**French mortality data**

We will use the French mortality data from the demography package, but we need to convert it into a tibble to begin.

library(tidyverse)

library(demography)

# Combine age groups above 100

frmort <- set.upperage(fr.mort, 100)

# Create tibble

frmort <- tibble(

year = rep(frmort$year, rep(length(frmort$age), length(frmort$year))),

age = rep(frmort$age, length(frmort$year)),

female = c(frmort$rate$female),

male = c(frmort$rate$male),

) %>%

gather(male, female, key = "sex", value = "mortrate")

frmort

## # A tibble: 38,582 x 4

## year age sex mortrate

##

## 1 1816 0 male 0.223

## 2 1816 1 male 0.0467

## 3 1816 2 male 0.0343

## 4 1816 3 male 0.0232

## 5 1816 4 male 0.0161

## 6 1816 5 male 0.0136

## 7 1816 6 male 0.0116

## 8 1816 7 male 0.00991

## 9 1816 8 male 0.00838

## 10 1816 9 male 0.00710

## # ... with 38,572 more rows

The first thing to do is to re-create the rainbow plots that are popular for this type of data (introduced in [my paper with Hanlin in JCGS in 2010](https://robjhyndman.com/publications/rainbow-fda/)). Here the year is mapped to colour. This works quite well for mortality data because it has trended consistently over time, allowing the colors to separate. It is one of the few situations where a rainbow palette is preferred to other palettes.

frmort %>%

ggplot(aes(x = age, y = mortrate, group = year, col = year)) +

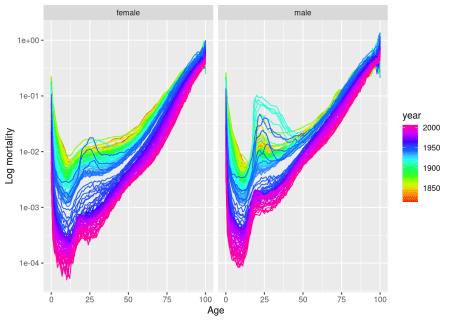
geom\_line() +

facet\_grid(~sex) +

scale\_y\_log10() +

xlab("Age") + ylab("Log mortality") +

scale\_color\_gradientn(colours = rainbow(10))



Another plot that has proved popular is to animate this rainbow plot by mapping year to animation time. With the new [gganimate](https://github.com/thomasp85/gganimate) package (still only on github), that is as easy as adding a few more lines to the end of the above code.

# This requires the transformr package on CRAN and the gganimate package on github

# Run the following two lines if you don't already have them.

#install.packages(c("transformr", "devtools"))

#devtools::install\_github("thomasp85/gganimate")

library(gganimate)

frmort %>%

filter(year > 1900) %>%

ggplot(aes(x = age, y = mortrate, group = year, col = year)) +

geom\_line() +

xlab("Age") + ylab("Log mortality") +

facet\_grid(~sex) +

scale\_y\_log10() +

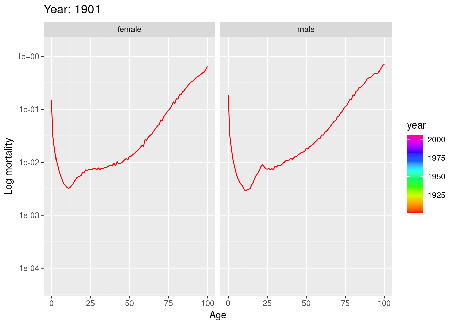
scale\_color\_gradientn(colours = rainbow(10)) +

transition\_time(year) +

ease\_aes('linear') +

shadow\_mark(colour = "grey70") +

labs(title = 'Year: {frame\_time}')



Another way of looking at the data is using an image map. Again, this is extremly easy using ggplot2.

