Spectral Clustering

Much of this uses information from A Tutorial on Spectral Clustering by Ulrike von Luxburg and Proximity in Statistical Machine Learning by Michael Trosset.

The Ratio Cut Problem

Ratio Cut for $k = 2$

It can be shown that in the relaxed case for $k = 2$, 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, as perscribed by Luxburg results in the same clustering as by assigning clusters by minimizing

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

in $\mathbb{R}^1$ and where x_is are ordered, i.e., $x_i \leq \cdots \leq x_n$. We also constrain the problem to $\sum_i^n x_i = 0$ and $\sum_i^n x_i^2 = n$.

The k-means centers in $\mathbb{R}$ are

$$m_1 = \frac{1}{k}\sum_{i=1}^{k}x_i$$

$$m_2 = \frac{1}{n-k}\sum_{i=k+1}^{n}x_i$$

Note that m_1 and m_2 are functions of k.

Numerical results

Using an arbitrary vector $\overrightarrow{x} \in \mathbb{R}^n$ such that $\sum_i x_i = 0$ and $|\overrightarrow{x}|_2^2 = n$:

```r
# packages
library(ggplot2)
import::from(magrittr, `%>%`, `%<>%`)
theme_set(theme_bw())

# --- functions --- #

normalize <- function(x) {
  y <- x - mean(x)
  z <- y / sqrt(mean(y ** 2))
```

```r
    return(z)
}

k.means <- function(x) {
  x <- sort(x)
  n <- length(x)

  sapply(seq(n - 1), function(k) {
    m1 <- 1 / k * sum(x[seq(k)])
    m2 <- 1 / (n - k) * sum(x[seq(k + 1, n)])

    W <- sum((x[seq(k)] - m1)^2) + sum((x[seq(k + 1, n)] - m2)^2)
    return(W)
  })
}

ratio.cut <- function(x) {
  x <- sort(x)
  n <- length(x)

  sapply(seq(n - 1), function(k) {
    Ac <- sqrt((n - k) / k)
    A <- sqrt(k / (n - k))

    R <- sum((x[seq(k)] + Ac)^2) + sum((x[seq(k + 1, n)] - A)^2)
    return(R)
  })
}

# generate random data
n <- 100
k <- 4
x <- c(rnorm(n / k, -1),
       rnorm(n / k, 0),
       rnorm(n / k, 1),
       rnorm(n / k, 3))
z <- normalize(x)
plot(density(z))
```
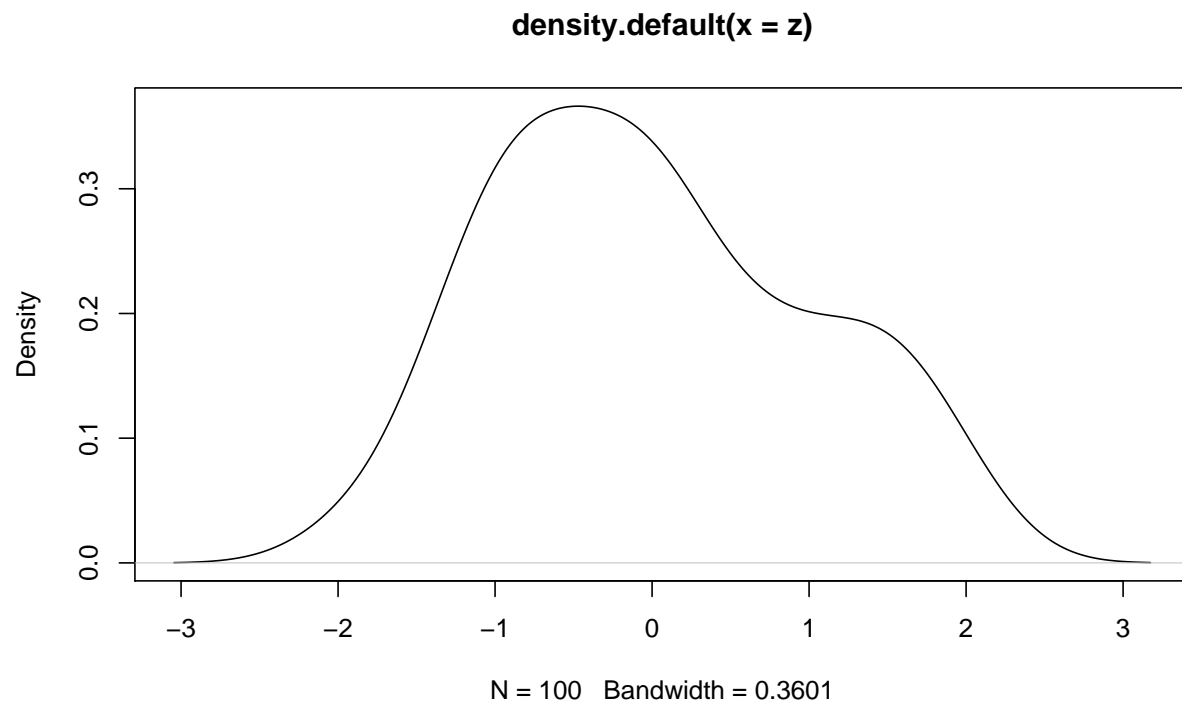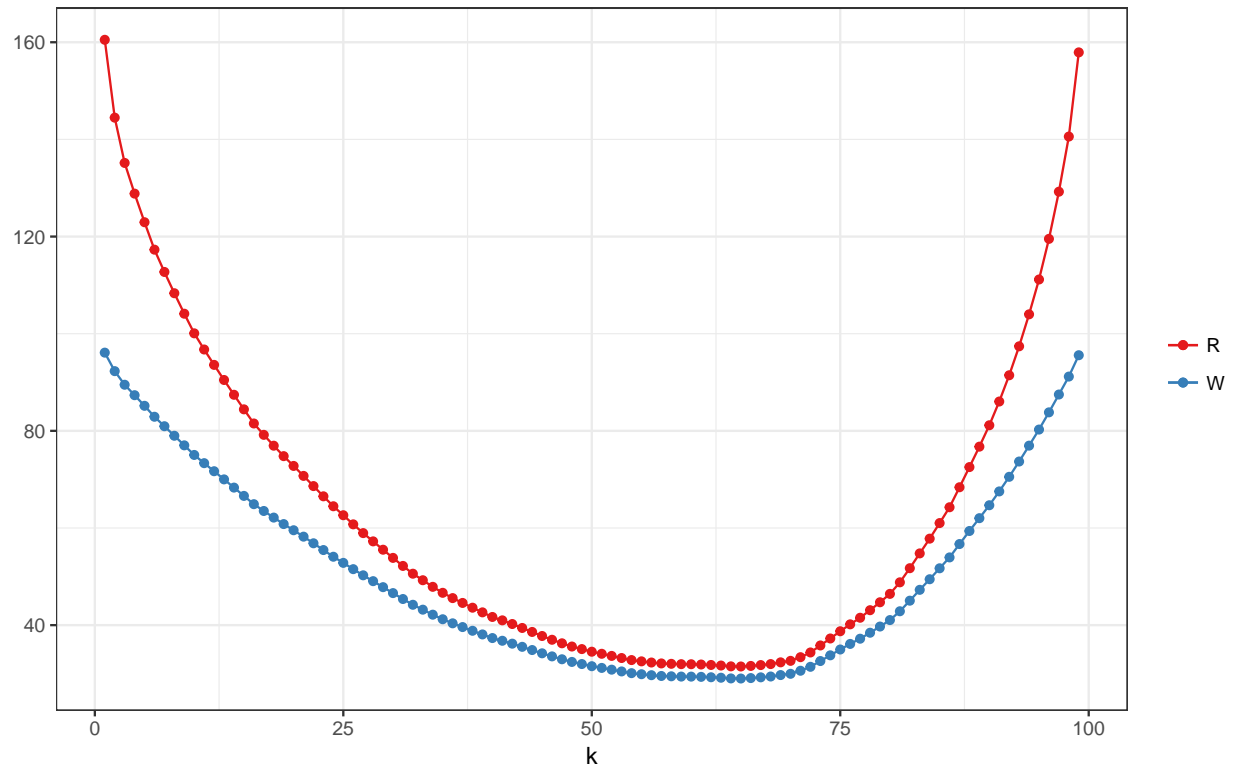
density.default(x = z)
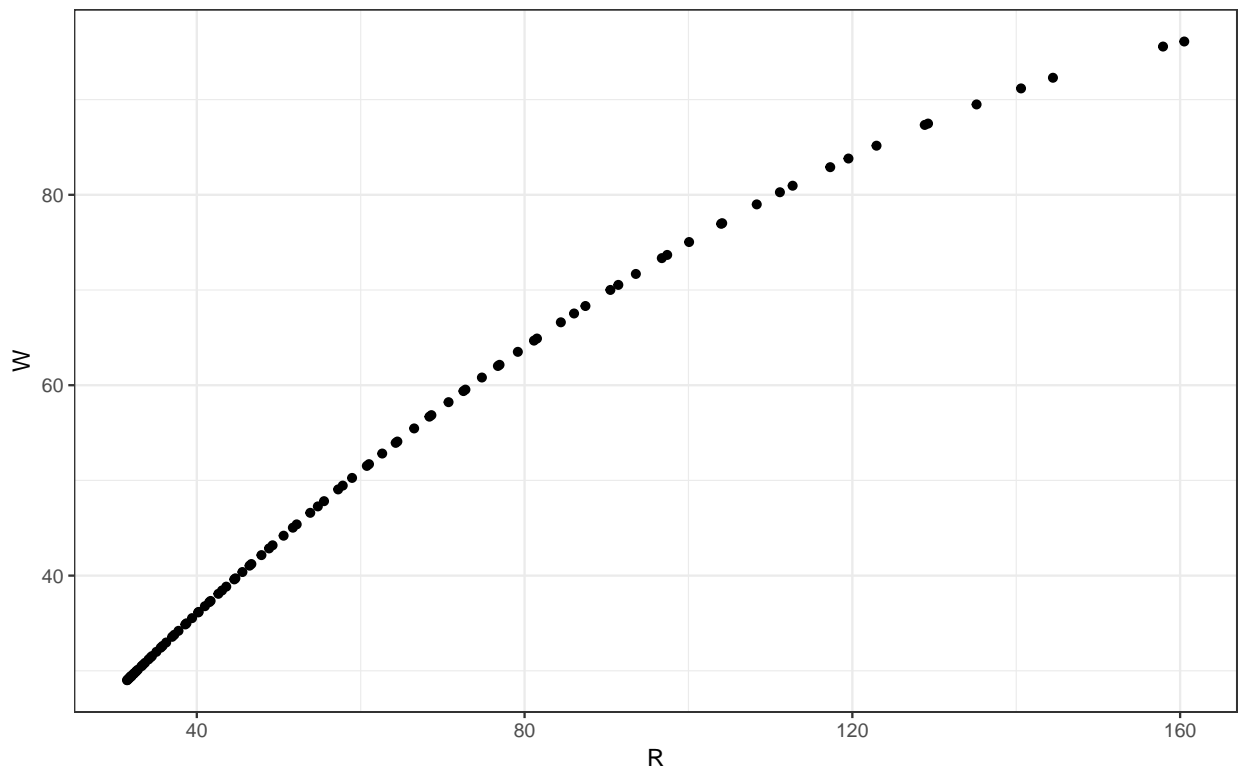

