Cluster Assignment and Stratified Covariate Constrained Randomization for Spatially-Correlated Data: Euclidean Distances

Patrick Iben

August 1, 2020

In malaria research, interventions are often applied to communities, not individuals, such as spraying of insecticides, distribution of medications to health centers, or bed net campaigns. CCR is a potentially beneficial method for assigning treatments to communities. However, spatial heterogeneity needs to be considered, as mosquitoes and people move frequently, and the potential for treatment contamination is high. Thus, it is desirable to optimize community assignment to clusters that are contiguous, yet still achieve random assignment of clusters to intervention arms. This document provides a method on selecting clusters and treatment assignments for stratified cluster randomized trials, via covariate-constrained randomization and from spatially-correlated data. Three clustering techniques are used: K-Means; Clustering Large Applications (CLARA); Ward Hierarchical Clustering. The distance metric is Euclidean distance. Optimal treatment assignments are determined by: balance on strata, evidence of spatial heterogeneity of treatment assignments, and randomly-selected CCR score. Optimal design selection is based on which of the three selected CCR schemes is the lowest, while also being balanced on the strata and exhibiting spatial heterogeneity, thus helping minimize spatial contamination. All clustering and CCR algorithms for each of the three methods are placed in while loops, with seeds starting at 1. If a selected randomization scheme for a given method is not balanced on strata and exhibiting spatial heterogeneity of treatment arms, the loop starts over with 1 added to the seed and repeat until such a scheme was chosen.

For this data scenario, the three covariates on which we are constraining CRT design are cluster-level: baseline malaria prevalence (via microscopy), household density, and proportion bordering a still body of water.

After loading necessary packages and setting the directory, spatial feature and geo-location data are read in and cleaned, to merge into spatial polygons data frame. After the study data has been merged with the geo-location data, subsets are made of each sub-county (stratum). This will allow us to maintain the separation of sub-counties during cluster optimization.

As CRTs tend to have only two treatment arms for which to randomize clusters (treatment and control), only even number of clusters can be considered to evenly assign units. For this example, the one stratification variable - subcounty - has two levels. Thus, to best assure even allocation of the two trial arms into the two strata levels (though not a guarantee), only cluster sizes divisible by four should be considered. If only four clusters are created, then there are a total of six (4 choose 2) configurations to select randomly from for CCR.

If the default cutoff for cvrall() of 10% is used, then this leaves only one treatment allocation scheme to select from, thus creating a deterministic selection for cluster assignment - and no longer qualifies as a randomized control trial. To remove this possibility, only multiples of four, that are greater than or equal to eight, are considered.

#Setting minimum K for cluster size -  
minclust <- 8   
#Setting maximum K for cluster size to ensure at least 3 units per cluster for buffer  
n\_3 <- round((nrow(study\_data\_wide))/3)  
n\_3\_even <- ifelse(n\_3 %% 2 != 0, n\_3-1, n\_3)#If odd, subtract 1  
n\_3\_divby4 <- ifelse(n\_3\_even %% 4 != 0, n\_3\_even-2, n\_3\_even)#If not divisible by 4, subtract 2  
  
#If ((closest even number under number of units divided by 3) \*3)) is greater than number of units, then subtract 4 from previous line for the maximum cluster size  
maxclust <- ifelse((n\_3\_divby4\*3) > nrow(study\_data\_wide), n\_3\_divby4-4, n\_3\_divby4)  
  
paste("For this example, a maximum of ", maxclust, " clusters will be tested", sep="")

## [1] "For this example, a maximum of 16 clusters will be tested"

####CCR Weights

“cvcrand” allows for custom weighting of covariate importance while performing CCR. For this example, we use standardized regression coefficients from a Poisson cell-means model with an offset via the glm() function. The outcome of interest is the count of positive malaria tests from the most survey data. We are using standardized village-level data (due to the sparsity of cluster-level data); to standardize the data, the scale() function is used. The regression coefficients are standardized such that their absolute values sum to 1 - the relative weight of a given covariate is defined as the absolute value of its model coefficient divided by the sum of the absolute values of all model coefficients. To keep consistent with package author guidelines of weights greater than 1, each weight is then multiplied by 100.

#Scaling covariates (not outcome or offset)  
study\_data\_wide$scaled\_microPos\_1 <- scale(study\_data\_wide$microPos\_1)  
study\_data\_wide$scaled\_BordersWater <- scale(study\_data\_wide$BordersWater)  
study\_data\_wide$scaled\_hhDensity <- scale(study\_data\_wide$hhDensity)  
  
#Creating Poisson model with offset  
weight\_model <- glm(microPos\_5 ~ scaled\_microPos\_1 + scaled\_BordersWater + scaled\_hhDensity+ offset(log(microTests\_5)) -1,   
 data=study\_data\_wide, family = poisson)  
summary(weight\_model)

##   
## Call:  
## glm(formula = microPos\_5 ~ scaled\_microPos\_1 + scaled\_BordersWater +   
## scaled\_hhDensity + offset(log(microTests\_5)) - 1, family = poisson,   
## data = study\_data\_wide)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -11.423 -8.787 -7.753 -6.016 -3.556   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## scaled\_microPos\_1 -0.22025 0.02326 -9.468 < 2e-16 \*\*\*  
## scaled\_BordersWater -0.09691 0.02287 -4.238 2.26e-05 \*\*\*  
## scaled\_hhDensity 0.11830 0.02147 5.510 3.59e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 3557.4 on 55 degrees of freedom  
## Residual deviance: 3321.9 on 52 degrees of freedom  
## AIC: 3487  
##   
## Number of Fisher Scoring iterations: 7

#Scaling and saving model coefficients for CCR weights  
weights\_raw <- c(abs(weight\_model$coefficients[1][[1]])/sum(abs(weight\_model$coefficients)), abs(weight\_model$coefficients[2][[1]])/sum(abs(weight\_model$coefficients)), abs(weight\_model$coefficients[3][[1]])/sum(abs(weight\_model$coefficients)))  
  
#Saving scaled weights  
weights <- weights\_raw \* 100  
  
#Subsetting variables of interest for cluster-level summary stats  
CCR\_covariates <- study\_data\_wide[,c("microPos\_1", "microTests\_1", "BordersWater", "hh", "areakm2", "county")]  
  
#Naming cluster summary statistics for CCR  
#For this data scenario, the overall cluster-level summary statistics are: baseline microscopy rate, household density, and proportion of cluster bordering water.  
Cluster\_SumStats <- c("Cluster\_mRate\_1", "Cluster\_hhDensity", "Cluster\_prop\_BordersWater")   
#Naming strata  
Strata <- "county"  
#Merging names  
CCR\_SumStats\_Strata <- c(Cluster\_SumStats, Strata)

CCR Optimization

Below, all three cluster optimization methods are optimized for stratified, weighted CCR.

