ETC1010: Data Modelling and Computing

Lecture 5: Reading different data formats

Di Cook (dicook@monash.edu, @visnut)

Week 5

Overview

- Shape files for maps
- Excel spreadsheets
- Googlesheets
- SPSS format (PISA data)
- Audio files
- `read_csv` vs `read.csv`
- feather for large binary files
- Handling large data sets by constructing a small database: `sqlite`
- Web format (json) data, `jsonlite` (crossrates)
- Web scraping?

Shape files

Download the Australian electorate shape files from [http://www.aec.gov.au/Electorates/gis/gis_datadownload.htm], 2016 national, mapinfo format. Its 11Mb.

```
OGR data source with driver: MapInfo File
Source: "data/national-midmif-09052016/COM_ELB.TAB", layer: "COM_ELB"
with 150 features
It has 9 fields
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame':    150 obs. of  9 variables:
  .. ..$ Elect_div          : Factor w/ 150 levels "Adelaide","Aston",..: 90 135 7 17
  .. ..$ State              : Factor w/ 8 levels "ACT","NSW","NT",..: 3 3 6 6 6 6 6 4
  .. ..$ Numccds            : int [1:150] 335 180 208 226 197 179 256 233 216 199 ...
  .. ..$ Actual             : int [1:150] 0 0 0 0 0 0 0 0 0 0 ...
  .. ..$ Projected          : int [1:150] 0 0 0 0 0 0 0 0 0 0 ...
  .. ..$ Total_Population   : int [1:150] 0 0 0 0 0 0 0 0 0 0 ...
  .. ..$ Australians_Over_18: int [1:150] 0 0 0 0 0 0 0 0 0 0 ...
  .. ..$ Area_SqKm          : num [1:150] 1352034 337 7379 20826 289 ...
  .. ..$ Sortname           : Factor w/ 150 levels "Adelaide","Aston",..: 90 135 7 17
  ..@ polygons   :List of 150
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
```

Your turn

How many Federal electorates in Australia?

Thinning out space

- The shape object created is 46Mb. Too big!
- Want smaller data set, that still effectively describes the spatial domain.
- Thinning a map object can be tricky, want to thin long straight areas but keep twisty boundaries detailed.

```
library(rmapshaper)
sFsmall <- ms_simplify(sF, keep=0.05)
```

Plot it

Map is still defined in shapefile structure. Difficult to work with, in conjunction with any other data.

Extract information on each electorate

```
nat_data <- sF@data
nat_data$id <- row.names(nat_data)
head(nat_data)
  Elect_div State Numccds Actual Projected Total_Population
1  Lingiari    NT     335      0         0                0
2   Solomon    NT     180      0         0                0
3      Bass   TAS     208      0         0                0
4   Braddon   TAS     226      0         0                0
5   Denison   TAS     197      0         0                0
6  Franklin   TAS     179      0         0                0
  Australians_Over_18    Area_SqKm Sortname id
1                   0 1352034.0451 Lingiari  1
2                   0     336.6861  Solomon  2
3                   0    7378.7516     Bass  3
4                   0   20826.1840  Braddon  4
5                   0     288.7177  Denison  5
6                   0    6514.2083 Franklin  6
```

Get map into tidy form

```
nat_map <- ggplot2::fortify(sFsmall)
head(nat_map)
      long       lat order  hole piece id group
1 137.9982 -23.52089     1 FALSE     1  1   1.1
2 137.9984 -23.58319     2 FALSE     1  1   1.1
3 137.9984 -23.66652     3 FALSE     1  1   1.1
4 137.9985 -23.74985     4 FALSE     1  1   1.1
5 137.9985 -23.83318     5 FALSE     1  1   1.1
6 137.9985 -23.91651     6 FALSE     1  1   1.1
```

