RUDN university

5100

Программирование в среде R

Шевцов Василий Викторович,
директор ДИТ РУДН, shevtsov_vv@rudn.university

Изменение компонентов

Новые темы

```
> print(theme_minimal)
function (base_size = 11, base_family = "", base_line_size = base_size/22,
    base_rect_size = base_size/22)
{
    theme_bw(base_size = base_size, base_family = base_family,
        base_line_size = base_line_size, base_rect_size = base_rect_size) %+replace%
        theme(axis.ticks = element_blank(), legend.background = element_blank(),
            legend.key = element_blank(), panel.background = element_blank(),
            panel.border = element_blank(), strip.background = element_blank(),
            plot.background = element_blank(), complete = TRUE)
}
<bytecode: 0x00000041cbd4c3b0>
<environment: namespace:ggplot2>
```

Новые темы

```
> print(theme_bw)
function (base_size = 11, base_family = "", base_line_size = base_size/22,
    base_rect_size = base_size/22)
{
    theme_grey(base_size = base_size, base_family = base_family,
        base_line_size = base_line_size, base_rect_size = base_rect_size) %+replace%
        theme(panel.background = element_rect(fill = "white",
            colour = NA), panel.border = element_rect(fill = NA,
            colour = "grey20"), panel.grid = element_line(colour = "grey92"),
            panel.grid.minor = element_line(size = rel(0.5)),
            strip.background = element_rect(fill = "grey85",
                colour = "grey20"), legend.key = element_rect(fill = "white",
                colour = NA), complete = TRUE)
}
<bytecode: 0x00000041cbd33370>
<environment: namespace:ggplot2>
```

Новые темы

> print(theme_grey)

```
function (base_size = 11, base_family = "", base_line_size = base_size/22,
    base_rect_size = base_size/22)
{
    half_line <- base_size/2
    t <- theme(line = element_line(colour = "black", size = base_line_size,
        linetype = 1, lineend = "butt"), rect = element_rect(fill = "white",
        colour = "black", size = base_rect_size, linetype = 1),
        text = element_text(family = base_family, face = "plain",
            colour = "black", size = base_size, lineheight = 0.9,
            hjust = 0.5, vjust = 0.5, angle = 0, margin = margin(),
            debug = FALSE), axis.line = element_blank(), axis.line.x = NULL,
        axis.line.y = NULL, axis.text = element_text(size = rel(0.8),
            colour = "grey30"), axis.text.x = element_text(margin = margin(t = 0.8 *
            half_line/2), vjust = 1), axis.text.x.top = element_text(margin = margin(b = 0.8 *
            half_line/2), vjust = 0), axis.text.y = element_text(margin = margin(r = 0.8 *
            half_line/2), hjust = 1), axis.text.y.right = element_text(margin = margin(l = 0.8 *
            half_line/2), hjust = 0), axis.ticks = element_line(colour = "grey20"),
        axis.ticks.length = unit(half_line/2, "pt"), axis.ticks.length.x = NULL,
        axis.ticks.length.x.top = NULL, axis.ticks.length.x.bottom = NULL,
        axis.ticks.length.y = NULL, axis.ticks.length.y.left = NULL,
        axis.ticks.length.y.right = NULL, axis.title.x = element_text(margin = margin(t = half_line/2),
            vjust = 1), axis.title.x.top = element_text(margin = margin(b = half_line/2),
            vjust = 0), axis.title.y = element_text(angle = 90,
            margin = margin(r = half_line/2), vjust = 1), axis.title.y.right = element_text(angle = -90,
            margin = margin(l = half_line/2), vjust = 0), legend.background = element_rect(colour = NA),
        legend.spacing = unit(2 * half_line, "pt"), legend.spacing.x = NULL,
        legend.spacing.y = NULL, legend.margin = margin(half_line,
            half_line, half_line, half_line), legend.key = element_rect(fill = "grey95",
            colour = NA), legend.key.size = unit(1.2, "lines"),
        legend.key.height = NULL, legend.key.width = NULL, legend.text = element_text(size = rel(0.8)),
        legend.text.align = NULL, legend.title = element_text(hjust = 0),
        legend.title.align = NULL, legend.position = "right",
        legend.direction = NULL, legend.justification = "center",
        legend.box = NULL, legend.box.margin = margin(0, 0, 0,
            0, "cm"), legend.box.background = element_blank(),
        legend.box.spacing = unit(2 * half_line, "pt"),
        panel.background = element_rect(fill = "grey92",
            colour = NA), panel.border = element_blank(), panel.grid = element_line(colour = "white"),
        panel.grid.minor = element_line(size = rel(0.5)), panel.spacing = unit(half_line,
            "pt"), panel.spacing.x = NULL, panel.spacing.y = NULL,
        panel.ontop = FALSE, strip.background = element_rect(fill = "grey85",
            colour = NA), strip.text = element_text(colour = "grey10",
            size = rel(0.8), margin = margin(0.8 * half_line,
                0.8 * half_line, 0.8 * half_line, 0.8 * half_line)),
        strip.text.x = NULL, strip.text.y = element_text(angle = -90),
        strip.text.y.left = element_text(angle = 90), strip.placement = "inside",
        strip.placement.x = NULL, strip.placement.y = NULL, strip.switch.pad.grid = unit(half_line/2,
            "pt"), strip.switch.pad.wrap = unit(half_line/2,
            "pt"), plot.background = element_rect(colour = "white"),
        plot.title = element_text(size = rel(1.2), hjust = 0,
            vjust = 1, margin = margin(b = half_line)), plot.title.position = "panel",
        plot.subtitle = element_text(hjust = 0, vjust = 1, margin = margin(b = half_line)),
        plot.caption = element_text(size = rel(0.8), hjust = 1,
            vjust = 1, margin = margin(t = half_line)), plot.caption.position = "panel",
        plot.tag = element_text(size = rel(1.2), hjust = 0.5,
            vjust = 0.5), plot.tag.position = "topleft",
        plot.margin = margin(half_line, half_line, half_line,
            half_line), complete = TRUE)
    ggplot_global$theme_all_null %+replace% t
}
<bytecode: 0x00000041dc5b0428>
<environment: namespace:ggplot2>
```

