Investigating fixture propensity in a school sports network using latent space models – Supporting material

Ian Hamilton

2020-04-19

Retrieving the data

The aim of this supporting document is to provide enough information that a reader of this article could readily recreate the results presented in the paper. It does not reproduce all figures and tables contained therein. It proceeds in an order that matches the order of the paper with sections matching those of the paper.

The data can be found in four CSV files:

- school_data.csv - contains the data for individual schools including:
  - private/state (P_S),
  - number of terms (Terms),
  - fees in £ (Fees),
  - year of foundation (Founded),
  - number of boys in sixth form (B_6th),
  - proportion of boys (Pct_Boy),
  - proportion of boarders (Pct_Boarder)
  - rating (Rating),
  - postcode,
  - longitude,
  - latitude
- match_matrix.csv - contains the number of matches between teams over three seasons
- travel_time.csv - contains the travel time in minutes between schools based on googlemaps
- travel_distance.csv - contains the travel distance in minutes between schools based on googlemaps

The travel times and distances are supplied here as they are based on estimations made in the first week of June 2019 based on a future date of 5th October 2019, and are therefore not recreatable. The travel_distance data are not used in this document but are mentioned in the paper and so are provided here.

We begin by importing and tidying the required data. The match matrix contains all matches played over three seasons and includes five instances of teams playing each other twice in a single season. For reasons given in the paper it is the number of seasons out of the three in which two teams play each other that are modeled and so the elements of match_matrix are capped at three.

```
school_data <- read.csv("school_data.csv", header = TRUE)
school_names <- school_data[,1]
school_data <- school_data[,-1]
rownames(school_data) <- school_names

match_matrix <- read.csv("match_matrix.csv", header = TRUE)
rownames(match_matrix) <- school_names
match_matrix <- replace(match_matrix,match_matrix>3,3)
```

```r
travel_time <- read.csv("travel_time.csv", header = TRUE)
rownames(travel_time) <- school_names
```

We also introduce functions that will be used later in this document to: 1. convert a list of coordinates into a distance matrix, 2. convert a matrix into a list of edges with values, 3. convert a list of nodal covariate values into a list of edges with absolute covariate distances.

```r
# function to create matrix of Euclidean distances from 2d coordinates
modeldist <- function (coords) {
  n <- nrow(coords)
  dist_matrix <- matrix(0,nrow=n, ncol=n)
  for ( i in 1:(n-1)) {
    for ( j in (i+1):n) {
      dist_matrix[i,j] <- ((coords[i,1]-coords[j,1])^2+(coords[i,2]-coords[j,2])^2)^0.5}
  }
  dist_matrix <- dist_matrix + t(dist_matrix)
  rownames(dist_matrix) <- rownames(coords)
  colnames(dist_matrix) <- rownames(coords)
  return(dist_matrix)
}

# function to transform (symmetrical) matrix to list
matrix_to_edgelist <- function (adjmatrix) {
  n <- nrow(adjmatrix)
  edgelist <- data.frame(0,nrow=1,ncol=3)
  for ( i in 1:(n-1)){
    for ( j in (i+1):n) {
      edgelist <- rbind(edgelist, c(rownames(adjmatrix)[i],rownames(adjmatrix)[j],as.numeric(adjmatrix[
    }
  }
  edgelist <- edgelist[-1,]
  edgelist[,3] <- as.numeric(edgelist[,3])
  return(edgelist)
}

# function to create edgelist of absolute covariate differences
nodedata_to_edgelist <- function (nodedata) {
  m <- nrow(nodedata)
  n <- ncol(nodedata)
  edgelist <- data.frame(matrix(0, nrow = 1,ncol = n+2))
  for ( i in 1:(m-1)){
    for ( j in (i+1):m) {
      new_row <- c(rownames(nodedata)[i],rownames(nodedata)[j])
      for ( k in 1:n) {
        new_row <- append(new_row, abs(nodedata[i,k]-nodedata[j,k]))
      }
      edgelist <- rbind(edgelist, new_row)
    }
  }
  edgelist <- edgelist[-1,]
  col_names <- c("Team i","Team j")
  for ( k in 1:n) {
    col_names <- append(col_names, colnames(nodedata)[k])
  }
```

```
  colnames(edgelist) <- col_names
  return(edgelist)
}
```

