Group exercise 2: Analysing spatial data

DATA5207: Data Analysis in the Social Sciences

Dr Shaun Ratcliff

Your next task is to complete your second group project. This task counts towards your group work grade and is worth 7.5 per cent of your total marks for the unit.

For your second group project, you will be looking at the opiate prescription crisis in the US. Some parts of the US have experienced significant problems with addiction to opiate-based painkillers. This has resulted in drug dependence and problems with overdoses and deaths. We will be examining the relationship between opiate drug prescriptions and poverty (your independent variables), and mortality caused by drugs and alcohol using the methods covered in this lab, and previously in this unit.

The spatial aspect of this is that we are going to use county-level data sourced from the *US Census Bureau* and the *Centers for Disease Control and Prevention (CDC)*.

When working on this project, you will need to explain what you are doing and why. This, plus your analysis (and code) will need to be written in *R Markdown*, and you are required to justify why you made specific design decisions.

You will submit your final written document (as a raw Markdown file) through a link that will be provided on canvas. You do not need to knit this, just submit the RMD with the data it needs to run, in a zipped folder. Only one member of your group needs to submit your work. As long as the groups are properly registered on canvas, all members of the group will receive the full grades awarded to the group.

You can take as long as you want to complete it, but over I do not expect each group member to spend much more than approximately four hours on the task. **Make sure your code runs as is on different computers**. Part of what you will be marked on is that your code runs when we try to knit the RMD file. **If a file fails to knit this will mean automatic failure of the task**.

We will use the same groups as the first task, unless you choose to move groups. Remember, your group (hopefully) has multiple members. Allocate different tasks to different members to increase your productivity.

Additional details on the requirements for this task are included below. Additionally, I have included some instructions on loading and cleaning these data on the following pages.

Instructions for this task

To complete this task properly, you should follow the instructions below. Remember, you are marked using the criteria on the rubric published on the *Assessment information* page on canvas.

Once you have loaded and cleaned your data, you will use these to examine the relationships between the variables. How does poverty and opiate prescriptions predict mortality rates associated with drugs and alcohol? Some other data is provided in the data folder, and you should take a few variables or controls from this.

To successfully complete this task, you should do the following:

1. Conduct some descriptive analyses of these data. You can examine this using scatter plots and correlation coefficients. Do you need to transform any of the variables?
2. Fit regressions to your data. What is the appropriate type of regression and why have you chosen it?
3. Create at least one map that helps explain your findings (we will show you how to do this in the next session).
4. Write up your key results.

Each step in your Markdown file should be clearly explained and labelled.

Submit your work

Once you are finished, submit the RMD containing your work using the *Group project 2* assessment on canvas, along with your data in a zipped folder. Only one version needs to be submitted. Full grades will be allocated to all registered members of the group.

All the files needed to run your code should be uploaded in a zipped folder. We should be able to run the code without changing it.

You will be marked on the quality of your R code (including whether it runs for us without errors), how well you have justified your variable selection, the proper use of appropriate methods.

If you need help

If you have any questions, do not hesitate to ask us for help. We cannot do the work for you — this is an assessment – but we can provide some advice.

Good luck with the exercise!

Loading and examining your data

On the canvas page for this session you will find a zipped folder to download, which contains several sub-folders of data files. These files contain US county-level data on a number of issues. The files we are interested in at this stage are county-level poverty rates and deaths due to alcohol and drug use.

First let us load the poverty dataset. This was obtained from **the United States Census Bureau**.

We want the file *poverty data - ACS_16_5YR_S1702_with_ann.csv* in the folder *poverty*. We can upload this using the `read.csv()` function with the code:

```
poverty.data <- read.csv(
  "Data/US data for group task/poverty/poverty data - ACS_16_5YR_S1702_with_ann.csv",
  skip = 1, header = T)
```

With this syntax we have specified that R should skip the first row of the file with "skip = 1", as there are two header rows in this dataset (the first a code, which we discard, and the second a more descriptive header, which we use). We have also told R to use the header, which is row 2 now that we have removed row 1, using "header = T".

Of course, your code will vary slightly depending on the file path you need to use to access the data.

This dataset contains 603 columns (or variables) and 3142 rows, which you can observe yourself with the `dim()` function.

