Currently existing parallel functions in R

Example: replicate

```
x <- cbind(mtcars$wt, mtcars$hp)
y <- replicate(n = 10, expr = x, simplify = F)
```

Parallel version

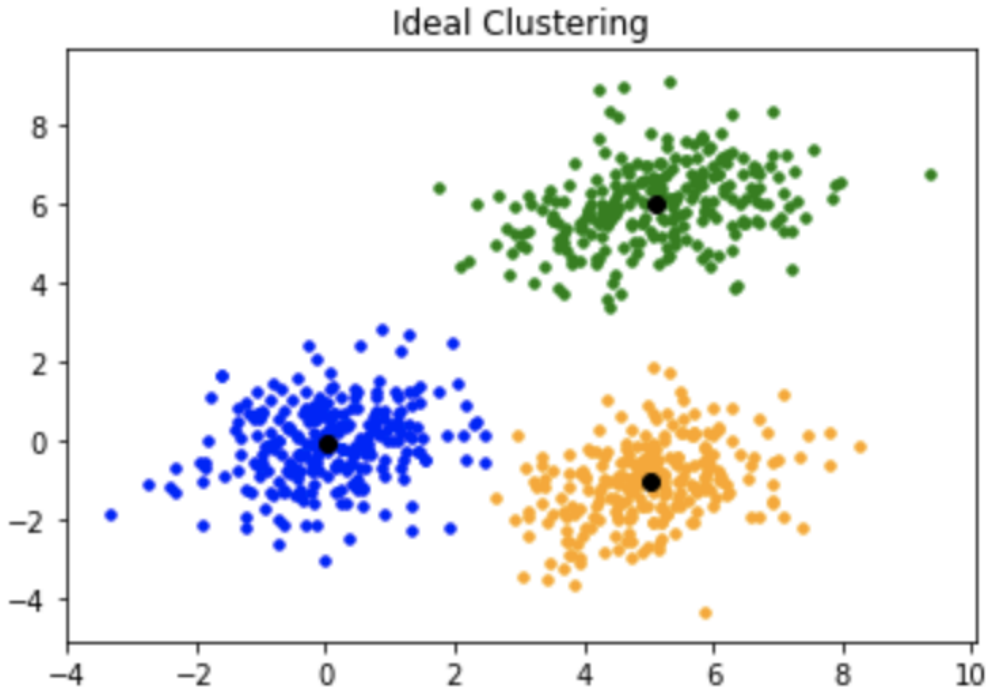
```
clone <- function(dest,source) {
  source
  }
library(parallel)
x <- cbind(mtcars$wt, mtcars$hp)
y <- vector(mode = "list", length = 10)
mclapply(y, mc.cores = [no_cores], clone, source = x)
```

Parallel K-mean

K-means clustering

- Unsupervised learning
- Grouping of data
- Other clustering methods exist

1. Start with **N** centroids
2. Group individual datapoint to nearest centroid
3. Calculate mean of all groups
4. Set new centroid as the mean
5. Repeat 2 to 4 until no change



K-means clustering

Basic function with linear execution

```
library(clusternor)
kmeans(data.frame(),centers=[centroids])
```

