STAT_680_HW2

Yudi Zhang, Wangqian Ju

10/12/2021

Problem 1

- The estimated h for the `paulsen` dataset is:

```
data("paulsen")
h.paulsen <- bi_search(paulsen$y, 1, 4, tol = 10^-6)
h.paulsen
```

```
## [1] 1.869225
```

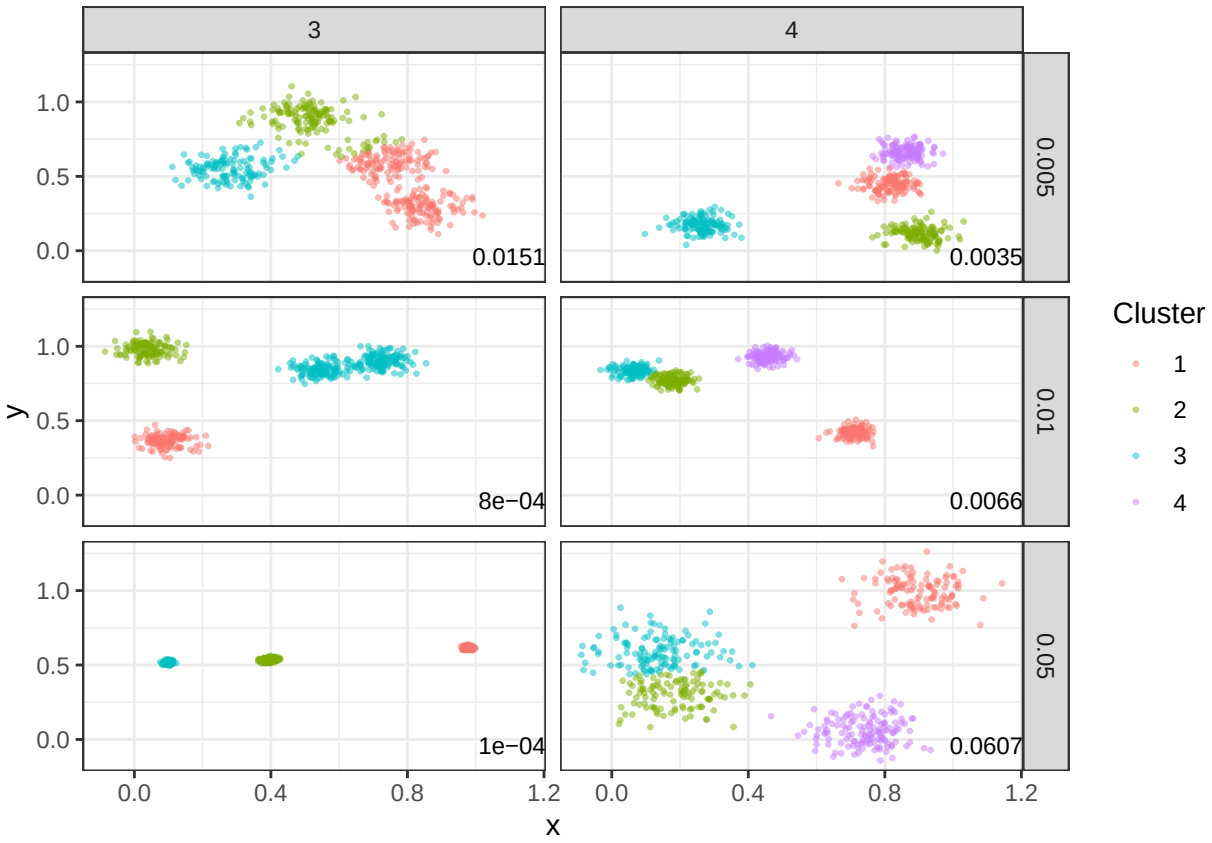
- the estimated p-value is:

```
boot.result <- boot(paulsen$y, boot_paulsen, R = 1000, h = h.paulsen)
mean(boot.result$t[,1])
```

```
## [1] 0.223
```

Problem 2

The probabilities of misclassification are labeled on the plots. According to the plots, we can see that the misclassification probability is large if two or more groups are close to each other. For $K = 3$, large values of $\ddot{\omega}$ make the group more concentrated, while for $K = 4$, $\ddot{\omega} = 0.05$ results in the most spread group distributions.



