Using Genetic Algorithm with TSP to Solve Resource Distribution During Outlier Event

Chih-Hao Huang

```
library(tidyverse)
library(knitr)
library(GA)
library(rgdal)
library(tidygeocoder)
```

1. Data

Veterans Health Administration Medical Facilities data collected from: link

VHA data in the US was collected (updated 2018), only contiguous US states including D.C. have been selected.

```
set.seed(2020)
# VHA data
VHA <- read.csv("data/VHA.csv", stringsAsFactors = F)
names(VHA)[
  names(VHA) == "ï..X"
] <- "X"
VHA <- subset(VHA, STATE != "AK")
VHA <- subset(VHA, STATE != "HI")
VHA <- subset(VHA, STATE != "AS")
VHA <- subset(VHA, STATE != "PR")
VHA <- subset(VHA, STATE != "GU")
VHA <- subset(VHA, STATE != "MP")
VHA <- subset(VHA, STATE != "VI")
```

VHA medical centers' rating data collected from *the U.S. Department of Veterans Affairs*: link

The End of Year Hospital Star Rating (FY2018) contains relative performance star rating of each medical center from 1 to 5 (5 being the best).

```
VHAr <- read.csv("data/VHArating.csv", stringsAsFactors = F)
VHAr <- VHAr[,1:3]
colnames(VHAr) <- c("VSIN", "NAME", "Performance")
```

Subset facilities name, longitude and latitude from `VHA`. Then select VHA medical centers' only from `VHA`, and merge with `VHAr`

```r
mc <- subset(VHA, select = c("NAME", "LONGITUDE", "LATITUDE", "STATE"))
mc <- subset(mc, NAME %in% VHAr$NAME)
# Remove duplicated data
mc <- mc[!duplicated(mc$NAME),]
VHAr <- VHAr[!duplicated(VHAr$NAME),]
# Order A->Z
mc <- mc[order(mc$NAME),]
VHAr <- VHAr[order(VHAr$NAME),]
# Bind Columns
mc <- cbind(mc, VHAr)
mc <- subset(mc, select = c(1:5, 7))
```

The *Provisional COVID-19 Death Counts by Sex, Age, and State* was collected from *CDC*: link

In this study, only contiguous states and D.C. are selected.

```r
# CDC COVID-19 Death Count Dataset
covid <- read.csv("data/CDCdeathCOVID.csv", stringsAsFactors = F)
# COVID-related specifically
covid <- subset(covid, select = -c(1:3,13))
# Keep data for each states, each sex, and each age group only
covid <- subset(covid, State != "United States" & Sex != "All" & Age.group != "All ages")
# Remove data for non-contiguous state (excluding D.C)
covid <- subset(covid, State != "Puerto Rico")
covid <- subset(covid, State != "Alaska")
covid <- subset(covid, State != "Hawaii")
covid <- subset(covid, State != "New York City")
# Replace NA with 0
covid[is.na(covid)] <- 0

# Replace State name with Abbreviation
covid$State <- state.abb[match(covid$State, state.name)]
covid$State[is.na(covid$State)] <- "DC"
```

Select COVID only patients for now

```r
cv <- covid[,1:4]
```

State center geological coordinates of the contiguous states and D.C. were collected from `state` function.

```r
# Collect State center information
sc <- data.frame(
  matrix(
    0,
    nrow = 49,
    ncol = 2
  )
)
sc <- data.frame(
  State = state.abb,
  long = state.center$x,
  lat = state.center$y
)
```

```
sc <- subset(sc, State != "AK")
sc <- subset(sc, State != "HI")
dc <- data.frame(
  State = "DC",
  long = -77.0369,
  lat = 38.9072
)
sc <- bind_rows(sc, dc)
sc <- sc[order(sc$State),]
```

Shape files of the US collected from: link

The shape files of the contiguous US states including D.C. used are based on data collected in 2019.

```
# Contiguous US (including D.C.) map shape
US <- readOGR(dsn = "map_shapes", layer = "tl_2019_us_state")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\andy5\Documents\Turing Research\TSPus\map_shapes", layer: "tl_2019_us_state"
## with 56 features
## It has 14 fields
## Integer64 fields read as strings:  ALAND AWATER
```

```
US <- subset(US, STUSPS != "AK")
US <- subset(US, STUSPS != "HI")
US <- subset(US, STUSPS != "AS")
US <- subset(US, STUSPS != "PR")
US <- subset(US, STUSPS != "GU")
US <- subset(US, STUSPS != "MP")
US <- subset(US, STUSPS != "VI")
US <- fortify(US)
```

```
## Regions defined for each Polygons
```

Voinsky, Irena et al. "Effects of age and sex on recovery from COVID-19: Analysis of 5769 Israeli patients." *The Journal of infection* vol. 81,2 (2020): e102-e103. doi:10.1016/j.jinf.2020.05.026 link

Note: This data is based on Israeli patients, for more accurate result, data on US patient should be used.

```
# Age group
ag <- c("19 and below", "20-29", "30-39", "40-49", "50-59", "60 and above")

# Age group in Male
rm <- c(13.61, 13.97, 14.46, 14.79, 14.81, 14.73)

# Age group in Female
rf <- c(13.24, 13.99, 14.17, 14.76, 14.18, 13.99)
```

Alter age group making data frames compatible

```r
ag <- cv$Age.group[1:11]
rm <- c(rm[1], rm[1], rm[1], mean(rm[1], rm[2]), mean(rm[2], rm[3]), mean(rm[3], rm[4]), mean(rm[4], rm

rf <- c(rf[1], rf[1], rf[1], mean(rf[1], rf[2]), mean(rf[2], rf[3]), mean(rf[3], rf[4]), mean(rf[4], rf

# Create data frame
rp <- data.frame(
  matrix(c(rm, rf), nrow = 2 * length(rm), ncol = 1)
)

sex <- data.frame(
  matrix(c(
    rep("Male", length(rm)), rep("Female", length(rf))
  ),
    nrow = 2 * length(rm),
    ncol = 1
  )
)

ag <- data.frame(
  matrix(
    c(ag, ag),
    nrow = 2* length(rm),
    ncol = 1
  )
)

rp <- bind_cols(sex, ag, rp)

colnames(rp) <- c("Sex", "Age", "RecoveryPeriod")
```

According to America's Health Insurance Plans, the average cost per patient per day is around $1500.

```r
ccpd <- 1500
```

Therefore, the average cost per person for COVID treatment in each group will be the following:

```r
rp$cost <- rp$RecoveryPeriod * ccpd
```

Calculate cost of treatment in each states

```r
# Calculate cost of treatment based on sex, age group and states.
cv$cost <- ifelse(cv$Sex == rp$Sex & cv$Age.group == rp$Age, cv$COVID.19.Deaths * rp$cost)

# Total cost of treatment in each states
covidState <- aggregate(cbind(cost) ~ State, data = cv, FUN = sum)
# Normalized cost data into index between 0 and 1
covidState$costID <- (covidState$cost - min(covidState$cost)) / (max(covidState$cost) - min(covidState$
```

2. Functions

Following functions are based on: link. This function calculate the total distance based on given tour (TSP):

```
tourLength <- function(tour, distMatrix) {
    tour <- c(tour, tour[1])
    route <- embed(tour, 2)[, 2:1]
    sum(distMatrix[route])
}
```

In this fitness function, only distance have been considered.

```
fitness1 <- function(tour, ...) 1/tourLength(tour, ...)
```

