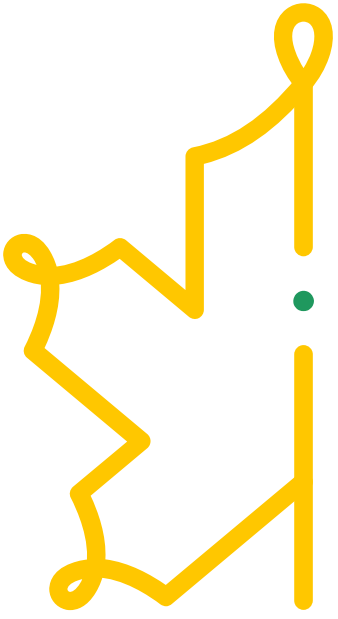


Razвед анализ данных

Строки и графика

stringr, forcats и ggplot2

Фонд интеллект

Факультет биоинженерии и биоинформатики МГУ

Анна Валяева

Лекция 4 - 2022

Строки

Работа с строками - stringr

```
library(stringr) # или library(tidyverse)
```

```
weekdays <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
weekdays
```

```
[1] "Monday"    "Tuesday"   "Wednesday" "Thursday"  "Friday"    "Saturday"
[7] "Sunday"
```

str_length

Подсчитать длину строки.

```
str_length(weekdays)
```

```
[1] 6 7 9 8 6 8 6
```

str_sub

Взять подстроку.

```
str_sub(weekdays, start = 1, end = 3)
```

```
[1] "Mon" "Tue" "Wed" "Thu" "Fri" "Sat" "Sun"
```

str_to_upper

Изменение регистра.

```
str_to_upper(weekdays)
```

```
[1] "MONDAY"    "TUESDAY"   "WEDNESDAY" "THURSDAY"  "FRIDAY"    "SATURDAY"
[7] "SUNDAY"
```

Аналогично работают `str_to_lower()`, `str_to_title()`.

str_detect

Содержит ли строка слово/шаблон - возвращает логический вектор.

```
str_detect(weekdays,"^S") # начинается на S
```

```
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
str_detect(weekdays,"day$") # заканчивается на day
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

str_which

Содержит ли строка слово/шаблон - возвращает индексы.

```
str_which(weekdays,"^S") # начинается на S
```

```
[1] 6 7
```

Регулярные выражения - cheat sheet

Regular Expressions - Regular expressions, or *regexps*, are a concise language for describing patterns in strings.

MATCH CHARACTERS

see <- function(rx) str_view_all("abc ABC 123\t.!?\\(){}\n", rx)

string (type this)	**regexp** (to mean this)	**matches** (which matches this)	**example**	
	a (etc.)	a (etc.)	see("a")	abc ABC 123 .!?\(){}
\\.	\.	.	see("\\.")	abc ABC 123 .!?\(){}
\\!	\!	!	see("\\!")	abc ABC 123 .!?\(){}
\\?	\?	?	see("\\?")	abc ABC 123 .!?\(){}
\\\\	\\	\	see("\\\\")	abc ABC 123 .!?\(){}
\\(	\(	(	see("\\(")	abc ABC 123 .!?\(){}
\\)	\)	)	see("\\)")	abc ABC 123 .!?\(){}
\\{	\{	{	see("\\{")	abc ABC 123 .!?\(){}
\\}	\}	}	see("\\}")	abc ABC 123 .!?\(){}
\\n	\n	new line (return)	see("\\n")	abc ABC 123 .!?\(){}
\\t	\t	tab	see("\\t")	abc ABC 123 .!?\(){}
\\s	\s	any whitespace (\S *for non-whitespaces*)	see("\\s")	abc ABC 123 .!?\(){}
\\d	\d	any digit (\D *for non-digits*)	see("\\d")	abc ABC 123 .!?\(){}
\\w	\w	any word character (\W *for non-word chars*)	see("\\w")	abc ABC 123 .!?\(){}
\\b	\b	word boundaries	see("\\b")	abc ABC 123 .!?\(){}
	[:digit:] [1]	digits	see("[:digit:]")	abc ABC 123 .!?\(){}
	[:alpha:] [1]	letters	see("[:alpha:]")	abc ABC 123 .!?\(){}
	[:lower:] [1]	lowercase letters	see("[:lower:]")	abc ABC 123 .!?\(){}
	[:upper:] [1]	uppercase letters	see("[:upper:]")	abc ABC 123 .!?\(){}
	[:alnum:] [1]	letters and numbers	see("[:alnum:]")	abc ABC 123 .!?\(){}
	[:punct:] [1]	punctuation	see("[:punct:]")	abc ABC 123 .!?\(){}
	[:graph:] [1]	letters, numbers, and punctuation	see("[:graph:]")	abc ABC 123 .!?\(){}
	[:space:] [1]	space characters (i.e. \s)	see("[:space:]")	abc ABC 123 .!?\(){}
	[:blank:] [1]	space and tab (but not new line)	see("[:blank:]")	abc ABC 123 .!?\(){}
	.	every character except a new line	see(".")	abc ABC 123 .!?\(){}

[1] Many base R functions require classes to be wrapped in a second set of [ ], e.g. **[[:digit:]]**

Регулярные выражения

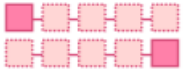
ALTERNATES

```
alt <- function(rx) str_view_all("abcde", rx)
```

regexp	matches	example	
ab|d	or	alt("ab|d")	abcde
[abe]	one of	alt("[abe]")	abcde
[^abe]	anything but	alt("[^abe]")	abcde
[a-c]	range	alt("[a-c]")	abcde

ANCHORS

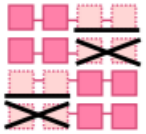
```
anchor <- function(rx) str_view_all("aaa", rx)
```



regexp	matches	example	
^a	start of string	anchor("^a")	aaa
a$	end of string	anchor("a$")	aaa

LOOK AROUNDS

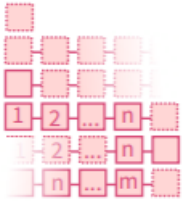
```
look <- function(rx) str_view_all("bacad", rx)
```



regexp	matches	example	
a(?=c)	followed by	look("a(?=c)")	bacad
a(?!c)	not followed by	look("a(?!c)")	bacad
(?<=b)a	preceded by	look("(?<=b)a")	bacad
(?<!b)a	not preceded by	look("(?<!b)a")	bacad

QUANTIFIERS

```
quant <- function(rx) str_view_all(".a.aa.aaa", rx)
```



regexp	matches	example	
a?	zero or one	quant("a?")	.a.aa.aaa
a*	zero or more	quant("a*")	.a.aa.aaa
a+	one or more	quant("a+")	.a.aa.aaa
a{n}	exactly **n**	quant("a{2}")	.a.aa.aaa
a{n,}	**n** or more	quant("a{2,}")	.a.aa.aaa
a{n, m}	between **n** and **m**	quant("a{2,4}")	.a.aa.aaa

GROUPS

```
ref <- function(rx) str_view_all("abbaab", rx)
```

Use parentheses to set precedent (order of evaluation) and create groups

regexp	matches	example	
(ab|d)e	sets precedence	alt("(ab|d)e")	abcde

Use an escaped number to refer to and duplicate parentheses groups that occur earlier in a pattern. Refer to each group by its order of appearance

string (type this)	regexp (to mean this)	matches (which matches this)	example (the result is the same as ref("abba"))	
\\1	\1 (etc.)	first () group, etc.	ref("(a)(b)\\2\\1")	abbaab

str_count

Подсчет обнаруженных совпадений в строке.

```
str_count(weekdays, "e")
```

```
[1] 0 1 2 0 0 0 0
```

Специальные символы нужно экранировать.

```
str_count(c("what?", "when??", "how?!!"), "?")
```

```
Error in stri_count_regex(string, pattern, opts_regex = opts(pattern)): Syntax error in regexp pattern.
   (U_REGEX_RULE_SYNTAX, context=`?`)
```

```
str_count(c("what?", "when??", "how?!!"), "\\?")
```

```
[1] 1 2 1
```

str_remove или str_remove_all

Удалить символы по шаблону.

```
str_remove_all(weekdays, "[aeiou]") # все гласные
```

```
[1] "Mndy"   "Tsdy"   "Wdnsdy" "Thrsdy" "Frdy"   "Strdy"  "Sndy"
```

`str_remove` удаляет только первое вхождение.

```
str_remove(weekdays, "[aeiou]") # первая гласная в слове
```

```
[1] "Mnday"    "Tesday"   "Wdnesday" "Thrsday"  "Frday"    "Sturday"  "Snday"
```

str_replace или str_replace_all

Заменить символы по шаблону.

```
str_replace_all(weekdays, "day", "dead")
```

```
[1] "Mondead"    "Tuesdead"   "Wednesdead" "Thursdead"  "Fridead"
[6] "Saturdead"  "Sundead"
```

`str_replace` заменит только первое вхождение.

str_subset

Возвращает строки, содержащие слово/шаблон .

```
str_subset(weekdays, "ur")
```

```
[1] "Thursday" "Saturday"
```

str_extract

Возвращает первое найденное слово/шаблон в каждой строке.