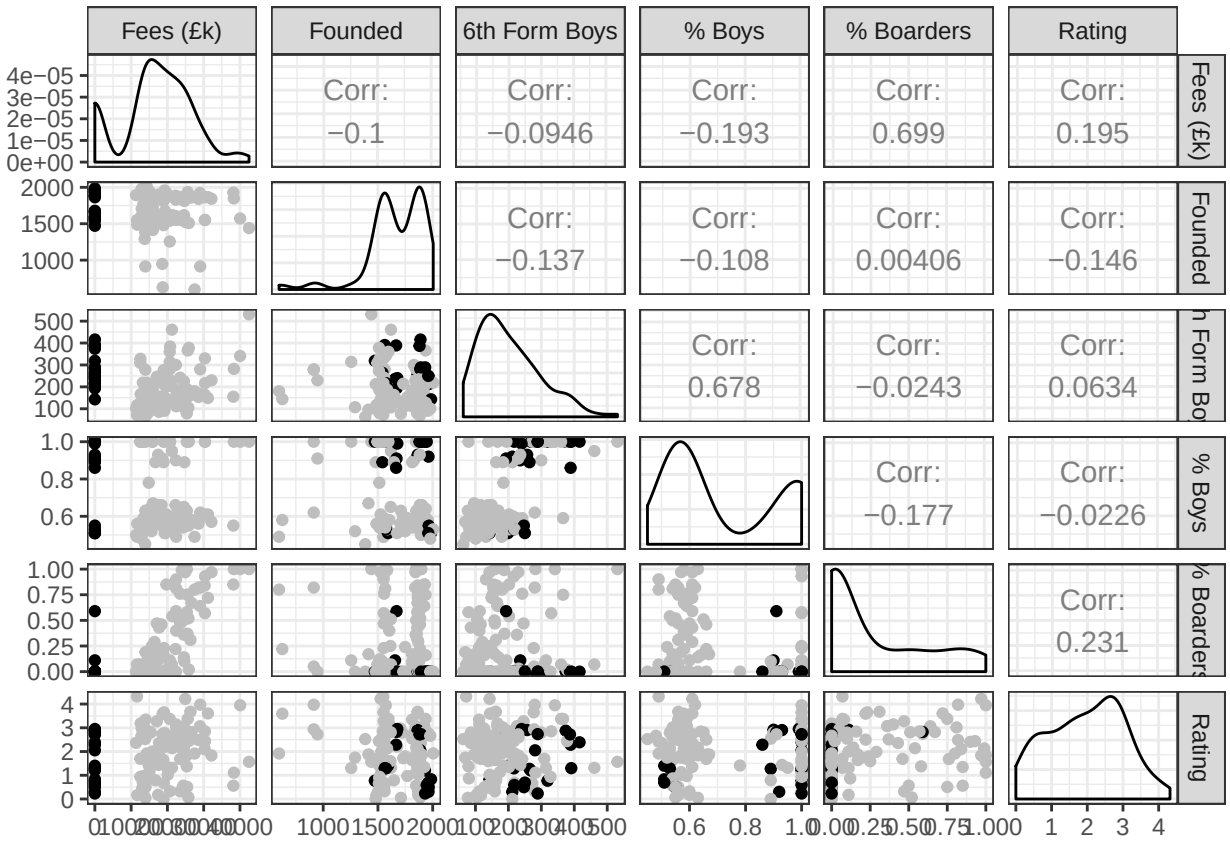
3 Data

3.3 Exploratory Data Analysis

We create a summary plot using the ggpairs function.

```
library(GGally) # also loads ggplot2
```

```
p <- ggpairs(school_data, lower = list(mapping = aes(color = P_S)),
             columns = c("Fees", "Founded", "B_6th", "Pct_Boy", "Pct_Boarder", "Rating"),
        columnLabels = c("Fees (£k)", "Founded", "6th Form Boys", "% Boys", "% Boarders", "Rating")) +
        theme_bw()
for(i in 1:p$nrow) {
  for(j in 1:p$ncol){
    p[i,j] <- p[i,j] +
      scale_colour_manual(values = c("grey","black"))
  }
}

p
```



4 Latent Space Model

4.2 Hierarchical Clustering

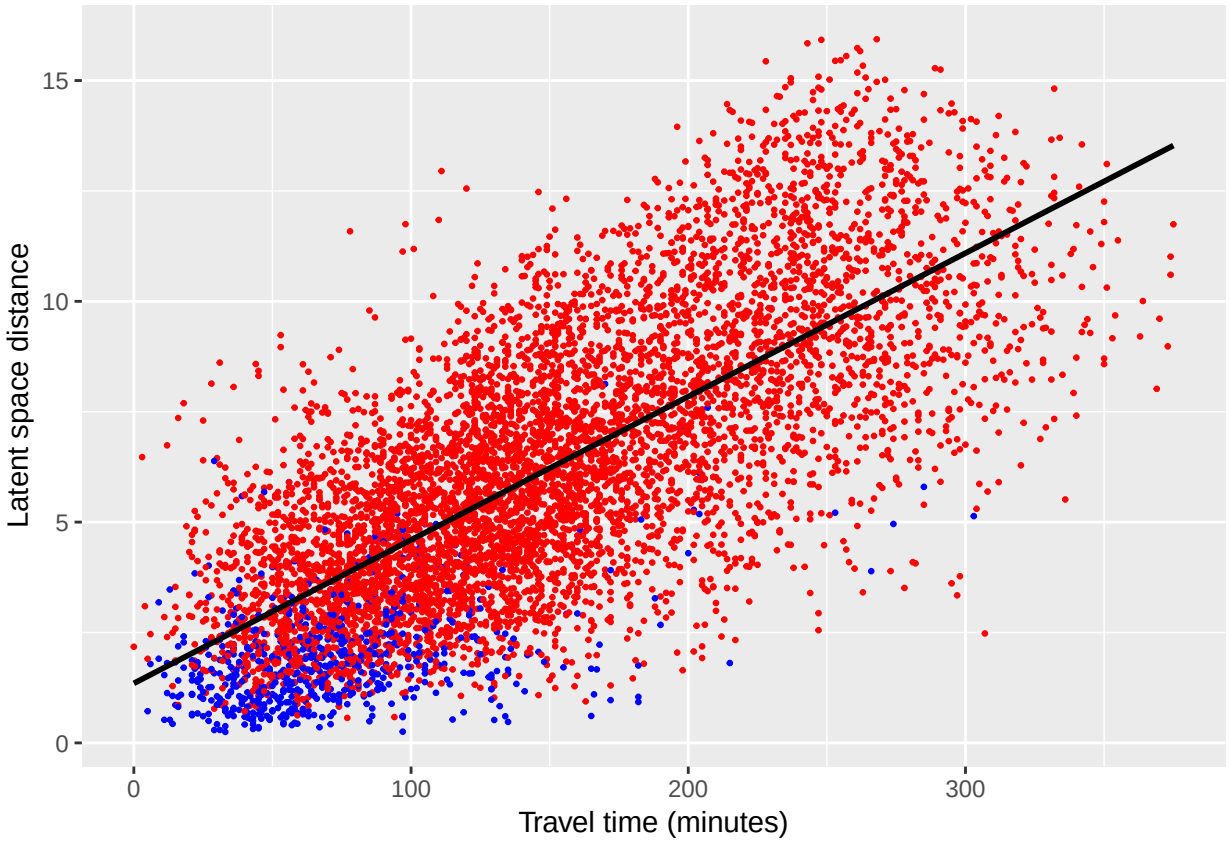
Here we fit the model and extract the latent space coordinates. In order to do this, match_matrix is converted to a network type as recognised by the ergmm function in latentnet. The model fitted is that with dimension = 2, zero clusters, from the binomial family with n=3 using maximum likelihood estimation. Going forward in this document a suffix convention (e.g. _mle_0) based on computation method (mle, mcmc), and number of groups (0,2,3,4,5,6,7) will be employed.

```
library(latentnet)  # used for the fitting
library(network)    # used to create network object required by latentnet function
```

```
g <- network(match_matrix, directed=FALSE)

fit_mle_0 <- ergmm(g ~ euclidean(d=2), family = "binomial", fam.par = list(trials=3), tofit = c("mle"),

# extract latent space coordinates
summary_mle_0 <- summary(fit_mle_0)
coords_mle_0 <- (summary_mle_0$mle)$Z
```

