

CSSS 569 Visualizing Data and Models

Replication: HW1 - Problem 2

Ramses Llobet

Department of Political Science, UW

January 29, 2026

Introduction

- ▶ Let's first do a quick review of how `ggplot2` works.

Grammar of graphics

- ▶ A statistical graphic is a **mapping** of **data** variables to **aesthetic** attributes of **geometric** objects. (Wilkinson 2005)

Grammar of graphics in ggplot2

- ▶ *What* data do you want to visualize?
 - ▶ `ggplot(data = ...)`
- ▶ *How* are variables mapped to specific aesthetic attributes?
 - ▶ `aes(... = ...)`
 - ▶ positions (x, y), shape, colour, size, fill, alpha, linetype, label...
 - ▶ If the value of an attribute do not vary w.r.t. some variable, don't wrap it within `aes(...)`
- ▶ *Which* geometric shapes do you use to represent the data?
 - ▶ `geom_{}`:
 - ▶ `geom_point`, `geom_line`, `geom_ribbon`, `geom_polygon`, `geom_label`...

ggplot2: A layered grammar

- ▶ ggplot2: A *layered* grammar of graphics (Wickham 2009)
 - ▶ Build a graphic from multiple layers; each consists of some geometric objects or transformation
 - ▶ Use + to stack up layers
- ▶ Within each `geom_{}` layer, two things are *inherited* from previous layers
 - ▶ Data: inherited from the master data
 - ▶ Aesthetics: inherited (`inherit.aes = TRUE`) from the master aesthetics
 - ▶ They are convenient but create unintended consequences
 - ▶ We'll revisit them very soon and learn how to overwrite them

Tidy data

- ▶ ggplot2 works well only with tidy data

- ▶ *Tidy data:*

- ▶ Each **variable** must have its own **column**
- ▶ Each **observation** must have its own **row**
- ▶ Each value must have its own cell

- ▶ Example: iverRevised.csv for Homework1

```
## # A tibble: 6 x 4
##   country povertyReduction effectiveParties partySystem
##   <chr>          <dbl>          <dbl> <chr>
## 1 Australia      42.2            2.38 Majoritarian
## 2 Belgium        78.8            7.01 Proportional
## 3 Canada         29.9            1.69 Majoritarian
## 4 Denmark        71.5            5.04 Proportional
## 5 Finland        69.1            5.14 Proportional
## 6 France         57.9            2.68 Majoritarian
```

Building a plot from scratch

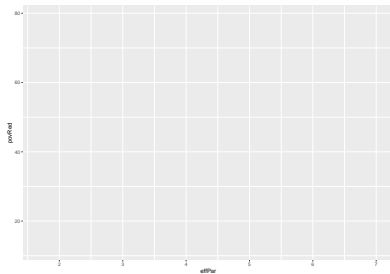
```
# Load packages
library(tidyverse)
library(RColorBrewer)
library(ggrepel)

# Load data
iver <- read_csv("https://faculty.washington.edu/cadolph/vis/iverRevised.csv")

# Shorten the variable names
iver <- iver %>%
  rename(povRed = povertyReduction,
         effPar = effectiveParties,
         parSys = partySystem)
```

Building a plot from scratch

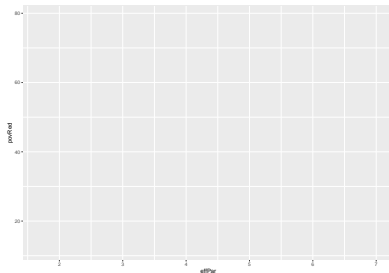
```
ggplot(  
  data = iver,  
  mapping = aes(y = povRed,  
                 x = effPar)  
)
```



Building a plot from scratch

`data = ...` and `mapping = ...` can be omitted for simplicity

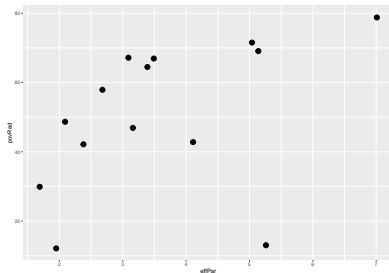
```
ggplot(  
  iver,  
  aes(y = povRed, x = effPar)  
)
```