```
str_extract(weekdays, "ur")
```

```
[1] NA   NA   NA   "ur" NA   "ur" NA
```

`str_extract_all` возвращает все найденные слова/шаблоны.

stringr - что почитать

- stringr cheatsheet
- Список всех функций пакета `stringr`: `help(package = "stringr")`
- Небольшой мануал на странице Tidyverse

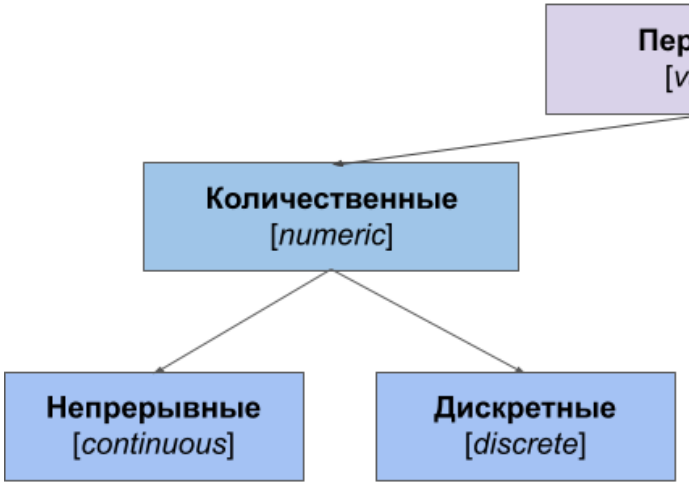
Факторы

Факторы

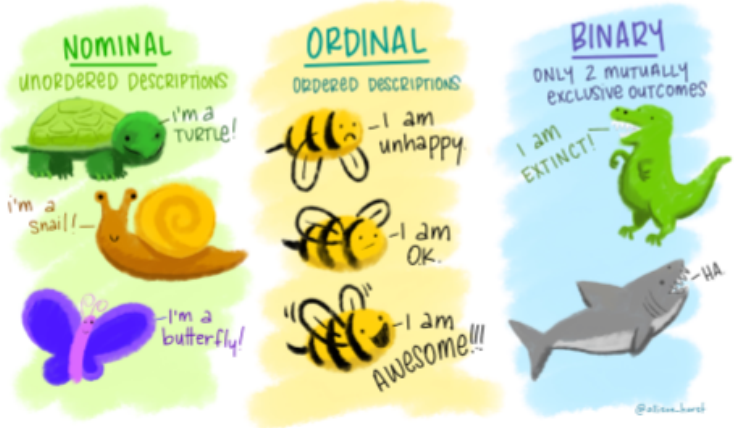
- Используются для работы с категориальными переменными
- Уровни фактора - ограниченное число известных значений категориальной переменной
- Для работы с факторами есть пакет `forcats` в составе `tidyverse`

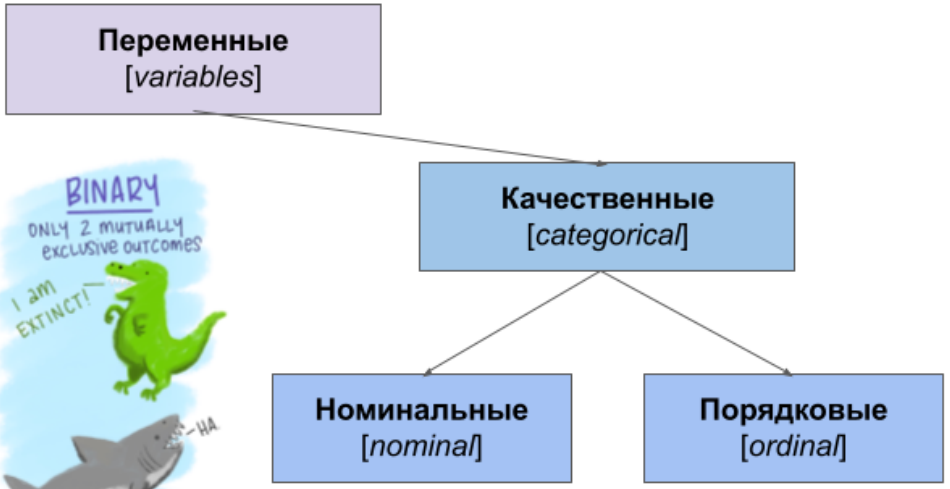
```
library(forcats) # или library(tidyverse)
```

Количественные переменные



Качественные переменные





Дни недели

```
weekdays <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
weekdays
```

```
[1] "Monday"    "Tuesday"   "Wednesday" "Thursday"  "Friday"    "Saturday"
[7] "Sunday"
```

```
as.factor(weekdays) # уровни по алфавиту
```

```
[1] Monday    Tuesday   Wednesday Thursday  Friday    Saturday  Sunday
Levels: Friday Monday Saturday Sunday Thursday Tuesday Wednesday
```

```
# forcats сохраняет порядок уровней, который был в векторе
weekdays_as_fct <- as_factor(weekdays)
weekdays_as_fct
```

```
[1] Monday    Tuesday   Wednesday Thursday  Friday    Saturday  Sunday
Levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday
```

```
typeof(weekdays)
```

```
[1] "character"
```

```
class(weekdays)
```

```
[1] "character"
```

```
as.integer(weekdays)
```

```
[1] NA NA NA NA NA NA NA
```

```
sort(weekdays[c(5:7,1:4)])
```

```
[1] "Friday"    "Monday"    "Saturday"  "Sunday"
   "Thursday"  "Tuesday"
[7] "Wednesday"
```

```
typeof(weekdays_as_fct)
```

```
[1] "integer"
```

```
class(weekdays_as_fct)
```

```
[1] "factor"
```

```
as.integer(weekdays_as_fct)
```

```
[1] 1 2 3 4 5 6 7
```

```
sort(weekdays_as_fct[c(5:7,1:4)])
```

```
[1] Monday    Tuesday   Wednesday Thursday  Friday
   Saturday  Sunday
Levels: Monday Tuesday Wednesday Thursday Friday
   Saturday Sunday
```

Уровни

```
levels(weekdays)
```

```
NULL
```

```
levels(weekdays_as_fct)
```

```
[1] "Monday"    "Tuesday"   "Wednesday" "Thursday"  "Friday"    "Saturday"
[7] "Sunday"
```

```
weekdays_as_fct <- weekdays_as_fct[1:3]
weekdays_as_fct
```

```
[1] Monday    Tuesday   Wednesday
Levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday
```

Создание вектора факторов

```
seasons_with_rep <- c("winter", "winter", "fall", "summer", "sommer", "fall", "fall")

seasons_levels <- c("winter", "spring", "summer", "fall")

seasons <- factor(seasons_with_rep, levels = seasons_levels)

seasons
```

```
[1] winter winter fall   summer <NA>   fall   fall
Levels: winter spring summer fall
```

fct_count

Подсчитать количество факторов каждого уровня.

```
seasons
```

```
[1] winter winter fall   summer <NA>   fall   fall
Levels: winter spring summer fall
```

```
fct_count(seasons)
```

```
# A tibble: 5 x 2
  f          n
  <fct>  <int>
1 winter     2
2 spring     0
3 summer     1
4 fall       3
5 <NA>       1
```

fct_drop

Удалить неиспользуемые уровни фактора: `spring`.

```
seasons
```

```
[1] winter winter fall   summer <NA>   fall   fall
Levels: winter spring summer fall
```

```
fct_drop(seasons)
```

```
[1] winter winter fall   summer <NA>   fall   fall
Levels: winter summer fall
```

fct_explicit_na

Приписать отсутствующим уровням (`NA`) явное название.

```
seasons
```

```
[1] winter winter fall   summer <NA>   fall   fall
Levels: winter spring summer fall
```

```
fct_explicit_na(seasons) # na_level задает название нового уровня
```

```
[1] winter    winter    fall      summer    (Missing) fall      fall
Levels: winter spring summer fall (Missing)
```

fct_inorder

Упорядочить уровни фактора в порядке встречаемости в векторе.

```
fct_drop(seasons) %>% fct_inorder() # не должно быть неиспользуемых уровней
```

```
[1] winter winter fall   summer <NA>   fall   fall
Levels: winter fall summer
```

fct_infreq

Упорядочить уровни фактора по частоте встречаемости.

```
fct_infreq(seasons)
```

```
[1] winter winter fall   summer <NA>   fall   fall
Levels: fall winter summer spring
```

fct_rev

Изменить порядок уровней на обратный.

```
levels(seasons)
```

```
[1] "winter" "spring" "summer" "fall"
```

```
fct_rev(seasons) %>% levels()
```

```
[1] "fall"   "summer" "spring" "winter"
```

fct_shuffle

Перемешать уровни.

```
fct_shuffle(seasons) %>% levels()
```

```
[1] "fall"   "winter" "summer" "spring"
```

fct_reorder

```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
iris$Species %>% as_factor() %>% levels
```

```
[1] "setosa"     "versicolor" "virginica"
```

fct_reorder

```
iris %>% group_by(Species) %>% summarise(min = mean(Sepal.Width))
```

```
# A tibble: 3 x 2
  Species      min
  <fct>      <dbl>
1 setosa      3.43
2 versicolor  2.77
3 virginica   2.97
```

```
iris$Species %>% as_factor() %>% fct_reorder(iris$Sepal.Width, min) %>% levels()
```

```
[1] "versicolor" "virginica"  "setosa"
```

См. еще `fct_reorder2` 

fct_lump

```
homeworld_fct <- starwars$homeworld %>% as_factor() %>% fct_explicit_na("Unknown")

levels(homeworld_fct) %>% length()
```

```
[1] 49
```

```
fct_count(homeworld_fct, sort = TRUE) %>% head(5)
```

```
# A tibble: 5 x 2
  f            n
  <fct>    <int>
1 Naboo       11
2 Tatooine    10
3 Unknown     10
4 Alderaan     3
5 Kamino       3
```

fct_lump

Семейство функций, которое "схлопывает" часть уровней по условию и создает уровень "Other" (`other_level`).

