Ratio Cut Examples

From Luxburg's tutorial[1], we saw that in the case for $k = 2$, if we construct a cluster assignment vector $\overrightarrow{f}$ as:

$$f_i = \begin{cases} \sqrt{k/(n-k)}, & x_i \in A \\ -\sqrt{(n-k)/k}, & x_i \in A^c \end{cases}$$

Then the "best" clustering is the one that minimizes $\text{Tr}(f^T L f)$, where L is the unnormalized graph Laplacian. Then we saw that the solution to a relaxed version of this problem:

$$\min_{\forall f} \text{Tr}(f^T L F)$$

is simply the Fiedler vector. The Fiedler vector itself does not provide a clustering, but we can use some clustering method on it (e.g., k-means).

Then we showed that performing k-means on the Fiedler vector with $k = 2$ is equivalent to clustering based on whether they are closer to $\sqrt{k/(n-k)}$ or $-\sqrt{(n-k)/k}$, i.e., minimizing:

$$W(k) = \sum_{i=1}^{k} (x_i - m_1)^2 + \sum_{i=k+1}^{n} (x_i - m_2)^2$$

where m_1 and m_2 are k-means centers, results in the same clustering as assigning clusters by minimizing

$$R(k) = \sum_{i=1}^{k} \left(x_i + \sqrt{\frac{n-k}{k}} \right)^2 + \sum_{i=k+1}^{n} \left(x_i - \sqrt{\frac{k}{n-k}} \right)^2$$

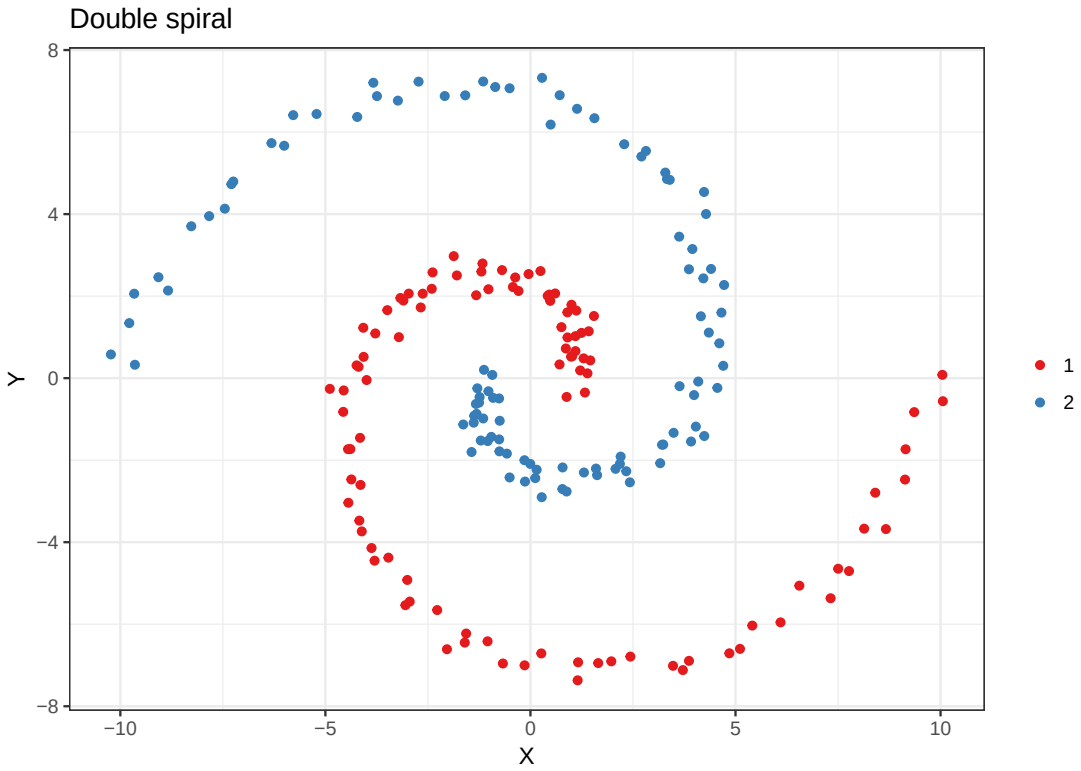
where the indices of x are the ordering of x_i's$.[2]

