

Грамматика графиков в Python. Библиотека plotnine.

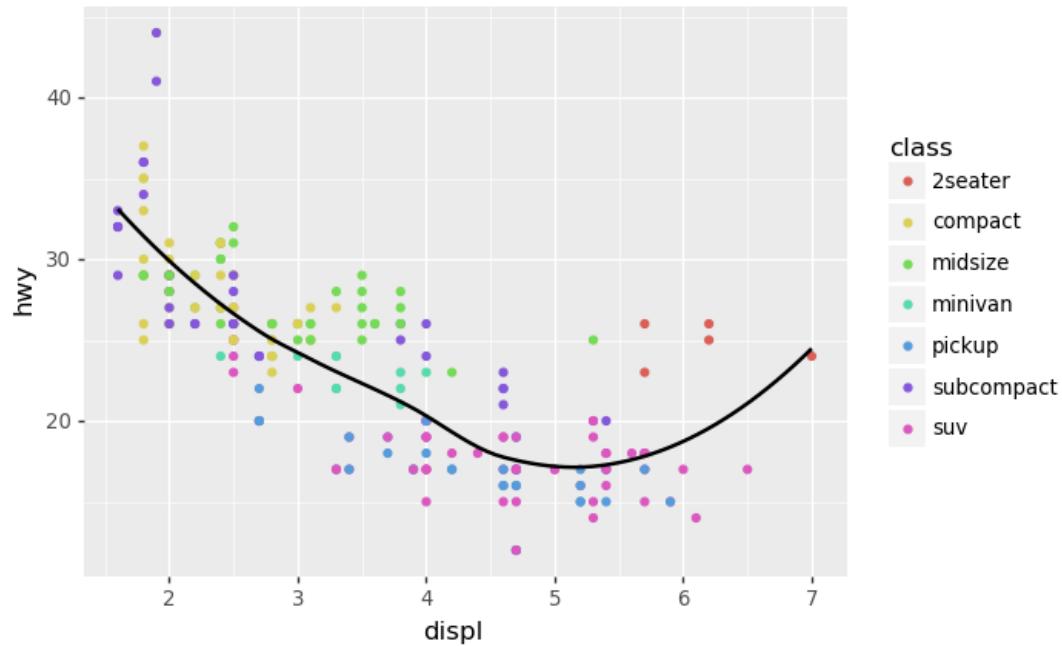
Использование графиков для коммуникации

Коммуникация посредством графиков

- Использование графиков для понимания данных – не единственная задача и, в большинстве случаев, не самоцель. Люди, для которых вы готовите графики, не имеют того же глубокого понимания ваших данных, как у вас. Поэтому важной задачей является сделать ваши графики настолько понятными без дополнительных объяснений, насколько это возможно.
- Для более глубокого погружения в тему можно порекомендовать книгу Alberto Cairo «The Truthful Art»

Заголовки

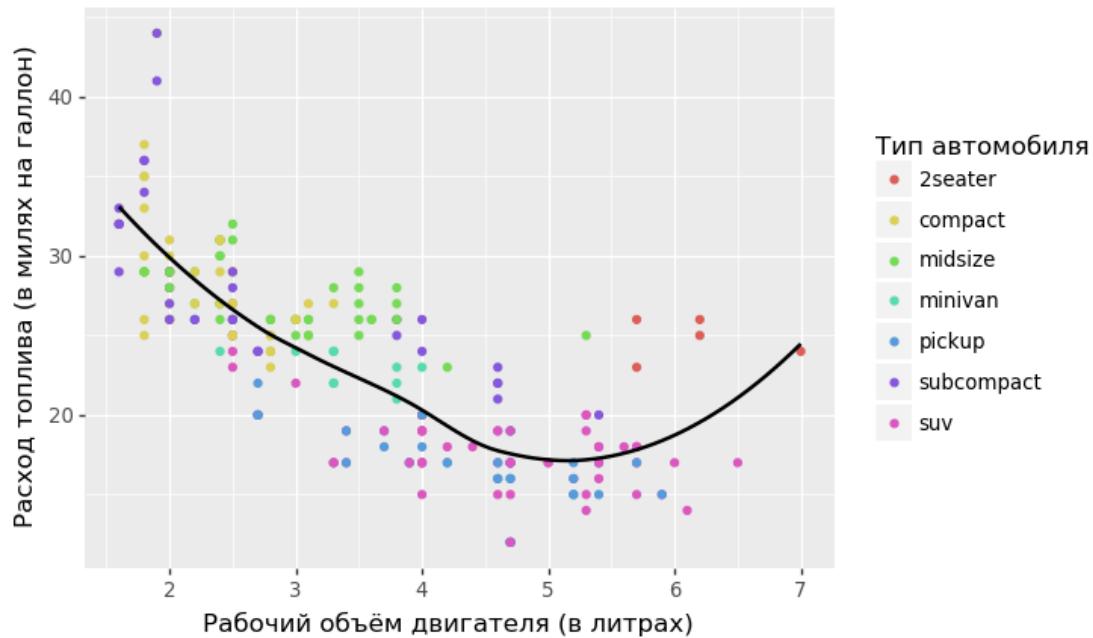
Расход топлива растёт с увеличением объёма двигателя



```
ggplot(mpg, aes("displ", "hwy")) +\\
  geom_point(aes(color="class")) +\\
  geom_smooth(se=False) +\\
  labs(title="Расход топлива растёт с  
увеличением объёма двигателя")
```

