TSA Assignment 8

Christian Hilscher - 1570550

4/5/2020

Question 13

I first rewrite c_j as a function of the transfer function:

$$\begin{aligned} c_j &= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{ij\lambda}\Psi(\lambda)d\lambda \\ &= \frac{1}{2\pi}\left(\underbrace{\int_{-\pi}^{-\lambda_h}(\cdot)d\lambda}_{0} + \int_{-\lambda_h}^{-\lambda_l}(\cdot)d\lambda + \underbrace{\int_{-\lambda_l}^{-\lambda_h}(\cdot)d\lambda}_{0} + \int_{\lambda_l}^{\lambda_h}(\cdot)d\lambda + \underbrace{\int_{\lambda_h}^{\pi}(\cdot)d\lambda}_{0}\right) \end{aligned}$$

This splitting up now allows me to analyse the terms which are not zero:

$$\begin{aligned} c_j =& \frac{1}{2\pi}\left(\int_{-\lambda_h}^{-\lambda_l} e^{ij\lambda}d\lambda + \int_{\lambda_l}^{\lambda_h} e^{ij\lambda}d\lambda\right) \\ =& \frac{1}{2\pi}\left(\left[\frac{1}{ij}e^{ij\lambda}\right]_{-\lambda_h}^{-\lambda_l} + \left[\frac{1}{ij}e^{ij\lambda}\right]_{\lambda_l}^{\lambda_h}\right) \qquad \text{for } j \neq 0 \\ =& \frac{1}{2\pi}\frac{1}{ij}\left(e^{ij\lambda_h} - e^{-ij\lambda_h}\right) - \left(e^{ij\lambda_l} - e^{-ij\lambda_l}\right) \\ =& \frac{1}{2\pi}\frac{1}{ij}\left(2i\sin(j\lambda_h)\right) - \left(2i\sin(j\lambda_l)\right) \\ c_j =& \frac{1}{\pi j}(\sin(j\lambda_h) - \sin(j\lambda_l)) \quad \text{for } j \neq 0 \end{aligned}$$

Coming to $j = 0$ one can show that

$$\begin{aligned} c_0 =& \frac{1}{2\pi}\left(\int_{-\lambda_h}^{-\lambda_l} e^{0i\lambda}d\lambda + \int_{\lambda_l}^{\lambda_h} e^{0i\lambda}d\lambda\right) \\ =& \frac{1}{2\pi}\left(\int_{-\lambda_h}^{-\lambda_l} 1d\lambda + \int_{\lambda_l}^{\lambda_h} 1d\lambda\right) \\ =& \frac{1}{2\pi}\left(-\lambda_h + \lambda_h + \lambda_h - \lambda_l\right) \\ c_0 =& \frac{1}{\pi}(\lambda_h - \lambda_l) \end{aligned}$$

Question 14

DGP

To generate the data I use 500 points to represent all λ_0 which is enough to make the grpahs look nice and convey the overall idea without being computationally too intensive.

```
### Initiating starting values and dataframes for storage
points <- 500
lambdas <- seq(from=0,
               to=pi,
               length.out = points)
q_list = c(1, 3, 10)

df_f <- f_lambda(lambdas, q_list)
df_VR <- VR(lambdas, q_list)
```

Getting the spectral density function

As a start I rewrite the process a little bit

$$
\begin{aligned}
x_t =& \frac{1}{2q+1} \sum_{|j|\leq q} \epsilon_{t-j} \\
B(L) =& \frac{1}{2q+1} \sum_{|j|\leq q} \\
B(e^{-i\lambda}) =& \frac{1}{2q+1} \sum_{|j|\leq q} e^{-i\lambda}
\end{aligned}
$$

Now using $\sigma = 1$, the spectral density $f_x(\lambda)$ is given by

$$
\begin{aligned}
f_x(\lambda) =& |B(e^{-i\lambda})|^2 \frac{\sigma}{2\pi} \\
=& \frac{1}{2\pi} \left(\frac{1}{2q+1} \sum_{|j|\leq q} e^{-i\lambda} \right) \left(\frac{1}{2q+1} \sum_{|j|\leq q} e^{i\lambda} \right) \\
=& \frac{1}{2\pi} \frac{1}{(2q+1)^2} \sum_{|j|\leq q} e^{-i\lambda} \sum_{|j|\leq q} e^{i\lambda} \qquad (1)
\end{aligned}
$$

The implementation of (1) is given by the function below with the argument *Mod* set to 0 to get imaginary numbers.

```
spectral_dens <- function(q, lambda, Mod=0){
  i = complex(real = 0, imaginary = 1)

  # First sum
  total_l <- 0
  for (l in seq((-q), q)){
    value <- exp(i*lambda*(-l))
    total_l <- total_l + value
  }
```

```
  # Second sum
  total_j <- 0
  for (j in seq((-q), q)){
    value <- exp(i*lambda*j)
    total_j <- total_j + value
  }

  T_c <- 1/(2*q + 1)**2 * (total_j*total_l)
  spectral_d <- T_c * 1/(2*pi)

  # Whether or not to take the norm of the vector to convert it back
  if (Mod==0){
    return(spectral_d)
  } else if (Mod==1){
    return(Mod(spectral_d))
  }
  else{
    stop('Please provide either 0 or 1 as argument for Mod')
  }
}
```

To actually compute the spectral density for $\lambda_0 \in [0, \pi]$ I run the following function which takes the list of 500 lambdas and the 3 values of q as arguments. The function *mapply* allows me to use the whole vector as input for the scalar valued function defined above. For plotting the values I take the norm and finally add them to a dataframe.