Building a plot from scratch

No data will be drawn until you supply
`geom_{}`

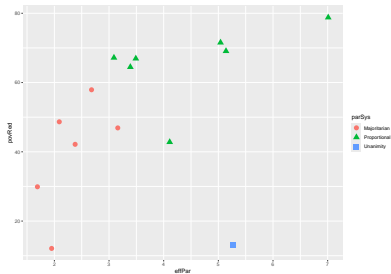
```
ggplot(  
  iver,  
  aes(y = povRed, x = effPar)  
) +  
  geom_point()
```



Building a plot from scratch

Map variable partySystem to aesthetics

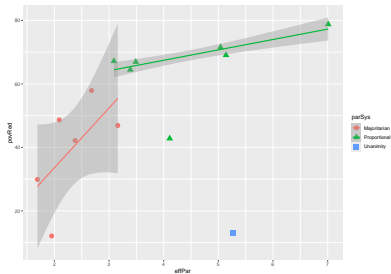
```
ggplot(  
  iver,  
  aes(y = povRed, x = effPar,  
      colour = parSys,  
      shape = parSys)  
) +  
  geom_point()
```



Building a plot from scratch

Why does it produce multiples smooth curves?

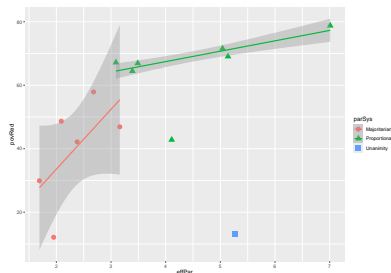
```
ggplot(
  iver,
  aes(y = povRed, x = effPar,
      colour = parSys,
      shape = parSys)
) +
  geom_point() +
  geom_smooth(method = MASS::rlm)
```



Building a plot from scratch

There is a hidden `inherit.aes = TRUE` default argument in every `geom_{}`

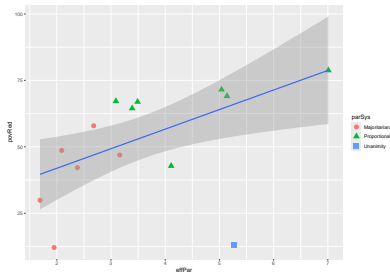
```
ggplot(  
  iver,  
  aes(y = povRed, x = effPar,  
      colour = parSys,  
      shape = parSys)  
) +  
  geom_point(  
    inherit.aes = TRUE,  
    aes(y = povRed, x = effPar,  
        colour = parSys,  
        shape = parSys)  
) +  
  geom_smooth(  
    inherit.aes = TRUE,  
    aes(y = povRed, x = effPar,  
        colour = parSys,  
        shape = parSys),  
    method = MASS::rlm  
)
```



Building a plot from scratch

One solution: localize different aesthetic settings to specific layers

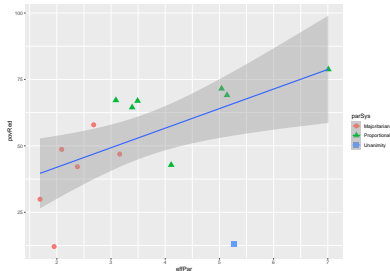
```
ggplot(  
  iver,  
  aes(y = povRed, x = effPar)  
) +  
  geom_point(  
    aes(colour = parSys,  
        shape = parSys),  
    size = 4  
  ) +  
  geom_smooth(method = MASS::rlm)
```



Building a plot from scratch

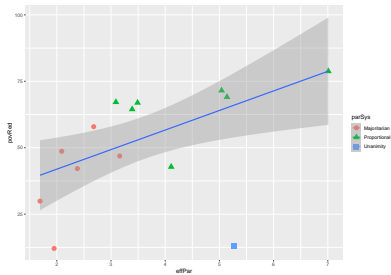
Another solution: override the grouping with `aes(group = 1)`

```
ggplot(
  iver,
  aes(y = povRed, x = effPar,
      colour = parSys,
      shape = parSys)
) +
  geom_point() +
  geom_smooth(
    aes(group = 1),
    method = MASS::rlm
  )
```