Problem 3

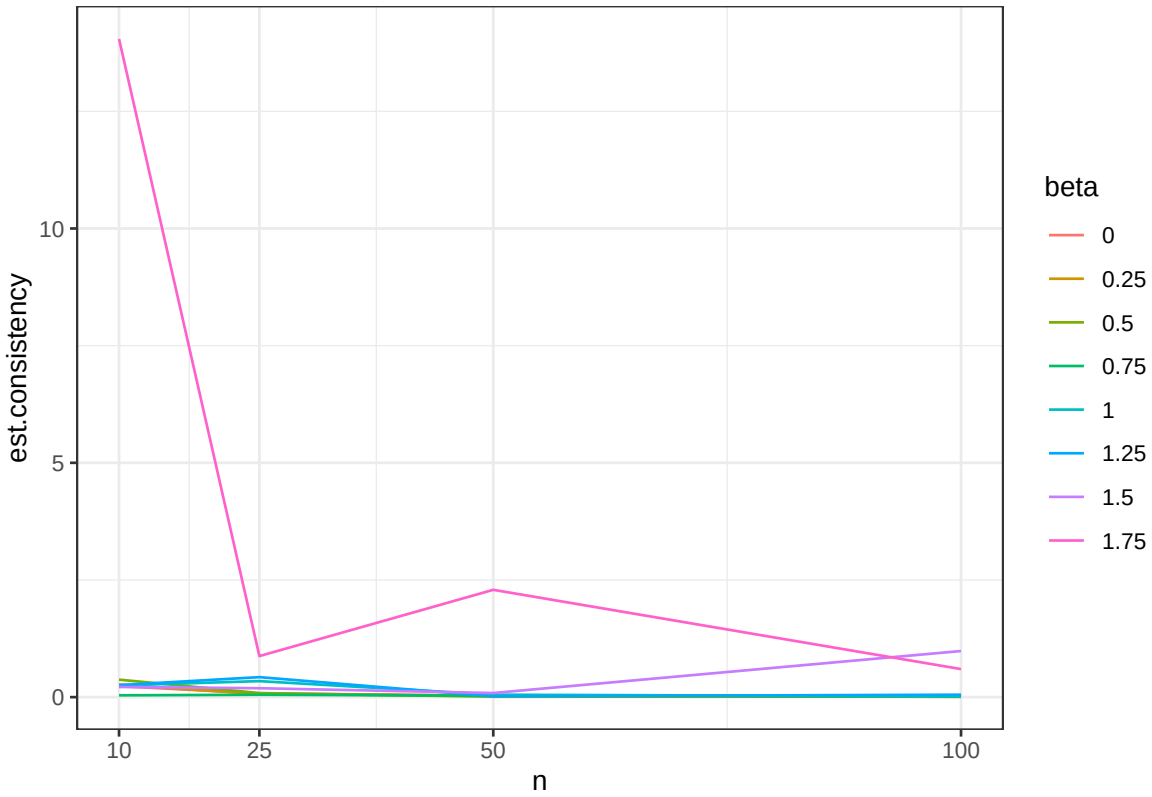
The following results show the consistency of (α, β) for different values of n and β.

```
##       n beta est.consistency
## 1    10 0.00     0.233879728
## 2    10 0.25     0.268228242
## 3    10 0.50     0.373500754
## 4    10 0.75     0.038784373
## 5    10 1.00     0.250828064
## 6    10 1.25     0.257261852
## 7    10 1.50     0.217207952
## 8    10 1.75    14.043985579
## 9    25 0.00     0.080108355
## 10   25 0.25     0.065346999
## 11   25 0.50     0.087013293
## 12   25 0.75     0.051183569
## 13   25 1.00     0.341642105
## 14   25 1.25     0.427147953
## 15   25 1.50     0.190146682
## 16   25 1.75     0.876200242
## 17   50 0.00     0.032720435
## 18   50 0.25     0.030058208
## 19   50 0.50     0.009884922
## 20   50 0.75     0.031520745
```