We can see this in action with a double spiral:

```
library(ggplot2)
import::from(magrittr, `%>%`, `%<>%`)
theme_set(theme_bw())
import::from(psych, tr)
library(qgraph)

source('http://pages.iu.edu/~mtrosset/Courses/675/manifold.r')

construct.H <- function(clustering) {
  # this function is limited to nonempty clusters
  # e.g., if there are 3 clusters, they must be assigned as 1, 2, 3
  clusters <- unique(clustering)
  if (length(clusters) != max(clustering)) {
    stop(simpleError('there are empty clusters'))
  }
  if (min(clustering) < 1) {
    stop(simpleError('cluster indexing starts at 1'))
  }
```

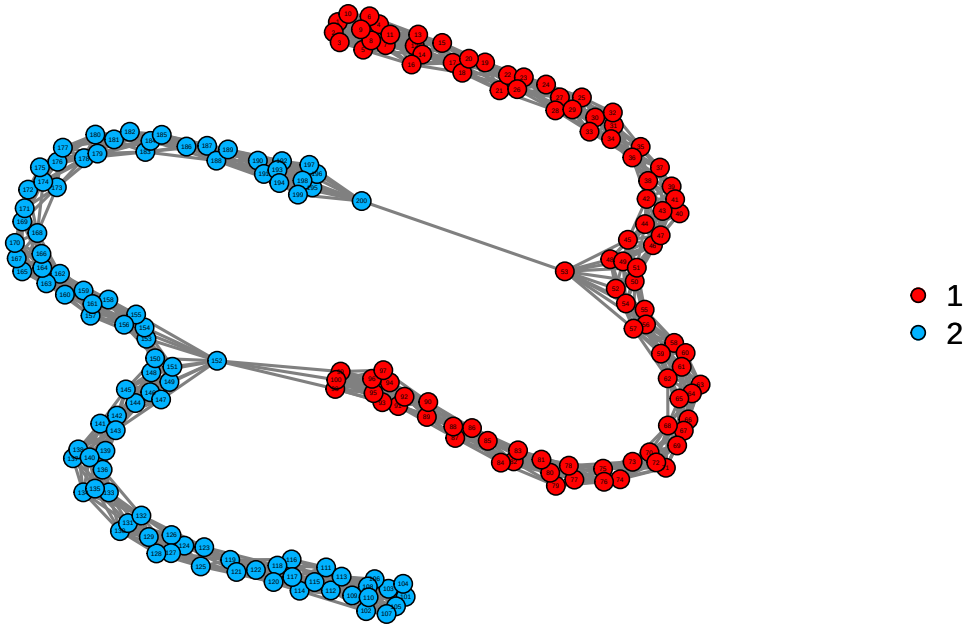
[1] https://arxiv.org/abs/0711.0189
[2] http://pages.iu.edu/~mtrosset/Courses/675/notes.pdf

```r
  # find |A_k|
  cluster.sizes <- sapply(clusters, function(i) {
    length(clustering[clustering == i])
  })

  # construct H
  H <- sapply(clustering, function(i) {
    h <- rep(0, length(clusters))
    h[i] <- 1 / sqrt(cluster.sizes[i])
    return(h)
  }) %>%
    t()

  return(H)
}

# parameters
set.seed(112358)
s <- 2 ** 5
eps <- 2 ** -2
k <- 10  # for constructing the knn graph
K <- 2  # number of clusters
cols2 <- colorRampPalette(c('blue', 'white', 'red'))(256)
rad.max <- 10
ang.max <- 2 * pi
angles <- seq(0, ang.max, length.out = 100)
radii <- seq(1, sqrt(rad.max), length.out = 100) ** 2
N <- 100  # number of times to try k-means clustering

# data
spiral.df <- dplyr::data_frame(X = radii * cos(angles),
                               Y = radii * sin(angles))
spiral.df <- dplyr::data_frame(X = radii * cos(angles),
                               Y = radii * sin(angles))
neg.spiral.df <- dplyr::mutate(spiral.df,
                               X = -X, Y = -Y,
                               id = '2')
spiral.df %<>%
  dplyr::mutate(id = '1') %>%
  dplyr::bind_rows(neg.spiral.df) %>%
  dplyr::mutate(X = X + rnorm(n = n(), sd = eps),
                Y = Y + rnorm(n = n(), sd = eps))

n <- nrow(spiral.df)  # number of vertices

ggplot(spiral.df) +
  geom_point(aes(x = X, y = Y, colour = id)) +
  scale_colour_brewer(palette = 'Set1') +
  labs(colour = NULL, title = 'Double spiral') +
  coord_fixed()
```