We then want to load the file *Drugs and alcohol Cause of Death, 1999-2016.csv* from the *mortality* folder. These data were obtained from **the Centers for Disease Control (CDC)** As this is a .txt file, we use the `read.table()` function to do this:

```
drugs.mort.data <- read.table(
  "Data/US data for group task/mortality/Drugs and alcohol Cause of Death, 1999-2016.txt",
    sep = '\t', header = T)
```

With this function, we specify that columns are separated with a tab using "sep = '\t'" and using "header = T" that the first line of each column (the header) designates the name of each variable in our data.

This dataset contains 9 variables and 2934 rows, which you can observe yourself with the `dim()` function. Both datasets contain the same number of rows, which in both cases are the number of countries in the United States.

We want to join these datasets, or elements of them. However, the variables we would use to do this, the code used to identify each county, is currently labelled differently in each data frame:

```
names(poverty.data)[3]
```

```
## [1] "Geography"
```

```
names(drugs.mort.data)[2]
```

```
## [1] "County"
```

When two datasets have a variable named and structured identically, it is simple to combine them. However, in this instance, the names of our ID variables don't match, even though their structure does:

```
head(poverty.data$Id2)
```

```
## [1] 1001 1003 1005 1007 1009 1011
```

```
head(drugs.mort.data$County.Code)
```

```
## [1] 1001 1003 1005 1007 1009 1011
```

Additionally, we need to recode at least one other variable name, and since the `poverty.data` file contains so many variables, we do not want to merge everything. Rather, we want to extract and merge certain variables. We could do this in multiple steps. However, it is faster and neater to do this with a single block of code. The following syntax can be used to rename the ID variable in one of our datasets, and then extract those variables we want from each file, before merging these data into a single new data frame we call `country.data`:

```
library(tidyverse)

patterns <- c(' County| Parish| city| Borough| Census Area')

county.data <- merge(
  poverty.data %>%
    dplyr::rename(below.poverty.rate =
             All.families....Percent.below.poverty.level..Estimate..Families)  %>%
    dplyr::select(Id2, below.poverty.rate),
  drugs.mort.data %>%
    dplyr::rename(Id2 = County.Code,
                  mortality.rate = Crude.Rate) %>%
    mutate(county = gsub(patterns, "", County))  %>%
    dplyr::select(Id2, county, mortality.rate))
```

These commands are derived from the `dplyr` package. Within the `merge()` function we use piping to include nested commands to rename and recode variable names using the `rename()` and `mutate()` functions respectively, and to extract only those variables we wish to include in the new merged frame with `select()`. As we discussed previously, the pipe operator allows for a block of code consisting of a number of functions to be chained together – the output of one function inserted it into the next – from left to right. We also remove the words 'County', 'Parish' and 'city' from county names, as these are not replicated in the shapefiles we will use later to map these data.

We now have a new dataset with 4 variables and 2928 rows. Viewing the first six rows of our new data frame, we can see the structure of these data:

```
##    Id2 below.poverty.rate      county mortality.rate
## 1 1001                9.4 Autauga, AL           12.5
## 2 1003                9.3 Baldwin, AL           22.6
## 3 1005               20.0 Barbour, AL            9.0
## 4 1007               11.7    Bibb, AL           14.1
## 5 1009               12.2  Blount, AL           18.1
## 6 1011               25.3 Bullock, AL           11.1
```

The first row is the ID variable we used to merge our data frames. The second, the percentage of each county's families with incomes below the poverty rate. Third, the names of each country (and the state within which they sit). Finally, the mortality rate associated with drug and alcohol use (per 100,000 people).

There is one last edit to make to these data before we examine them. If you run the code

`levels(as.factor(county.data$mortality.rate))`

you will see that some of the observations for the variable `mortality.rate` are coded as 'Unreliable'. This creates two issues for us. Those observations are effectively missing for our purposes, and this converts the variable to a string when we want it to be a number.

```
county.data <- county.data %>%
  mutate(mortality.rate = ifelse(mortality.rate == 'Unreliable',
                                 NA, as.numeric(as.character(mortality.rate))))
```

Within the `mutate()` function we use a combination of a conditional statement, which recode those values of `mortality.rate` that equal "Unreliable" as missing while retaining the value of all other observations. It then converts the variable into a numeric vector.