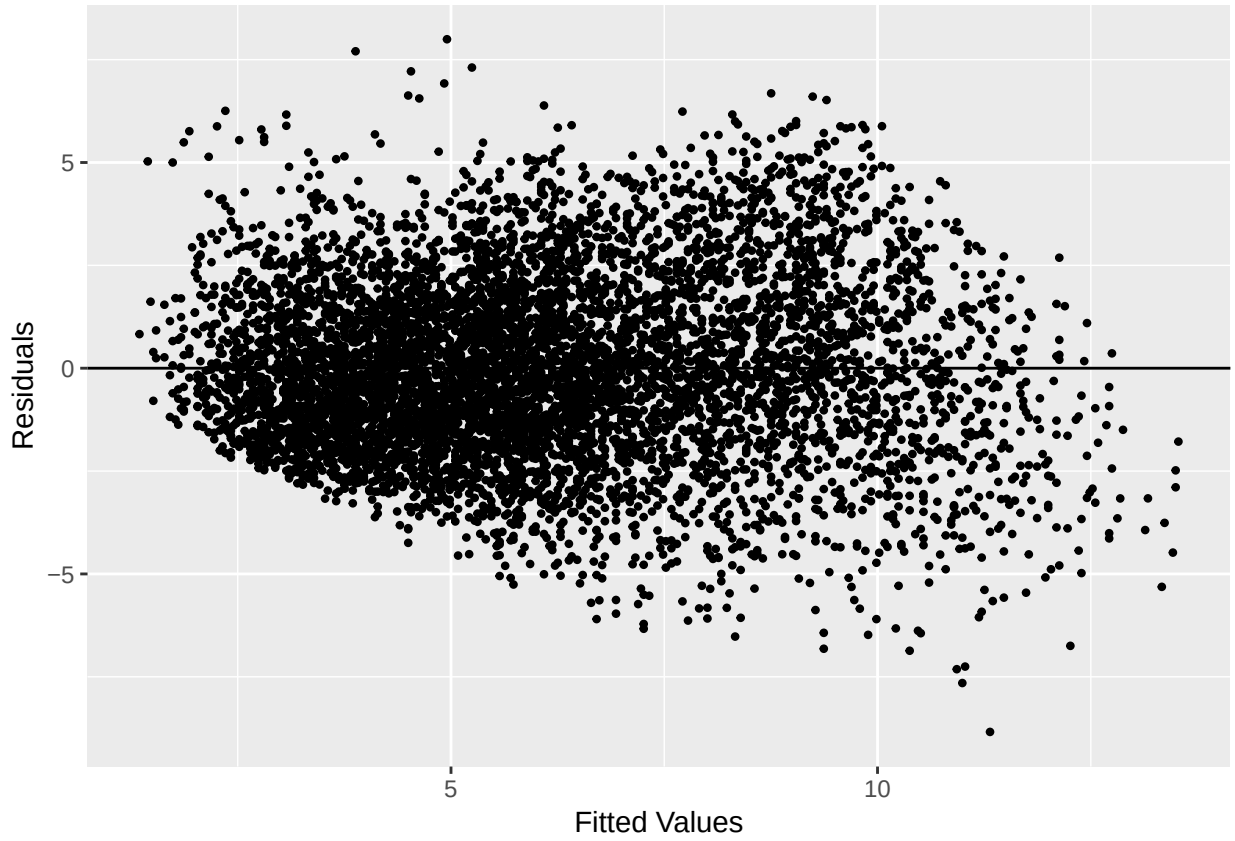
Then using these we plot the model distance against the travel time. For the purpose of this chart points are coloured by whether at least one match has occurred during the three seasons.

```
# create distance matrix for latent space model with zero groups fitted by MLE
dist_mle_0 <- modeldist(coords = coords_mle_0)

# convert matrices to edgelists
edgelist_matches <- matrix_to_edgelist(adjmatrix = match_matrix)
edgelist_travel_time <- matrix_to_edgelist(adjmatrix = travel_time)
edgelist_mle_0 <- matrix_to_edgelist(adjmatrix = dist_mle_0)

edgelist_matches[,4] <- edgelist_matches[,3]>0 # match occurred = TRUE/FALSE

# combine the edgelists into single data frame
edgelist <- cbind(edgelist_matches, edgelist_travel_time[,3])
edgelist <- cbind(edgelist, edgelist_mle_0[,3])
colnames(edgelist) <- c("Team_i","Team_j","matches","matches_bool","travel_time","dist_mle_0")

# model distance against travel time plotted
ggplot(edgelist, aes(x = travel_time, y = dist_mle_0)) +
  geom_point(size=0.5, aes(color=matches_bool)) +
  geom_smooth(method="lm",se=FALSE, color="black") +
  labs(x = "Travel time (minutes)", y = "Latent space distance", colour = "Number of matches") +
  scale_color_manual(breaks = c(TRUE, FALSE), values=c("red", "blue")) +
  theme(legend.position = "none")
```



The linear model of latent space distance against travel time is then fitted and a residual plot craeted.

```
#linear model fitted
m <- lm(formula = dist_mle_0 ~ travel_time , data = edgelist)

# residual plot created
ggplot(aes(x = .fitted, y= .resid), data = m) +
  geom_point(size = 0.85) +
  geom_hline(yintercept = 0) +
  labs(x = "Fitted Values",y = "Residuals")
```



