Temperature cluster analysis

Rosie

7/16/2020

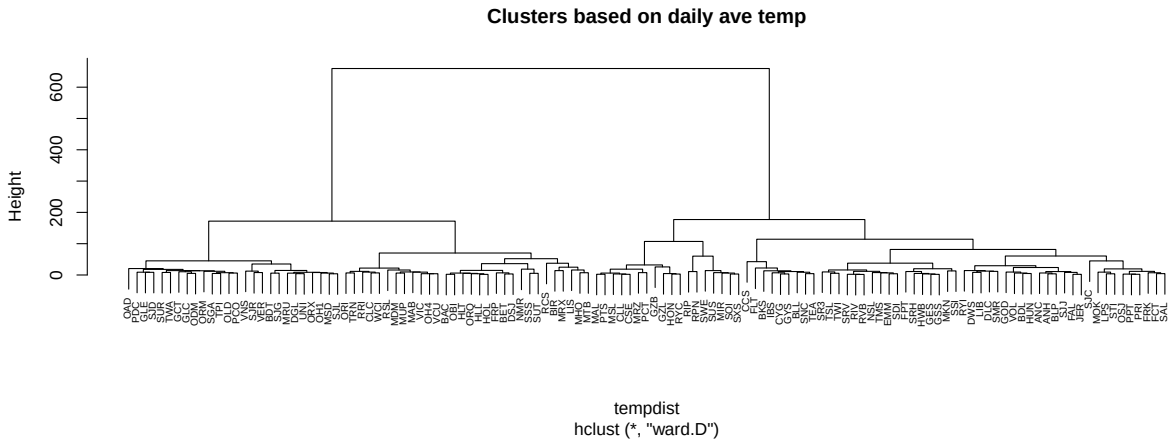
Cluster analysis

So, Larry was unceratain as to how to do cluster analysis when we "just have one variable". While it's true that we only have temperature to work with, if we really only just had one variable we could group stations with similar average temperatures and be done with it. However, we really do have a multivariate dataset because each station has different average temperatures at different times of year, and different variance in tempareture. We want to group stations with similar temperature regimes, rather than just similar averages.

Now, there are a number of different ways to describe the temperature regime, but I thought I would start by taking the average daily mean temperature over the past five years.

I used Cat's continuous data set rather than the descrete data set because it is a more balanced design where we will have a mean temperature for each day of the year and we know it was taken at the exact same spot.

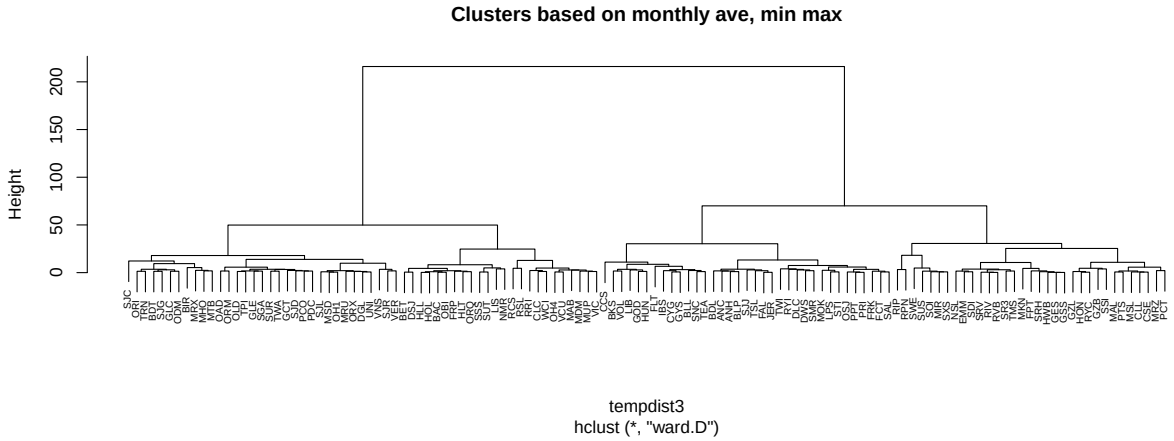
```
#first calculate the daily means
tempmean = temps %>%
  filter(Date > as.Date("2014-1-1"), Station != "RYF") %>%
  mutate(julian = yday(Date)) %>%
  group_by(Station, julian) %>%
  summarize(Temp = mean(Temp, na.rm = T))

#put it into wide format for the cluster analysis
tempwide = pivot_wider(tempmean, id_cols = c(Station),
                       names_from = julian, values_from = Temp)

row.names(tempwide) = tempwide$Station

#calculate distance and cluster
tempdist = dist(tempwide, "euclidean")
tempfit = hclust(tempdist, method = "ward.D")
plot(tempfit, main = "Clusters based on daily ave temp", cex = 0.6)
```



