Advanced Visualizations

Statistical Computing & Programming

Shawn Santo

05-27-20

Supplementary materials

Companion videos

- Using `ggcorrplot()` and `geom_parliment()`
- Package `patchwork`
- Creating animations
- Creating interactive plots

Additional resources

- Top 50 ggplot2 visualizations
- Extend ggplot2 by creating your own stat, geom, and theme

`ggplot2` extensions

Packages

For these slides we will use the following packages.

```
library(tidyverse)
library(gapminder)   # some data
library(ggcorrplot) # correlogram plots
library(ggpol)       # parliment plots and more
library(patchwork)   # combining plots
library(gganimate)   # animations
library(ggiraph)     # interactive plots
```

Install any CRAN packages you do not have with `install.packages("package_name")`. Package `patchwork` needs to be installed by running `devtools::install_github("thomasp85/patchwork")`.

Code not shown for plots is available in the presentation notes. Press `P`.

Data: Flint water crisis

```
flint <- read_csv("http://www2.stat.duke.edu/~sms185/data/health/flint.cs
flint
```

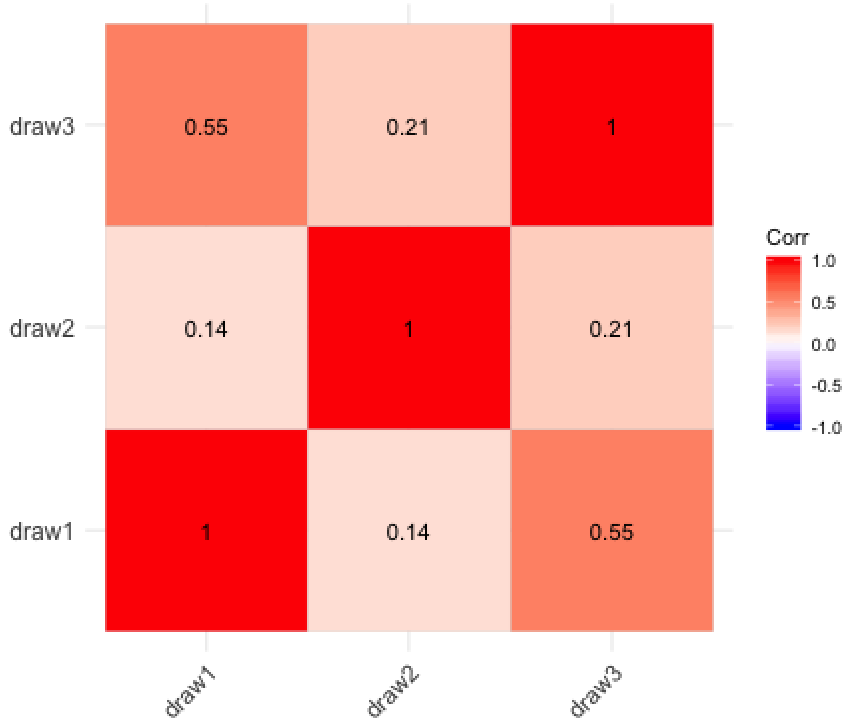
```
#> # A tibble: 271 x 6
#>       id   zip  ward  draw1  draw2 draw3
#>    <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
#>  1     1 48504     6  0.344  0.226 0.145
#>  2     2 48507     9  8.13  10.8   2.76
#>  3     4 48504     1  1.11   0.11  0.123
#>  4     5 48507     8  8.01   7.45  3.38
#>  5     6 48505     3  1.95   0.048 0.035
#>  6     7 48507     9  7.2    1.4   0.2
#>  7     8 48507     9 40.6    9.73  6.13
#>  8     9 48503     5  1.1    2.5   0.1
#>  9    12 48507     9 10.6    1.04  1.29
#> 10    13 48505     3  6.2    4.2   2.3
#> # … with 261 more rows
```

Correlogram: `ggcorrplot`

Full matrix

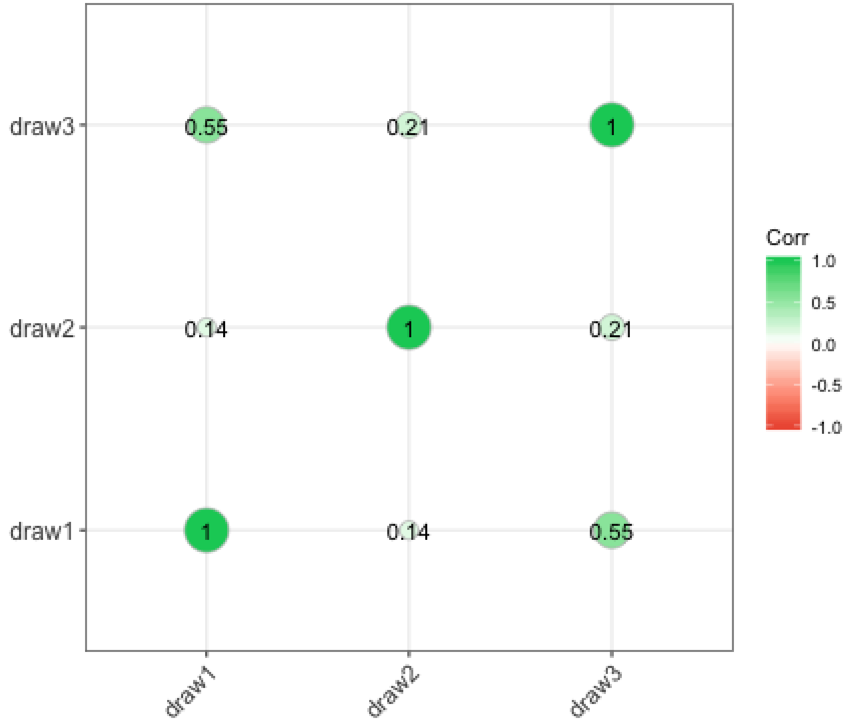
```
corr_mat <- round(cor(flint[, c("draw1", "draw2", "draw3")]), 2)

ggcorrplot(corr = corr_mat, lab = TRUE)
```



Full matrix

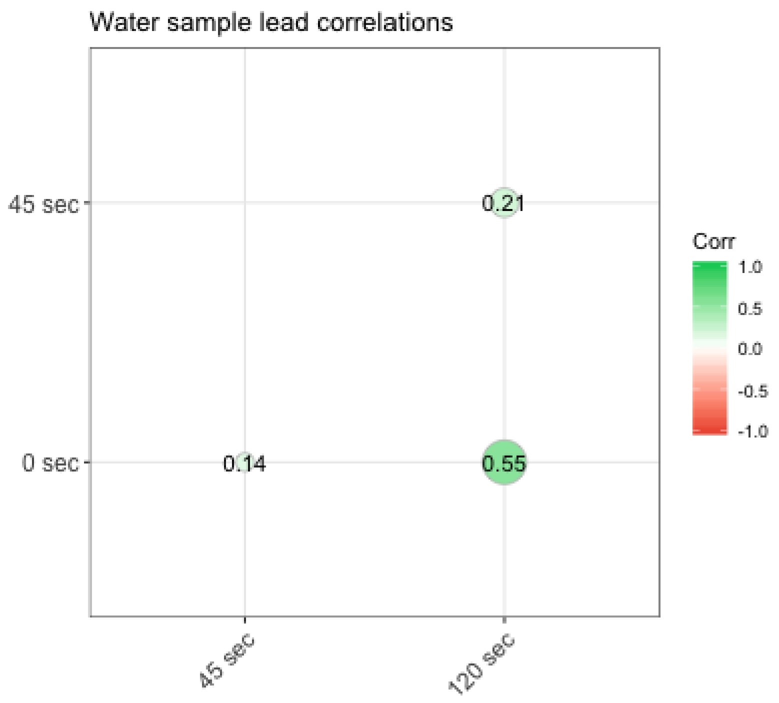
```
ggcorrplot(corr = corr_mat, method = "circle", type = "full", lab = TRUE,
           colors = c("tomato2", "white", "springgreen3"),
           ggtheme = theme_bw)
```



