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

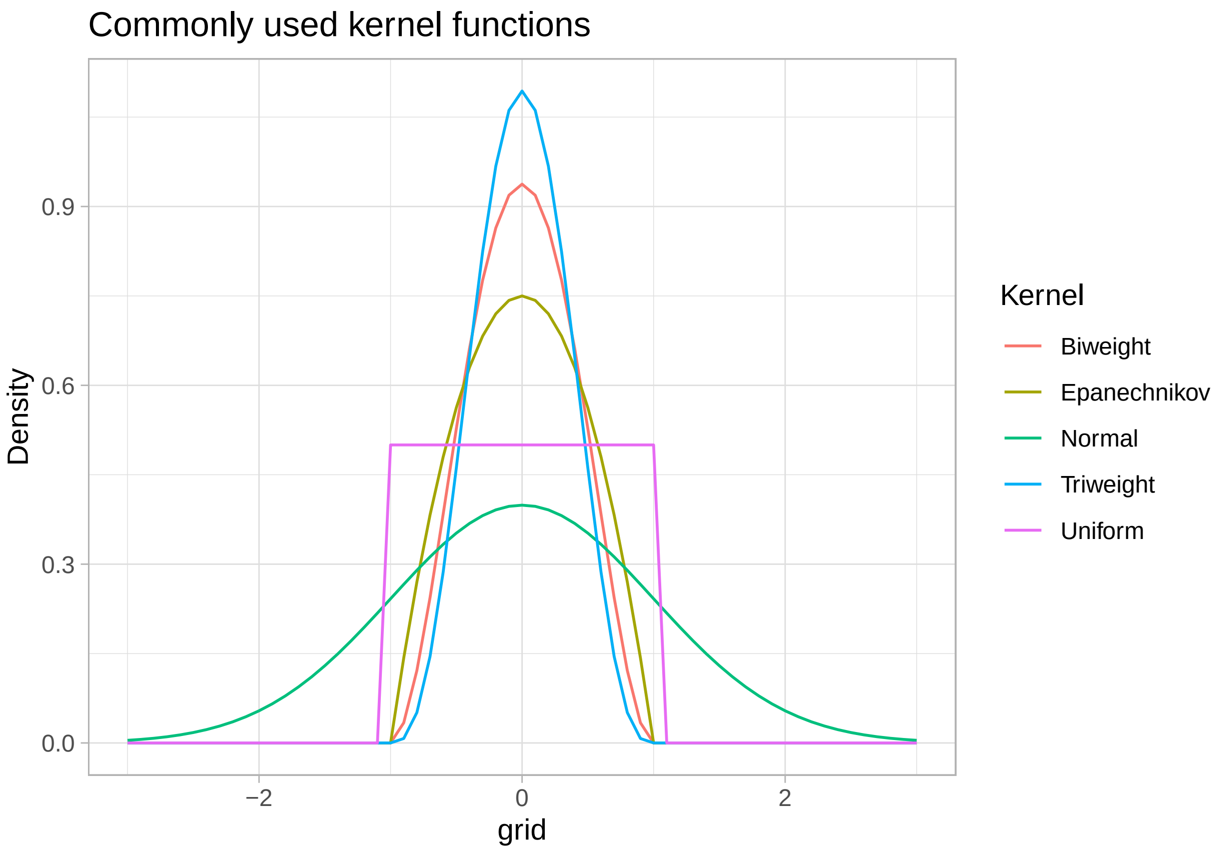
In the beta family kernel density, lambda=0,1,2,3 correspond to the uniform, the Epanech- nikov, the biweight, and the triweight kernel functions, respectively.

```
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 estimte of $f(u_k)$), and $\{u_k\}$ are the frid 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 calculated the true denstities of the grid points. If the sample is not from the familar distributions, we can use the built-in function "denstiy()" in R to get the value of $f(u_k)$)s.

```
made <- function(f_est,f_true) mean(abs(f_est-f_true))
```

Different sample sizes & kernels

```
# Define some functions for gievn h=bandwidth,
# 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))
}


sim_made <- function(ns, kes, x, grid, true_f){
  # generates every combination of parameters (ns:kers)
  simulation_params_big <- list(
  n = ns,
  kernel_type = kers)
  n_ker_est_big <- cross_df(simulation_params_big)

  n_ker_est_big <- n_ker_est_big %>%
  mutate(
    f_ests = kde_n_ker_est(x, n, kernel_type, grid),
    f_true = map(1:length(n), ~true_f)
  ) %>%
  mutate(
    MADE = map2_dbl(.x = f_ests, .y = f_true, ~made(.x,.y))
  )
  return(n_ker_est_big$MADE)
}
```

Simulation

Generate samples from t(3) distribution.

```
x <- rt(50000, df=15)
ns <- c(100, 500, 1000, 1500)
kers  <- c("normal", "uniform", "epanechnikov", "biweight", "triweight")
grid <- seq(-5,15, 0.1)
true_f <- dt(grid, df=15)

n.sim <- 100

sim_rlt <- map(1:n.sim, ~sim_made(ns, kers, x, grid,true_f))

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

made_comapre <- tibble(
  n =rep(ns, each=length(kers)),
  kernel_type = rep(kers, time=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"))
made_comapre
```

```
## # A tibble: 20 x 4
```

```
##         n kernel_type  made_mean  made_sd
##     <dbl> <chr>            <dbl>    <dbl>
##  1   100 normal         0.00731 0.00241
##  2   100 uniform        0.00472 0.00134
##  3   100 epanechnikov   0.00414 0.00108
##  4   100 biweight       0.00386 0.000753
##  5   100 triweight      0.00913 0.00215
##  6   500 normal         0.00441 0.00109
##  7   500 uniform        0.00316 0.000774
##  8   500 epanechnikov   0.00265 0.000692
##  9   500 biweight       0.0104  0.00230
## 10   500 triweight      0.00480 0.00105
## 11  1000 normal         0.00340 0.000832
## 12  1000 uniform        0.00284 0.000667
## 13  1000 epanechnikov   0.0118  0.00313
## 14  1000 biweight       0.00513 0.00101
## 15  1000 triweight      0.00363 0.000771
## 16  1500 normal         0.00294 0.000664
## 17  1500 uniform        0.0118  0.00265
## 18  1500 epanechnikov   0.00546 0.00116
## 19  1500 biweight       0.00376 0.000669
## 20  1500 triweight      0.00313 0.000620
```