N = 100   Bandwidth = 0.3601

```r
# compute W and R
W <- k.means(z)
R <- ratio.cut(z)

# visualizations
k <- seq(n - 1)

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

```
ggplot() +
  geom_point(aes(x = R, y = W))
```


It appears that there is a definite relationship between W and R. Using quadratic regression:

4

```
summary(lm(W ~ R + I(R ** 2)))
```

```
Call:
lm(formula = W ~ R + I(R^2))

Residuals:
       Min         1Q     Median         3Q        Max
-1.363e-13 -4.573e-15 -9.690e-16  4.646e-15  5.455e-14

Coefficients:
              Estimate Std. Error    t value Pr(>|t|)
(Intercept) -3.428e-14  8.972e-15 -3.821e+00 0.000236 ***
R            1.000e+00  2.591e-16  3.859e+15  < 2e-16 ***
I(R^2)      -2.500e-03  1.552e-18 -1.611e+15  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.789e-14 on 96 degrees of freedom
Multiple R-squared:        1, Adjusted R-squared:        1
F-statistic: 6.105e+31 on 2 and 96 DF,  p-value: < 2.2e-16
```

... we get a perfect fit. We can try several values of the data size n:

```
N <- 2 ** 4  # number of obs to try
coefs.df <- lapply(seq(4, N), function(n) {
  x <- normalize(rnorm(n))  # generate data

  # compute results
  W <- k.means(x)
  R <- ratio.cut(x)

  # compute coefs for quadratic equation
  quad.coefs <- lm(W ~ R + I(R ** 2))$coefficients
  a0 <- quad.coefs['(Intercept)']
  a1 <- quad.coefs['R']
  a2 <- quad.coefs['I(R^2)']