Lower triangular

```r
lbl <- c("0 sec", "45 sec", "120 sec")

ggcorrplot(corr = corr_mat, method = "circle", type = "lower", lab = TRUE,
           colors = c("tomato2", "white", "springgreen3"), ggtheme = theme_bw) +
  labs(title = "Water sample lead correlations") +
  scale_x_discrete(labels = lbl[2:3]) +
  scale_y_discrete(labels = lbl[1:2])
```



Parliment plots: `ggpol`

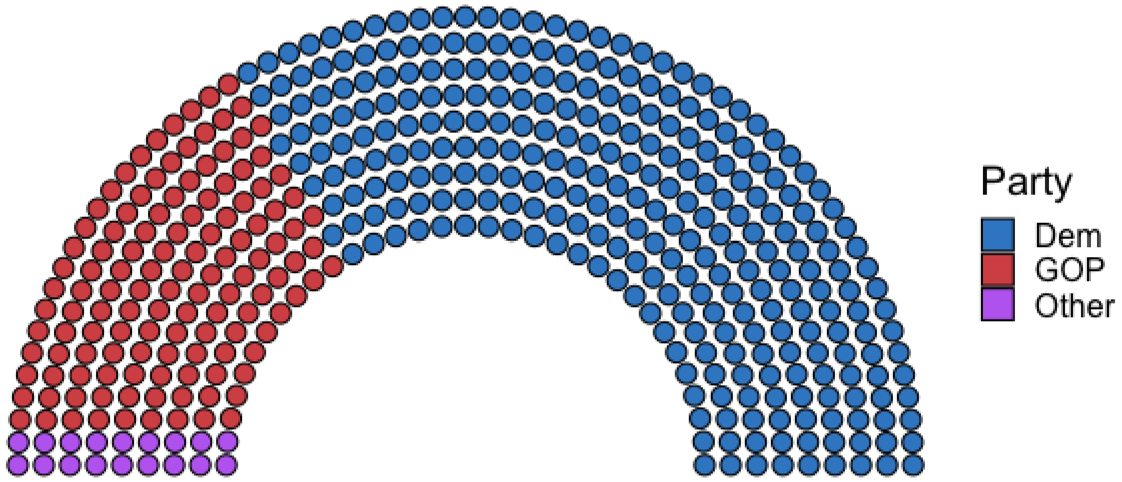
Data: Congressional seats

```
congress <- read_csv("http://www2.stat.duke.edu/~sms185/data/politics/cor
congress
```

```
#> # A tibble: 432 x 5
#>    year_start year_end party  branch seats
#>         <dbl>    <dbl> <chr>  <chr>  <dbl>
#>  1       1913     1915 dem    house    290
#>  2       1913     1915 dem    senate    51
#>  3       1913     1915 gop    house    127
#>  4       1913     1915 gop    senate    44
#>  5       1913     1915 other  house     18
#>  6       1913     1915 other  senate     1
#>  7       1913     1915 vacant house     NA
#>  8       1913     1915 vacant senate    NA
#>  9       1915     1917 dem    house    231
#> 10       1915     1917 dem    senate    56
#> # … with 422 more rows
```

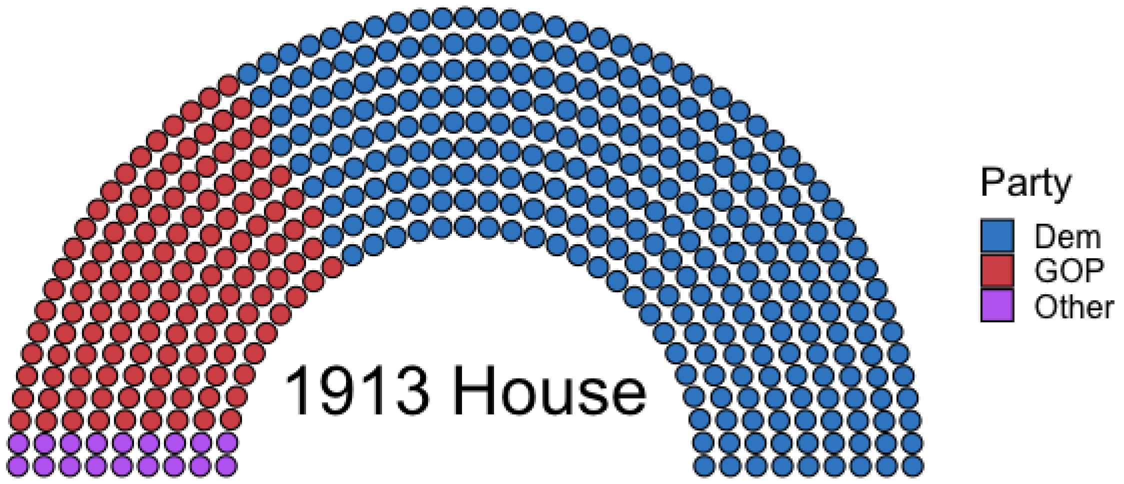
Parliment plot

```
ggplot(data = congress[congress$year_start == 1913 & congress$branch == "house", ]) +
  geom_parliament(aes(seats = seats, fill = factor(party)), show.legend = TRUE, color = "black"
  scale_fill_manual(values = c("#3A89CB", "#D65454", "#BF6FF0", "Grey"),
                    labels = c("Dem", "GOP", "Other", "Vacant")) +
  labs(fill = "Party") +
  coord_fixed() +
  theme_void(base_size = 20)
```



Annotation

```r
ggplot(data = congress[congress$year_start == 1913 & congress$branch == "house", ]) +
  geom_parliament(aes(seats = seats, fill = factor(party)), show.legend = TRUE, color = "black"
  scale_fill_manual(values = c("#3A89CB", "#D65454", "#BF6FF0", "Grey"),
                    labels = c("Dem", "GOP", "Other", "Vacant")) +
  annotate("text", x = 0, y = .5, label = "1913 House", size = 12) +
  labs(fill = "Party") +
  coord_fixed() +
  theme_void(base_size = 20)
```



Package `ggpol`

- Package `ggpol` supports a few other `geom` functions:
    - `geom_arcbar()`,
    - `geom_bartext()`,
    - `geom_circle()`,
    - `geom_tshighlight()`,
    - `geom_boxjitter()`.
- See https://github.com/erocoar/ggpol

Organizing plots: package `patchwork`

My function: `plot_congress()`

```r
plot_congress <- function(data, year, leg_branch, legend = TRUE, text_size = 12) {
  data %>%
    filter(year_start == year, branch == leg_branch) %>%
    ggplot() +
    geom_parliament(aes(seats = seats, fill = factor(party)),
                    show.legend = legend, color = "black") +
    scale_fill_manual(values = c("#3A89CB", "#D65454", "#BF6FF0", "Grey"),
                      labels = c("Dem", "GOP", "Other", "Vacant")) +
    annotate("text", x = 0, y = .5, label = paste(year, leg_branch),
             size = text_size) +
    labs(fill = "Party") +
    coord_fixed() +
    theme_void(base_size = 20)
}
```

Use package `patchwork` to organize multiple plots in a single window. No need to facet.

```r
my_plot <- ggplot()
class(my_plot)
```

```
#> [1] "gg"     "ggplot"
```