Building a plot from scratch:

How to override the default colors? Let's learn how to get nice colors first



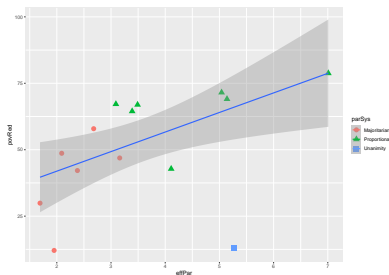
Building a plot from scratch:

Get nice colors with RColorBrewer package; see [here for palettes](#)

```
library(RColorBrewer)
colors <- brewer.pal(n = 3, "Set1")
red <- colors[1]
blue <- colors[2]
green <- colors[3]

print(c(red, blue, green))

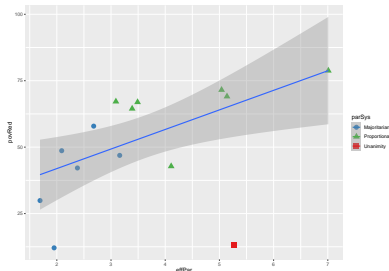
## [1] "#E41A1C" "#377EB8" "#4DAF4A"
```



Building a plot from scratch:

You can scale every aesthetic
(i.e. overwrite the default) you mapped

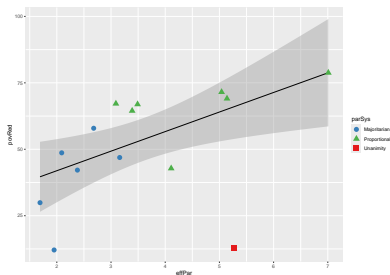
```
ggplot(  
  iver,  
  aes(y = povRed, x = effPar,  
      colour = parSys,  
      shape = parSys)  
) +  
  geom_point() +  
  geom_smooth(  
    aes(group = 1),  
    method = MASS::rlm  
  ) +  
  scale_color_manual(  
    values = c(  
      "Majoritarian" = blue,  
      "Proportional" = green,  
      "Unanimity" = red  
    )  
  )  
)
```



Building a plot from scratch:

Two tweaks: (1) plot `geom_smooth` first, then `geom_point` (why?); (2) adjust the color and size of `geom_smooth` (no need in aes; why?)

```
ggplot(  
  iver,  
  aes(y = povRed, x = effPar,  
      colour = parSys,  
      shape = parSys)  
) +  
  geom_smooth(  
    aes(group = 1),  
    method = MASS::rlm,  
    color = "black",  
    linewidth = 0.5  
) +  
  geom_point() +  
  scale_color_manual(  
    values = c(  
      "Majoritarian" = blue,  
      "Proportional" = green,  
      "Unanimity" = red  
    )  
  )  
)
```



Building a plot from scratch:

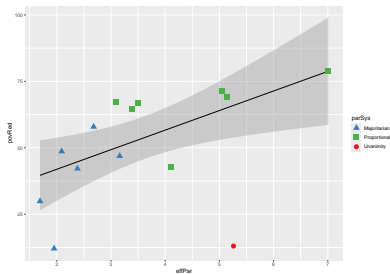
Let's first save what we have so far

```
p <- ggplot(  
  iver,  
  aes(y = povRed, x = effPar,  
      colour = parSys,  
      shape = parSys)  
) +  
geom_smooth(  
  aes(group = 1),  
  method = MASS::rlm,  
  color = "black",  
  linewidth = 0.5  
) +  
geom_point() +  
scale_color_manual(  
  values = c(  
    "Majoritarian" = blue,  
    "Proportional" = green,  
    "Unanimity" = red  
  )  
)
```

Building a plot from scratch:

Similarly, you can scale shape; see [here](#) for all shapes.