We are now ready to study the association between these two variables at the county level.

Load opiate prescription data

In addition to the poverty data, we also have information on the number of opiate prescriptions issued in a given county per 100,000 population. This can be found in the *opioids* folder, saved as the file *opioids prescribing rates*. We can upload this using the `read.csv()` function with the code:

```
opioids.data <- read.csv("Data/US data for group task/opioids/opioids prescribing rates.csv")
```

We then use the same steps as above to merge this data with our `county.data` frame:

```
library(tidyverse)

county.data <- merge(
  county.data  %>%
  mutate(z.poverty = (below.poverty.rate - mean(below.poverty.rate, na.rm=T)) /
           sd(below.poverty.rate, na.rm=T)),
  opioids.data %>%
  dplyr::rename(Id2 = FIPS.County.Code) %>%
  mutate(z.opiates = (X2016.Prescribing.Rate - mean(X2016.Prescribing.Rate, na.rm=T)) /
           sd(X2016.Prescribing.Rate, na.rm=T)) %>%
    dplyr::select(Id2, z.opiates, X2016.Prescribing.Rate))
```

In this code, we also standardise the poverty prescription rates in the same line of block of code that we use to merge the data.

Loading is US county-level shapefiles

When it comes time to map these data, you will want to link your dataframe with county-level shapefiles. These provide the information for the shapes on the map for the geographic units we are plotting data for. They do so using polygons, which we will overlay on background map tiles. In this instance, we are working with a shape file containing data on the boundaries of American counties, which matches our data.

To load the shape files call the `sf` package and then run the following code:

```
library(sf)

us.shape.1 <-
  st_read("Data/US data for group task/US shapefiles/cb_2016_us_county_20m.shp")
```

```
## Reading layer `cb_2016_us_county_20m' from data source
##   `/Users/shaunratcliff/Dropbox/Academia/Teaching/DATA 5207 - Data science in the social sciences/Se
##   using driver `ESRI Shapefile'
## Simple feature collection with 3220 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -179.1743 ymin: 17.91377 xmax: 179.7739 ymax: 71.35256
## Geodetic CRS:  NAD83
```

We then want to link these shapefiles to the county results. We start by adding the state name to the shapefile data. To do this we load a dataset saved in the US geospatial data folder on canvas called *fips concordance.csv*:

```
fip.concordance <- read.csv("Data/US data for group task/fips concordance.csv")
```

This file, which we were required to modify a little above, can be matched to the shapefile data using similar syntax to above:

```
us.shape.2 <- us.shape.1 %>%
  merge(fip.concordance %>%
          mutate(STATEFP =  ifelse(Numeric.code >= 0 & Numeric.code <= 9,
                                   paste0(0 , Numeric.code), Numeric.code)) %>%
          dplyr::rename(state = Alpha.code))
```

We edit the variable containing county names in shapefile using the `paste0()` function, which combines the county name with the state abbreviation (from the fip concordance file). We then modify county names with some different spelling in the shapefiles and our county level data:

```
us.shape.2$county <- paste0(us.shape.2$NAME, ", ", us.shape.2$state)

us.shape.2$county <- gsub("Doña Ana, NM", "Dona Ana, NM", us.shape.2$county)
us.shape.2$county <- gsub("LaSalle, LA", "La Salle, LA", us.shape.2$county)
us.shape.2$county <- gsub("Oglala Lakota, SD", "Oglala, SD", us.shape.2$county)
```

And then we merge the shapefile and the county-level data:

```
us.shape.2 <- full_join(us.shape.2,
  county.data %>%
    dplyr::select(county, below.poverty.rate, mortality.rate, X2016.Prescribing.Rate))
```

```
## Joining with `by = join_by(county)`
```

```
## Warning in sf_column %in% names(g): Detected an unexpected many-to-many relationship between `x` and
## i Row 1205 of `x` matches multiple rows in `y`.
## i Row 1138 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

You are now ready to map these data. Good luck with the rest of the exercise!

Other data

Several other files are included in the data folder. These include a large number county-level predictors in the file *county.data.RData* in the folder *Other census data*. The codebook for these data can be found here. Additional information on these data are available here.