RUDN university

5100

Программирование в среде R

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

Темы

Темы

theme()	Modify components of a theme
theme_grey() theme_gray() theme_bw() theme_linedraw() theme_light() theme_dark() theme_minimal() theme_classic() theme_void() theme_test()	Complete themes
theme_get() theme_set() theme_update() theme_replace() `%+replace%`	Get, set, and modify the active theme
margin() element_blank() element_rect() element_line() element_text() rel()	Theme elements

Компоненты

Система теминга состоит из четырех основных компонентов:

elements указывают элементы, которыми можно управлять. Например, элемент plot.title управляет внешним видом объекта название; axis.ticks.x метки на оси x; legend.key.height высота ключей в легенде.

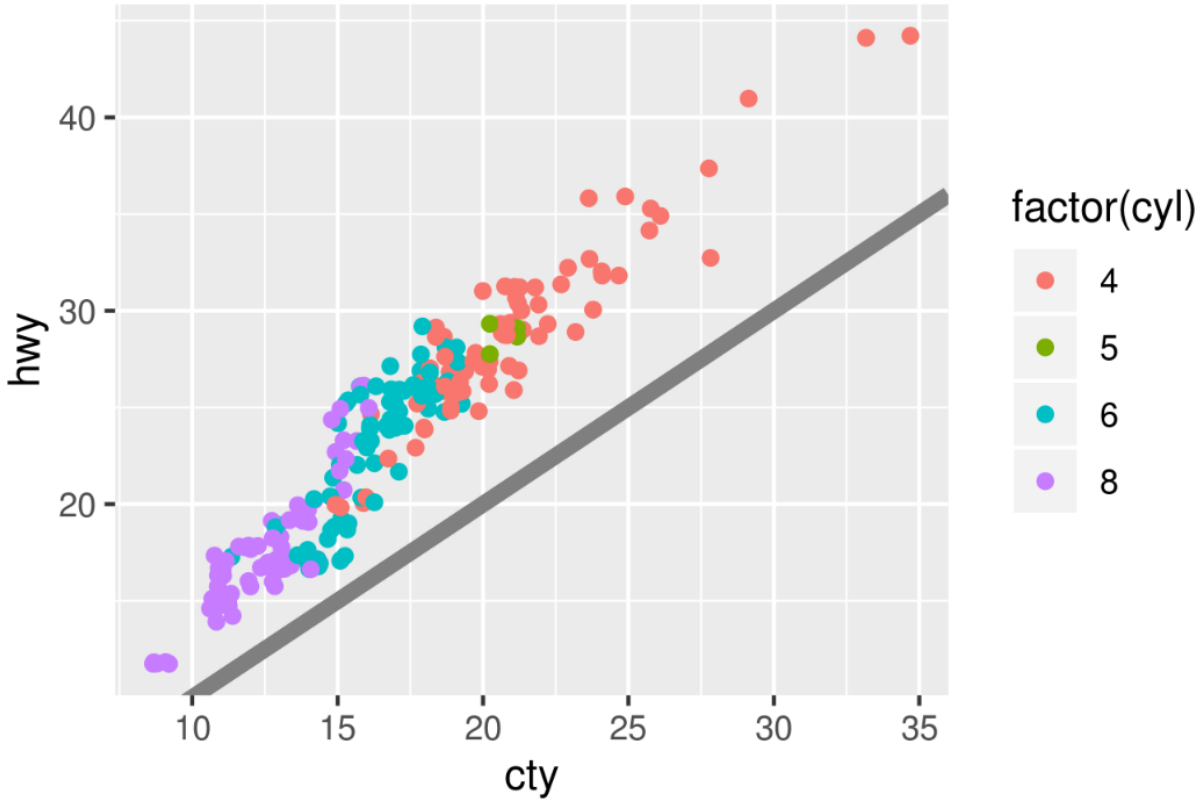
Каждый элемент связан с функцией элемента **element function**, которая описывает визуальные свойства элемента. Например, element_text() управляет размером шрифта, цветом текстовых элементов, таких как plot.title.

theme() функция, которая позволяет переопределить тему по умолчанию элементы путем вызова функций элементов, таких как theme(plot.title = element_text(colour = "red")).

Complete **themes**, например theme_grey() завершает редактирование и записывает все измеения.

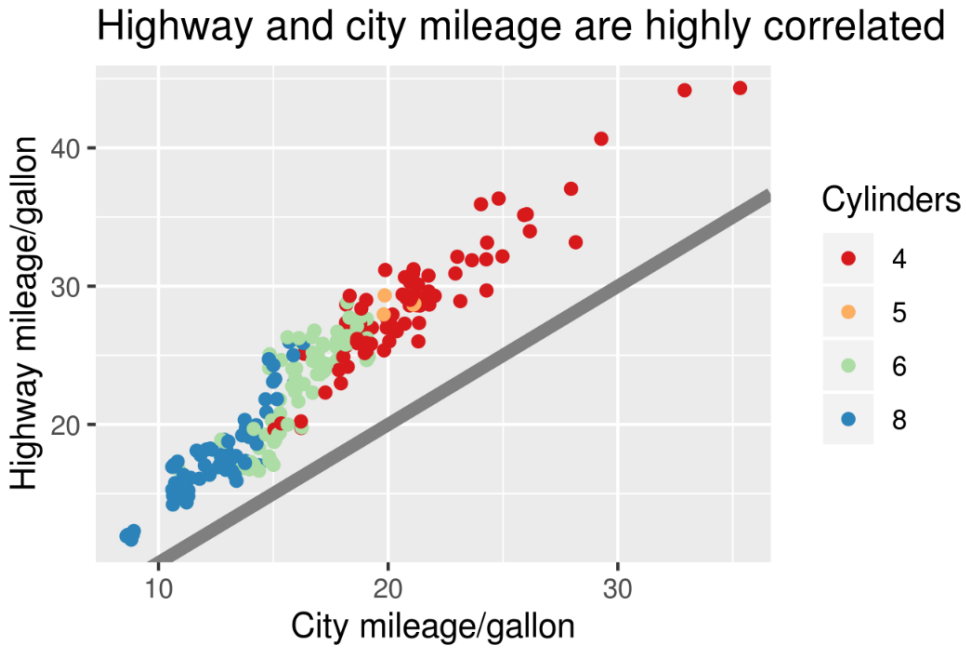
Базовый вариант

```
base <- ggplot(mpg, aes(cty, hwy, color = factor(cyl))) +
  geom_jitter() +
  geom_abline(colour = "grey50", size = 2)
base
```



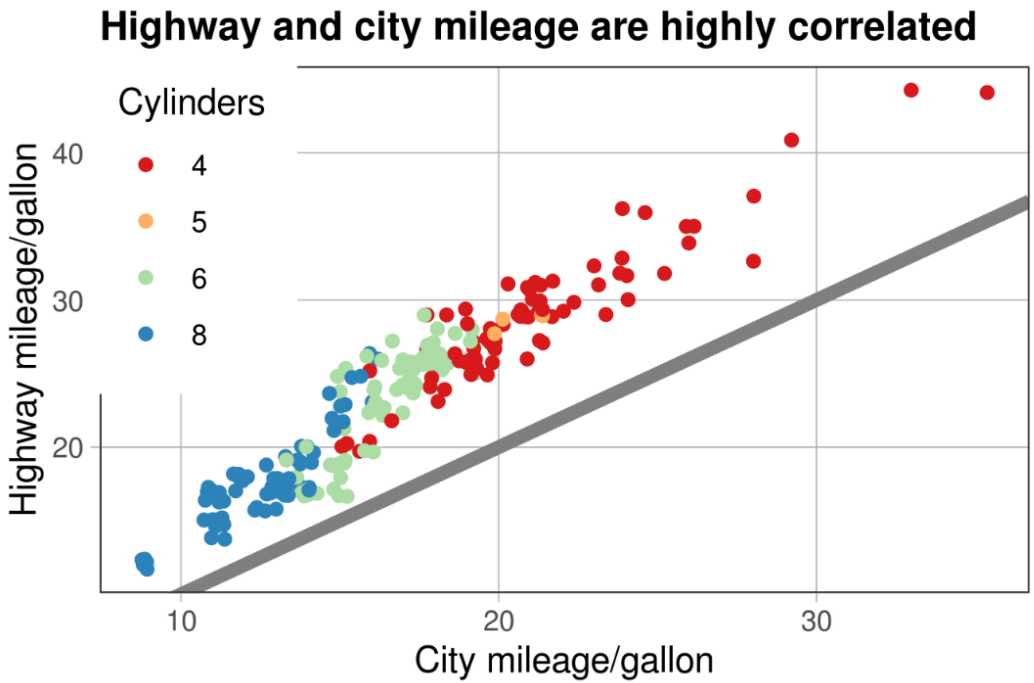
Улучшение 1

```
labelled <- base +
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  ) +
  scale_colour_brewer(type = "seq", palette = "Spectral")
labelled
```



Улучшение 2

```
styled <- labelled +
  theme_bw() +
  theme(
    plot.title = element_text(face = "bold", size = 12),
    legend.background = element_rect(fill = "white", size = 4, colour = "white"),
    legend.justification = c(0, 1),
    legend.position = c(0, 1),
    axis.ticks = element_line(colour = "grey70", size = 0.2),
    panel.grid.major = element_line(colour = "grey70", size = 0.2),
    panel.grid.minor = element_blank()
  )
styled
```



Встроенные темы

ggplot2 поставляется с рядом встроенных тем. Самая главная - это theme_grey() фирменная тема ggplot2 со светло-серым фоном и белыми линиями сетки. Эта тема предназначена для представления данных при поддержке сопоставлений, следуя стандартным рекомендациям.

Серый фон придает сюжету схожий типографский цвет с текстом, гарантируя, что графика вписывается в документ без противоречия с ярким белым фоном.

Наконец, серый фон создает непрерывное цветовое поле, которое гарантирует, что сюжет воспринимается как единый визуальный объект.

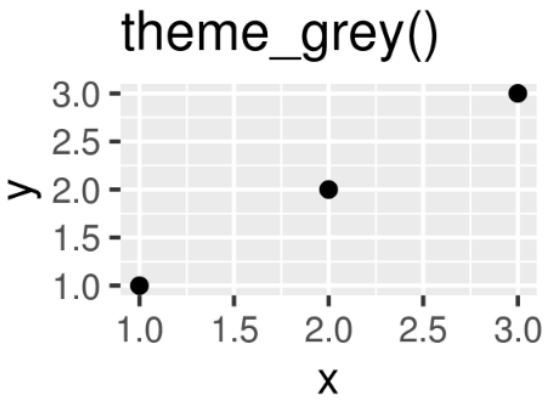
Встроенные темы

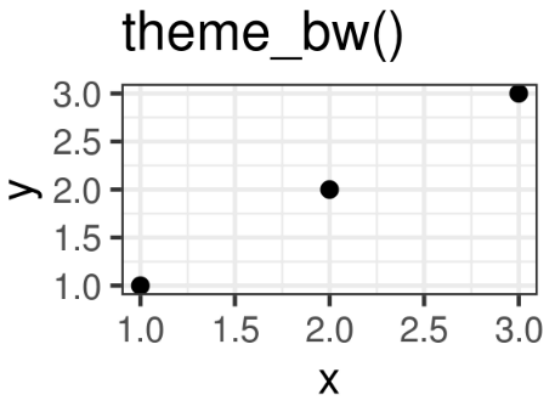
Есть семь других тем, встроенных в ggplot2 1.1.0:

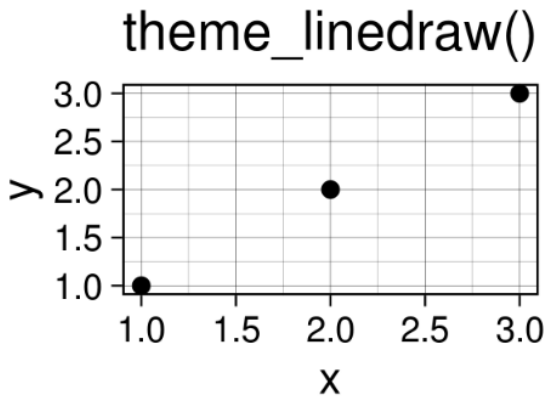
- theme_bw(): вариант, в theme_grey() котором используется белый фон и тонкие серые линии сетки.
- theme_linedraw(): Тема с только черными линиями различной ширины на белом фоне фоны, напоминающие линейный рисунок.
- theme_light(): похоже на theme_linedraw() светло-серые линии и оси, чтобы направить больше внимания на данные.
- theme_dark(): темная theme_light(), с аналогичными линейными размерами но на темном фоне.
- theme_minimal(): Минималистичная тема без каких-либо фоновых аннотаций.
- theme_classic(): Классическая тема, с линиями оси x и y, нет сетки.
- theme_void(): Совершенно пустая тема.