K-Means:

kmeans\_best\_ccr<-NA  
seed <- 1  
while (is.na(kmeans\_best\_ccr)){  
  
#Initializing listing of cluster assignments for each value of K  
clustering\_list <- list()  
  
#Collecting cluster quality statistics across range of cluster sizes  
for(i in seq(minclust,maxclust, by = 4)){  
 set.seed(seed)  
 clustering\_list[[i-(minclust-1)]] <- rbind(c(kmeans(geo\_data\_subcounty\_1, i/2)$cluster,kmeans(geo\_data\_subcounty\_3, i/2)$cluster),geo\_data\_subcountyname$subcounty)  
 clustering\_list[[i-(minclust-1)]][1,] <- ifelse(clustering\_list[[i-(minclust-1)]][2,]=="1", clustering\_list[[i-(minclust-1)]][1,] + (i/2), clustering\_list[[i-(minclust-1)]][1,])  
}  
  
#Stripping null list elements  
clustering\_list[sapply(clustering\_list, is.null)] <- NULL  
  
#Adding labels  
names(clustering\_list) <- seq(minclust,maxclust, by= 4)  
  
#Converting from list to matrix  
#Initializing  
kmeans\_cluster\_assignments <- matrix(data = NA, nrow = nrow(geo\_data), ncol = length(clustering\_list), byrow = FALSE,dimnames = NULL)  
  
for(i in 1:length(clustering\_list)){  
 kmeans\_cluster\_assignments[,i] <- clustering\_list[[i]][1,]  
 colnames(kmeans\_cluster\_assignments) <- cat("kmeans\_",max(kmeans\_cluster\_assignments[,i]), sep = "")  
}  
  
colnames(kmeans\_cluster\_assignments) <- paste("kmeans\_", seq(minclust,maxclust, by= 4), sep = "")  
rownames(kmeans\_cluster\_assignments) <- names(clustering\_list[[1]][1,])  
  
#Merging into main dataframe  
study\_data\_wide <- merge(kmeans\_cluster\_assignments, study\_data\_wide,by = "row.names")  
  
study\_data\_wide$Row.names <- NULL  
  
#K-Means CCR  
  
#Adding in village code to merge with lat/long. (for spatial clustering test)  
village\_code <- rownames(kmeans\_cluster\_assignments)  
kmeans\_cluster\_assignments\_df <- data.frame(cbind(kmeans\_cluster\_assignments, village\_code))  
  
#Saving spatial location as df for merging  
geo\_data\_df <- data.frame(geo\_data)  
colnames(geo\_data\_df) <- c("village\_long", "village\_lat")  
geo\_data\_df$village\_code <- rownames(geo\_data\_df)  
  
#Initializing listing of cluster assignments for each value of K  
kmeans\_sumstats <- list()  
kmeans\_ccr <-list()  
kmeans\_spat\_clust <- list()  
trt\_loc <- list()  
split\_trt\_loc <- list()  
spat\_clust <- matrix(data = NA, nrow = ncol(kmeans\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
ccr\_selected <- matrix(data = NA, nrow = ncol(kmeans\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
balanced\_strata <- matrix(data = NA, nrow = ncol(kmeans\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
  
#Saving cluster summary statistics for balancing  
for(i in 1:ncol(kmeans\_cluster\_assignments)){   
kmeans\_sumstats[[i]] <- data.frame(cbind(kmeans\_cluster\_assignments[,i],CCR\_covariates))  
  
kmeans\_sumstats[[i]][,"Cluster\_mRate\_1"] <- with(kmeans\_sumstats[[i]], ave(microPos\_1, kmeans\_sumstats[[i]][,1], FUN=sum)) / with(kmeans\_sumstats[[i]], ave(microTests\_1, kmeans\_sumstats[[i]][,1], FUN=sum))  
  
kmeans\_sumstats[[i]][,"Cluster\_hhDensity"] <- with(kmeans\_sumstats[[i]], ave(hh, kmeans\_sumstats[[i]][,1], FUN=sum)) / with(kmeans\_sumstats[[i]], ave(areakm2, kmeans\_sumstats[[i]][,1], FUN=sum))  
  
kmeans\_sumstats[[i]][,"Cluster\_prop\_BordersWater"] <- with(kmeans\_sumstats[[i]], ave(as.numeric(BordersWater), kmeans\_sumstats[[i]][,1], FUN=sum)) / with(kmeans\_sumstats[[i]], ave(as.numeric(BordersWater), kmeans\_sumstats[[i]][,1], FUN=length))  
  
#Ensuring sorted by clustering  
kmeans\_sumstats[[i]] <- kmeans\_sumstats[[i]][order(kmeans\_sumstats[[i]][,1]),]  
  
#Outputting one row per subgroup summary stats  
kmeans\_sumstats[[i]] <- kmeans\_sumstats[[i]][!duplicated(kmeans\_sumstats[[i]][,1]),]  
  
#CCR  
kmeans\_ccr[[i]] <- cvrall(clustername = kmeans\_sumstats[[i]][,1],  
 balancemetric = "l2",  
 x = data.frame(kmeans\_sumstats[[i]][ ,CCR\_SumStats\_Strata]),  
 ntotal\_cluster = length(kmeans\_sumstats[[i]][,1]),  
 ntrt\_cluster = length(kmeans\_sumstats[[i]][,1])/2,  
 weights = c(weights, 1000),  
 categorical = Strata,  
 cutoff = 0.1,  
 size = 25000000,  
 bhist = F,  
 seed = seed)  
  
#Labeling treatment assignments  
names(kmeans\_ccr[[i]]$allocation) <- c(paste("kmeans\_", length(kmeans\_sumstats[[i]][,1]), sep = ""), paste("kmeans\_", length(kmeans\_sumstats[[i]][,1]), "\_trt", sep = ""))  
  
kmeans\_cluster\_assignments\_df <- merge(kmeans\_cluster\_assignments\_df, kmeans\_ccr[[i]]$allocation)  
kmeans\_cluster\_assignments\_df$village\_code <- as.character(kmeans\_cluster\_assignments\_df$village\_code)  
  
#Merging with left\_join() to keep row order  
kmeans\_cluster\_assignments\_df <- left\_join(geo\_data\_df, kmeans\_cluster\_assignments\_df)  
  
#Saving separate dataframes of treatment assignments with their spatial location data (coordinates)  
trt\_loc[[i]] <- data.frame(cbind(select(kmeans\_cluster\_assignments\_df, ends\_with("\_trt"))[i]), geo\_data\_df)  
  
#Split by treatment group  
split\_trt\_loc[[i]] <- split(trt\_loc[[i]], trt\_loc[[i]][1])  
  
#Testing for significant differences in mean nn-dist with Welch Two Sample t-test  
spat\_clust[i,] <- ifelse(t.test(nndist(split\_trt\_loc[[i]]$`0`$village\_lat, split\_trt\_loc[[i]]$`0`$village\_long), nndist(split\_trt\_loc[[i]]$`1`$village\_lat, split\_trt\_loc[[i]]$`1`$village\_long))$p.value < 0.05, 1, 0)  
  
#Testing for balance on strata  
balanced\_strata[i,] <- ifelse(regmatches(kmeans\_ccr[[i]]$baseline\_table$`arm = 0`[5], gregexpr("(?<=\\().\*?(?=\\))", kmeans\_ccr[[i]]$baseline\_table$`arm = 0`[5], perl=T))[[1]] == "50.0" && regmatches(kmeans\_ccr[[i]]$baseline\_table$`arm = 1`[5], gregexpr("(?<=\\().\*?(?=\\))", kmeans\_ccr[[i]]$baseline\_table$`arm = 1`[5], perl=T))[[1]] == "50.0"  
,1,0)  
  
#Saving selected ccr scores  
ccr\_selected[i,] <- as.numeric(kmeans\_ccr[[i]]$bscores[1,2])  
}  
  
#Combining selected scores, whether or not there is evidence of spatial   
#heterogeneity, and whether or not strata are balanced  
ccr\_all <- data.frame(ccr\_selected,spat\_clust,balanced\_strata)  
  
#Selecting schemes that are balanced on strata and exhibit spatial heterogeneity  
ccr\_bal\_clust <- ccr\_all[which(ccr\_all$spat\_clust == 1 & ccr\_all$balanced\_strata == 1),]  
  