Be clear about id variables

- Ensure group and piece variables are treated as factors, not numbers
- Add electorate names to the polygons

```
nat_map$group <- paste("g",nat_map$group,sep=".")
nat_map$piece <- paste("p",nat_map$piece,sep=".")
nms <- sFsmall@data %>% select(Elect_div, State)
nms$id <- as.character(1:150)
nat_map <- left_join(nat_map, nms, by="id")
head(nat_map)
      long       lat order  hole piece id group Elect_div State
1 137.9982 -23.52089     1 FALSE   p.1  1 g.1.1  Lingiari    NT
2 137.9984 -23.58319     2 FALSE   p.1  1 g.1.1  Lingiari    NT
3 137.9984 -23.66652     3 FALSE   p.1  1 g.1.1  Lingiari    NT
4 137.9985 -23.74985     4 FALSE   p.1  1 g.1.1  Lingiari    NT
5 137.9985 -23.83318     5 FALSE   p.1  1 g.1.1  Lingiari    NT
6 137.9985 -23.91651     6 FALSE   p.1  1 g.1.1  Lingiari    NT
```

Map it, using area of the electorate to colour. With joins to data from other sources, e.g. census, other variables could be mapped to colour.

```
ggplot(aes(map_id=id), data=nat_data) +
  geom_map(aes(fill=Area_SqKm), map=nat_map) +
  expand_limits(x=nat_map$long, y=nat_map$lat) +
  theme_map()
```

Interactivity

Mouseover names more effective

```
p <- ggplot(aes(map_id=id), data=nat_data) +
  geom_map(aes(fill=Area_SqKm, label=Elect_div), map=nat_map) +
  expand_limits(x=nat_map$long, y=nat_map$lat) +
  theme_map()
ggplotly(p)
```

Add centroids

Using the geographic centroid for each electorate is an alternative. It can also be extracted from the shape files.

```
centroid <- function(i, polys) {
  ctr <- Polygon(polys[i])@labpt
  data.frame(long_c=ctr[1], lat_c=ctr[2])
}
centroids <- seq_along(polys) %>% purrr::map_df(centroid, polys=polys)
head(centroids)
    long_c     lat_c
1 133.3706 -19.48052
2 130.9355 -12.42392
3 147.5081 -41.15828
4 145.4985 -41.75995
5 147.2439 -42.88836
6 146.6272 -43.24309
```

joined to the other information about each electorate...

```
nat_data <- bind_cols(nat_data, centroids)
head(nat_data)
  Elect_div State Numccds Actual Projected Total_Population
1  Lingiari    NT     335      0         0                0
2   Solomon    NT     180      0         0                0
3      Bass   TAS     208      0         0                0
4   Braddon   TAS     226      0         0                0
5   Denison   TAS     197      0         0                0
6  Franklin   TAS     179      0         0                0
  Australians_Over_18    Area_SqKm Sortname id   long_c     lat_c
1                   0 1352034.0451 Lingiari  1 133.3706 -19.48052
2                   0      336.6861  Solomon  2 130.9355 -12.42392
3                   0    7378.7516     Bass  3 147.5081 -41.15828
4                   0   20826.1840  Braddon  4 145.4985 -41.75995
5                   0      288.7177  Denison  5 147.2439 -42.88836
6                   0    6514.2083 Franklin  6 146.6272 -43.24309
```

and plotted as is, or spread out

```
p1 <- ggplot(aes(map_id=id), data=nat_data) +
  geom_map(aes(fill=Area_SqKm), map=nat_map) +
  expand_limits(x=nat_map$long, y=nat_map$lat) +
  theme_map() + theme(legend.position="none") +
  geom_point(data=nat_data, aes(x=long_c, y=lat_c), colour="orange")
p2 <- ggplot(aes(map_id=id), data=nat_data) +
  geom_map(aes(fill=Area_SqKm), map=nat_map) +
  expand_limits(x=nat_map$long, y=nat_map$lat) +
  theme_map() + theme(legend.position="none") +
  geom_jitter(data=nat_data, aes(x=long_c, y=lat_c),
              colour="orange", width=2, height=2)
grid.arrange(p1, p2, ncol=2)
```

Excel spreadsheets

- Often data comes in multiple excel format files
- It it tedious, and inefficient to manually convert each to csv and read
- Easier to automate reading multiple files, in the original format
- Example: Rental market in Tasmania from data.gov.au