Встроенные темы

```
df <- data.frame(x = 1:3, y = 1:3)
base <- ggplot(df, aes(x, y)) + geom_point()
base + theme_grey() + ggtitle("theme_grey()")
base + theme_bw() + ggtitle("theme_bw()")
base + theme_linedraw() + ggtitle("theme_linedraw()")
```

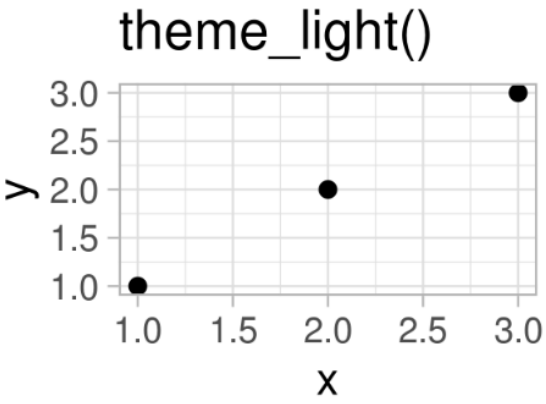


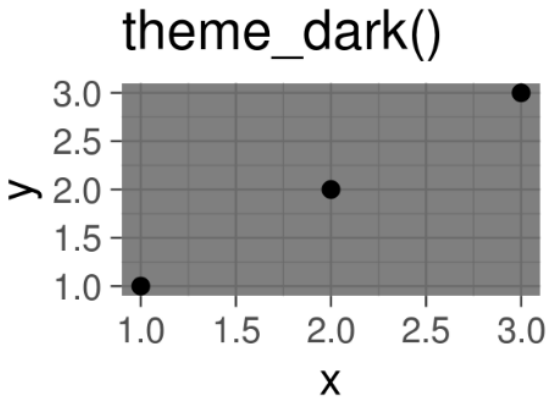


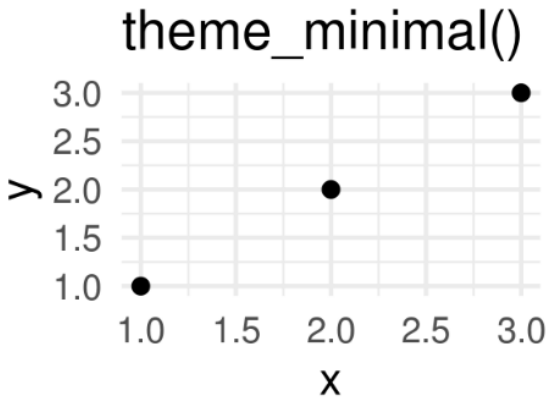


Встроенные темы

```
base + theme_light() + ggtitle("theme_light()")
base + theme_dark() + ggtitle("theme_dark()")
base + theme_minimal()  + ggtitle("theme_minimal()")
```

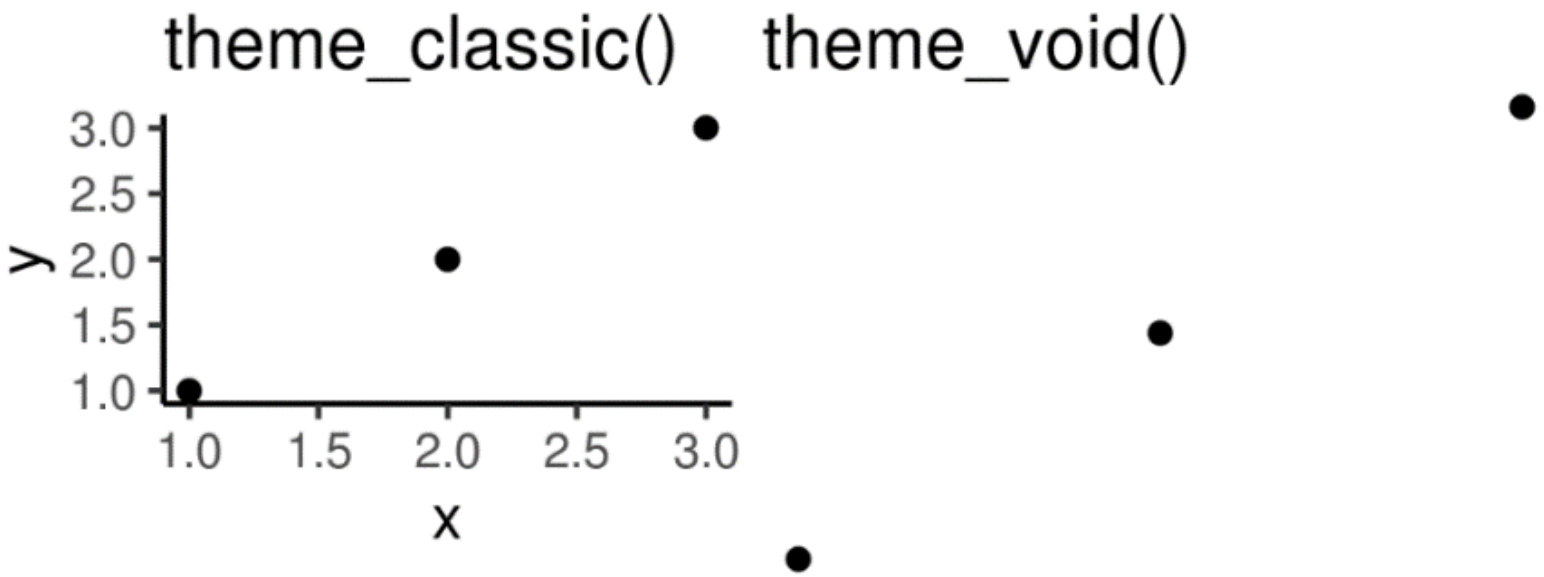






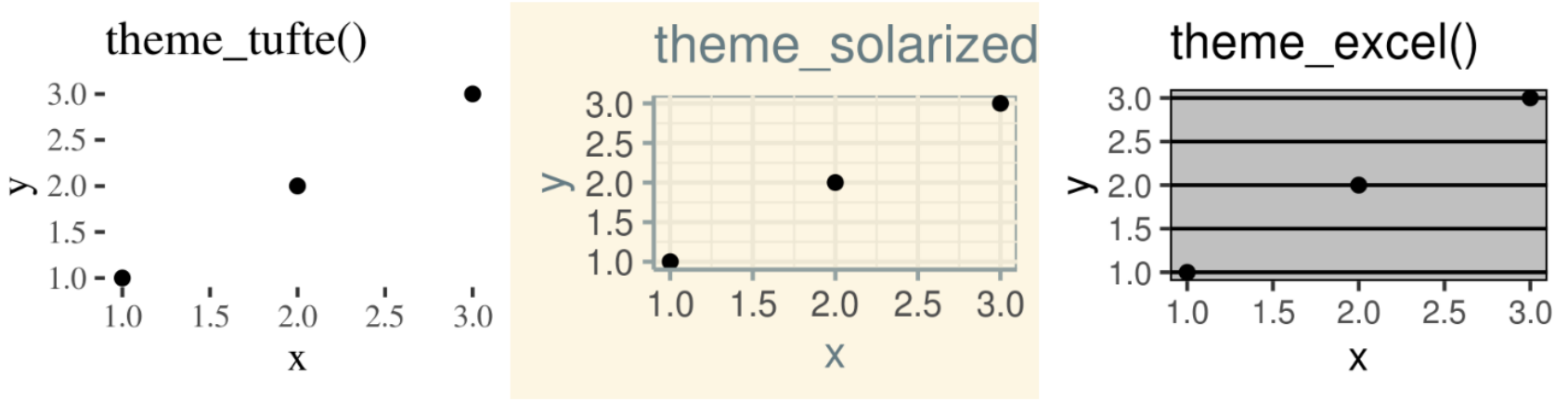
Встроенные темы

base + theme_classic() + ggtitle("theme_classic()")
base + theme_void() + ggtitle("theme_void()")



Дополнительные темы

```
install.packages("ggthemes")
library(ggthemes)
base + theme_tufte() + ggtitle("theme_tufte()")
base + theme_solarized() + ggtitle("theme_solarized()")
base + theme_excel() + ggtitle("theme_excel()") # ;)
```



Изменение темы

Для изменения отдельного компонента темы используется код типа plot + theme(element.name = element_function()).

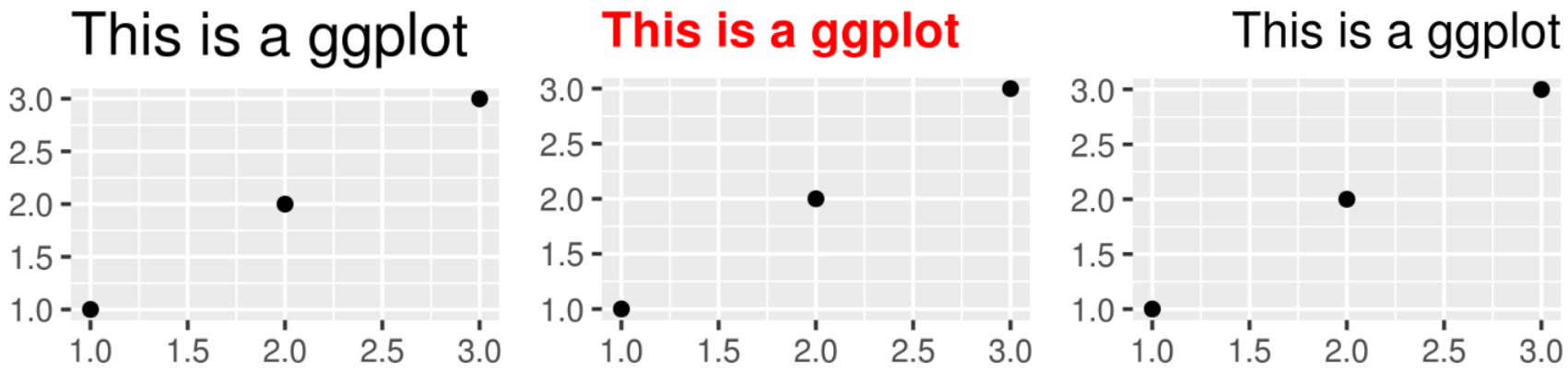
Существует четыре основных типа встроенных функций элементов: **text, lines, rectangles, blank**. Каждая функция элемента имеет набор параметров, управляющих внешним видом:

element_text() рисует этикетки и заголовки.
управлление шрифтом family, face, colour, size(в пунктах),hjust, vjust,angle (в градусах) и lineheight(как отношениefontcase).