```
homeworld_fct %>% fct_lump_n(3) %>% table()
```

```
.
Tatooine    Naboo  Unknown    Other
      10       11       10       56
```

- `fct lump prop` - оставляет уровни, встречающиеся не чаще указанной частоты
- `fct lump min` - оставляет уровни, встречающиеся не реже указанного числа раз
- `fct_lump_lowfreq` - максимально наполняет группу Other так, чтобы она все равно оставалась самой малопредставленной

forcats - что почитать

- forcats cheatsheet
- Список всех функций пакета `forcats`: `help(package = "forcats")`
- Factors Chapter in R4DS

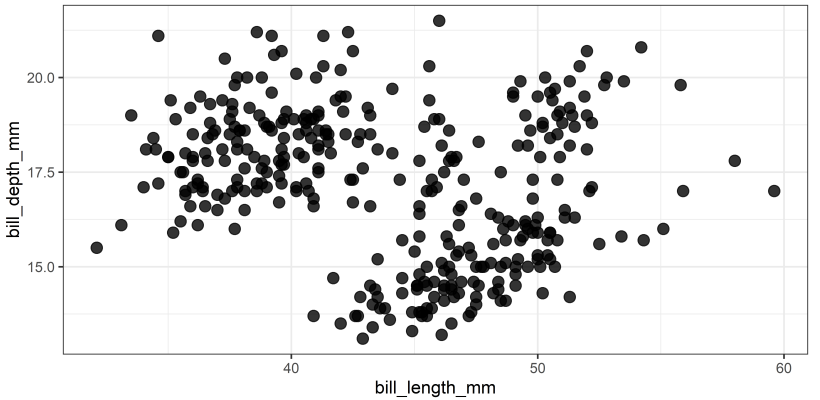
Ggplot2 графика

ggplot2

```
ggplot(
  data = <DATA>,
  mapping = aes(<MAPPINGS>)) +

  <GEOM_FUNCTION>() +

  ...
```

Данные -> оси -> тип графика -> ...

```
ggplot(penguins) +
  aes(
    x = bill_length_mm,
    y = bill_depth_mm) +
  geom_point(
    size = 3,
    alpha = 0.8) +
  theme_bw()
```



ggplot2

data =

x	y	color	size	…
16	198	cat	3	…
10	245	cat	2	…
7	178	dog	1	…
26	282	dog	2	…

mapping = aesthetic properties

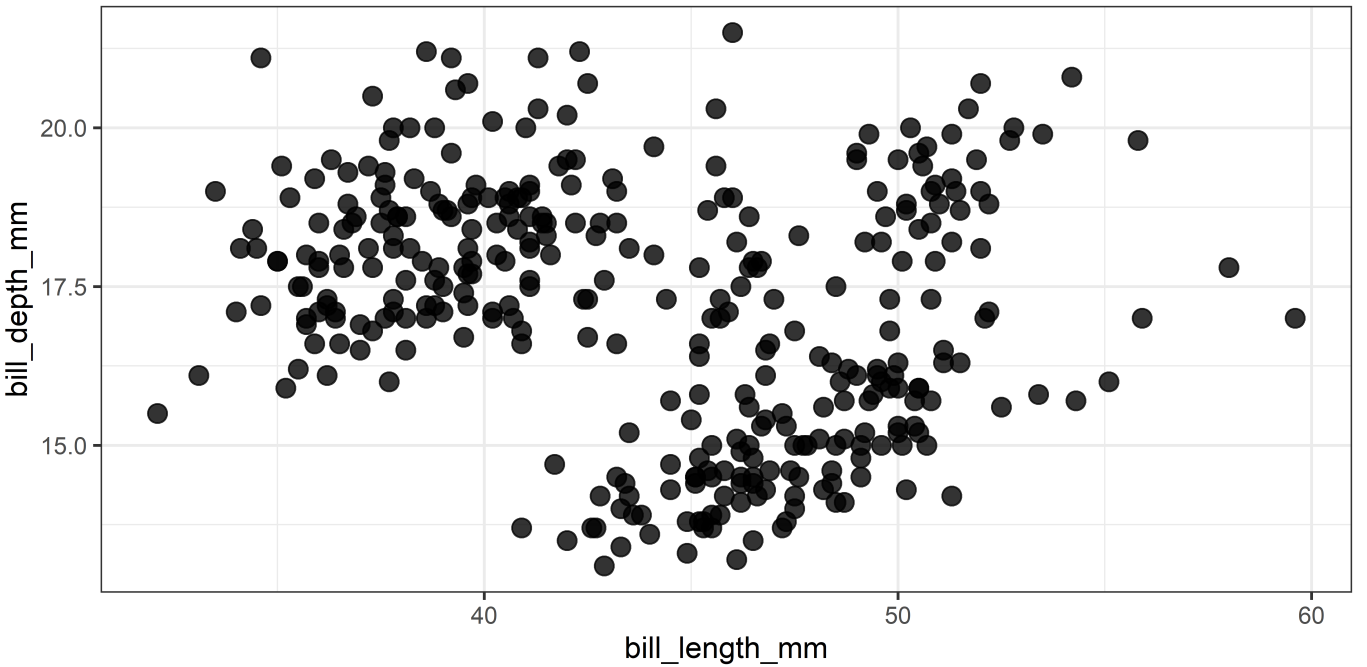
x = x_col
y = y_col
color = color_col
size = size_col
…

geometric objects

geom_point()
geom_bar()
geom_boxplot()
geom_line()
…

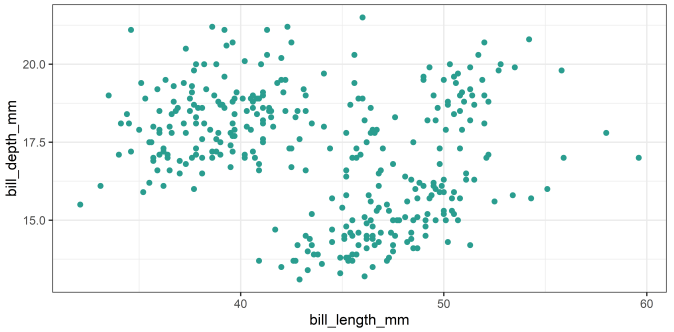
Запишем кусок графика в переменную

```
p <- ggplot(penguins, aes(x = bill_length_mm,  y = bill_depth_mm)) +
  theme_bw()

p + geom_point(size = 3, alpha = 0.8)
```

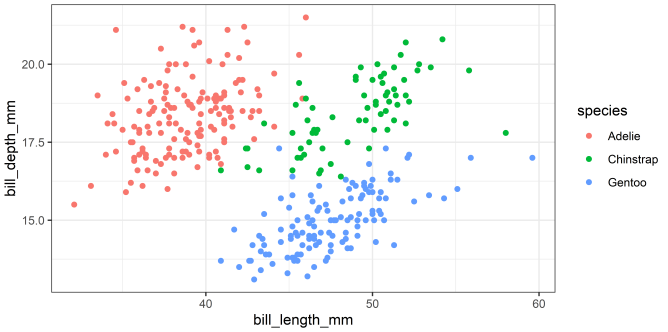


Константа *vs* переменная

```
p +
  geom_point(color = "#2a9d8f")
```



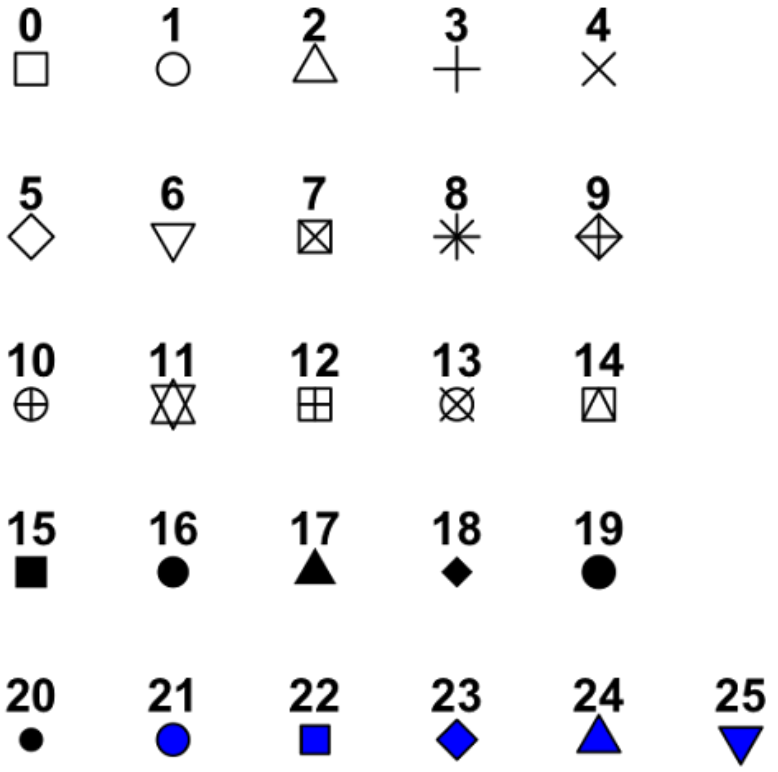

