STAT-S675

Homework 10

John Koo

Link to assignment

```r
dp <- loadNamespace('dplyr')
import::from(magrittr, `%>%`, `%<>%`)
import::from(ggplot2, ggplot, aes,
             geom_point, stat_bin_hex, geom_histogram,
             geom_abline, geom_vline,
             coord_map, coord_fixed,
             labs, theme_set, theme_bw, scale_colour_brewer,
             facet_wrap,
             scale_x_continuous, scale_y_log10)
import::from(viridis, scale_fill_viridis)

theme_set(theme_bw())

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

set.seed(675)
```

Exercise 1

Part a

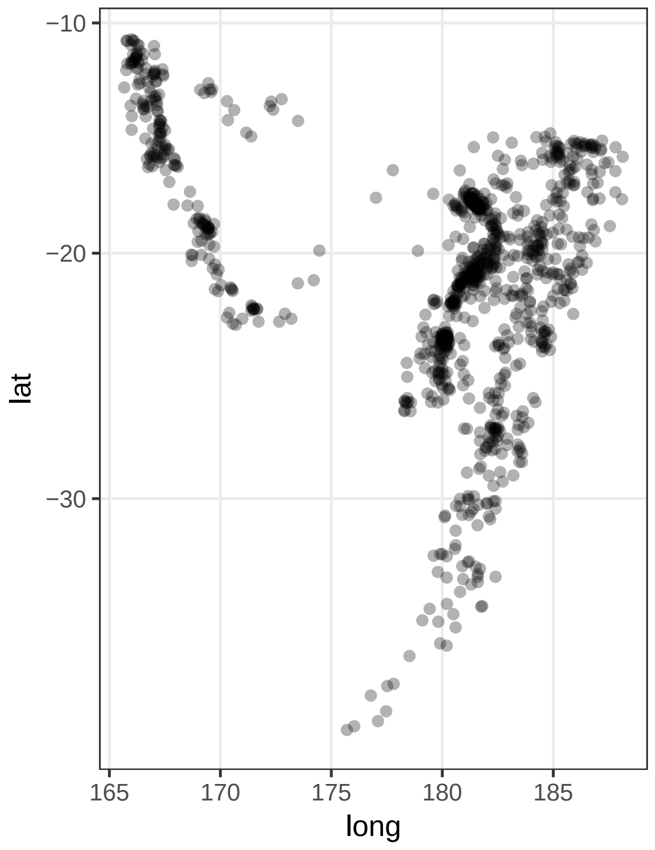
`kmeans.start` selects k random objects from the data as the initial cluster centers.

Part b

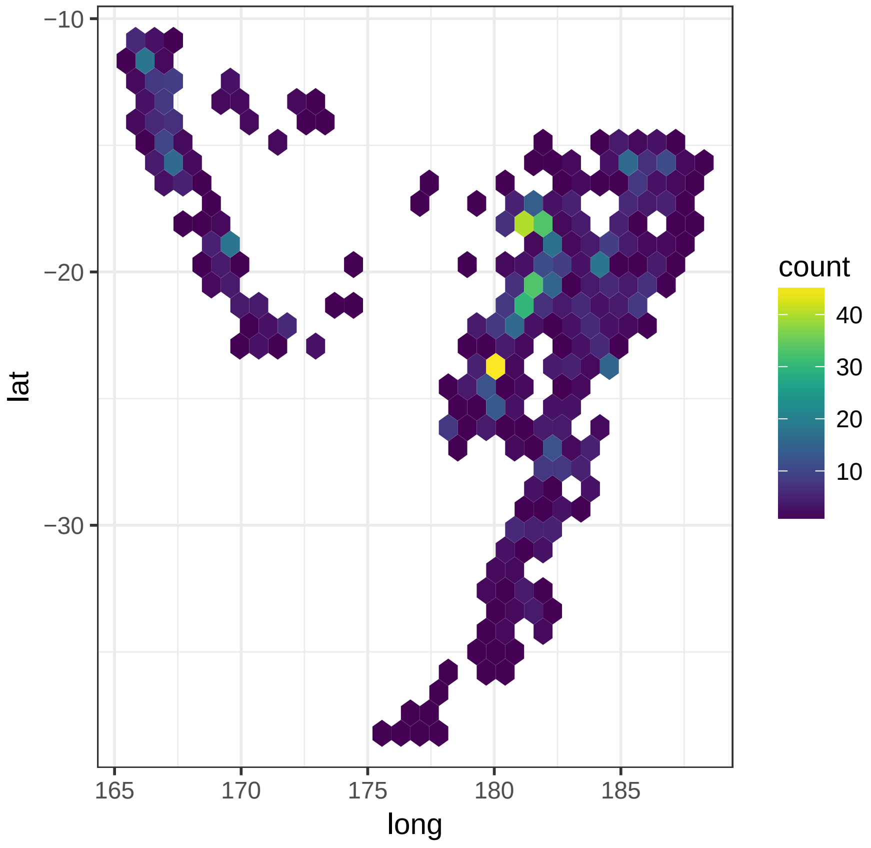
`kmeans.exchange` picks a random point at each iteration whereas `kmeans.exchange2` uses a cyclic exchange method.

Exercise 2

```r
ggplot(quakes) +
  coord_map() +
  geom_point(aes(x = long, y = lat), alpha = .3)
```



```r
ggplot(quakes) +
  stat_bin_hex(aes(x = long, y = lat)) +
  scale_fill_viridis() +
  coord_fixed()
```