Aesthetic specifications - vignette("ggplot2-specs")

Изменение темы

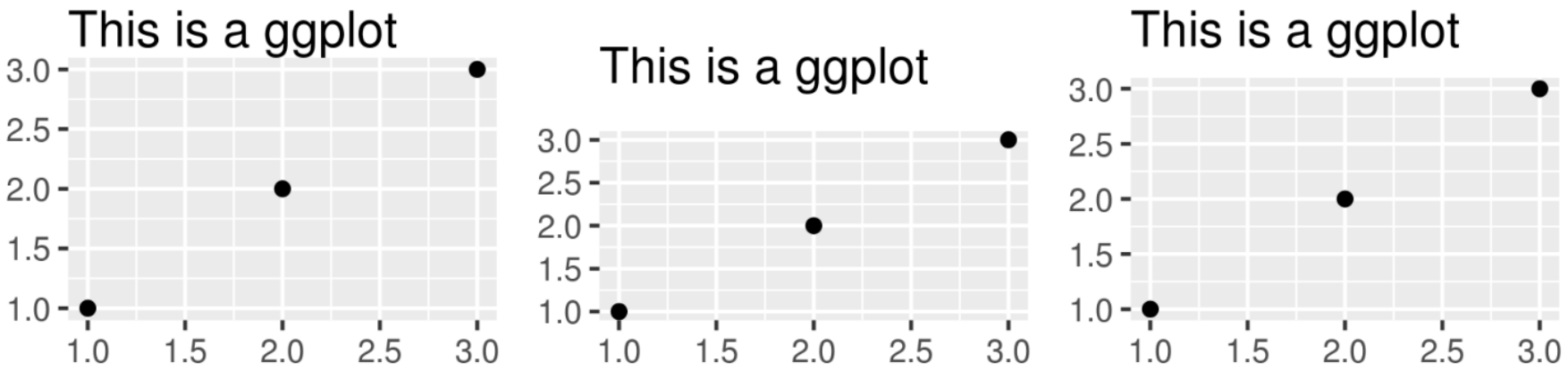
```
base_t <- base + labs(title = "This is a ggplot") + xlab(NULL) + ylab(NULL)
base_t + theme(plot.title = element_text(size = 16))
base_t + theme(plot.title = element_text(face = "bold", colour = "red"))
base_t + theme(plot.title = element_text(hjust = 1))
```



Изменение темы

Вы можете управлять полями вокруг текста с помощью margin аргумента и margin() функция. margin()имеет четыре аргумента: размер(в пунктах) для добавления в верхнюю, правую, нижнюю и левую части текста. Все элементы, не указанные по умолчанию, равны 0.

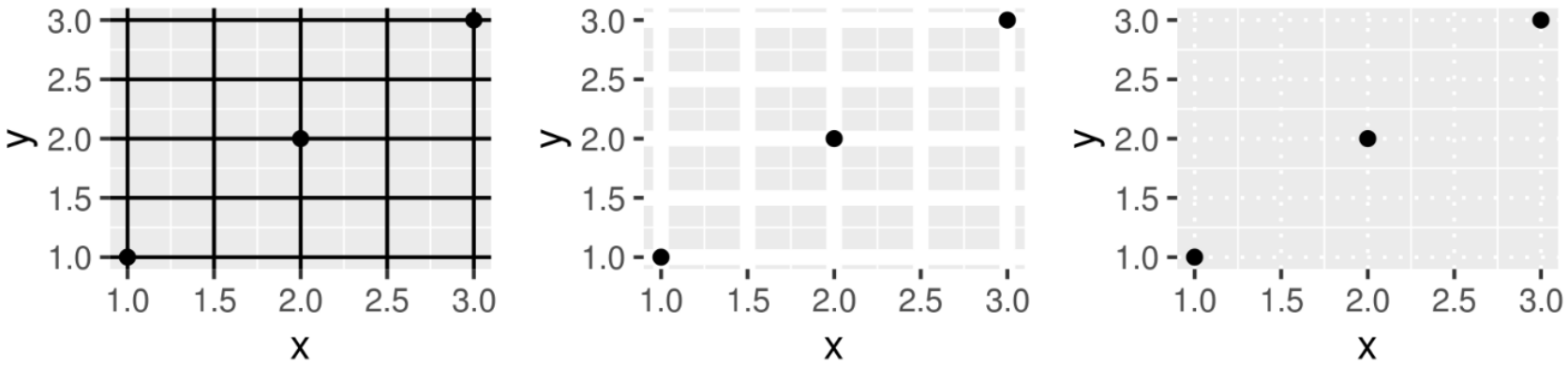
```
# The margins here look asymmetric because there are also plot margins
base_t + theme(plot.title = element_text(margin = margin()))
base_t + theme(plot.title = element_text(margin = margin(t = 10, b = 10)))
base_t + theme(axis.title.y = element_text(margin = margin(r = 10)))
```



Изменение темы

element_line() рисует линии, параметризованные colour, size и linetype:

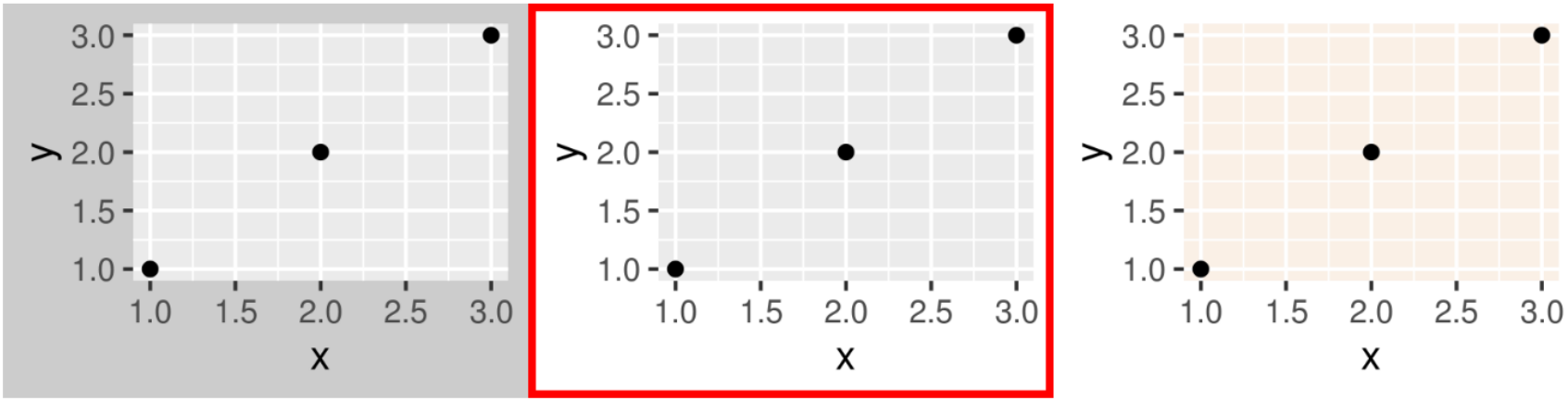
```
base + theme(panel.grid.major = element_line(colour = "black"))
base + theme(panel.grid.major = element_line(size = 2))
base + theme(panel.grid.major = element_line(linetype = "dotted"))
```



Изменение темы

element_rect() рисует прямоугольники, в основном используемые для фона, параметризованные по colour, size, linetype.

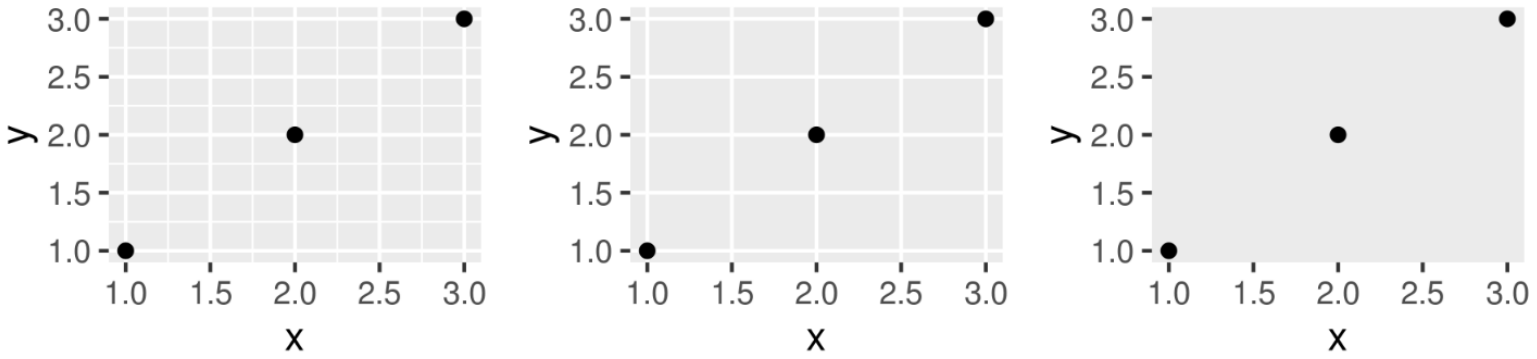
base + theme(plot.background = element_rect(fill = "grey80", colour = NA))
base + theme(plot.background = element_rect(colour = "red", size = 2))
base + theme(panel.background = element_rect(fill = "linen"))



Изменение темы

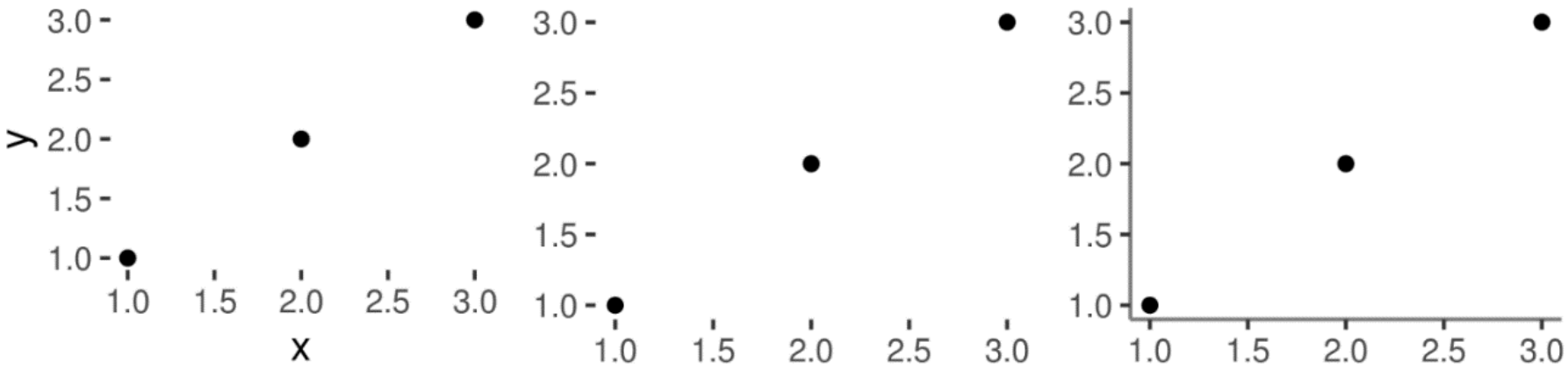
element_blank() ничего не рисует. Используйте это, если вы не хотите ничего рисовать, никакого места, выделенного для этого элемента. В следующем примере используется element_blank() чтобы постепенно подавлять появление элементов. Обратите внимание, как участок автоматически восстанавливается пространство, ранее используемое этими элементами: если вы не хотите, чтобы это случаются (возможно, потому что им нужно выстраиваться в ряд с другими сюжетами на странице), используйте colour = NA, fill = NA для создания невидимых элементов, которые все равно занимают место.

base
last_plot() + theme(panel.grid.minor = element_blank())
last_plot() + theme(panel.grid.major = element_blank())



Изменение темы

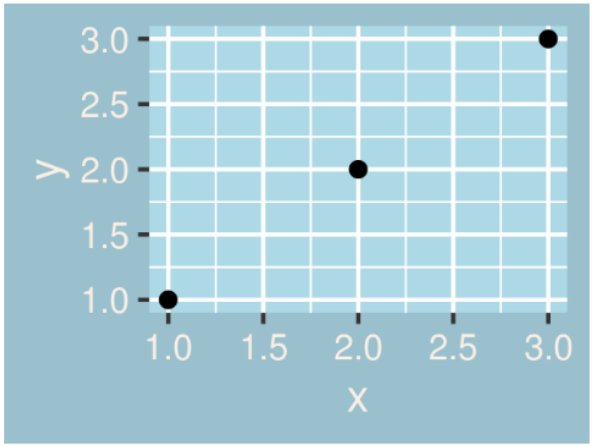
```
last_plot() + theme(panel.background = element_blank())
last_plot() + theme(
  axis.title.x = element_blank(),
  axis.title.y = element_blank()
)
last_plot() + theme(axis.line = element_line(colour = "grey50"))
```

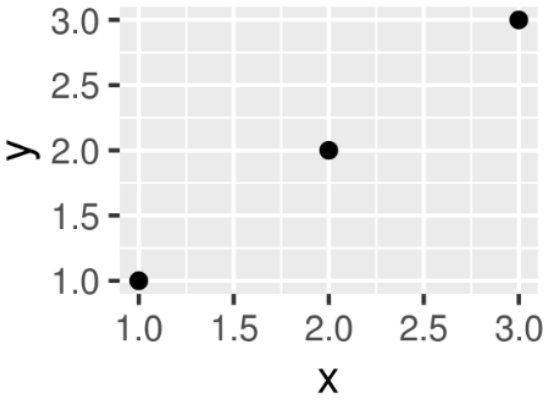


Некоторые другие настройки принимают единицы сетки.
Пример - unit(1, "cm"), unit(0.25, "in").