#IDing lowest selected score  
lowest\_index <- as.numeric(rownames(ccr\_bal\_clust[which.min(ccr\_bal\_clust[,1]),]))  
kmeans\_best\_ccr <- ifelse(length(lowest\_index) == 0, NA, kmeans\_ccr[lowest\_index])  
seed <- seed + 1  
}

## kmeans\_8kmeans\_12kmeans\_16kmeans\_8kmeans\_12kmeans\_16kmeans\_8kmeans\_12kmeans\_16

paste("Seed: ",seed,sep = "")

## [1] "Seed: 4"

seed\_kmeans <- seed  
seed\_mat <- data.matrix(seed\_kmeans)  
colnames(seed\_mat) <- "Seed"  
  
#Isolating clustername and treatment arm  
kmeans\_ccr\_mat <- cbind(kmeans\_best\_ccr[[1]]$allocation[1], kmeans\_best\_ccr[[1]]$allocation[2])  
colnames(kmeans\_ccr\_mat) <- c("kmeans\_cluster", "kmeans\_treatment\_arm")  
  
#Saving best configuration  
kmeans\_optimized\_treatments <- as.matrix(kmeans\_cluster\_assignments[,lowest\_index])  
kmeans\_optimized\_treatments <- cbind(kmeans\_optimized\_treatments,rownames(kmeans\_optimized\_treatments))  
colnames(kmeans\_optimized\_treatments) <- c("kmeans\_cluster", "village\_code")  
  
kmeans\_optimized\_treatments <- merge(kmeans\_optimized\_treatments, kmeans\_ccr\_mat, by = "kmeans\_cluster")  
  
K\_kmeans <- max(as.numeric(kmeans\_optimized\_treatments$kmeans\_cluster))  
  
#Adding treatment allocations to clusters for proper labeling with spplot()  
#Cluster assignments  
kmeans\_clustering <- as.matrix(kmeans\_cluster\_assignments[,lowest\_index])  
colnames(kmeans\_clustering) <- "kmeans\_cluster"  
#Adding village codes as a variable for linking  
kmeans\_names <- as.matrix(rownames(kmeans\_clustering))  
colnames(kmeans\_names) <- "village\_code"  
kmeans\_clustering <- as.matrix(cbind(kmeans\_clustering,kmeans\_names))  
  
#Treatment assignments  
kmeans\_treatments <- as.matrix(kmeans\_optimized\_treatments$kmeans\_treatment\_arm)  
rownames(kmeans\_treatments) <- kmeans\_optimized\_treatments$village\_code  
colnames(kmeans\_treatments) <- "Treatment\_Group"  
  
kmeans\_trtnames <- as.matrix(rownames(kmeans\_treatments))  
colnames(kmeans\_trtnames) <- "village\_code"  
  
kmeans\_treatments <- cbind(kmeans\_treatments,kmeans\_trtnames)  
  
#Saving as DFs for merging  
kmeans\_clustering\_df <- data.frame(kmeans\_clustering)  
kmeans\_treatments\_df <- data.frame(kmeans\_treatments)  
  
#Maintaining order of village codes with left\_join from dplyr   
kmeans\_df <- left\_join(kmeans\_clustering\_df, kmeans\_treatments\_df, by ="village\_code")  
  
#Saving as matrix for spatial plots  
kmeans\_treatments <- as.matrix(kmeans\_df[,2:3])  
  
#Labeling  
village\_code <- rownames(geo\_data)  
colnames(geo\_data) <- c("village\_long", "village\_lat")  
#Merging location with unit label  
village\_coords <- data.frame(cbind(geo\_data, village\_code))  
  
#Merging centroids with optimized treatment assignments  
kmeans\_best\_df <- left\_join(kmeans\_df, village\_coords, by ="village\_code")  
#Merging with seed that was ultimately used for configuration  
kmeans\_best\_df <- data.frame(kmeans\_best\_df, seed\_mat)

CLARA:

seed <- 1  
clara\_best\_ccr<-NA  
while (is.na(clara\_best\_ccr)){  
#Initializing listing of cluster assignments for each value of K  
clustering\_list <- list()  
  
#Collecting cluster quality statistics across range of cluster sizes  
for(i in seq(minclust,maxclust, by = 4)){  
 clustering\_list[[i-(minclust-1)]] <- rbind(c(clara(geo\_data\_subcounty\_1, metric = "euclidean", k = i/2)$cluster,clara(geo\_data\_subcounty\_3, metric = "euclidean", k = i/2)$cluster), geo\_data\_subcountyname$subcounty)  
 clustering\_list[[i-(minclust-1)]][1,] <- ifelse(clustering\_list[[i-(minclust-1)]][2,]=="1", clustering\_list[[i-(minclust-1)]][1,] + (i/2), clustering\_list[[i-(minclust-1)]][1,])  
}  
  
  
#Stripping null list elements  
clustering\_list[sapply(clustering\_list, is.null)] <- NULL  
  
#Adding labels  
names(clustering\_list) <- seq(minclust,maxclust, by= 4)  
  
#Converting from list to matrix  
#Initializing  
clara\_cluster\_assignments <- matrix(data = NA, nrow = nrow(geo\_data), ncol = length(clustering\_list), byrow = FALSE,dimnames = NULL)  
  
for(i in 1:length(clustering\_list)){  
 clara\_cluster\_assignments[,i] <- clustering\_list[[i]][1,]  
 colnames(clara\_cluster\_assignments) <- cat("clara\_",max(clara\_cluster\_assignments[,i]), sep = "")  
}  
  
colnames(clara\_cluster\_assignments) <- paste("clara\_", seq(minclust,maxclust, by= 4), sep = "")  
rownames(clara\_cluster\_assignments) <- names(clustering\_list[[1]][1,])  
  
#Merging into main dataframe  
study\_data\_wide <- merge(clara\_cluster\_assignments, study\_data\_wide,by = "row.names")  
  
study\_data\_wide$Row.names <- NULL  
  
#CLARA CCR  
  
#Adding in village code to merge with lat/long. (for spatial clustering test)  
village\_code <- rownames(clara\_cluster\_assignments)  
clara\_cluster\_assignments\_df <- data.frame(cbind(clara\_cluster\_assignments, village\_code))  
  
#Saving spatial location as df for merging  
geo\_data\_df <- data.frame(geo\_data)  
colnames(geo\_data\_df) <- c("village\_long", "village\_lat")  
geo\_data\_df$village\_code <- rownames(geo\_data\_df)  
  
#Initializing listing of cluster assignments for each value of K  
clara\_sumstats <- list()  
clara\_ccr <-list()  
clara\_spat\_clust <- list()  
trt\_loc <- list()  
split\_trt\_loc <- list()  
spat\_clust <- matrix(data = NA, nrow = ncol(clara\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
ccr\_selected <- matrix(data = NA, nrow = ncol(clara\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
balanced\_strata <- matrix(data = NA, nrow = ncol(clara\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
  
#Saving cluster summary statistics for balancing  
for(i in 1:ncol(clara\_cluster\_assignments)){   
clara\_sumstats[[i]] <- data.frame(cbind(clara\_cluster\_assignments[,i],CCR\_covariates))  
  
clara\_sumstats[[i]][,"Cluster\_mRate\_1"] <- with(clara\_sumstats[[i]], ave(microPos\_1, clara\_sumstats[[i]][,1], FUN=sum)) / with(clara\_sumstats[[i]], ave(microTests\_1, clara\_sumstats[[i]][,1], FUN=sum))  
  
clara\_sumstats[[i]][,"Cluster\_hhDensity"] <- with(clara\_sumstats[[i]], ave(hh, clara\_sumstats[[i]][,1], FUN=sum)) / with(clara\_sumstats[[i]], ave(areakm2, clara\_sumstats[[i]][,1], FUN=sum))  
  
clara\_sumstats[[i]][,"Cluster\_prop\_BordersWater"] <- with(clara\_sumstats[[i]], ave(as.numeric(BordersWater), clara\_sumstats[[i]][,1], FUN=sum)) / with(clara\_sumstats[[i]], ave(as.numeric(BordersWater), clara\_sumstats[[i]][,1], FUN=length))  
  