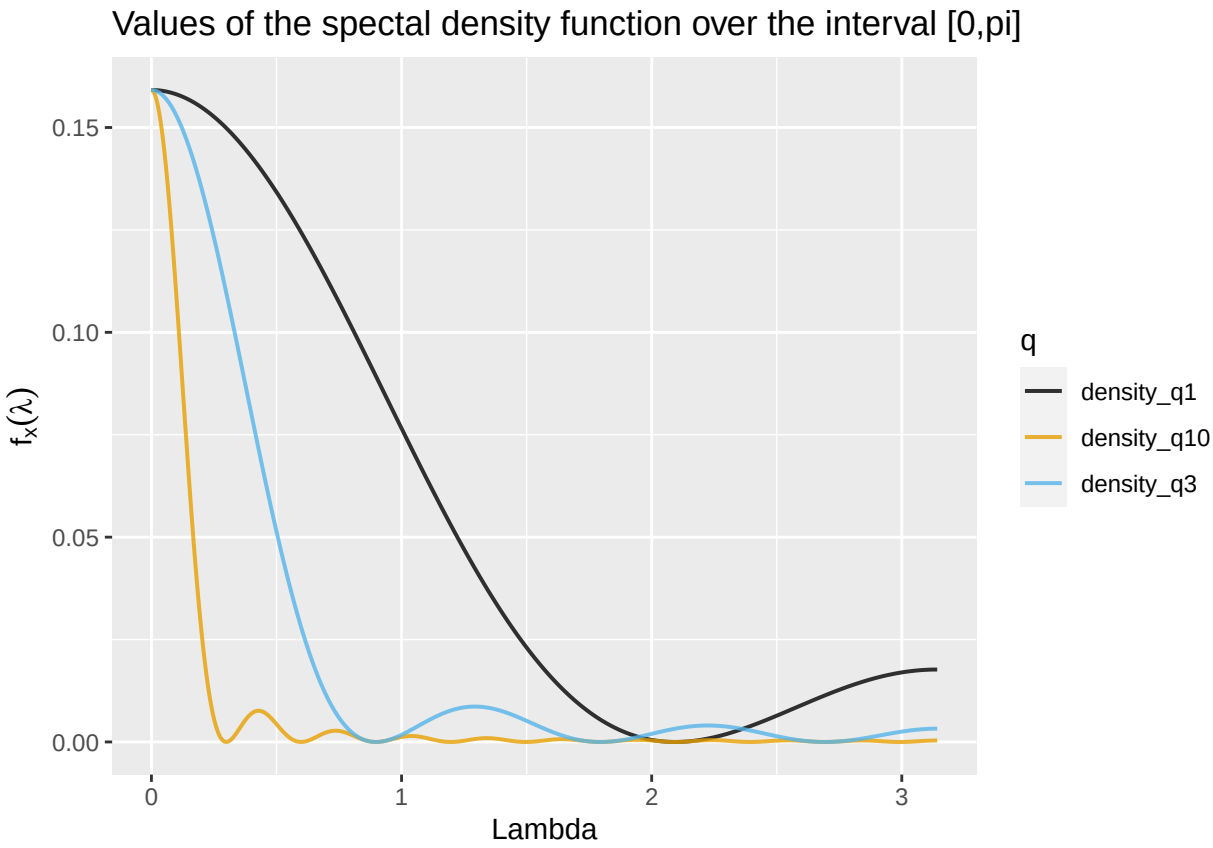
```
f_lambda <- function(lambdas, q_list){
  df <- data.frame(lambdas)
  for (q in q_list){

    sp_densitites <- mapply(spectral_dens, q=q, lambda=lambdas)
    num_sp_densitites <- Mod(sp_densitites)

    name <- "density_q"
    fullname <- paste(name, toString(q), sep="")

    df[[fullname]] <- num_sp_densitites
  }
  return(df)
}
```

The result of this can be seen in the first figure



Getting the variance shares

For getting the share of the varainces whose components happen with a frequency lower than λ_0, I run the following function. Similar to the spectral density function it takes as arguments the list of lambdas and the different values of q. Although all of the results of the integration have only real parts, I need to convert them since R still thinks of them as complex values and then the integration function does not work. At the end again a dataframe is being filled up since this way of storing the values makes plotting easier afterwards.

```
VR <- function(lambdas, q_list){
  df <- data.frame(lambdas)
  for (q in q_list){
    s_lambda0 <- vector()
    for (lambda in lambdas){
      s_current <- 2* integrate(spectral_dens, 0, lambda, q=q, Mod=1)$value
      s_lambda0 <- rbind(s_lambda0, s_current)
    }

    s_pi <- 2* integrate(spectral_dens, 0, pi, q=q, Mod=1)$value
    results <- s_lambda0/s_pi

    name <- "VR_q"
    fullname <- paste(name, toString(q), sep="")

    df[[fullname]] <- results
  }

  return(df)
```

```
}
```

Talking about plotting, $VR(\lambda_0; q)$ is