simulation_week7

Tingyu Zhu

11/12/2020

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

Inspection about the bandwidth. Compare the performance under different methods of selecting the bandwidth.

Bulid functions that can simulate over different sample sizes & kernels & bandwidths.

```
kde_est_big <- function(x, n, ker, h, grid){
  # x: large population
  # n: sample sizes
  # ker: name of kernels
  # h: bandwidths
  # grid grid points
  l <- data.frame(n=n, ker=ker, h=h) %>%
    mutate( x = map(.x=n, ~sample(x, .x, replace = FALSE)))
  l$n <- NULL

  pmap(l, KDE_est, grid=grid)
}



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

  est_big <- cross_df(simulation_params_big)

  est_big <- est_big %>%
  mutate(
```

```r
    f_ests = kde_est_big(x, n, kernel_type, h, grid),
    f_true = map(1:length(n), ~true_f)
  ) %>%
  mutate(
    MADE = map2_dbl(.x = f_ests, .y = f_true, ~made(.x,.y))
  )
  return(est_big$MADE)
}
```

Simulation

```r
x <- rt(5000, df=15)

ns <- c(100, 500)
kers  <- c("normal","epanech", "uniform", "biweight", "triweight")
hs <- c(0.1,0.5,1)

grid <- seq(-5,15, 0.1)
true_f <- dt(grid, df=15)

n.sim <- 50

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

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


made_comapre <- cross_df(list(n = ns, kernel_type = kers, h = hs)) %>%
  mutate(made_mean=apply(made_mat_big, 1, mean),
         made_sd=apply(made_mat_big, 1, sd))

rownames(made_mat_big) <- paste(made_comapre$n,
                                made_comapre$kernel_type, made_comapre$h, sep = ", ")

write_rds(made_mat_big , here("results", "week7-sim-1.rds"))
made_comapre
```

```
## # A tibble: 30 x 5
##        n kernel_type     h made_mean made_sd
##    <dbl> <chr>       <dbl>     <dbl>   <dbl>
##  1   100 normal        0.1   0.0147  0.00220
##  2   500 normal        0.1   0.00698 0.00112
##  3   100 epanech       0.1   0.0220  0.00230
##  4   500 epanech       0.1   0.00990 0.00117
##  5   100 uniform       0.1   0.0202  0.00235
##  6   500 uniform       0.1   0.00911 0.00115
##  7   100 biweight      0.1   0.0243  0.00277
##  8   500 biweight      0.1   0.0108  0.00116
##  9   100 triweight     0.1   0.0251  0.00252
## 10   500 triweight     0.1   0.0120  0.00127
## # ... with 20 more rows
```

Types of bandwidth selectors

1. Normal reference bandwidth selector

The normal reference bandwidth selector is defined by:

$$\hat{h}_{opt} = \begin{cases} 1.06sn^{-1/5} \text{ for the Gaussian kernel} \\ 2.34sn^{-1/5} \text{ for the Epanechnikov kernel} \end{cases}$$

The normal reference bandwidth selector is only a simple rule of thumb. It is a good selector when the data are nearly Gaussian distributed. However, it can lead to over-smooth when the underlying distribution is asymmetric or multi-modal.

2. plug-in bandwidth selector

There are quite a few important techniques for selecting the bandwidth such as cross-validation (CV) and plug-in bandwidth selectors.

Function dpik() in the package KernSmooth in R selects a bandwidth for estimating the kernel density estimation using the plug-in method.

```r
library("KernSmooth")
```

```
## Warning: package 'KernSmooth' was built under R version 4.0.2
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```r
nx <- 5000
x <- rt(nx, df=15)

(h_opt_Gau <- 1.06*sd(x)*nx^(-0.2))
```

```
## [1] 0.2038434
```

```r
(h_opt_Epa <- 2.34*sd(x)*nx^(-0.2))
```

```
## [1] 0.4499939
```

```r
(h_plug_Gau <- dpik(x,kernel = "normal") )
```

```
## [1] 0.1835798
```

```r
(h_plug_Epa <- dpik(x,kernel = "epanech"))
```

```
## [1] 0.4064099
```

```r
(h_ran_Gau <- 0.5*nx^(-0.2))
```

```
## [1] 0.09102821
```

```r
(h_ran_Epa <- 0.5*nx^(-0.2))
```

```
## [1] 0.09102821
```

```r
ns <- c(4000)
kers  <- c("normal","epanech")
hs <- c(h_opt_Gau,h_plug_Gau,h_ran_Gau, h_opt_Epa, h_plug_Epa,h_ran_Epa)

grid <- seq(-5,15, 0.1)
true_f <- dt(grid, df=15)

n.sim <- 50

sim_rlt_bdwt <- map(1:n.sim, ~sim_big_made(x, ns, kers, hs, grid,true_f))

made_mat_bdwt <- matrix(unlist(sim_rlt_bdwt), nrow=length(ns)*length(hs)*length(kers))


made_comapre_bdwt <- cross_df(list(n = ns, kernel_type = kers, h = hs)) %>%
  mutate(made_mean=apply(made_mat_bdwt, 1, mean),
         made_sd=apply(made_mat_bdwt, 1, sd))

rownames(made_mat_bdwt) <- paste(made_comapre_bdwt$n,
                             made_comapre_bdwt$kernel_type,
                             round(made_comapre_bdwt$h,4), sep = ", ")

write_rds(made_mat_bdwt , here("results", "week7-sim-2.rds"))
made_comapre_bdwt
```

```
## # A tibble: 12 x 5
##        n kernel_type      h made_mean  made_sd
##    <dbl> <chr>        <dbl>     <dbl>    <dbl>
##  1  4000 normal      0.204    0.00181 0.000212
##  2  4000 epanech     0.204    0.00316 0.000267
##  3  4000 normal      0.184    0.00199 0.000194
##  4  4000 epanech     0.184    0.00334 0.000170
##  5  4000 normal      0.0910   0.00320 0.000244
##  6  4000 epanech     0.0910   0.00456 0.000290
##  7  4000 normal      0.450    0.00415 0.000256
##  8  4000 epanech     0.450    0.00184 0.000202
##  9  4000 normal      0.406    0.00342 0.000273
## 10  4000 epanech     0.406    0.00183 0.000201
## 11  4000 normal      0.0910   0.00314 0.000228
## 12  4000 epanech     0.0910   0.00466 0.000244
```

Results show that under this setting, the optimal rule of thumb bandwidths and plug-in bandwidths have the smallest MADE under their corresponding kernels.