  # compile into data frame
  dplyr::data_frame(n, a0, a1, a2)
}) %>%
  dplyr::bind_rows()

ggplot(coefs.df) +
  geom_point(aes(x = n, y = a0, colour = 'intercept')) +
  geom_point(aes(x = n, y = a1, colour = 'linear term')) +
  geom_point(aes(x = n, y = a2, colour = 'quadratic term')) +
  labs(colour = NULL, y = NULL)
```
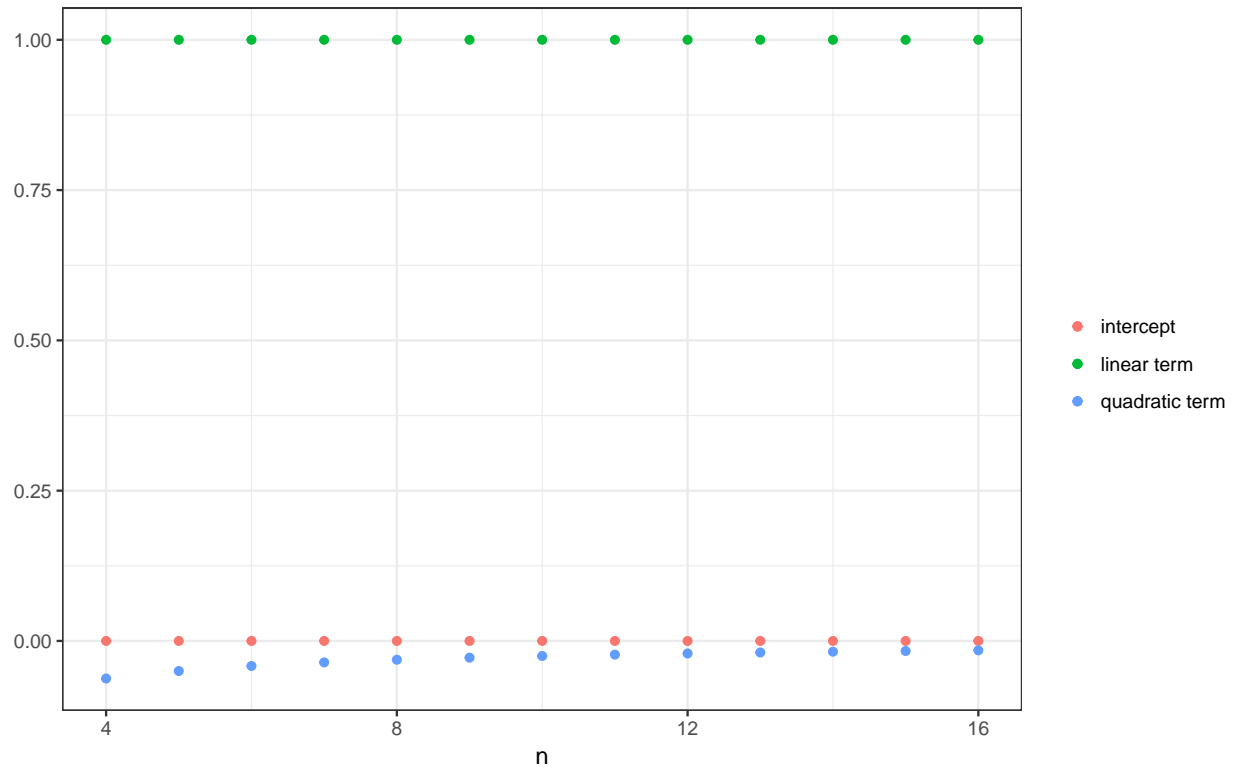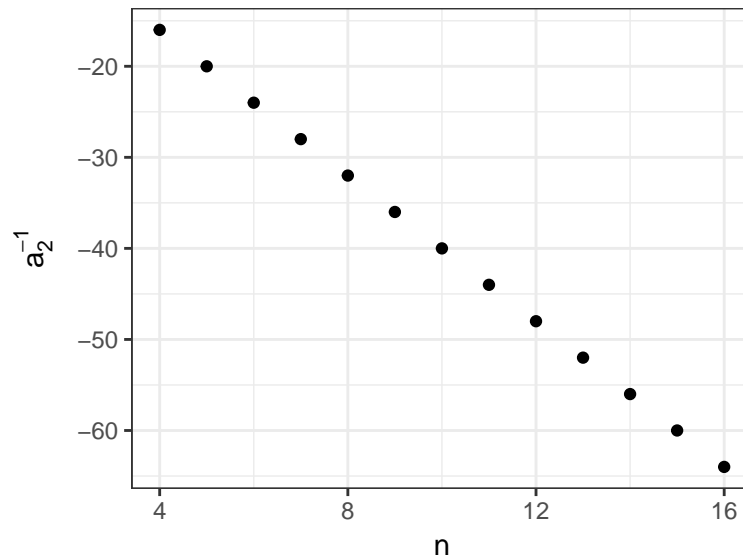
The intercept term stays at 0 and the linear term stays at 1. Looking closer at the quadratic term:

```
ggplot(coefs.df) +
  geom_point(aes(x = n, y = a2 ** -1)) +
  labs(y = expression(a[2]^-1))
```


Then we arrive at the result $W = R - \frac{1}{4n} R^2$.

Analytic result

We can show:

$$W(k) = R(k) - \frac{(R(k))^2}{4n}$$

or $W(R) = R - \frac{1}{4n}R^2$. This function is strictly increasing for $R(k) \leq 2n$. Expanding $R(k)$, we get:

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

It can be shown that $R(k)$ is maximized at the endpoints.

Note that since $\sum_i^n x_i = 0$, $\sum_i^{k<n} x_i \leq 0$, $x_n \geq 0$, and $x_i \leq 0$.

k can range from 1 to $n-1$. If $k = n-1$, we get $R(n-1) = 2n + 2\sqrt{\frac{1}{k}} \sum_i^{n-1} x_i - 2\sqrt{n-1}x_n$. The second term is ≤ 0 and the third term is ≥ 0, so we get $R \leq 2n$. On the other hand, if $k = 1$, $R(1) = 2n + 2\sqrt{n-1}x_1 - \frac{2}{\sqrt{n-1}} \sum_{i=2}^{n} x_i = 2n + 2\sqrt{n-1}x_1 - \frac{2}{\sqrt{n-1}}(-x_1) = 2n + x_1\left(2\sqrt{n-1} + \frac{2}{\sqrt{n-1}}\right) \leq 2n$.

Expanding $W(k)$, we get:

$$W(k) = \sum_{i=1}^{k} x_i^2 - 2m_1 \sum_{i=1}^{k} x_i + km_1^2 + \sum_{i=k+1}^{n} x_i^2 - 2m_2 \sum_{i=k+1}^{n} x_i + m_2^2(n-k)$$

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

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

$$= n - km_1^2 - (n-k)m_2^2$$

Since $\sum_i^n x_i = 0$, $km_1 + (n-k)m_2 = 0$, or, $-nm_2 = k(m_1 - m_2)$. Then

$$W(k) = n - km_1^2 - (n-k)m_2^2$$
$$= n - km_1^2 - nm_2^2 + km_2^2$$
$$= n - km_1^2 + (nm_2)m_2 + km_2^2$$
$$= n - km_1 + k(m_1 - m_2)m_2 + km_2 2$$
$$= n + k(-m_1^2 + m_1m_2 - m_2^2 + m_2^2)$$
$$= n + km_1(m_2 - m_1)$$

Using the relationship $-nm_2 = k(m_1 - m_2) \implies m_2 - m_1 = \frac{nm_2}{k}$ from before, we can again rewrite $W(k)$:

$$W(k) = n - (n-k)m_2\frac{nm_2}{k}$$
$$= n\frac{(n-k)n}{k}m_2^2$$

And we use the same relationship again: $m_2 - m_1 = \frac{n}{k}m_2 \implies m_2 = (m_2 - m_1)\frac{k}{n}$.

Then we can finally write $W(k)$ as:

$$W(k) = n - \frac{(n-k)n}{k}\frac{k^2}{n^2}(m_1 - m_2)^2$$

$$\boxed{W(k) = n - \frac{(n-k)k}{n}(m_1 - m_2)^2}$$

On the other hand, if we expand $R(k)$:

$$R(k) = \sum_1^k x_i^2 + 2\sqrt{\frac{n-k}{k}}\sum_1^k x_i + n - k + \sum_{k+1}^n x_i^2 - 2\sqrt{\frac{k}{n-k}}\sum_{k+1}^n x_i + k$$

$$= 2n + 2\sqrt{k(n-k)}m_1 - 2\sqrt{k(n-k)}m_2$$

If we expand and simplify $-\frac{(R(k))^2}{4n}$, we get:

$$-\frac{(R(k))^2}{4n} = -n - \frac{k(n-k)}{n}m_1^2 - \frac{k(n-k)}{n}m_2^2 - 2m_1\sqrt{k(n-k)} + 2m_2\sqrt{k(n-k)} + \frac{2k(n-k)}{n}m_1 m_2$$

Then noting that some terms cancel each other out, $R(k) - \frac{(R(k))^2}{4n}$:

$$n - \frac{k(n-k)}{n}m_1^2 - \frac{k(n-k)}{n}m_2^2 + 2\frac{k(n-k)}{n}m_1 m_2$$

$$= n - \frac{k(n-k)}{n}(m_1^2 + m_2^2 - 2m_1 m_2)$$

$$\boxed{= n - \frac{k(n-k)}{n}(m_1 - m_2)^2}$$

Which is exactly the same as our expression for $W(k)$.

Arbitrary k

When $k > 2$, the methods as prescribed by Luxburg involve embedding the graph to $\mathbb{R}^k$ and then performing k-means clustering. In this case, we cannot perform the same sort of analysis since there is no way to "order" the x_i's.

Numerical experiments for $k = 2$ and $\mathbb{R}^2$

Double spiral

We will generate a "double spiral" to be partitioned into two clusters. This is an example that k-means would fail but is fairly distinguishable visually or intuitively. After generating the spiral, we will construct a k-nearest neighbors graph.