If we consider the latent space model as predicting the travel time then it is interesting to consider if there is any pattern to the schools for which it under- or over-estimates. In order to do this we consider the pairs of schools where the residuals are above(below) some arbitrary level and take the frequency with which each school appears in this list. This can then be mapped.

```
# add the residuals for each pair of schools
edgelist <- cbind(edgelist, m$residuals)
names(edgelist)[7] <- "residuals"

# consider subsets above/below specific level (4)
edgelist_below <- subset(edgelist, residuals < -4)
edgelist_above <- subset(edgelist, residuals > +4)

# create data frames of schools by the number of times they appear in a list of pairs with residuals le
occurences_below <- as.data.frame(table(unlist(edgelist_below[,1:2])))
occurences_below <- data.frame(occurences_below[,-1], row.names = occurences_below[,1])
occurences_above <- as.data.frame(table(unlist(edgelist_above[,1:2])))
occurences_above <- data.frame(occurences_above[,-1], row.names = occurences_above[,1])

# add these counts to school_data, replace NAs, and rename columns
school_data_res <- transform(merge(school_data, occurences_below, by.x = 0, by.y = 0, all.x = TRUE),
                             row.names = Row.names, Row.names = NULL)
school_data_res <- transform(merge(school_data_res, occurences_above, by.x = 0, by.y = 0, all.x = TRUE)
                             row.names = Row.names, Row.names = NULL)
school_data_res[is.na(school_data_res)] <- 0
names(school_data_res)[ncol(school_data_res)-1] <- "res_below"
names(school_data_res)[ncol(school_data_res)] <- "res_above"
```

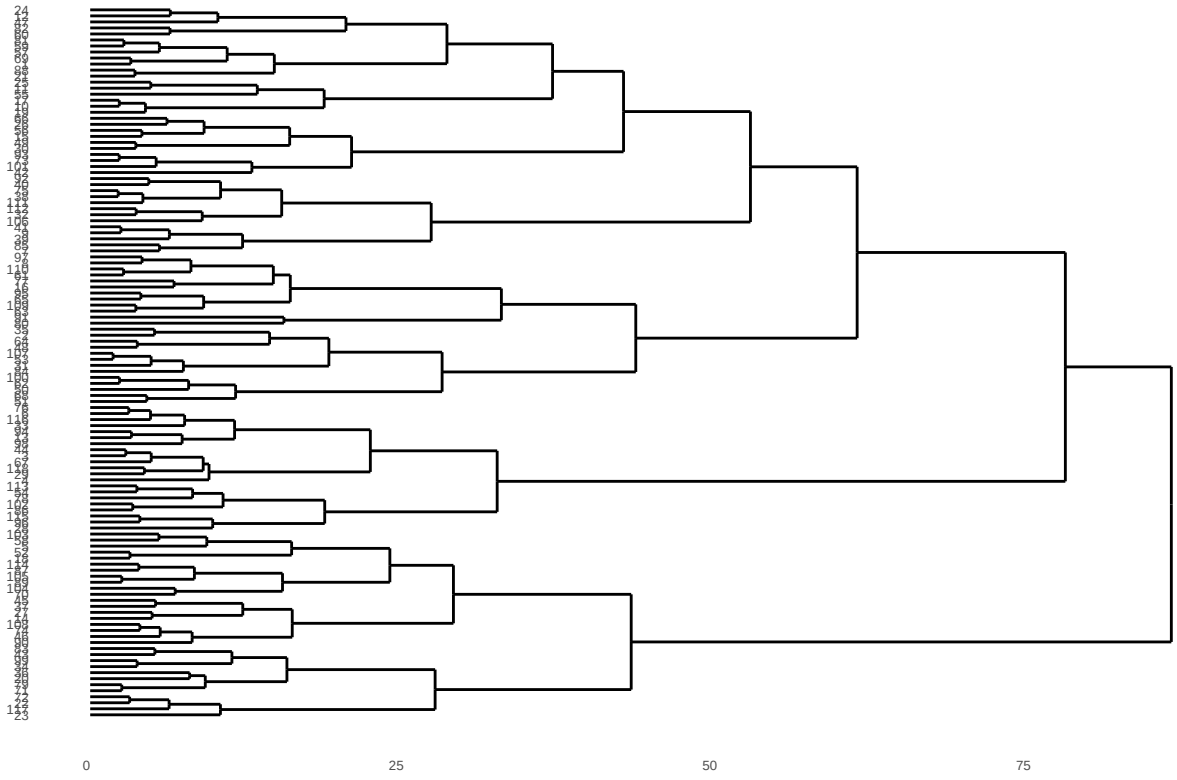
```r
# create map of British mainland
UK <- map_data(map = "world", region = "UK") # retrieve map of UK
GB <- subset(UK,group == 15) # just look at mainland Britain

# map with required format
GBmap <- ggplot(data = GB, aes(x = long, y = lat)) +
  geom_polygon(fill = "white", color = "black") +
  theme(
    panel.background = element_rect(fill = "light grey", colour = NA),
    panel.border     = element_rect(fill = NA, colour = "black"),
    panel.grid = element_line(colour = "light grey"),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank()) +
    coord_cartesian(xlim = c(-4,2), ylim = c(50,55.5), expand = FALSE)

# create map of schools with dot size equal to count
GBmap +
  geom_point(aes(x = LON, y = LAT, size = res_below), data = school_data_res) +
  theme(legend.position = "none")
```

```r
GBmap +
  geom_point(aes(x = LON, y = LAT, size = res_above), data = school_data_res) +
  theme(legend.position = "none")
```