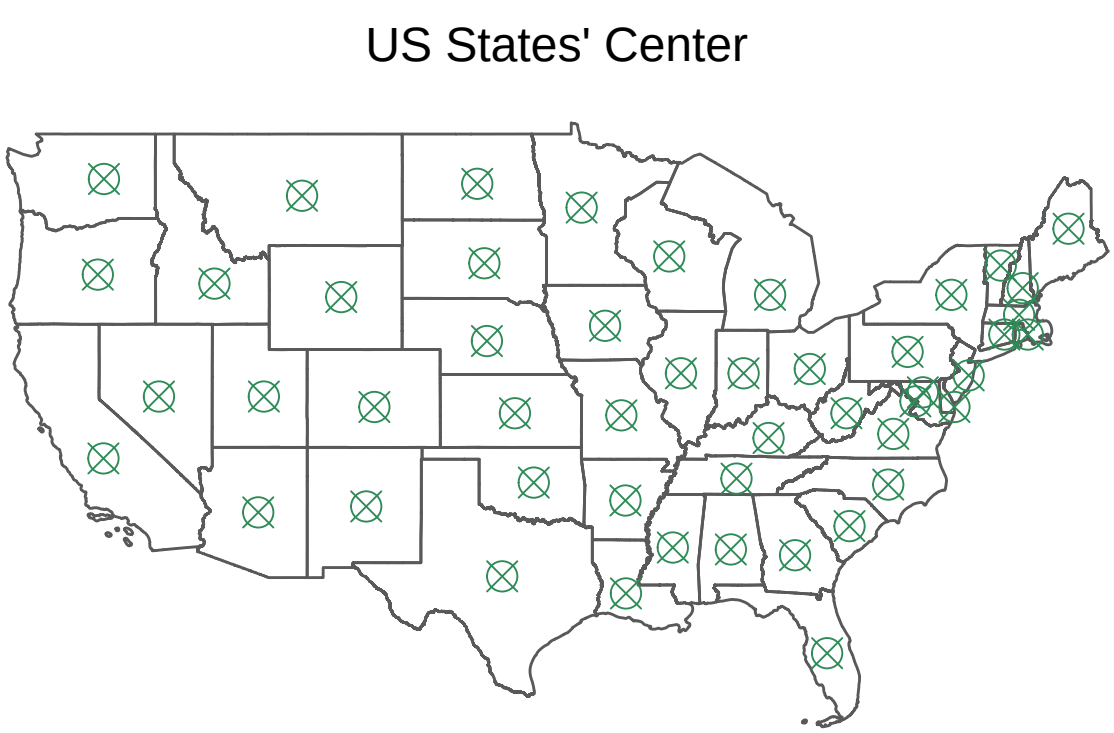
3. State Centers without Route

Distance matrix between states centers

```
scd <- as.matrix(stats::dist(select(sc, long, lat), diag = TRUE, upper = TRUE))
colnames(scd) <- sc$State
rownames(scd) <- sc$State
```

```
p1 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc,
    shape = 13,
    size = 5,
    color = "seagreen",
    aes(
      x = long,
      y = lat,

    )
  ) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "US States' Center"
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
```

```
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p1
```




```
# Save image file
pdf(
  file = "image/p1.pdf",
  width = 12,
  height = 7
)

p1

dev.off()
```

```
## pdf
##   2
```

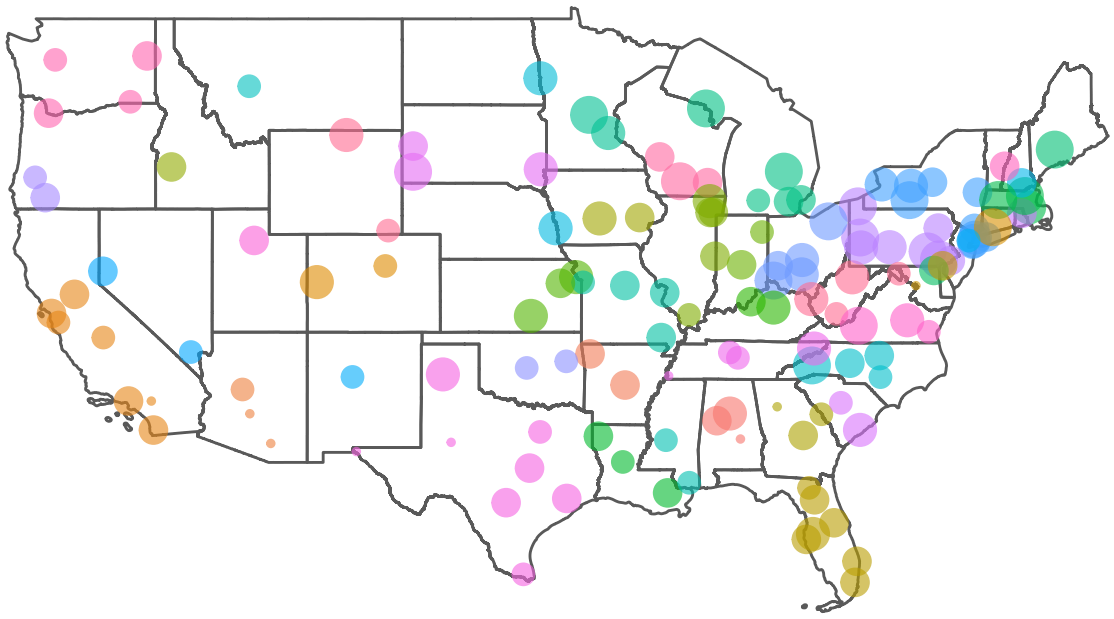
4. VHA without Route

Distance matrix between medical centers

```r
distance <- as.matrix(stats::dist(select(mc, LONGITUDE, LATITUDE), diag = TRUE, upper = TRUE))
colnames(distance) <- mc$NAME
rownames(distance) <- mc$NAME
```

Veterans Health Administration Medical Facilities data on US map. Size based on the Performance Rating from 1 to 5. Color based on the state.

```r
p2 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = mc,
    alpha = 0.6,
    aes(
      x = LONGITUDE,
      y = LATITUDE,
      size = Performance,
      color = STATE
    )
  ) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "VHA Medical Center"
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p2
```



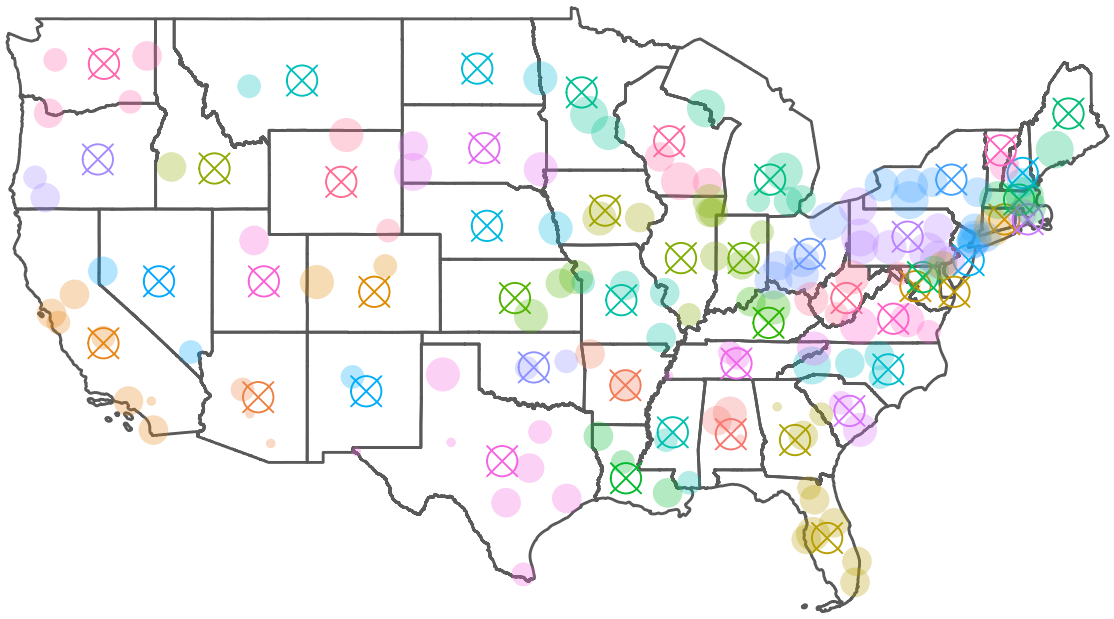

```
# Save image file
pdf(
  file = "image/p2.pdf",
  width = 15,
  height = 7
)

p2

dev.off()
```

```
## pdf
##   2
```