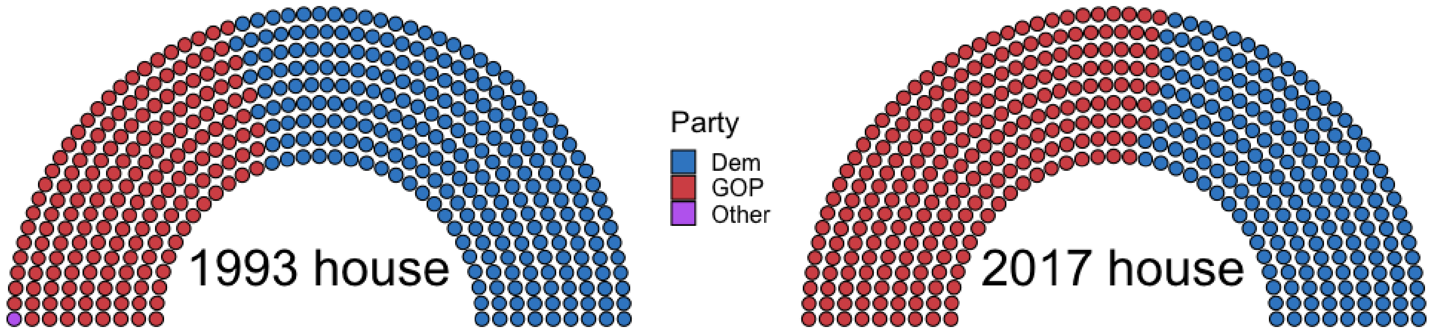
Plot creation

```r
ph_1993 <- plot_congress(congress, 1993, "house")
ph_2001 <- plot_congress(congress, 2001, "house", legend = FALSE)
ph_2009 <- plot_congress(congress, 2009, "house", legend = FALSE)
ph_2017 <- plot_congress(congress, 2017, "house", legend = FALSE)
```

Object `ph_1993` has a legend, the rest do not.

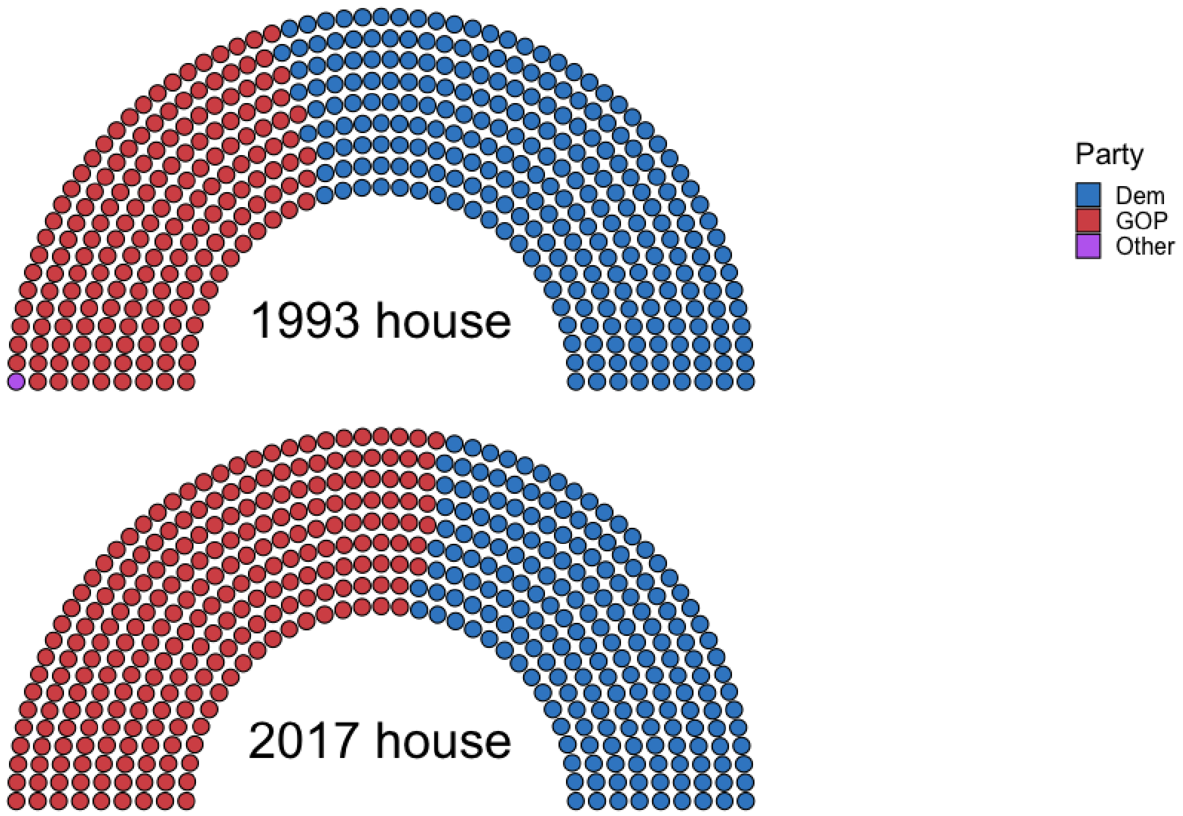
Horizontal patchwork

```
ph_1993 + ph_2017
```



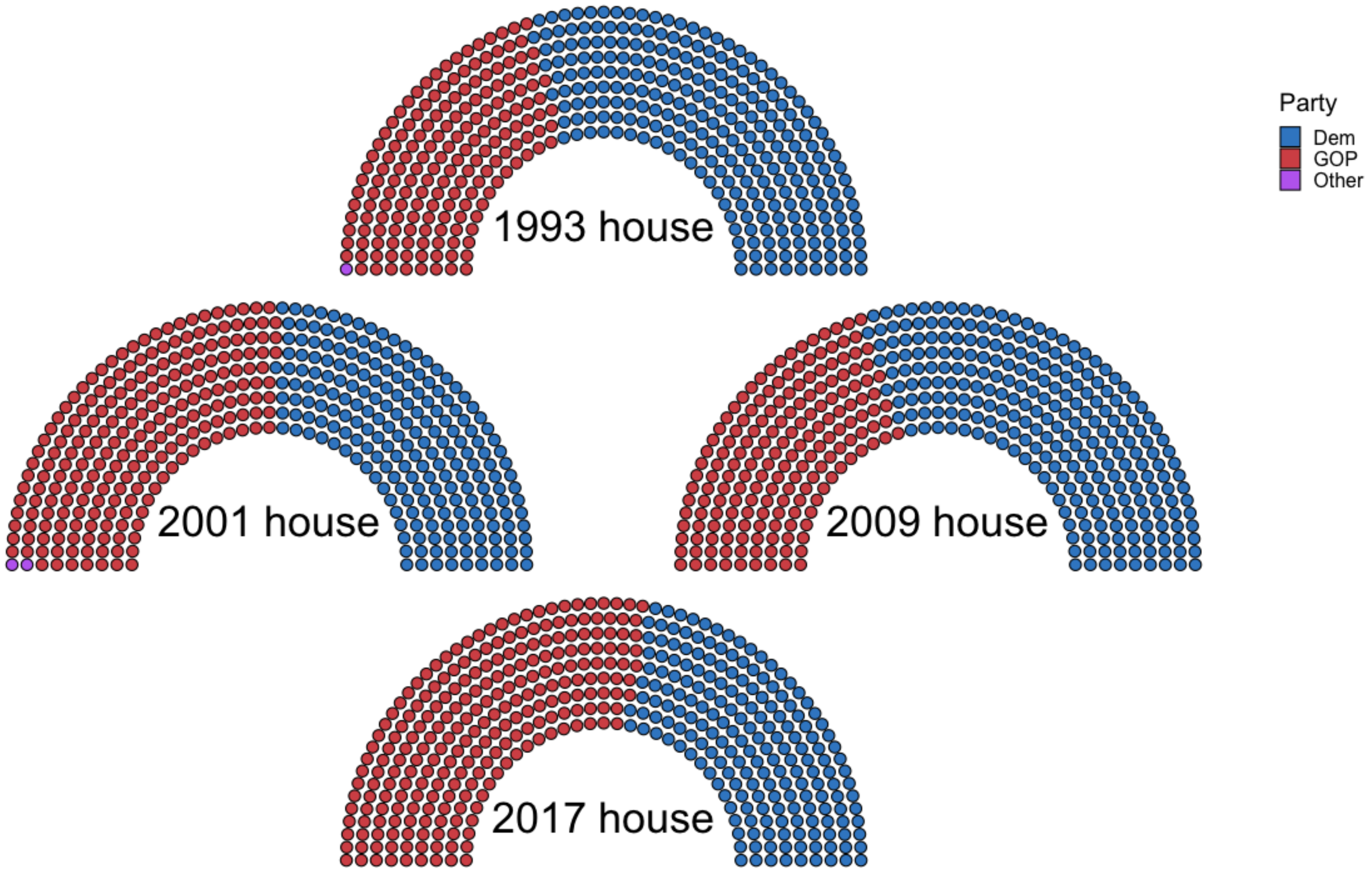
Vertical patchwork

```
ph_1993 + ph_2017 + plot_layout(ncol = 1)
```