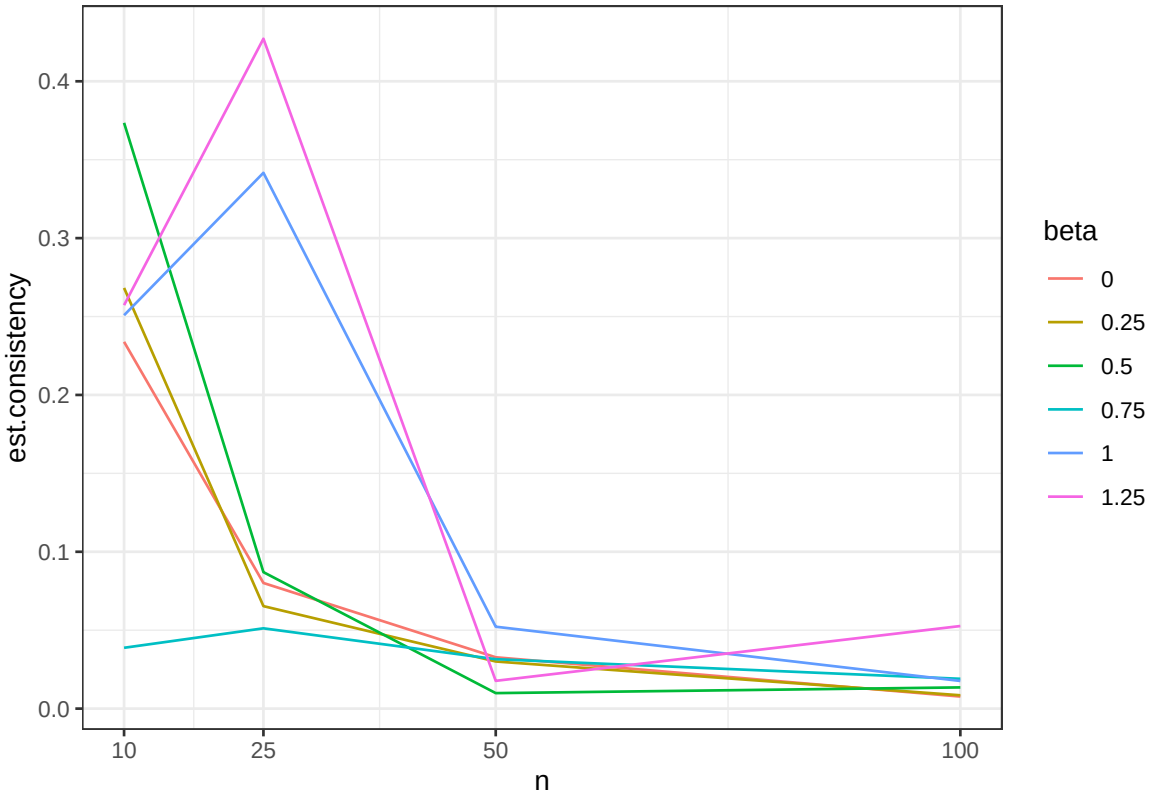
```
## 21  50 1.00     0.052222162
## 22  50 1.25     0.017700397
## 23  50 1.50     0.088199906
## 24  50 1.75     2.290428867
## 25 100 0.00     0.007689316
## 26 100 0.25     0.008503166
## 27 100 0.50     0.013524298
## 28 100 0.75     0.019005493
## 29 100 1.00     0.017640382
## 30 100 1.25     0.052663190
## 31 100 1.50     0.984557397
## 32 100 1.75     0.598832524
```

These results are summarized in the following plots:

- The first plot shows the estimated consistency for all pairs of n and β. It shows that when β increases and n is small, we tend to have very bad estimation. It makes sense because in this case, most samples have the same value, and we can not obtain a reasonable estimation.