```
p +
  geom_point(aes(color = species))
```



Варианты `aes` - aesthetic

Для geom_point()

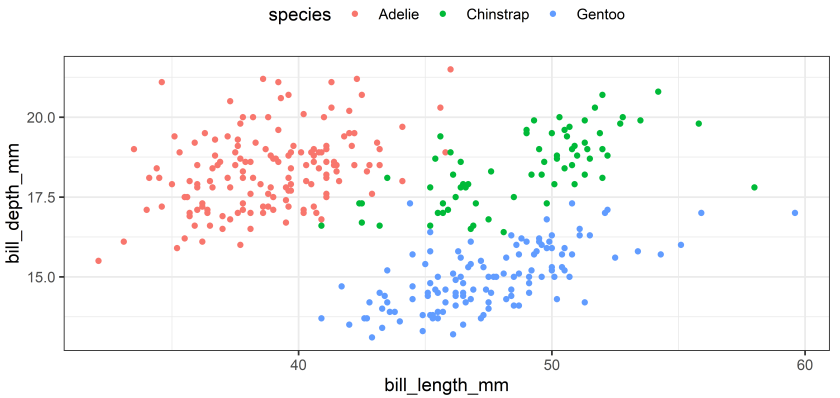
- `shape` - тип символа
- `color` - цвет общий / цвет обводки
- `fill` - заливка
- `size` - размер
- `stroke` - толщина обводки
- `alpha` - прозрачность



[Cheat sheets]

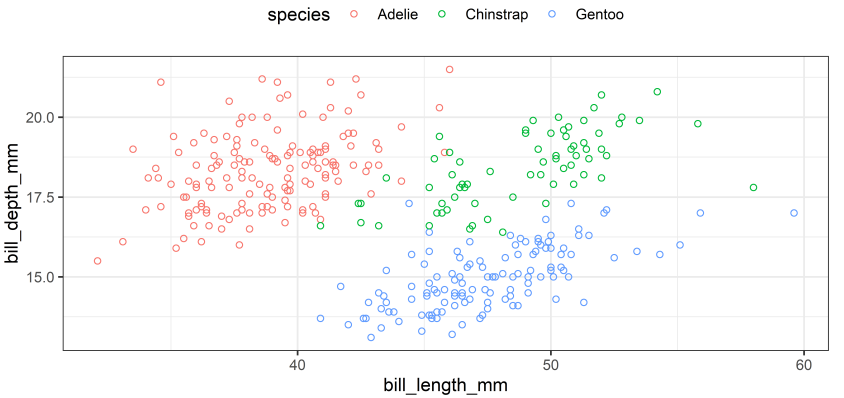
Только `color`, определяет цвет всей фигуры - кружочка.

```
p +
  theme(legend.position = "top") +
  geom_point(aes(color = species), shape = 16)
```



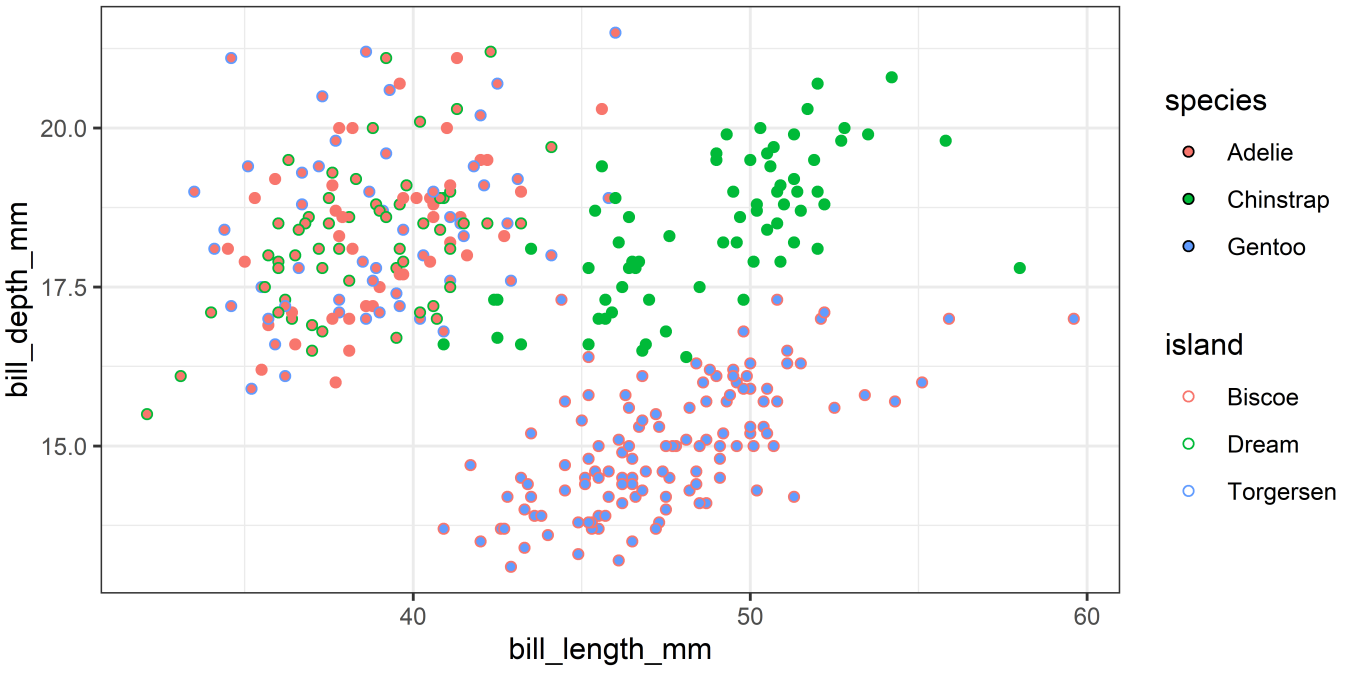
Только `color`, определяет цвет всей фигуры - окружности.

```
p +
  theme(legend.position = "top") +
  geom_point(aes(color = species), shape = 1)
```



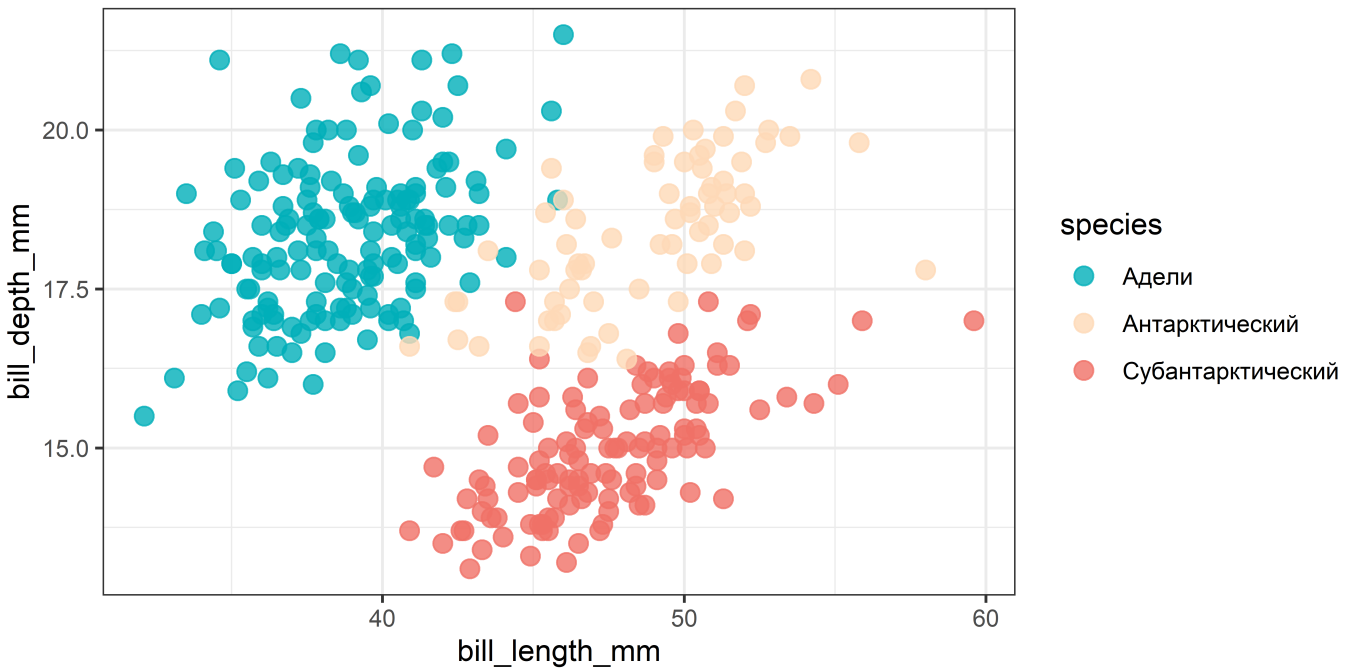
И `color`, и `fill` для обводки и заливки.

```
p +
  geom_point(aes(fill = species, color = island), shape = 21)
```



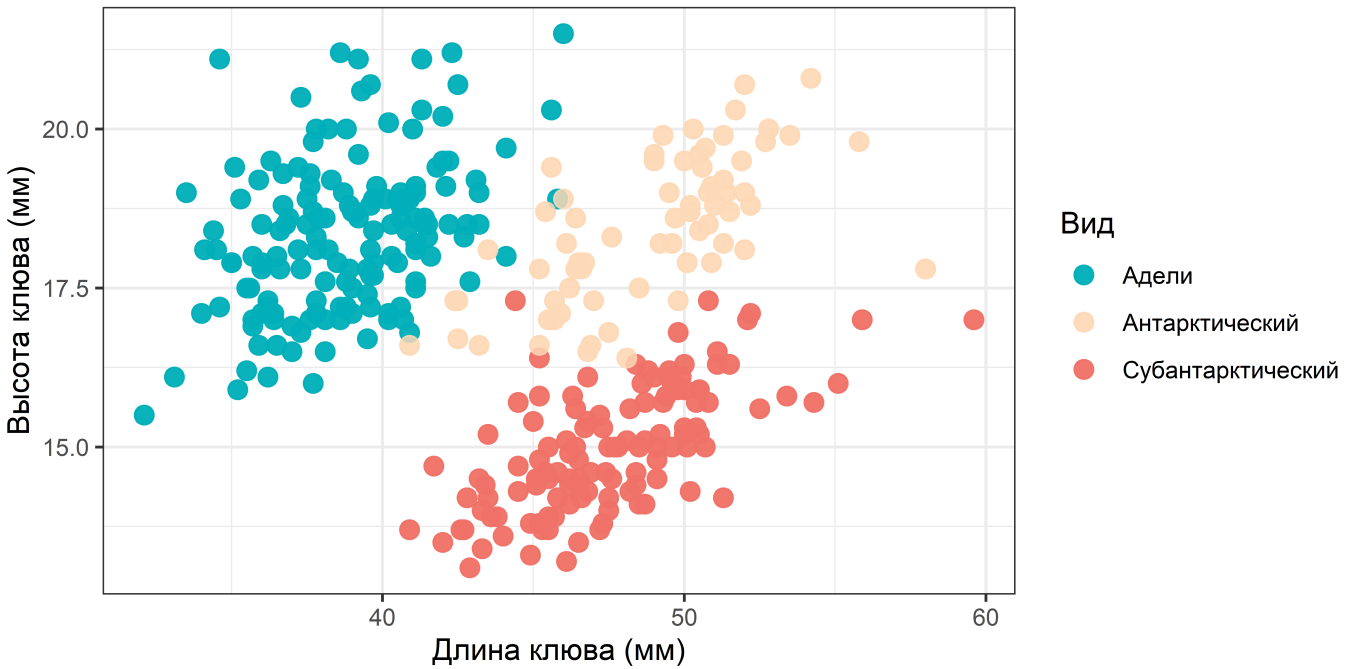
Цветовая шкала