```r
# borrow some functions from S675
source('http://pages.iu.edu/~mtrosset/Courses/675/manifold.r')

# parameters
set.seed(112358)
s <- 2 ** 5
eps <- 2 ** -2
k <- 10
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

# 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)

# viz
ggplot(spiral.df) +
  geom_point(aes(x = X, y = Y)) +
  coord_fixed()
```
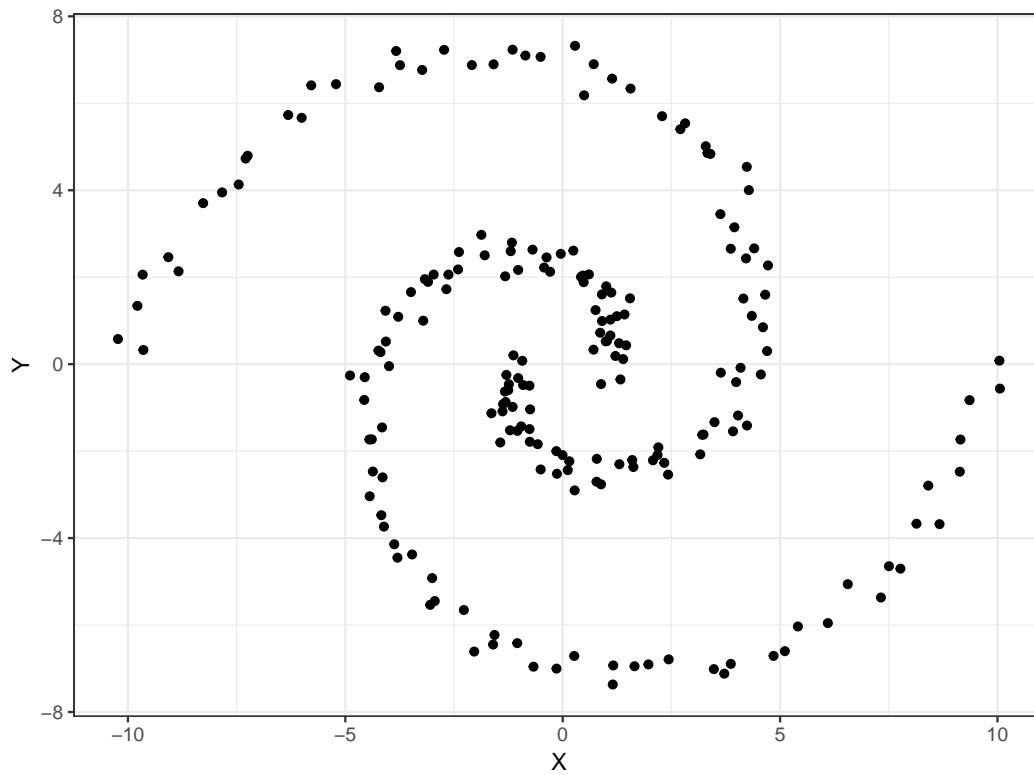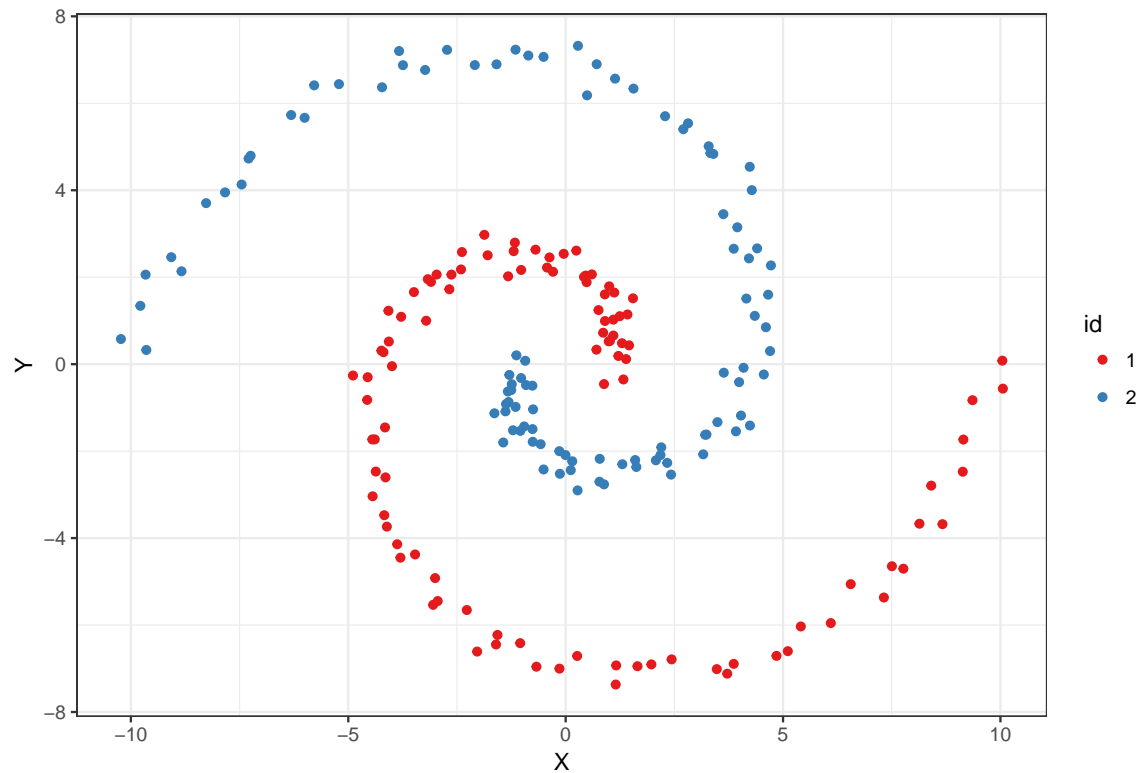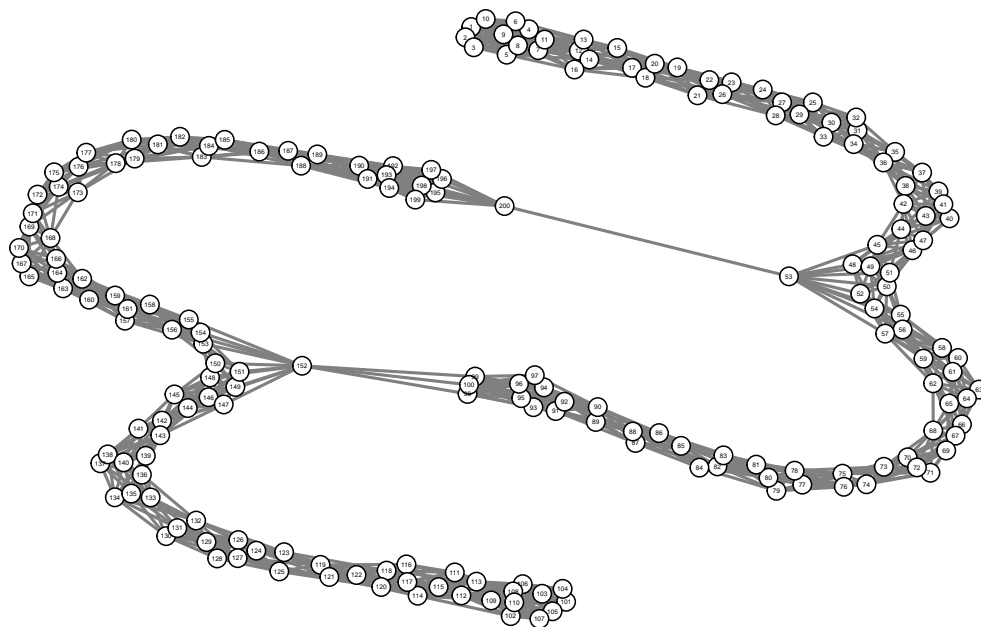
```
ggplot(spiral.df) +
  geom_point(aes(x = X, y = Y, colour = id)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1')
```

Then we will construct a(n) 10-nearest-neighbors graph:

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

# viz
qgraph::qgraph(W)
```

Here it is pretty obvious how we should cut the graph.

