

# simulation\_\_week6

Tingyu Zhu

11/2/2020

```
library(tidyverse)
library(purrr)
library(ggplot2)
library(here)
devtools::load_all()
set.seed(1222)
```

## Commonly used kernel functions

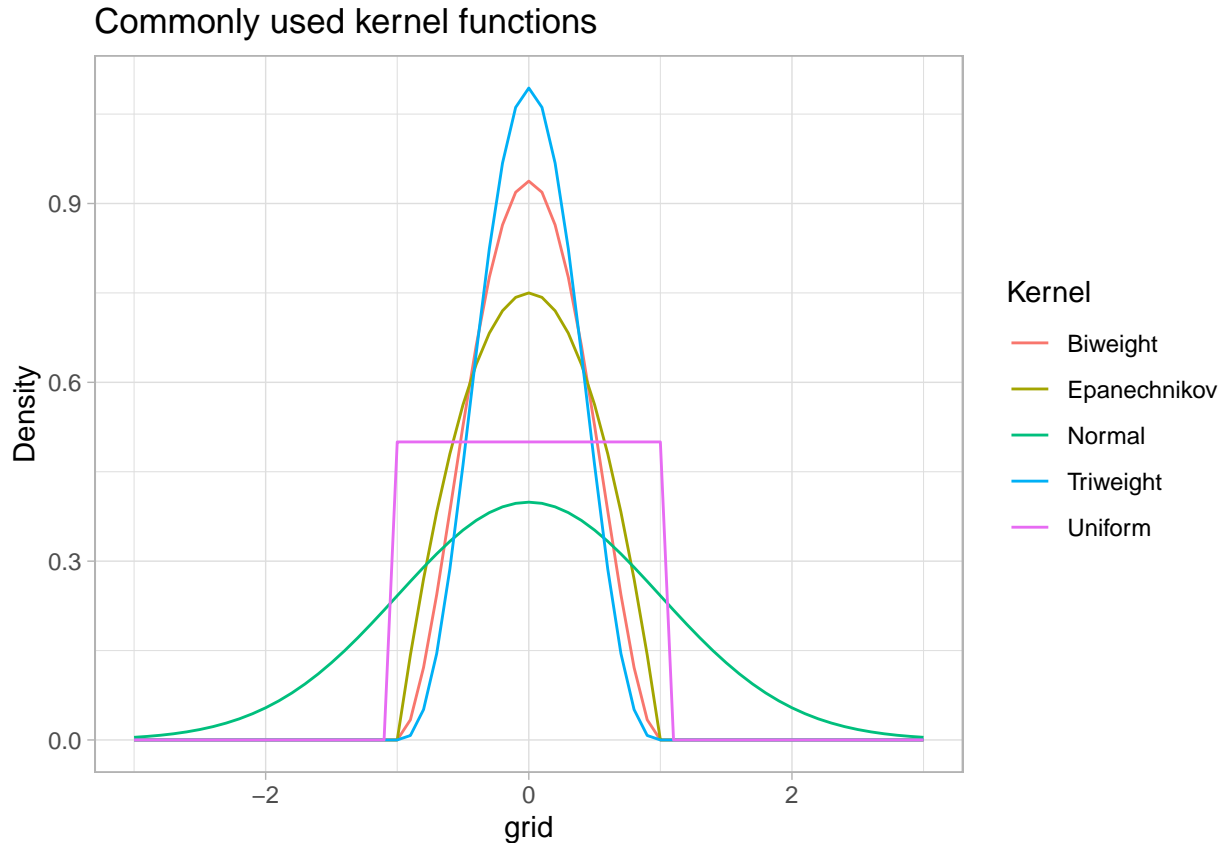
In the beta family kernel density,  $\lambda=0,1,2,3$  correspond to the uniform, the Epanechnikov, the biweight, and the triweight kernel functions, respectively.

```
kernel_beta <- function(x,lambda){(1-x^2)^lambda*(abs(x)<=1)/beta(lambda+1,0.5)}

kernel_comp <- tibble(
  grid = seq(-3,3,by=0.1),
  Normal = dnorm(grid),
  Uniform = kernel_beta(grid,0),
  Epanechnikov = kernel_beta(grid,1),
  Biweight = kernel_beta(grid,2),
  Triweight = kernel_beta(grid,3)
)

kernel_comp_long <- kernel_comp %>%
  pivot_longer(c(`Normal`, `Uniform`, `Epanechnikov`,
                `Biweight`, `Triweight`), names_to = "Kernel", values_to = "Density")

ggplot(data=kernel_comp_long) +
  geom_line(aes(x = grid, y = Density, color = Kernel)) +
  labs(title = "Commonly used kernel functions") +
  theme_light()
```