```
cols <- c("#00afb9", "#fed9b7", "#f07167")
p_labels <- c("Адели", "Антарктический", "Субантарктический")

p <- p +
  geom_point(aes(color = species), size = 3, alpha = 0.8) +
  scale_color_manual(values = cols, labels = p_labels)

p
```



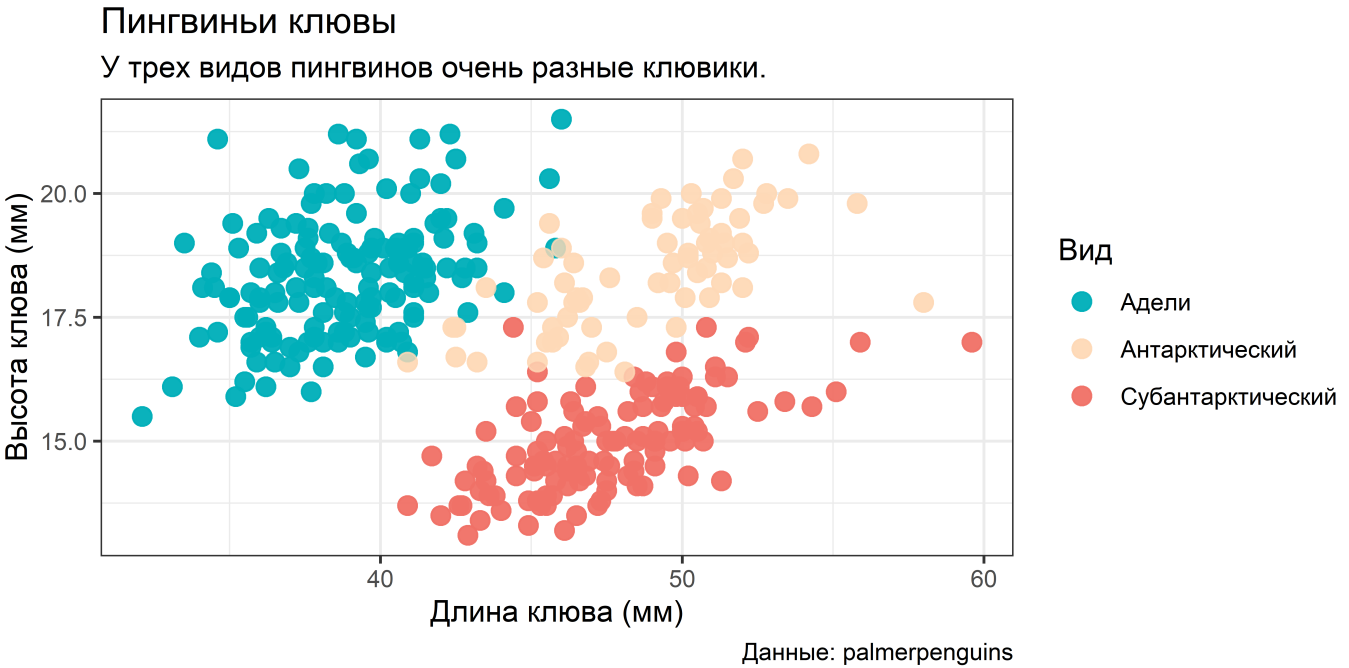
Названия осей

```
p <- p +
  geom_point(aes(color = species), size = 3, alpha = 0.8) +
  labs(x = "Длина клюва (мм)", y = "Высота клюва (мм)", color = "Вид")

p
```



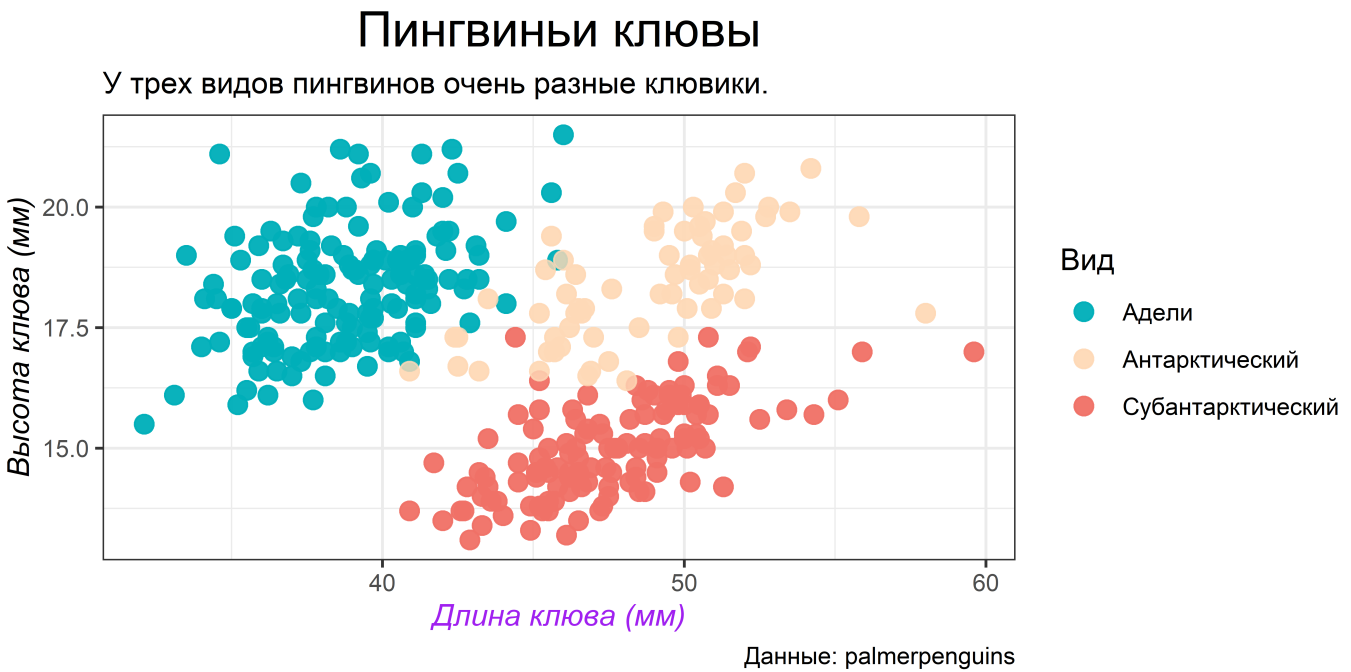
Название графика

```
p <- p +
  labs(title = "Пингвиньи клювы",
       subtitle = "У трех видов пингвинов очень разные клювики.",
       caption = "Данные: palmerpenguins")
p
```



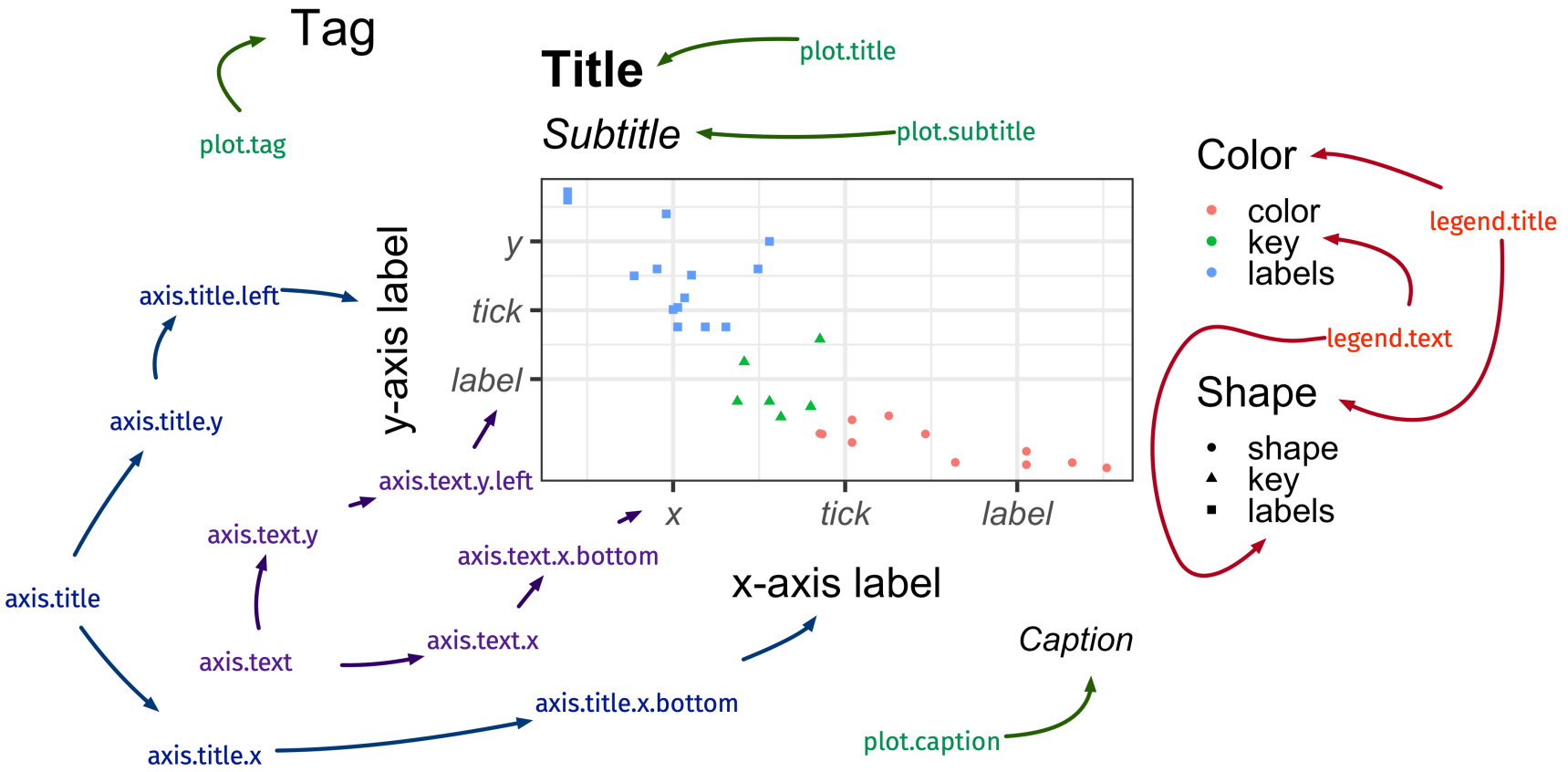
Сделаем красиво...