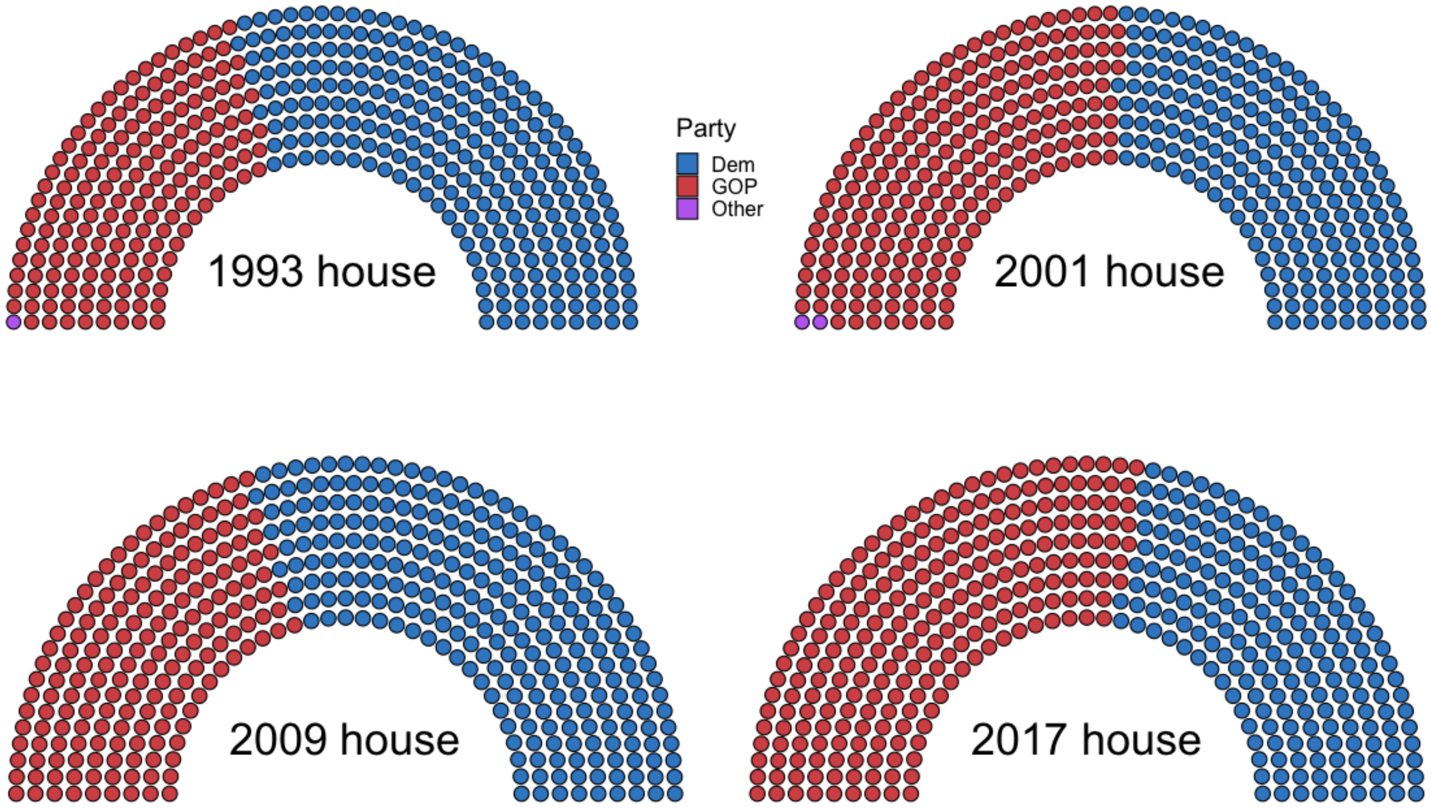


Group patchwork

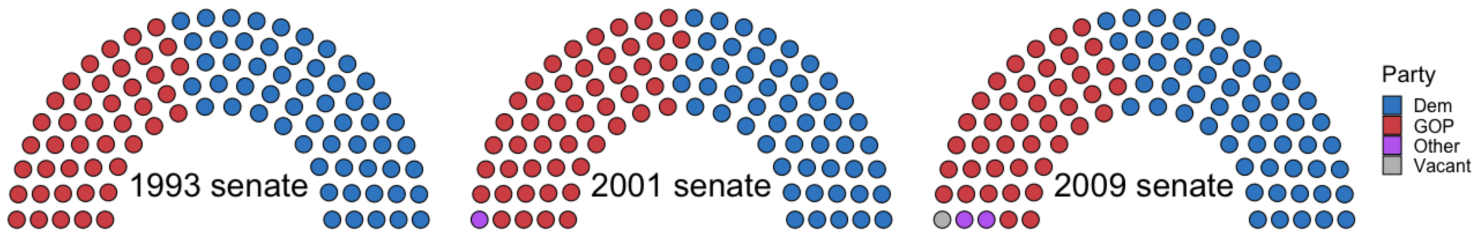
```
ph_1993 + (ph_2001 + ph_2009) + ph_2017 + plot_layout(ncol = 1)
```

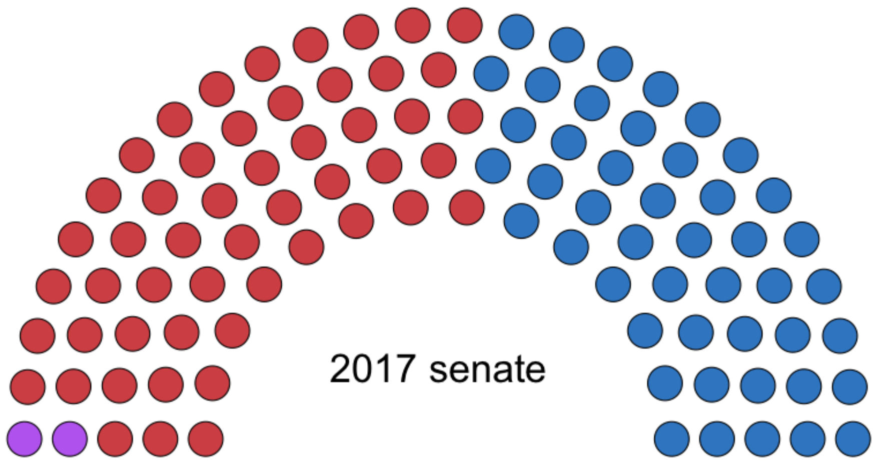


```
(ph_1993 | ph_2001) / (ph_2009 | ph_2017)
```



```
(ps_1993 | ps_2001 | ps_2009) /  ps_2017
```