Новые темы

> print(theme_void)

```
function (base_size = 11, base_family = "", base_line_size = base_size/22,
    base_rect_size = base_size/22)
{
    half_line <- base_size/2
    t <- theme(line = element_blank(), rect = element_blank(),
        text = element_text(family = base_family, face = "plain",
            colour = "black", size = base_size, lineheight = 0.9,
            hjust = 0.5, vjust = 0.5, angle = 0, margin = margin(),
            debug = FALSE), axis.text = element_blank(), axis.title = element_blank(),
        axis.ticks.length = unit(0, "pt"), axis.ticks.length.x = NULL,
        axis.ticks.length.x.top = NULL, axis.ticks.length.x.bottom = NULL,
        axis.ticks.length.y = NULL, axis.ticks.length.y.left = NULL,
        axis.ticks.length.y.right = NULL, legend.box = NULL,
        legend.key.size = unit(1.2, "lines"), legend.position = "right",
        legend.text = element_text(size = rel(0.8)), legend.title = element_text(hjust = 0),
        strip.text = element_text(size = rel(0.8)), strip.switch.pad.grid = unit(half_line/2,
            "pt"), strip.switch.pad.wrap = unit(half_line/2,
            "pt"), panel.ontop = FALSE, panel.spacing = unit(half_line,
            "pt"), plot.margin = unit(c(0, 0, 0, 0), "lines"),
        plot.title = element_text(size = rel(1.2), hjust = 0,
            vjust = 1, margin = margin(t = half_line)), plot.title.position = "panel",
        plot.subtitle = element_text(hjust = 0, vjust = 1, margin = margin(t = half_line)),
        plot.caption = element_text(size = rel(0.8), hjust = 1,
            vjust = 1, margin = margin(t = half_line)), plot.caption.position = "panel",
        plot.tag = element_text(size = rel(1.2), hjust = 0.5,
            vjust = 0.5), plot.tag.position = "topleft",
        complete = TRUE)
    ggplot_global$theme_all_null %+replace% t
}
<bytecode: 0x00000041cbd5f338>
<environment: namespace:ggplot2>
```

