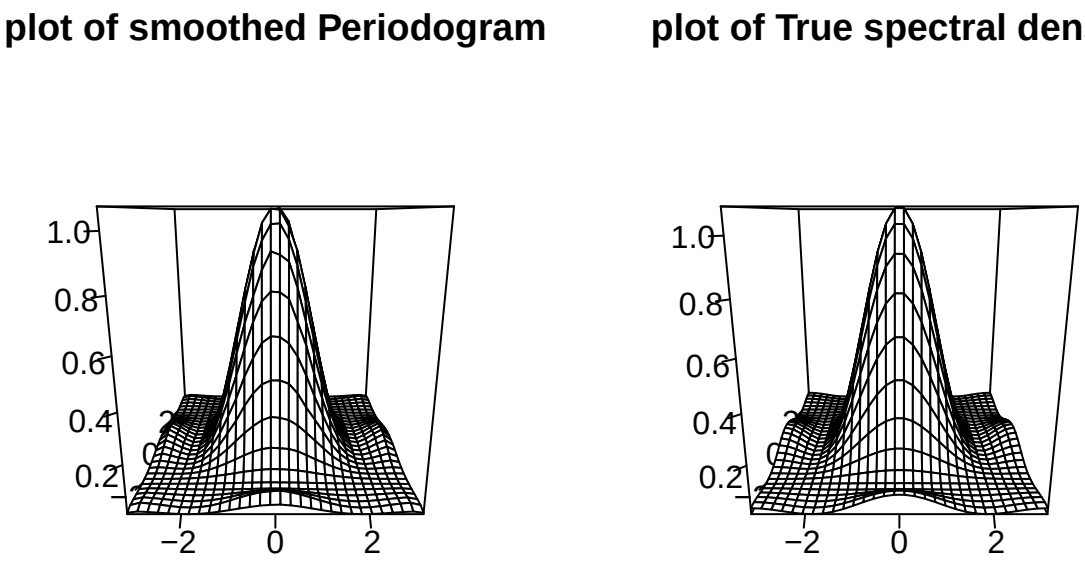




Konsistenz zwischen den Schätzern

Lass uns jetzt die Spektraldichte für den $MA(q)$ schätzen und dann mit der wahren Dichte vergleichen:

```
# simulating a large sample and estimating the spectral density

x <- gridMA(1500, 1500, K)
par(mfrow = c(1,2))
plot(I_smooth(I(x), tr_k, hr = .3, hc = .3))
plot(f_true_MA(K))
```



Wir sehen einen leichten Bias für die Ränder, was typisch für die Kernel density estimation ist.

Aber sonst scheint das Problem gelöst zu sein. Lass uns jetzt eine MSE funktion implementieren, welche den Fehler der Schätzung ausrechnet:

```
mse_hat <- function(est, f) {
  vals_est <- evaluate(est)$z
  vals_f <- evaluate(f)$z

  return(sum((vals_est - vals_f)^2)/length(vals_est))
}

x <- gridMA(20, 20, K)
mse_hat(I_smooth(I(x), tr_k), f_true_MA(K))
```

```
## [1] 0.01274306
```

Jetzt können wir uns auch die Konvergenzrate anschauen:

```r
errors <- numeric(20)
for (i in 1:20) {
  x <- gridMA(10*i, 10*i, K)
  errors[i] <- mse_hat(I_smooth(I(x), tr_k, hr=1, hc =1), f_true_MA(K))
}
plot(((1:20)*10)^2, errors, main = "error convergence", type="l",
     xlab = "n")
```