Изменение темы и восстановление темы

```
old_theme <- theme_update(
  plot.background = element_rect(fill = "lightblue3", colour = NA),
  panel.background = element_rect(fill = "lightblue", colour = NA),
  axis.text = element_text(colour = "linen"),
  axis.title = element_text(colour = "linen")
)
base
theme_set(old_theme)
base
```





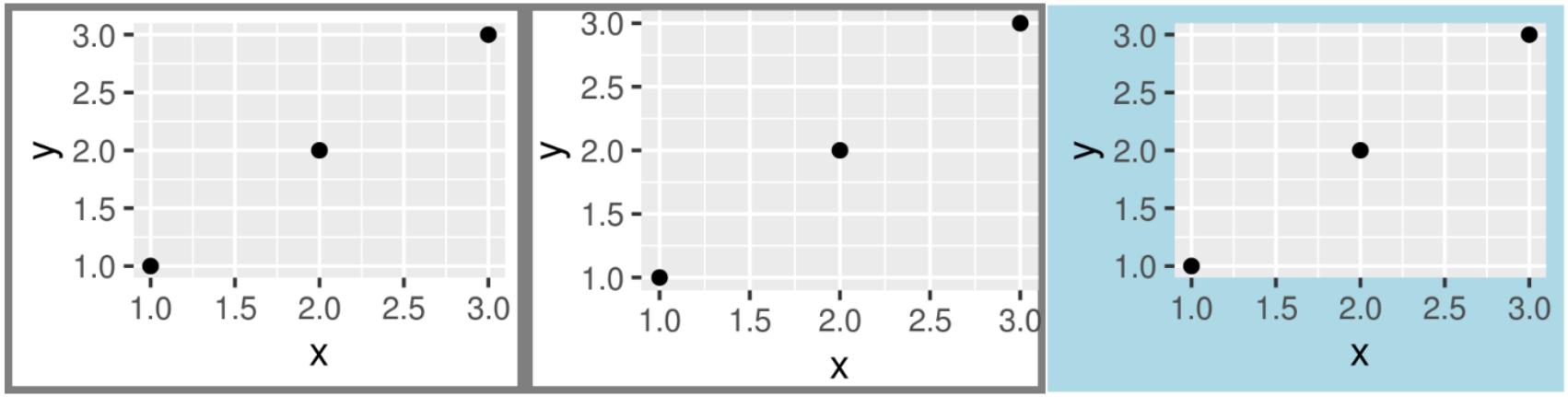
Элементы темы

Существует около 40 уникальных элементов, которые управляют внешним видом участка. Они могут быть грубо сгруппированы в пять категорий: plot, axis, legend, panel and facet.

Element	Setter	Description
plot.background	element_rect()	plot background
plot.title	element_text()	plot title
plot.margin	margin()	margins around plot

plot.background рисует прямоугольник, который лежит в основе всего остального на графике. По умолчанию ggplot2 использует белый фон, который гарантирует, что сюжет можно использовать везде, где он может закончиться (например, даже если вы сохраните его в формате png и поместите на слайд с черным фоном). При экспорте эпюр для использования в других системах может потребоваться сделать фон прозрачным fill = NA. Аналогично, если вы внедряете участок в систему, которая уже имеет поля, вы можете захотеть исключить встроенные поля. Обратите внимание, что небольшой запас необходим, если вы хотите нарисовать границу вокруг участка.

```
base + theme(plot.background = element_rect(colour = "grey50", size = 2))
base + theme(
  plot.background = element_rect(colour = "grey50", size = 2),
  plot.margin = margin(2, 2, 2, 2)
)
base + theme(plot.background = element_rect(fill = "lightblue"))
```



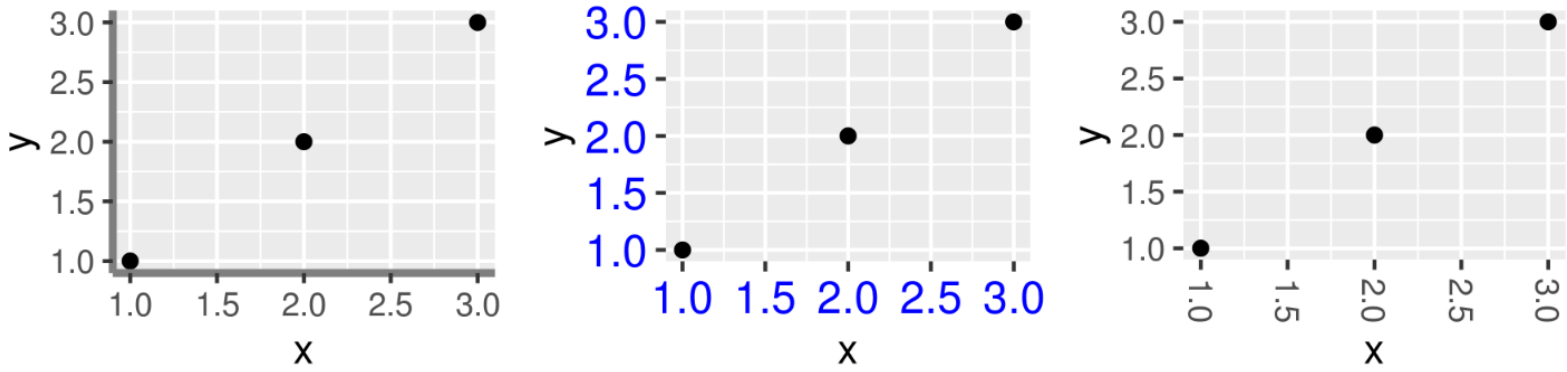
Элементы оси

Element	Setter	Description
axis.line	element_line()	line parallel to axis (hidden in default themes)
axis.text	element_text()	tick labels
axis.text.x	element_text()	x-axis tick labels
axis.text.y	element_text()	y-axis tick labels
axis.title	element_text()	axis titles
axis.title.x	element_text()	x-axis title
axis.title.y	element_text()	y-axis title
axis.ticks	element_line()	axis tick marks
axis.ticks.length	unit()	length of tick marks

Обратите внимание, что axis.text(и axis.title) поставляется в трех формах:axis.text,axis.text.x, и axis.text.y. Используйте первую форму, если вы хотите изменить свойства обеих осей сразу, любые свойства, которые вы не задали явно axis.text.x и axis.text.y будут унаследованы от axis.text.

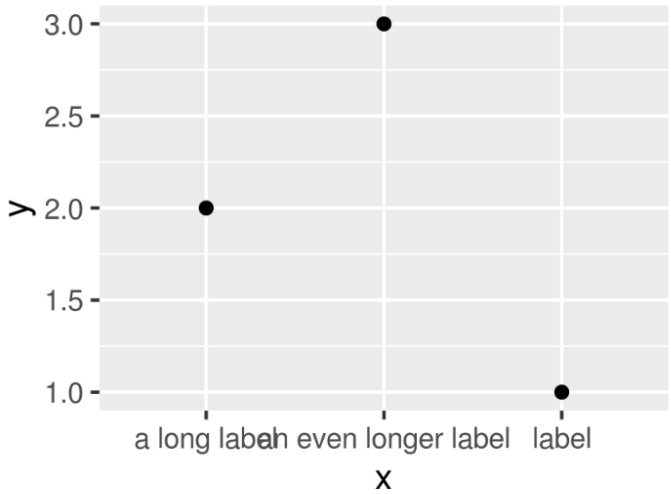
Элементы оси

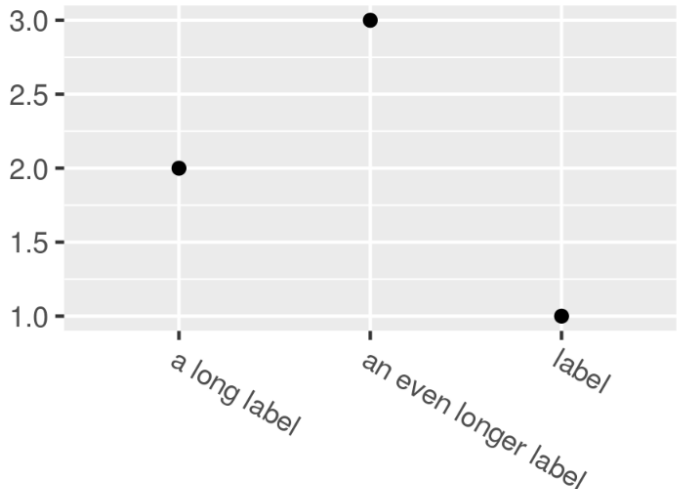
```
df <- data.frame(x = 1:3, y = 1:3)
base <- ggplot(df, aes(x, y)) + geom_point()

# Accentuate the axes
base + theme(axis.line = element_line(colour = "grey50", size = 1))
# Style both x and y axis labels
base + theme(axis.text = element_text(color = "blue", size = 12))
# Useful for long labels
base + theme(axis.text.x = element_text(angle = -90, vjust = 0.5))
```



Элементы оси

```
df <- data.frame(
  x = c("label", "a long label", "an even longer label"),
  y = 1:3
)
base <- ggplot(df, aes(x, y)) + geom_point()
base
base +
  theme(axis.text.x = element_text(angle = -30, vjust = 1, hjust = 0)) +
  xlab(NULL) +
  ylab(NULL)
```





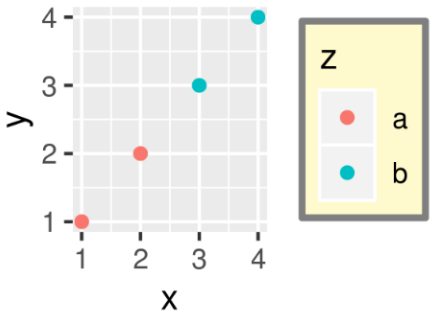
Наиболее распространенной настройкой является поворот меток оси x, чтобы избежать длинных перекрывающихся меток. Если вы сделаете это, обратите внимание, что отрицательные углы, как правило, выглядят лучше всего, и вы должны установить hjust = 0 и vjust = 1

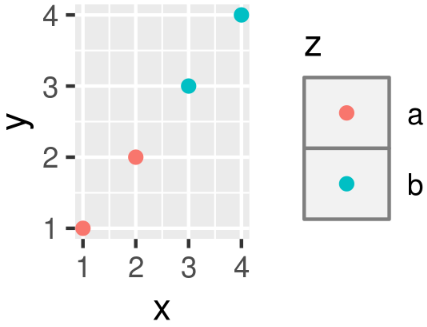
Элементы легенды

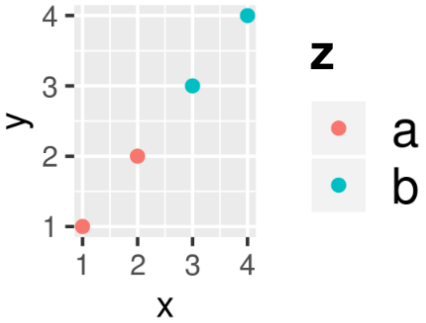
Element	Setter	Description
legend.background	element_rect()	legend background
legend.key	element_rect()	background of legend keys
legend.key.size	unit()	legend key size
legend.key.height	unit()	legend key height
legend.key.width	unit()	legend key width
legend.margin	unit()	legend margin
legend.text	element_text()	legend labels
legend.text.align	0–1	legend label alignment (0 = right, 1 = left)
legend.title	element_text()	legend name
legend.title.align	0–1	legend name alignment (0 = right, 1 = left)

Элементы легенды

```
df <- data.frame(x = 1:4, y = 1:4, z = rep(c("a", "b"), each = 2))
base <- ggplot(df, aes(x, y, colour = z)) + geom_point()

base + theme(
  legend.background = element_rect(
    fill = "lemonchiffon",
    colour = "grey50",
    size = 1
  )
)
base + theme(
  legend.key = element_rect(color = "grey50"),
  legend.key.width = unit(0.9, "cm"),
  legend.key.height = unit(0.75, "cm")
)
base + theme(
  legend.text = element_text(size = 15),
  legend.title = element_text(size = 15, face = "bold")
)
```





