Report

Makarios Ewing

2025-04-24

Project summary

This project takes American Community Survey household-count data for every U.S. county, fits low-order polynomials to the time series of married-couple and unmarried-solo households (2009–2022) to extract slopes, accelerations, and "no-curve" flags, normalizes those ten features per county, and then uses k-means (after removing Los Angeles as an outlier) to partition counties into four demographic-trend clusters. It visualizes cluster structure with elbow plots and t-SNE embeddings, shows example county trajectories for each group, and finally trains a random-forest on 2020 Social Vulnerability Index variables to see how well SVI can predict those household-trend clusters.

Non-Technical improvement

Right now most plots are just dropped into the report with generic file names. For a stronger case on a project, it is best to give each figure a descriptive caption.

For Example for the first graph in this project, I would write a caption stating:

Title: "Figure 1. Married vs. Unmarried Household Trends in San Francisco County (2009–2022)"

Caption: "Blue markers show married-couple households rising sharply until 2019 and then plateauing; red markers show unmarried-solo households inching up through 2020 before a slight dip. Data are ACS 5-year estimates."

This will tie the visuals back to our interpretations.

Technical Improvement

In the cluster.R code, i replaced this chunk of code:

```
scaled_df[['no_curve_married']] <- (is.na(scaled_df[['married.slope_2022']]) - 0.5) * 2
scaled_df[['no_curve_unmarried']] <- (is.na(scaled_df[['unmarried.slope_2022']]) - 0.5) *
2 for(i in seq_len(ncol(scaled_df))){ is_na_vals <- is.na(scaled_df[, i]) scaled_df[is_na_vals,
i] <- 0 # replaced with mean, just so kmeans will run! } no_m_yes_u <- (scaled_df[['no_curve_married']]
== 1) & (scaled_df[['no_curve_unmarried']] == -1) yes_m_no_u <- (scaled_df[['no_curve_married']]
== -1) & (scaled_df[['no_curve_unmarried']] == 1) no_m_no_u <- (scaled_df[['no_curve_married']]
== 1) & (scaled_df[['no_curve_unmarried']] == 1)
```

as the code does not match the comment, replacing the N/A values with), which You explicitly warned that "zero-imputation after scaling collapses all missing patterns to the same '0' value, hiding real signal and distorting distances," and that "median imputation is more robust to outliers." Yet the original code did exactly the opposite—imputing 0 after scaling, which collapses all missing patterns to the same "0" value therefore hiding real signal and distorting distances.

This was the chunk I inserted instead:

```
scaled_df <- scaled_df %>% mutate( no_curve_married   = as.integer(is.na(married.slope_2022)),
no_curve_unmarried = as.integer(is.na(unmarried.slope_2022)) )

num_cols <- setdiff( names(scaled_df), c("NAME", "no_curve_married", "no_curve_unmarried") )

for(col in num_cols) { med <- median(scaled_df[[col]], na.rm = TRUE) scaled_df[[col]][is.na(scaled_df[[co
<- med }

scaled_df[num_cols] <- lapply(scaled_df[num_cols], scale)

no_m_yes_u   <- (scaled_df$no_curve_married == 1) & (scaled_df$no_curve_unmarried == 0)
yes_m_no_u   <- (scaled_df$no_curve_married == 0) & (scaled_df$no_curve_unmarried == 1)
no_m_no_u    <- (scaled_df$no_curve_married == 1) & (scaled_df$no_curve_unmarried == 1)
```