5. VHA and US State Center

```
p3 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
```

```
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc,
    shape = 13,
    size = 5,
    aes(
      x = long,
      y = lat,
      color = State
    )
  ) + geom_point(
    data = mc,
    alpha = 0.3,
    aes(
      x = LONGITUDE,
      y = LATITUDE,
      color = STATE,
      size = Performance
    )
  ) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "U.S. State Centers with VHA Medical Centers"
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p3
```




```
# Save image file
pdf(
  file = "image/p3.pdf",
  width = 12,
  height = 7
)

p3

dev.off()
```

```
## pdf
##   2
```

6. Suggested Route Between States' Center Based on GA and TSP (Distance Only)

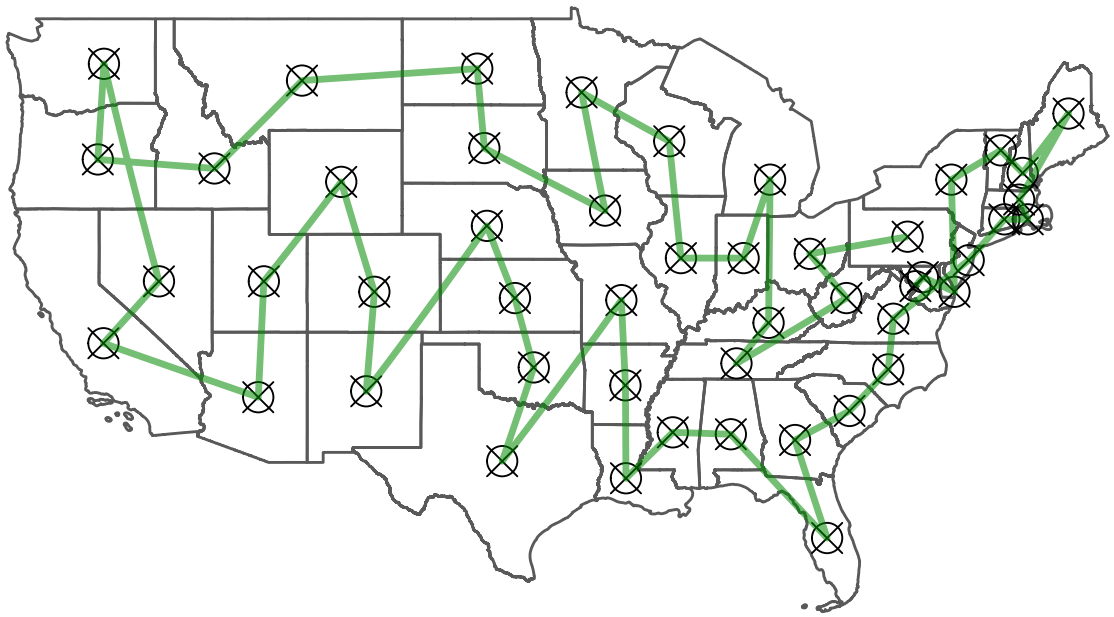
```
nr <- nrow(sc)
GA.fit <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd,
```

```
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fit)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 5000
## Fitness function value = 0.004735849
## Solution =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10  ...  x48 x49
## [1,] 37 34 48 41 16 21 14 13 47  22        19   7
```

```
n = nrow(sc)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fit@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis <- cbind(id, sc)
colnames(vis)[1] <- "id"
vis <- merge(vis, tour)
vis <- vis %>%
  arrange(path_order)

p4 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
```

```
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc,
    size = 5,
    shape = 13,
    aes(
      x = long,
      y = lat
    )
  ) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "State Centers - Trad. TSP - Unnormalized"
) + geom_path(
  data = vis,
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  ),
  color = "green4",
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p4
```




```
# Save image file
pdf(
  file = "image/p4.pdf",
  width = 12,
  height = 7
)

p4

dev.off()
```

```
## pdf
##   2
```

```
jpeg(
  file = "image/sc_ff1_un.jpeg"
)

p4

dev.off()
```

```
## pdf
##   2
```

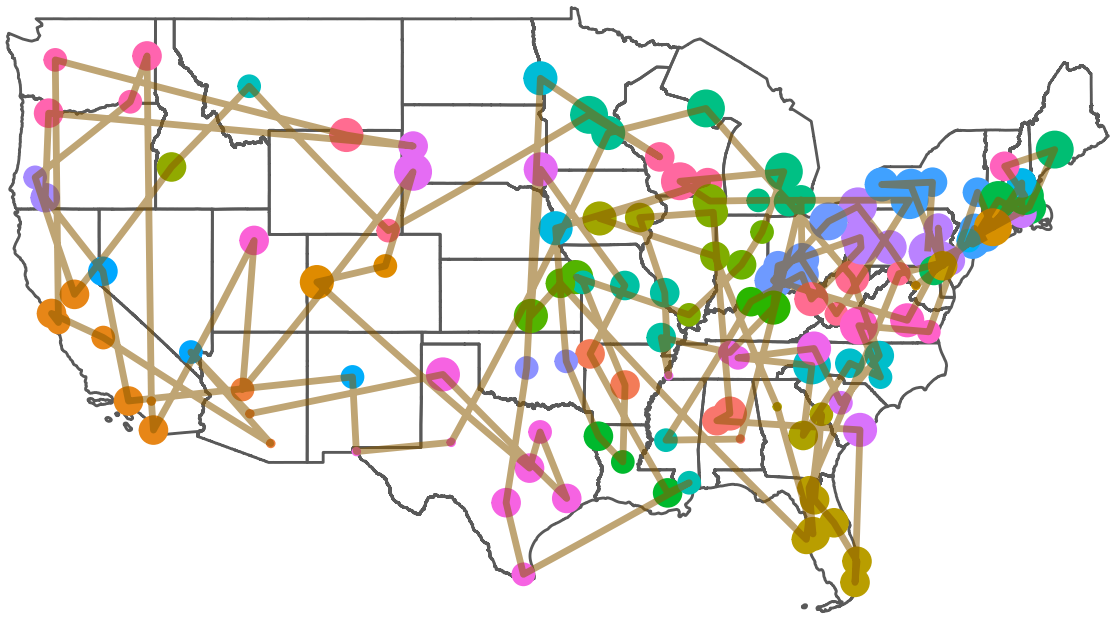
7. Suggested Route Between VHA Medical Centers Based on GA and TSP (Distance Only)

```
nr <- nrow(mc)
GA.fit <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = distance,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fit)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 5000
## Fitness function value = 0.001507911
## Solution =
##      x1 x2 x3  x4 x5 x6  x7  x8 x9 x10  ...  x140 x141
## [1,] 85 15 59 126 31 73 127 100 48  99        107   67
```

```
n = nrow(mc)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fit@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis <- cbind(id, mc)
colnames(vis)[1] <- "id"
vis <- merge(vis, tour)
vis <- vis %>%
  arrange(path_order)
```

```r
p5 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = mc,
    aes(
      x = LONGITUDE,
      y = LATITUDE,
      size = Performance,
      color = STATE,
    )
  ) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "VAH - Trad. TSP - Unnormalized"
) + geom_path(
  data = vis,
  size = 1.2,
  aes(
    x = LONGITUDE,
    y = LATITUDE,
    alpha = 0.9
  ),
  color = "orange4",
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p5
```




```
# Save image file
pdf(
  file = "image/p5.pdf",
  width = 12,
  height = 7
)

p5

dev.off()
```

```
## pdf
##   2
```

```
jpeg(
  file = "image/vah_ff1_un.jpeg"
)

p5

dev.off()
```

```
## pdf
##   2
```