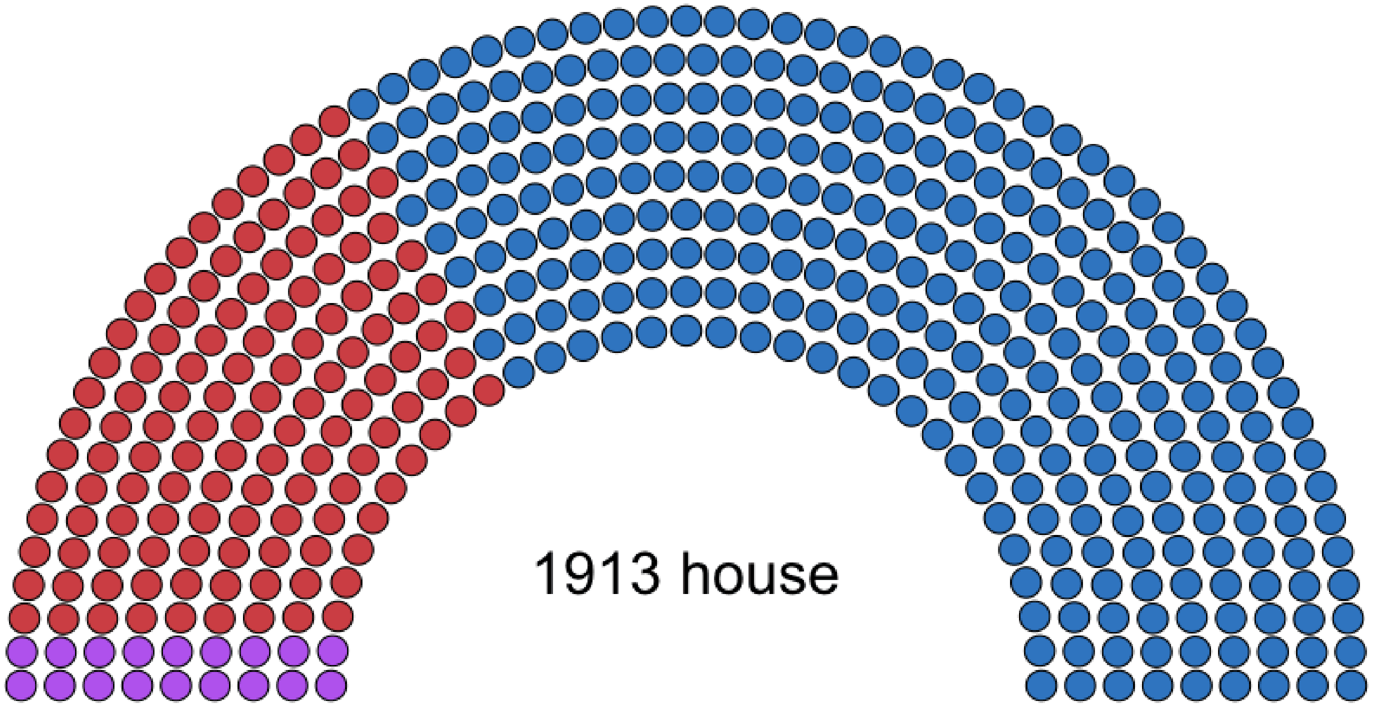


Package `patchwork`

- Supports operators `+`, `-`, `|` (besides), `/` (over)
- Specify layouts and spacing with `plot_layout()`, `plot_spacer()`, respectively
- Add grouping with `{ }` or `( )`
- Use `&` or `*` to add elements to all subplots, `*` only affects current nesting level
- See https://github.com/thomasp85/patchwork

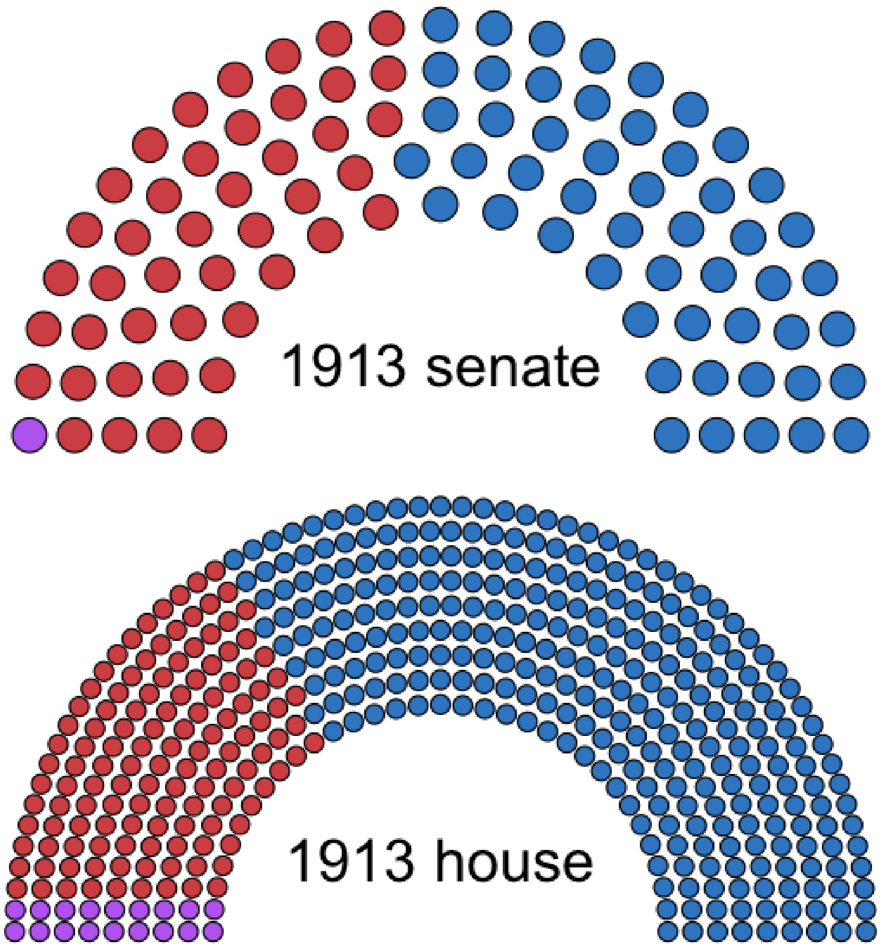
GIF: `gifski`

Using `gifski`



Dem: blue, GOP: red, Other: purple, Vacant: grey

Fast GIF with patchwork



Creating a GIF

1. Install `gifski` with `install.packages('gifski')`
2. Use chunk options

   ~~~
   ```{r animation.hook="gifski", interval = .75}

   ```
   ~~~

3. Add code for plots in a loop

   ~~~
   ```{r animation.hook="gifski", interval = .75}
   for (i in seq(1913, 2019, 2)) {
     print({
       plot_congress(congress, year = i, leg_branch = "house", legend =
     })
   }
   ```
   ~~~