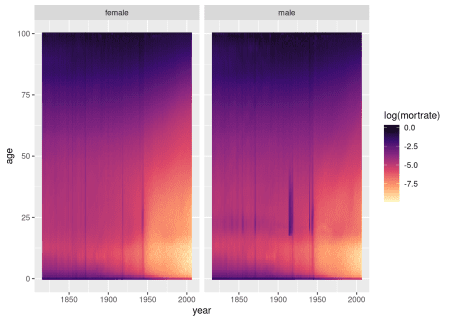
frmort %>%

ggplot(aes(x = year, y = age, fill = log(mortrate))) +

geom\_raster() +

facet\_grid(~sex) +

scale\_fill\_viridis\_c(option = "A", direction = -1)



Note the various wars and epidemics (seen as vertical lines), and the decrease in mortality rates over time (seen as the growing light-coloured area).

Since this is time series data, we should also look at the autocorrelation function. Because the data are functions of age, the autocorrelation is a surface for each lag value. The function facf below computes a functional ACF surface (giving correlations between different ages and across lagged years). There is some tricky non-standard evaluation used here to allow for non-quoted variables to be used when the function is called.

facf <- function(df, xvar, yvar, time, lag.max=20) {

key <- enquo(xvar)

value <- enquo(yvar)

timeindex <- enquo(time)

x <- df %>%

select(!!key, !!value, !!timeindex) %>%

spread(value=!!value, key=!!key) %>%

select(-!!timeindex) %>%

as.ts() %>%

acf(plot=FALSE, lag.max=lag.max, na.action=na.pass)

nx <- dim(x$acf)[2]

output <- NULL

for(i in seq(lag.max+1)) {

output <- bind\_rows(output,

tibble(

lag = i-1,

x1 = rep(rep(0:(nx-1), nx)),

x2 = rep(0:(nx-1), rep(nx,nx)),

acf = c(x$acf[i,,])

))

}

colnames(output)[2:3] <- paste0(as.character(key)[[2]],1:2)

return(output)

}

# Compute FACF for the French mortality data

fracf <- frmort %>%

nest(-sex) %>%

mutate(

acf = map(data, ~ facf(df=., xvar=age, yvar=mortrate, time=year))

) %>%

select(-data) %>%

unnest()

fracf %>%

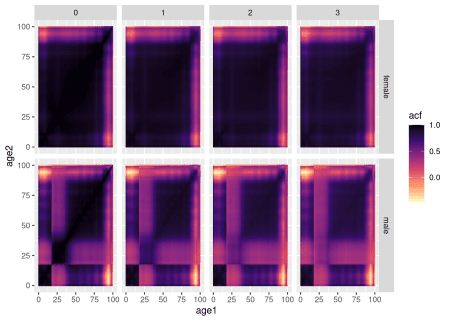
filter(lag < 4) %>%

ggplot(aes(x = age1, y = age2, fill = acf)) +

geom\_raster() +

facet\_grid(sex~lag) +

scale\_fill\_viridis\_c(option = "A", direction = -1)



Here there is a striking difference between males and females, with relatively low correlations between mortality rates of males aged 18-35 and males of other ages. This is largely driven by the wars where males of those ages die at much greater rates than other males, but only for a few years. If we start the analysis from 1950, the effect is much reduced.

fracf <- frmort %>%

filter(year > 1950) %>%

nest(-sex) %>%

mutate(

acf = map(data, ~ facf(df=., xvar=age, yvar=mortrate, time=year))

) %>%

select(-data) %>%

unnest()

fracf %>%

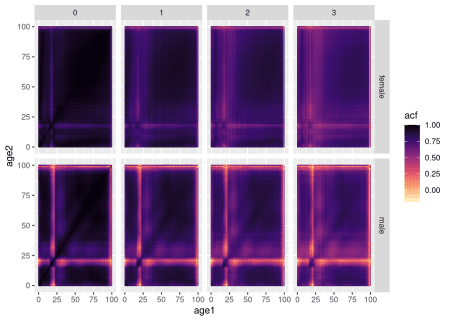
filter(lag < 4) %>%

ggplot(aes(x = age1, y = age2, fill = acf)) +

geom\_raster() +

facet\_grid(sex~lag) +

scale\_fill\_viridis\_c(option = "A", direction = -1)



There is still a section of low correlation around ages 18-22, with the correlations being lower for males than females. I suspect this is to do with the well-known accident bump, where young people tend to have higher mortality due to accidents and suicides than people of other ages.

