Основы программирования в R

Визуализация с `ggplot2`: часть 2

Алла Тамбовцева, НИУ ВШЭ

Содержание

Столбиковые диаграммы (*bar charts*)

Для начала загрузим библиотеку `tidyverse` и возьмем наш любимый датафрейм по показателям *WGI* и *Freedom House*. Заодно удалим строки с пропущенными значениями:

```
library(tidyverse)
dat <- read.csv("https://allatambov.github.io/rprog/data/wgi-new.csv")
dat <- na.omit(dat)
```

Построим столбиковую диаграмму для визуализации частот значений в столбце `fh_type`, чтобы показать сколько стран разного типа в датафрейме. Для начала получим таблицу с частотами: сгруппируем данные по столбцу `fh_type` и посчитаем число наблюдений в каждой группе.

```
tab <- dat %>% group_by(fh_type) %>% tally
tab
```

```
## # A tibble: 3 x 2
##   fh_type         n
##   <fct>       <int>
## 1 free           85
## 2 not_free       50
## 3 partly_free    60
```

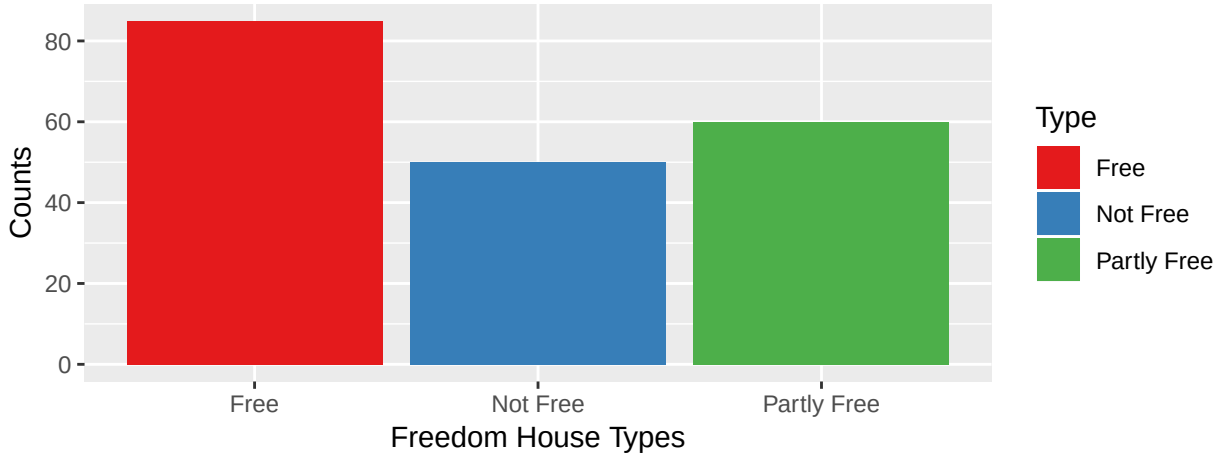
Теперь построим столбиковую диаграмму, по оси `x` у нас будут идти значения `fh_type`, а по оси `y` — частоты из столбца `n`. Для этого нам потребуется функция `geom_bar()`, внутри которой укажем `stat = 'identity'`, чтобы показать, что в `tab` уже хранятся готовые частоты.

{r. fig.height = 2.5} ggplot(data = tab, aes(x = fh_type, y = n)) + geom_bar(stat = 'identity')

Добавим цвет заливки, зависящий от значений в `fh_type`, выберем готовую палитру для столбцов и подпишем оси:

```
# scale_fill_brewer - настройки легенды
# scale_x_discrete - настройки подписей по оси x

ggplot(data = tab, aes(x = fh_type, y = n,
                       fill = fh_type)) +
  geom_bar(stat = 'identity') +
  scale_fill_brewer(palette = 'Set1',
                    name = 'Type',
                    labels = c('Free', 'Not Free', 'Partly Free')) +
```

```
  scale_x_discrete(labels = c('Free', 'Not Free', 'Partly Free')) +
  labs(x = 'Freedom House Types',
       y = 'Counts')
```