4. To speed up future knits use chunk option `cache=TRUE`.

Animation: `gganimate()`

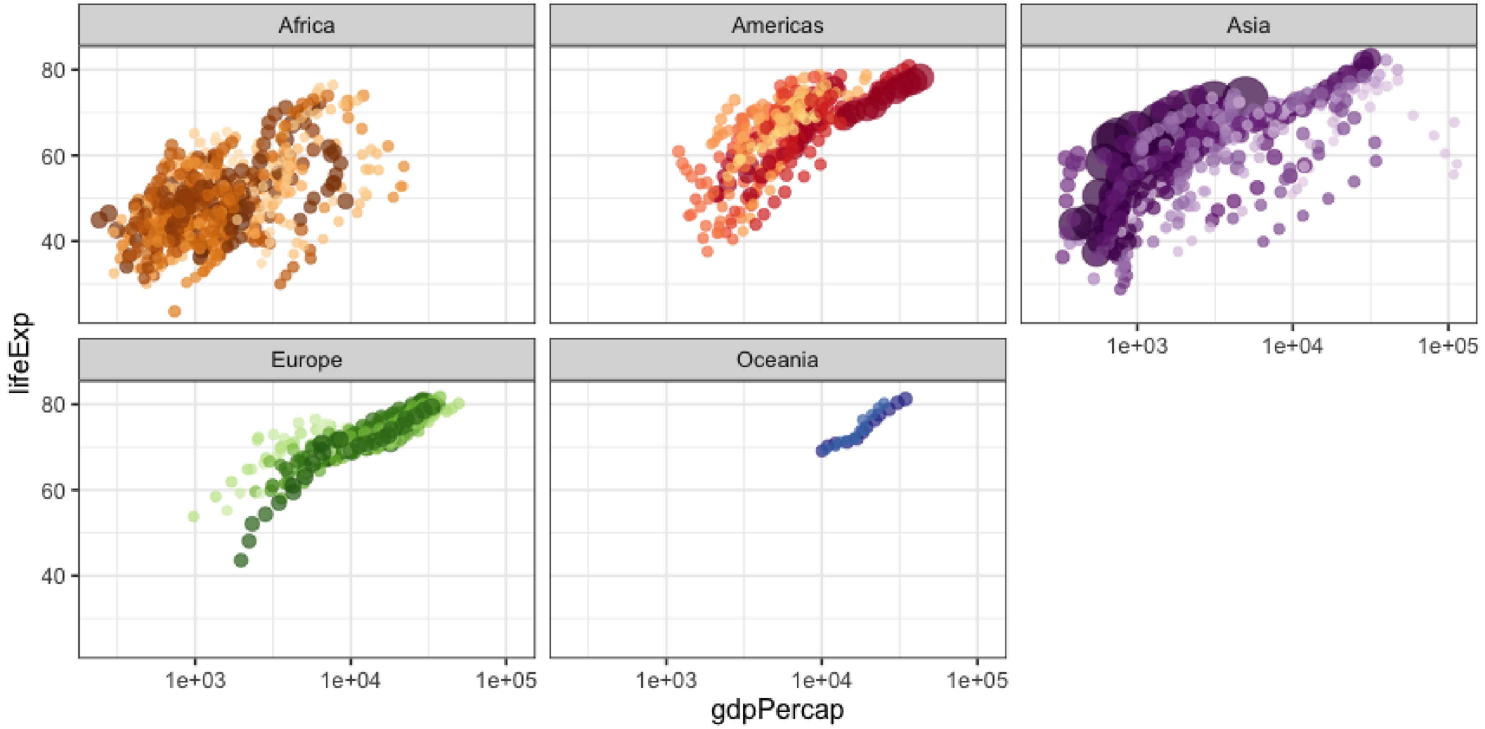
Data: gapminder

```r
library(gapminder)
gapminder
```

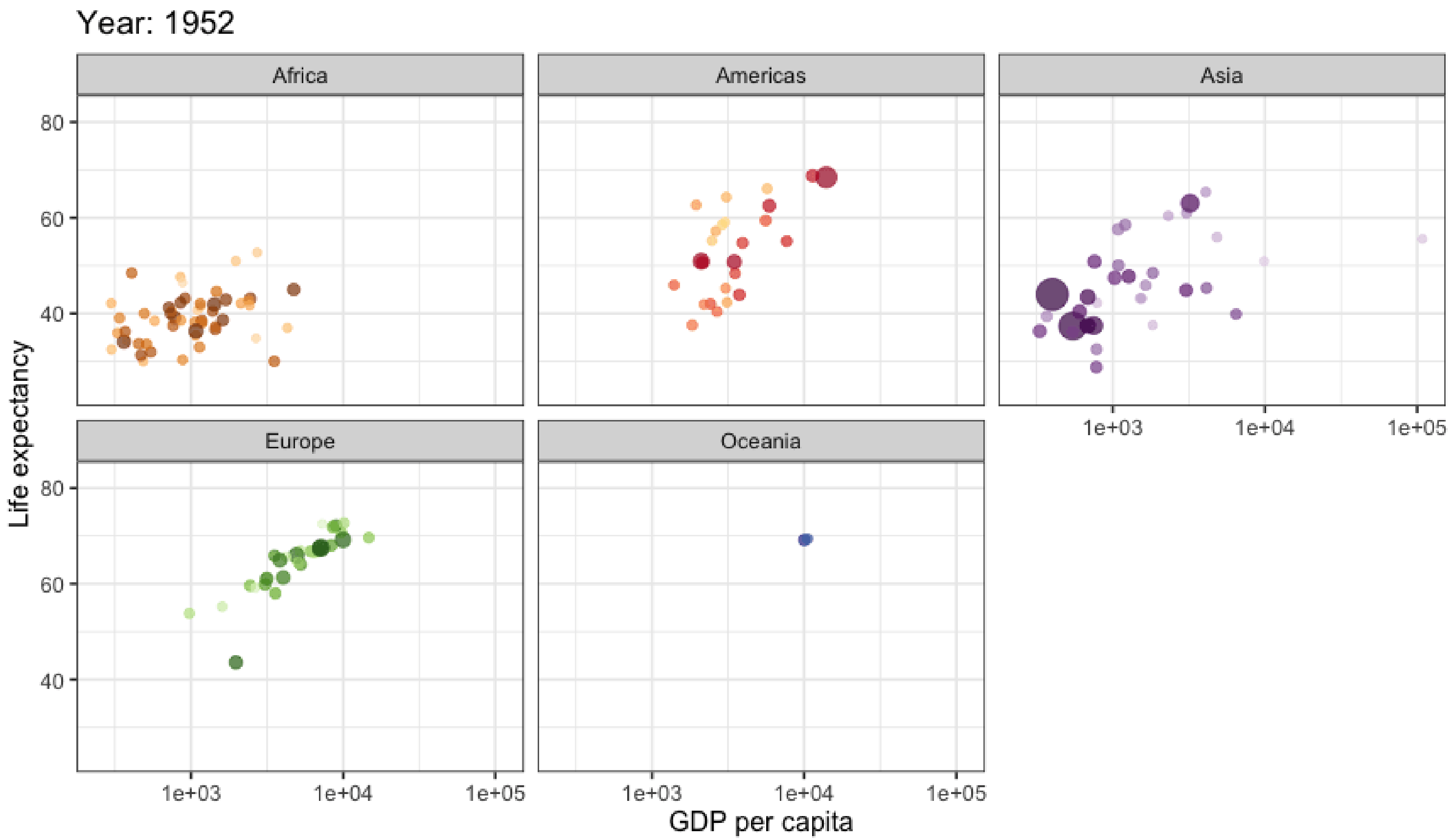
```
#> # A tibble: 1,704 x 6
#>    country     continent  year lifeExp      pop gdpPercap
#>    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
#>  1 Afghanistan Asia       1952    28.8  8425333      779.
#>  2 Afghanistan Asia       1957    30.3  9240934      821.
#>  3 Afghanistan Asia       1962    32.0 10267083      853.
#>  4 Afghanistan Asia       1967    34.0 11537966      836.
#>  5 Afghanistan Asia       1972    36.1 13079460      740.
#>  6 Afghanistan Asia       1977    38.4 14880372      786.
#>  7 Afghanistan Asia       1982    39.9 12881816      978.
#>  8 Afghanistan Asia       1987    40.8 13867957      852.
#>  9 Afghanistan Asia       1992    41.7 16317921      649.
#> 10 Afghanistan Asia       1997    41.8 22227415      635.
#> # … with 1,694 more rows
```

Nothing new

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7, show.legend = FALSE) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  facet_wrap(~continent) +
  theme_bw(base_size = 16)
```



Animate with `gganimate()`



What did we add?

Base plot

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7, show.legend = FALSE) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  facet_wrap(~continent) +
  theme_bw(base_size = 16)
```