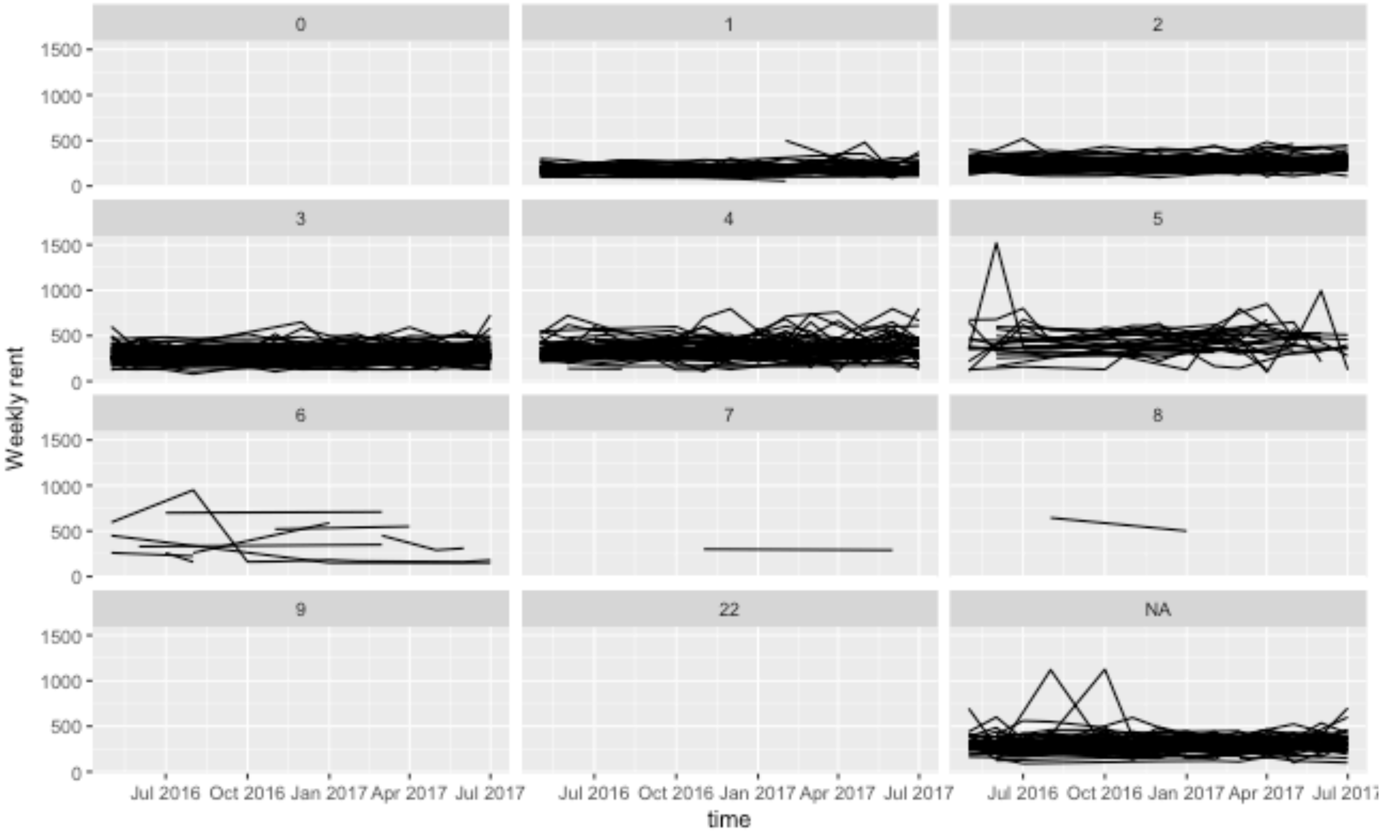
```
library(readxl)
library(sawfish) # devtools::install_github("AnthonyEbert/sawfish")
url<-"http://data.gov.au/dataset/rental-bond-and-rental-data-tasmania-2016-to
fls <- find_files(url, "xlsx")
f1 <- tempfile()
download.file(fls[1], f1, mode="wb")
t1 <- read_xlsx(path=f1, sheet=1)
t1
# A tibble: 1,368 x 11
     `Street Name`      Suburb State Postcode `Bond Amount` `Weekly Rent`
             <chr>       <chr> <chr>    <dbl>         <dbl>         <dbl>
 1 BANGALEE STREET  LAUDERDALE   TAS     7021          1440         360.0
 2     WILMOT ROAD   HUONVILLE   TAS     7109          1180         295.0
 3     FOREST ROAD WEST HOBART   TAS     7000           350          87.5
 4     FOREST ROAD WEST HOBART   TAS     7000           350          87.5
 5     CENTRAL AVE      MOONAH   TAS     7009           800         200.0
 6      CHARLES ST      MOONAH   TAS     7009          1200         300.0
 7   SPINIFEX ROAD RISDON VALE   TAS     7016          1080         270.0
 8   GARDENIA ROAD RISDON VALE   TAS     7016           780         195.0
 9  BRITTEN STREET NEW NORFOLK   TAS     7140          1140         285.0
10      BOXHILL RD   CLAREMONT   TAS     7011           600         150.0
# ... with 1,358 more rows, and 5 more variables: `Bond Lodgement date
#   (DD/MM/YYYY)` <dttm>, `Bond Activation date (DD/MM/YYYY)` <dttm>, `No
#   of Bedrooms` <dbl>, `Dwelling/Premises Type` <chr>, `Length of Tenancy
#   (In Months)` <dbl>
```

