Class 4: High dimensional visualisation

Andrew Parnell
andrew.parnell@mu.ie

PRESS RECORD
https://andrewcparnell.github.io/dataviz_course

Learning outcomes

- Learn how to create multiple scatter plots
- Plot some map data
- Plot some time series data
- Use some of the 3D geoms
- Our last session where we mostly use ggplot

New data set for this session

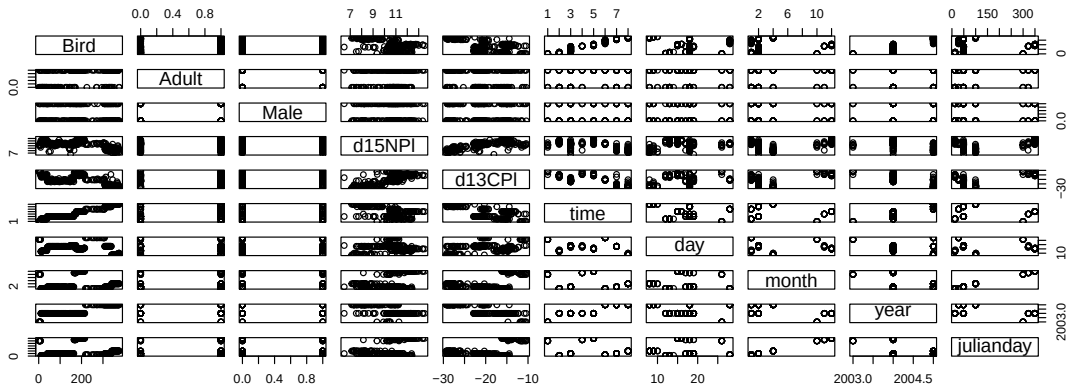
Geese isotopes: Two isotopes taken from the blood plasms of geese together with information about each of the birds

```
geese <- read.csv('../data/geese_isotopes.csv')
geese %>%  glimpse
```

```
## Rows: 248
## Columns: 10
## $ Bird      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
## $ Adult     <int> 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
## $ Male      <int> 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
## $ d15NPl    <dbl> 10.22, 10.37, 10.44, 10.52, 10.19, 10.45, 9.91, 11.27, 9.34,
## $ d13CPl    <dbl> -11.36, -11.88, -10.60, -11.25, -11.66, -10.41, -10.88, -14.
## $ time      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## $ day       <int> 26, 26, 26, 26, 26, 26, 26, 26, 26, 12, 12, 12, 12, 12, 12,
## $ month     <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 1, 1, 1, 1, 1, 1, 1, 1,
## $ year      <int> 2003, 2003, 2003, 2003, 2003, 2003, 2003, 2003, 2003, 2004,
## $ julianday <dbl> 299.75, 299.75, 299.75, 299.75, 299.75, 299.75, 299.75, 299.
```

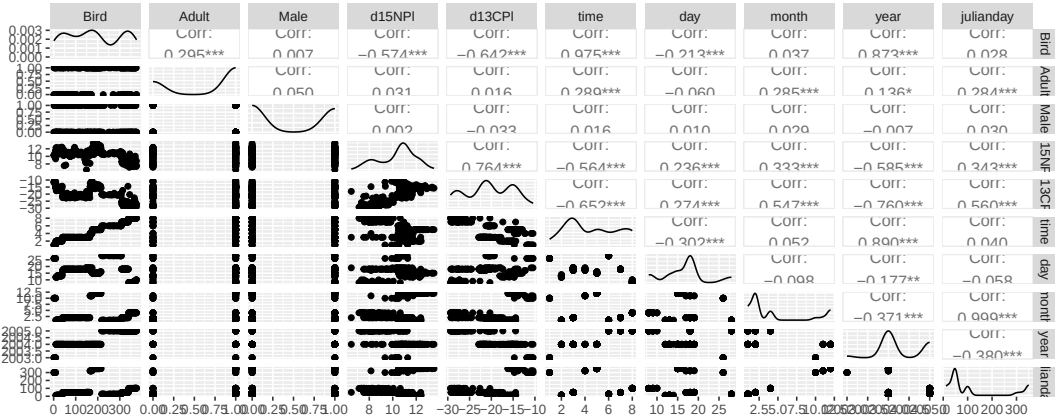
Multiple scatter plots: `pairs`

```
pairs(geese)
```