- Then we removed the β's with large values from the plot. Again, we observe that it's hard to make estimation for large β values and small number observations. However, increasing the sample size improves the consistency of the estimators.



Appendix

R codes used for this homework:

```
library(tidyverse)
library(boot)
library(zoo)

data("paulsen")

# p1
k_dens <- function(x, h, data) {
  n <- length(data)
  sum(dnorm((x - data) / h)) / n
}

n_peak <- function(data, h = 1, x_length.out = 100, xmin = -10, xmax = 30) {
  x <- seq(xmin, xmax, length.out = x_length.out)
  y <- sapply(x, k_dens, h = h, data = data)
  check.uni <- sum(zoo::rollapply(y, 3, function(x) {x[2] == max(x)}))
  return(check.uni)
}

bi_search <- function(data, lower, upper, tol = 10^-3){
  ll <- n_peak(data, lower)
  rr <- n_peak(data, upper)
```

```r
  if(!(ll > 1 & rr == 1)) {
    stop("choose lower and upper again")
  }

  repeat{
    mid <- mean(c(lower, upper))
    mm <- n_peak(data, mid)

    if(mm > 1) {
      lower <- mid
    } else {
      upper <- mid
    }

    if(abs(lower - upper) < tol) break
  }

  return(upper)
}
data("paulsen")
h.paulsen <- bi_search(paulsen$y, 1, 4, tol = 10^-6)
h.paulsen

boot_paulsen <- function(data, i, h) {
    x_star <- data[i] + h * rnorm(length(data))

    n_peak(x_star, h = h) > 1
}

boot.result <- boot(paulsen$y, boot_paulsen, R = 1000, h = h.paulsen)
mean(boot.result$t[,1])

# p2
# library(MixSim)
library(tidyverse)
get_n_rs_tbl <- function(tt, k, center) {
  cc <- combn(k, 2)
  nrstbl <- tibble(cluster1 = c(cc[1,], cc[2,]),
                   cluster2 = c(cc[2,], cc[1,]))
  nrstbl$l_center <- lapply(nrstbl$cluster2, function(idx) center[idx,])
  nrstbl$Xi <- lapply(nrstbl$cluster1, function(cluster) {
    tt$data[tt$cluster == cluster, ]
  })
  nrstbl <- nrstbl %>% mutate(
    n_rs_cluster = purrr::map2(l_center, Xi, function(cent, dd) {
      sapply(1:nrow(dd), function(idx) {
        sqrt(sum((dd[idx, ] - cent)^2))
      })
    })
  ) %>% unnest(n_rs_cluster)
  return(nrstbl)
}
```

```r
grid <- expand.grid(omega = c(0.005, 0.01, 0.05), K_value = 3:4)
grid <- as_tibble(grid)

grid <- grid %>% mutate(
  data_result = purrr::map2(omega, K_value, function(oo, kk) {
    Q <- MixSim::MixGOM(goMega=oo, p = 2, hom = T, sph = T, K = 4)
    A <- MixSim::simdataset(n = 500, Pi = Q$Pi, Mu = Q$Mu, S = Q$S, n.out = 0, int = c(0, 1))

    dd <- A$X
    k.rr <- kmeans(dd, centers = kk, iter.max = 20, nstart = 10)

    tt <- tibble(data = dd, cluster = k.rr$cluster)
    tt <- tt %>% left_join(tibble(center = k.rr$centers, cluster = 1:nrow(k.rr$centers)),
                           by = "cluster")
    tt$n_rs <- sapply(1:nrow(tt), function(idx) {
      x <- tt$data[idx, ]
      center <- tt$center[idx, ]

      sqrt(sum((x - center)^2))
    })