## Compute the MADE (Mean Absolute Deviation Errors)

The Mean Absolute Deviation Errors for  $\hat{f}(\cdot)$  is defined as

$$MADE = \frac{1}{n} \sum_{k=1}^n |\hat{f}(u_k) - f(u_k)|$$

, where  $\hat{f}(u_k)$  is the kernel estimate of  $f(u_k)$ , and  $\{u_k\}$  are the grid points taken to be arbitrary within the range of data.

When the sample is generated from the known distribution like exp, beta, gamma, etc., we can use corresponding built-in functions (`dexp()`, `dgamma`, etc.) to calculate the true densities of the grid points. If the sample is not from the familiar distributions, we can use the built-in function “`density()`” in R to get the value of  $f(u_k)$ s.

## Simulation

Explore the simulation results with different sample sizes & kernels

### Build simulation functions that can save the MADE.

Note: The following functions are similar to `kde_est_big`, `sim_big_made`, and `save_made` that are used in week 7, which are listed in the `kde_est_functions.R`. However, they are the first versions and are much simpler. Thus, we will not modify this part and keep those functions here as the records.

Here `kde_n_ker_est()` is used to simulate over different sample size and kernels for a given bandwidth  $h=1.06*100^{(-0.2)}$ , the output is the estimated densities.

`sim_made` is used to first generate combination of parameters, and then for each combination of parameters, using `kde_n_ker_est()` to generate the estimated densities, and the corresponding true densities. Then, `made()` is used to compute the MADE. The output is the MADEs under different combination of parameters.

```
# Define some functions for given h=1.06*100^(-0.2),

# simulate over different sample size and kernels
kde_n_ker_est <- function(x, n, ker, grid){
  # x large population
  # n sample sizes
  # ker name of kernels
  # grid grid points
  map2(.x = n, .y = ker,
    ~KDE_est(sample(x, .x, replace = FALSE),
      grid, h= 1.06*100^(-0.2), ker = .y)$f_est)
}

sim_made <- function(ns, kes, x, grid, true_f){

  # generates every combination of parameters (ns:kernels)
  simulation_params_big <- list(
    n = ns,
    kernel_type = kes)
  n_ker_est_big <- cross_df(simulation_params_big)

  n_ker_est_big <- n_ker_est_big %>%
  mutate(
    # for each combination of the paras, using the kde_n_ker_est()
    f_est = kde_n_ker_est(x, n, kernel_type, grid),
    f_true = map(1:length(n), ~true_f) # save the corresponding true density
  ) %>%
  mutate(
    MADE = map2_dbl(.x = f_est, .y = f_true, ~made(.x,.y)) # calculate the MADE
  )
  return(n_ker_est_big$MADE)
}
```

## Simulation

Generate samples from  $t(3)$  distribution. Then, using five types of kernel function and different size of samples that are drawn from the large population.

Simulation is repeated `n.sim` times.

```
# population
x <- rt(50000, df=3)

# simulation parameters
ns <- c(100, 500, 1000, 1500) # sample sizes
```

```

kers <- c("normal", "uniform", "epanech", "biweight", "triweight") # kernel functions
grid <- seq(-5,15, 0.1)

# true density
true_f <- dt(grid, df=3)

# simulation
n.sim <- 100 # number of replicates
sim_rlt <- map(1:n.sim, ~sim_made(ns, kers, x, grid,true_f))

```

## Save result

```

made_mat <- matrix(unlist(sim_rlt), nrow = length(ns)*length(kers) )

made_comapre <- tibble(
  n =rep(ns, time=length(kers)),
  kernel_type = rep(kers, each=length(ns)),
  made_mean = apply(made_mat, 1, mean),
  made_sd = apply(made_mat, 1, sd)
)
write_rds(made_mat , here("results", "week6-sim.rds"))
write_rds(made_comapre, here("results", "week6-sim-plot.rds"))

```

## Analysis

Results and plots are documented in `Summary_week6` in the `analysis` file.