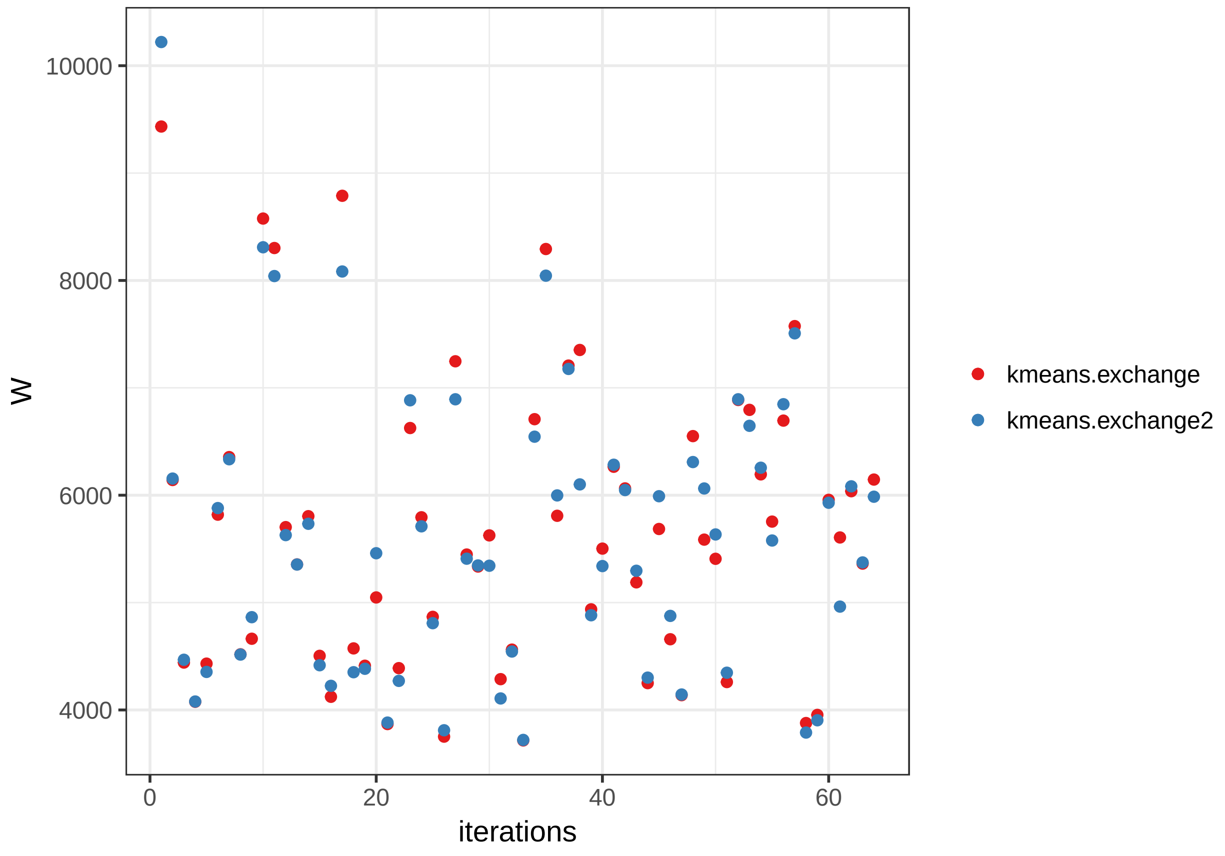


Exercise 3

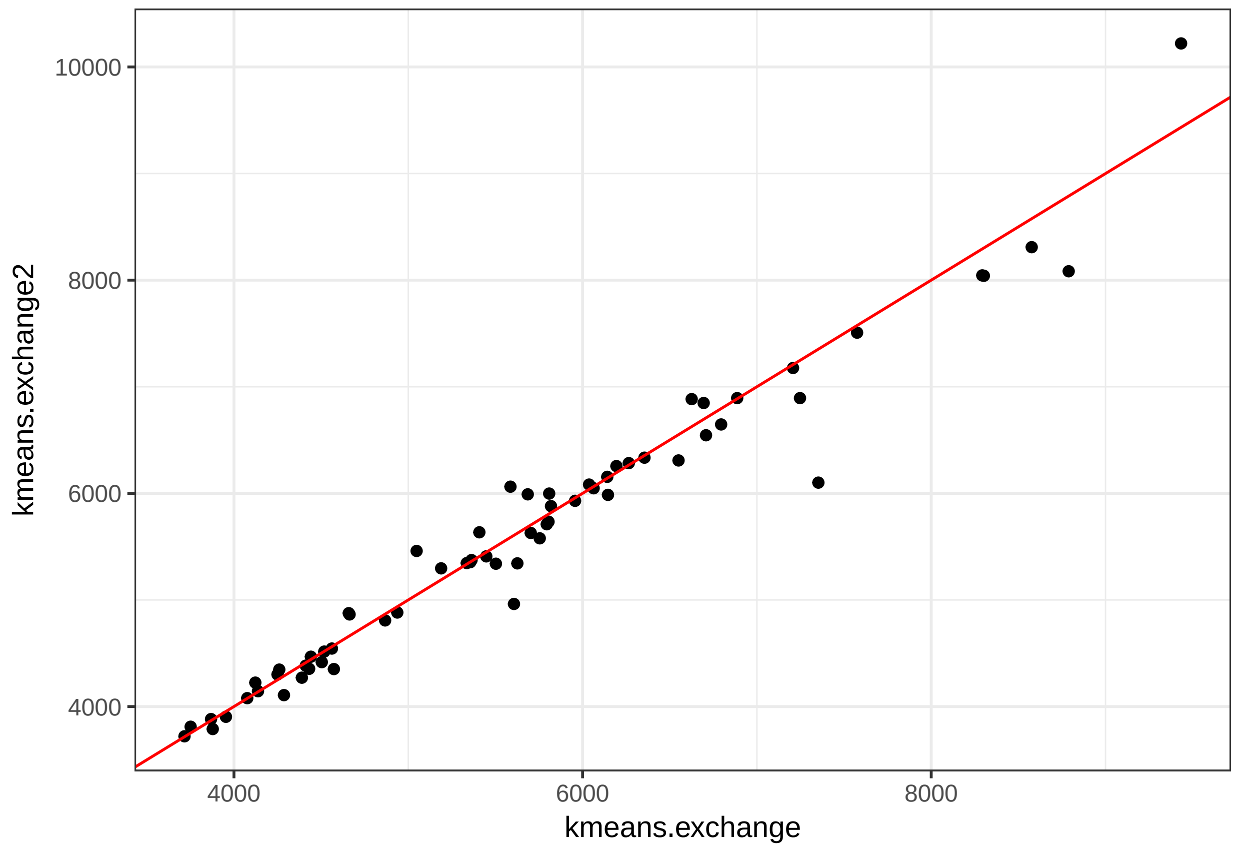
```
# number of clusters
k <- 8

# number of times to try
trials <- 2 ** 6

# convert the data into a matrix
# so we can use with kmeans.exchange and kmeans.exchange2
X <- quakes %>%
  dp$select(lat, long) %>%
  as.matrix()

# initialize the centers for clustering
init.centers <- lapply(seq(trials), function(i) kmeans.start(nrow(X), k))
```

```
# apply clustering functions
clusterings <- lapply(init.centers, function(i) kmeans.exchange(X, i))
clusterings.2 <- lapply(init.centers, function(i) kmeans.exchange(X, i))

# within-cluster similarity
W <- sapply(clusterings, function(clustering) clustering$W)
W.2 <- sapply(clusterings.2, function(clustering) clustering$W)
```

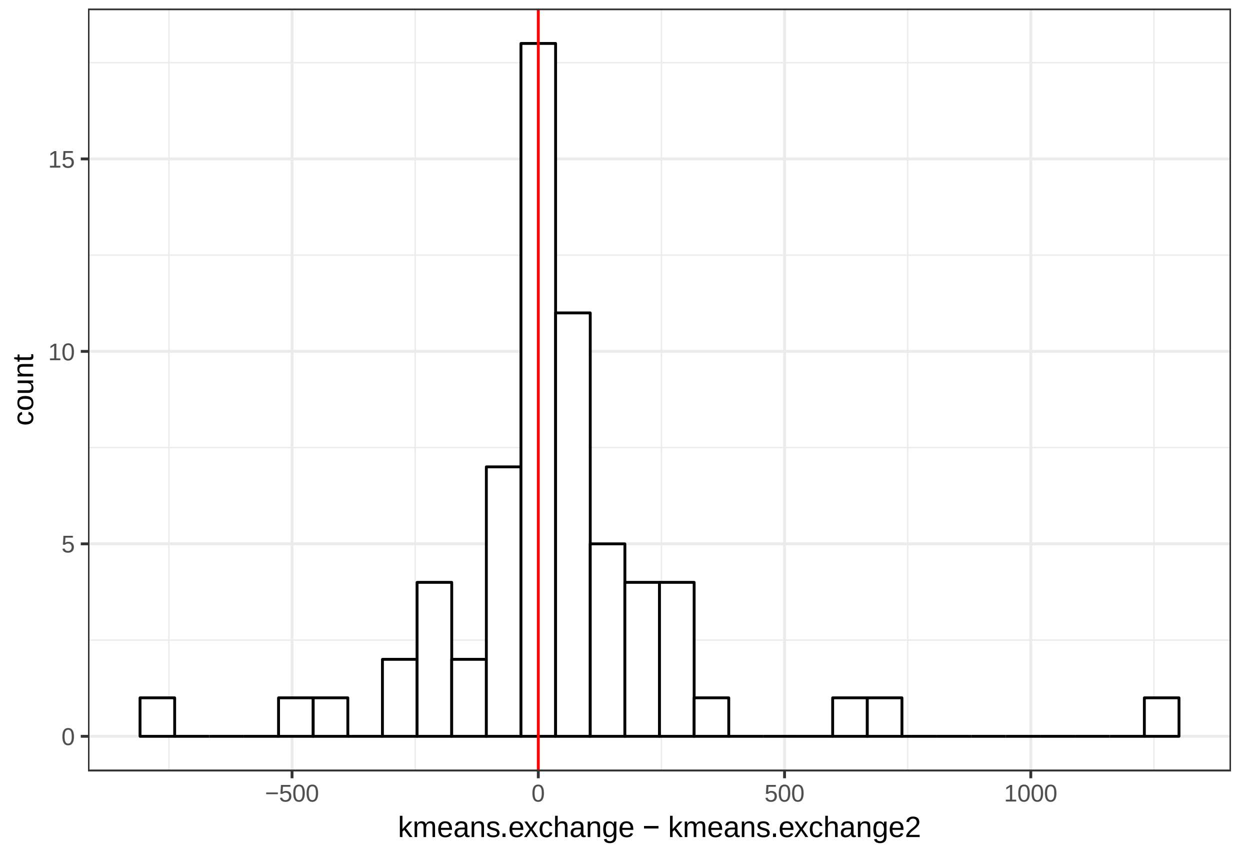
```
ggplot() +
  geom_point(aes(x = seq(trials), y = W, colour = 'kmeans.exchange')) +
  geom_point(aes(x = seq(trials), y = W.2, colour = 'kmeans.exchange2')) +
  scale_colour_brewer(palette = 'Set1') +
  labs(x = 'iterations', colour = NULL)
```



```
ggplot() +
  geom_point(aes(x = W, y = W.2)) +
  geom_abline(colour = 'red') +
  labs(x = 'kmeans.exchange', y = 'kmeans.exchange2')
```



```
ggplot() +
  geom_histogram(aes(x = W - W.2),
                 colour = 'black', fill = 'white') +
  labs(x = 'kmeans.exchange - kmeans.exchange2') +
  geom_vline(xintercept = 0, colour = 'red')
```



```
summary(W - W.2)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-787.11  -53.18   18.99   40.12  126.21 1252.07 
```

```
t.test(W, W.2, paired = TRUE)
```

```

	Paired t-test

data:  W and W.2
t = 1.1946, df = 63, p-value = 0.2367
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -26.99187 107.22875
sample estimates:
mean of the differences 
               40.11844 
```