8. GA using Patients, Hospitals, and TSP

Here I assume *the state center is the distribution center for each state.*

In the data, we have the number of patients based on their states, sex, and age group. These information were used to calculate the *total cost* to treat them based on number of patients in each state, the cost per day for the patient treatment and the recovery period data based on sex and age group.

`Cost` is based on `Sex`, and `Age`. Different `Sex` and `Age` group have different average `Recovery Period`, therefore the cost will be different. `Cost` and `Recovery Period` are proportional in this study. Here average cost per day per COVID-19 patient was used.

```
# Total patients per state
sc$cost <- covidState$cost
patient <- aggregate(cbind(COVID.19.Deaths) ~ State, data = covid, FUN = sum)
sc$patient <- patient$COVID.19.Deaths
```

`Hospital performance rating` include hospital staffs and resources and other factors. Here this will be use to calculate index which shows the total resources in each state.

```
# Total hospital performance per state
hp <- aggregate(cbind(Performance) ~ STATE, data = mc, FUN = sum)
sc$hp <- hp$Performance
```

In the data, there are two states without patient data, these will be removed from this study.

```
# Remove states with no patients
sc <- subset(sc, patient != 0)
```

Calculate hospital resources per patient index using:

$$\frac{HospitalPerformanceRating}{Patient}$$

```
sc$hrp <- sc$hp / sc$patient
```

The lower the above index, less resources per patient in the state, which will be more emergeny then other states.

GA

Now apply the data to the GA fitness function based on the following function:

$$\frac{Patient}{Cost \times Distance \times HospitalPerformanceRating}$$

```
scd <- as.matrix(stats::dist(select(sc, long, lat), diag = TRUE, upper = TRUE))
scd <- as.data.frame(scd)
colnames(scd) <- sc$State
rownames(scd) <- sc$State
```

```r
sc$crp <- sc$cost * sc$hrp
crp <- sc$crp
crp <- as.data.frame(crp)
row.names(crp) <- sc$State

# transpose
crp <- t(crp)
crp <- as.data.frame(crp)

# Create new distance matrix accounting all variables
scd_w <- data.frame(mapply(`*`, scd, crp))
colnames(scd_w) <- sc$State
row.names(scd_w) <- sc$State
```

Apply the fitness function to GA to find out the best route.

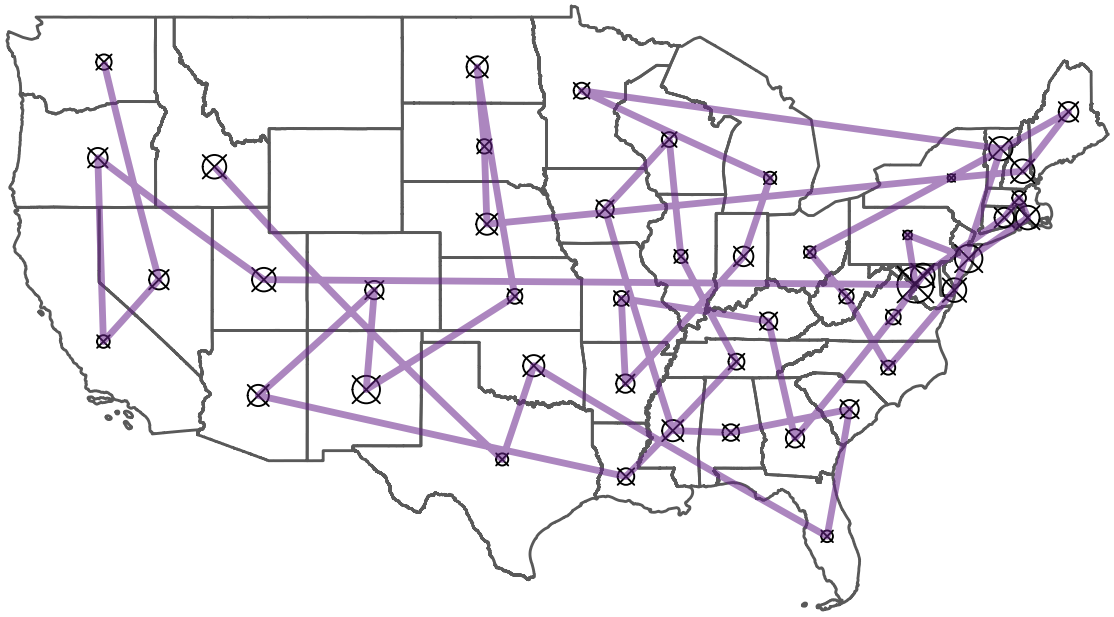
```r
nr <- nrow(scd_w)
GA.fit <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd_w,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fit)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 1000
## Fitness function value = 1.913961e-08
## Solution =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10  ...  x46 x47
## [1,] 45 31  4 35 42  7 36 29 37  18        41  12
```

```r
n = nrow(sc)
id = list(1:n)
id <- as.data.frame(id)
```

```
tour <- GA.fit@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis <- cbind(id, sc)
colnames(vis)[1] <- "id"
vis <- merge(vis, tour)
vis <- vis %>%
  arrange(path_order)

p6 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc,
    shape = 13,
    aes(
      x = long,
      y = lat,
      size = (1/crp)
    )
  ) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "State Centers - FF2 - Unnormalzied"
) + geom_path(
  data = vis,
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  ),
  color = "darkorchid4",
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
```

```
  panel.border = element_blank(),
  panel.background = element_blank()
)

p6
```




```
# Save image file
pdf(
  file = "image/p6.pdf",
  width = 12,
  height = 7
)

p6

dev.off()
```

```
## pdf
##   2
```

```
jpeg(
  file = "image/sc_ff2_un.jpeg"

)
```

```
p6

dev.off()
```

```
## pdf
##   2
```

8. K-means Clustering

Cluster state center using k-means clustering approach into *four* clusters.

Note: Number of clusters and cluster centers can be changed based on different situations, such as amount of distribution center and location of the distribution center.