Package clusternor provide parallel k-means

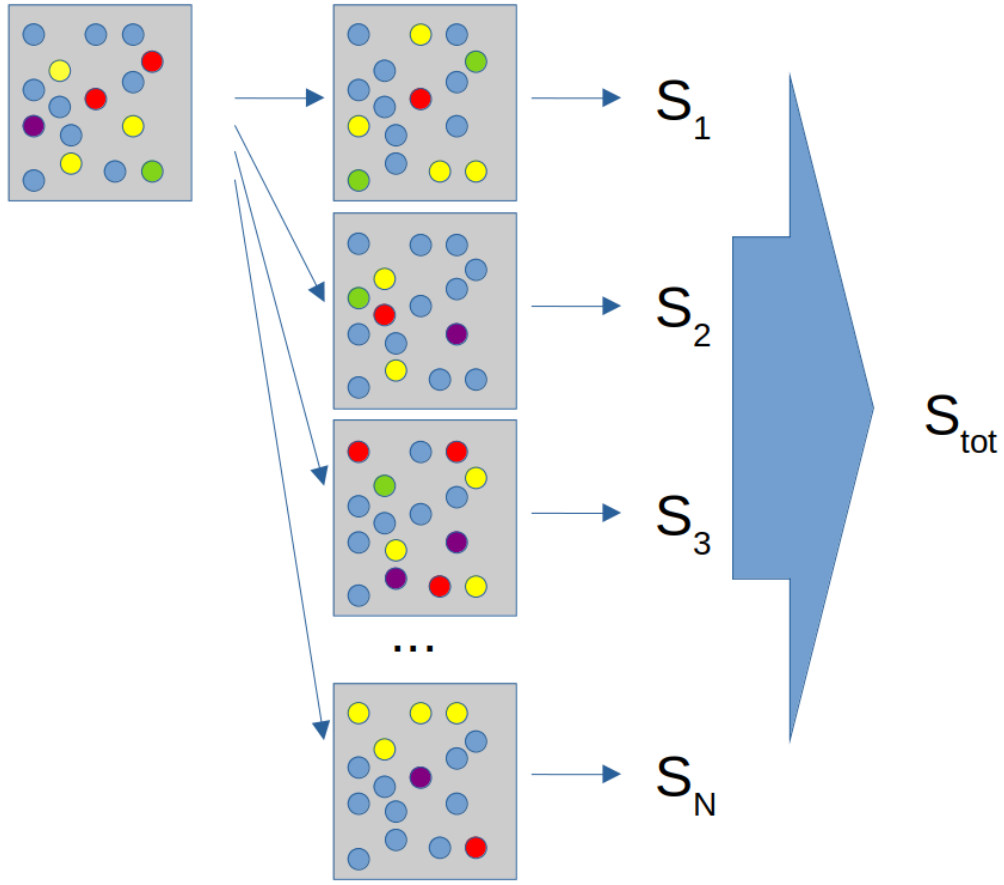
```
library(clusternor)
Kmeans(data.matrix(),centers=[centroids],nthread=[threads])
```

```
library(clusternor)
iris.mat <- as.matrix(iris[,1:4])
k <- length(unique(iris[, dim(iris)[2]])) # Number of unique classes
kms <- Kmeans(iris.mat, k, nthread=1)
```

Parallel bootstrap

What is bootstrap

- Drawing random samples, with replacement, from original samples to create **N** simulated datasets
- Allows for the calculation of standard errors, confidence intervals, etc
- Avoids cost of repeating the experiment to get other groups of sampled data



Bootstrap example

Bootstrap analysis to report confidence interval of cars horsepower

```
library(boot)
Mean <- function(data, idx) {
  c <- data[idx,]
  return(mean(c))
  }
x <- data.frame(mtcars$hp)
res <- boot(x, Mean, R=1000)
plot(res)
ci <- boot.ci(res, type="basic")
sprintf("95%% CI from %f - %f", ci$basic[1,4], ci$basic[1,5])
```

Permutation

- Testing *null hypothesis* under all possible rearrangements of the observed data points
- Are primarily used to provide a p-value
- Drawing random samples, without replacement, from original samples to create **R** datasets

sim	type
ordinary	bootstrap
permutation	permutation

```
boot.ci(x, type="basic", sim="[type]")
```

How to use this function in parallel

Parameter: parallel

type	library	cl
No (Default)		
multicore	parallel	
snow	snow, permutation	cluster

```
boot.ci(x, type="basic", parallel="[type]", ncpus=[cores], cl=[cluster])
```

There might be more already parallel functions

Job Arrays

If you are interested in many single jobs running with different set of parameters, for instance, you may use the **Job Arrays** feature available in *SLURM*

```
#SBATCH --array=1-28
```