Новые статистики

В то время как большинство людей склонны думать о geom как о главном графическом слое, добавляемом к участкам, разнообразие geom в основном зависит от различных статистических данных. Из этого следует, что расширение статистики является одним из наиболее полезных способов расширения возможностей ggplot2. Одно из преимуществ статистики заключается в том, что она предназначена исключительно для преобразования данных, к чему привыкло большинство пользователей R. До тех пор, пока необходимое поведение может быть инкапсулировано в stat, нет необходимости возиться с любыми вызовами grid.

Основная логика stat инкапсулирована в многоуровневой последовательности вызовов: compute_layer(), compute_panel(), and compute_group(). Поведение по умолчанию compute_layer() состоит в том, чтобы разделить данные по PANEL столбцу, вызвать compute_panel() и повторно собрать результаты. Аналогично, поведение по умолчанию compute_panel() заключается в том, чтобы разделить данные панели по group столбцу, вызвать compute_group() и повторно собрать результаты. Таким образом, необходимо только определить compute_group(), т. е. как должна быть преобразована одна группа, чтобы иметь рабочий stat. Существует множество примеров перезаписи compute_panel() чтобы получить лучшую производительность, поскольку это позволяет векторизировать вычисления и избежать дорогостоящего шага разделения-объединения, но в целом часто бывает полезно начать на compute_group() уровне и посмотреть, является ли производительность адекватной.

Новые статистики

Кроме compute_*() функций, остальная логика находится в функциях setup_params() и setup_data(). Они вызываются перед compute_*() функциями и позволяют Stat реагировать и изменять себя в ответ на заданные параметры и данные (особенно данные, так как это недоступно при построении stat). Функция setup_params() получает параметры, заданные при построении вместе с данными слоя, и возвращает измененный список параметров. Параметры должны соответствовать именам аргументов в compute_*() функциях, чтобы быть доступными. setup_data() функция получает измененные параметры вместе с данными слоя и возвращает измененные данные слоя. Важно, чтобы независимо от того, какие изменения происходят в setup_data() PANEL и group колонках и остаются неизменными. Иногда, с соответствующими статистиками, все, что необходимо, - это создать подкласс и предоставить новые setup_params()/ setup_data() методы.

При создании новой статистики часто рекомендуется предоставить сопутствующий geom_*() конструктор, поскольку большинство пользователей привыкли использовать эти, а не stat_*() конструкторы. Отклонения от этого правила могут быть сделаны, если нет очевидного значения geom по умолчанию для нового stat, или если stat предназначен для предложения небольшой модификации существующей пары geom+stat.

Новые geoms

В то время как многие вещи могут быть достигнуты путем создания новой статистики, есть ситуации, когда необходимо создать новый geom.

Некоторые из них

- Не имеет смысла возвращать данные из stat в форме, понятной для любых текущих geom.
- Слой должен объединить выходные данные нескольких geom.
- geom должен вернуть grobs, которые в настоящее время не доступны из существующих geom.

Создание новых geom может показаться немного более сложным, чем создание новой статистики, так как конечный результат-это коллекция grobs, а не измененные данные.

Новые geoms

Основная функциональность geoms - это, как и для статистики, многоуровневая последовательность вызовов: draw_layer(),draw_panel(), и draw_group(), и она следует во многом той же логике, что и для статистики.
Обычно легче реализовать draw_group() метод, но если ожидается, что слой будет обрабатывать большое количество различных групп, вы должны рассмотреть, можно ли использовать draw_panel() вместо этого, поскольку производительность сетки будет испытывать трудности при работе со многими отдельными grobs (например, 10 000 pointGrobs с одной точкой каждый против одного pointGrob с 10 000 точками).
В соответствии со статистикой, GEOM также имеют пару setup_params()+setup_data(), которые функционируют примерно так же. Но есть одна заметка: setup_data() вызывается до выполнения позиционирования положения объекта в рамках этапа сборки.