Or we could use monthly means, mins, and maxes

I haven't sat down to compare the monthly mins, means, and maxes with the daily means, but a quick look seems like they are pretty similar

```
#monthly mean, min, and max per year
tempmo = temps %>%
  filter(Date > as.Date("2014-1-1"), Station != "RYF") %>%
  mutate(Month = month(Date), Year = year(Date)) %>%
  group_by(Station, Month, Year) %>%
  summarize(Temp = mean(Temp, na.rm = T), min = min(Temp), max = max(Temp))

#average over five years
tempmo2 = tempmo %>%
  group_by(Station, Month) %>%
  summarize(Temp = mean(Temp, na.rm = T), min = mean(min, na.rm = T), max = mean(max, na.rm = T))

#put it into wide format for the cluster analysis
tempmowide = pivot_wider(tempmo2, id_cols = c(Station),
                         names_from = Month, values_from = Temp)
tempmowide2 = pivot_wider(tempmo2, id_cols = c(Station),
                          names_from = Month, values_from = min)
tempmowide3 = pivot_wider(tempmo2, id_cols = c(Station),
                          names_from = Month, values_from = max)

tempmowide4 = cbind(tempmowide, tempmowide2[,-1], tempmowide3[,-1])

row.names(tempmowide4) = tempmowide4$Station

#calculate distance and cluster
tempdist3 = dist(tempmowide4, "euclidean")
tempfit3 = hclust(tempdist3, method = "ward.D")
plot(tempfit3, main = "Clusters based on monthly ave, min max", cex = 0.6)
```