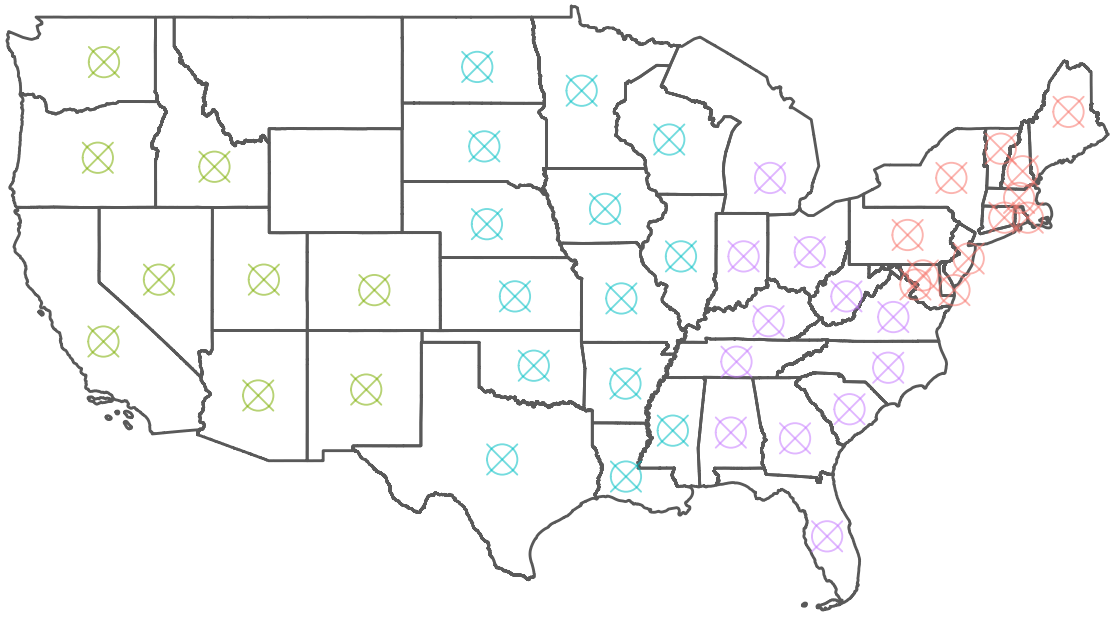
```
# Clustering based on distance
km <- kmeans(scd, centers = 4)

sc$region <- km$cluster
```

```
p7 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc,
    shape = 13,
    size = 5,
    aes(
      x = long,
      y = lat,
      colour = factor(region),
      alpha = 0.9
    )
  ) + coord_map() +
  scale_fill_brewer(
    palette = "set1"
  )+ labs(
  x = NULL,
  y = NULL,
  title = "K-means State Clusters"
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
```

```
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)
```

```
## Warning in pal_name(palette, type): Unknown palette set1
```

```
p7
```




```
# Save image file
pdf(
  file = "image/p7.pdf",
  width = 12,
  height = 7
)

p7

dev.off()
```

```
## pdf
```

```
##   2
```

Subset dataframe into four sub dataframe based on region cluster:

```
# Region 1
sc1 <- subset(sc, region == 1)
sccoord1 <- subset(sc1, select = c("long", "lat"))
row.names(sccoord1) <- sc1$State

#####################################################

# Region 2
sc2 <- subset(sc, region == 2)
sccoord2 <- subset(sc2, select = c("long", "lat"))
row.names(sccoord2) <- sc2$State

#####################################################

# Region 3
sc3 <- subset(sc, region == 3)
sccoord3 <- subset(sc3, select = c("long", "lat"))
row.names(sccoord3) <- sc3$State

#####################################################

# Region 4
sc4 <- subset(sc, region == 4)
sccoord4 <- subset(sc4, select = c("long", "lat"))
row.names(sccoord4) <- sc4$State
```

Distance matrix for each region

```
# Region 1
scd1 <- as.matrix(stats::dist(select(sc1, long, lat), diag = TRUE, upper = TRUE))
scd1 <- as.data.frame(scd1)
colnames(scd1) <- sc1$State
rownames(scd1) <- sc1$State

#####################################################

# Region 2
scd2 <- as.matrix(stats::dist(select(sc2, long, lat), diag = TRUE, upper = TRUE))
scd2 <- as.data.frame(scd2)
colnames(scd2) <- sc2$State
rownames(scd2) <- sc2$State

#####################################################

# Region 3
scd3 <- as.matrix(stats::dist(select(sc3, long, lat), diag = TRUE, upper = TRUE))
scd3 <- as.data.frame(scd3)
colnames(scd3) <- sc3$State
rownames(scd3) <- sc3$State
```

```
#####################################################

# Region 4
scd4 <- as.matrix(stats::dist(select(sc4, long, lat), diag = TRUE, upper = TRUE))
scd4 <- as.data.frame(scd4)
colnames(scd4) <- sc4$State
rownames(scd4) <- sc4$State
```

Create new distance matrix for each region for fitness function based on:

$$\frac{Patient}{Cost \times Distance \times HospitalPerformanceRating}$$

```
# Region 1
sc1$crp <- sc1$cost * sc1$hrp
crp1 <- sc1$crp
crp1 <- as.data.frame(crp1)
row.names(crp1) <- sc1$State

## transpose
crp1 <- t(crp1)
crp1 <- as.data.frame(crp1)

## Create new distance matrix accounting all variables
scd1_w <- data.frame(mapply(`*`, scd1, crp1))
colnames(scd1_w) <- sc1$State
row.names(scd1_w) <- sc1$State

#####################################################

# Region 2
sc2$crp <- sc2$cost * sc2$hrp
crp2 <- sc2$crp
crp2 <- as.data.frame(crp2)
row.names(crp2) <- sc2$State

## transpose
crp2 <- t(crp2)
crp2 <- as.data.frame(crp2)

## Create new distance matrix accounting all variables
scd2_w <- data.frame(mapply(`*`, scd2, crp2))
colnames(scd2_w) <- sc2$State
row.names(scd2_w) <- sc2$State

#####################################################

# Region 3
sc3$crp <- sc3$cost * sc3$hrp
crp3 <- sc3$crp
crp3 <- as.data.frame(crp3)
row.names(crp3) <- sc3$State

## transpose
```

```r
crp3 <- t(crp3)
crp3 <- as.data.frame(crp3)

## Create new distance matrix accounting all variables
scd3_w <- data.frame(mapply(`*`, scd3, crp3))
colnames(scd3_w) <- sc3$State
row.names(scd3_w) <- sc3$State

#####################################################

# Region 4
sc4$crp <- sc4$cost * sc4$hrp
crp4 <- sc4$crp
crp4 <- as.data.frame(crp4)
row.names(crp4) <- sc4$State

## transpose
crp4 <- t(crp4)
crp4 <- as.data.frame(crp4)

## Create new distance matrix accounting all variables
scd4_w <- data.frame(mapply(`*`, scd4, crp4))
colnames(scd4_w) <- sc4$State
row.names(scd4_w) <- sc4$State
```

Apply the fitness function to GA for each region.

```r
# Region 1
nr <- nrow(scd1_w)
GA.fitr1 <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd1_w,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fitr1)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
```

```
##
## GA results:
## Iterations             = 1011
## Fitness function value = 1.909534e-07
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12
## [1,] 10  3  8  1 11  4  7  6 12   9   2   5
## [2,]  2  5 10  3  8  1 11  4  7   6  12   9
## [3,] 12  9  2  5 10  3  8  1 11   4   7   6
## [4,]  9  2  5 10  3  8  1 11  4   7   6  12
## [5,]  1 11  4  7  6 12  9  2  5  10   3   8
```

```
#####################################################

# Region 2
nr <- nrow(scd2_w)
GA.fitr2 <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd2_w,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fitr2)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 1000
## Fitness function value = 1.669706e-07
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9
## [1,]  5  1  6  2  7  9  4  8  3
## [2,]  9  4  8  3  5  1  6  2  7
## [3,]  8  3  5  1  6  2  7  9  4
## [4,]  1  6  2  7  9  4  8  3  5
## [5,]  6  2  7  9  4  8  3  5  1
```

```
#######################################################

# Region 3
nr <- nrow(scd3_w)
GA.fitr3 <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd3_w,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fitr3)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 1079
## Fitness function value = 9.212388e-08
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10  ...  x13 x14
## [1,] 14  2  6  9 12 10  4 11 13   5         7   3
## [2,] 12 10  4 11 13  5  8  1  7   3         6   9
## [3,]  4 11 13  5  8  1  7  3 14   2        12  10
## [4,]  9 12 10  4 11 13  5  8  1   7         2   6
## [5,]  2  6  9 12 10  4 11 13  5   8         3  14
```

```
#######################################################

# Region 4
nr <- nrow(scd4_w)
GA.fitr4 <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd4_w,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
```

```
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fitr4)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 1041
## Fitness function value = 1.028311e-07
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12
## [1,]  1  3  2  9  7 11 12  8  6   4   5  10
## [2,] 11 12  8  6  4  5 10  1  3   2   9   7
## [3,]  8  6  4  5 10  1  3  2  9   7  11  12
## [4,]  4  5 10  1  3  2  9  7 11  12   8   6
## [5,]  6  4  5 10  1  3  2  9  7  11  12   8
```