Новые geoms

Если вы хотите создать новый geom, который является версией существующего geom, но с различными входными ожиданиями, его обычно можно обработать путем перезаписи setup_data() метода существующего geom. Этот подход можно увидеть с geom_spoke() помощью которого является простой перепараметризацией geom_segment():

```
print(GeomSpoke$setup_data)
#> <ggproto method>
#>   <Wrapper function>
#>     function (...)
#> f(...)
#>
#>   <Inner function (f)>
#>     function (data, params)
#> {
#>     data$radius <- data$radius %||% params$radius
#>     data$angle <- data$angle %||% params$angle
#>     transform(data, xend = x + cos(angle) * radius, yend = y +
#>         sin(angle) * radius)
#> }
```

Новые geoms

Если вы хотите объединить функциональные возможности нескольких geom, это обычно может быть достигнуто путем подготовки данных для каждого из geom внутри draw_*() вызова и отправьте его в разные geoms, собирая выходные данные в a gList(список grobs), если вызов есть draw_group(), или AgTree(grob, содержащий несколько дочерних grobs), если вызов есть draw_panel(). Пример этого можно увидеть в geom_smooth() котором совмещает geom_line() и geom_ribbon():

```
print(GeomSmooth$draw_group)
#> <ggproto method>
#>   <Wrapper function>
#>     function (...)
#> f(...)
#>
#>   <Inner function (f)>
#>     function (data, panel_params, coord, se = FALSE, flipped_aes = FALSE)
#> {
#>     ribbon <- transform(data, colour = NA)
#>     path <- transform(data, alpha = NA)
#>     ymin = flipped_names(flipped_aes)$ymin
#>     ymax = flipped_names(flipped_aes)$ymax
#>     has_ribbon <- se && !is.null(data[[ymax]]) && !is.null(data[[ymin]])
#>     gList(if (has_ribbon)
#>         GeomRibbon$draw_group(ribbon, panel_params, coord, flipped_aes = flipped_aes),
#>         GeomLine$draw_panel(path, panel_params, coord))
#> }
```

Новые координаты

В своем самом основном виде coord отвечает за масштабирование эстетики положения в диапазон [0, 1], потенциально преобразуя их в этом процессе. Единственное место, где можно вызвать любые методы из coord, находится в методе geoms draw_*() transform(), где метод вызывается для данных, чтобы превратить данные позиции в правильный формат перед созданием grobs из него. Наиболее распространенным (и по умолчанию) является CoordCartesian, который просто масштабирует данные о местоположении:

```
> print(CoordCartesian$transform)
<ggproto method>
  <Wrapper function>
    function (...)
f(...)

  <Inner function (f)>
    function (data, panel_params)
{
    data <- transform_position(data, panel_params$x$rescale,
        panel_params$y$rescale)
    transform_position(data, squish_infinite, squish_infinite)
}
```

Новые координаты

Помимо этого у coords есть много возможностей, которые разработчики расширений могут использовать, но которые, вероятно, лучше оставить в покое.

Coords заботится о визуализации осей, меток осей и панелей переднего плана и фона, а также может перехватывать данные слоя и компоновку фасета и изменять их.

Тем не менее, с введением coord_sf() новой системы координат существует небольшая потребность в новых системах координат, поскольку большинство не связанных с картографией методов использования захватываются существующими системами координат и coord_sf() поддерживают все различные проекции, необходимые в картографии