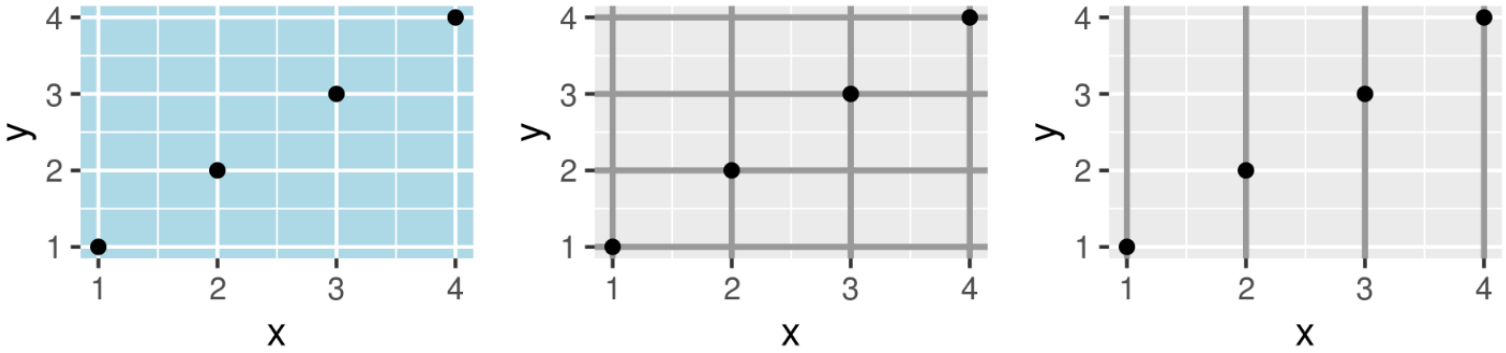


Элементы панели

Element	Setter	Description
panel.background	element_rect()	panel background (under data)
panel.border	element_rect()	panel border (over data)
panel.grid.major	element_line()	major grid lines
panel.grid.major.x	element_line()	vertical major grid lines
panel.grid.major.y	element_line()	horizontal major grid lines
panel.grid.minor	element_line()	minor grid lines
panel.grid.minor.x	element_line()	vertical minor grid lines
panel.grid.minor.y	element_line()	horizontal minor grid lines
aspect.ratio	numeric	plot aspect ratio

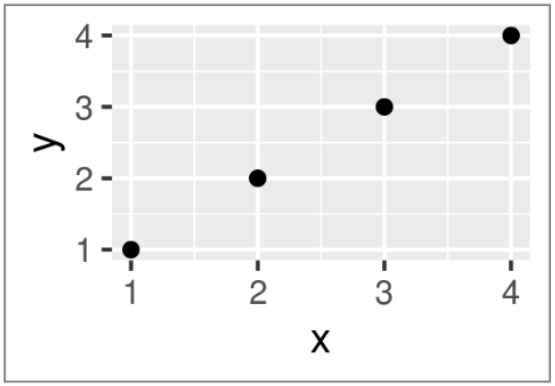
Элементы панели

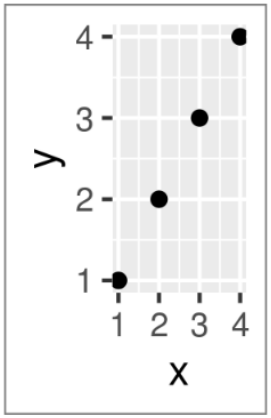
```
base <- ggplot(df, aes(x, y)) + geom_point()
# Modify background
base + theme(panel.background = element_rect(fill = "lightblue"))

# Tweak major grid lines
base + theme(
  panel.grid.major = element_line(color = "gray60", size = 0.8)
)
# Just in one direction
base + theme(
  panel.grid.major.x = element_line(color = "gray60", size = 0.8)
)
```

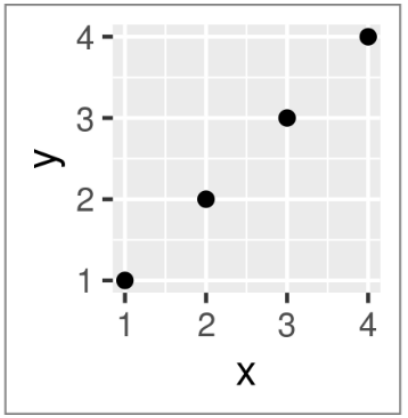


Элементы панели

```
base2 <- base + theme(plot.background = element_rect(colour =
"grey50"))
# Wide screen
base2 + theme(aspect.ratio = 9 / 16)
# Long and skiny
base2 + theme(aspect.ratio = 2 / 1)
# Square
base2 + theme(aspect.ratio = 1)
```





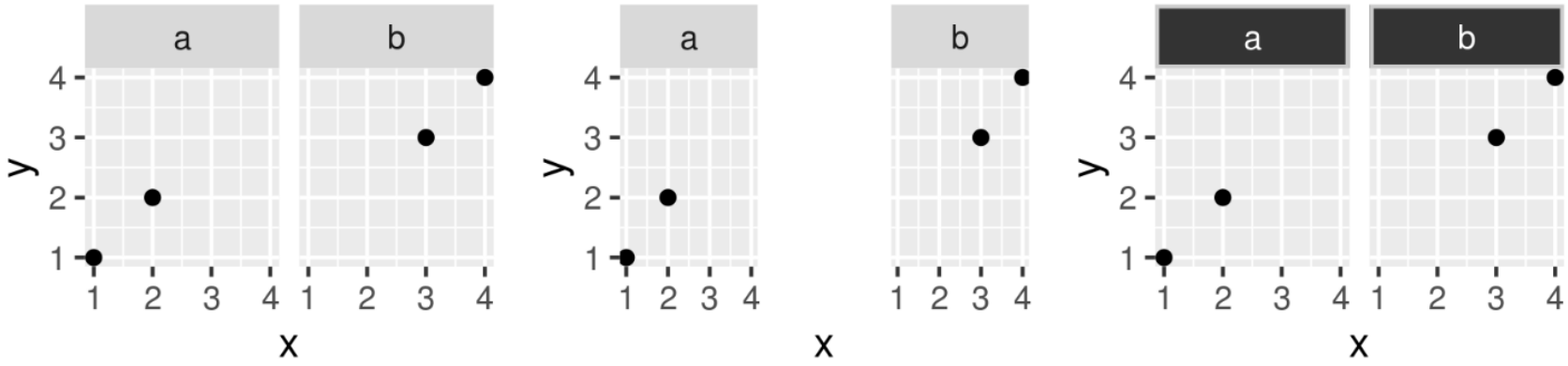


Фасеты

Element	Setter	Description
strip.background	element_rect()	background of panel strips
strip.text	element_text()	strip text
strip.text.x	element_text()	horizontal strip text
strip.text.y	element_text()	vertical strip text
panel.margin	unit()	margin between facets
panel.margin.x	unit()	margin between facets (vertical)
panel.margin.y	unit()	margin between facets (horizontal)

Фасеты

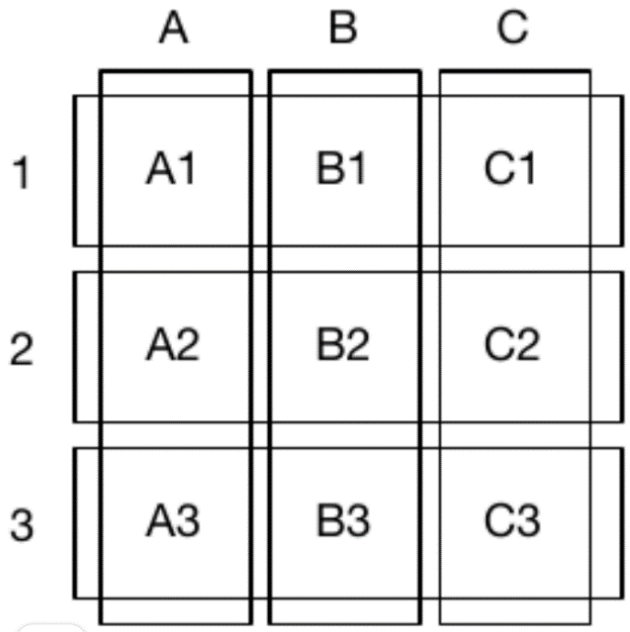
```
df <- data.frame(x = 1:4, y = 1:4, z = c("a", "a", "b", "b"))
base_f <- ggplot(df, aes(x, y)) + geom_point() + facet_wrap(~z)

base_f
base_f + theme(panel.margin = unit(0.5, "in"))
#> Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
#> instead
base_f + theme(
  strip.background = element_rect(fill = "grey20", color = "grey80", size = 1),
  strip.text = element_text(colour = "white")
)
```



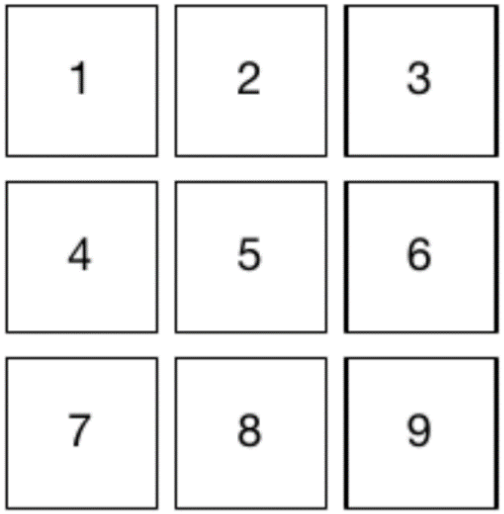
Фасеты

Существует три типа

- facet_null(): один участок, значение по умолчанию.
- facet_wrap(): "обертывает" 1d ленту панелей в 2d.
- facet_grid(): создает двумерную сетку панелей, определяемую переменными, которые сформируют строки и столбцы.



facet_grid



facet_wrap

facet_grid()(левый) принципиально 2d, будучи составленным из двух независимых компонентов.

facet_wrap()(справа) - это 1d, но завернутый в 2d, чтобы сэкономить место.

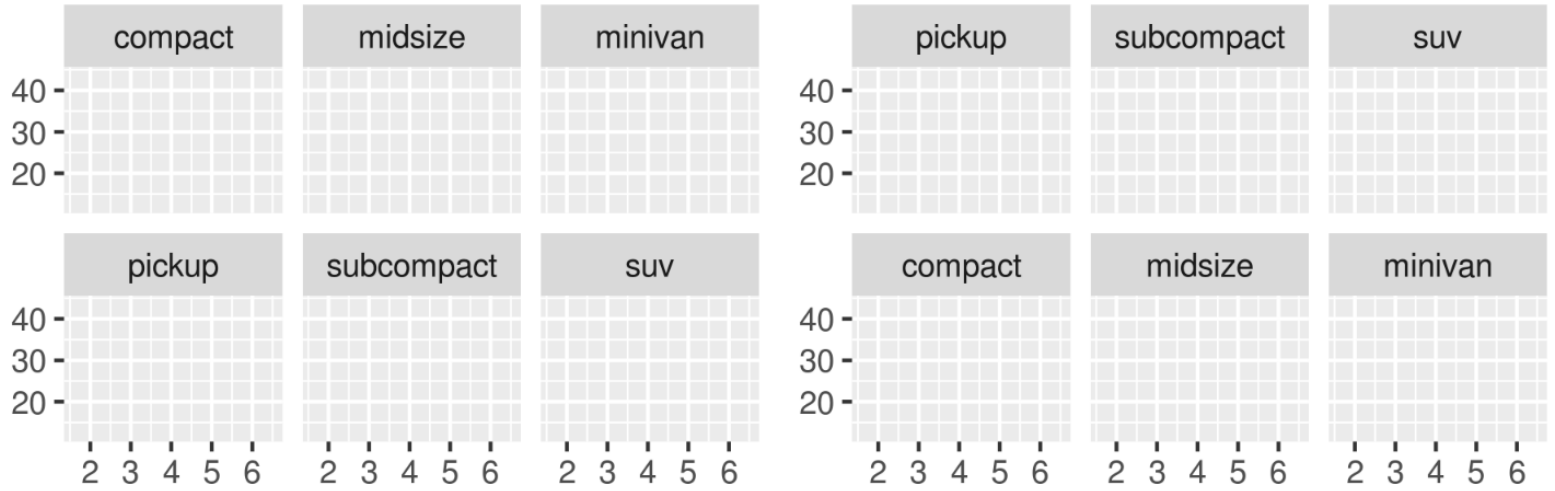
Данные

Для работы с фасетами будем использовать подмножество набора данных mpg, которое имеет управляемое число уровней:
три цилиндра (4, 6, 8), два типа трансмиссии (4 и f) и шесть классов.

```
mpg2 <- subset(mpg, cyl != 5 & drv %in% c("4", "f") & class != "2seater")
```