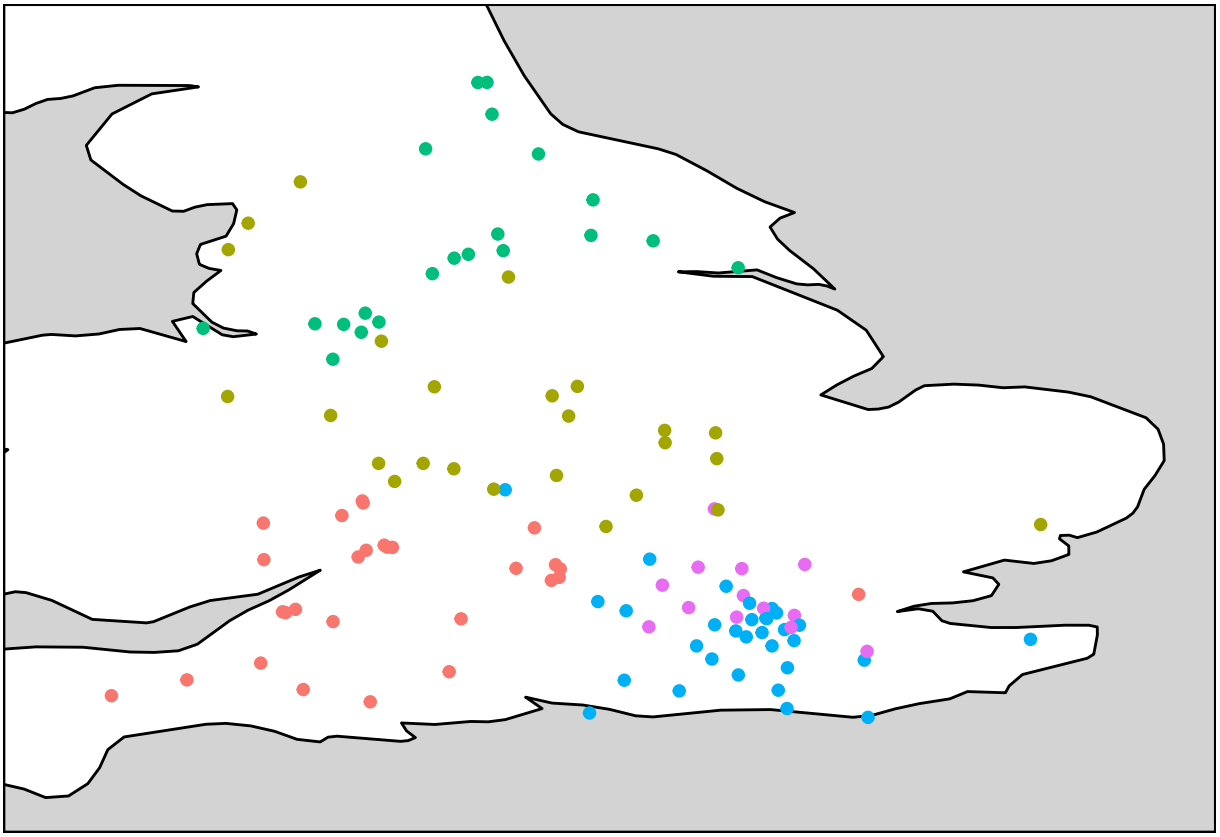
Based on these latent space distances we can use hierarchical clustering to investigate communities.

```
library(ggdendro) # requires ggplot2
```

```
# identify communities
com_mle_0 <- hclust(dist(dist_mle_0), method = "complete")

# plot dendrogram
ggdendrogram(com_mle_0, rotate = TRUE) +
  theme(axis.text = element_text(size=5)) +
  theme(panel.border = element_blank())
```




```
# assign to communities by cutting tree
com_mle_h5 <- cutree(com_mle_0,5)

# add to school_data
school_data_mle_h5 <- cbind(school_data, com_mle_h5)
names(school_data_mle_h5)[ncol(school_data_mle_h5)] <- "com_mle_h5"

# plot on map with different communities as different colours
GBmap + geom_point(aes(x = LON, y = LAT, colour = as.factor(com_mle_h5)), data = school_data_mle_h5, si
  theme(legend.position = "none")
```



4.3 Latent Position Cluster Model

We now move on to fit the model discussed in Handcock et al(2007), with dimension=2, communities = 4, from binomial family with n=3, using maximum likelihood estimation. The same model is then fitted using MCMC with burnin = 10,000, #iterations = 1,000,000, sampling every 50th iteration.

```
# maximum likelihood estimation
fit_mle_4 <- ergmm(g ~ euclidean(d=2, G=4), family = "binomial", fam.par = list(trials=3), tofit = c("m

# extract latent space coordinates
summary_mle_4 <- summary(fit_mle_4)
coords_mle_4 <- (summary_mle_4$mle)$Z
```

```
set.seed(3141)

# MCMC estimation
fit_mcmc_4 <- ergmm(g ~ euclidean(d=2, G=4) ,family = "binomial", fam.par = list(trials=3),
                    control = ergmm.control(burnin = 10000, sample.size = 20000, interval = 50, Z.delta
                    seed=3141, verbose = FALSE)

# extract latent space coordinates
summary_mcmc_4 <- summary(fit_mcmc_4)
```

```
## NOTE: It is not certain whether it is appropriate to use latentnet's BIC to select latent space dimen
```

```
coords_mcmc_4 <- (summary_mcmc_4$pmean)$Z
comprob_mcmc_4 <- (summary_mcmc_4$pmean)$Z.pZK
```

As before these may be used to create a scatterplot of latent space distance against travel time or a geographical map where community is represented by colour. Additionally we can retrieve the BIC value for the MCMC estimation from the summary,

```
bic.ergmm(fit_mcmc_4)$overall
```