Then proceed with the clustering method as described by Luxburg:

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

# eigendecomposition
L.eigen <- eigen(L)

two.eigen.df <- L.eigen$vectors[, c(n - 2, n - 1)] %>%
  as.data.frame() %>%
  dplyr::mutate(id = spiral.df$id) %>%
  dplyr::mutate(X1 = V1 / sqrt(L.eigen$values[n - 2]),
                X2 = V2 / sqrt(L.eigen$values[n - 1]))
```
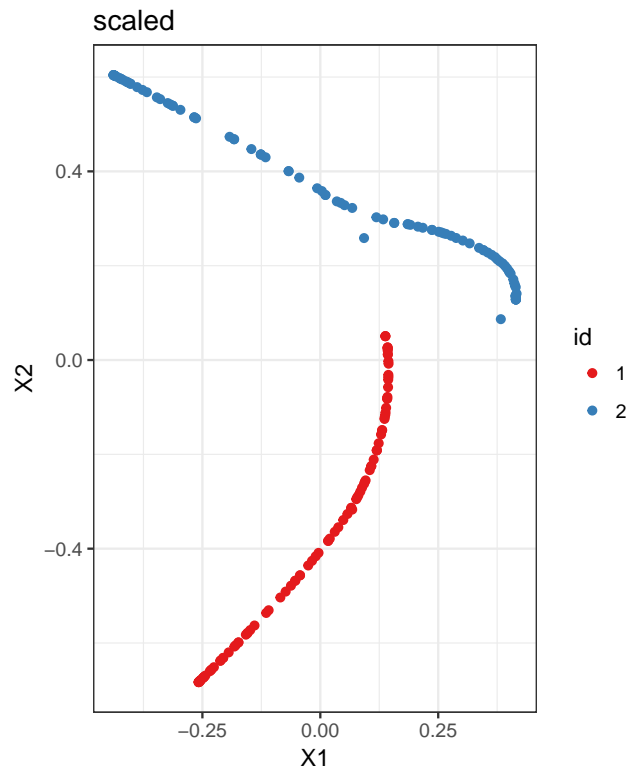
Then we can take a look at the projection to $\mathbb{R}^2$:
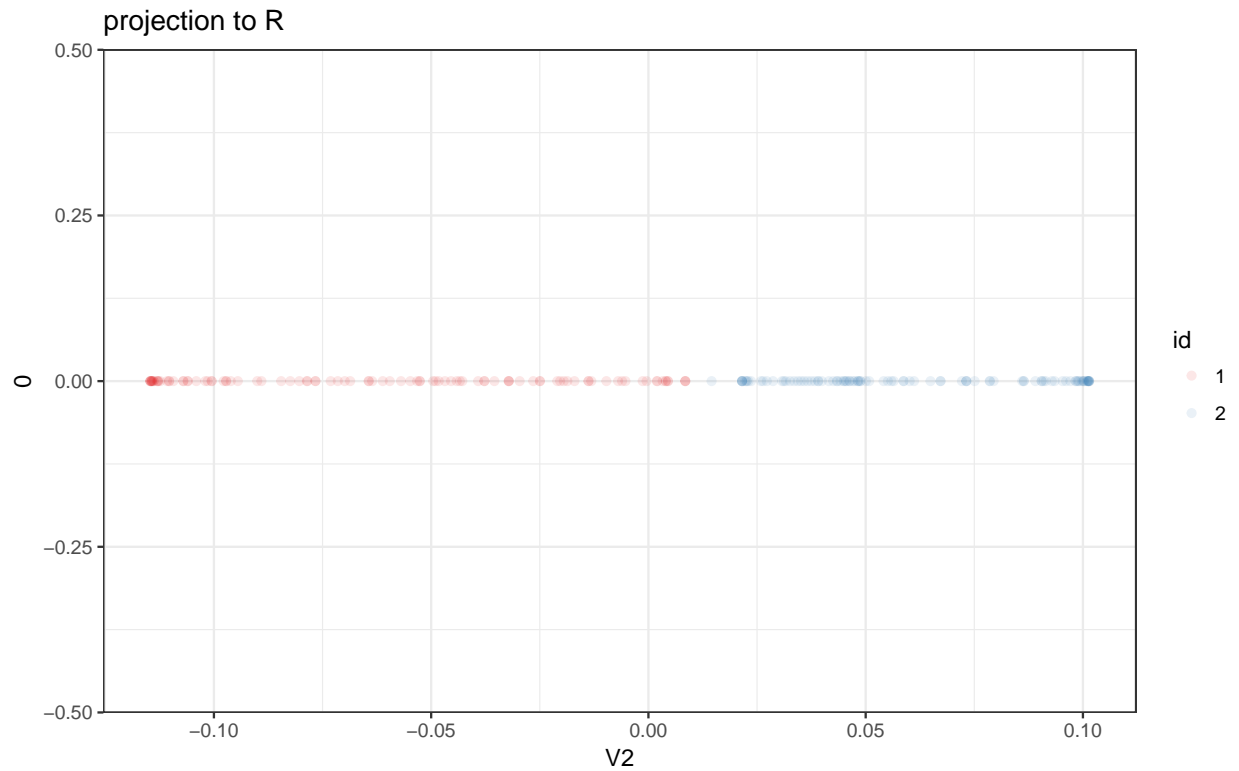
```
ggplot(two.eigen.df) +
  geom_point(aes(x = V1, y = V2, colour = id)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1') +
  labs(title = 'unscaled')
```
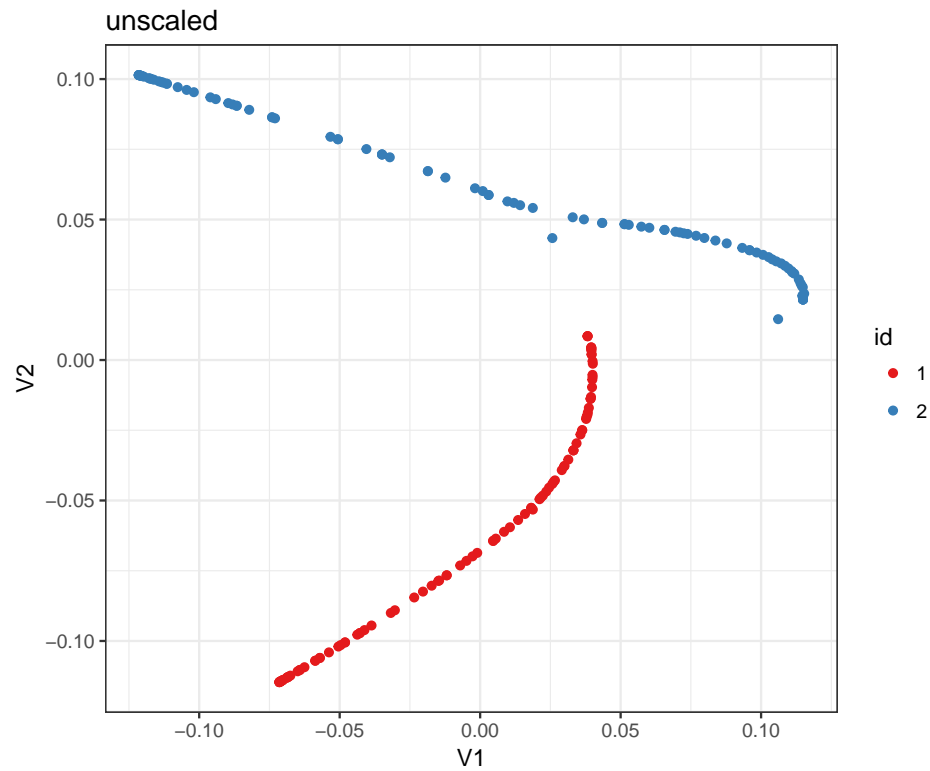
```r
ggplot(two.eigen.df) +
  geom_point(aes(x = X1, y = X2, colour = id)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1') +
  labs(title = 'scaled')
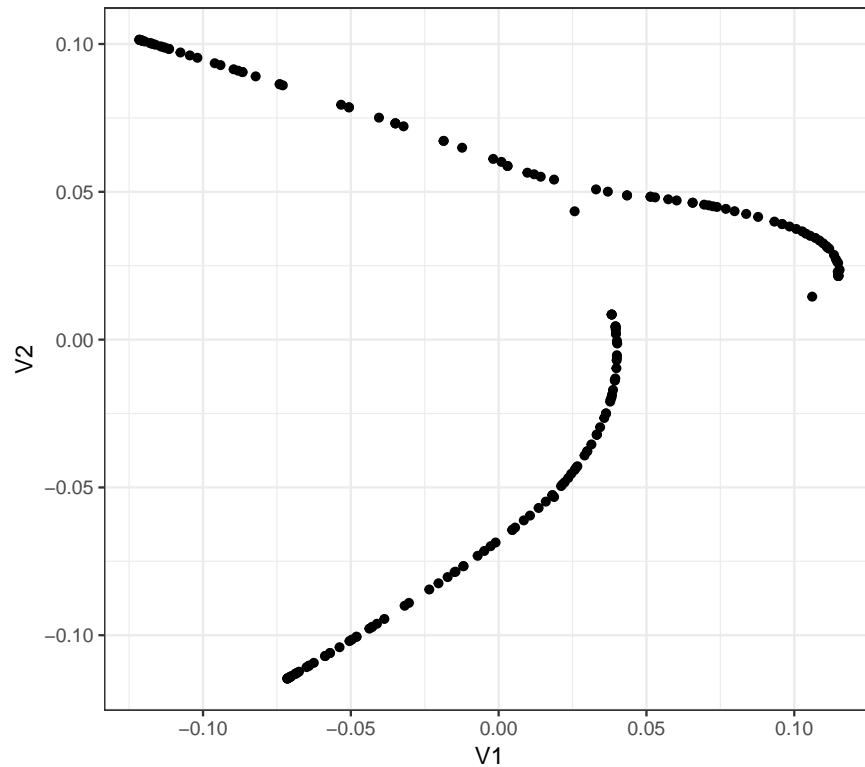```