Расположение

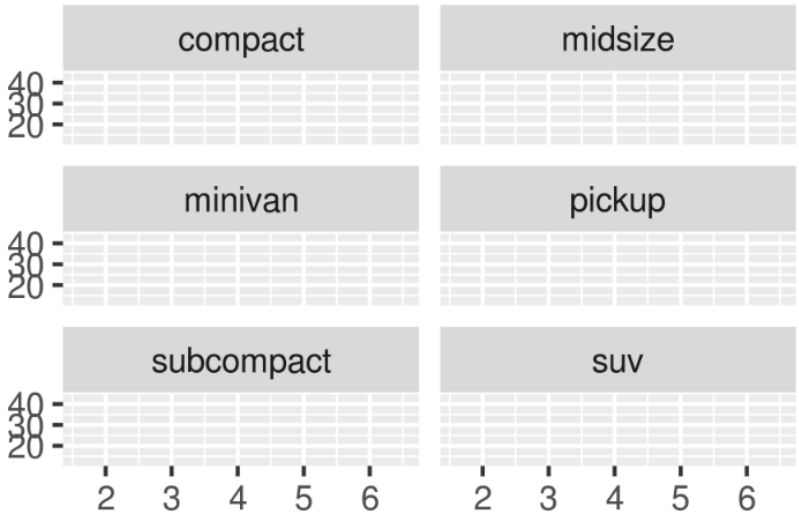
```
base <- ggplot(mpg2, aes(displ, hwy)) +
  geom_blank() +
  xlab(NULL) +
  ylab(NULL)

base + facet_wrap(~class, ncol = 3)
base + facet_wrap(~class, ncol = 3, as.table = FALSE)
```

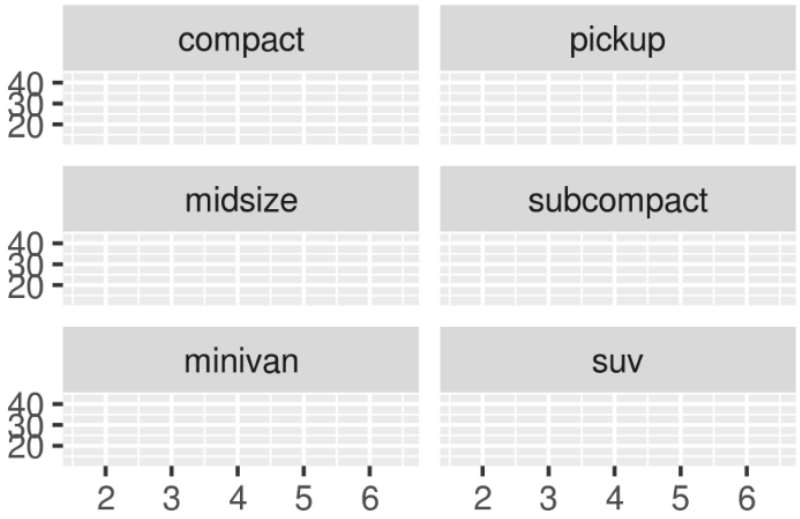


as.table определяет, будут ли фасеты расположены как таблица (TRUE) с наибольшими значениями в правом нижнем углу или (FALSE) с наибольшими значениями в правом верхнем углу.
dir управляет направлением обертывания: h orizontal или V ertical.

Расположение

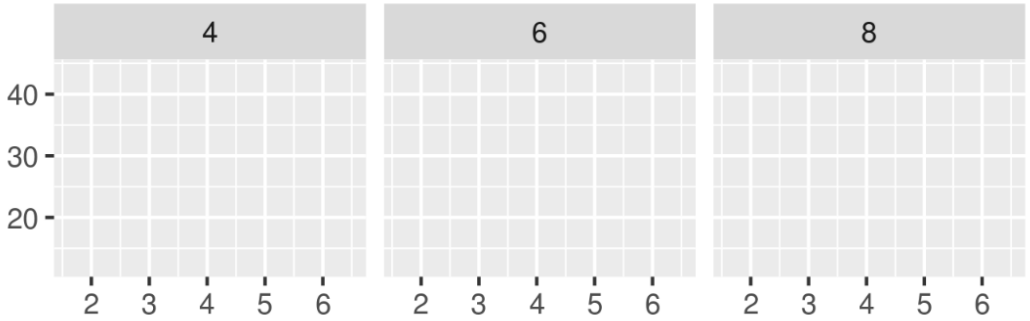
base + facet_wrap(~class, nrow = 3)
base + facet_wrap(~class, nrow = 3, dir = "v")



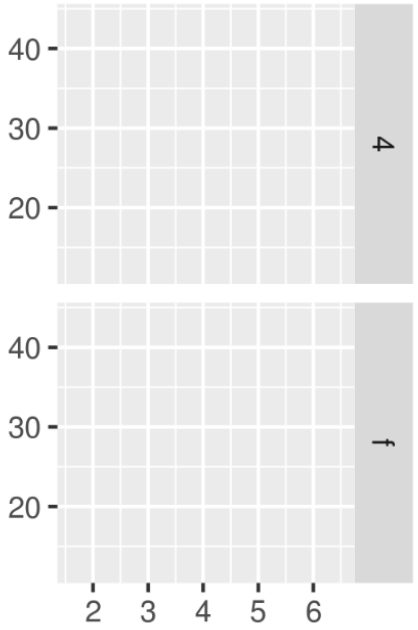


Сетка

base + facet_grid(. ~ cyl)

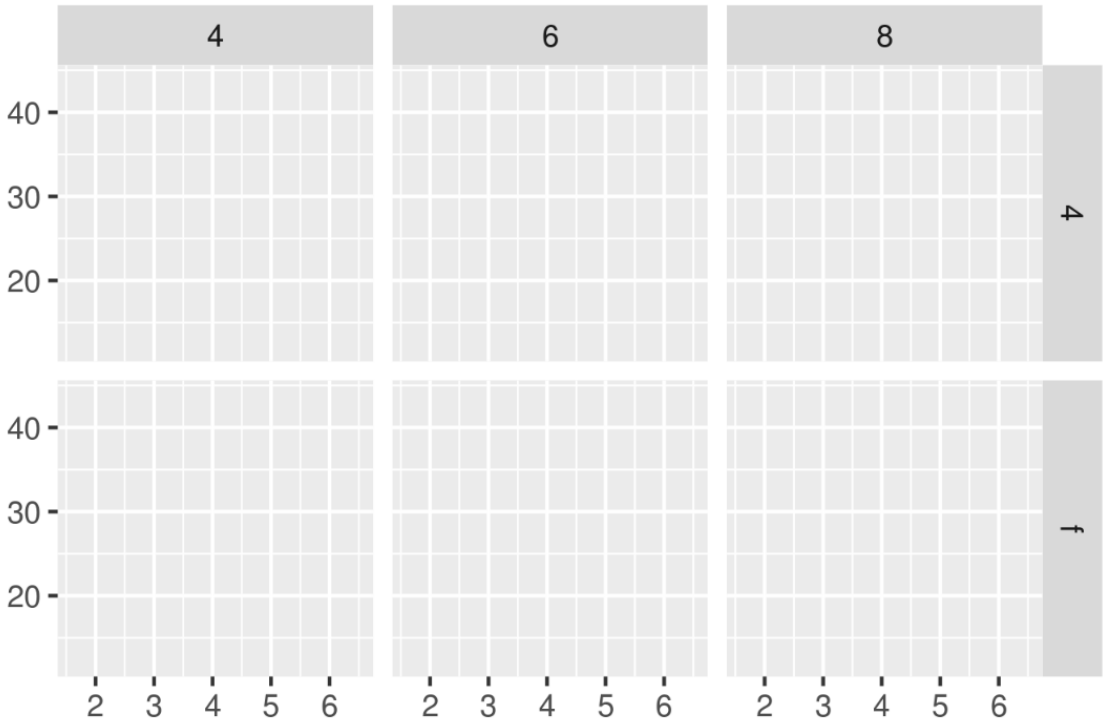


base + facet_grid(drv ~ .)



Сетка

base + facet_grid(drv ~ cyl)



Шкалы

Для facet_wrap() и facet_grid() можно управлять тем, одинаковы ли шкалы позиций во всех панелях (фиксированные) или разрешено варьироваться между панелями (свободные) с scales параметром:

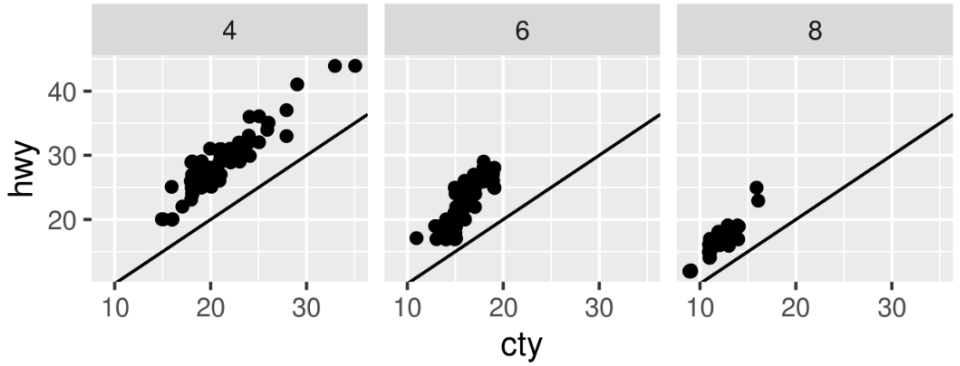
- scales = "fixed": шкалы x и y фиксируются на всех панелях.
- scales = "free_x": шкала x свободна, а шкала y фиксирована.
- scales = "free_y": шкала y свободна, а шкала x фиксирована.
- scales = "free": масштабы x и y различаются на разных панелях.

facet_grid() накладывает дополнительное ограничение на масштабы: все панели в столбце должны иметь одинаковый масштаб x, а все панели в строке - одинаковый масштаб Y. Это происходит потому, что каждый столбец имеет общую ось x, а каждая строка-общую ось y.

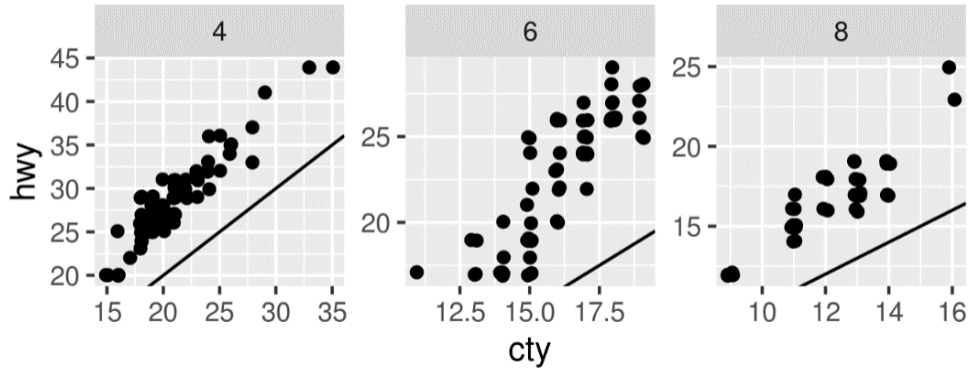
Фиксированные масштабы облегчают просмотр шаблонов на панелях; свободные масштабы облегчают просмотр шаблонов на панелях.