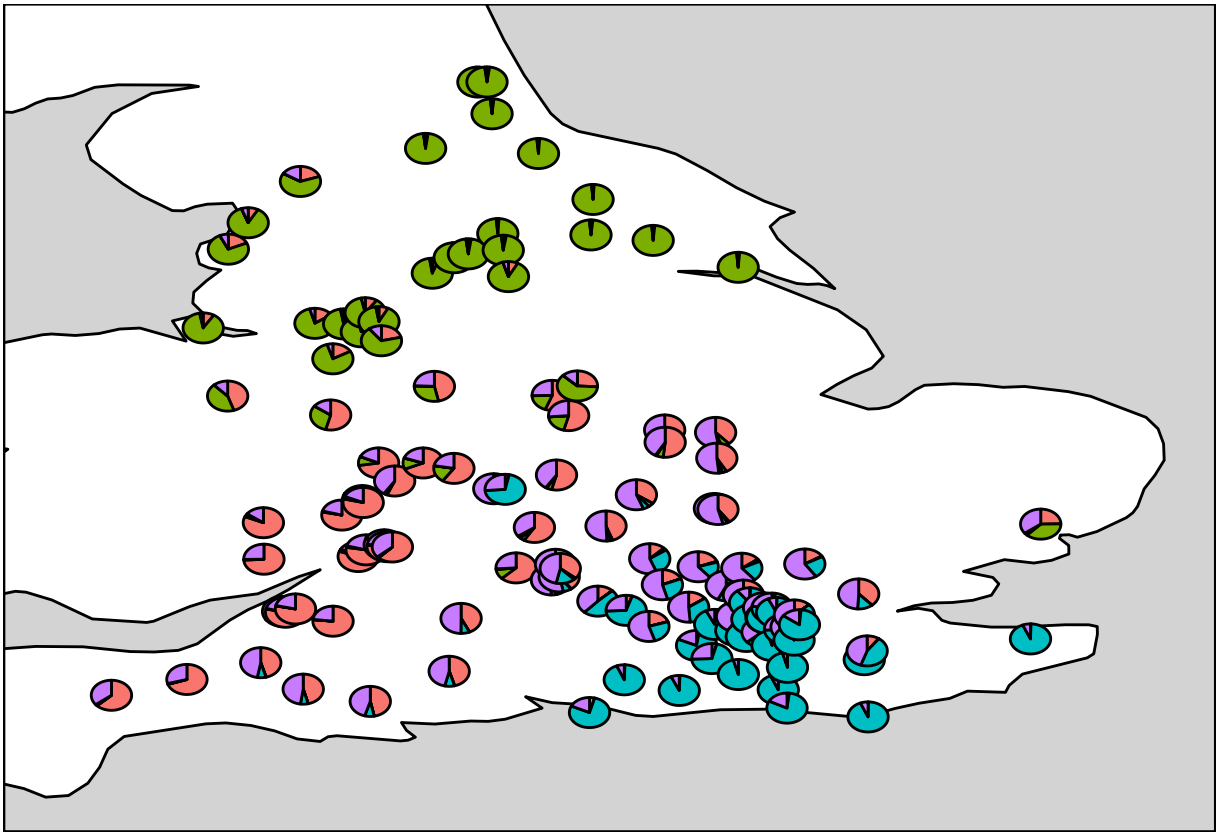
```
## NOTE: It is not certain whether it is appropriate to use latentnet's BIC to select latent space dimen
```

```
## [1] 3646.798
```

and produce a plot of the probabilistic community membership for each school.

```
library(scatterpie)
```

```
# probability map
school_data_comprob <- school_data[,10:11]
school_data_comprob <- data.frame(Team = row.names(school_data_comprob), school_data_comprob)
school_data_comprob <- cbind(school_data_comprob,comprob_mcmc_4)
colnames(school_data_comprob)[4:7] <- c("A","B","C","D")

GBmap + geom_scatterpie(aes(x = LON, y = LAT, group = Team, r = 0.1), data = school_data_comprob, cols =
  theme(legend.position = "none")
```

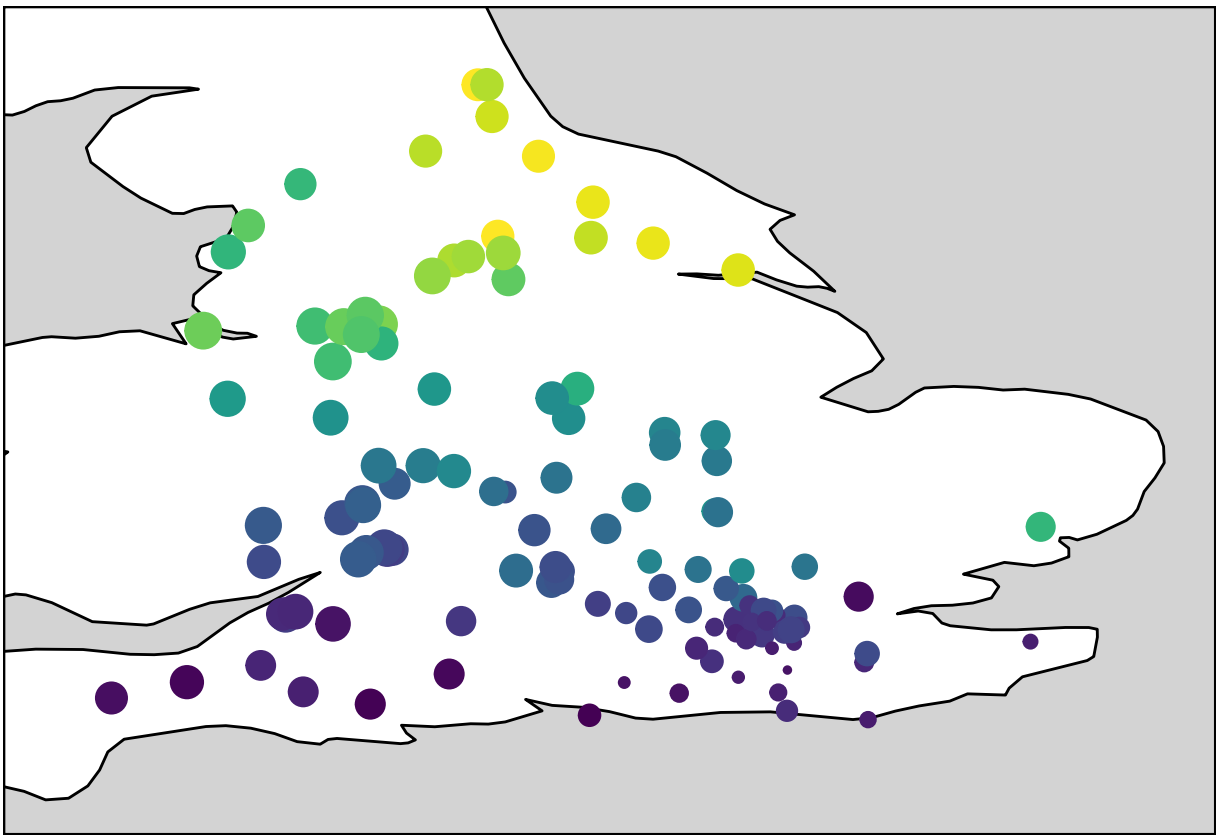


It is also informative to represent the latent space coordinates of each school on the map.

```
library(viridis)
```

```
school_data_coords <- cbind(school_data, coords_mcmc_4)
n <- ncol(school_data_coords)
colnames(school_data_coords)[(n-1):n] <- c("Z1_mcmc_4", "Z2_mcmc_4")
```

```
GBmap + geom_point(aes(x = LON, y = LAT, colour = -Z1_mcmc_4, size=-Z2_mcmc_4), data = school_data_coor
  scale_color_viridis() +
  theme(legend.position = "none")
```