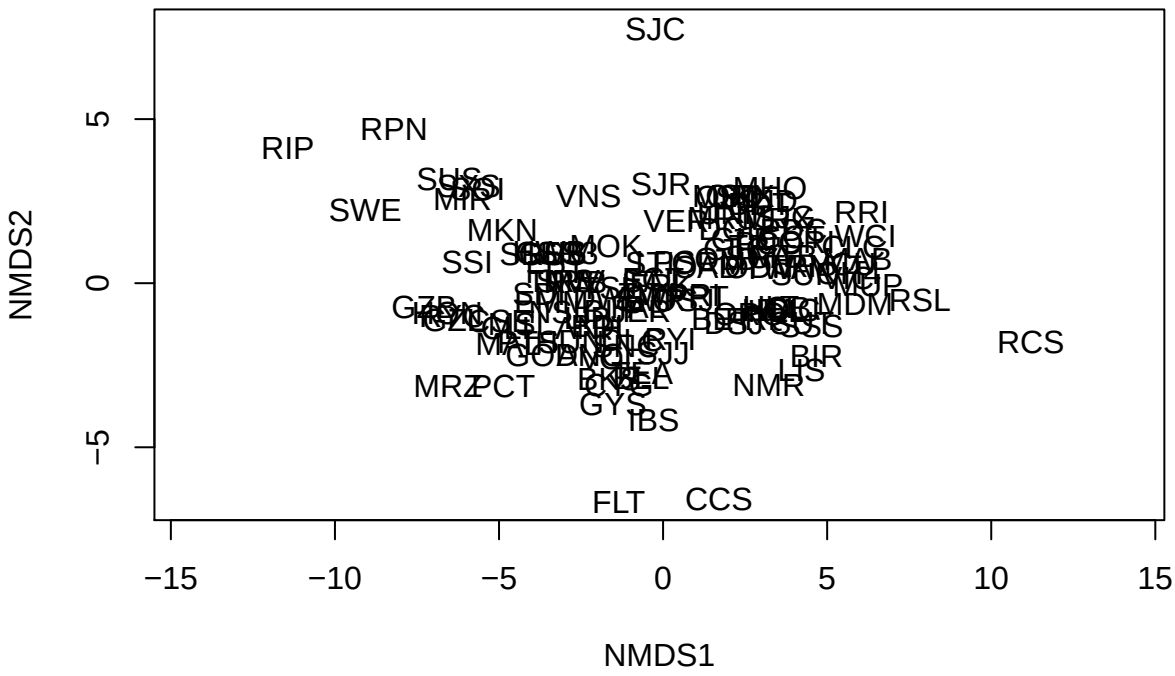


I thought I'd look at a quick NMDS too, just for fun

```
## Run 0 stress 0.09461362
## Run 1 stress 0.4134158
## Run 2 stress 0.1198215
## Run 3 stress 0.1235378
## Run 4 stress 0.1100874
## Run 5 stress 0.1230903
## Run 6 stress 0.1034393
## Run 7 stress 0.1311104
## Run 8 stress 0.1212343
## Run 9 stress 0.1213021
## Run 10 stress 0.103813
## Run 11 stress 0.1071406
## Run 12 stress 0.1254605
## Run 13 stress 0.1020636
## Run 14 stress 0.1302668
## Run 15 stress 0.1020669
## Run 16 stress 0.1204735
## Run 17 stress 0.1052757
## Run 18 stress 0.1020665
## Run 19 stress 0.1163383
## Run 20 stress 0.1170579
## Run 21 stress 0.1184351
## Run 22 stress 0.1112557
## Run 23 stress 0.1148043
## Run 24 stress 0.1226171
## Run 25 stress 0.09400237
## ... New best solution
## ... Procrustes: rmse 0.01105611  max resid 0.1131554
## Run 26 stress 0.1146205
## Run 27 stress 0.1144019
## Run 28 stress 0.1225051
## Run 29 stress 0.413817
## Run 30 stress 0.09400343
## ... Procrustes: rmse 0.001036729  max resid 0.01024476
## Run 31 stress 0.1180943
## Run 32 stress 0.1238813
## Run 33 stress 0.1078742
```

```
## Run 34 stress 0.1296998
## Run 35 stress 0.1206526
## Run 36 stress 0.125537
## Run 37 stress 0.1154987
## Run 38 stress 0.1121056
## Run 39 stress 0.1167273
## Run 40 stress 0.1038152
## Run 41 stress 0.1202771
## Run 42 stress 0.1287304
## Run 43 stress 0.09461239
## Run 44 stress 0.1067135
## Run 45 stress 0.09358632
## ... New best solution
## ... Procrustes: rmse 0.004396179  max resid 0.04207121
## Run 46 stress 0.1038147
## Run 47 stress 0.1034394
## Run 48 stress 0.1197594
## Run 49 stress 0.120465
## Run 50 stress 0.1181313
## Run 51 stress 0.1097443
## Run 52 stress 0.09461702
## Run 53 stress 0.1091478
## Run 54 stress 0.1165857
## Run 55 stress 0.1145759
## Run 56 stress 0.120821
## Run 57 stress 0.1265386
## Run 58 stress 0.1181592
## Run 59 stress 0.1038192
## Run 60 stress 