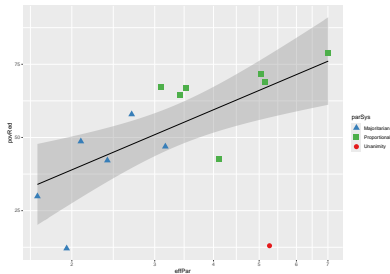
```
p <- p +  
  scale_shape_manual(  
    values = c(  
      "Majoritarian" = 17,  
      "Proportional" = 15,  
      "Unanimity" = 16  
    )  
  )  
  
print(p)
```



Building a plot from scratch:

Similarly, you can scale y and x (they are also inside aes!)

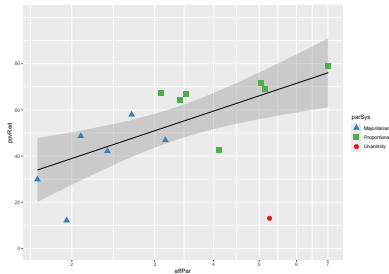
```
p <- p +  
  scale_x_continuous(  
    trans = "log",  
    breaks = 2:7  
  )  
  
print(p)
```



Building a plot from scratch:

But limits of y must be large enough to incorporate the confidence regions produced by `geom_smooth`

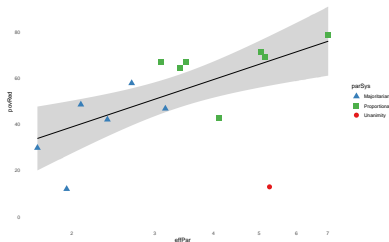
```
p <- p +  
  scale_y_continuous(  
    breaks = seq(0, 80, 20),  
    limits = c(0, 100)  
  )  
  
print(p)
```



Building a plot from scratch:

Remove unhelpful elements (e.g. grey background, gridlines etc.) using `theme`

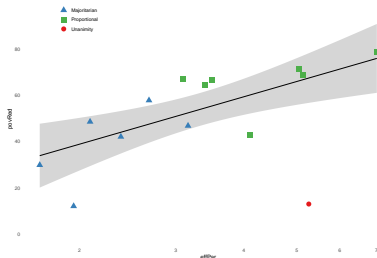
```
p <- p +  
  theme(  
    panel.background =  
      element_rect(fill = NA),  
    axis.ticks.x =  
      element_blank(),  
    axis.ticks.y =  
      element_blank(),  
  )  
  
print(p)
```



Building a plot from scratch:

How do we embed the legend within the plot and remove unhelpful elements?

```
p <- p +  
  theme(  
    legend.position =  
      c(0.15, 0.8),  
    legend.title =  
      element_blank(),  
    legend.background =  
      element_blank(),  
    legend.key =  
      element_rect(fill = NA,  
                    color = NA)  
  )  
  
print(p)
```



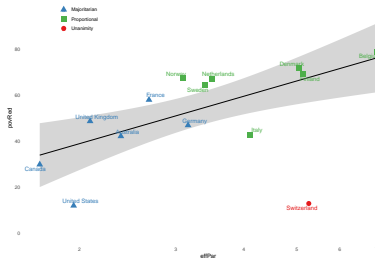
Building a plot from scratch:

With a much cleaner graph, we can augment the graph with more information: `label`

```
library(ggrepel)

p +
  geom_text_repel(
    aes(label = country)
  )

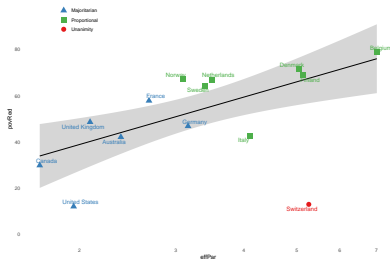
print(p)
```



Building a plot from scratch:

Something is wrong with the legend once we have too many mappings:

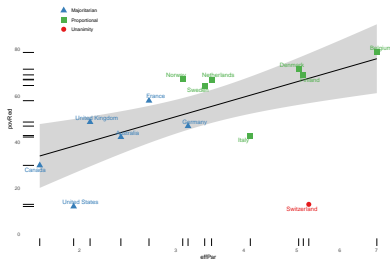
```
p <- p +  
  geom_text_repel(  
    aes(label = country),  
    show.legend = FALSE  
  )  
  
print(p)
```