A consideration of covariates

Relative importance

At this point we turn to looking at the relative importance of the covariates. A convenient way to use the calc.relimp function is to supply a data frame with the response variable in the first column and only dependent variables in other columns, and so the data is sorted to allow this.

```
library(relaimpo)
```

```
# rescale relevant variables and remove surplus columns
school_data$Fees <- school_data$Fees / 10000
school_data$Founded <- school_data$Founded / 100
school_data$B_6th <- school_data$B_6th / 100
school_data <- school_data[,1:8]

# change type for P_S so function may be applied
school_data$P_S <- as.integer(school_data$P_S)

# create edgelist for relative importance calculations
edgelist_relimp <- nodedata_to_edgelist(nodedata = school_data)

dist_mcmc_4 <- modeldist(coords=coords_mcmc_4) # create LS distance matrix for MCMC, communities=4
```

```r
edgelist_mcmc_4 <- matrix_to_edgelist(adjmatrix = dist_mcmc_4) # convert LS distance matrix to edgelist

# create required data frame
edgelist_relimp <- edgelist_relimp[,-(1:2)]
edgelist_relimp <- as.data.frame(sapply(edgelist_relimp, as.numeric))
edgelist_relimp <- cbind(edgelist_mcmc_4[3], edgelist_relimp)
edgelist_relimp <- cbind(edgelist_relimp, edgelist_travel_time[3])
colnames(edgelist_relimp)[1] <- "dist_mcmc_4"
colnames(edgelist_relimp)[ncol(edgelist_relimp)] <- "travel_time"

# calculate relative importance
relimp <- calc.relimp(object=edgelist_relimp, type = c("lmg", "pratt"), x = NULL, rela = TRUE, rank = F
round(relimp$lmg, 4)  # relative importance under LMG method
```

```
##         P_S       Terms        Fees     Founded       B_6th     Pct_Boy
##      0.0031      0.0001      0.0086      0.0007      0.0008      0.0003
## Pct_Boarder      Rating travel_time
##      0.0079      0.0012      0.9772
```

```r
round(relimp$pratt, 4)  # relative importance under Pratt method
```

```
##         P_S       Terms        Fees     Founded       B_6th     Pct_Boy
##     -0.0007      0.0000      0.0117      0.0004      0.0006      0.0003
## Pct_Boarder      Rating travel_time
##      0.0077      0.0000      0.9798
```

5.2 Covariate inclusion

Finally we fit the model with edge covariates included. Here we use travel time as the relevant edge covariate. Travel_time has been directly supplied; for other covariates the relevant matrix may be created from school_data.

```r
travel_time <- as.matrix(travel_time)
travel_time <- travel_time / 60 # convert from minutes to hours

fit_mcmc_4_traveltime <- ergmm(g ~ euclidean(d=2, G=4) + edgecov(travel_time), family = "binomial", fam
                               control = ergmm.control(burnin = 10000, sample.size = 20000, interval = 5
                               seed=3141, verbose = FALSE)

summary_mcmc_4_traveltime <- summary(fit_mcmc_4_traveltime)
coords_mcmc_4_traveltime <- (summary_mcmc_4_traveltime$pmean)$Z
rownames(coords_mcmc_4_traveltime) <- school_names
```

```r
# summary provides beta estimate, q values and BIC
summary_mcmc_4_traveltime
```

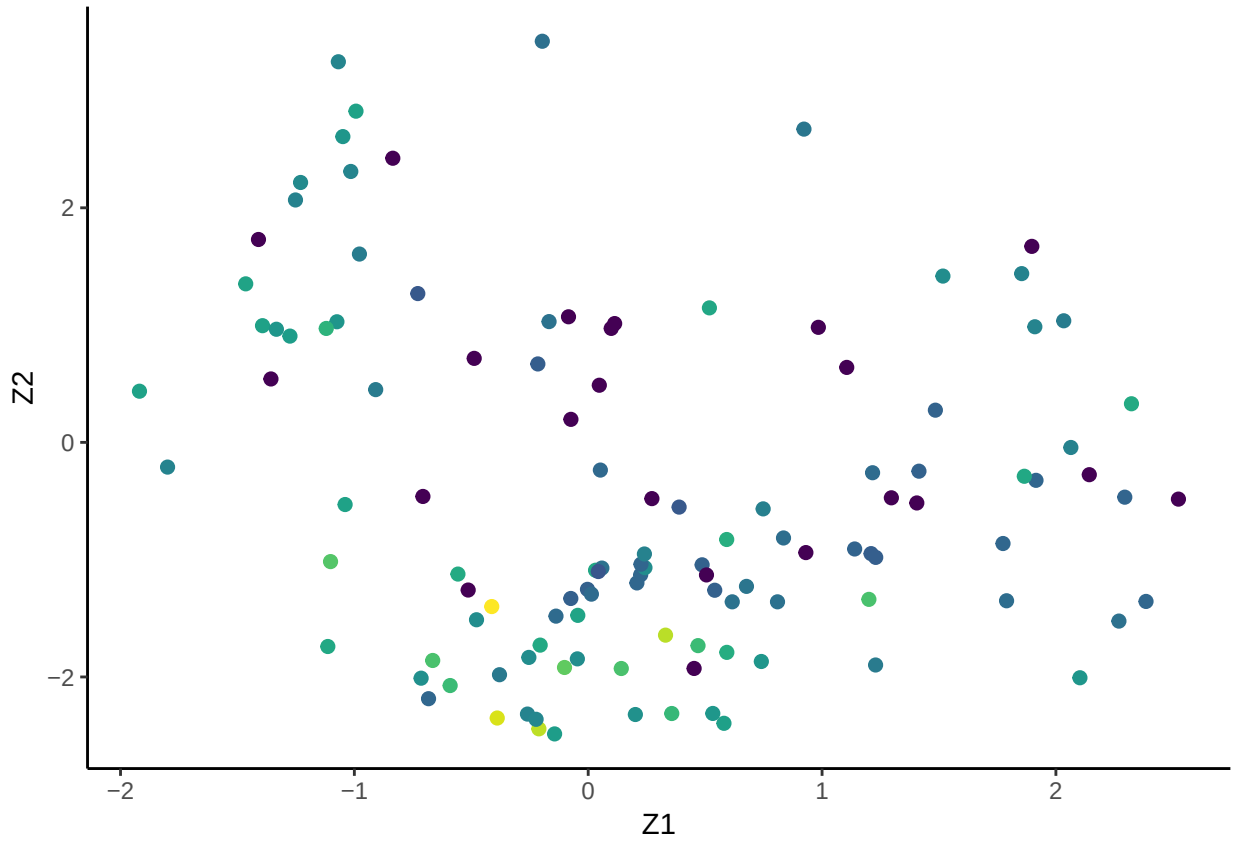
```
##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   g ~ euclidean(d = 2, G = 4) + edgecov(travel_time)
## Attribute: edges
## Model:     binomial
## MCMC sample of size 20000, draws are 50 iterations apart, after burnin of 10000 iterations.
## Covariate coefficients posterior means:
```

```
##                      Estimate     2.5%   97.5% 2*min(Pr(>0),Pr(<0))
## (Intercept)           1.07238  0.83219  1.3121            < 2.2e-16 ***
## edgecov.travel_time -1.52008 -1.64840 -1.3914            < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Overall BIC:        3294.989
## Likelihood BIC:     2695.017
## Latent space/clustering BIC:     599.9714
##
## Covariate coefficients MKL:
##                        Estimate
## (Intercept)           0.3400155
## edgecov.travel_time -1.4597726
```