`kmeans.exchange2` tends to produce smaller values of W, although the difference may not be significant.

Exercise 4

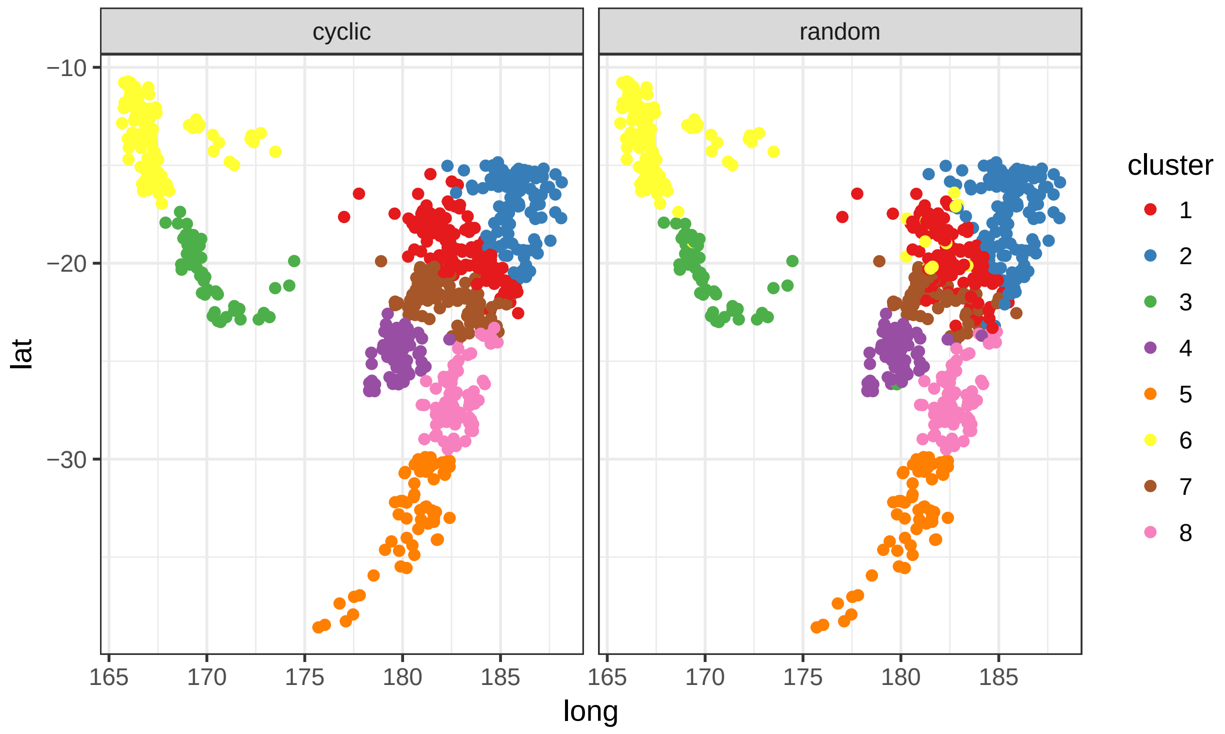
```
# copy/paste Dr. Trosset's code with modifications
kmeans.exchange <- function(X,j,niter=2000) {
```

```r
  N <- nrow(X)
  p <- ncol(X)
  k <- length(j)
  means <- X[j,]
  X <- cbind(rep(0,N),X)
  for (j in 1:N) {
    Y <- means-matrix(rep(X[j,-1],k),byrow=T,ncol=p)
    dist2 <- apply(Y^2,1,sum)
    X[j,1] <- order(dist2)[1]
  }
  X <- X[order(X[,1]),]
  n <- as.vector(summary(as.factor(X[,1])))

  r <- 0
  for (i in 1:k) {
    Y <- X[(r+1):(r+n[i]),-1]
    means[i,] <- apply(matrix(Y,ncol=p),2,mean)
    r <- r+n[i]
  }

  for (iter in 1:niter) {
    j <- sample(x=1:N,size=1)
    x <- X[j,-1]
    currC <- X[j,1]
    dist2 <- sum((x-means[currC,])^2)
    bestC <- currC
    for (i in (1:k)[-currC]) {
      if (n[i] > 0) {
        newdist2 <- sum((x-means[i,])^2)
      }
      if (newdist2 < dist2) {
        bestC <- i
        dist2 <- newdist2
      }
    }
    if (bestC != currC) {
      X[j,1] <- bestC
      n1 <- n[currC]
      n[currC] <- n1-1
      if (n[currC] > 0) {
        means[currC,] <- (n1*means[currC,]-x)/n[currC]
      }
      n2 <- n[bestC]
      n[bestC] <- n2+1
      means[bestC,] <- (n2*means[bestC,]+x)/n[bestC]
    }

  }

  W <- sum((X[order(X[,1]),-1]-matrix(rep(means,rep(n,p)),nrow=N))^2)
  return(list(X = X, W=W,k=k))
}
```

```r
kmeans.exchange2 <- function(X,j,niter=2000) {

  N <- nrow(X)
  p <- ncol(X)
  k <- length(j)
  means <- X[j,]
  X <- cbind(rep(0,N),X)
  for (j in 1:N) {
    Y <- means-matrix(rep(X[j,-1],k),byrow=T,ncol=p)
    dist2 <- apply(Y^2,1,sum)
    X[j,1] <- order(dist2)[1]
  }
  X <- X[order(X[,1]),]
  n <- as.vector(summary(as.factor(X[,1])))

  r <- 0
  for (i in 1:k) {
    Y <- X[(r+1):(r+n[i]),-1]
    means[i,] <- apply(matrix(Y,ncol=p),2,mean)
    r <- r+n[i]
  }

  jj <- sample(x=1:N,size=min(c(N,niter)),replace=F)
  for (iter in 1:niter) {
    j <- iter %% N
    if (j<1) j <- N
    j <- jj[j]
    x <- X[j,-1]
    currC <- X[j,1]
    dist2 <- sum((x-means[currC,])^2)
    bestC <- currC
    for (i in (1:k)[-currC]) {
      if (n[i] > 0) {
        newdist2 <- sum((x-means[i,])^2)
      }
      if (newdist2 < dist2) {
        bestC <- i
        dist2 <- newdist2
      }
    }
    if (bestC != currC) {
      X[j,1] <- bestC
      n1 <- n[currC]
      n[currC] <- n1-1
      if (n[currC] > 0) {
        means[currC,] <- (n1*means[currC,]-x)/n[currC]
      }
      n2 <- n[bestC]
      n[bestC] <- n2+1
      means[bestC,] <- (n2*means[bestC,]+x)/n[bestC]
    }

  }
  W <- sum((X[order(X[,1]),-1]-matrix(rep(means,rep(n,p)),nrow=N))^2)
  return(list(X = X, W=W,k=k))
}

# apply the functions with one of the initial configurations
X.kmeans <- kmeans.exchange(X, init.centers[[1]])
X.kmeans.2 <- kmeans.exchange2(X, init.centers[[1]])

# extract the clusterings
exchange.df <- dp$bind_rows(
  X.kmeans$X %>%
    as.data.frame() %>%
    dp$transmute(method = 'random',
                 cluster = as.character(V1),
                 long, lat),
  X.kmeans.2$X %>%
    as.data.frame() %>%
    dp$transmute(method = 'cyclic',
                 cluster = as.character(V1),
                 long, lat)
)

print(c(X.kmeans$W, X.kmeans.2$W))
```

```
[1] 9571.009 3769.142
```