Implementing the code

```
library(glue)
library(Rtsne)
```

```
## Warning: package 'Rtsne' was built under R version 4.4.3
```

```
library(RColorBrewer)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df <- read.csv("C:/Users/newli/Downloads/curve_feats_counties.csv")
df <- df[, -which(grepl('2023', names(df)))]
ts <- read.csv("C:/Users/newli/Downloads/with_geo_household_cnt.csv")

scaled_df <- apply(df[, -1], 2, scale, center=TRUE, scale=TRUE)
scaled_df <- as.data.frame(scaled_df)
scaled_df[['NAME']] <- df$X
head(scaled_df)
```

```
##   married.slope_2022 married.acc_2022 married.steady_slope married.val_2022
## 1         -0.1984301      0.007678976             0.813345       -0.2501364
## 2         -0.8186742     -0.948533132            -1.229038       -0.1384813
## 3         -0.1927101     -0.141894893            -1.229038       -0.2234970
## 4          2.0119921      0.007678976             0.813345        1.3801127
## 5         -0.1642366      0.007678976             0.813345       -0.2977631
## 6                 NA               NA                   NA       -0.2602941
```

```
##   unmarried.slope_2022 unmarried.acc_2022 unmarried.steady_slope
## 1                   NA                 NA                    NA
## 2           -0.6646374        -1.20485763            -1.1612969
## 3           -0.3901150        -0.06956781             0.8607721
## 4            2.3664557        -0.06956781             0.8607721
## 5           -0.1672607         0.79520880            -1.1612969
## 6                   NA                 NA                    NA
##   unmarried.val_2022                             NAME
## 1         -0.2405890 Abbeville County, South Carolina
## 2         -0.1716303         Acadia Parish, Louisiana
## 3         -0.2164870        Accomack County, Virginia
## 4          1.2480324                Ada County, Idaho
## 5         -0.2819396               Adair County, Iowa
## 6         -0.2649378             Adair County, Kentucky
```

```
scaled_df <- scaled_df %>%
  mutate(
    no_curve_married   = as.integer(is.na(married.slope_2022)),
    no_curve_unmarried = as.integer(is.na(unmarried.slope_2022))
  )

num_cols <- setdiff(
  names(scaled_df),
  c("NAME", "no_curve_married", "no_curve_unmarried")
)


for(col in num_cols) {
  med <- median(scaled_df[[col]], na.rm = TRUE)
  scaled_df[[col]][is.na(scaled_df[[col]])] <- med
}


scaled_df[num_cols] <- lapply(scaled_df[num_cols], scale)



no_m_yes_u   <- (scaled_df$no_curve_married == 1) & (scaled_df$no_curve_unmarried == 0)
yes_m_no_u   <- (scaled_df$no_curve_married == 0) & (scaled_df$no_curve_unmarried == 1)
no_m_no_u    <- (scaled_df$no_curve_married == 1) & (scaled_df$no_curve_unmarried == 1)
head(scaled_df)
```

```
##   married.slope_2022 married.acc_2022 married.steady_slope married.val_2022
## 1         -0.1888886      0.007048454            0.7104528       -0.2501364
## 2         -0.8633451     -1.034752749           -1.4071167       -0.1384813
## 3         -0.1826687     -0.155913545           -1.4071167       -0.2234970
## 4          2.2147352      0.007048454            0.7104528        1.3801127
## 5         -0.1517064      0.007048454            0.7104528       -0.2977631
## 6         -0.1437936      0.007048454            0.7104528       -0.2602941
##   unmarried.slope_2022 unmarried.acc_2022 unmarried.steady_slope
## 1           -0.1989526        -0.06215935             0.7178772
## 2           -0.6898102        -1.33162277            -1.3925640
## 3           -0.3842269        -0.06215935             0.7178772
## 4            2.6842363        -0.06215935             0.7178772
```

```
## 5           -0.1361577         0.90482040             -1.3925640
## 6           -0.1989526        -0.06215935              0.7178772
##   unmarried.val_2022                              NAME no_curve_married
## 1         -0.2405890 Abbeville County, South Carolina                0
## 2         -0.1716303         Acadia Parish, Louisiana                0
## 3         -0.2164870        Accomack County, Virginia                0
## 4          1.2480324                Ada County, Idaho                0
## 5         -0.2819396               Adair County, Iowa                0
## 6         -0.2649378            Adair County, Kentucky                1
##   no_curve_unmarried
## 1                  1
## 2                  0
## 3                  0
## 4                  0
## 5                  0
## 6                  1
```

As we can see, everything runs smoothly, and was corrected