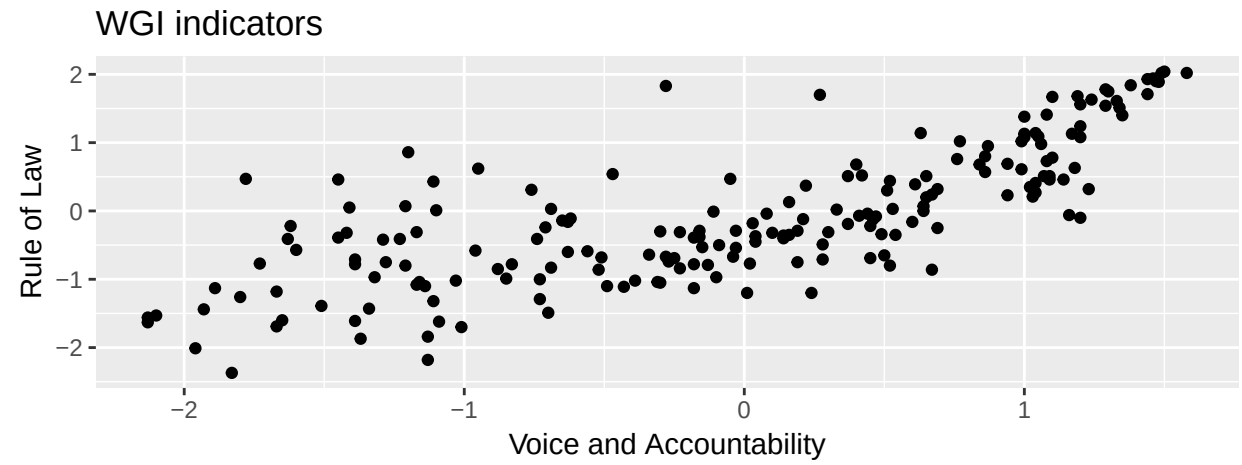


Подробнее про разные столбиковые диаграммы в `ggplot2` можно почитать здесь.

Диаграммы рассеяния (*scatter plots*) и пузырьковые диаграммы (*bubble plots*)

Построим диаграмму рассеяния для индексов *Voice & Accountability* (`va`) и Rule of Law (`rl`).

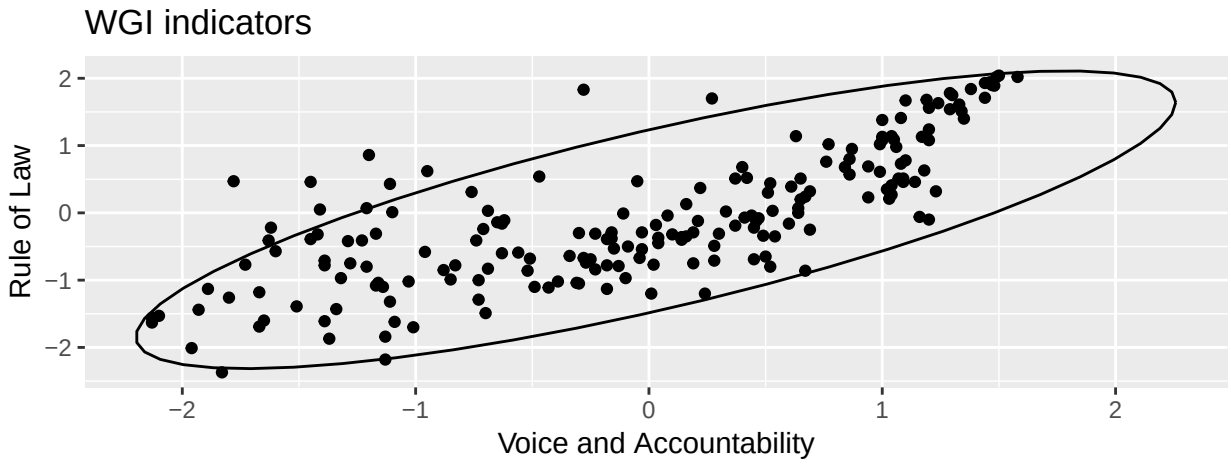
```
ggplot(data = dat, aes(x = va, y = rl)) +
geom_point() +
labs(title = "WGI indicators",
     x = "Voice and Accountability",
     y = "Rule of Law")
```