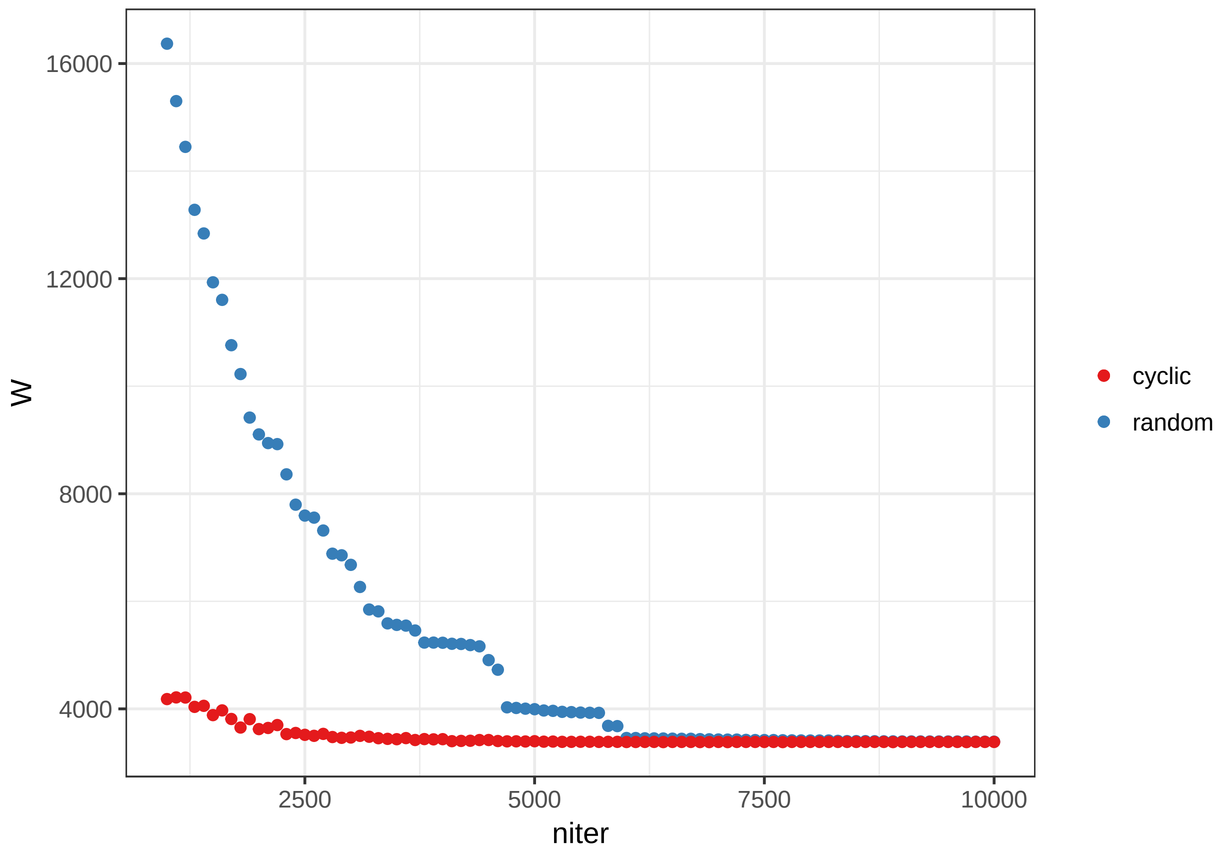
```
# plot
ggplot(exchange.df) +
  geom_point(aes(x = long, y = lat, colour = cluster)) +
  coord_fixed() +
  facet_wrap(~ method) +
  scale_colour_brewer(palette = 'Set1')
```



It appears that the random method `kmeans.exchange` results in some stragglers that are not assigned to the appropriate cluster. This makes sense since objects are chosen randomly in this method.