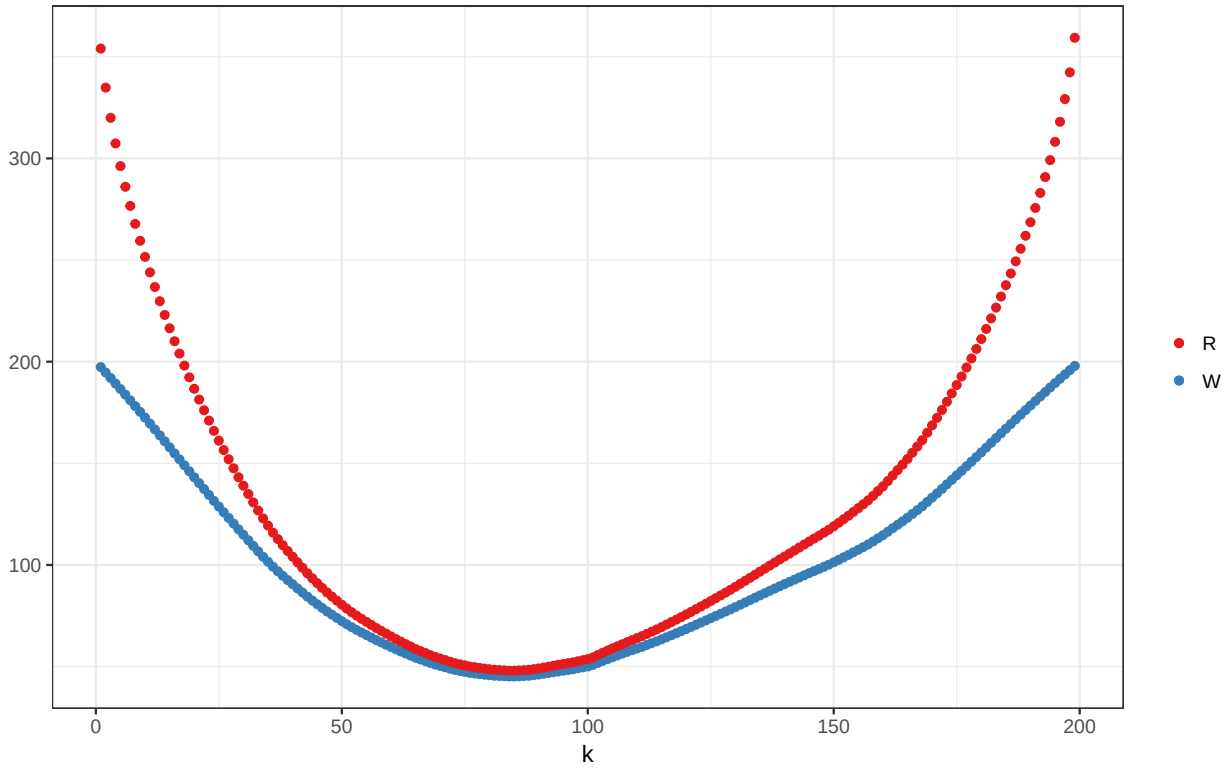



```
# construct W
W <- spiral.df %>%
  dplyr::select(X, Y) %>%
  as.matrix() %>%
  mds.edm1() %>%
  graph.knn(k) %>%
  graph.adj()

qgraph(W, groups = spiral.df$id,
       title = 'kNN nearest graph of the double spiral',
       layout = 'spring')
```