Шкалы

```
p <- ggplot(mpg2, aes(cty, hwy)) +
  geom_abline() +
  geom_jitter(width = 0.1, height = 0.1)
p + facet_wrap(~cyl)
```



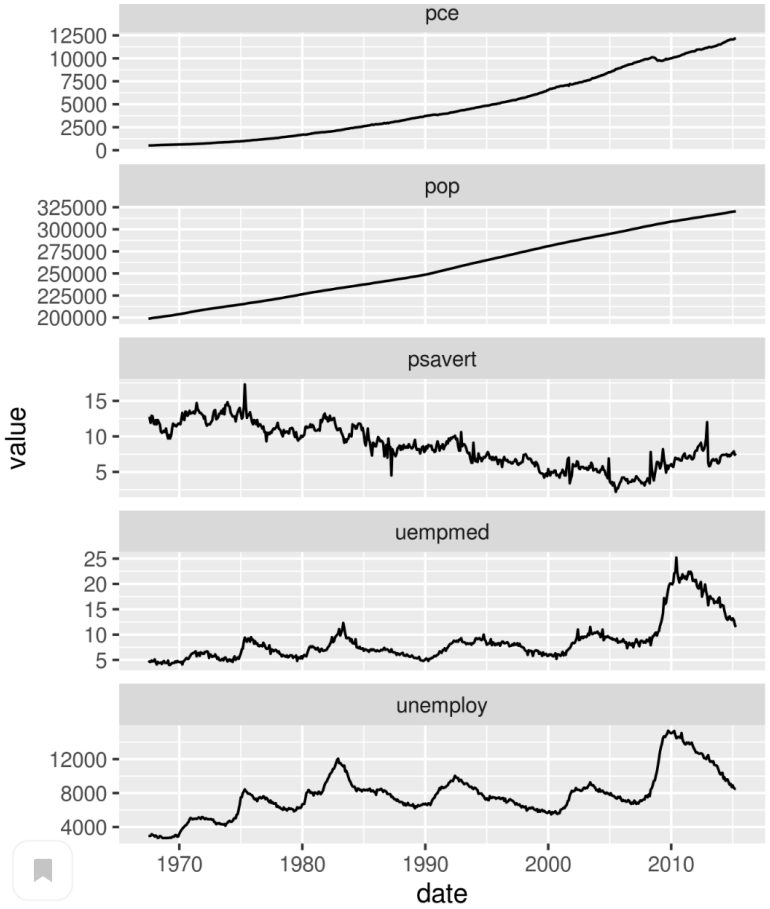

```
p + facet_wrap(~cyl, scales = "free")
```



Шкалы

Свободные шкалы также полезны, когда мы хотим отобразить несколько временных рядов, которые были измерены на разных масштабах. Для этого нам сначала нужно перейти от "широких" к "длинным" данным, складывая отдельные переменные в один столбец.

```
ggplot(economics_long, aes(date,
value)) +
  geom_line() +
  facet_wrap(~variable, scales =
"free_y", ncol = 1)
```



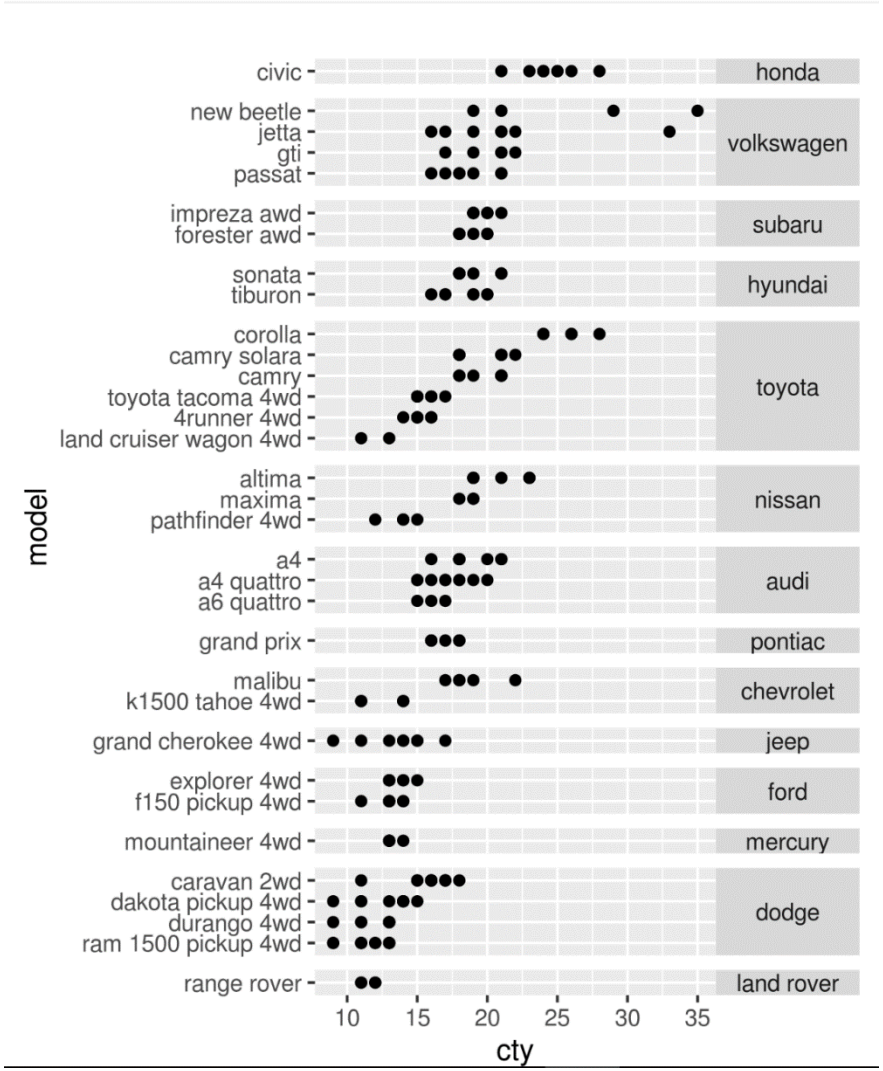
Шкалы

facet_grid () имеет дополнительный параметр space, который принимает те же значения, что и scales. Когда пространство “свободно", каждый столбец (или строка) будет иметь ширину (или высоту), пропорциональную диапазону шкалы для этого столбца (или строки). Это делает масштабирование равным по всему участку: 1 см на каждой панели соответствует одному и тому же диапазону данных.

Это в некоторой степени аналогично "нарезанным" границам оси решетки. Например, если бы панель a имела диапазон 2, а панель в-диапазон 4, то одна треть пространства была бы отведена a, а две трети - в. Это наиболее полезно для категориальных шкал, где мы можем распределить пространство пропорционально на основе количества уровней в каждом аспекте, как показано ниже.

Шкалы

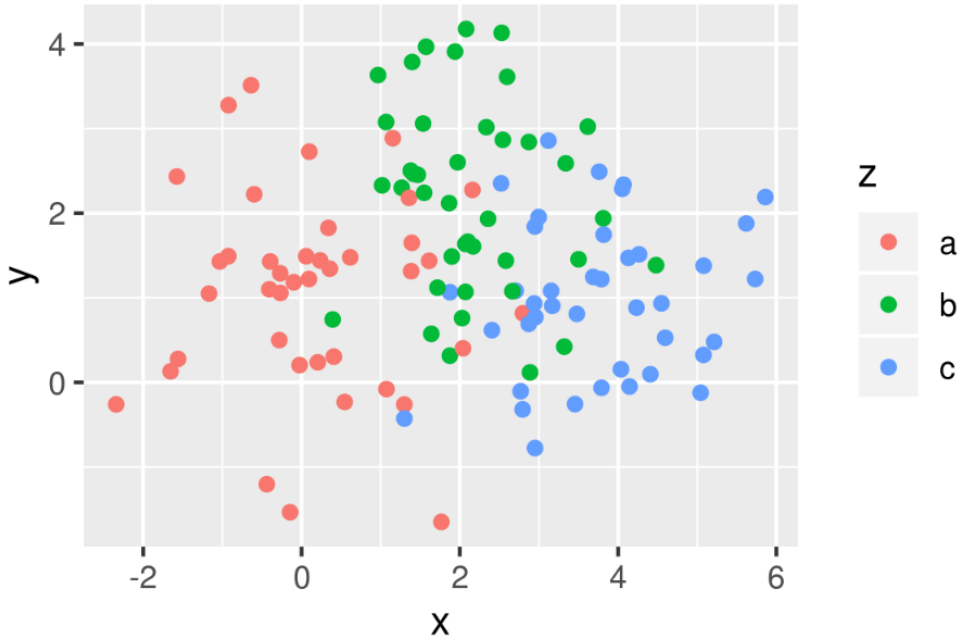
```
mpg2$model <- reorder(mpg2$model,
mpg2$cty)
mpg2$manufacturer <-
reorder(mpg2$manufacturer, -
mpg2$cty)
ggplot(mpg2, aes(cty, model)) +
  geom_point() +
  facet_grid(manufacturer ~ ., scales =
"free", space = "free") +
  theme(strip.text.y =
element_text(angle = 0))
```



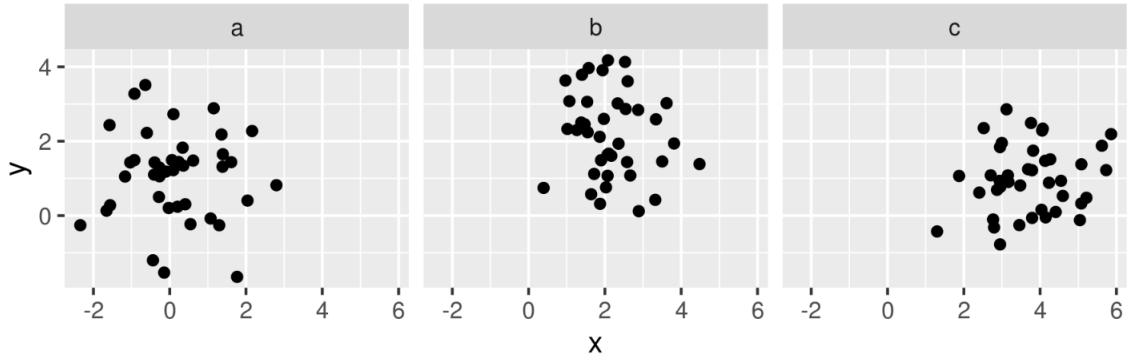
Группировка

```
df <- data.frame(
  x = rnorm(120, c(0, 2, 4)),
  y = rnorm(120, c(1, 2, 1)),
  z = letters[1:3]
)
```

```
ggplot(df, aes(x, y)) +
  geom_point(aes(colour = z))
```

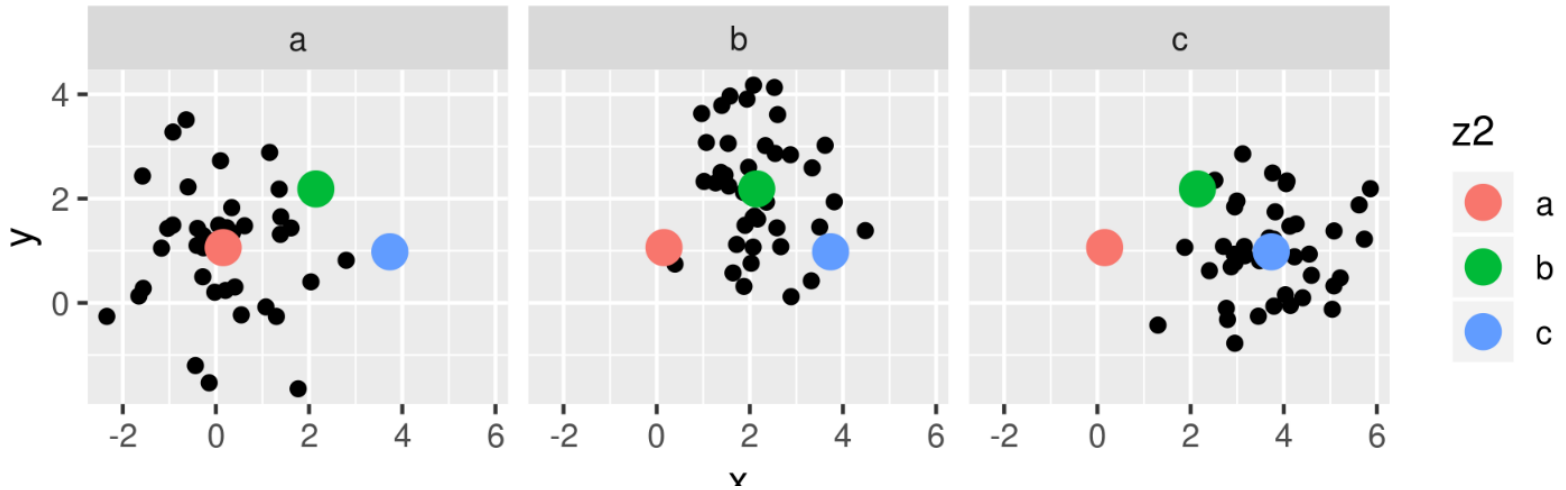



```
ggplot(df, aes(x, y)) +
  geom_point() +
  facet_wrap(~z)
```