0.1317586
## Run 61 stress 0.122477
## Run 62 stress 0.1047098
## Run 63 stress 0.1062928
## Run 64 stress 0.1273197
## Run 65 stress 0.1208449
## Run 66 stress 0.1181106
## Run 67 stress 0.108735
## Run 68 stress 0.1159627
## Run 69 stress 0.1170873
## Run 70 stress 0.1236524
## Run 71 stress 0.09399793
## ... Procrustes: rmse 0.00437635  max resid 0.04000557
## Run 72 stress 0.1070442
## Run 73 stress 0.1161488
## Run 74 stress 0.1038184
## Run 75 stress 0.1116412
## Run 76 stress 0.1181513
## Run 77 stress 0.1215347
## Run 78 stress 0.1071399
## Run 79 stress 0.1263651
## Run 80 stress 0.1165159
## Run 81 stress 0.1185614
## Run 82 stress 0.1020646
## Run 83 stress 0.1184144
## Run 84 stress 0.1020643
```

```
## Run 85 stress 0.1177551
## Run 86 stress 0.1133771
## Run 87 stress 0.1184125
## Run 88 stress 0.1140713
## Run 89 stress 0.09491905
## Run 90 stress 0.1227924
## Run 91 stress 0.1143188
## Run 92 stress 0.1222437
## Run 93 stress 0.09399829
## ... Procrustes: rmse 0.004444192  max resid 0.04057667
## Run 94 stress 0.1226167
## Run 95 stress 0.1071389
## Run 96 stress 0.1030985
## Run 97 stress 0.1093229
## Run 98 stress 0.1181532
## Run 99 stress 0.09400023
## ... Procrustes: rmse 0.004332456  max resid 0.04033447
## Run 100 stress 0.1124853
## Run 101 stress 0.1147481
## Run 102 stress 0.1177528
## Run 103 stress 0.1158283
## Run 104 stress 0.1125075
## Run 105 stress 0.1071487
## Run 106 stress 0.1177085
## Run 107 stress 0.1263039
## Run 108 stress 0.1107153
## Run 109 stress 0.09461893
## Run 110 stress 0.1071496
## Run 111 stress 0.1165808
## Run 112 stress 0.117511
## Run 113 stress 0.09491558
## Run 114 stress 0.1052768
## Run 115 stress 0.1150838
## Run 116 stress 0.09400224
## ... Procrustes: rmse 0.004542968  max resid 0.04273873
## Run 117 stress 0.09461758
## Run 118 stress 0.09359307
## ... Procrustes: rmse 0.0005252891  max resid 0.003457407
## ... Similar to previous best
## *** Solution reached
```