ggplot version using package GGally

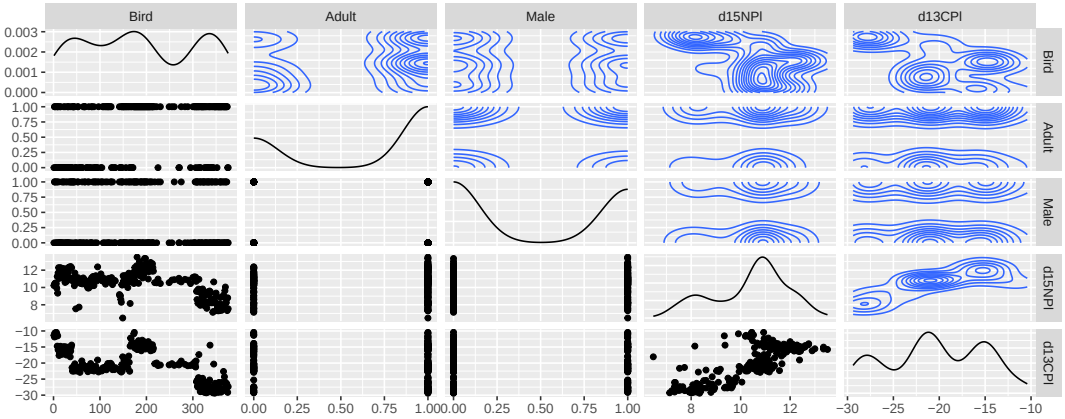
```
library(GGally)
ggpairs(geese)
```



Very customisable too - can change what the upper, lower and diagonal panels contain or add colours to different groups

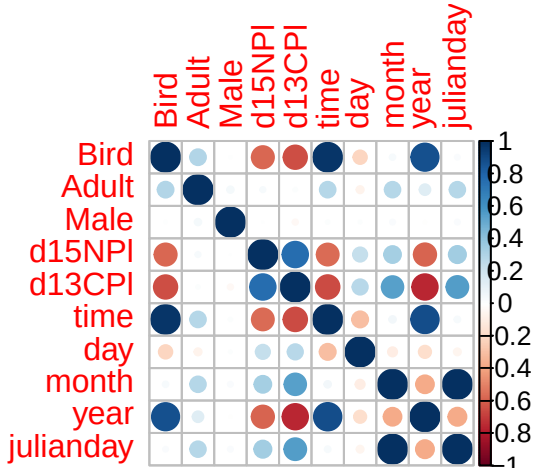
Extended ggpairs plot

```
ggpairs(geese[,1:5],
        upper = list(continuous = "density"),
        lower = list(continuous = "points"))
```



Visualise correlations

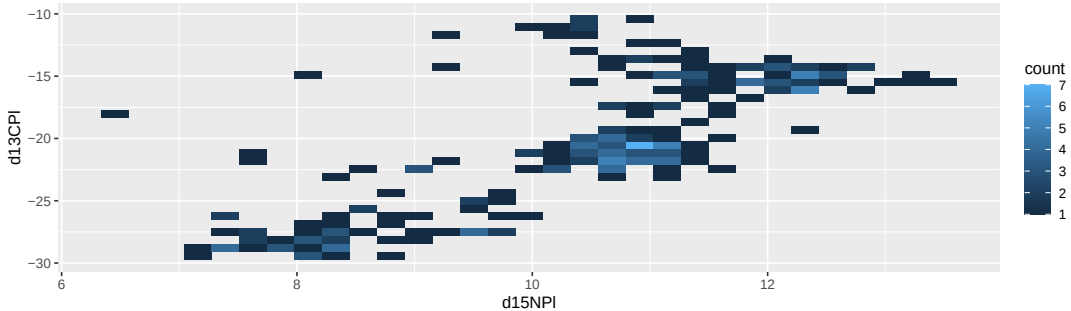
```
library(corrplot)
corrplot(cor(geese)) # Note: need a correlation matrix!
```