#Ensuring sorted by clustering  
clara\_sumstats[[i]] <- clara\_sumstats[[i]][order(clara\_sumstats[[i]][,1]),]  
  
#Outputting one row per subgroup summary stats  
clara\_sumstats[[i]] <- clara\_sumstats[[i]][!duplicated(clara\_sumstats[[i]][,1]),]  
  
#CCR  
clara\_ccr[[i]] <- cvrall(clustername = clara\_sumstats[[i]][,1],  
 balancemetric = "l2",  
 x = data.frame(clara\_sumstats[[i]][ ,CCR\_SumStats\_Strata]),  
 ntotal\_cluster = length(clara\_sumstats[[i]][,1]),  
 ntrt\_cluster = length(clara\_sumstats[[i]][,1])/2,  
 weights = c(weights, 1000),  
 categorical = Strata,  
 cutoff = 0.1,  
 size = 25000000,  
 bhist = F,  
 seed = seed)  
  
#Labeling treatment assignments  
names(clara\_ccr[[i]]$allocation) <- c(paste("clara\_", length(clara\_sumstats[[i]][,1]), sep = ""), paste("clara\_", length(clara\_sumstats[[i]][,1]), "\_trt", sep = ""))  
  
clara\_cluster\_assignments\_df <- merge(clara\_cluster\_assignments\_df, clara\_ccr[[i]]$allocation)  
clara\_cluster\_assignments\_df$village\_code <- as.character(clara\_cluster\_assignments\_df$village\_code)  
  
#Merging with left\_join() to keep row order  
clara\_cluster\_assignments\_df <- left\_join(geo\_data\_df, clara\_cluster\_assignments\_df)  
  
#Saving separate dataframes of treatment assignments with their spatial location data (coordinates)  
trt\_loc[[i]] <- data.frame(cbind(select(clara\_cluster\_assignments\_df, ends\_with("\_trt"))[i]), geo\_data\_df)  
  
#Split by treatment group  
split\_trt\_loc[[i]] <- split(trt\_loc[[i]], trt\_loc[[i]][1])  
  
#Testing for significant differences in mean nn-dist with Welch Two Sample t-test  
spat\_clust[i,] <- ifelse(t.test(nndist(split\_trt\_loc[[i]]$`0`$village\_lat, split\_trt\_loc[[i]]$`0`$village\_long), nndist(split\_trt\_loc[[i]]$`1`$village\_lat, split\_trt\_loc[[i]]$`1`$village\_long))$p.value < 0.05, 1, 0)  
  
#Testing for balance on strata  
balanced\_strata[i,] <- ifelse(regmatches(clara\_ccr[[i]]$baseline\_table$`arm = 0`[5], gregexpr("(?<=\\().\*?(?=\\))", clara\_ccr[[i]]$baseline\_table$`arm = 0`[5], perl=T))[[1]] == "50.0" && regmatches(clara\_ccr[[i]]$baseline\_table$`arm = 1`[5], gregexpr("(?<=\\().\*?(?=\\))", clara\_ccr[[i]]$baseline\_table$`arm = 1`[5], perl=T))[[1]] == "50.0"  
,1,0)  
  
#Saving selected ccr scores  
ccr\_selected[i,] <- as.numeric(clara\_ccr[[i]]$bscores[1,2])  
}  
  
#Combining selected scores, whether or not there is evidence of spatial   
#heterogeneity, and whether or not strata are balanced  
ccr\_all <- data.frame(ccr\_selected,spat\_clust,balanced\_strata)  
  
#Selecting schemes that are balanced on strata and exhibit spatial heterogeneity  
ccr\_bal\_clust <- ccr\_all[which(ccr\_all$spat\_clust == 1 & ccr\_all$balanced\_strata == 1),]  
  
#IDing lowest selected score  
lowest\_index <- as.numeric(rownames(ccr\_bal\_clust[which.min(ccr\_bal\_clust[,1]),]))  
clara\_best\_ccr <- ifelse(length(lowest\_index) == 0, NA, clara\_ccr[lowest\_index])  
  
seed = seed + 1   
}

## clara\_8clara\_12clara\_16

paste("Seed: ",seed,sep = "")

## [1] "Seed: 2"

#Saving seed  
seed\_clara <- seed  
seed\_mat <- data.matrix(seed\_clara)  
colnames(seed\_mat) <- "Seed"  
  
#Isolating clustername and treatment arm  
clara\_ccr\_mat <- cbind(clara\_best\_ccr[[1]]$allocation[1], clara\_best\_ccr[[1]]$allocation[2])  
colnames(clara\_ccr\_mat) <- c("clara\_cluster", "clara\_treatment\_arm")  
  
#Saving best configuration  
clara\_optimized\_treatments <- as.matrix(clara\_cluster\_assignments[,lowest\_index])  
clara\_optimized\_treatments <- cbind(clara\_optimized\_treatments,rownames(clara\_optimized\_treatments))  
colnames(clara\_optimized\_treatments) <- c("clara\_cluster", "village\_code")  
  
clara\_optimized\_treatments <- merge(clara\_optimized\_treatments, clara\_ccr\_mat, by = "clara\_cluster")  
  
K\_clara <- max(as.numeric(clara\_optimized\_treatments$clara\_cluster))  
  
#Adding treatment allocations to clusters for proper labeling with spplot()  
#Cluster assignments  
clara\_clustering <- as.matrix(clara\_cluster\_assignments[,lowest\_index])  
colnames(clara\_clustering) <- "clara\_cluster"  
#Adding village codes as a variable for linking  
clara\_names <- as.matrix(rownames(clara\_clustering))  
colnames(clara\_names) <- "village\_code"  
clara\_clustering <- as.matrix(cbind(clara\_clustering,clara\_names))  
  
#Treatment assignments  
clara\_treatments <- as.matrix(clara\_optimized\_treatments$clara\_treatment\_arm)  
rownames(clara\_treatments) <- clara\_optimized\_treatments$village\_code  
colnames(clara\_treatments) <- "Treatment\_Group"  
  
clara\_trtnames <- as.matrix(rownames(clara\_treatments))  
colnames(clara\_trtnames) <- "village\_code"  
  
clara\_treatments <- cbind(clara\_treatments,clara\_trtnames)  
  
#Saving as DFs for merging  
clara\_clustering\_df <- data.frame(clara\_clustering)  
clara\_treatments\_df <- data.frame(clara\_treatments)  
  
#Maintaining order of village codes with left\_join from dplyr   
clara\_df <- left\_join(clara\_clustering\_df, clara\_treatments\_df, by ="village\_code")  
  
#Saving as matrix for spatial plots  
clara\_treatments <- as.matrix(clara\_df[,2:3])  
  
#Labeling  
village\_code <- rownames(geo\_data)  
colnames(geo\_data) <- c("village\_long", "village\_lat")  
#Merging location with unit label  
village\_coords <- data.frame(cbind(geo\_data, village\_code))  
  
#Merging centroids with optimized treatment assignments  
clara\_best\_df <- left\_join(clara\_df, village\_coords, by ="village\_code")  
#Merging df with ultimate seed  
clara\_best\_df <- data.frame(clara\_best\_df, seed\_clara)

Ward Hierarchical Clustering:

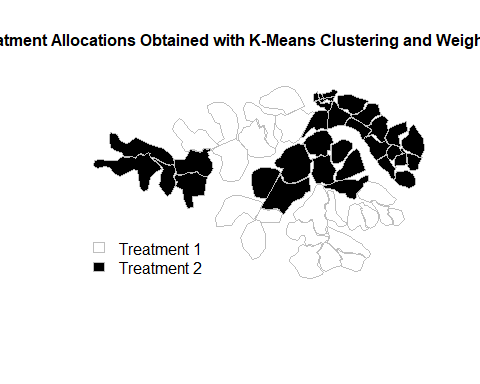
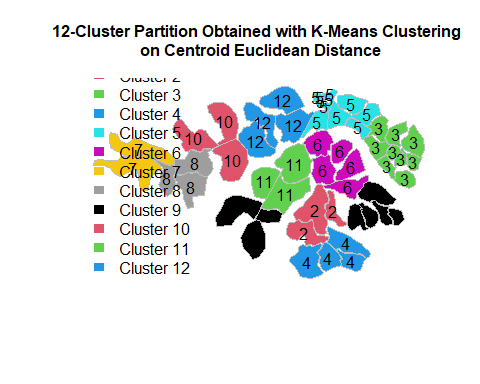
hclust\_best\_ccr<-NA  
seed <- 1  
while (is.na(hclust\_best\_ccr)){  
#Initializing listing of cluster assignments for each value of K  
clustering\_list <- list()  
hclust\_cluster\_assignments <- matrix(data = NA, nrow = nrow(geo\_data), ncol = maxclust-minclust+1, byrow = FALSE,dimnames = NULL)  
colnames(hclust\_cluster\_assignments) <- minclust:maxclust  
  
#Geographical distance  
dist\_1 <- as.dist(as.matrix(daisy(geo\_data\_subcounty\_1, metric = "euclidean")))  
dist\_3 <- as.dist(as.matrix(daisy(geo\_data\_subcounty\_3, metric = "euclidean")))  
  
#Performing hierarchical clustering with Ward.D metric  
tree\_1 <- hclust(dist\_1, "ward.D")  
tree\_3 <- hclust(dist\_3, "ward.D")  
  
#Collecting cluster quality statistic across range of cluster sizes  
for(i in seq(minclust,maxclust, by = 4)){  
 #Making K cuts  
 clustering\_list[[i-(minclust-1)]] <- rbind(c(cutree(tree\_1,i/2),cutree(tree\_3,i/2)),geo\_data\_subcountyname$subcounty)  
 clustering\_list[[i-(minclust-1)]][1,] <- ifelse(clustering\_list[[i-(minclust-1)]][2,]=="1", clustering\_list[[i-(minclust-1)]][1,] + (i/2), clustering\_list[[i-(minclust-1)]][1,])  
}   
  
#Stripping null list elements  
clustering\_list[sapply(clustering\_list, is.null)] <- NULL  
  
#Adding labels  
names(clustering\_list) <- seq(minclust,maxclust, by= 4)  
  
#Converting from list to matrix  
#Initializing  
hclust\_cluster\_assignments <- matrix(data = NA, nrow = nrow(geo\_data), ncol = length(clustering\_list), byrow = FALSE,dimnames = NULL)  
  
for(i in 1:length(clustering\_list)){  
 hclust\_cluster\_assignments[,i] <- clustering\_list[[i]][1,]  
}  
  
colnames(hclust\_cluster\_assignments) <- paste("hclust\_", seq(minclust,maxclust, by= 4), sep = "")  
rownames(hclust\_cluster\_assignments) <- names(clustering\_list[[1]][1,])  
  
#Merging into main dataframe  
study\_data\_wide <- merge(hclust\_cluster\_assignments, study\_data\_wide,by = "row.names")  
  
study\_data\_wide$Row.names <- NULL  
  
#Adding in village code to merge with lat/long. (for spatial clustering test)  
village\_code <- rownames(hclust\_cluster\_assignments)  
hclust\_cluster\_assignments\_df <- data.frame(cbind(hclust\_cluster\_assignments, village\_code))  
  
#Saving spatial location as df for merging  
geo\_data\_df <- data.frame(geo\_data)  
colnames(geo\_data\_df) <- c("village\_long", "village\_lat")  
geo\_data\_df$village\_code <- rownames(geo\_data\_df)  
  
#H-Clust CCR  
  
#Initializing listing of cluster assignments for each value of K  
hclust\_sumstats <- list()  
hclust\_ccr <-list()  
hclust\_spat\_clust <- list()  
trt\_loc <- list()  
split\_trt\_loc <- list()  
spat\_clust <- matrix(data = NA, nrow = ncol(hclust\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
ccr\_selected <- matrix(data = NA, nrow = ncol(hclust\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
balanced\_strata <- matrix(data = NA, nrow = ncol(hclust\_cluster\_assignments), ncol = 1, byrow = FALSE,dimnames = NULL)  
  
#Saving cluster summary statistics for balancing  
for(i in 1:ncol(hclust\_cluster\_assignments)){   
hclust\_sumstats[[i]] <- data.frame(cbind(hclust\_cluster\_assignments[,i],CCR\_covariates))  
  
hclust\_sumstats[[i]][,"Cluster\_mRate\_1"] <- with(hclust\_sumstats[[i]], ave(microPos\_1, hclust\_sumstats[[i]][,1], FUN=sum)) / with(hclust\_sumstats[[i]], ave(microTests\_1, hclust\_sumstats[[i]][,1], FUN=sum))  
  
hclust\_sumstats[[i]][,"Cluster\_hhDensity"] <- with(hclust\_sumstats[[i]], ave(hh, hclust\_sumstats[[i]][,1], FUN=sum)) / with(hclust\_sumstats[[i]], ave(areakm2, hclust\_sumstats[[i]][,1], FUN=sum))  
  
hclust\_sumstats[[i]][,"Cluster\_prop\_BordersWater"] <- with(hclust\_sumstats[[i]], ave(as.numeric(BordersWater), hclust\_sumstats[[i]][,1], FUN=sum)) / with(hclust\_sumstats[[i]], ave(as.numeric(BordersWater), hclust\_sumstats[[i]][,1], FUN=length))  
  
#Ensuring sorted by clustering  
hclust\_sumstats[[i]] <- hclust\_sumstats[[i]][order(hclust\_sumstats[[i]][,1]),]  
  
#Outputting one row per subgroup summary stats  
hclust\_sumstats[[i]] <- hclust\_sumstats[[i]][!duplicated(hclust\_sumstats[[i]][,1]),]  
  
#CCR  
hclust\_ccr[[i]] <- cvrall(clustername = hclust\_sumstats[[i]][,1],  
 balancemetric = "l2",  
 x = data.frame(hclust\_sumstats[[i]][ ,CCR\_SumStats\_Strata]),  
 ntotal\_cluster = length(hclust\_sumstats[[i]][,1]),  
 ntrt\_cluster = length(hclust\_sumstats[[i]][,1])/2,  
 weights = c(weights, 1000),  
 categorical = Strata,  
 cutoff = 0.1,  
 size = 25000000,  
 bhist = F,  
 seed = seed)  
  
#Labeling treatment assignments  
names(hclust\_ccr[[i]]$allocation) <- c(paste("hclust\_", length(hclust\_sumstats[[i]][,1]), sep = ""), paste("hclust\_", length(hclust\_sumstats[[i]][,1]), "\_trt", sep = ""))  
  
hclust\_cluster\_assignments\_df <- merge(hclust\_cluster\_assignments\_df, hclust\_ccr[[i]]$allocation)  
hclust\_cluster\_assignments\_df$village\_code <- as.character(hclust\_cluster\_assignments\_df$village\_code)  
  