Now pull all and merge

```
rentals <- NULL
for (i in 1:length(fls)) {
  download.file(fls[i], f1, mode="wb")
  t1 <- read_xlsx(path=f1, sheet=1)
  rentals <- bind_rows(rentals, t1)
}
dim(rentals)
[1] 18263    12
```

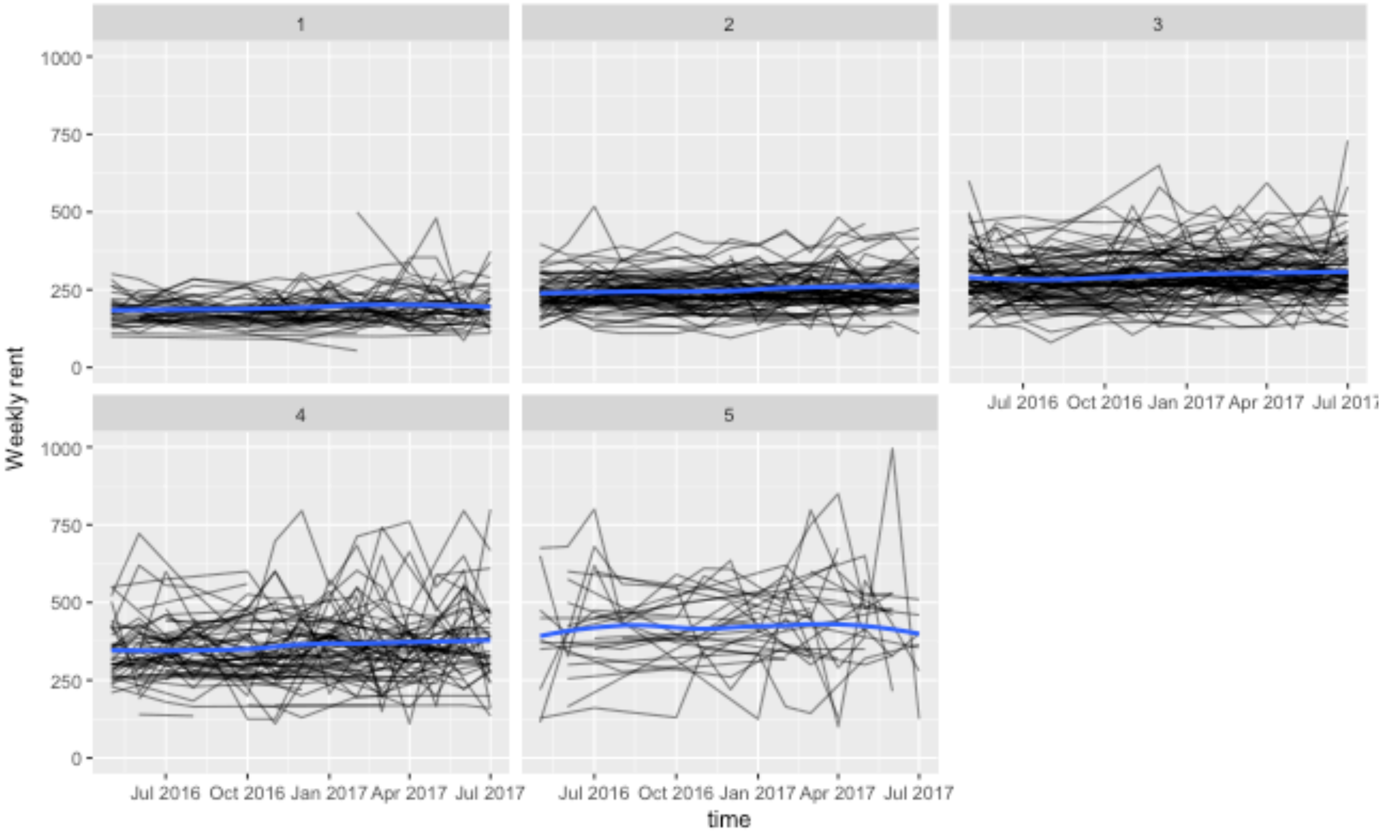
How have rental rates changed over time

```
rentals %>%
  mutate(month=month(`Bond Lodgement date (DD/MM/YYYY)`),
         year=year(`Bond Lodgement date (DD/MM/YYYY)`)) %>%
  group_by(Postcode, month, year, `No of Bedrooms`) %>%
  summarise(rent=mean(`Weekly Rent`, na.rm=TRUE)) %>%
  mutate(time=dmy(paste("01", month, year, sep="-"))) %>%
  ggplot(aes(x=time, y=rent)) +
    geom_line(aes(group=Postcode)) +
    facet_wrap(~`No of Bedrooms`, ncol = 3) +
    ylab("Weekly rent")
```