2D density plots

ggplot2 has specific functions for plotting 3D density data

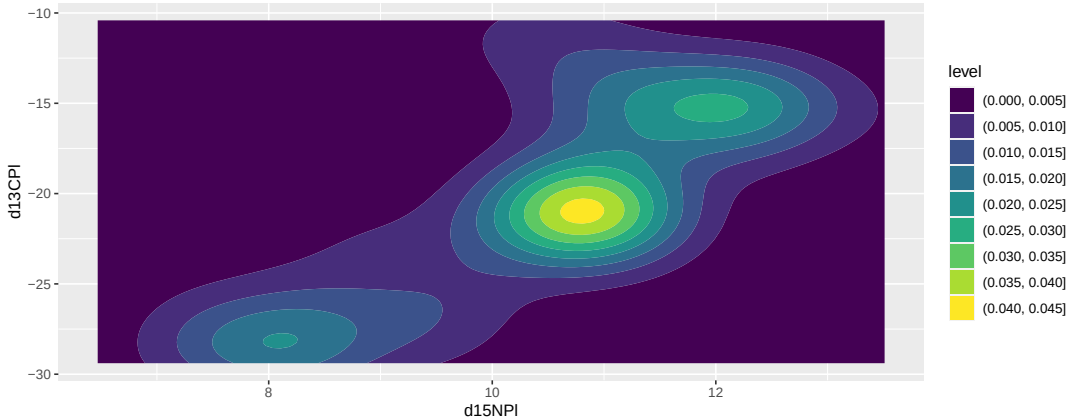
```
ggplot(geese, aes(x = d15NPl, y = d13CPl)) +
  geom_bin_2d()
```



- There are related functions `geom_tile` and `geom_raster` when you have a variable by which you want to colour the pixels.

Contour plots

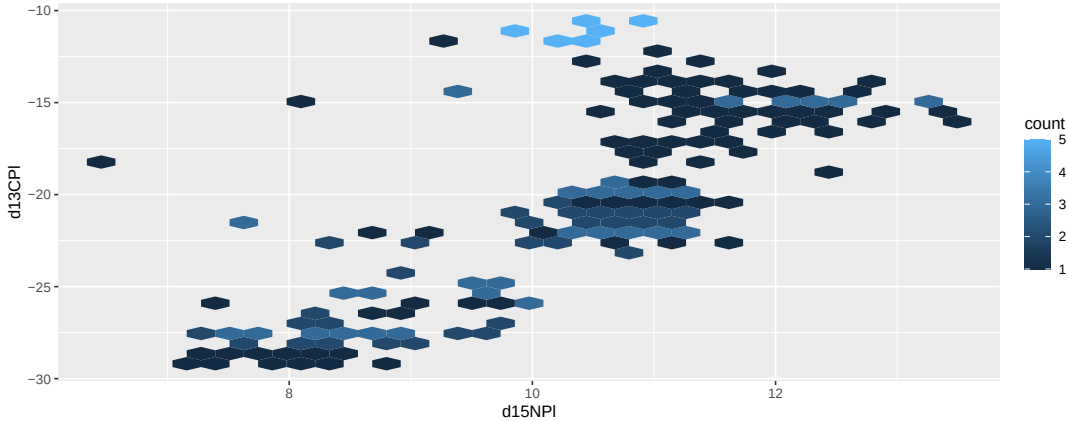
```
ggplot(geese, aes(x = d15NPl, y = d13CPl)) +
  geom_density_2d_filled()
```



- Again, use `geom_contour` if you have a variable to directly create the contour values

Hexagonal plots

```
ggplot(geese, aes(x = d15NPl, y = d13CPl)) +
  geom_hex()
```



Plotting data with maps

- Most base R plots of maps are ugly. `ggplot` has an interface to the `maps` package but these are quite ugly and out of date
- Better package are `ggmap`, `ggspatial` and `sf` (simple features) which have loads of examples and interfaces for both base R and ggplots
- Three common use cases:
  - You want to take a map of an area and add information to it by filling in the areas or placing points on the map
  - You want to take an e.g. Google or OpenStreetMap (raster) map and add information on top
  - You have your own shape file and want to create a plot on top of that (not covered – see `read_sf`)