#Merging with left\_join() to keep row order  
hclust\_cluster\_assignments\_df <- left\_join(geo\_data\_df, hclust\_cluster\_assignments\_df)  
  
#Saving separate dataframes of treatment assignments with their spatial location data (coordinates)  
trt\_loc[[i]] <- data.frame(cbind(select(hclust\_cluster\_assignments\_df, ends\_with("\_trt"))[i]), geo\_data\_df)  
  
#Split by treatment group  
split\_trt\_loc[[i]] <- split(trt\_loc[[i]], trt\_loc[[i]][1])  
  
#Testing for significant differences in mean nn-dist with Welch Two Sample t-test  
spat\_clust[i,] <- ifelse(t.test(nndist(split\_trt\_loc[[i]]$`0`$village\_lat, split\_trt\_loc[[i]]$`0`$village\_long), nndist(split\_trt\_loc[[i]]$`1`$village\_lat, split\_trt\_loc[[i]]$`1`$village\_long))$p.value < 0.05, 1, 0)  
  
#Testing for balance on strata  
balanced\_strata[i,] <- ifelse(regmatches(hclust\_ccr[[i]]$baseline\_table$`arm = 0`[5], gregexpr("(?<=\\().\*?(?=\\))", hclust\_ccr[[i]]$baseline\_table$`arm = 0`[5], perl=T))[[1]] == "50.0" && regmatches(hclust\_ccr[[i]]$baseline\_table$`arm = 1`[5], gregexpr("(?<=\\().\*?(?=\\))", hclust\_ccr[[i]]$baseline\_table$`arm = 1`[5], perl=T))[[1]] == "50.0"  
,1,0)  
  
#Saving selected ccr scores  
ccr\_selected[i,] <- as.numeric(hclust\_ccr[[i]]$bscores[1,2])  
}  
  
#Combining selected scores, whether or not there is evidence of spatial   
#heterogeneity, and whether or not strata are balanced  
ccr\_all <- data.frame(ccr\_selected,spat\_clust,balanced\_strata)  
  
#Selecting schemes that are balanced on strata and exhibit spatial heterogeneity  
ccr\_bal\_clust <- ccr\_all[which(ccr\_all$spat\_clust == 1 & ccr\_all$balanced\_strata == 1),]  
  
#IDing lowest selected score  
lowest\_index <- as.numeric(rownames(ccr\_bal\_clust[which.min(ccr\_bal\_clust[,1]),]))  
hclust\_best\_ccr <- ifelse(length(lowest\_index) == 0, NA, hclust\_ccr[lowest\_index])  
seed = seed + 1   
}  
  
paste("Seed: ",seed,sep = "")

## [1] "Seed: 2"

#Saving seed  
seed\_hclust <- seed  
seed\_mat <- data.matrix(seed\_hclust)  
colnames(seed\_mat) <- "Seed"  
  
#Isolating clustername and treatment arm  
hclust\_ccr\_mat <- cbind(hclust\_best\_ccr[[1]]$allocation[1], hclust\_best\_ccr[[1]]$allocation[2])  
colnames(hclust\_ccr\_mat) <- c("hclust\_cluster", "hclust\_treatment\_arm")  
  
#Saving best configuration  
hclust\_optimized\_treatments <- as.matrix(hclust\_cluster\_assignments[,lowest\_index])  
hclust\_optimized\_treatments <- cbind(hclust\_optimized\_treatments,rownames(hclust\_optimized\_treatments))  
colnames(hclust\_optimized\_treatments) <- c("hclust\_cluster", "village\_code")  
  
hclust\_optimized\_treatments <- merge(hclust\_optimized\_treatments, hclust\_ccr\_mat, by = "hclust\_cluster")  
  
K\_hclust <- max(as.numeric(hclust\_optimized\_treatments$hclust\_cluster))  
  
  
#Adding treatment allocations to clusters for proper labeling with spplot()  
#Cluster assignments  
hclust\_clustering <- as.matrix(hclust\_cluster\_assignments[,lowest\_index])  
colnames(hclust\_clustering) <- "hclust\_cluster"  
#Adding village codes as a variable for linking  
hclust\_names <- as.matrix(rownames(hclust\_clustering))  
colnames(hclust\_names) <- "village\_code"  
hclust\_clustering <- as.matrix(cbind(hclust\_clustering,hclust\_names))  
  
#Treatment assignments  
hclust\_treatments <- as.matrix(hclust\_optimized\_treatments$hclust\_treatment\_arm)  
rownames(hclust\_treatments) <- hclust\_optimized\_treatments$village\_code  
colnames(hclust\_treatments) <- "Treatment\_Group"  
  
hclust\_trtnames <- as.matrix(rownames(hclust\_treatments))  
colnames(hclust\_trtnames) <- "village\_code"  
  
hclust\_treatments <- cbind(hclust\_treatments,hclust\_trtnames)  
  
#Saving as DFs for merging  
hclust\_clustering\_df <- data.frame(hclust\_clustering)  
hclust\_treatments\_df <- data.frame(hclust\_treatments)  
  
#Maintaining order of village codes with left\_join from dplyr   
hclust\_df <- left\_join(hclust\_clustering\_df, hclust\_treatments\_df, by ="village\_code")  
  
#Saving as matrix for spatial plots  
hclust\_treatments <- as.matrix(hclust\_df[,2:3])  
  
#Labeling  
village\_code <- rownames(geo\_data)  
colnames(geo\_data) <- c("village\_long", "village\_lat")  
#Merging location with unit label  
village\_coords <- data.frame(cbind(geo\_data, village\_code))  
  
#Merging centroids with optimized treatment assignments  
hclust\_best\_df <- left\_join(hclust\_df, village\_coords, by ="village\_code")  
#Merging with ultimate seed  
hclust\_best\_df <- data.frame(hclust\_best\_df, seed\_hclust)

Selecting the best overall clustering optimization method and study design:

best\_scores <- c(as.numeric(kmeans\_best\_ccr[[1]]$bscores[1,2]),as.numeric(clara\_best\_ccr[[1]]$bscores[1,2]),as.numeric(hclust\_best\_ccr[[1]]$bscores[1,2]))  
   
if (min(best\_scores) == best\_scores[1]) {  
overall\_best\_design <- kmeans\_best\_ccr   
overall\_best\_design\_df <- kmeans\_best\_df  
best<-"K-Means"  
K\_best <- K\_kmeans  
best\_clustering <- kmeans\_clustering  
best\_treatments <- kmeans\_treatments  
 paste("The configuration with the lowest selected CCR score, exhibiting spatial heterogeneity of treatment assignments and balance on strata, is ", best, " with ", K\_best, " clusters.",sep = "")  
   
#Summary stats for best selected design  
overall\_best\_design[[1]]$baseline\_table   
  
#Visualizing clusters  
sp::plot(map,border="grey",col=best\_clustering[,1], main= paste(K\_best,"-Cluster Partition Obtained with ", best, " Clustering \n on Centroid Euclidean Distance", sep = ""),cex.main=1)  
legend("bottomleft", legend=paste("Cluster",1:K\_best), fill=1:K\_best, bty="n",border="white")  
text(coordinates(map), labels = best\_clustering[,1])  
  
#Visualizing treatment assignments  
sp::plot(map,border="grey",col= best\_treatments[,2], main= paste("Treatment Allocations Obtained with ", best," Clustering and Weighted CCR", sep = ""),cex.main=1)  
legend("bottomleft", legend=paste("Treatment",1:2), fill=0:1, bty="n",border="grey")  
  