Exercise 5

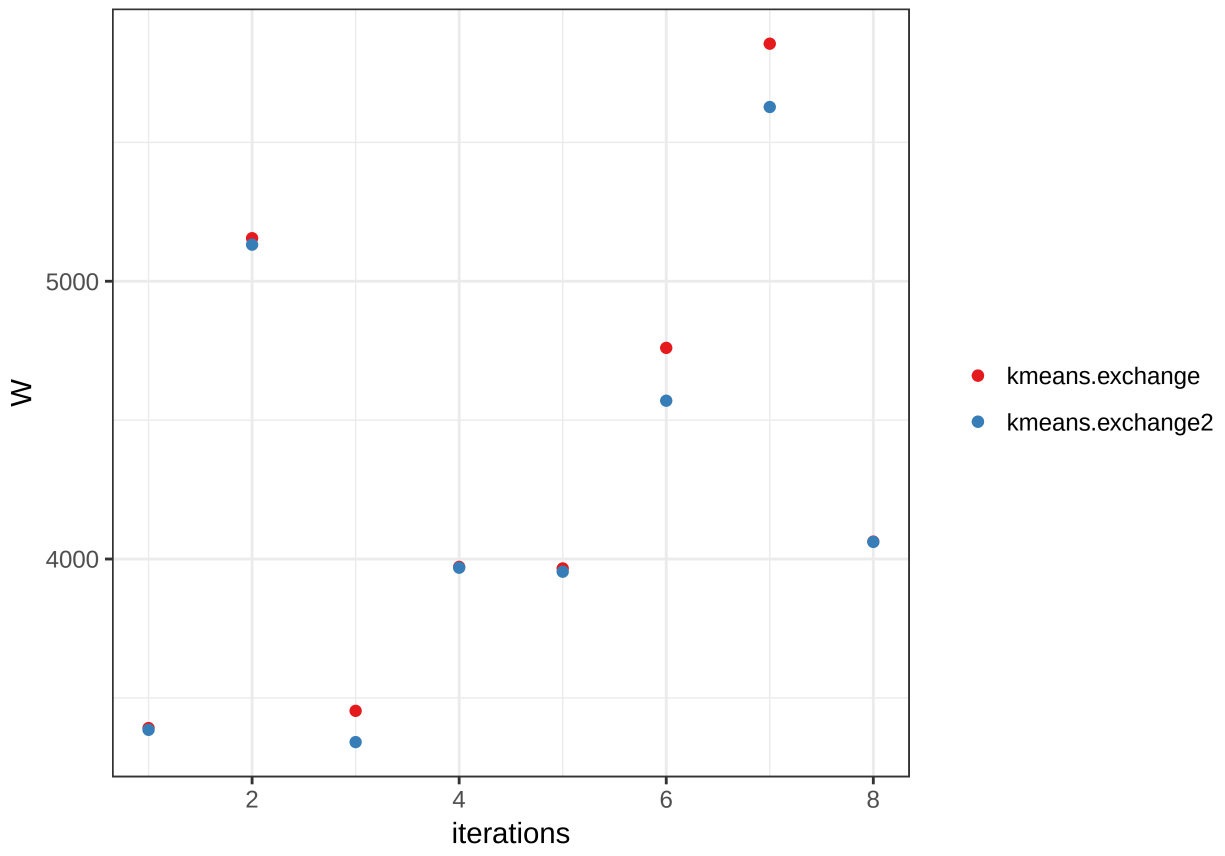
```
# vary the number of iterations
iter.vec <- seq(1000, 10000, 100)

# try for first method (random)
W.vec <- sapply(iter.vec, function(i) {
  # same seed for each since there's randomness in this method
  set.seed(675)

  kmeans.exchange(X, init.centers[[1]], i)$W
})

# second method (cyclic)
W.2.vec <- sapply(iter.vec, function(i) {
  kmeans.exchange2(X, init.centers[[1]], i)$W
})

ggplot() +
  geom_point(aes(x = iter.vec, y = W.vec, colour = 'random')) +
  geom_point(aes(x = iter.vec, y = W.2.vec, colour = 'cyclic')) +
  scale_colour_brewer(palette = 'Set1') +
  labs(x = 'niter', y = 'W', colour = NULL)
```



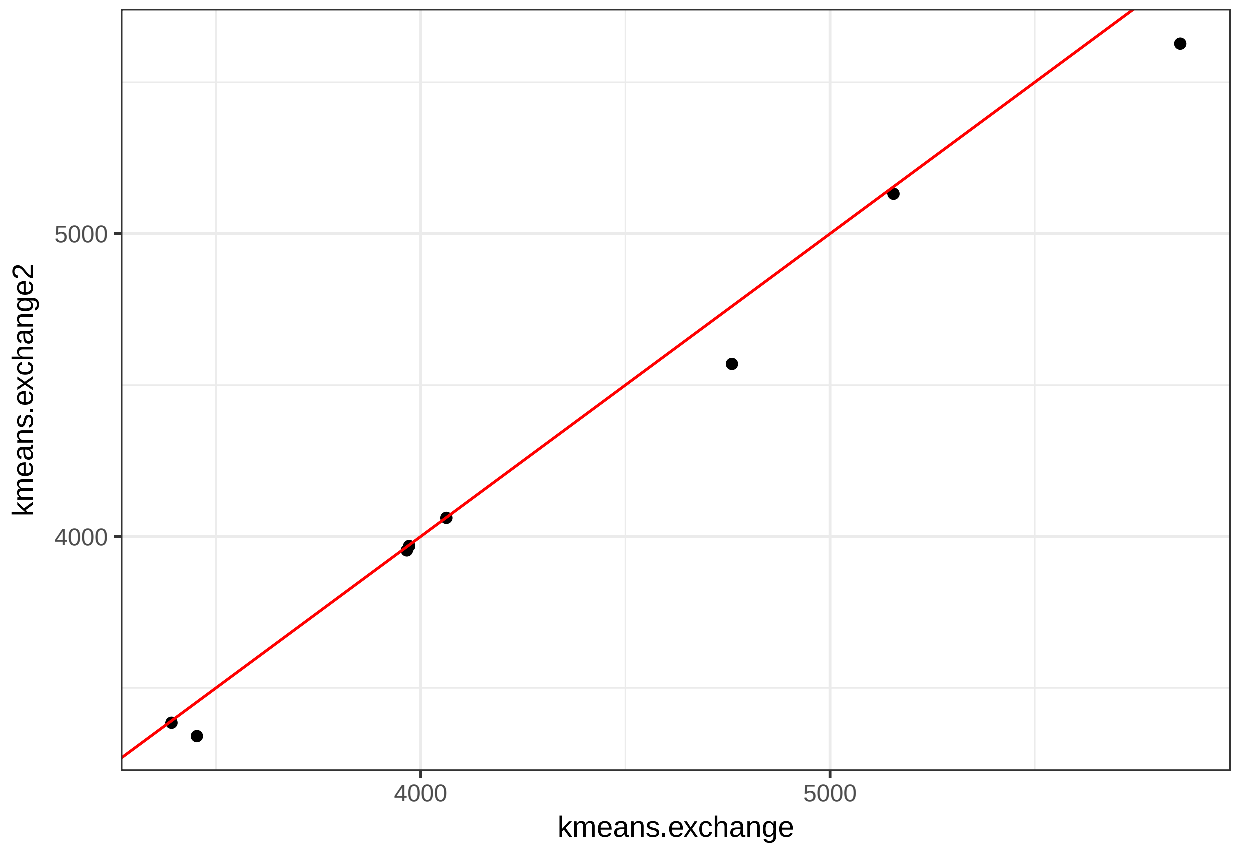
From this plot, we can see that as we increase `niter`, we converge to the optimal clustering for the given k. The cyclic method converges faster, although this may depend on the initial configuration.