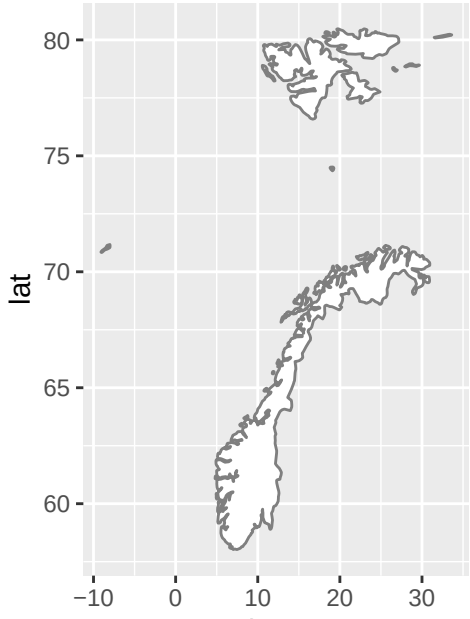
Scenario 1: adding information to a map of a given area

```
library(maps)
norway <- map_data("world", "norway")
norway %>% glimpse
```

```
## Rows: 1,985
## Columns: 6
## $ long      <dbl> 5.085840, 5.089063, 4.996973, 4.955566, 4.943555, 4.95
## $ lat       <dbl> 60.30756, 60.18877, 60.19776, 60.24331, 60.27241, 60.3
## $ group     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
## $ order     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17
## $ region    <chr> "Norway", "Norway", "Norway", "Norway", "Norway", "Nor
## $ subregion <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1"
```

```
ggplot(norway, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "grey50") +
  coord_quickmap()
```

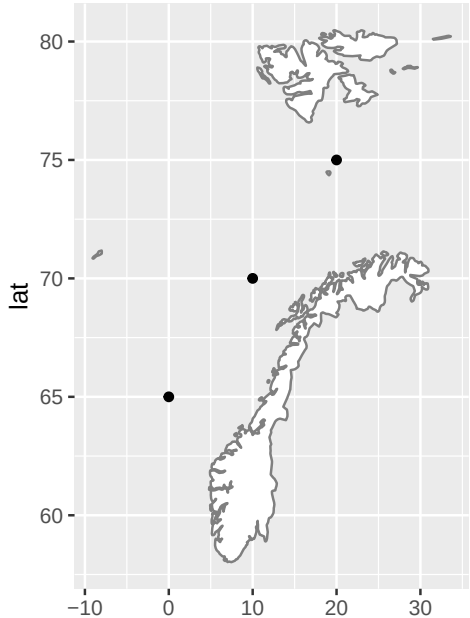
Scenario 1: plot



Adding information

```
points <- tibble(
  long = c(0, 10, 20),
  lat = c(65, 70, 75),
  group = 1 # Necessary to match the fitting command
)
ggplot(norway, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "grey50") +
  coord_quickmap() +
  geom_point(data = points) # Add colours etc here
```

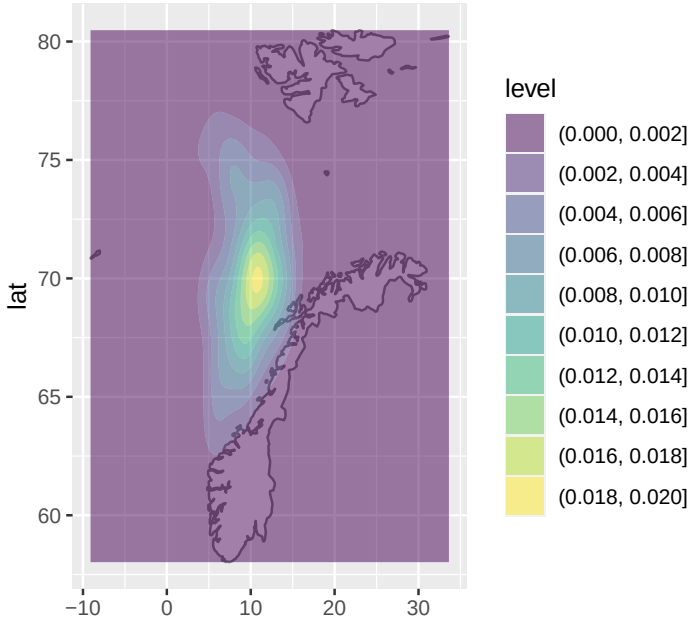
Plot with added information



Scenario 1 (cont): with e.g. contours

```
new_data <- MASS::mvrnorm(n = 100,
                          mu = c(10, 70),
                          Sigma = diag(10, 2)) %>%
  as_tibble() %>%
  rename(long = V1, lat = V2) %>%
  mutate(group = 1)
ggplot(norway, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "grey50") +
  coord_quickmap() +
  geom_density_2d_filled(data = new_data, alpha = 0.5)
```