CDEC stations

Now the question is whether teh clusters actually work out geographically. I'm going to have to spend some time with this map printed out and a highlighter to see how it lines up.

Cutting down the trees

Now I can cut my trees into groups to see how they map out.

```
cutday = as.data.frame(cutree(tempfit, k = c(2,4,6,12)))
cutday$Station = row.names(cutday)
```

Plot

I trimmed the tree into different numbers of groups, so see what we get when we try to divide it into 2,4,6, or 12 regions.

```
library("sf")
```

```
## Linking to GEOS 3.8.0, GDAL 3.0.4, PROJ 6.3.1
```

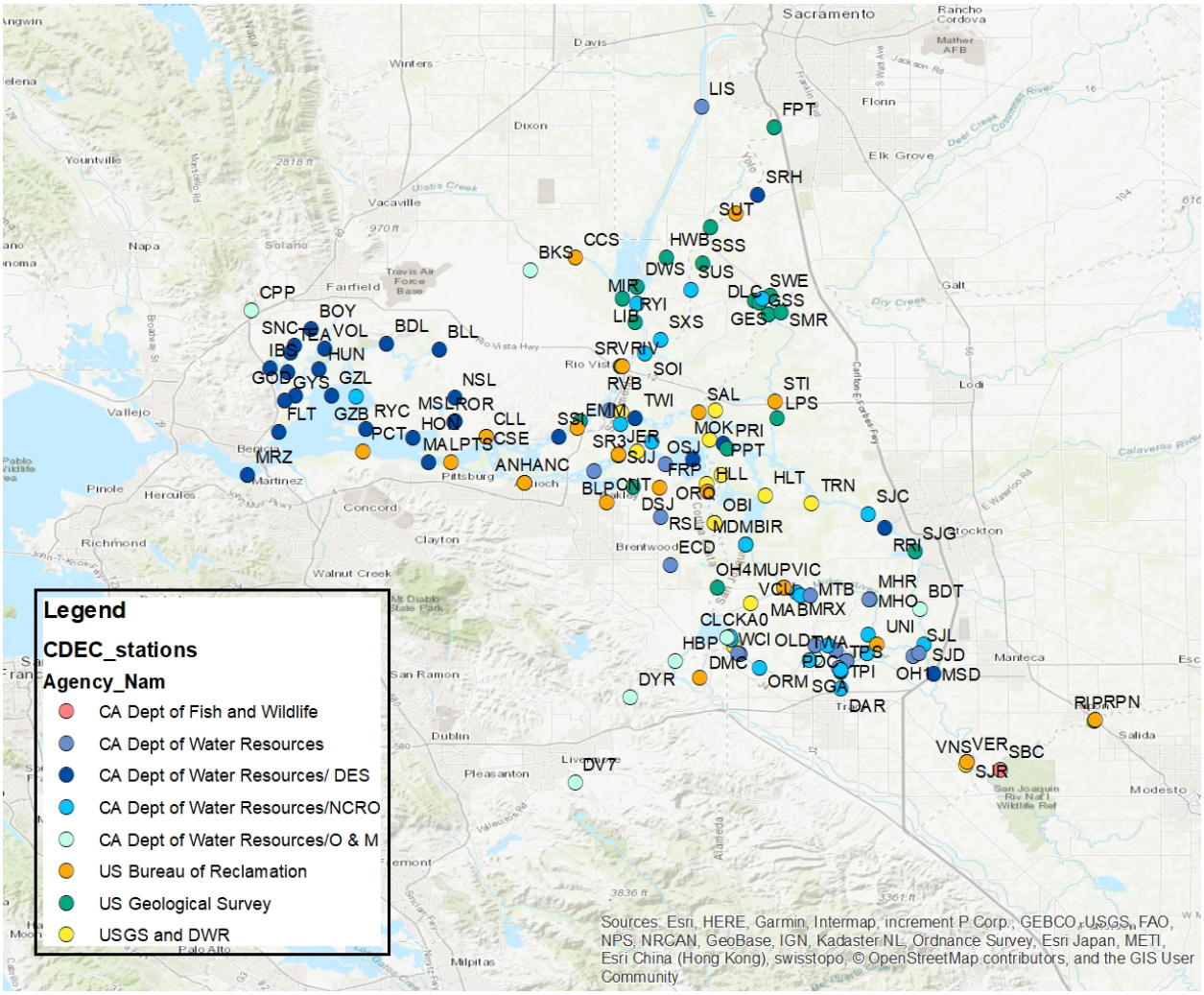


Figure 1: CDEC stations included in analysis

```
library(dplyr)
library(ggplot2)
library(leaflet)
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
#read in shapefile of the delta
delta = read_sf("deltashap/hydro_delta_marsh.shp")

#add lat/longs for the stations
stas = read.csv("StationLatLongs.csv")

cutday = merge(cutday, stas)
names(cutday) = c("Station", "grps2", "grps4", "grps6", "grps12", "Latitude","Longitude")

#turn it into a spatial object
stashap = st_as_sf(cutday,
                   coords = c("Longitude", "Latitude"),
                   crs = 4326)

gp4 = ggplot() +
  geom_sf(data = delta, alpha = 0.5)+
  geom_sf(data = stashap, mapping =aes(color = as.factor(grps4)))+
  theme_bw() + guides(color = FALSE) +
  coord_sf(xlim = c(-122.2, -121), ylim = c(37.6, 38.8))+
  ggtitle("4 groups")

gp6 = ggplot() +
  geom_sf(data = delta, alpha = 0.5)+
  geom_sf(data = stashap, mapping =aes(color = as.factor(grps6)))+
  theme_bw() + guides(color = FALSE) +
  coord_sf(xlim = c(-122.2, -121), ylim = c(37.6, 38.8))+
  ggtitle("6 groups")

gp12 = ggplot() +
```

```
  geom_sf(data = delta, alpha = 0.5)+
  geom_sf(data = stashap, mapping =aes(color = as.factor(grps6)))+
  theme_bw() + guides(color = FALSE) +
  coord_sf(xlim = c(-122.2, -121), ylim = c(37.6, 38.8))+
  ggtitle("6 groups")

library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(gp4, gp6, gp12, nrow = 1, ncol = 3)
```