Jim Ramsay pointed out in my talk that it would be nice to remove the redundancy due to symmetry and show the males in the top left triangles, with the females below. It turns that this is also very easy to do.

fracf %>%

filter(

lag < 4,

(sex=="male" & age2 > age1) | (sex=="female" & age2 < age1)

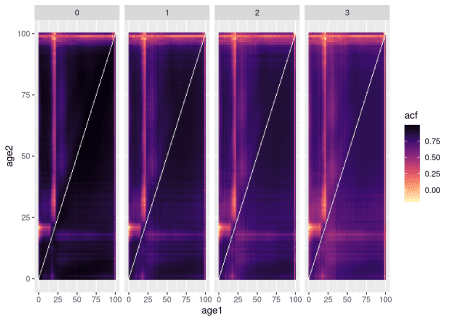
) %>%

ggplot(aes(x = age1, y = age2, fill = acf)) +

geom\_raster() +

facet\_grid(~lag) +

scale\_fill\_viridis\_c(option = "A", direction = -1)



Finally, the diagonals where age1=age2 are of particular interest, as these correspond to the ACFs of the univariate time series comprising each age group.

I will plot them in three different ways – against age, against lag, and as a 2-d image plot.

fracf %>%

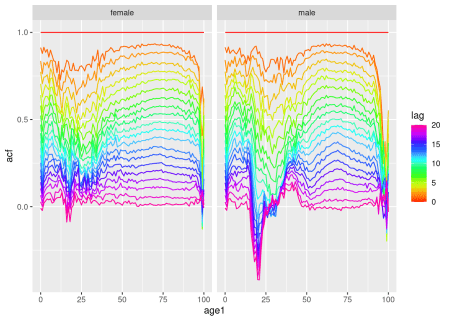
filter(age1==age2) %>%

ggplot(aes(x = age1, y = acf, group = lag, col = lag)) +

facet\_grid(~sex) +

geom\_line() +

scale\_color\_gradientn(colours = rainbow(10))



fracf %>%

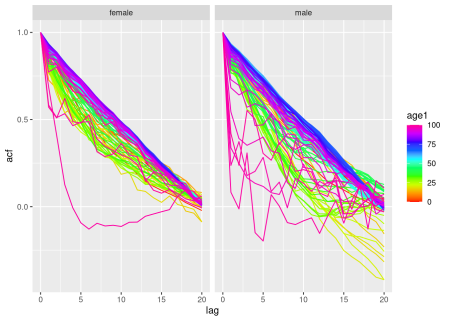
filter(age1==age2) %>%

ggplot(aes(x = lag, y = acf, group = age1, col = age1)) +

geom\_line() +

facet\_grid(~sex) +

scale\_color\_gradientn(colours = rainbow(10))



fracf %>%

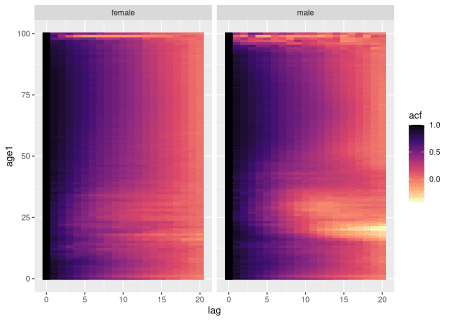
filter(age1==age2) %>%

ggplot(aes(x = lag, y = age1, fill = acf)) +

geom\_raster() +

facet\_grid(~sex) +

scale\_fill\_viridis\_c(option = "A", direction = -1)



**Melbourne pedestrian data**

My second example involves pedestrian traffic near Flinders St Station in Melbourne city. The data can be downloaded using the [rwalkr](https://cran.r-project.org/package=rwalkr) package, but some data is pre-packaged in the [sugrrants](https://cran.r-project.org/package=sugrrants) package, which we will use here.