Prepare data for visuallization

```
# Region 1
n = nrow(sc1)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr1@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis1 <- cbind(id, sc1)
colnames(vis1)[1] <- "id"
vis1 <- merge(vis1, tour)
vis1 <- vis1 %>%
  arrange(path_order)

#####################################################

# Region 2
n = nrow(sc2)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr2@solution[1,]
tour <- as.data.frame(tour)
```

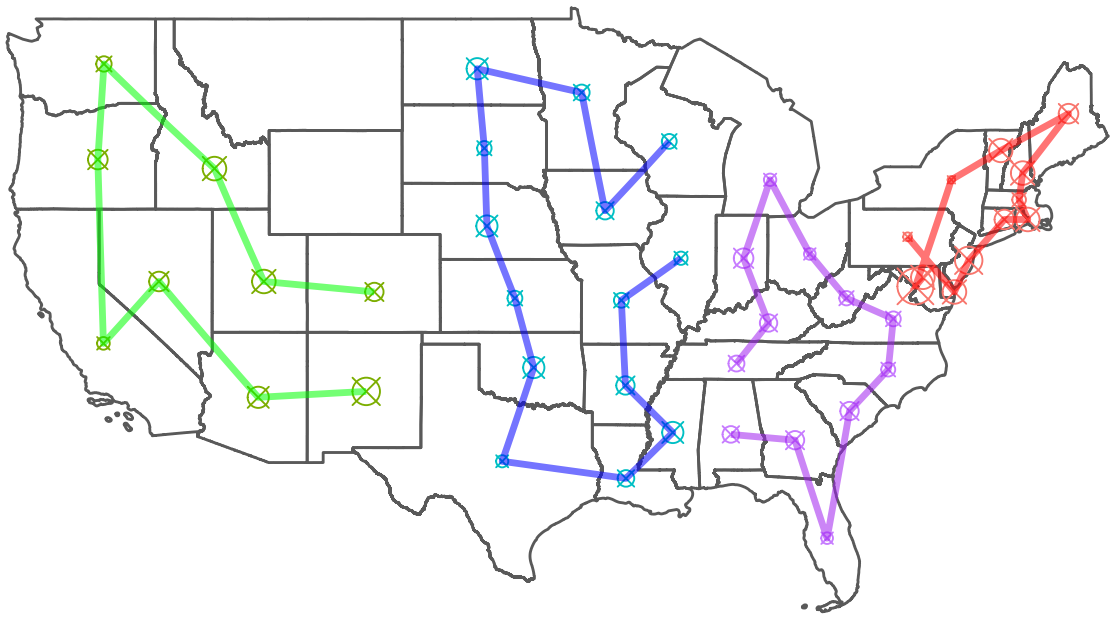
```
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis2 <- cbind(id, sc2)
colnames(vis2)[1] <- "id"
vis2 <- merge(vis2, tour)
vis2 <- vis2 %>%
  arrange(path_order)

#####################################################

# Region 3
n = nrow(sc3)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr3@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis3 <- cbind(id, sc3)
colnames(vis3)[1] <- "id"
vis3 <- merge(vis3, tour)
vis3 <- vis3 %>%
  arrange(path_order)

#####################################################

# Region 4
n = nrow(sc4)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr4@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis4 <- cbind(id, sc4)
colnames(vis4)[1] <- "id"
vis4 <- merge(vis4, tour)
vis4 <- vis4 %>%
  arrange(path_order)
```

Visualization

```
p8 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
```

```
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc,
    shape = 13,
    aes(
      x = long,
      y = lat,
      colour = factor(region),
      size = (1/crp),
    )
  ) + geom_path(
  data = vis1,
  color = "red",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
)+ geom_path(
  data = vis2,
  color = "green",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
)+ geom_path(
  data = vis3,
  color = "blue",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
)+ geom_path(
  data = vis4,
  color = "purple",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
) + coord_map() + labs(
```

```
  x = NULL,
  y = NULL,
  title = "SC - Kmeans - FF2 - Unnormalized"
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p8
```




```
# Save image file
pdf(
  file = "image/p8.pdf",
  width = 12,
  height = 7
)

p8
```

```
dev.off()
```

```
## pdf
##   2
```

```
jpeg(
  file = "image/sc_ff2_un_kmeans.jpeg"
)

p8

dev.off()
```

```
## pdf
##   2
```

9. Using Normalized Values

```
scd <- as.matrix(stats::dist(select(sc, long, lat), diag = TRUE, upper = TRUE))
scd <- as.data.frame(scd)
colnames(scd) <- sc$State
rownames(scd) <- sc$State
scd_n <- scd/max(scd)
```

```
sc_n <- sc
sc_n$cost <- sc_n$cost/max(sc_n$cost)
sc_n$patient <- sc_n$patient/max(sc_n$patient)
sc_n$hp <- sc_n$hp/max(sc_n$hp)
sc_n$hrp <- sc_n$hp/sc_n$patient
sc_n$hrp <- sc_n$hrp/max(sc_n$hrp)
sc_n$crp <- sc_n$cost * sc_n$hrp
sc_n$crp <- sc_n$crp/max(sc_n$crp)

crp_n <- sc_n$crp
crp_n <- as.data.frame(crp_n)


# transpose
crp_n <- t(crp_n)
crp_n <- as.data.frame(crp_n)

# Create new distance matrix accounting all variables
scd_nw <- data.frame(mapply(`*`, scd_n, crp_n))
colnames(scd_nw) <- sc_n$State
row.names(scd_nw) <- sc_n$State
```

Apply the fitness function to GA to find out the best route.