## scaled



```
ggplot(two.eigen.df) +
  geom_point(aes(x = V2, y = 0, colour = id),
             alpha = .1) +
  scale_colour_brewer(palette = 'Set1') +
  labs(title = 'projection to R')
```

Next, we can take a look at how the three methods correspond to our original labels:

```
ggplot(two.eigen.df) +
  geom_point(aes(x = V1, y = V2)) +
  coord_fixed()
```

```
unscaled.clustering <- two.eigen.df %>%
  dplyr::select(V1, V2) %>%
  dist() %>%
  kmeans(2)

scaled.clustering <- two.eigen.df %>%
  dplyr::select(X1, X2) %>%
  dist() %>%
  kmeans(2)

one.d.clustering <- two.eigen.df %>%
  dplyr::select(X2) %>%
  dist() %>%
  kmeans(2)

spiral.df %<>%
  dplyr::mutate(unscaled = as.character(unscaled.clustering$cluster),
                scaled = as.character(scaled.clustering$cluster),
                one.d = as.character(one.d.clustering$cluster))

table(spiral.df$id, spiral.df$unscaled)


     1  2
  1 43 57
  2 40 60

table(spiral.df$id, spiral.df$scaled)
```

```
        1   2
1  20  80
2 100   0
```

```r
table(spiral.df$id, spiral.df$one.d)
```
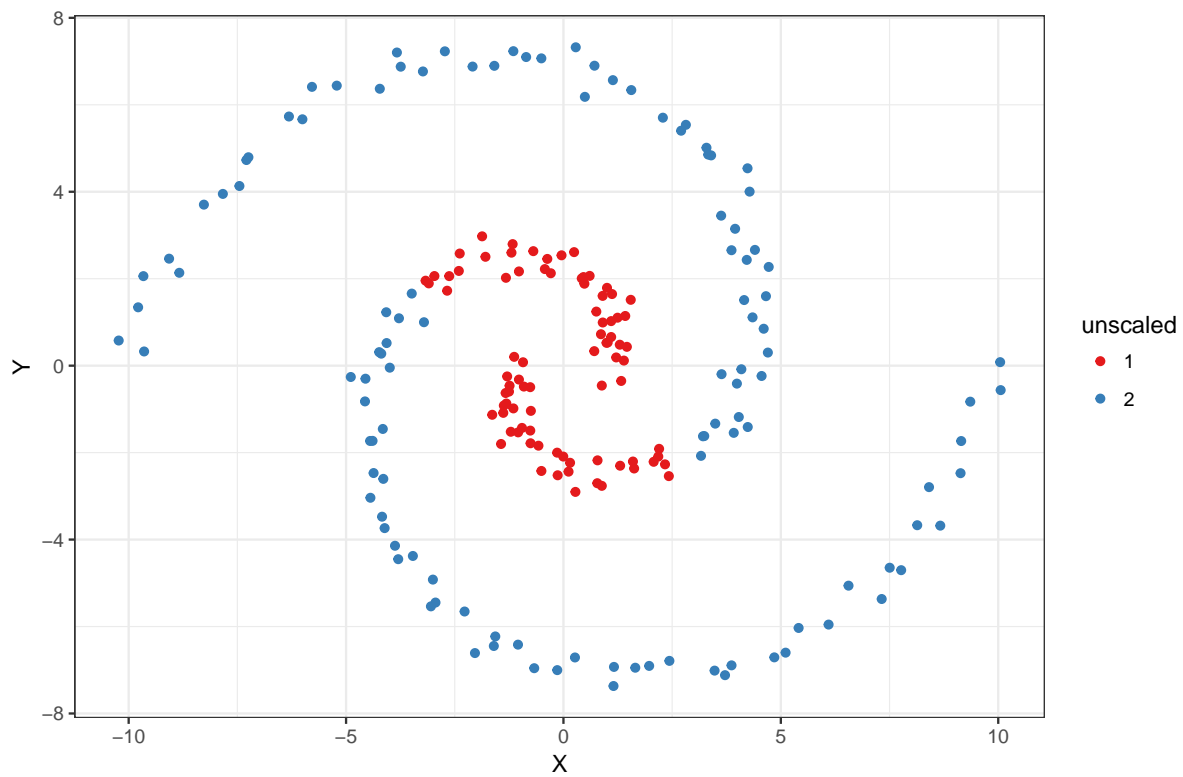
```
        1   2
1  19  81
2 100   0
```

```r
ggplot(spiral.df) +
  geom_point(aes(x = X, y = Y, colour = unscaled)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1')
```
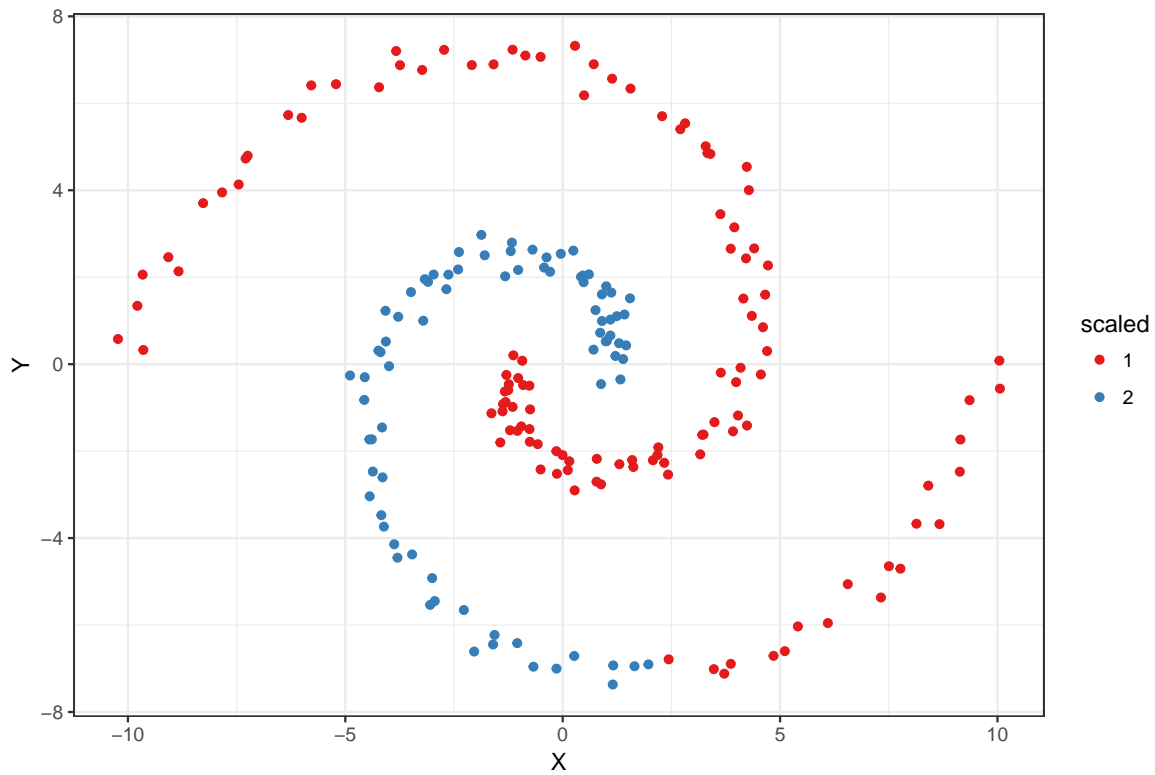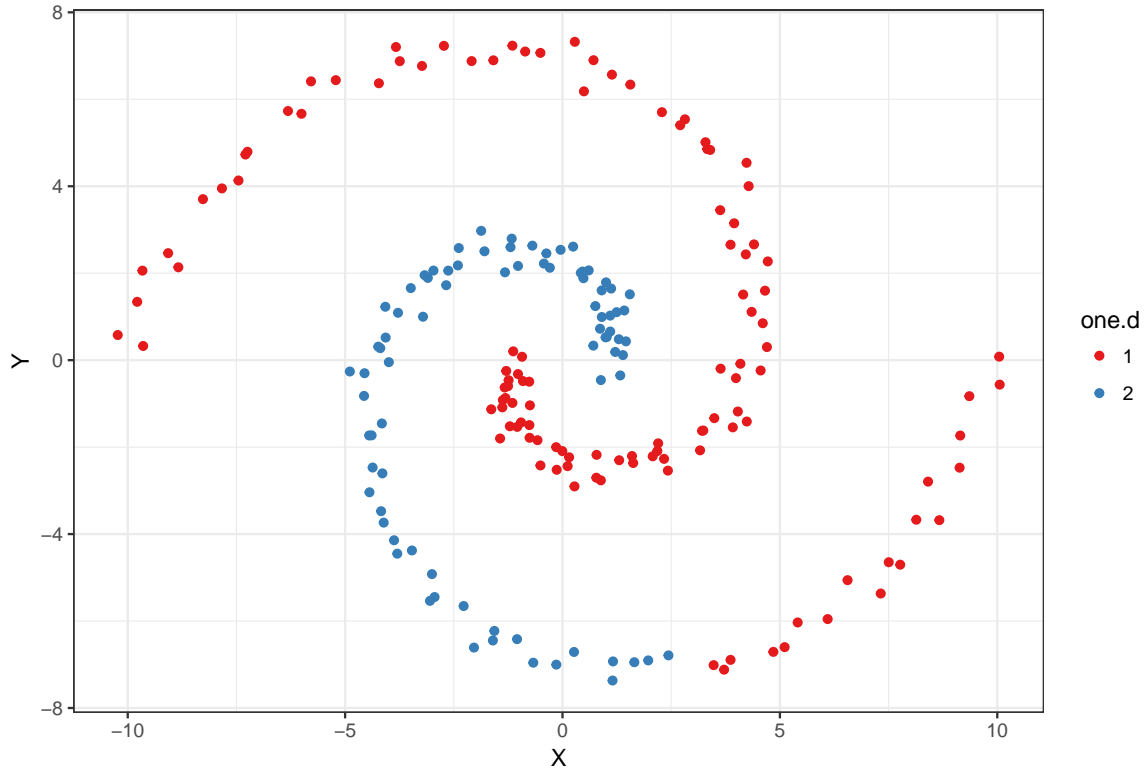


```r
ggplot(spiral.df) +
  geom_point(aes(x = X, y = Y, colour = scaled)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1')
```

```
ggplot(spiral.df) +
  geom_point(aes(x = X, y = Y, colour = one.d)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1')
```