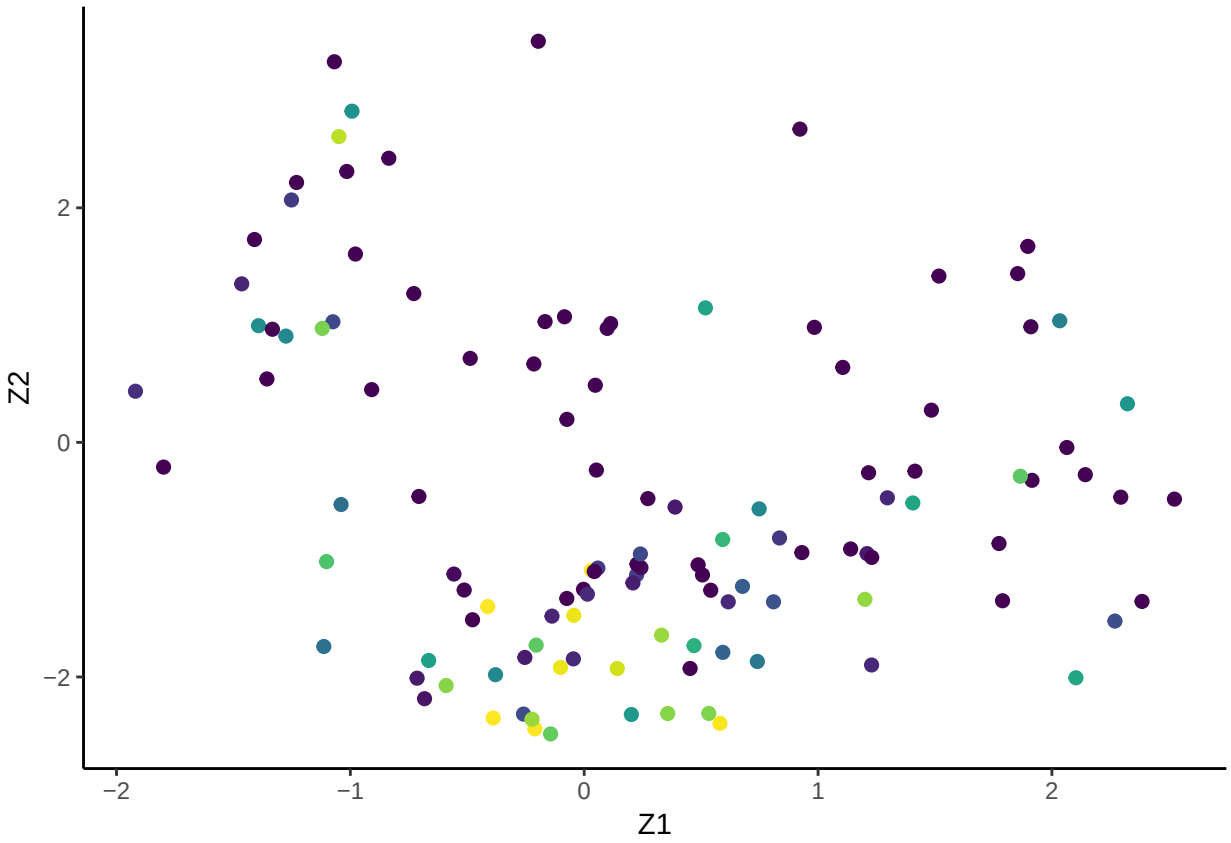
5.3 Graphical inspection

We can then plot the latent space directly using colours to highlight the nature of clustering in other covariates, in particular by creating plots where the colour represents the fees and the proportion of boarders.

```
school_data_mcmc_4_traveltime <- cbind(school_data, coords_mcmc_4_traveltime)
n <- ncol(school_data_mcmc_4_traveltime)
colnames(school_data_mcmc_4_traveltime)[(n-1):n] <- c("Z1", "Z2")

ggplot(data=school_data_mcmc_4_traveltime, aes(x = Z1, y = Z2, color = Fees)) +
  geom_point(size=2) + scale_colour_viridis() +
  theme(legend.position="none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"))
```




```
ggplot(data=school_data_mcmc_4_traveltime, aes(x=Z1,y=Z2,color=Pct_Boarder)) +
  geom_point(size=2) + scale_colour_viridis() +
  theme(legend.position="none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"))
```


Z2
2
0
−2
−2
−1
0
1
2
Z1