Можем наложить на эту диаграмму эллипс рассеяния, чтобы было проще судить о направлении и силе связи:

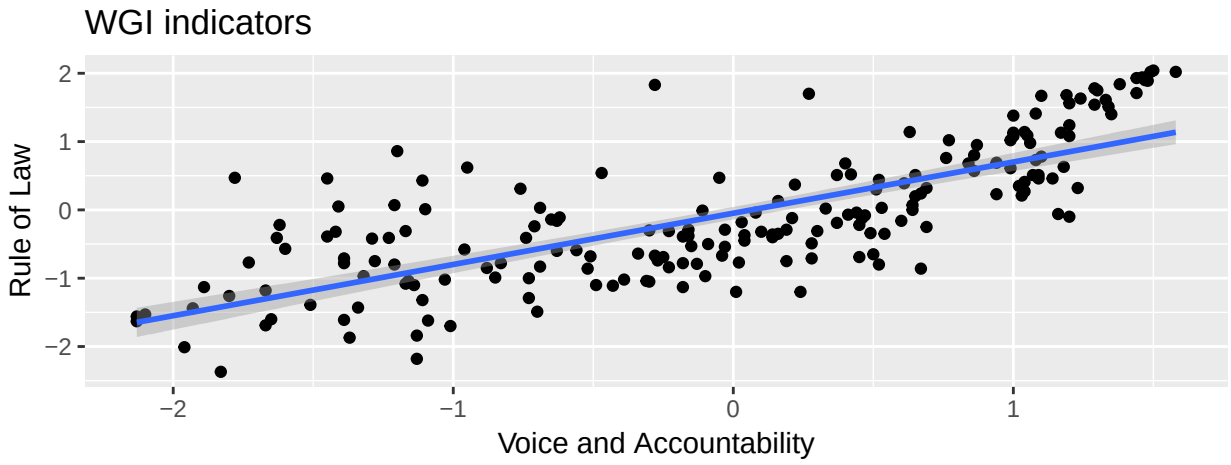
```
# слой stat_ellipse

ggplot(data = dat, aes(x = va, y = rl)) +
geom_point() +
labs(title = "WGI indicators",
```

```
     x = "Voice and Accountability",
     y = "Rule of Law") + stat_ellipse()
```



А можно добавить регрессионную прямую, которая будет иллюстрировать, насколько изменяется значение индекса *Voice & Accountability* при увеличении индекса *Rule of Law* на единицу:

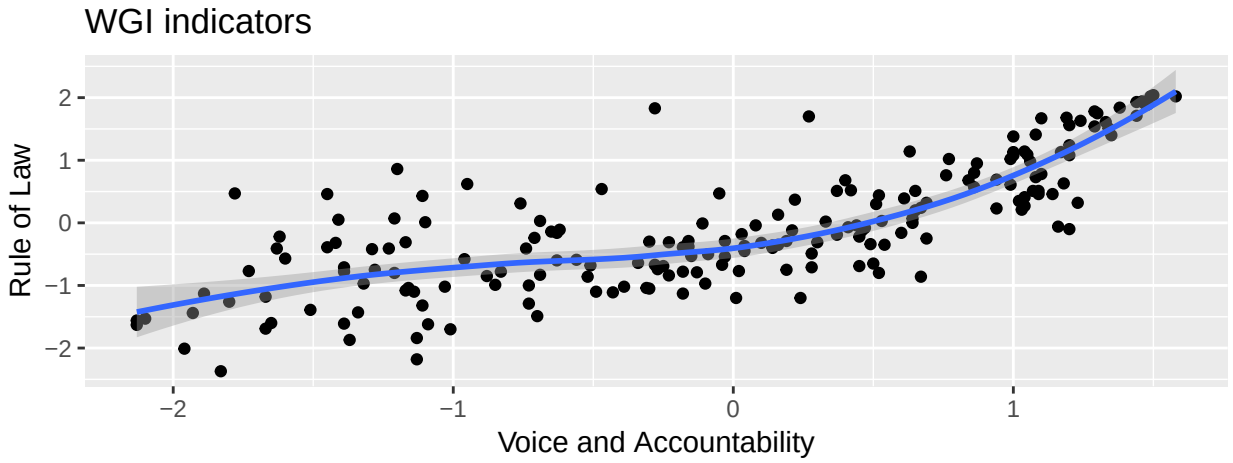
```
# lm - om linear model

ggplot(data = dat, aes(x = va, y = rl)) +
geom_point() +
labs(title = "WGI indicators",
     x = "Voice and Accountability",
     y = "Rule of Law") +
  geom_smooth(method = lm)
```