Exercise 6

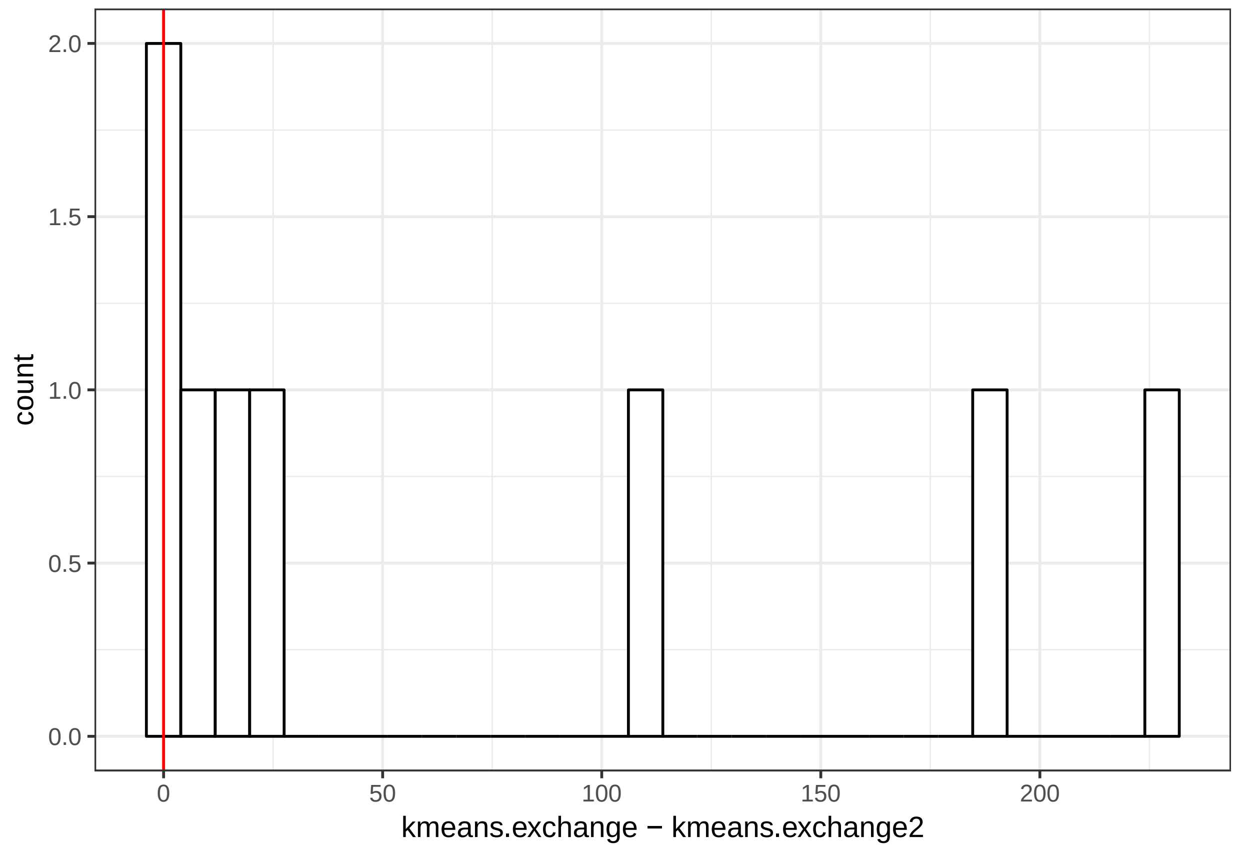
```r
# for the sake of time, going to just look at 8 initial configurations
init.centers %<>% magrittr::extract(seq(8))

# clusterings with 10000 iterations
niter <- 10000
clusterings <- lapply(init.centers, function(i) kmeans.exchange(X, i, niter))
clusterings.2 <- lapply(init.centers, function(i) kmeans.exchange2(X, i, niter))

# within-cluster similarity

W <- sapply(clusterings, function(clustering) clustering$W)
W.2 <- sapply(clusterings.2, function(clustering) clustering$W)

ggplot() +
  geom_point(aes(x = seq(8), y = W, colour = 'kmeans.exchange')) +
  geom_point(aes(x = seq(8), y = W.2, colour = 'kmeans.exchange2')) +
  scale_colour_brewer(palette = 'Set1') +
  labs(x = 'iterations', colour = NULL)
```



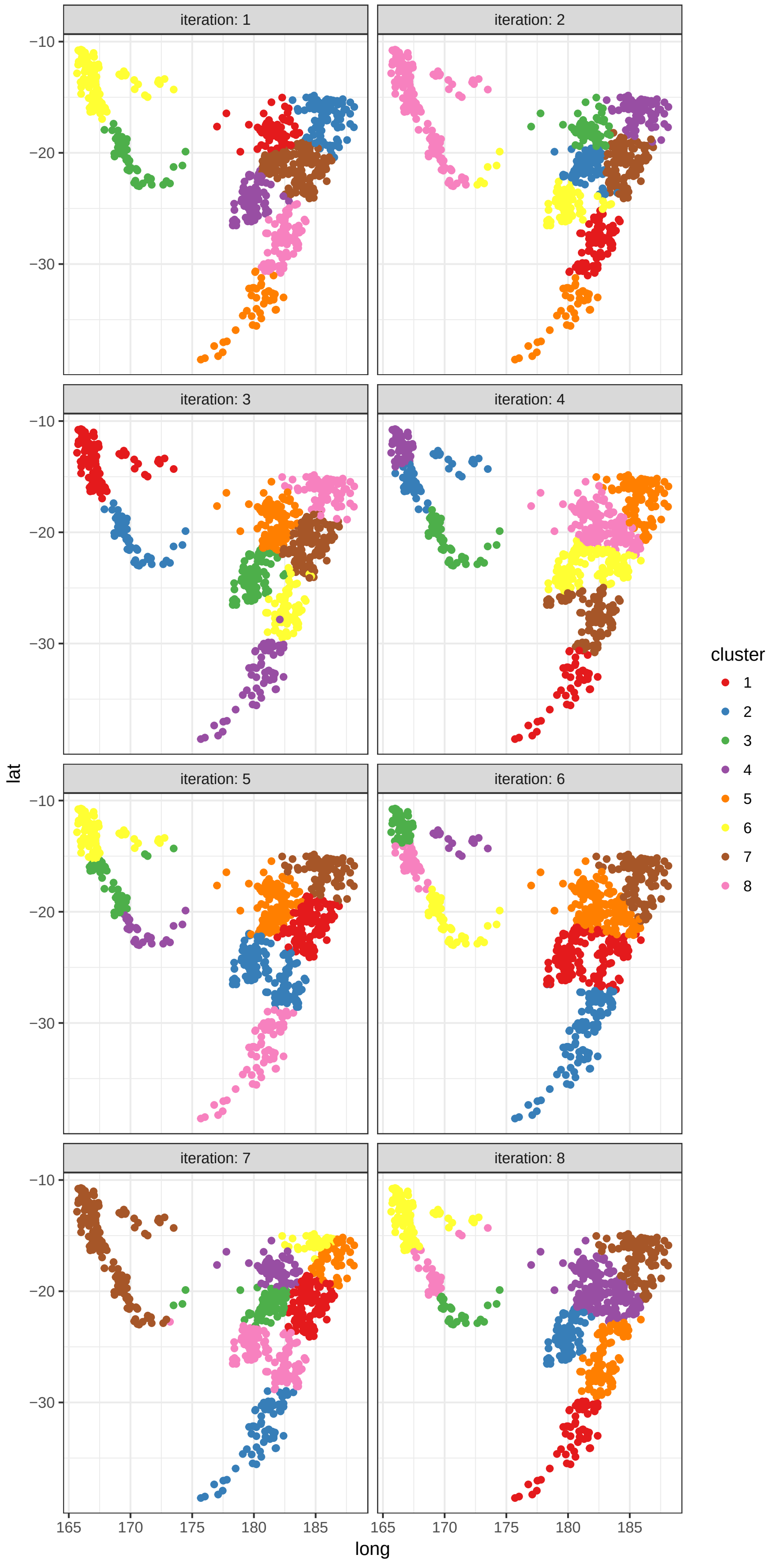
```r
ggplot() +
  geom_point(aes(x = W, y = W.2)) +
  geom_abline(colour = 'red') +
  labs(x = 'kmeans.exchange', y = 'kmeans.exchange2')
```