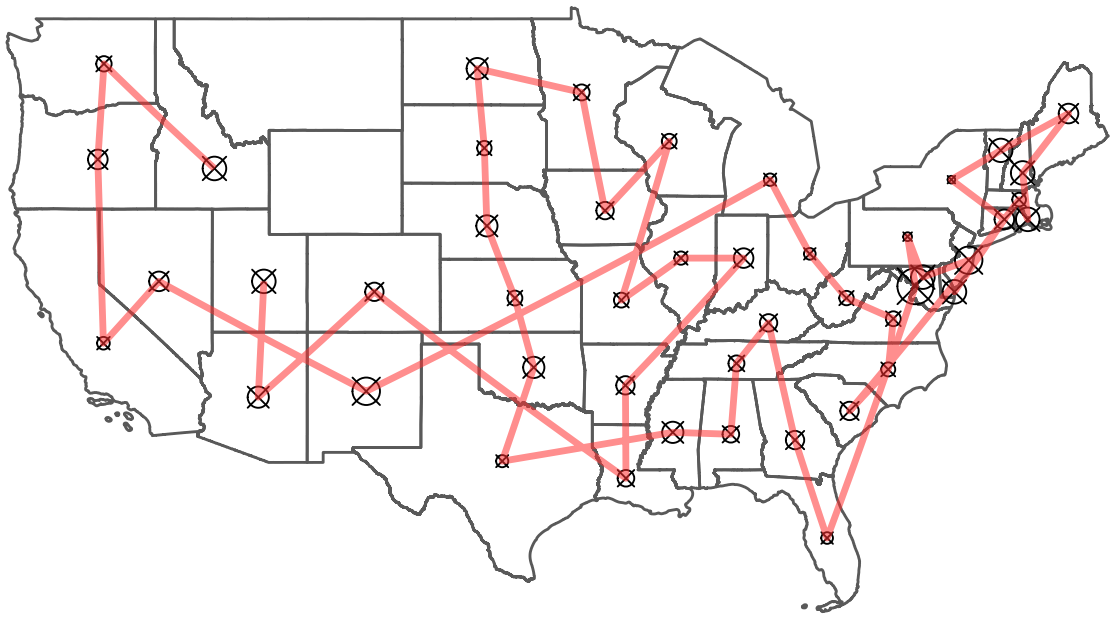
```
nr <- nrow(scd_nw)
GA.fit <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd_nw,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fit)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 5000
## Fitness function value = 1.043082
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10  ...  x46 x47
## [1,] 42  3  5 17  2 14 13 23 46  11       45  12
## [2,] 12 42  3  5 17  2 14 13 23  46       35  45
## [3,]  3  5 17  2 14 13 23 46 11  22       12  42
```

Visualization

```
n = nrow(sc_n)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fit@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis <- cbind(id, sc_n)
colnames(vis)[1] <- "id"
vis <- merge(vis, tour)
vis <- vis %>%
  arrange(path_order)

p9 <- ggplot() +
```

```
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc_n,
    shape = 13,
    aes(
      x = long,
      y = lat,
      size = (1/crp)
    )
  ) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "State Centers - FF2 - Normalized"
) + geom_path(
  data = vis,
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  ),
  color = "firebrick1",
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p9
```




```
# Save image file
pdf(
  file = "image/p9.pdf",
  width = 12,
  height = 7
)

p9

dev.off()
```

```
## pdf
##   2
```

```
jpeg(
  file = "image/sc_ff2_nor.jpeg"
)

p9

dev.off()
```

```
## pdf
##   2
```

10. K-means Clustering with Normalized Data

Cluster state center using k-means clustering approach into *four* clusters.

Note: Number of clusters and cluster centers can be changed based on different situations, such as amount of distribution center and location of the distribution center.

```
# Clustering based on distance
km_n <- kmeans(scd_n, centers = 4)

sc_n$region <- km_n$cluster
```

Subset dataframe into four sub dataframe based on region cluster:

```
# Region 1
sc1_n <- subset(sc_n, region == 1)
sccoord1_n <- subset(sc1_n, select = c("long", "lat"))
row.names(sccoord1_n) <- sc1_n$State

#####################################################

# Region 2
sc2_n <- subset(sc_n, region == 2)
sccoord2_n <- subset(sc2_n, select = c("long", "lat"))
row.names(sccoord2_n) <- sc2_n$State

#####################################################

# Region 3
sc3_n <- subset(sc_n, region == 3)
sccoord3_n <- subset(sc3_n, select = c("long", "lat"))
row.names(sccoord3_n) <- sc3_n$State

#####################################################

# Region 4
sc4_n <- subset(sc_n, region == 4)
sccoord4_n <- subset(sc4_n, select = c("long", "lat"))
row.names(sccoord4_n) <- sc4_n$State
```

Distance matrix for each region

```
# Region 1
scd1_n <- as.matrix(stats::dist(select(sc1_n, long, lat), diag = TRUE, upper = TRUE))
scd1_n <- as.data.frame(scd1_n)
colnames(scd1_n) <- sc1_n$State
rownames(scd1_n) <- sc1_n$State
scd1_n <- scd1_n/max(scd1_n)

#####################################################

# Region 2
scd2_n <- as.matrix(stats::dist(select(sc2_n, long, lat), diag = TRUE, upper = TRUE))
```

```r
scd2_n <- as.data.frame(scd2_n)
colnames(scd2_n) <- sc2_n$State
rownames(scd2_n) <- sc2_n$State
scd2_n <- scd2_n/max(scd2_n)

#####################################################

# Region 3
scd3_n <- as.matrix(stats::dist(select(sc3_n, long, lat), diag = TRUE, upper = TRUE))
scd3_n <- as.data.frame(scd3_n)
colnames(scd3_n) <- sc3_n$State
rownames(scd3_n) <- sc3_n$State
scd3_n <- scd3_n/max(scd3_n)

#####################################################

# Region 4
scd4_n <- as.matrix(stats::dist(select(sc4_n, long, lat), diag = TRUE, upper = TRUE))
scd4_n <- as.data.frame(scd4_n)
colnames(scd4_n) <- sc4_n$State
rownames(scd4_n) <- sc4_n$State
scd4_n <- scd4_n/max(scd4_n)
```

Create new distance matrix for each region for fitness function based on:

$$\frac{Patient}{Cost \times Distance \times HospitalPerformanceRating}$$

```r
# Region 1
crp1_n <- sc1_n$crp
crp1_n <- as.data.frame(crp1_n)
row.names(crp1_n) <- sc1_n$State

## transpose
crp1_n <- t(crp1_n)
crp1_n <- as.data.frame(crp1_n)

## Create new distance matrix accounting all variables
scd1_wn <- data.frame(mapply(`*`, scd1_n, crp1_n))
colnames(scd1_wn) <- sc1_n$State
row.names(scd1_wn) <- sc1_n$State

#####################################################

# Region 2
crp2_n <- sc2_n$crp
crp2_n <- as.data.frame(crp2_n)
row.names(crp2_n) <- sc2_n$State

## transpose
crp2_n <- t(crp2_n)
crp2_n <- as.data.frame(crp2_n)

## Create new distance matrix accounting all variables
```

```r
scd2_wn <- data.frame(mapply(`*`, scd2_n, crp2_n))
colnames(scd2_wn) <- sc2_n$State
row.names(scd2_wn) <- sc2_n$State

#####################################################

# Region 3
crp3_n <- sc3_n$crp
crp3_n <- as.data.frame(crp3_n)
row.names(crp3_n) <- sc3_n$State

## transpose
crp3_n <- t(crp3_n)
crp3_n <- as.data.frame(crp3_n)

## Create new distance matrix accounting all variables
scd3_wn <- data.frame(mapply(`*`, scd3_n, crp3_n))
colnames(scd3_wn) <- sc3_n$State
row.names(scd3_wn) <- sc3_n$State

#####################################################

# Region 4
crp4_n <- sc4_n$crp
crp4_n <- as.data.frame(crp4_n)
row.names(crp4_n) <- sc4_n$State

## transpose
crp4_n <- t(crp4_n)
crp4_n <- as.data.frame(crp4_n)

## Create new distance matrix accounting all variables
scd4_wn <- data.frame(mapply(`*`, scd4_n, crp4_n))
colnames(scd4_wn) <- sc4_n$State
row.names(scd4_wn) <- sc4_n$State
```