kNN nearest graph of the double spiral

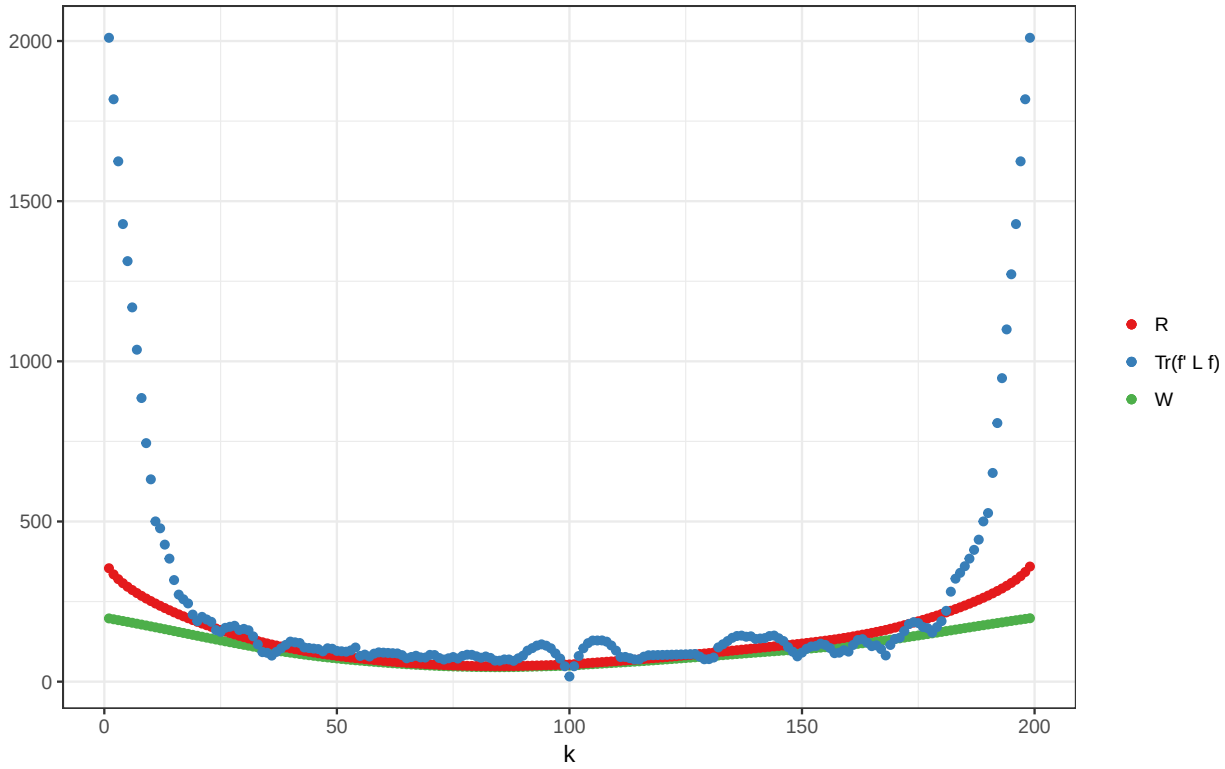



```
# graph laplacian
L <- graph.laplacian(W)

# fiedler vector
fiedler.vec <- eigen(L)$vectors[, n - 1]
x <- sort(fiedler.vec) * sqrt(n)
ind <- order(fiedler.vec)

# W(k) and R(k)
W.and.R.df <- lapply(seq(n - 1), function(k) {
  clust.1 <- x[seq(k)]
  clust.2 <- x[seq(k + 1, n)]

  m1 <- mean(clust.1)
  m2 <- mean(clust.2)

  W <- sum((clust.1 - m1) ** 2) + sum((clust.2 - m2) ** 2)
  R <- sum((clust.1 + sqrt((n - k) / k)) ^ 2) +
    sum((clust.2 - sqrt(k / (n - k))) ^ 2)

  dplyr::data_frame(k = k, W = W, R = R)
}) %>%
  dplyr::bind_rows()

ggplot(W.and.R.df) +
  geom_point(aes(x = k, y = W, colour = 'W')) +
  geom_point(aes(x = k, y = R, colour = 'R')) +
  labs(colour = NULL, y = NULL) +
  scale_colour_brewer(palette = 'Set1')
```