Если убрать `method` в слое `geom_smooth()` и оставить настройки по умолчанию, то будет построена сглаженная регрессия (*lowess* или *loess*, мы ее отчасти обсуждали, см. здесь):

```
ggplot(data = dat, aes(x = va, y = rl)) +
geom_point() +
labs(title = "WGI indicators",
     x = "Voice and Accountability",
     y = "Rule of Law") +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

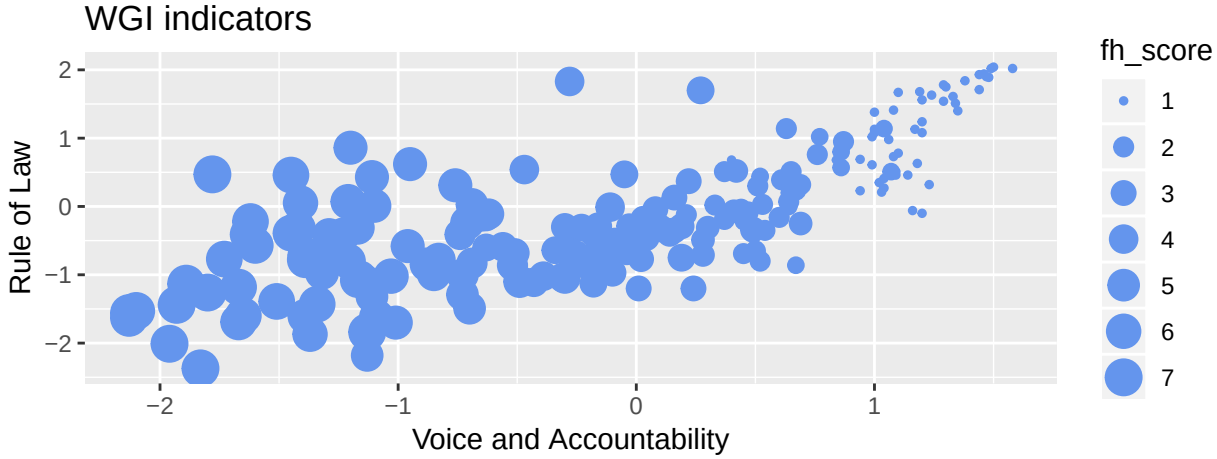


О том, что маркеры для точек можно менять, мы уже знаем (параметр `shape` в `geom_point()`). Попрактикуемся на семинаре. А пока познакомимся с пузырьковой диаграммой (*bubble plot*).

Bubble plot позволяет делать график как будто «объемным» — добавлять дополнительные измерения. На диаграмму рассеяния выше мы можем добавить значение ещё одной переменной, не превращая при этом график в какую-то трёхмерную конструкцию. Каким образом? Сделав размер точек на диаграмме рассеяния зависимым от значений третьей переменной! Более того, можно добавить и четвертое измерение — закрасить точки на графике разным цветом в зависимости от значений ещё одного показателя.

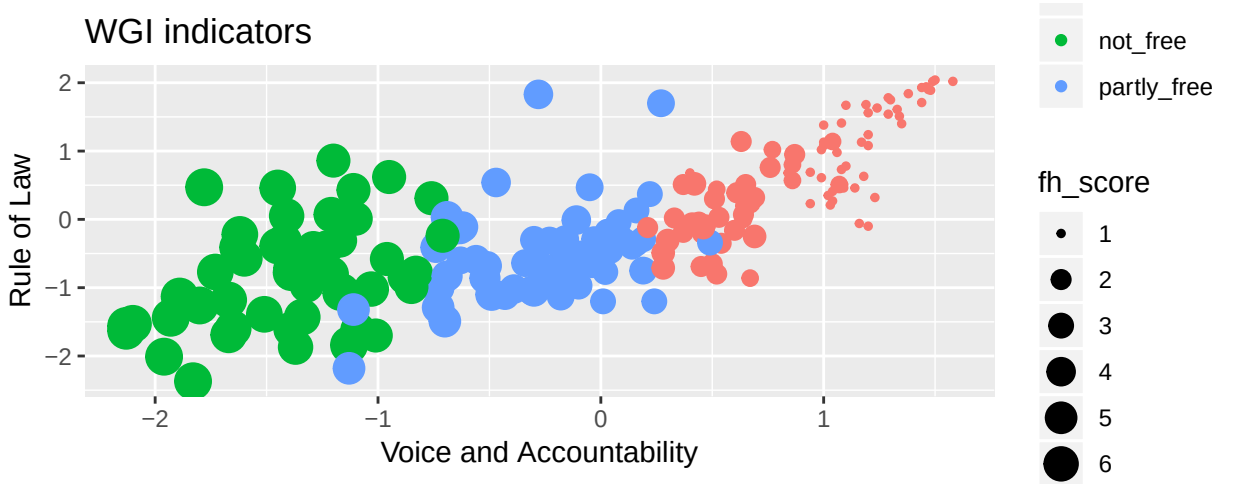
Давайте сейчас сделаем следующее: построим диаграмму рассеяния для индексов *Voice & Accountability* и *Rule of Law*, учитывая при этом значение индекса *Freedon House* в интересующих нас государствах.