    list(tt = tt, k.rr = k.rr)
  }),
  nrstbl_result = purrr::map2(K_value, data_result, function(kk, dd) {
    get_n_rs_tbl(dd$tt, k = kk, dd$k.rr$centers)
  })
)

saveRDS(grid, "homework2/grid.rds")

# grid <- readRDS("grid.rds")
# k_result <- lapply(1:nrow(grid), function(idx) {
#   SynClustR::kcdf(grid$data_result[[idx]]$tt$n_rs, xgrid = grid$nrstbl_result[[idx]]$n_rs_cluster)
# })
# saveRDS(k_result, "k_result.rds")

k_result <- readRDS("~/STAT_680/STAT680/homework2/k_result.rds")

grid$nrstbl_result <- lapply(1:nrow(grid), function(idx) {
  grid$nrstbl_result[[idx]]$k_est <- k_result[[idx]]$Fhat
  grid$nrstbl_result[[idx]]
})

grid.r <- grid %>% mutate(
  p_misclass = purrr::map_dbl(nrstbl_result, function(nn) {
    nn %>%
      nest(k_result = n_rs_cluster:k_est) %>%
      mutate(
        p_est = purrr::map_dbl(k_result, function(kk) {
          mean(kk$k_est)
        }),
        mis_p = 1 - p_est
      ) %>% { sum(.$mis_p) }
```

```
  })
)

check.idx <- 3
grid.r
grid.r$data_result[[check.idx]]$tt$data %>% { plot(.[,1], .[,2]) }
grid.r$data_result[[check.idx]]$k.rr$centers %>% { points(.[,1], .[,2], col = "red") }

grid.r$plot_data <- lapply(grid.r$data_result, function(dd) {
  xx <- dd[[1]]
  xx$x <- xx$data[,1]
  xx$y <- xx$data[,2]
  xx %>% select(x, y, cluster)
})

dat_text <- grid.r %>% select(omega, K_value, p_misclass)

p <- grid.r %>% select(omega, K_value, plot_data, p_misclass) %>%
  unnest(plot_data) %>%
  ggplot() +
  geom_point(aes(x = x, y = y, color = factor(cluster)),
             alpha = 0.5, size = 0.5) +
  facet_grid(factor(omega)~factor(K_value)) +
  theme_bw() +
  labs(color = "Cluster") +
  geom_text(data = dat_text,
            mapping = aes(x = Inf, y = -Inf, label = round(p_misclass, 4)),
            hjust = 1, vjust = -1, size = 3)
p

# p3
library(bayesImageS)

n <- 50
beta <- 1
mask <- matrix(1,n,n) # basically the grid
neigh <- getNeighbors(mask, c(2,2,0,0)) # the neighborhood structure
# 1st order neighborhood in 2D
block <- getBlocks(mask, 2)
k <- 2 #(number of classes, k=2 makes Potts' to be an Ising model)
result <- swNoData(beta = beta,k = k,neigh = neigh, block = block)
z <- matrix(max.col(result$z)[1:nrow(neigh)], nrow=nrow(mask))
z <- z - 1
# the output is the last realization, since we only have two
# classes,  one of the $k=2$ columns is adequate.

find_neighbor <- function(z, i, j) {
  dim.z <- dim(z)

  if(i > dim.z[1] | j > dim.z[2]) return(NULL)

  count <- 0
```

```
  neigh.idx <- list()
  if(i - 1 >= 1) {
    count <- count + 1
    neigh.idx[[count]] <- c(i-1, j)
  }

  if(i + 1 <= dim.z[1]) {
    count <- count + 1
    neigh.idx[[count]] <- c(i + 1, j)
  }

  if(j - 1 >= 1) {
    count <- count + 1
    neigh.idx[[count]] <- c(i, j - 1)
  }

  if(j + 1 <= dim.z[2]) {
    count <- count + 1