```
p +
  theme(
  plot.title = element_text(size = 18, hjust = 0.5),
  axis.title = element_text(face = "italic"),
  axis.title.x = element_text(color = "purple"))
```



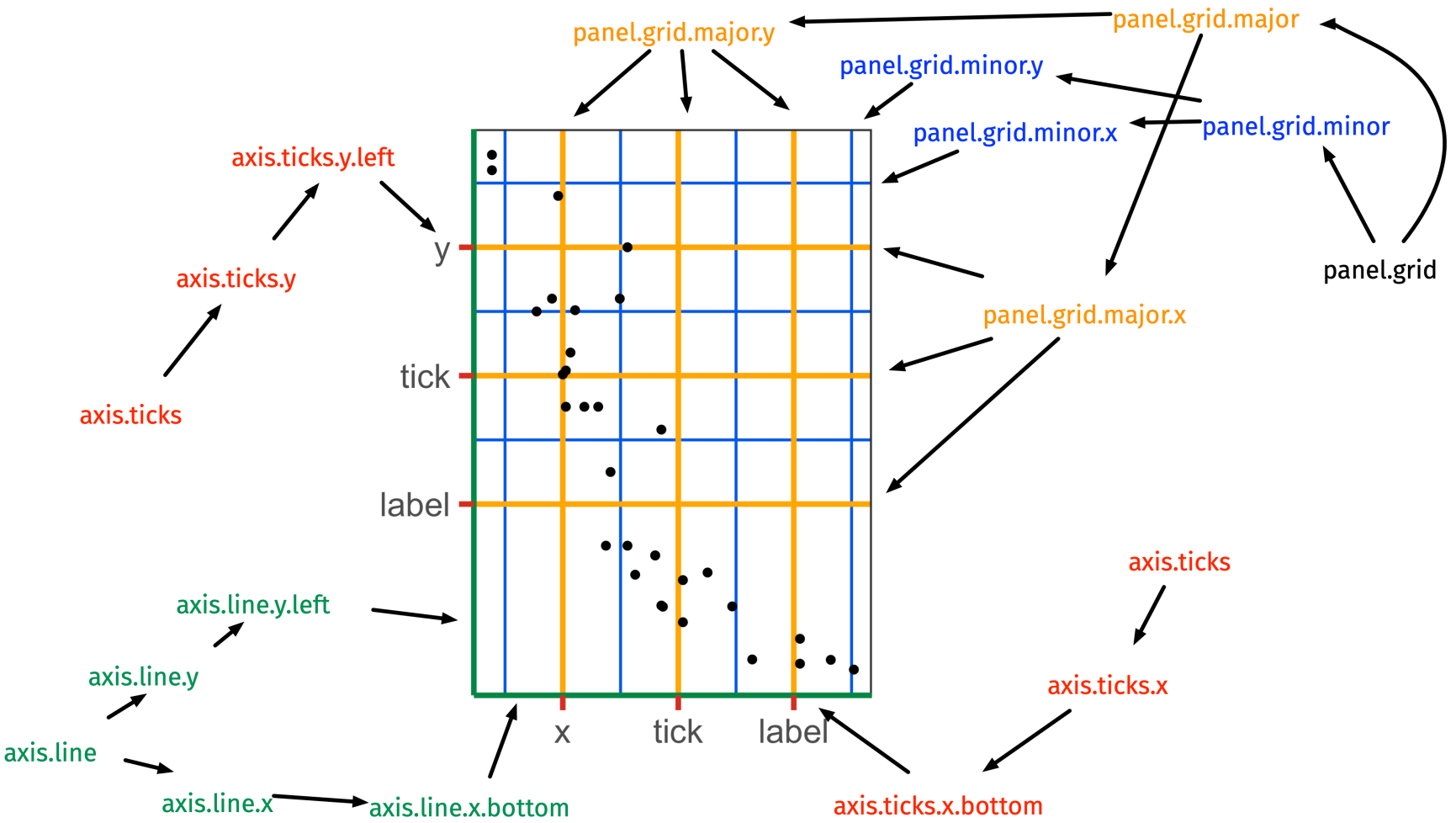
Тема: текстовые элементы

Image credit: Emi Tanaka



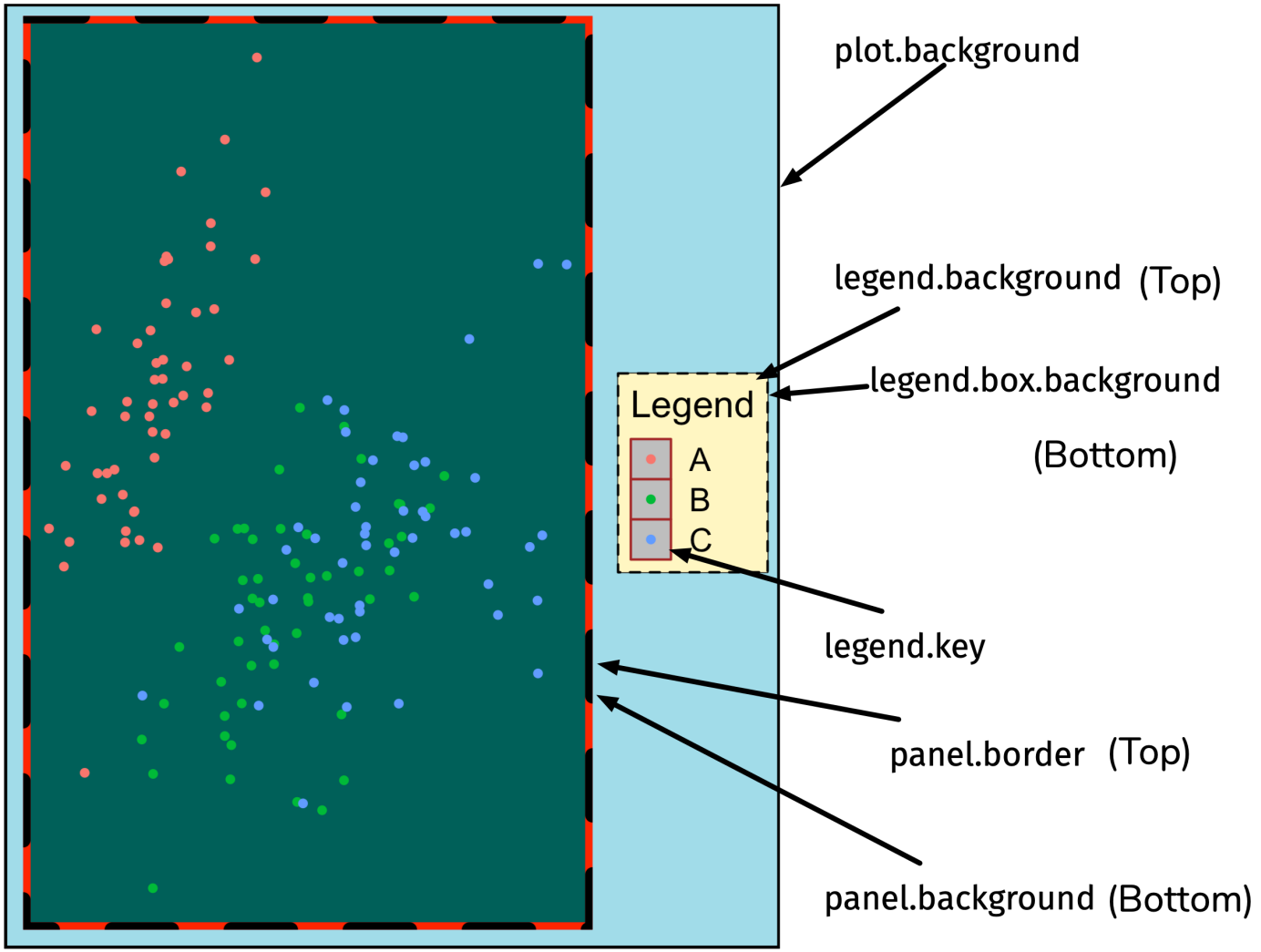
Тема: сетка графика

Image credit: Emi Tanaka



Тема: зоны графика

Image credit: Emi Tanaka



Сохранение в файл

```r
my_plot <- ggplot(...) + ...