```
# тот же aes, но теперь еще и в geom_point
# величина точки зависит от fh_score - внутри aesthetics
# цвет пока у всех точек один, поэтому он задан вне aesthetics

ggplot(data = dat, aes(x = va, y = rl)) +
  geom_point(aes(size = fh_score),
             color = "cornflowerblue") +
  labs(title = "WGI indicators",
       x = "Voice and Accountability",
       y = "Rule of Law")
```



Добавим цвет в зависимости от типа страны:

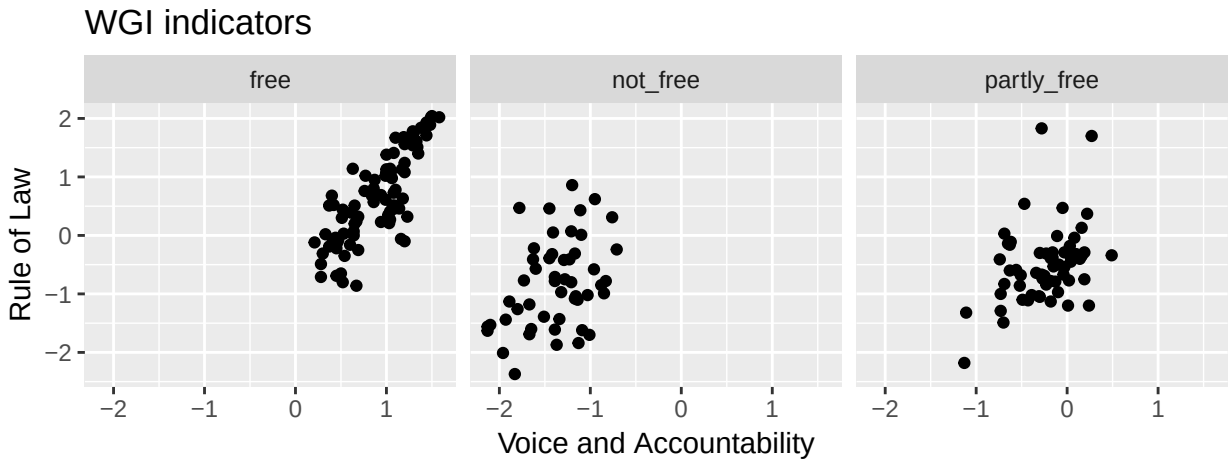
```
ggplot(data = dat, aes(x = va, y = rl)) +
  geom_point(aes(size = fh_score,
                 color = fh_type)) +
  labs(title = "WGI indicators",
       x = "Voice and Accountability",
       y = "Rule of Law")
```



Графики по группам в отдельных ячейках

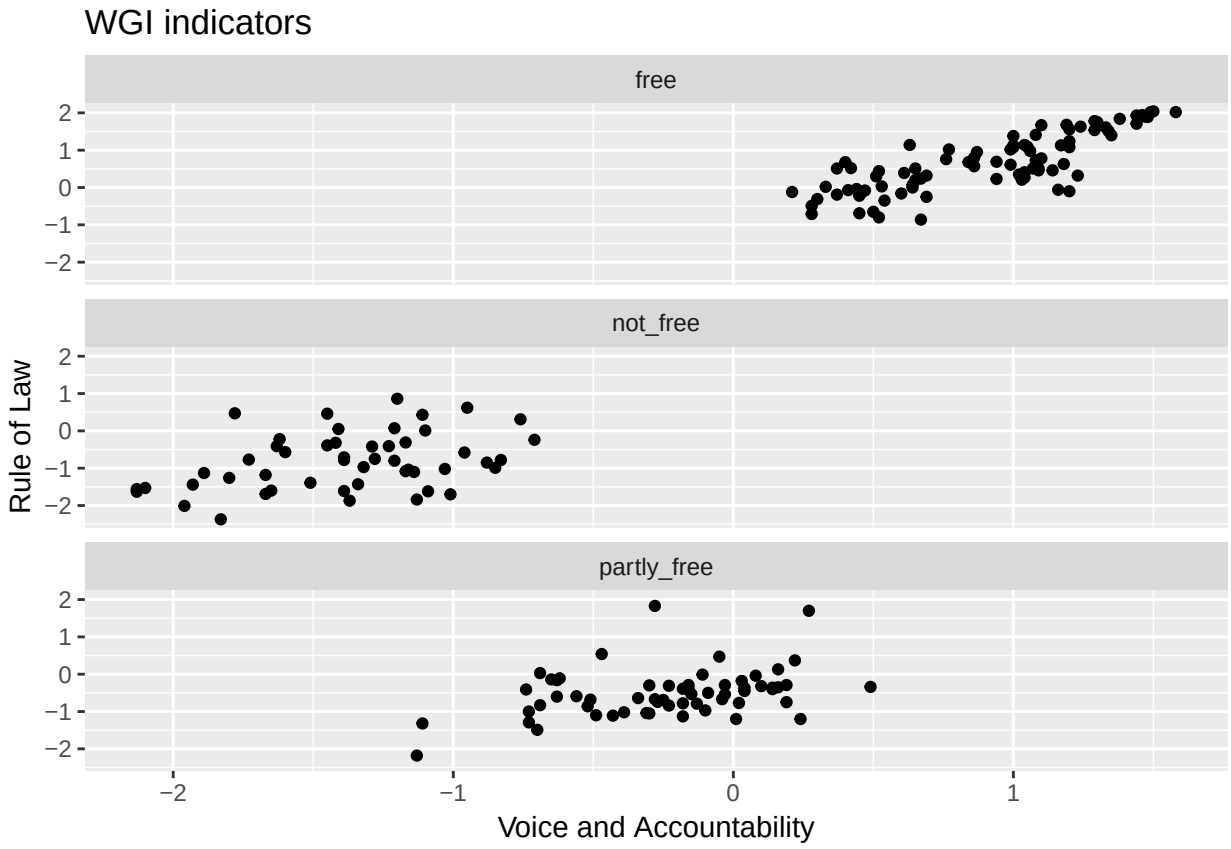
Графики можно строить по группам так, чтобы графики для каждой группы были в отдельной ячейке («фасетке»). Для этого понадобится слой `facet_wrap()`:

```
# в нем через ~ указан показатель, по которому делим на группы

ggplot(data = dat, aes(x = va, y = rl)) +
  geom_point() +
  labs(title = "WGI indicators",
       x = "Voice and Accountability",
       y = "Rule of Law") +
  facet_wrap(~fh_type)
```



С помощью `facet_wrap()` можно расположить графики в таблице с определенным числом строк и столбцов. Например, сделаем три строки и один столбец:

```
ggplot(data = dat, aes(x = va, y = rl)) +
  geom_point() +
  labs(title = "WGI indicators",
       x = "Voice and Accountability",
       y = "Rule of Law") +
  facet_wrap(~fh_type, ncol = 1, nrow = 3)
```



Библиотека `ggflags`

Небольшой бонус — библиотека `ggflags`, с помощью которой вместо точек на график можно наносить флаги государств. См. описание здесь.

Установим ее. Эта библиотека интересна тем, что она устанавливается не из официального «хранилища» (CRAN), а с Github. Установим сначала библиотеку для разработчиков `devtools`, а затем с ее помощью поставим `ggflags`.