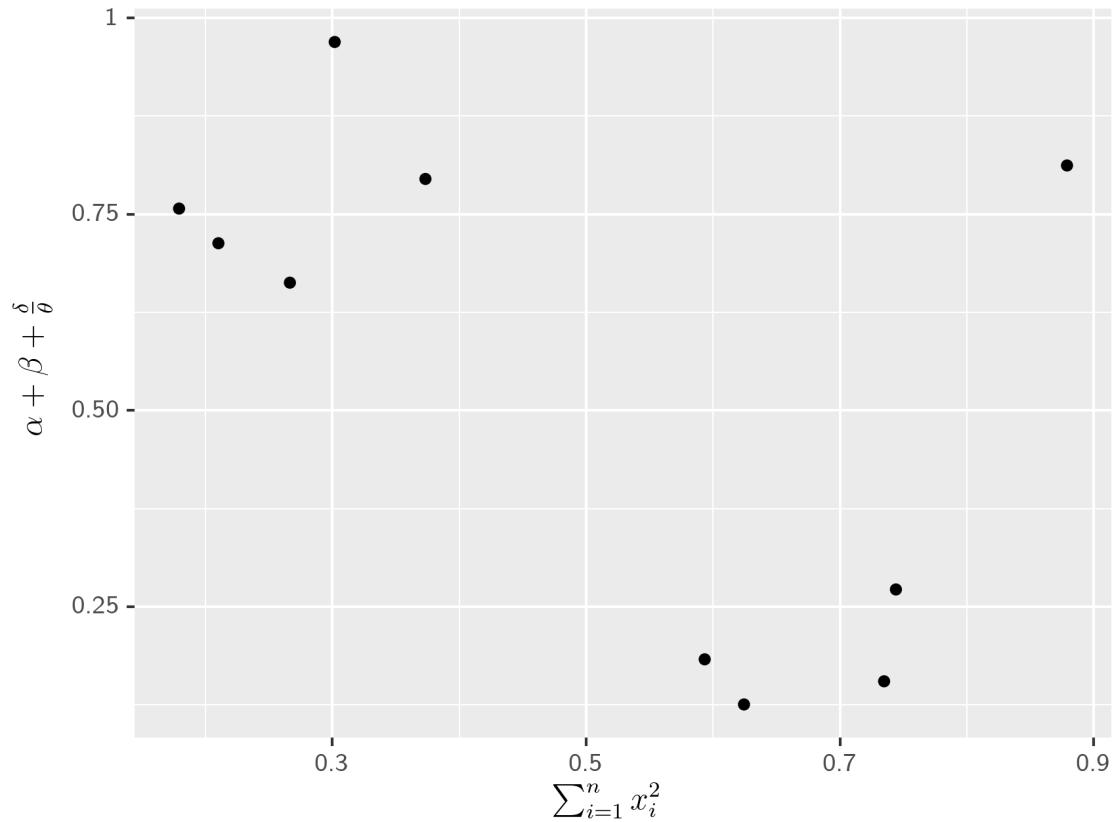
Вообще говоря, лучше избегать заголовков в духе «Зависимость x от у» – это и так понятно. Напишите, что здесь должен увидеть читатель.

Подписи к осям



```
ggplot(mpg, aes("displ", "hwy")) +\n  geom_point(aes(colour="class")) +\n  geom_smooth(se=False) +\n  labs(x="Рабочий объём двигателя (в\nлитрах)",\n       y="Расход топлива (в милях на\nгаллон)",\n       colour="Тип автомобиля")
```

Подписи-формулы.