Building a plot from scratch:

With a much cleaner graph, we can augment the graph with more information: `geom_rug`

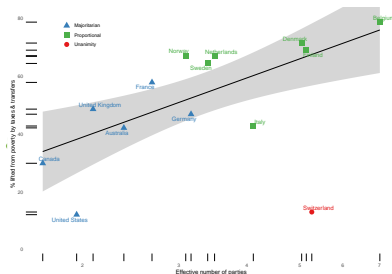
```
p <- p +  
  geom_rug(color = "black")  
  
print(p)
```



Building a plot from scratch:

Final tweaks: x-axis title, y-axis title, coordinate limits

```
p <- p +  
  labs(  
    x = "Effective number of parties",  
    y = "% lifted from poverty by taxes",  
    #title = ...  
  ) +  
  coord_cartesian(ylim = c(0, 80))  
  
print(p)
```



Building a plot from scratch:

Full code to reproduce the graph:

```
ggplot(iver, aes(y = povRed, x = effPar, color = parSys, shape = parSys)) +  
  geom_smooth(aes(group = 1), colour = "black", linewidth = 0.25,  
              method = MASS::rlm, method.args = list(method = "MM")) +  
  geom_point(size = 2) +  
  geom_text_repel(aes(label = country), show.legend = FALSE) +  
  geom_rug(color = "black", size = 0.25) +  
  scale_shape_manual(values = c(17, 15, 16)) +  
  scale_color_manual(values = c(blue, green, red)) +  
  scale_x_continuous(trans = "log", breaks = 2:7) +  
  scale_y_continuous(breaks = seq(0, 80, 20), limits = c(0, 100)) +  
  theme(panel.background = element_rect(fill = NA),  
        axis.ticks.x = element_blank(),  
        axis.ticks.y = element_blank(),  
        legend.position = c(0.15, 0.89),  
        legend.title = element_blank(),  
        legend.background = element_blank(),  
        legend.key = element_rect(fill = NA, color = NA)) +  
  coord_cartesian(ylim = c(0, 80)) +  
  labs(x = "Effective number of parties",  
       y = "% lifted from poverty by taxes & transfers")
```

Building a plot from scratch:

How to save a graph into PDF?

```
width <- 8  
ggsave("iverPlot.pdf", width = width, height = width/1.618, units = "in")
```

Customized theme

- ▶ You won't be alone in thinking that it's quite tedious...
 - ▶ Beginner-friendly defaults come at a cost of painstakingly overwriting them
- ▶ Chris and I wrote a `ggplot2` theme that implements visual principles taught in lectures and his graphic style
 - ▶ `theme_caviz.R` can be found [here](#)
 - ▶ which contains three theme objects: `theme_caviz`, `theme_caviz_hgrid`, `theme_caviz_vgrid`

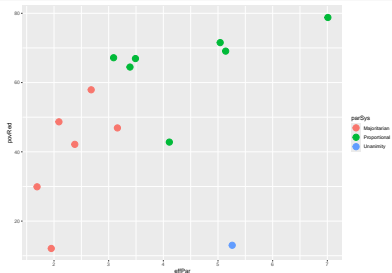
Customized theme

► To use it, simply:

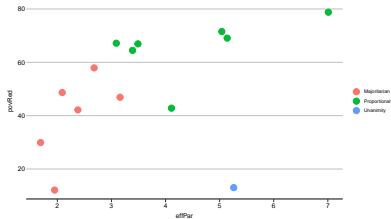
```
# Source the R script  
source("your_local_directory/theme_cavis.R")  
  
# Then add it to your ggplot object as usual  
some_ggplot_object +  
  theme_cavis
```

Quick showcase

```
ggplot(
  iver,
  aes(x = effPar, y = povRed,
      color = parSys)
) +
  geom_point(size = 5)
```



```
ggplot(
  iver,
  aes(x = effPar, y = povRed,
      color = parSys)
) +
  geom_point(size = 5) +
  theme_cavis_hgrid
```



FIN