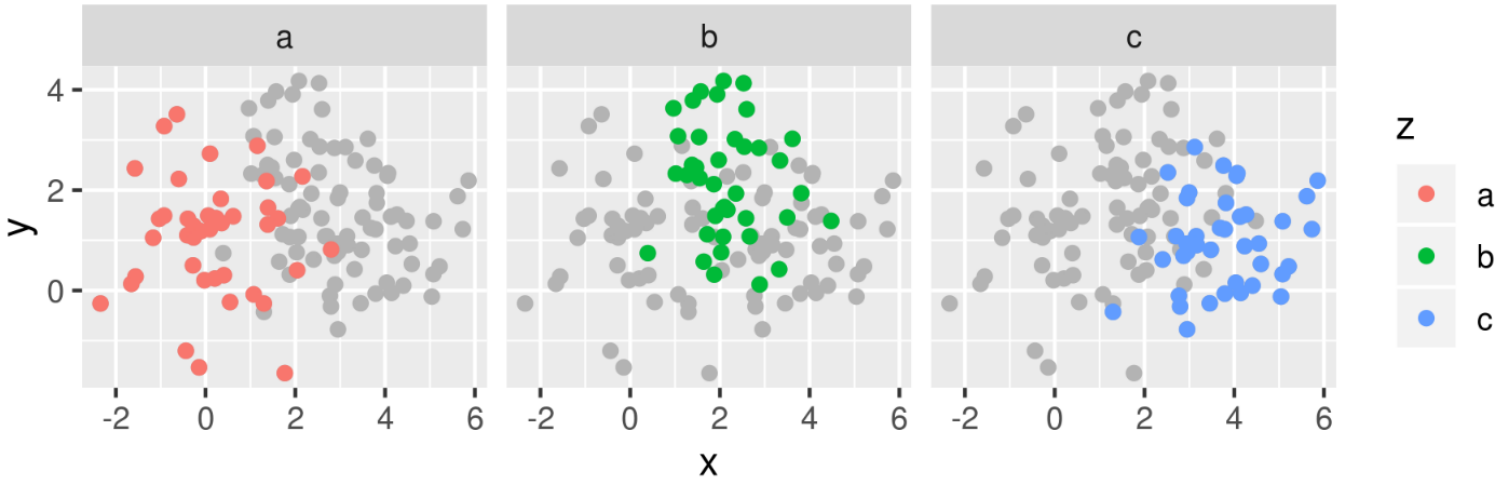
Группировка

```
install.packages("dplyr")
library(dplyr)
df_sum <- df %>%
  group_by(z) %>%
  summarise(x = mean(x), y = mean(y)) %>%
  rename(z2 = z)
ggplot(df, aes(x, y)) +
  geom_point() +
  geom_point(data = df_sum, aes(colour = z2), size = 4) +
  facet_wrap(~z)
```



Группировка

```
df2 <- dplyr::select(df, -z)

ggplot(df, aes(x, y)) +
  geom_point(data = df2, colour = "grey70") +
  geom_point(aes(colour = z)) +
  facet_wrap(~z)
```



Размещение нескольких графиков

Совместить несколько ggplot-графиков на одной или нескольких иллюстрациях, с помощью вспомогательных функций, доступных в пакетах R ggpubr, cowplot и gridExtra.

Стандартные функции R — par() и layout() — нельзя использовать, чтобы поместить несколько ggplot2-графиков на одну иллюстрацию.

Простое решение — использовать пакет R gridExtra, в котором есть такие функции: grid.arrange() и arrangeGrob(), чтобы совместить несколько ggplot-графиков в один marrangeGrob(), чтобы разместить несколько ggplot-графиков на нескольких иллюстрациях

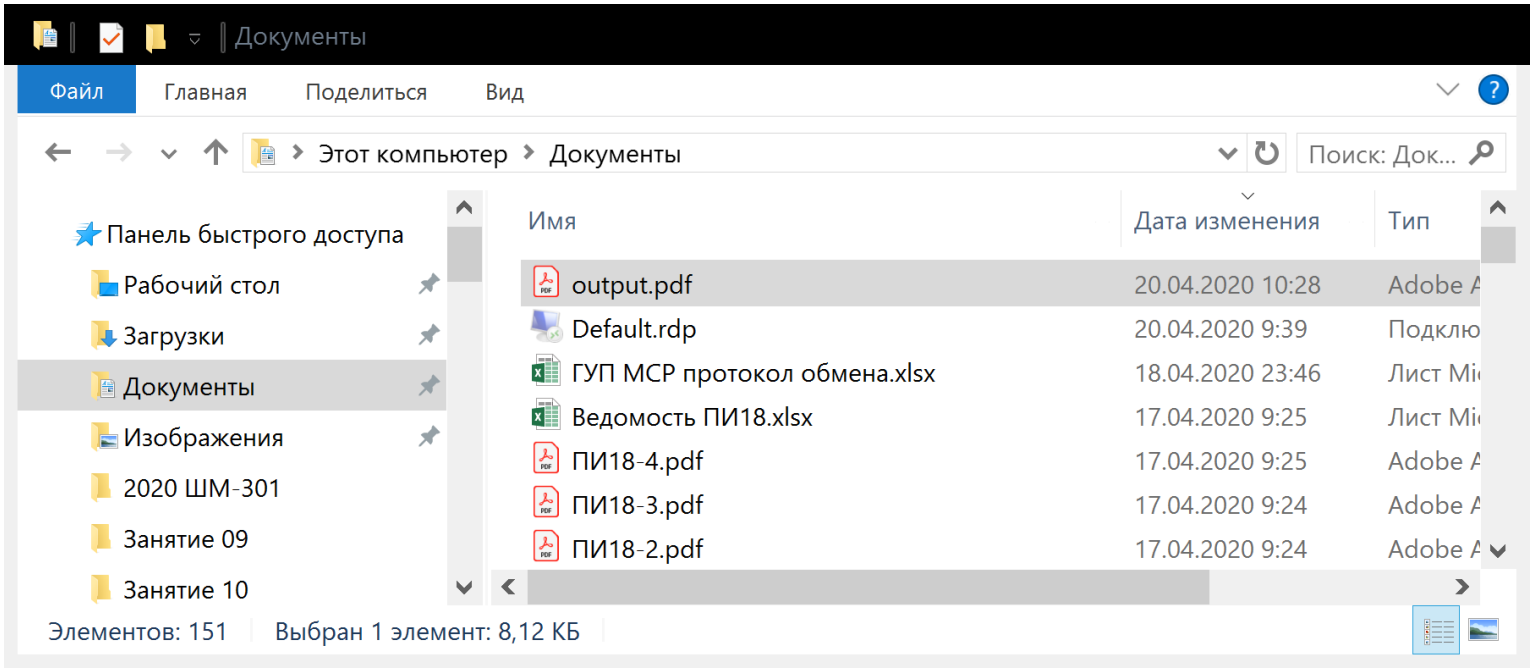
Однако, эти функции не выравнивают графики друг относительно друга; вместо этого они просто помещаются в координатную сетку, как есть, т.е. оси используют разные шкалы.

Если нужно привести оси к единой размерности, можно использовать пакет cowplot, в котором есть функция plot_grid() с аргументом align. Но этот пакет, в свою очередь, не содержит решения для размещения нескольких графиков на разных иллюстрациях. Чтобы сделать это, применяют функцию ggarrange() [в ggpubr], обёртку над функцией plot_grid(), которая умеет упорядочивать графики на нескольких иллюстрациях. Она также поможет создать общую легенду для нескольких графиков.

Сохранение результатов

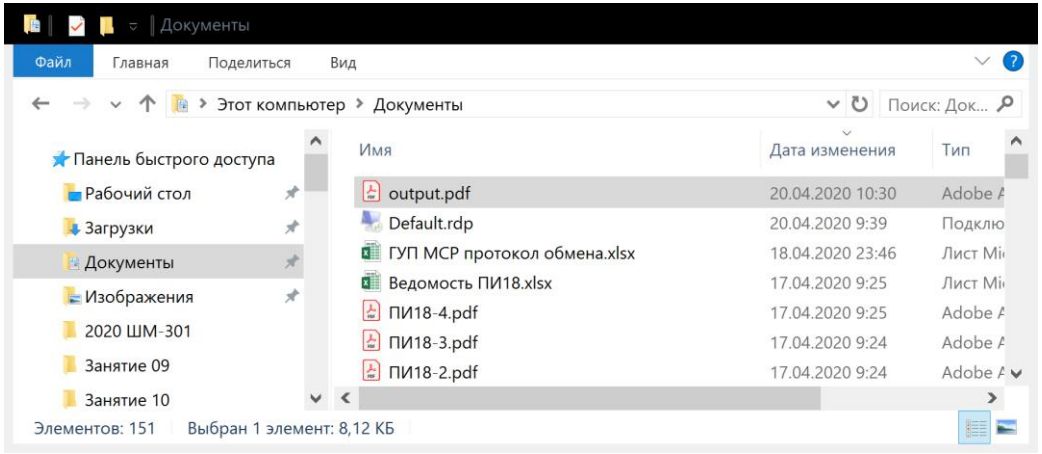
Есть два способа сохранить выходные данные из ggplot2. Вы можете использовать стандартный подход R, когда вы открываете графическое устройство, генерируете график, а затем закрываете устройство:

```
pdf("output.pdf", width = 6, height = 6)
ggplot(mpg, aes(displ, cty)) + geom_point()
dev.off()
```



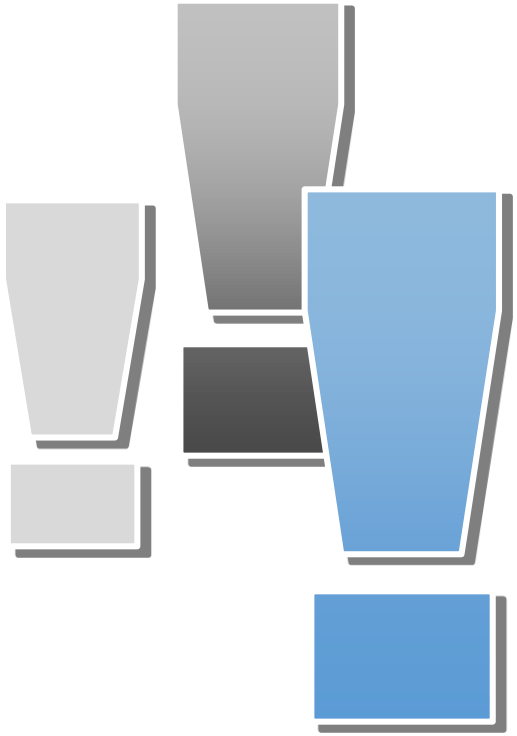
Сохранение результатов

```
ggplot(mpg, aes(displ, cty))
    + geom_point()
ggsave("output.pdf")
```



ggsave () оптимизирован для интерактивного использования: вы можете использовать его после того, как нарисовали график. Он имеет следующие важные аргументы: первый аргумент, path, указывает путь, по которому должно быть сохранено изображение. Расширение файла будет использоваться для автоматического выбора правильного графического устройства.
ggsave() может сохранить .eps, .pdf, .svg, .wmf, .png, .jpg, .bmp, .tiff.
width and height регулируют выходной размер, указанный в дюймах. Если оставить их пустыми, они будут использовать размер графического устройства на экране. Для растровой графики (т. е.формат PNG, .jpg), аргумент dpi управляет разрешением графика. По умолчанию он равен 300, что подходит для большинства принтеров, но вы можете использовать 600 для вывода с особенно высоким разрешением или 96 для отображения на экране (например, в интернете).
?ggsave для получения более подробной информации.

Спасибо за внимание!



Шевцов Василий Викторович

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