#Saving design  
write.csv(overall\_best\_design[[1]]$baseline\_table, "overall\_best\_ccr\_table\_Euclidean.csv")  
write.csv(overall\_best\_design\_df, "overall\_best\_ccr\_assignments\_Euclidean.csv")  
} else if (min(best\_scores)==best\_scores[2]) {  
overall\_best\_design <- clara\_best\_ccr  
overall\_best\_design\_df <- clara\_best\_df  
best <-"CLARA"  
K\_best <- K\_clara  
best\_clustering <- clara\_clustering  
best\_treatments <- clara\_treatments  
 paste("The configuration with the lowest selected CCR score, exhibiting spatial heterogeneity of treatment assignments and balance on strata, is ", best, " with ", K\_best, " clusters.",sep = "")  
   
#Summary stats for best selected design  
overall\_best\_design$baseline\_table   
  
#Visualizing clusters  
sp::plot(map,border="grey",col=best\_clustering[,1], main= paste(K\_best,"-Cluster Partition Obtained with ", best, " Clustering \n on Centroid Euclidean Distance", sep = ""),cex.main=1)  
legend("bottomleft", legend=paste("Cluster",1:K\_best), fill=1:K\_best, bty="n",border="white")  
text(coordinates(map), labels = best\_clustering[,1])  
  
#Visualizing treatment assignments  
sp::plot(map,border="grey",col= best\_treatments[,2], main= paste("Treatment Allocations Obtained with ", best," Clustering and Weighted CCR", sep = ""),cex.main=1)  
legend("bottomleft", legend=paste("Treatment",1:2), fill=0:1, bty="n",border="grey")  
  
#Saving design  
write.csv(overall\_best\_design[[1]]$baseline\_table, "overall\_best\_ccr\_table\_Euclidean.csv")  
write.csv(overall\_best\_design\_df, "overall\_best\_ccr\_assignments\_Euclidean.csv")  
} else {  
overall\_best\_design <- hclust\_best\_ccr  
overall\_best\_design\_df <- hclust\_best\_df  
best<-"Ward Hierarchical"  
K\_best <- K\_hclust  
best\_clustering <- hclust\_clustering  
best\_treatments <- hclust\_treatments  
 paste("The configuration with the lowest selected CCR score, exhibiting spatial heterogeneity of treatment assignments and balance on strata, is ", best, " with ", K\_best, " clusters.",sep = "")  
  
#Saving design  
write.csv(overall\_best\_design[[1]]$baseline\_table, "overall\_best\_ccr\_table\_Euclidean.csv")  
write.csv(overall\_best\_design\_df, "overall\_best\_ccr\_assignments\_Euclidean.csv")  
}



#Saving visualization of clusters  
png("best\_cluster\_assignments\_Euclidean.png")  
sp::plot(map,border="grey",col=best\_clustering[,1], main= paste(K\_best,"-Cluster Partition Obtained with ", best, " Clustering \n on Centroid Euclidean Distance", sep = ""),cex.main=1.45)  
legend("bottomleft", legend=paste("Cluster",1:K\_best), fill=1:K\_best, bty="n",border="white")  
text(coordinates(map), labels = best\_clustering[,1])  
dev.off()

## png   
## 2

#Saving visualization of treatment assignments  
png(file="best\_ccr\_assignments\_Euclidean.png")  
sp::plot(map,border="grey",col= best\_treatments[,2], main= paste("Treatment Allocations Obtained with \n ", best," Clustering and Weighted CCR", sep = ""),cex.main=1.45)  
legend("bottomleft", legend=paste("Treatment",1:2), fill=0:1, bty="n",border="grey")  
dev.off()

## png   
## 2

#Treatment assignments and summary stats  
overall\_best\_design\_df

## kmeans\_cluster village\_code Treatment\_Group village\_long  
## 1 8 01\_1 1 33.8830960935419  
## 2 12 02\_1 0 33.9237745011031  
## 3 8 03\_1 1 33.8853283079411  
## 4 10 04\_1 0 33.8842217278614  
## 5 7 05\_1 1 33.8535994357203  
## 6 9 06\_1 0 33.9190695733212  
## 7 9 07\_1 0 33.9094579270079  
## 8 10 08\_1 0 33.9066931683657  
## 9 11 09\_1 1 33.942453536588  
## 10 9 10\_1 0 33.9278261715395  
## 11 11 11\_1 1 33.9372169816813  
## 12 8 12\_1 1 33.8692483873845  
## 13 12 13\_1 0 33.9201148561385  
## 14 10 14\_1 0 33.9023331821086  
## 15 12 15\_1 0 33.9415894197082  
## 16 11 16\_1 1 33.9248362713632  
## 17 12 17\_1 0 33.9354806841561  
## 18 7 18\_1 1 33.8498574790397  
## 19 5 37\_3 1 33.966671464021  
## 20 4 36\_3 0 33.9619671957218  
## 21 3 35\_3 1 34.0006254831978  
## 22 6 34\_3 1 33.9730949660441  
## 23 6 33\_3 1 33.9747422876993  
## 24 3 32\_3 1 34.0103138982819  
## 25 3 31\_3 1 33.9990138786655  
## 26 6 30\_3 1 33.9589069905711  
## 27 2 29\_3 0 33.9482127865288  
## 28 4 28\_3 0 33.9496723594713  
## 29 6 27\_3 1 33.9687152166616  
## 30 3 26\_3 1 34.0106867025061  
## 31 3 25\_3 1 34.0057246599155  
## 32 5 24\_3 1 33.955476002428  
## 33 5 23\_3 1 33.9842789779899  
## 34 5 22\_3 1 33.9788577264304  
## 35 5 21\_3 1 33.9747828644178  
## 36 1 20\_3 0 33.9853036860516  
## 37 1 19\_3 0 33.993213497593  
## 38 3 18\_3 1 34.0032785309383  
## 39 1 17\_3 0 34.000618541207  
## 40 6 16\_3 1 33.9561589576021  
## 41 2 15\_3 0 33.9645248377678  
## 42 1 14\_3 0 33.9775684598162  
## 43 2 13\_3 0 33.9540215577645  
## 44 1 12\_3 0 33.9889393954854  
## 45 3 11\_3 1 33.9893702094293  
## 46 3 10\_3 1 33.990865309573  
## 47 3 09\_3 1 33.9955196367327  
## 48 4 08\_3 0 33.9751381376352  
## 49 4 07\_3 0 33.9718113178386  
## 50 5 01\_3 1 33.9545871363619  
## 51 5 02\_3 1 33.9579037373227  
## 52 5 03\_3 1 33.957243694366  
## 53 5 04\_3 1 33.9596581739821  
## 54 5 05\_3 1 33.9625681454407  
## 55 5 06\_3 1 33.962515145531  
## village\_lat Seed  
## 1 1.70374990016726 4  
## 2 1.74094398764963 4  
## 3 1.71698668478128 4  
## 4 1.7316940814823 4  
## 5 1.72510693544983 4  
## 6 1.67367191092344 4  
## 7 1.69041737822697 4  
## 8 1.71894201476505 4  
## 9 1.71677712716864 4  
## 10 1.68681878913766 4  
## 11 1.69925343183211 4  
## 12 1.7087233579221 4  
## 13 1.72967484653754 4  
## 14 1.74150794401599 4  
## 15 1.73904620047961 4  
## 16 1.70686362008421 4  
## 17 1.75331008578031 4  
## 18 1.71449420423106 4  
## 19 1.74379064715084 4  
## 20 1.66290414082828 4  
## 21 1.7339478639296 4  
## 22 1.70366111123287 4  
## 23 1.71520803509617 4  
## 24 1.71789151388862 4  
## 25 1.72338221389036 4  
## 26 1.71311837951238 4  
## 27 1.67719274015451 4  
## 28 1.66079312898661 4  
## 29 1.72412123879957 4  
## 30 1.7290194198793 4  
## 31 1.70801979385237 4  
## 32 1.74079783225925 4  
## 33 1.74523978255956 4  
## 34 1.73762242569853 4  
## 35 1.75126112176979 4  
## 36 1.68710115298599 4  
## 37 1.68808586973452 4  
## 38 1.71933948848979 4  
## 39 1.6867034792363 4  
## 40 1.72837477751585 4  
## 41 1.689803076147 4  
## 42 1.68581351800363 4  
## 43 1.69015365854076 4  
## 44 1.69934576002351 4  
## 45 1.72626796579083 4  
## 46 1.73716043408116 4  
## 47 1.71530204212957 4  
## 48 1.66110042808835 4  
## 49 1.67120098521374 4  
## 50 1.75494535606699 4  
## 51 1.75301896895237 4  
## 52 1.7553538958109 4  
## 53 1.75017937060329 4  
## 54 1.75665408189818 4  
## 55 1.75395988826288 4