Apply the fitness function to GA for each region.

```r
# Region 1
nr <- nrow(scd1_wn)
GA.fitr1_n <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd1_wn,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)
```

```r
summary(GA.fitr1_n)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 1132
## Fitness function value = 1.557263
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12
## [1,] 12  9  2  5 10  3  8  1 11   4   7   6
## [2,]  6 12  9  2  5 10  3  8  1  11   4   7
## [3,]  2  5 10  3  8  1 11  4  7   6  12   9
## [4,] 11  4  7  6 12  9  2  5 10   3   8   1
## [5,]  4  7  6 12  9  2  5 10  3   8   1  11
```

```r
#####################################################

# Region 2
nr <- nrow(scd2_wn)
GA.fitr2_n <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd2_wn,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fitr2_n)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
```

```
## Iterations             = 1059
## Fitness function value = 1.243633
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12
## [1,]  2  9  7 11 12  8  6  4  5  10   1   3
## [2,]  1  3  2  9  7 11 12  8  6   4   5  10
## [3,]  7 11 12  8  6  4  5 10  1   3   2   9
## [4,] 10  1  3  2  9  7 11 12  8   6   4   5
## [5,]  6  4  5 10  1  3  2  9  7  11  12   8
```

```
#####################################################

# Region 3
nr <- nrow(scd3_wn)
GA.fitr3_n <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd3_wn,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fitr3_n)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 1181
## Fitness function value = 1.316765
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10  ...  x13 x14
## [1,]  2  6  9 12 10  4 11 13  5   8         3  14
## [2,]  7  3 14  2  6  9 12 10  4  11         8   1
## [3,] 14  2  6  9 12 10  4 11 13   5         7   3
## [4,]  1  7  3 14  2  6  9 12 10   4         5   8
## [5,] 10  4 11 13  5  8  1  7  3  14         9  12
```

```
#####################################################

# Region 4
```

```
nr <- nrow(scd4_wn)
GA.fitr4_n <- ga(
  type = "permutation",
  fitness = fitness1,
  distMatrix = scd4_wn,
  lower = 1,
  upper = nr,
  popSize = 100,
  maxiter = 5000,
  run = 1000,
  pmutation = 0.2,
  monitor = NULL
)

summary(GA.fitr4_n)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  permutation
## Population size       =  100
## Number of generations =  5000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.2
##
## GA results:
## Iterations             = 1067
## Fitness function value = 2.456996
## Solutions =
##      x1 x2 x3 x4 x5 x6 x7 x8 x9
## [1,]  4  8  3  5  1  6  2  7  9
## [2,]  5  1  6  2  7  9  4  8  3
## [3,]  6  2  7  9  4  8  3  5  1
## [4,]  1  6  2  7  9  4  8  3  5
## [5,]  3  5  1  6  2  7  9  4  8
```

Prepare data for visualization

```
# Region 1
n = nrow(sc1_n)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr1_n@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis1 <- cbind(id, sc1_n)
colnames(vis1)[1] <- "id"
vis1 <- merge(vis1, tour)
vis1 <- vis1 %>%
```

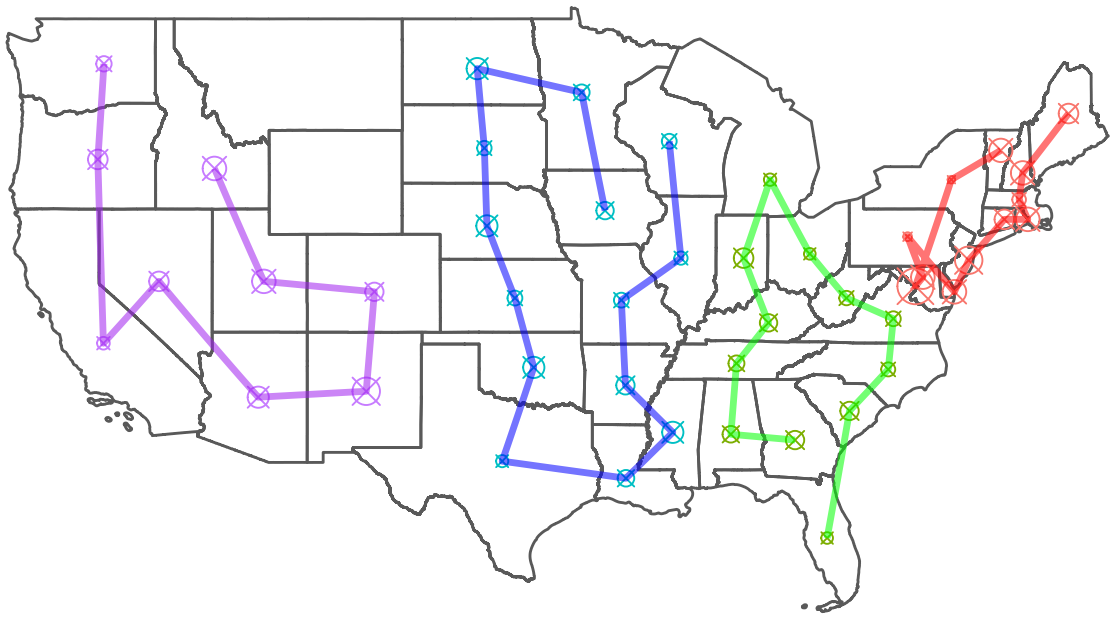
```
  arrange(path_order)

#######################################################

# Region 2
n = nrow(sc2_n)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr2_n@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis2 <- cbind(id, sc2_n)
colnames(vis2)[1] <- "id"
vis2 <- merge(vis2, tour)
vis2 <- vis2 %>%
  arrange(path_order)

#######################################################

# Region 3
n = nrow(sc3_n)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr3_n@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis3 <- cbind(id, sc3_n)
colnames(vis3)[1] <- "id"
vis3 <- merge(vis3, tour)
vis3 <- vis3 %>%
  arrange(path_order)

#######################################################

# Region 4
n = nrow(sc4_n)
id = list(1:n)
id <- as.data.frame(id)
tour <- GA.fitr4_n@solution[1,]
tour <- as.data.frame(tour)
path <- list(1:n)
path <- as.data.frame(path)
tour <- cbind(tour, path)
colnames(tour) <- c("id", "path_order")
vis4 <- cbind(id, sc4_n)
colnames(vis4)[1] <- "id"
vis4 <- merge(vis4, tour)
```

```r
vis4 <- vis4 %>%
  arrange(path_order)
```

Visualization

```r
p10 <- ggplot() +
  geom_polygon(
    data = US,
    aes(
      x = long,
      y = lat,
      group = group,
      fill = NA
    ),
    color = "grey35",
    fill = NA,
    size = 0.5
  ) + geom_point(
    data = sc_n,
    shape = 13,
    aes(
      x = long,
      y = lat,
      colour = factor(region),
      size = (1/crp),
    )
  ) + geom_path(
  data = vis1,
  color = "red",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
)+ geom_path(
  data = vis2,
  color = "green",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
)+ geom_path(
  data = vis3,
  color = "blue",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
```

```
)+ geom_path(
  data = vis4,
  color = "purple",
  size = 1.2,
  aes(
    x = long,
    y = lat,
    alpha = 0.9
  )
) + coord_map() + labs(
  x = NULL,
  y = NULL,
  title = "State Centers - FF2 - Normalized - Kmeans"
) + theme(
  axis.line =  element_blank(),
  axis.text =  element_blank(),
  axis.ticks =  element_blank(),
  legend.position = "none",
  plot.title = element_text(hjust = 0.5, size = 18),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank()
)

p10
```




```
# Save image file
pdf(
  file = "image/p10.pdf",
  width = 12,
  height = 7
)

p10

dev.off()
```

```
## pdf
##   2
```

```
jpeg(
  file = "image/sc_ff2_nor_kmeans.jpeg"
)

p10

dev.off()
```

```
## pdf
##   2
```