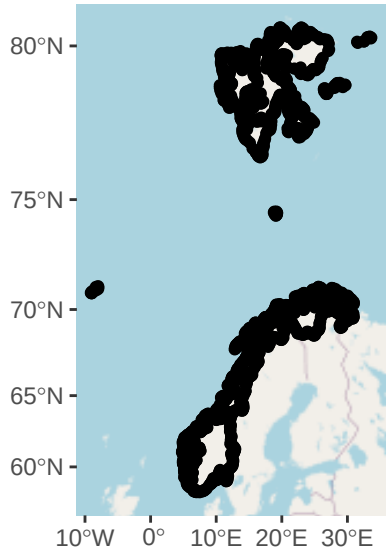
Scenario 1: plot



Scenario 2: overlaying an existing web map

```
library(ggspatial)
norway2 <- sf::st_as_sf(norway, coords = c("long", "lat"),
                        crs = 4326, agr = "constant")
ggplot() +
  annotation_map_tile(type = "osm") +
  layer_spatial(norway2)
```

Scenario 2: plot

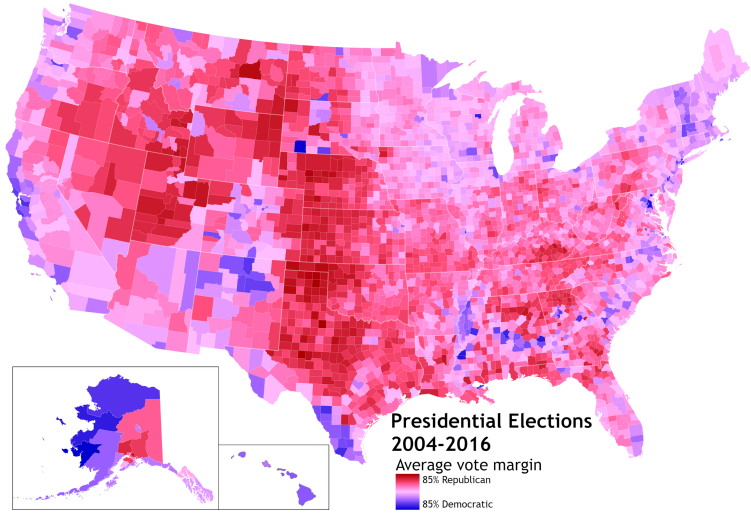


Leaflet

See demo in Rstudio:

```
library(leaflet)
leaflet() %>%
  addTiles() %>%
  setView(lng = 10, lat = 62, zoom = 5) %>%
  addMarkers(lng= 10, lat= 62) %>%
  addCircles(lng= 11, lat= 63, radius = 5000)
```

Common problems with chloropleth maps



From Wikipedia

Plotting time series data

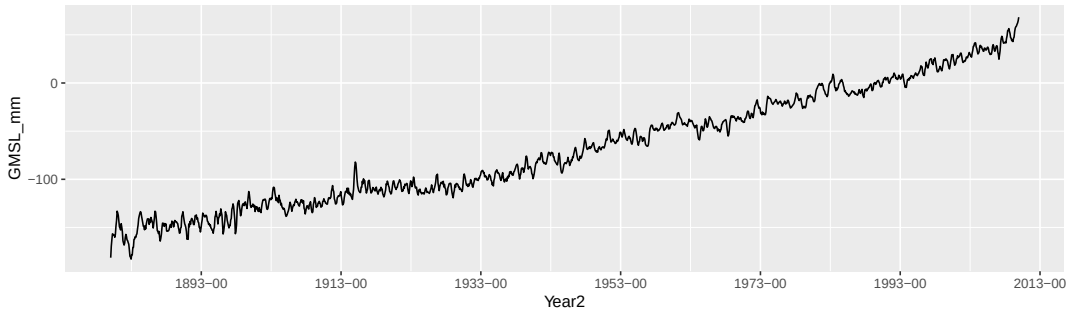
Sea level data set:

```
sl_month <- read.csv('../data/sea_level_monthly.csv')
glimpse(sl_month)
```

```
## Rows: 1,560
## Columns: 3
## $ Year                <dbl> 1880.042, 1880.125, 1880.208, 1880.292, 1880
## $ GMSL_mm             <dbl> -181.4, -169.5, -162.7, -156.6, -157.1, -158
## $ GMSL_uncertainty_mm <dbl> 24.2, 24.2, 24.2, 24.2, 24.2, 24.2, 24.2, 24
```