# Если не указать переменную с ggplot графиком, то сохранится последний нарисованный график
ggsave("figures/my_plot.png", my_plot)
```

```r
ggsave("figures/my_plot.png", my_plot, dpi = 300, width = 10, height = 10, units = "cm")
```

```r
ggsave("figures/my_plot.pdf", my_plot)
```

```r
pdf("figures/my_plot.pdf") # Открыть файл для записи
my_plot                    # Нарисовать график
dev.off()                  # Закрыть файл
```

Geoms

- `geom_point()`
- `geom_line()`
- `geom_histogram()`
- `geom_boxplot()`
- `geom_bar()` и `geom_col()`
- `geom_text()` и `geom_label()`
- ...

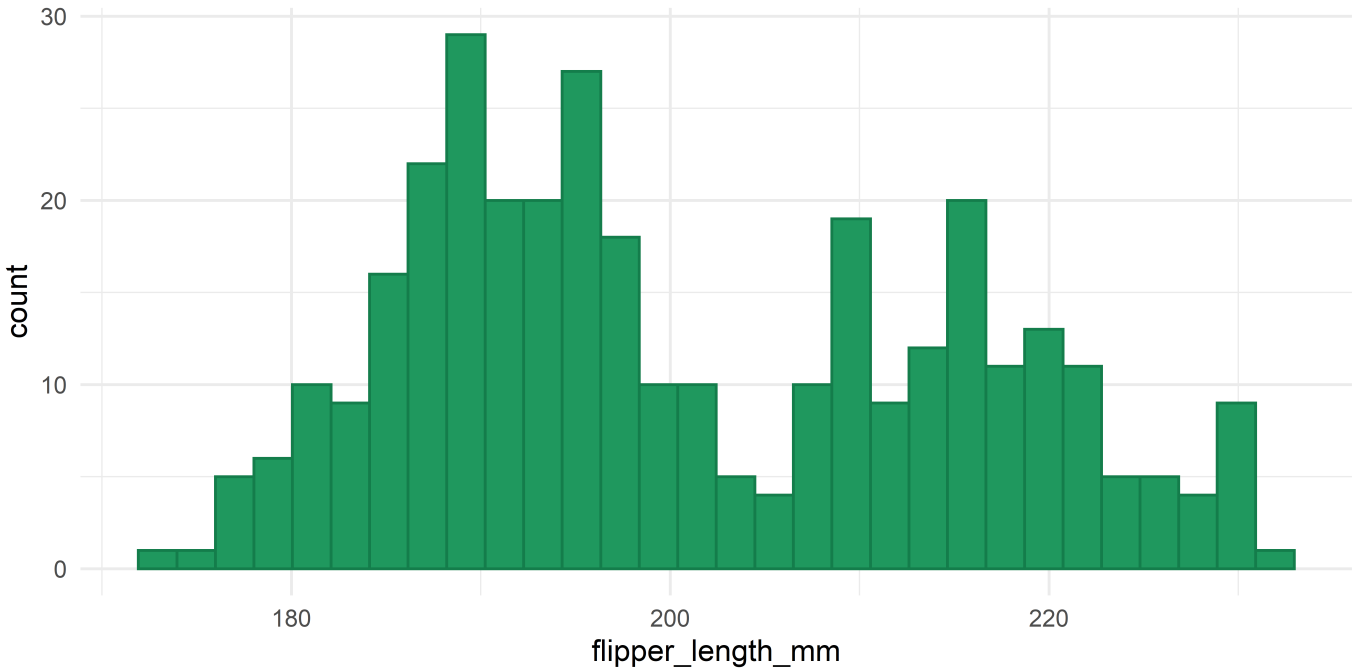
- точковая диаграмма / scatter plot
- линейная диаграмма
- гистограмма
- ящик с усами / боксплот
- столбчатая диаграмма
- текст / подписи
- ...

Как выбрать тип графика под ваши данные

Гистограмма

Ширину бина или их количество можно задать с помощью `binwidth = ...` или `bins = ...`.

```
ggplot(penguins, aes(x = flipper_length_mm)) +
  geom_histogram(color = "#147d4b", fill = "#1f985e") +
  theme_minimal()
```



Гистограмма

```
ggplot(penguins, aes(x = flipper_length_mm, fill = species)) +
  geom_histogram(color = "white") +
  scale_fill_manual(values = c("#1f985e", "#ffc700", "#147d4b")) +
  theme_minimal()
```

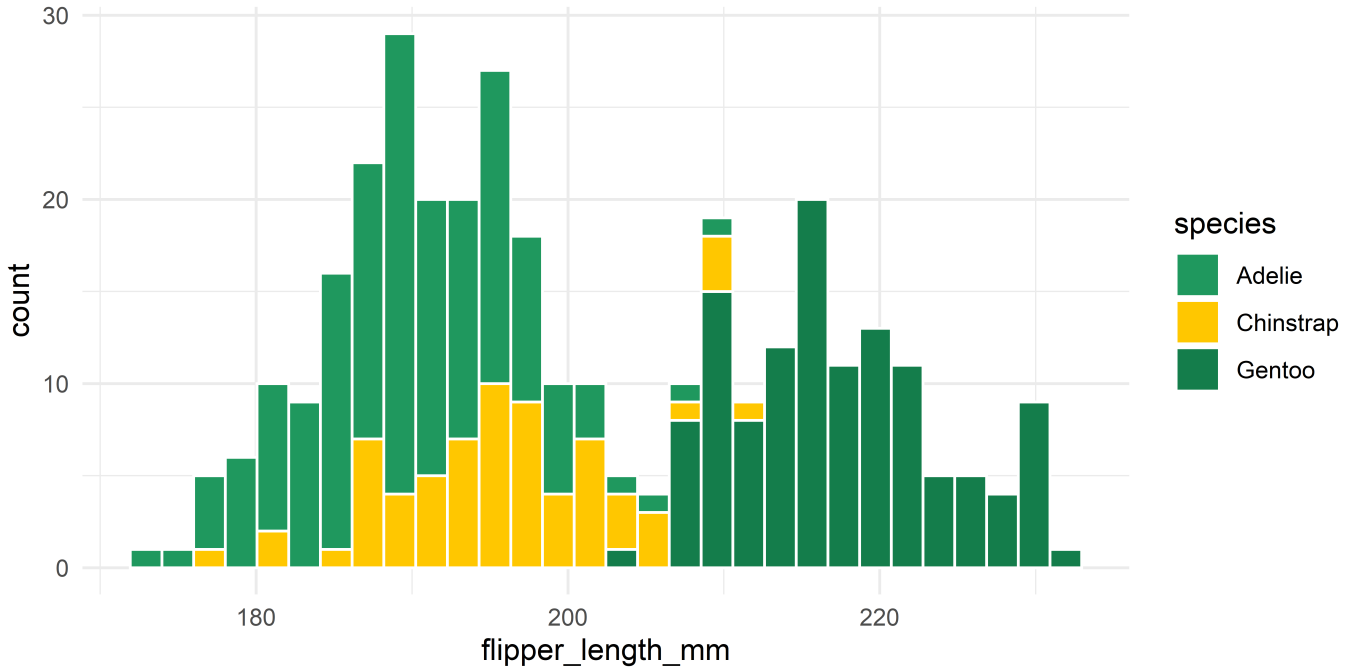
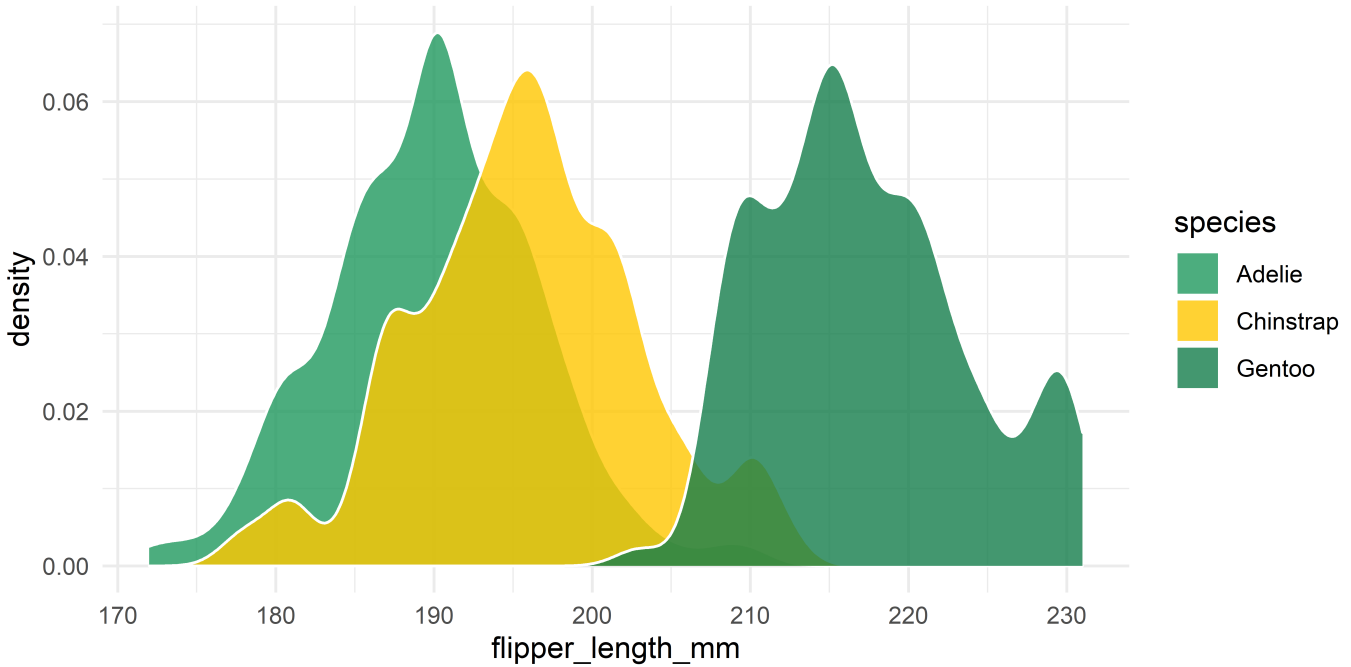


График плотности

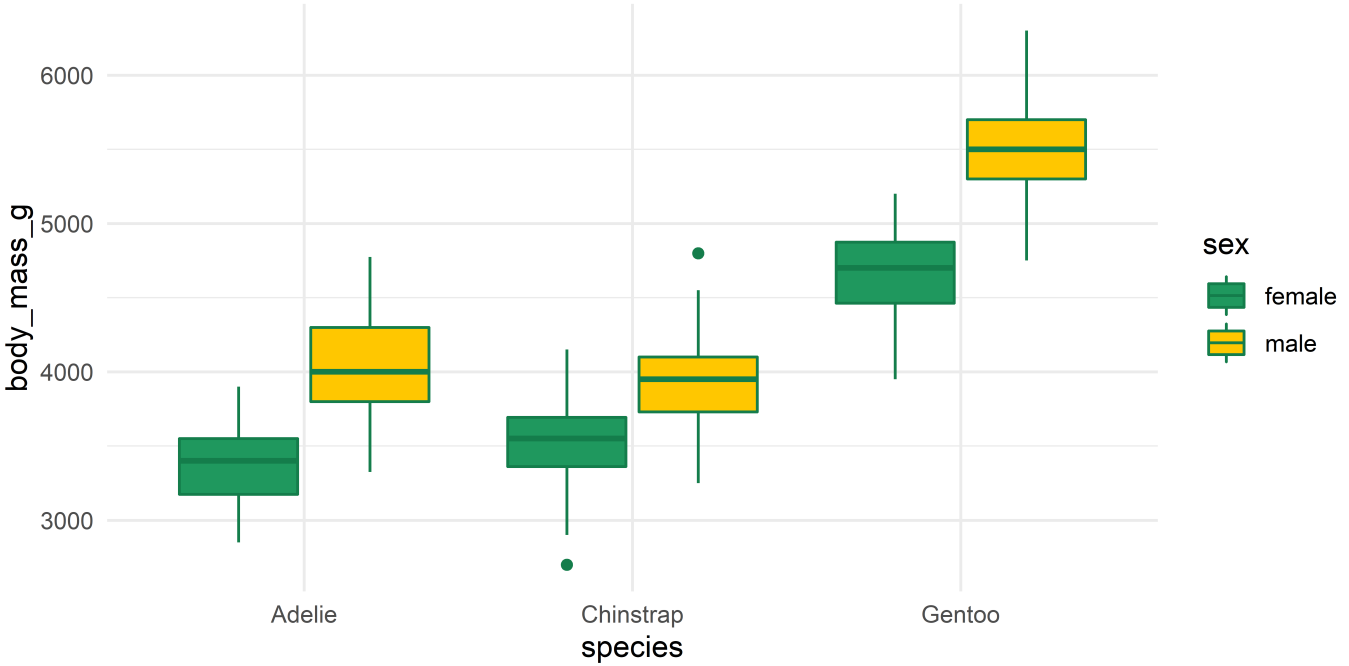
Сглаживание можно регулировать с помощью `bw = ...` и других параметров.