However, neither W nor R are the metric we use to determine the best clustering according to the ratio cut method. W is the best k-means clustering of the Fiedler vector while R finds the best cluster assignment vector $\overrightarrow{f}$ based on its proximity to the Fiedler vector. Instead, we should be looking at $\mathrm{Tr}(f^T L f)$:

```
ratio.cut.metric <- sapply(seq(n - 1), function(k) {
  f <- c(rep(-sqrt((n - k) / k), k),
         rep(sqrt(k / (n - k)), n - k))
  tr(t(f) %*% L[ind, ind] %*% f)
})

W.and.R.df$ratio.cut <- ratio.cut.metric

ggplot(W.and.R.df) +
  geom_point(aes(x = k, y = W, colour = 'W')) +
  geom_point(aes(x = k, y = R, colour = 'R')) +
  geom_point(aes(x = k, y = ratio.cut, colour = 'Tr(f\' L f)')) +
  labs(colour = NULL, y = NULL) +
  scale_colour_brewer(palette = 'Set1')
```



TO DO:

- Relate the ratio cut metric to either R or W
- Explore other ways of recovering f from the Fiedler vector

Luxburg's tutorial extends this to arbitrary k (including $k = 2$ by constructing the cluster assignment matrix $H \in \mathbb{R}^{n \times k}$ as follows:

$$h_{ij} = \begin{cases} 1/\sqrt{|A_j|}, & x_i \in A_j \\ 0, & x_i \notin A_j \end{cases}$$

then choosing such an H that minimizes $\text{Tr}(H^T L H)$. Luxburg then relaxes this problem for arbitrary H and finds that:

$$argmin \text{Tr}(H^T L H) = \begin{bmatrix} v_0 & \cdots & v_{k-1} \end{bmatrix}$$

where v_i's are the eigenvectors of L in increasing order of their corresponding eigenvalues. Then we can come up with an analogue for R in the $k = 2$ case:

$$R(H) = ||H_* - H||_F^2$$

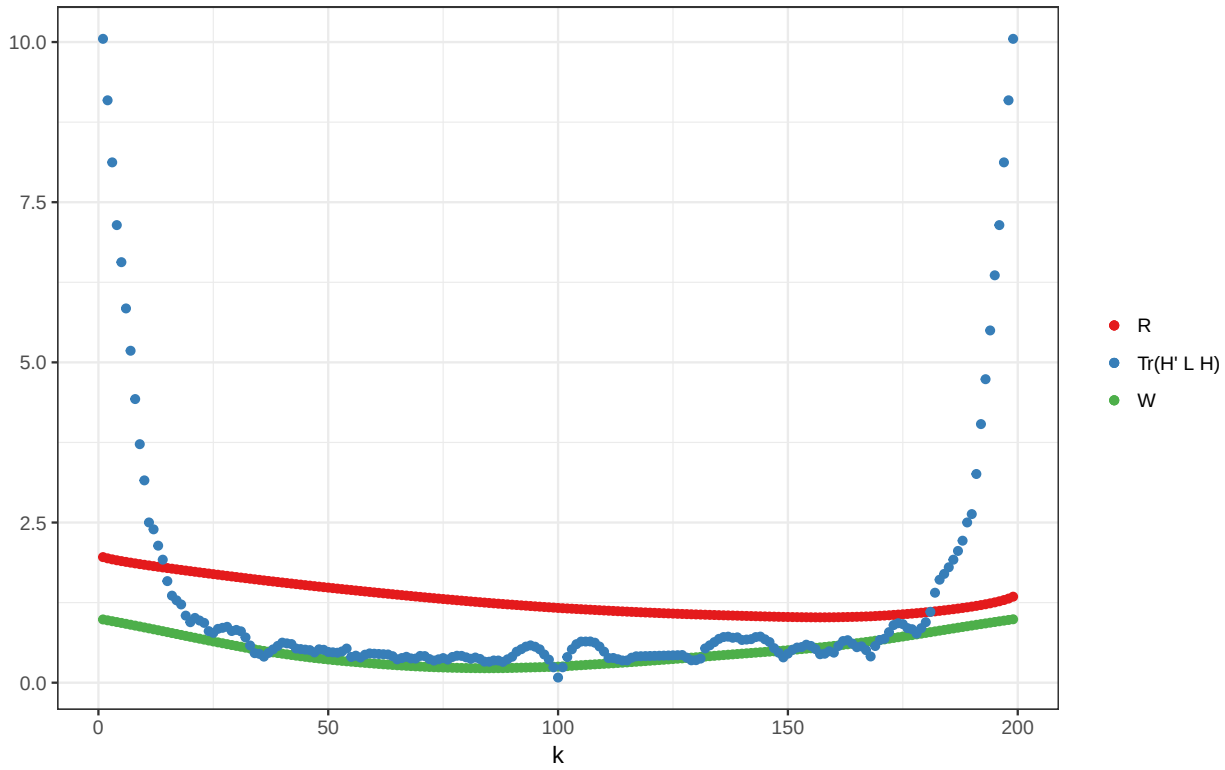
where H_* is the solution to the relaxed optimization problem and H is a matrix constructed from a particular clustering. And of course we can compute W for k-means clustering.

$v_0 = \pm \frac{1}{\sqrt{n}} \mathbb{1}$, so for $k = 2$, we can come up with an analogous setup as for the strictly $k = 2$ case.

```
two.df <- eigen(L)$vectors[, seq(n, n - 1)] %>%
  as.data.frame() %>%
  dplyr::mutate(V1 = 1 / sqrt(n) * sign(V1)) %>%
```

```
  dplyr::mutate(ind = seq(n())) %>%
  dplyr::arrange(V2)

H <- two.df %>%
  dplyr::select(V1, V2) %>%
  as.matrix()

ind <- two.df$ind

k2.df <- lapply(seq(n - 1), function(k) {
  clust.1 <- H[seq(k), ]
  clust.2 <- H[seq(k + 1, n), ]

  if (k == 1) {
    W1 <- 0
  } else {
    m1 <- apply(clust.1, 2, mean)
    W1 <- clust.1 %>%
      apply(1, function(h) h - m1) %>%
      t() %>%
      apply(2, function(h) sum(h ** 2)) %>%
      sum()
  }
  if (k == (n - 1)) {
    W2 <- 0
  } else {
    m2 <- apply(clust.2, 2, mean)
    W2 <- clust.2 %>%
      apply(1, function(h) h - m2) %>%
      t() %>%
      apply(2, function(h) sum(h ** 2)) %>%
      sum()
  }
  W <- W1 + W2

  clustering <- c(rep(1, k),
                  rep(2, n - k))
  H.clust <- construct.H(clustering)

  R <- norm(H.clust - H, type = 'F')

  ratio.cut <- tr(t(H.clust) %*% L[ind, ind] %*% H.clust)

  dplyr::data_frame(
    k = k,
    W = W,
    R = R,
    ratio.cut = ratio.cut
  )
}) %>%
  dplyr::bind_rows()

ggplot(k2.df) +
```

```
  geom_point(aes(x = k, y = W, colour = 'W')) +
  geom_point(aes(x = k, y = R, colour = 'R')) +
  geom_point(aes(x = k, y = ratio.cut, colour = 'Tr(H\' L H)')) +
  labs(y = NULL, colour = NULL) +
  scale_colour_brewer(palette = 'Set1')
```



And in this case, W, R, and $\mathrm{Tr}(H^T LH)$ all give different solutions.

TO DO:

- Reconcile the three methods somehow
- Explore other ways to recover H from H_*