    neigh.idx[[count]] <- c(i, j + 1)
  }

  return(neigh.idx)

}

find_neighbor(z, 49, 49)

compute_log_cond_prob <- function(z, i, j, alpha = 0, beta = 0) {
  xi <- z[i,j]
  xi_neighbor <- sapply(find_neighbor(z, i, j), function(nn) {
    z[nn[1], nn[2]]
  })

  beta_coef <- sum(xi == xi_neighbor)
  num <- xi * alpha + beta_coef * beta
  dem <- log(exp(num) + exp( (1-xi)*alpha + (length(xi_neighbor) - beta_coef)*beta ))

  return(num - dem)
}

compute_neg_log_prof_llh <- function(par, z) {
  alpha <- par[1]
  beta <- par[2]

  result <- sapply(1:nrow(z), function(i) {
    tt <- sapply(1:ncol(z), function(j) {
      compute_log_cond_prob(z, i, j, alpha, beta)
    })
    sum(tt)
  })

  return(-sum(result))
}
```

```r
n <- 100
beta <- 1
mask <- matrix(1,n,n) # basically the grid
neigh <- getNeighbors(mask, c(2,2,0,0)) # the neighborhood structure
# 1st order neighborhood in 2D
block <- getBlocks(mask, 2)
k <- 2 #(number of classes, k=2 makes Potts' to be an Ising model)
result <- swNoData(beta = beta,k = k,neigh = neigh, block = block)
z <- matrix(max.col(result$z)[1:nrow(neigh)], nrow=nrow(mask))
z <- z - 1

opt.rr <- optim(par = c(0, 0.5), compute_neg_log_prof_llh, z = z)
eval.consistency <- sqrt(sum((opt.rr$par - c(0, beta))^2))

grid <- expand.grid(beta = c(0, .25, .5, .75, 1, 1.25, 1.5, 1.75),
                    n = c(10, 25, 50, 100))
grid <- grid %>% mutate(
  sim.data = purrr::map2(beta,n, function(bb, nn) {
    mask <- matrix(1,nn,nn) # basically the grid
    neigh <- getNeighbors(mask, c(2,2,0,0)) # the neighborhood structure
    # 1st order neighborhood in 2D
    block <- getBlocks(mask, 2)
    k <- 2 #(number of classes, k=2 makes Potts' to be an Ising model)
    result <- swNoData(beta = bb,k = k,neigh = neigh, block = block)
    z <- matrix(max.col(result$z)[1:nrow(neigh)], nrow=nrow(mask))
    z <- z - 1
    z
  })
)

grid.r3 <- grid %>% mutate(
  est.par = purrr::map(sim.data, function(z) {
    opt.rr <- optim(par = c(0, 0.75), compute_neg_log_prof_llh, z = z)
    opt.rr$par
  })
)

grid.r3 <- grid.r3 %>% mutate(
  est.consistency = purrr::map2_dbl(beta, est.par, function(bb, pp) {
    sqrt(sum((pp - c(0, bb))^2))
  })
)

opt.rr <- optim(par = c(0, 1.75), compute_neg_log_prof_llh, z = grid.r3$sim.data[[8]])
eval.consistency <- sqrt(sum((opt.rr$par - c(0, 1.75))^2))

saveRDS(grid.r3, "homework2/result_p3.rds")

p1 <- grid.r3 %>% ggplot() +
  geom_line(aes(x = n, y = est.consistency,
                color = factor(beta))) +
  labs(color = "beta") +
  scale_x_continuous(breaks = unique(grid.r3$n)) +
```

```
  theme_bw()

p2 <- grid.r3 %>% filter(beta < 1.5) %>%
  ggplot() +
  geom_line(aes(x = n, y = est.consistency,
                color = factor(beta))) +
  labs(color = "beta") +
  scale_x_continuous(breaks = unique(grid.r3$n)) +
  theme_bw()
```