```
ggplot(penguins, aes(x = flipper_length_mm, fill = species)) +
  geom_density(color = "white", alpha = 0.8, bw = 1.5) +
  scale_fill_manual(values = c("#1f985e", "#ffc700", "#147d4b")) +
  theme_minimal()
```



Боксплот

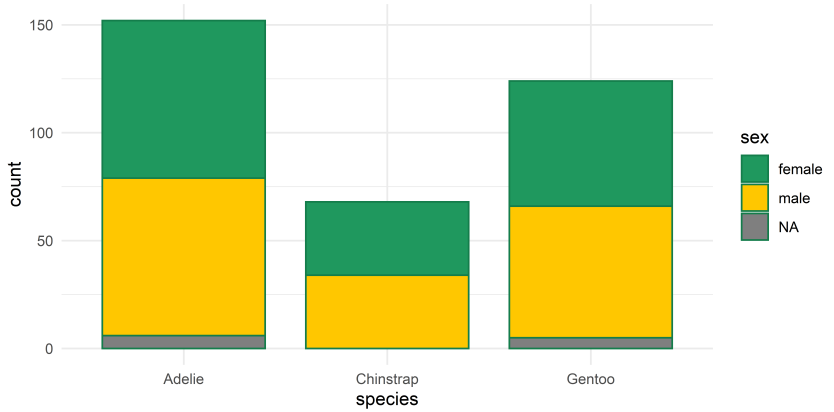
```
ggplot(drop_na(penguins, sex), aes(x = species, y = body_mass_g, fill = sex)) +
  geom_boxplot(color = "#147d4b", width = 0.8) +
  scale_fill_manual(values = c("#1f985e", "#ffc700")) +
  theme_minimal()
```



Столбчатая диаграмма

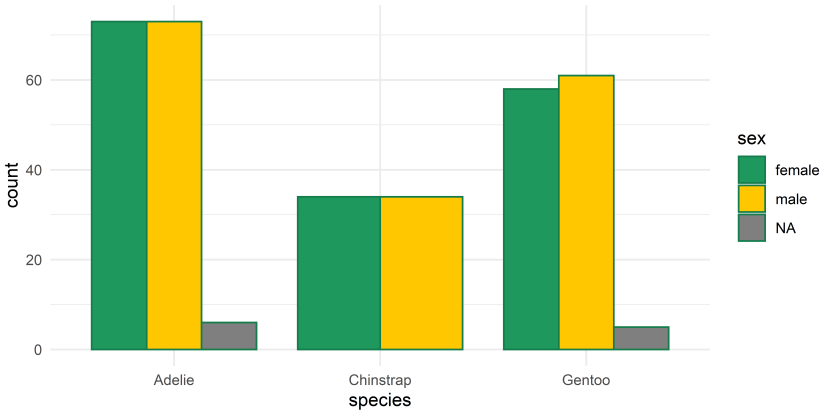
По умолчанию `position = "stack"`

```
ggplot(penguins, aes(x = species, fill = sex)) +
  geom_bar(color = "#147d4b", width = 0.8) +
  scale_fill_manual(values = c("#1f985e",
   "#ffc700")) +
  theme_minimal()
```



Можно поменять на `position = "dodge"`

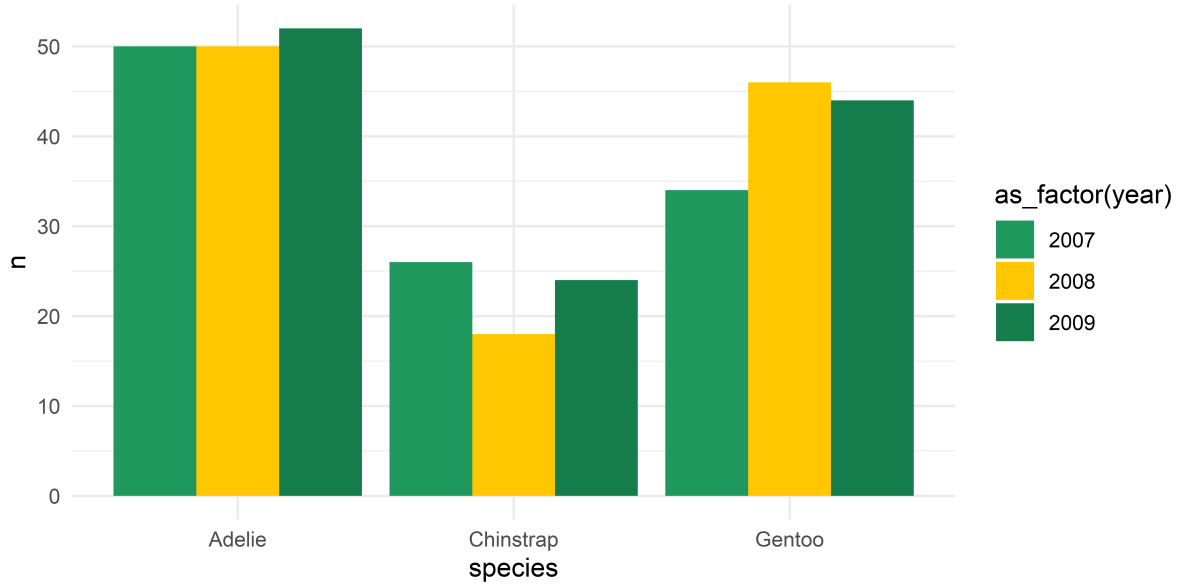
```
ggplot(penguins, aes(x = species, fill = sex)) +
  geom_bar(
    color = "#147d4b", width = 0.8,
    position = "dodge") +
  scale_fill_manual(values = c("#1f985e",
   "#ffc700")) +
  theme_minimal()
```



Столбчатая диаграмма

Если уже есть столбец с числами, которые хотим изобразить, то нужно использовать `geom_bar(stat = "identity")` или `geom_col()`.

```
penguins %>%
  count(species, year) %>%
  ggplot(aes(x = species, y = n, fill = as_factor(year))) +
  geom_col(position = "dodge") +
  scale_fill_manual(values = c("#1f985e", "#ffc700", "#147d4b")) +
  theme_minimal()
```



Что почитать и посмотреть

- ggplot2 cheatsheet
- ggplot2 website
- 'A ggplot2 Tutorial for Beautiful Plotting in R' by Cédric Scherer
- The R Graph Gallery
- Как выбрать график под ваши данные
- Data Visualization Chapter in R4DS
- ggplot2: elegant graphics for data analysis