Plots of time series

```
sl_month %>%
  mutate(Year2 = lubridate::date_decimal(Year)) %>%
  ggplot(aes(x = Year2, y = GMSL_mm)) +
  geom_line() +
  scale_x_datetime(date_breaks = "20 years",
                   date_labels = "%Y-%M")
```



dygraphs - interactive time series

See Rstudio demo

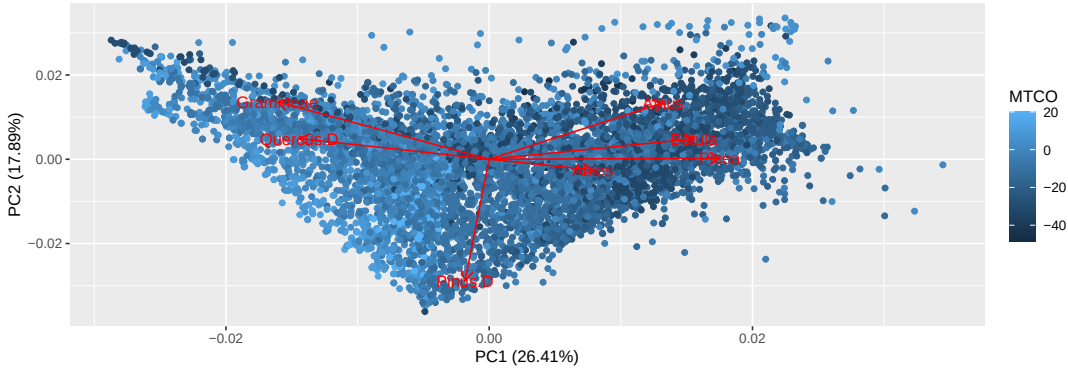
```
library(dygraphs)
sl_month %>%
  select(Year, GMSL_mm) %>%
  dygraph() %>%
  dyRangeSelector()
```

Automatic plotting with `ggfortify`

- There is a function in `ggplot2` called `autoplot` that tries to take into account the object class when creating a plot
- Other packages can use and extend this function, for example the `forecast` package uses `autoplot` to create nice time series plots
- The `ggfortify` package uses this even more generally to create nice plots for a load of different classes of object

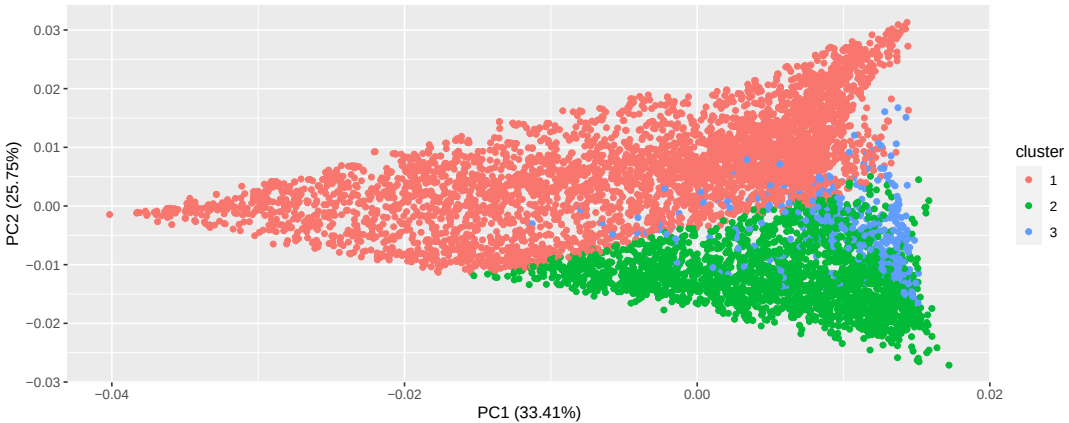
Principal Components Analysis and ggfortify

```
library(ggfortify)
pollen <- read.csv('../data/pollen.csv')
pca <- prcomp(pollen[,3:9], scale. = TRUE)
autoplot(pca, data = pollen, colour = "MTCO",
         loadings = TRUE, loadings.label = TRUE)
```



Cluster analysis and ggfortify

```
autoplot(kmeans(pollen[,3:9], 3),
         data = pollen)
```



- Can also do generalised linear models, maps, time series, other dimension reduction methods (factor analysis and multidimensional scaling), and survival data

Exercise

Run some of the above dimension reduction techniques on the `geese` data above and produce some plots. Try removing different columns (e.g. the time columns. What does the PCA tell you? (Post plots and answers in Slack)

Summary

- Lots of options for high dimensional data. Start with matrix scatter plots
- Plotting maps is a minefield. Pick a package and work through it for what you want to produce. There are many other packages and complications
- Always a good idea to try some dimension reduction when plotting high dimensional data; `ggfortify` a simple first choice