Transform to animation

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7, show.legend = FALSE) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  facet_wrap(~continent) +
  theme_bw(base_size = 16) +
  labs(title = 'Year: {frame_time}', x = 'GDP per capita', y = 'Life expectancy') +
  transition_time(year) +
  ease_aes('linear')
```

Package `gganimate`

- Core functions
  - `transition_*()` defines how the data should be spread out and how it relates to itself across time.
  - `view_*()` defines how the positional scales should change along the animation.
  - `shadow_*()` defines how data from other points in time should be presented in the given point in time.
  - `enter_*()`/`exit_*()` defines how new data should appear and how old data should disappear during the course of the animation.
  - `ease_aes()` defines how different aesthetics should be eased during transitions.
- Label variables
  - function dependent, use `{ }` to access their values.
- See https://gganimate.com

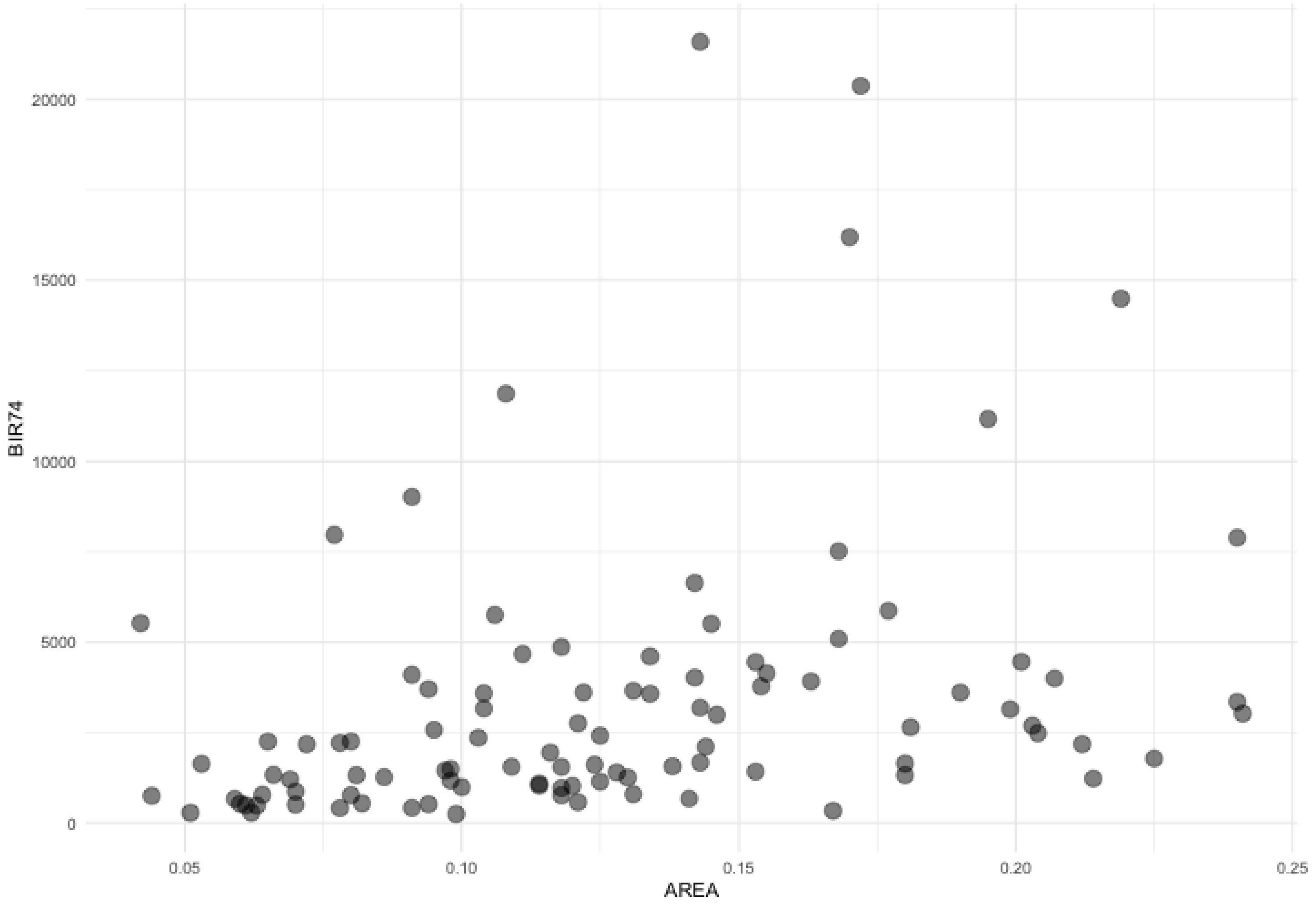
Interactive plots: `ggiraph`

Data: NC births and SID

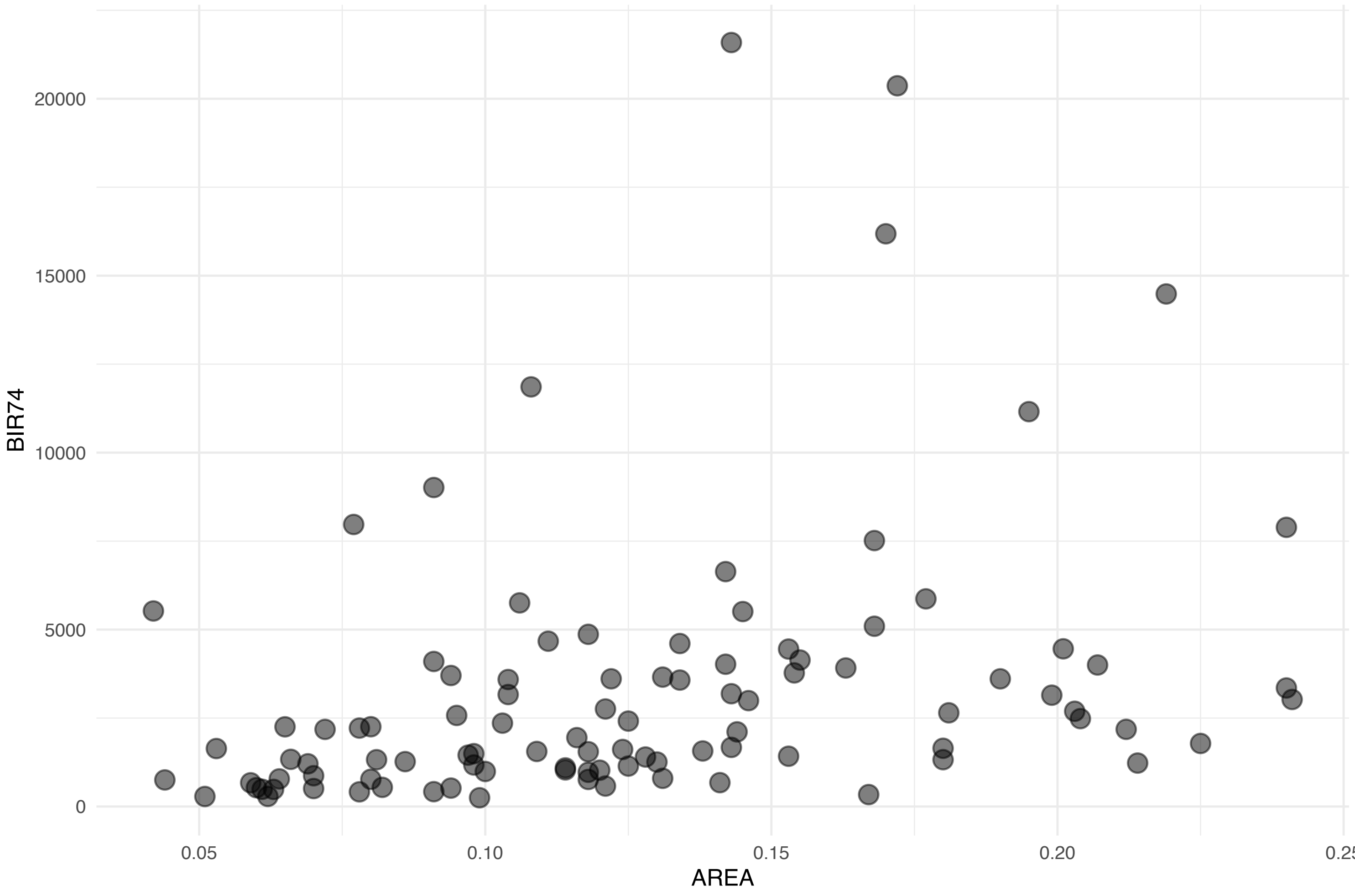
```
nc <- read_csv("http://www2.stat.duke.edu/~sms185/data/health/nc_birth_si
nc
```