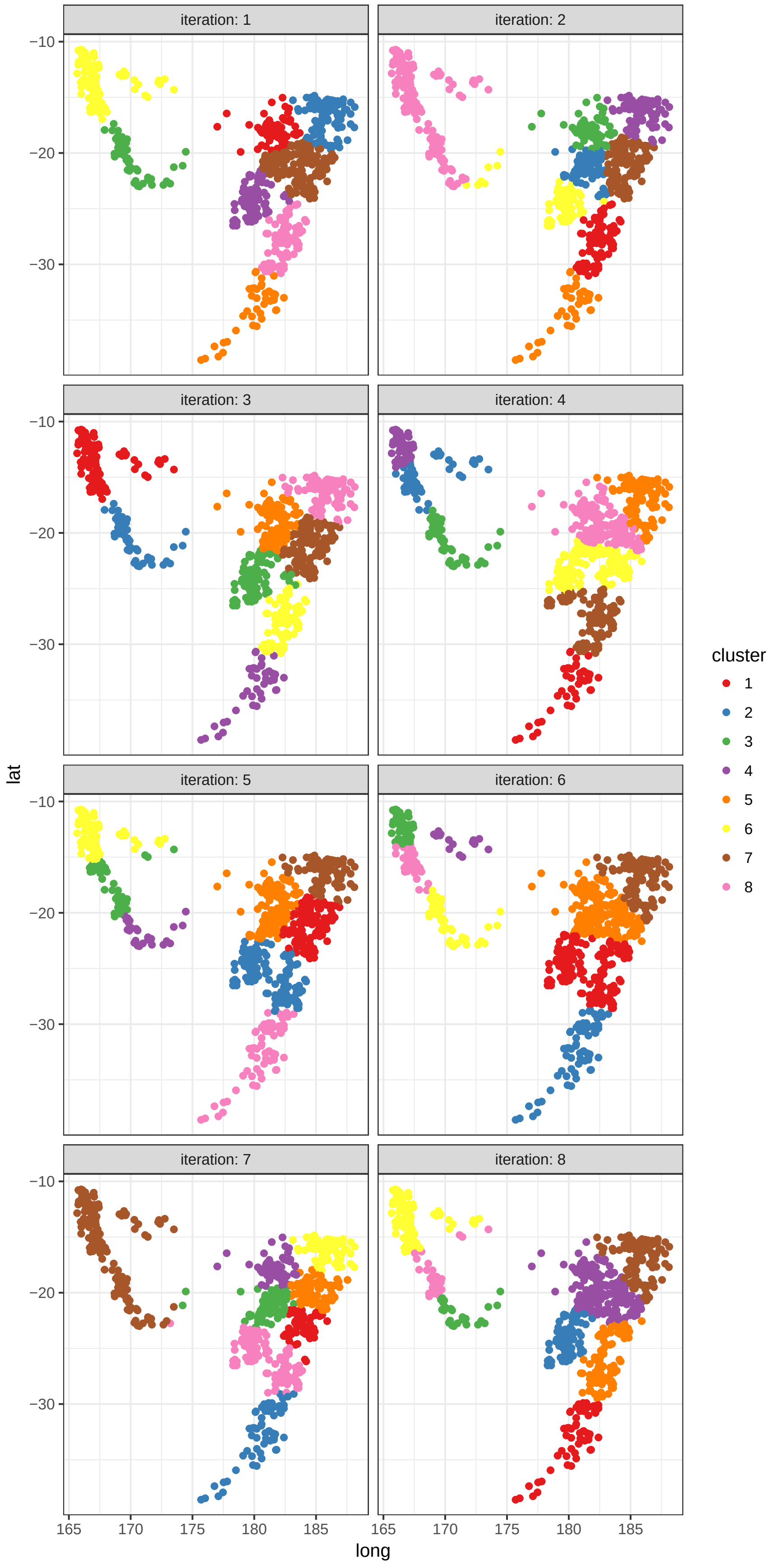
```
ggplot() +
  geom_histogram(aes(x = W - W.2),
                 colour = 'black', fill = 'white') +
  labs(x = 'kmeans.exchange - kmeans.exchange2') +
  geom_vline(xintercept = 0, colour = 'red')
```



```r
# actual clusterings

random.df <- lapply(seq(8), function(i) {
  clusterings[[i]]$X %>%
    as.data.frame() %>%
    dp$transmute(iteration = i,
                 cluster = as.character(V1),
                 long, lat)
}) %>%
  dp$bind_rows()

cyclic.df <- lapply(seq(8), function(i) {
  clusterings.2[[i]]$X %>%
    as.data.frame() %>%
    dp$transmute(iteration = i,
                 cluster = as.character(V1),
                 long, lat)
}) %>%
  dp$bind_rows()

ggplot(random.df) +
  geom_point(aes(x = long, y = lat, colour = cluster)) +
  facet_wrap(~ iteration, ncol = 2, labeller = 'label_both') +
  scale_colour_brewer(palette = 'Set1')
```


iteration: 1
iteration: 2
iteration: 3
iteration: 4
iteration: 5
iteration: 6
iteration: 7
iteration: 8
−10
−20
−30
lat
165
170
175
180
185
long
cluster
1
2
3
4
5
6
7
8

```r
ggplot(cyclic.df) +
  geom_point(aes(x = long, y = lat, colour = cluster)) +
  facet_wrap(~ iteration, ncol = 2, labeller = 'label_both') +
  scale_colour_brewer(palette = 'Set1')
```


iteration: 1
iteration: 2
iteration: 3
iteration: 4
iteration: 5
iteration: 6
iteration: 7
iteration: 8
lat
−10
−20
−30
long
165
170
175
180
185
cluster
1
2
3
4
5
6
7
8

With 10000 iterations, both methods achieve much closer results.

k-means clustering prefers spherical clusters, so some of the longer clusters are separated into two different clusters. We can see this with the first trial, which produced the smallest value of W for both the random and cyclic methods. The left-most group of points, which we might think should be in one long cluster, is separated into two shorter clusters.

Exercise 7

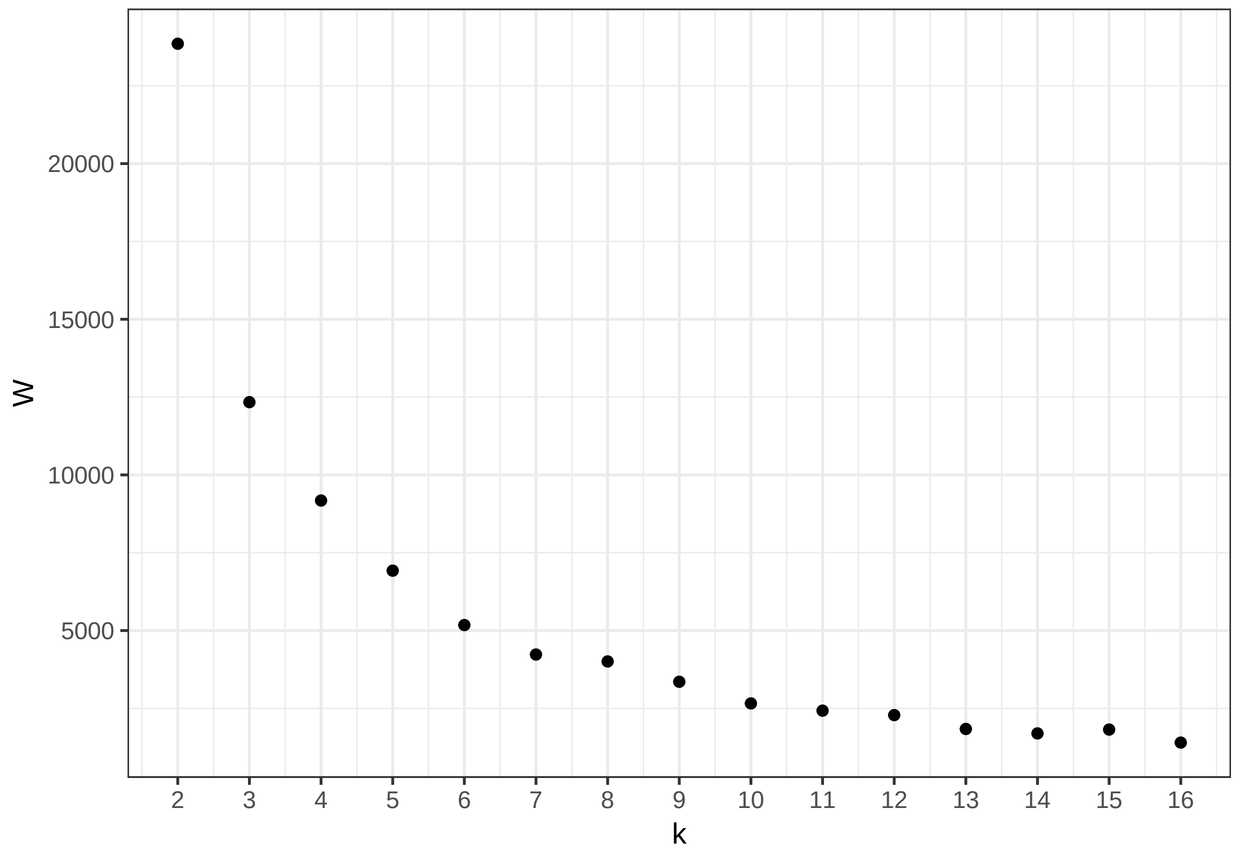
Just from the plots in exercise 2, I would guess 2 clusters. We can try some data-driven methods to test this hypothesis.