```
install.packages("devtools")
```

Если `devtools` не хочет устанавливаться и пишет что-то про `usethis`, можно попробовать перед строчкой выше установить `usethis`, а потом повторить установку `devtools`:

```
install.packages("usethis")
```

Если все установилось, продолжаем установку `ggflags`:

```
library(devtools)
```

```
install_github("rensa/ggflags") # имя пользователя и библиотека
```

Выберем случайным (псевдослучайным) образом 5 строк из нашей базы данных для примера:

```
# set.seed(111) - для воспроизводимости
# чтобы у всех выбрались одинаковые строки
# sample_n() - случайный выбор

set.seed(111)
cnt_sample <- dat %>% sample_n(5)
cnt_sample
```

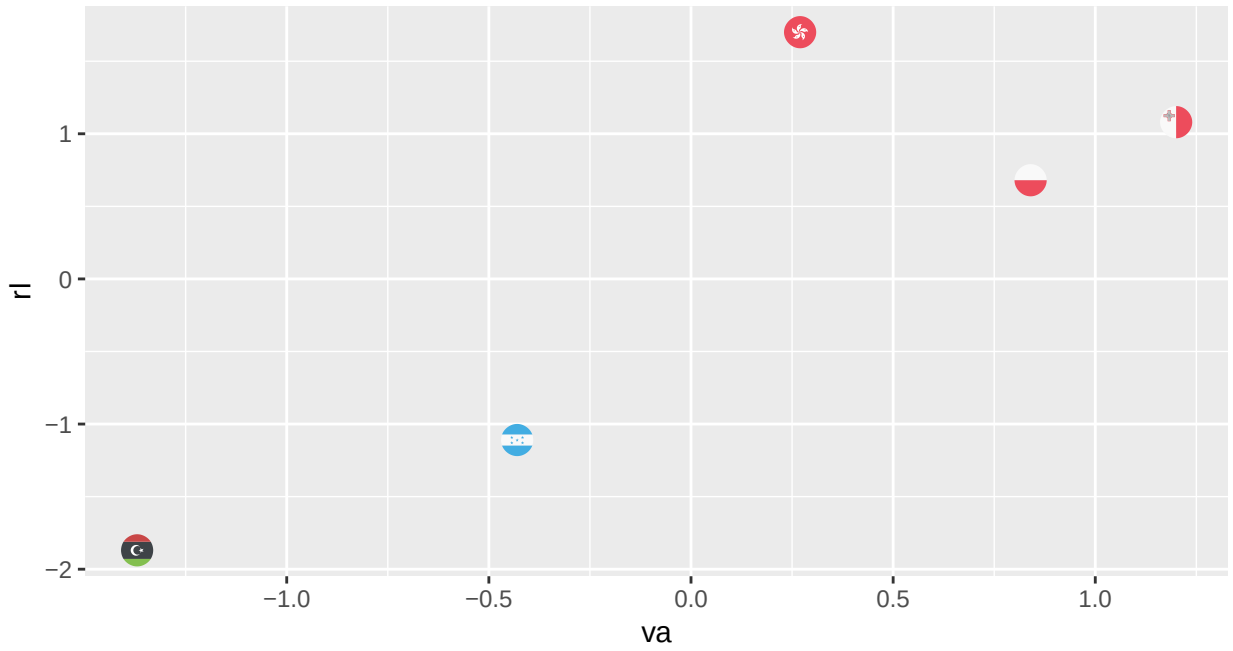
```
##   X.1   X              country cnt_code year    va    ps    ge    rq    rl
## 1 122 129                Malta      MLT 2016  1.20  1.08  0.95  1.16  1.08
## 2 147 156               Poland      POL 2016  0.84  0.51  0.69  0.95  0.68
## 3  77  83 Hong Kong SAR, China      HKG 2016  0.27  0.84  1.86  2.15  1.70
## 4 104 110                Libya      LBY 2016 -1.37 -2.21 -1.89 -2.27 -1.87
## 5  78  84             Honduras      HND 2016 -0.43 -0.36 -0.73 -0.51 -1.11
##      cc fh_score not_free partly_free free     fh_type
## 1  0.72      1.0        0           0    1        free
## 2  0.75      1.0        0           0    1        free
## 3  1.58      3.5        0           1    0 partly_free
## 4 -1.57      6.0        1           0    0    not_free
## 5 -0.69      4.0        0           1    0 partly_free
```

Нам нужны флаги следующих стран: Malta, Poland, Hong Kong, Libya, Honduras. Создадим для этих стран метки в соответствии с метками стран в документации библиотеки:

```
library(ggflags)

# коды стран для флагов
cnt_sample$codes = c("mt", "pl", "hk", "ly", "hn")
```

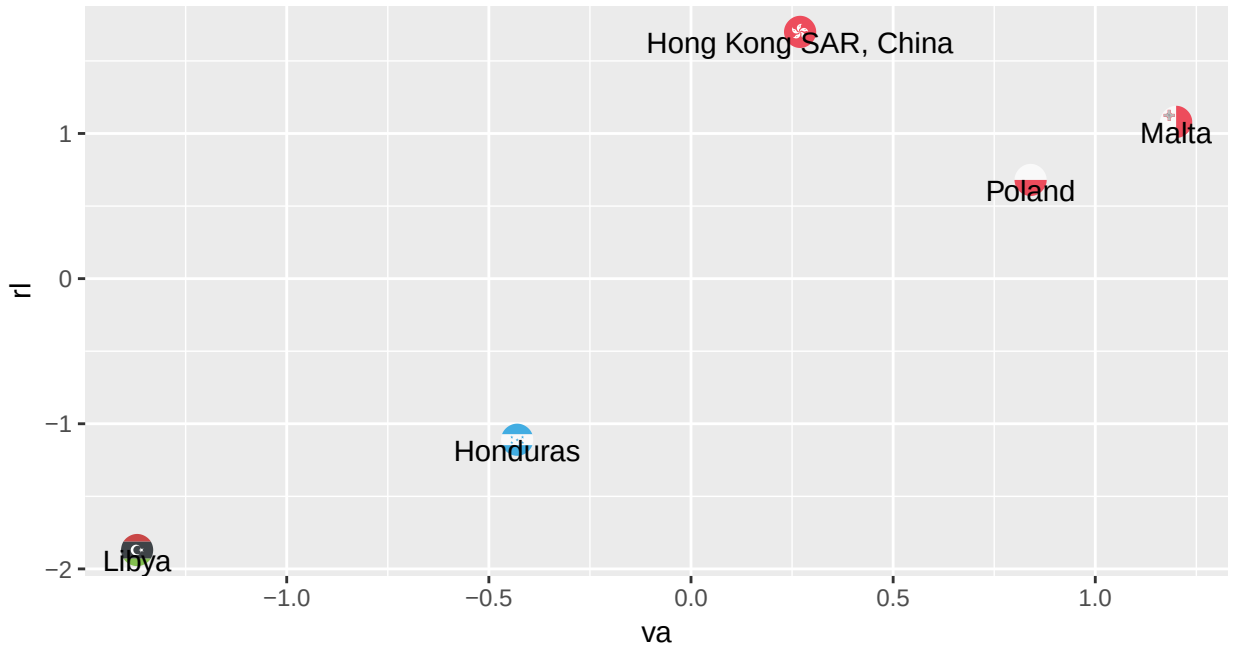
Создадим график:

```
ggplot(data = cnt_sample, aes(x = va, y = rl,
              country = codes)) +
       geom_flag()
```



Заодно добавим названия стран:

```
# label в aes()
# слой geom_text() для подписей

ggplot(data = cnt_sample,
       aes(x = va, y = rl,
           country = codes,
           label = country)) +
  geom_flag() +
  geom_text(hjust=0.5, vjust=1)
```



В коде выше `hjust` и `vjust` нужны для того, чтобы подписи к точкам были немного сдвинуты относи-

тельно самой точки (флага) во избежание наложения текста на флаг.

P.S. Я не знаю, насколько часто обновляется эта библиотека и насколько актуальны флаги государств, используемые в ней.