0
1
2
3
4
5
6
7
8
9
22
NA
1500
1000
500
0
Weekly rent
Jul 2016
Oct 2016
Jan 2017
Apr 2017
Jul 2017
time

Clean data and re-plot

```
rentals %>%
  mutate(month=month(`Bond Lodgement date (DD/MM/YYYY)`),
         year=year(`Bond Lodgement date (DD/MM/YYYY)`)) %>%
  group_by(Postcode, month, year, `No of Bedrooms`) %>%
  summarise(rent=mean(`Weekly Rent`, na.rm=TRUE)) %>%
  mutate(time=dmy(paste("01", month, year, sep="-"))) %>%
  filter(!is.na(`No of Bedrooms`)) %>%
  filter(`No of Bedrooms`<6, `No of Bedrooms`>0) %>%
  ggplot(aes(x=time, y=rent)) +
    geom_line(aes(group=Postcode), alpha=0.5) +
    facet_wrap(~`No of Bedrooms`, ncol = 3) +
    ylab("Weekly rent") + ylim(c(0, 1000)) +
    geom_smooth(se=FALSE)
```


1
2
3
4
5
Weekly rent
1000
750
500
250
0
time
Jul 2016
Oct 2016
Jan 2017
Apr 2017
Jul 2017

Googlesheets

- Google sheets are effectively excel spreadsheets
- We can read these directly also
- More efficient than download and read in

```
library(readxl) # Read from moodle excel sheet
class <- read_xlsx("ETC1010 - S2 2017 Grades.xlsx")

library(googlesheets) # Now get lab scores
gs_ls()
Waiting for authentication in browser...
Press Esc/Ctrl + C to abort
lss <- gs_title("ETC1010")
labs <- gs_read(lss, col_types=c("ccccddc"))

lab_scores <- full_join(class, labs[,c(1,5,6,7)],
                        by=c("ID number"="ID number")) %>%
  filter(!is.na(Surname)) %>%
  rename("Lab 1"="Lab 1(Out of 10)") %>%
  replace_na(list(`Lab 1`=0))
```

Share and share alike