```
#> # A tibble: 100 x 4
#>    NAME          AREA BIR74 SID74
#>    <chr>        <dbl> <dbl> <dbl>
#>  1 Ashe         0.114  1091     1
#>  2 Alleghany    0.061   487     0
#>  3 Surry        0.143  3188     5
#>  4 Currituck    0.07    508     1
#>  5 Northampton 0.153  1421     9
#>  6 Hertford     0.097  1452     7
#>  7 Camden       0.062   286     0
#>  8 Gates        0.091   420     0
#>  9 Warren       0.118   968     4
#> 10 Stokes       0.124  1612     1
#> # … with 90 more rows
```

Standard scatter plot



Which counties are these?



What changed?

A scatter plot with `geom_point()`

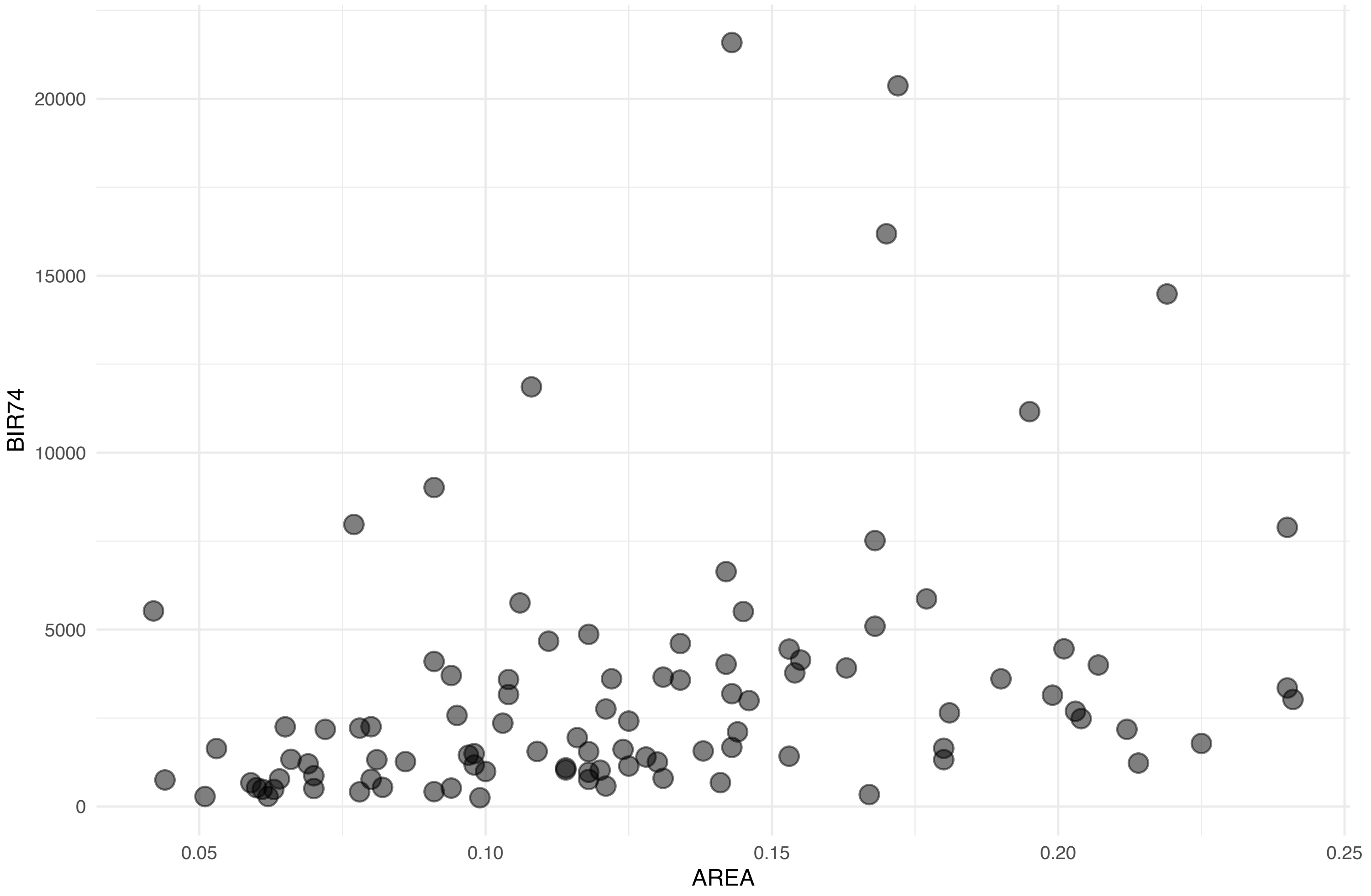
```r
ggplot(nc, mapping = aes(x = AREA, y = BIR74)) +
  geom_point(size = 4, alpha = .5) +
  theme_minimal()
```

A scatter plot with `geom_point_interactive()`

```r
gg_name <- ggplot(nc, mapping = aes(x = AREA, y = BIR74)) +
  geom_point_interactive(aes(tooltip = NAME), size = 4, alpha = .5) +
  theme_minimal()

girafe(code = {print(gg_name)}, height_svg = 6, width_svg = 9)
```

On hover functionality



What changed?

A scatter plot with `geom_point_interactive()`

```
gg_name <- ggplot(nc, mapping = aes(x = AREA, y = BIR74)) +
  geom_point_interactive(aes(tooltip = NAME), size = 4, alpha = .5) +
  theme_minimal()

girafe(code = {print(gg_name)}, height_svg = 6, width_svg = 9)
```

On hover functionality with `data_id`

```
gg_hover <- ggplot(nc, mapping = aes(x = AREA, y = BIR74)) +
  geom_point_interactive(aes(data_id = NAME, tooltip = NAME),
                         size = 4, alpha = .5) +
  theme_minimal()

girafe(code = {print(gg_hover)}, height_svg = 6, width_svg = 9)
```

On click functionality

```
nc$wiki <- sprintf("window.open(\"%s%s\")",
                   "https://www.ncpedia.org/geography/", tolower(nc$NAME)
                   )

gg_name <- ggplot(nc, mapping = aes(x = AREA, y = BIR74)) +
  geom_point_interactive(aes(tooltip = NAME, onclick = wiki), size = 4, a
  theme_minimal()

girafe(code = {print(gg_name)})
```

Package `ggiraph`

- Add tooltips, animations, and JavaScript actions to ggplot graphics
- In general, instead of `geom_<plot_type>()` use `geom_<plot_type>_interactive()`
- Interactivity is added to ggplot geometries, legends and theme elements, via the following aesthetics:
    - tooltip: tooltips to be displayed when mouse is over elements.
    - onclick: JavaScript function to be executed when elements are clicked.
    - data_id: id to be associated with elements (used for hover and click actions)
- Function `girafe()` translates the graphic into an interactive web-based graphic
- See https://github.com/davidgohel/ggiraph

Exercise

Flint water data

```
flint <- read_csv("http://www2.stat.duke.edu/~sms185/data/health/flint.cs
flint
```

```
#> # A tibble: 271 x 6
#>        id   zip  ward  draw1  draw2 draw3
#>     <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
#>  1      1 48504     6  0.344  0.226 0.145
#>  2      2 48507     9  8.13  10.8   2.76
#>  3      4 48504     1  1.11   0.11  0.123
#>  4      5 48507     8  8.01   7.45  3.38
#>  5      6 48505     3  1.95   0.048 0.035
#>  6      7 48507     9  7.2    1.4   0.2
#>  7      8 48507     9 40.6    9.73  6.13
#>  8      9 48503     5  1.1    2.5   0.1
#>  9     12 48507     9 10.6    1.04  1.29
#> 10     13 48505     3  6.2    4.2   2.3
#> # … with 261 more rows
```

Create a visualization of the data from object `flint`. Incorporate topics from today's lecture.

More visualization resources

- http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html
- https://plot.ly/ggplot2/

References

- https://yihui.name/en/2018/08/gifski-knitr/
- http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html
- https://ggplot2.tidyverse.org/articles/extending-ggplot2.html
- https://github.com/davidgohel/ggiraph
- https://gganimate.com
- https://github.com/thomasp85/patchwork
- https://github.com/erocoar/ggpol