Again, the first task is to put the data into a suitable form. We will use only data from Flinders St Station Underpass in 2016, and add in holiday information to the data set.

library(sugrrants)

pedestrian <- pedestrian %>%

filter(

Sensor\_Name == "Flinders Street Station Underpass",

Date <= as.Date("2016-12-31"),

) %>%

rename\_all(tolower) %>%

rename(

hour = "time",

number = "hourly\_counts"

) %>%

left\_join(tsibble::holiday\_aus(2016, state = "VIC")) %>%

mutate(

daytype = ifelse(

day %in% c("Saturday", "Sunday") | !is.na(holiday),

"Holiday", "Workday"

)

) %>%

select(date, hour, day, daytype, month, number)

pedestrian

## # A tibble: 8,783 x 6

## date hour day daytype month number

##

## 1 2016-01-01 0 Friday Holiday January 3643

## 2 2016-01-01 1 Friday Holiday January 2009

## 3 2016-01-01 2 Friday Holiday January 3238

## 4 2016-01-01 3 Friday Holiday January 2164

## 5 2016-01-01 4 Friday Holiday January 1161

## 6 2016-01-01 5 Friday Holiday January 682

## 7 2016-01-01 6 Friday Holiday January 388

## 8 2016-01-01 7 Friday Holiday January 373

## 9 2016-01-01 8 Friday Holiday January 275

## 10 2016-01-01 9 Friday Holiday January 545

## # ... with 8,773 more rows

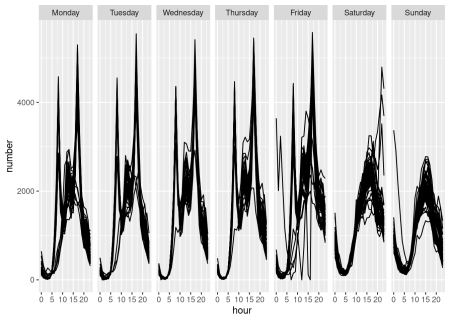
The differences between days is clearly seen. It is also apparent that there were a handful of very unusual days.

pedestrian %>%

ggplot(aes(x = hour, y = number, group = date)) +

geom\_line() +

facet\_grid(~day)



For sub-daily data, a calendar plot is extremely useful for identifying them, along with other interesting features in the data. The public holidays on weekdays are clearly marked here in a different colour. Can you spot deviations from the regular pattern that are not explained by holidays?

p <- pedestrian %>%

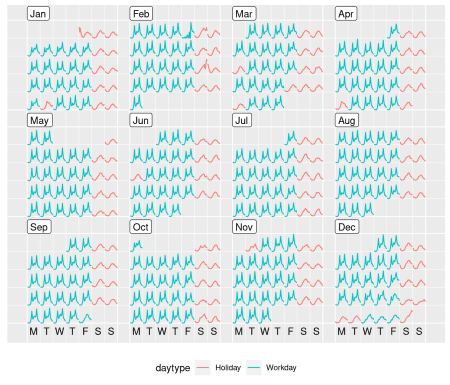
frame\_calendar(x = hour, y = number, date = date) %>%

ggplot(aes(x = .hour, y = .number, group = date, colour = daytype)) +

geom\_line() +

theme(legend.position = "bottom")

prettify(p)



For the ACF, I will look only at the “diagonal surface” — the equivalent of the univariate ACFs for each hour, plotted for different lags.

pedestrian %>%

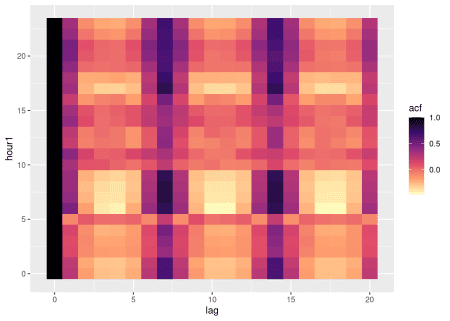
facf(xvar=hour, yvar=number, time=date, lag.max=20) %>%

filter(hour1==hour2) %>%

ggplot(aes(x = lag, y = hour1, fill = acf)) +

geom\_raster() +

scale\_fill\_viridis\_c(option = "A", direction = -1)



Here it is interesting to note that the weekly seasonality is strongest at hours 6-9am and around 4-5pm, corresponding to the peak hours for workers. There is relatively weak correlation between 10am and 3pm, when workers are mostly working.