In a way, we have "unraveled" the spirals, resulting in two curves. Eye-balling this representation gives us two obvious clusters, but k-means fails to separate them the way we would expect since they are not spherical clusters. Projection to the unscaled space seems to perform especially poorly, and the plots of the columns of H reveal why.

```r
two.eigen.df %<>%
  dplyr::mutate(unscaled.clust = as.character(unscaled.clustering$cluster),
                scaled.clust = as.character(scaled.clustering$cluster),
                one.d.clust = as.character(one.d.clustering$cluster))

clust.list <- list('unscaled.clust', 'scaled.clust', 'one.d.clust')

clust.plots <- lapply(clust.list, function(i) {
  ggplot(two.eigen.df) +
    geom_point(aes_string(x = 'X1', y = 'X2', colour = i)) +
    scale_colour_brewer(palette = 'Set1') +
    coord_fixed() +
    guides(colour = FALSE) +
    labs(x = NULL, y = NULL)
})

.gridarrange <- function(...) gridExtra::grid.arrange(..., ncol = 3)
do.call(.gridarrange, clust.plots)
```
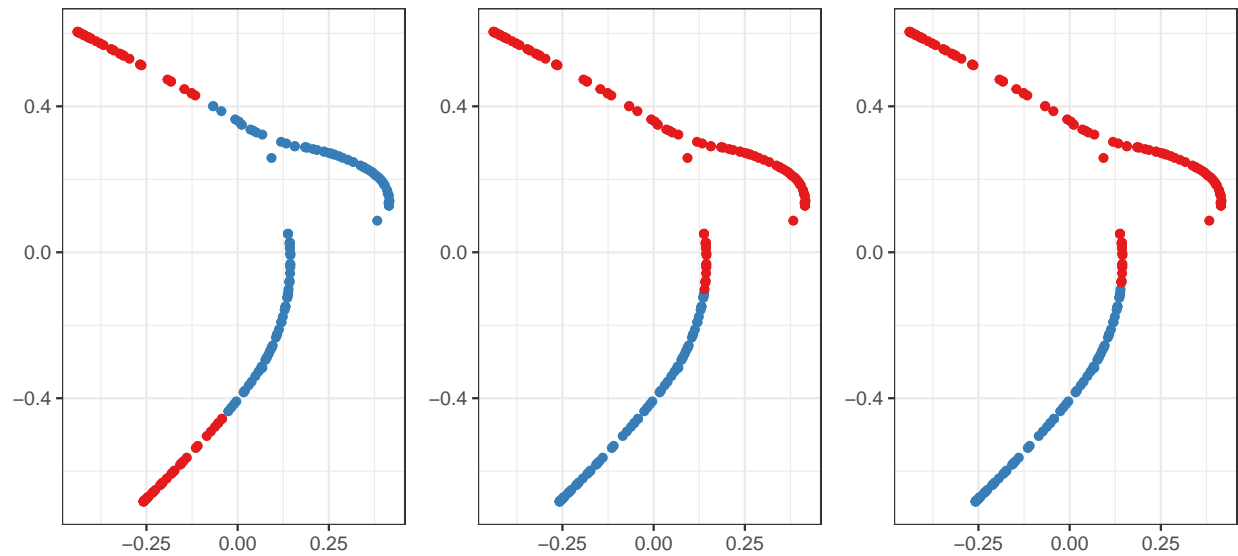
In fact, it almost looks as if another clustering method is more appropriate here, in particular, single-linkage

```r
single.link.clust <- two.eigen.df %>%
  dplyr::select(V1, V2) %>%
  dist() %>%
  hclust(method = 'single') %>%
  cutree(2)

two.eigen.df %<>%
  dplyr::mutate(single.link.clust = as.character(single.link.clust))
spiral.df %<>%
  dplyr::mutate(single.link.clust = as.character(single.link.clust))

ggplot(two.eigen.df) +
  geom_point(aes(x = V1, y = V2, colour = single.link.clust)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1')
```
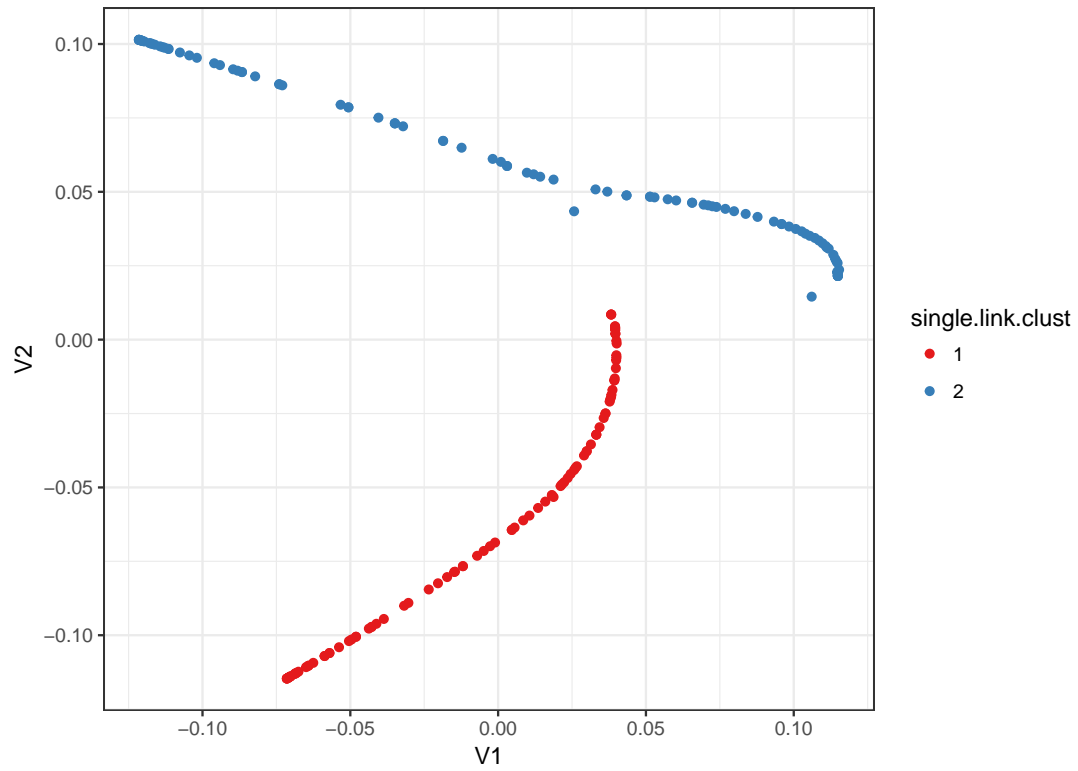
```r
ggplot(spiral.df) +
  geom_point(aes(x = X, y = Y, colour = single.link.clust)) +
  coord_fixed() +
  scale_colour_brewer(palette = 'Set1')
```
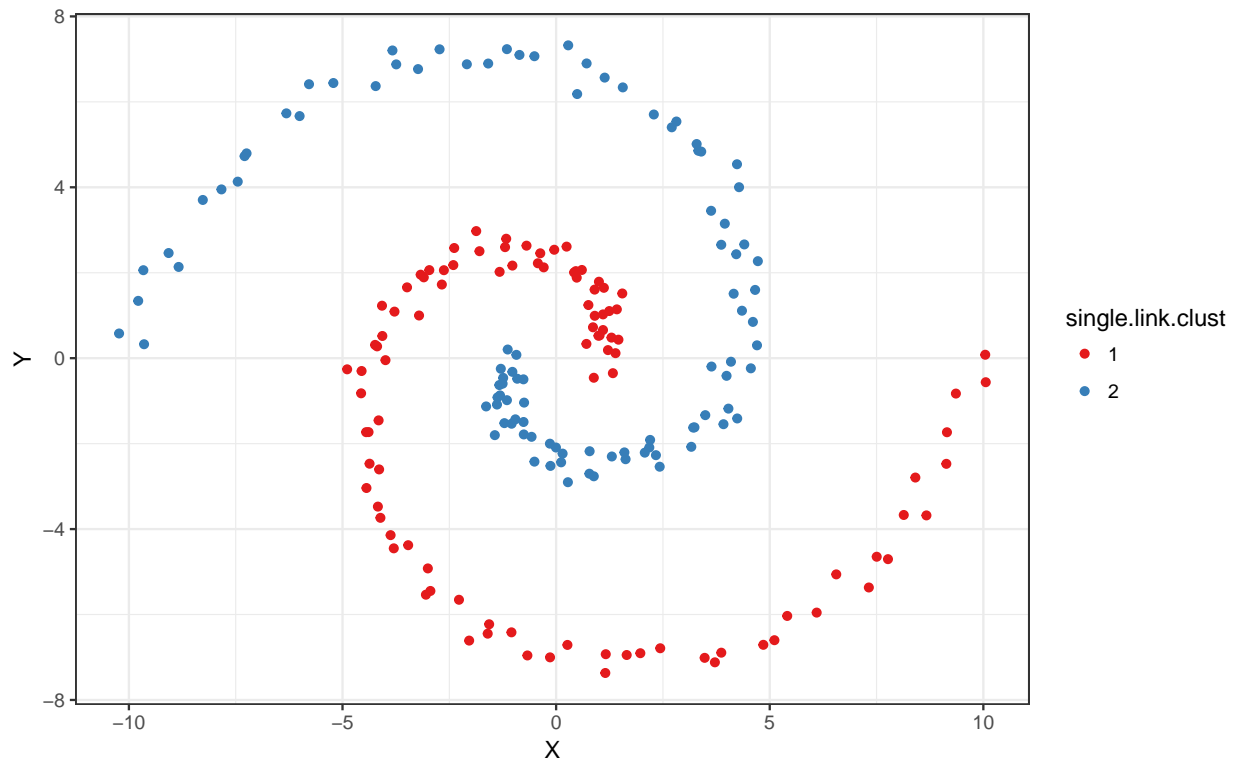
```
table(spiral.df$id, spiral.df$single.link.clust)
```

```
      1   2
1 100   0
2   0 100
```

However, this may be a consequence of this particular problem. The two spirals are basically two intertwined strands.