For this exercise, I'll just use the cyclic method since it has been shown in previous exercises that it converges faster. We will also try the clusterings 16 times for each choice of k and pick the best clustering for each k.

One thing to note is that k-means clustering isn't perfect. It prefers spherical clusters, which may not be the best choice for our data.

W vs. k

```r
# different values of k
k.vector <- seq(2, 16)

clusterings.k <- lapply(k.vector, function(K) {
  # for each choice of k ...

  # initialize 16 configurations
  init.centers <- lapply(seq(16), function(i) kmeans.start(nrow(X), K))

  # clustering for each initial configuration
  temp.clusterings <- lapply(init.centers, function(init.center) {
    kmeans.exchange2(X, init.center)
  })

  # figure out which one has lowest W
  argmin.W <- which.min(sapply(temp.clusterings, function(i) i$W))

  # return the clustering with the lowest W
  return(temp.clusterings[[argmin.W]])
})

ggplot() +
  geom_point(aes(x = k.vector, y = sapply(clusterings.k, function(i) i$W))) +
  labs(x = 'k', y = 'W') +
  scale_x_continuous(breaks = k.vector)
```



W always decreases as k increases (assuming we iterated to convergence). But perhaps one way of choosing the optimal number of clusters is by looking for the point of diminishing returns, which appears to be around $k = 11$ in this case. However, k-means clustering might not be optimal for these data.

Exercise 8

Implement Lloyd's algorithm

```r
#' @title Lloyd's algorithm for k-means clustering
#' @requires dplyr, magrittr, testthat
#' @param X (data frame) The data to be clustered
#' @param init.centers (numeric) The row numbers of X for which the cluster
#'   centers are initialized
#' @param n.iter (numeric)
#' @return (list) The clustered data, W, k
kmeans.lloyd <- function(X, init.center, n.iter = 100) {
  # how many clusters?
  k <- length(init.center)

  # how many objects?
  N <- nrow(X)

  # initialize
  # find the centers of the clusters
  centers.df <- lapply(init.center, function(i) X[i, ]) %>%
    dplyr::bind_rows()
  # init cluster assignments
  X$cluster <- rep(0, N)

  # iteration counter for breaking out when there's no convergence
  iter.counter <- 0

  # also need an "old centers.df" to compare against current
  # if they're the same then we're done
  old.centers.df <- dplyr::data_frame()

  # keep iterating until old.centers.df is the same as centers.df
  while(!identical(old.centers.df, centers.df)) {
    # check if we converged within set number of iterations
    iter.counter %<>% magrittr::add(1)
    if (iter.counter >= n.iter) {
      warning('did not converge :(')
      break
    }

    # assign each object to a cluster by euclidean distance to centers
    X$cluster <- sapply(seq(N), function(i) {
      sapply(seq(k), function(j) {
        sum((X[i, -ncol(X)] - centers.df[j, ]) ** 2)
      }) %>%
        which.min()
    })

    # copy current centers.df to old.centers.df
    old.centers.df <- centers.df

    # update centers.df
    centers.df <- X %>%
      dplyr::group_by(cluster) %>%
      dplyr::summarise_all(mean) %>%
      dplyr::ungroup() %>%
      dplyr::arrange(cluster) %>%
      dplyr::select(-cluster)
  }

  # compute W
  W <- X %>%
    dplyr::group_by(cluster) %>%
    dplyr::summarise_all(dplyr::funs(sum((. - mean(.)) ** 2))) %>%
    dplyr::ungroup() %>%
    dplyr::select(-cluster) %>%
    rowSums() %>%
    sum()


  return(list(X = X, centers = centers.df, W = W, k = k, niter = iter.counter))
}
```

```
quakes.df <- dp$select(quakes, long, lat)
init.centers <- sample(seq(nrow(quakes.df)), 8)

clustering.lloyd <- kmeans.lloyd(quakes.df, init.centers)
clustering.lloyd$W
```

```
[1] 3502.041
```

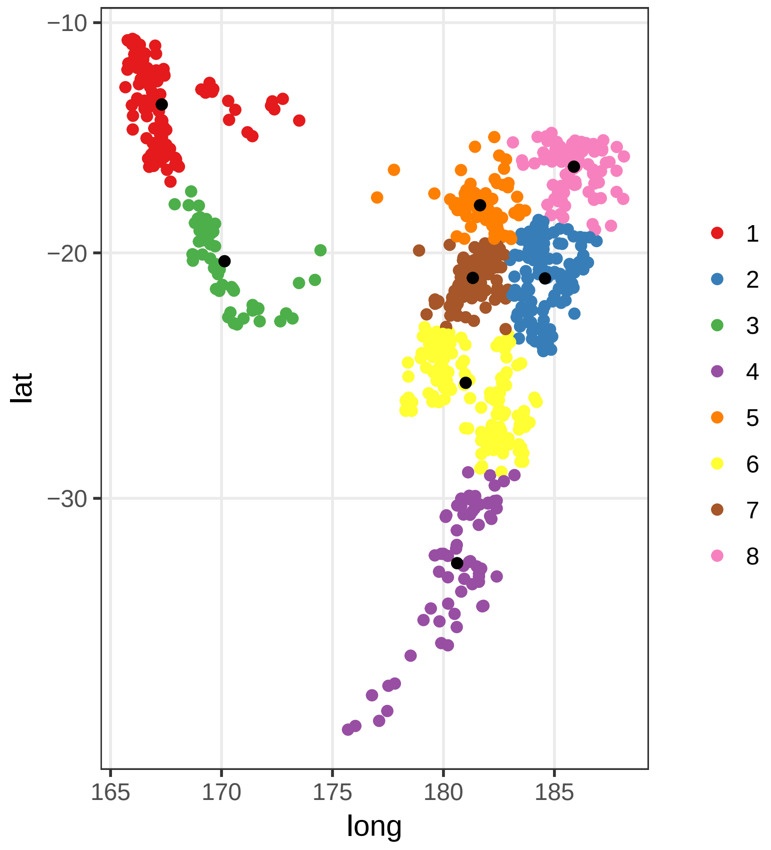
```
clustering.lloyd$k
```

```
[1] 8
```

```
clustering.lloyd$niter
```

```
[1] 11
```

```
ggplot() +
  geom_point(data = clustering.lloyd$X,
             aes(x = long, y = lat, colour = as.character(cluster))) +
  geom_point(data = clustering.lloyd$centers,
             aes(x = long, y = lat)) +
  coord_map() +
  scale_colour_brewer(palette = 'Set1') +
  labs(colour = NULL)
```



Lloyd's algorithm achieved a smaller W and required fewer iterations. However, each iteration took much longer. In the exchange algorithm, each iteration reassigns one point at a time whereas in Lloyd's algorithm, all points are reassigned each iteration.