Новые шкалы

Есть три способа, которыми можно было бы расширить ggplot2 с помощью новых шкал. Самый простой - это случай, когда вы хотите предоставить удобную оболочку для новой палитры к существующей шкале (это часто означает новую палитру цвета/заливки). В этом случае будет достаточно предоставить новый конструктор шкалы, который передает соответствующую палитру в соответствующий базовый конструктор шкалы. Это используется в ggplot2 себя как, например, в шкале viridis :

```
> print(scale_fill_viridis_c)
function (..., alpha = 1, begin = 0, end = 1, direction = 1,
    option = "D", values = NULL, space = "Lab", na.value = "grey50",
    guide = "colourbar", aesthetics = "fill")
{
    continuous_scale(aesthetics, "viridis_c", gradient_n_pal(viridis_pal(alpha,
        begin, end, direction, option)(6), values, space), na.value = na.value,
        guide = guide, ...)
}
<bytecode: 0x00000041cb8fe860>
<environment: namespace:ggplot2>
```

Новые шкалы

Другой относительно простой случай - это когда вы предоставляете geom, который принимает новый тип эстетики, который должен быть масштабирован. Предположим, что вы создали новую линию geom, и вместо эстетики размера вы решили использовать эстетику ширины. Чтобы получить масштабирование ширины таким же образом, как вы привыкли ожидать масштабирования размера, вы должны предоставить шкалу по умолчанию для эстетики. Шкалы по умолчанию определяются на основе их названия и типа данных, предоставленных пользователю. Если вы назначаете непрерывные значения ширине, то ggplot2 будет искать функцию scale_width_continuous() и использовать ее, если на графике не было добавлено никакого другого масштаба ширины. Если такая функция не найдена (и масштаб ширины не был добавлен явно), то эстетика не будет масштабироваться.

Новые шкалы

Последняя возможность - это возможность создания нового первичного типа шкалы.
ggplot2 исторически имел два основных масштаба: непрерывный и дискретный. Недавно к ним присоединился тип биннированной шкалы, который позволяет разбивать непрерывные данные на дискретные ячейки.

Можно развить дальнейшие первичные масштабы, следуя примеру ScaleBinned.

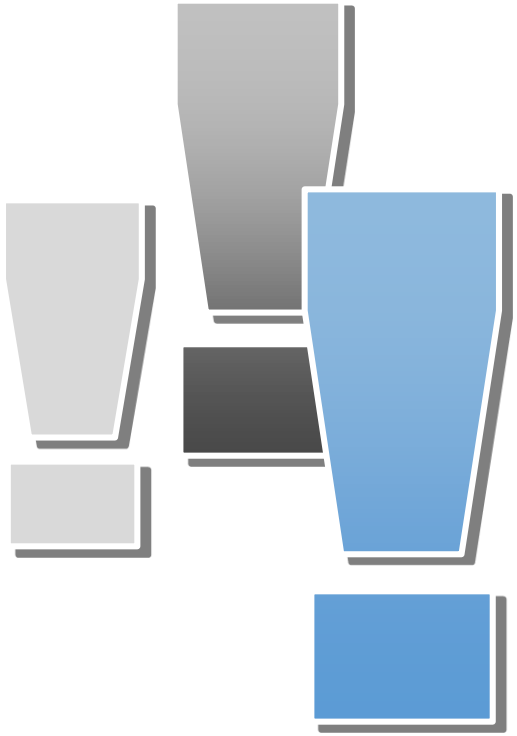
Для этого требуется масштабирование подклассов или один из предоставленных первичных масштабов, а также создание новых методов train() и map (), в частности.

Новые фасеты

Фасеты являются одним из самых мощных инструментов в ggplot2, и из этого следует, что расширение фасетов является одним из самых мощных способов изменить способ работы ggplot2.
Фасеты отвечают за получение всех различных панелей, прикрепление осей (и полос) к ним и размещение их ожидаемым образом. Все это требует большого количества манипуляций gtable и понимания сетки разметки.

Если ваш новый фасет закончится прямоугольным расположением панелей, то часто можно подклассировать FacetWrap либо FacetGrid, и просто предоставить новые compute_layout(), и map_data() методы.
Первый занимается получением исходных данных и созданием спецификации макета, в то время как второй получает созданный макет вместе с данными и присоединяет к нему PANEL, сопоставляя данные с одной из панелей в макете.

Спасибо за внимание!



Шевцов Василий Викторович

shevtsov_vv@rudn.university
+7(903)144-53-57