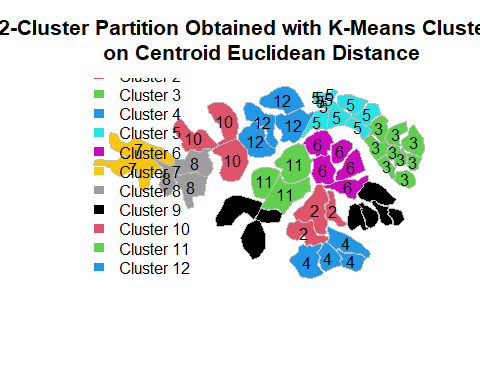
#Optimal selected configuration treatment table  
overall\_best\_design[[1]]$baseline\_table

## arm = 0 arm = 1  
## n 6 6  
## Cluster\_mRate\_1 (mean (SD)) 0.24 (0.06) 0.24 (0.03)  
## Cluster\_hhDensity (mean (SD)) 51.90 (9.52) 54.43 (15.85)  
## Cluster\_prop\_BordersWater (mean (SD)) 0.51 (0.42) 0.58 (0.35)  
## county = 3 (%) 3 (50.0) 3 (50.0)

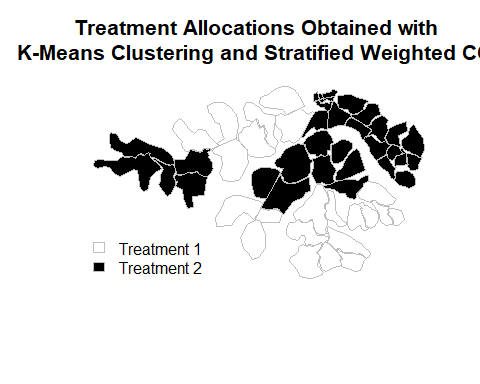
#CCR scores  
overall\_best\_design[[1]]$bscores

##   
## 1 score (selected scheme) 429.534  
## 2 cutoff score 2941.384  
## 3 Mean 3011374.428  
## 4 SD 4087601.446  
## 5 Min 111.955  
## 6 5% 1578.637  
## 7 10% 3060.958  
## 8 20% 6670.574  
## 9 25% 8524.586  
## 10 30% 10588.121  
## 11 50% 3669107.691  
## 12 75% 3678553.360  
## 13 95% 14678305.852  
## 14 Max 33032641.563

#Visualization of clusters  
sp::plot(map,border="grey",col=best\_clustering[,1], main= paste(K\_best,"-Cluster Partition Obtained with ", best, " Clustering \n on Centroid Euclidean Distance", sep = ""),cex.main=1.3)  
legend("bottomleft", legend=paste("Cluster",1:K\_best), fill=1:K\_best, bty="n",border="white")  
text(coordinates(map), labels = best\_clustering[,1])



#Visualization of treatment assignments  
sp::plot(map,border="grey",col= best\_treatments[,2], main= paste("Treatment Allocations Obtained with \n ", best," Clustering and Stratified Weighted CCR", sep = ""),cex.main=1.3)  
legend("bottomleft", legend=paste("Treatment",1:2), fill=0:1, bty="n",border="grey")



References

[1] Moulton LH. Covariate-based constrained randomization of group-randomized trials. Clinical Trials 2004;1(3):297–305. <https://doi.org/10.1191/1740774504cn024oa>, pMID: 16279255.

[2] Dickinson LM, Beaty B, Fox C, Pace W, Dickinson WP, Emsermann C, et al. Pragmatic Cluster Randomized Trials Using Covariate Constrained Randomization: A Method for Practice-based Research Networks (PBRNs). The Journal of the American Board of Family Medicine 2015;28(5):663–672. <https://jabfm.org/content/28/5/663>.

[3] Keogh-Brown M, Bachmann M, Shepstone L, Hewitt C, Howe A, Ramsay C, et al. Contamination in trials of educational interventions. Health technology assessment (Winchester, England) 2007 11;11:iii, ix–107.

[4] Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge University Press; 2018.

[5] Kaufman L, Rousseeuw P. Finding Groups in Data: An Introduction To Cluster Analysis; 1990.

[6] Bishop CM. Neural Networks for Pattern Recognition. New York, NY, USA: Oxford University Press, Inc.; 1995.

[7] Distance;. <http://mathworld.wolfram.com/Distance.html>.

[8] Jin X, Han J. In: Sammut C, Webb GI, editors. K-Medoids Clustering Boston, MA: Springer US; 2010. p. 564–565. <https://doi.org/10.1007/978-0-387-30164-8_426>.

[9] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria; 2018, <https://www.R-project.org/>.

[10] Raab GM, Butcher I. Balance in cluster randomized trials. Statistics in Medicine 2001;20(3):351–365.

[11] Yu H, Li F, Gallis JA, Turner EL. cvcrand: Efficient Design and Analysis of Cluster Randomized Trials; 2019, <https://CRAN.R-project.org/package=cvcrand>, r package version 0.0.3.

[12] Elizabeth A Albright P, Two Independent Samples Unequal Variance (Welch’s Test). Duke; 2018. <https://sites.nicholas.duke.edu/statsreview/means/welch/>.

[13] Ram A, Jalal S, Jalal AS, Kumar M. A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases. International Journal of Computer Applications 2010;3(6):1–4.

[14] Kamil I, Almamory S. Data Partitioning Technique to Enhance DBSCAN Clustering Algorithm 2018 03;Vol.(25).

[15] Hahsler M, Piekenbrock M, Doran D. dbscan: Fast Density-Based Clustering with R. Journal of Statistical Software 2019;91(1):1–30.

[16] Pebesma EJ, Bivand RS. Classes and methods for spatial data in R. R News 2005 November;5(2):9–13. <https://CRAN.Rproject.org/doc/Rnews/>.

[17] Bivand RS, Pebesma E, Gomez-Rubio V. Applied spatial data analysis with R, Second edition. Springer, NY; 2013. <http://www.asdar-book.org/>.

[18] Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K. cluster: Cluster Analysis Basics and Extensions; 2018, r package version 2.0.7-1.

[19] Hartigan JA, Wong MA. Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society Series C (Applied Statistics) 1979;28(1):100–108. <http://www.jstor.org/stable/2346830>.

[20] Baddeley A, Turner R. spatstat: An R Package for Analyzing Spatial Point Patterns. Journal of Statistical Software 2005;12(6):1–42. <http://www.jstatsoft.org/v12/i06/>.