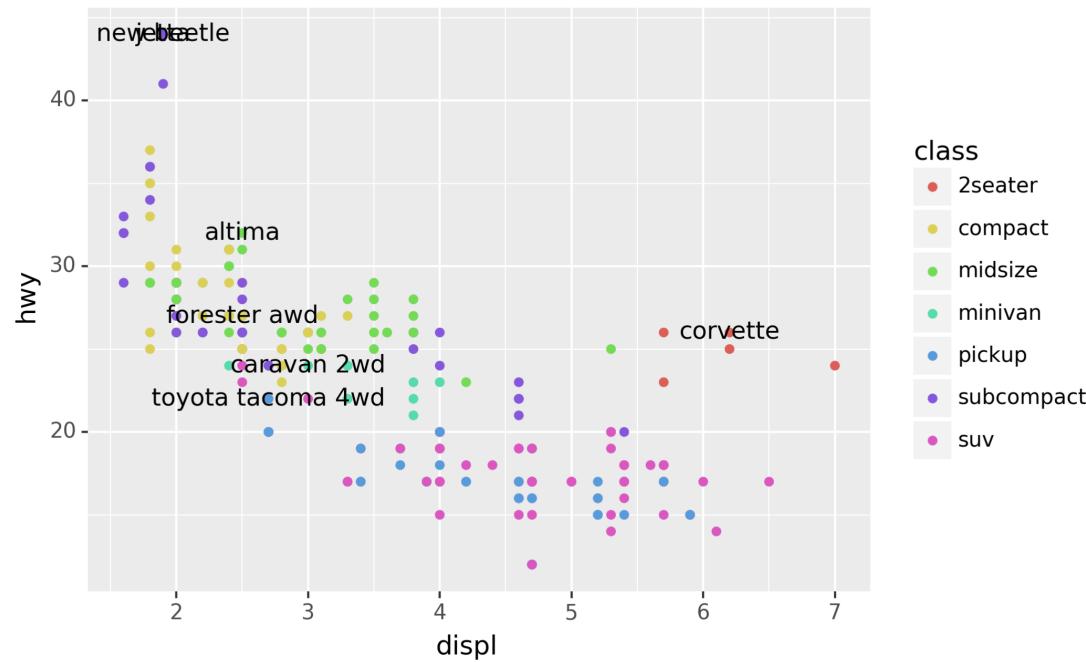
```
from matplotlib import rc
rc('text', usetex=True)

df = pd.DataFrame({ "x": np.random.uniform(size=10),
                     "y": np.random.uniform(size=10) })

ggplot(df, aes("x", "y")) +\
  geom_point() +\
  labs(x="$\\sum_{i = 1}^n x_i^2$",
       y="$\\alpha + \\beta + \\\frac{\\delta}{\\theta}$")
```

Очень удобно, если вы умеете в TeX.

Аннотации...

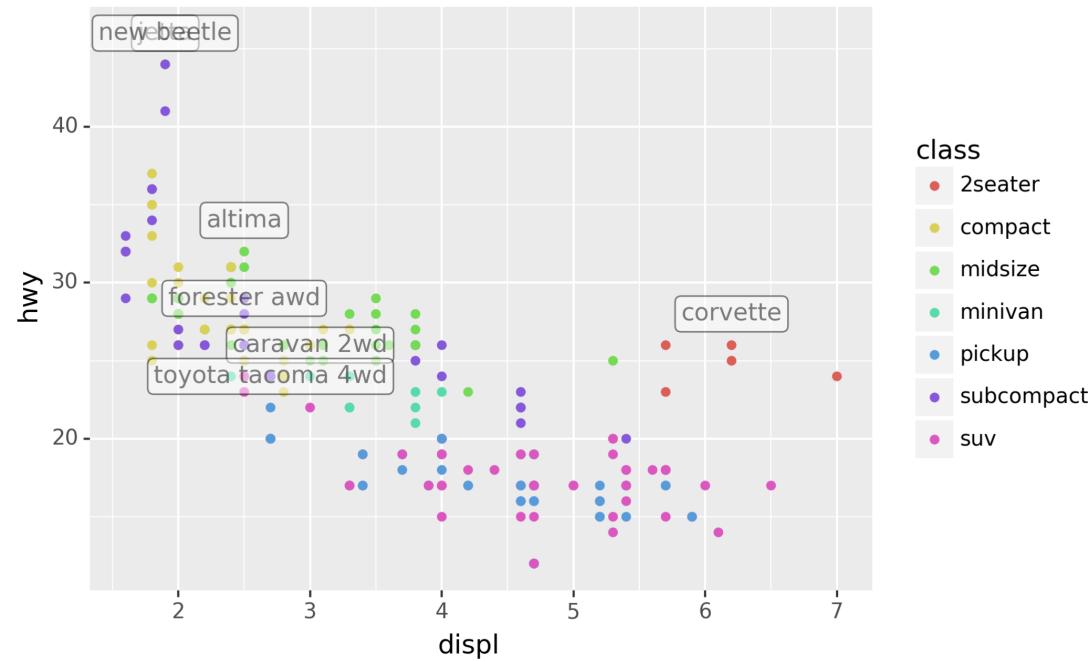


```
best_in_class = mpg\  
.sort_values(by="hwy",  
ascending=False) \  
.groupby("class") \  
.first()
```

```
ggplot(mpg, aes("displ", "hwy")) +\  
geom_point(aes(colour="class")) +\  
geom_text(aes(label="model"),  
data=best_in_class)
```

Это просто ужасно...

... Аннотации...

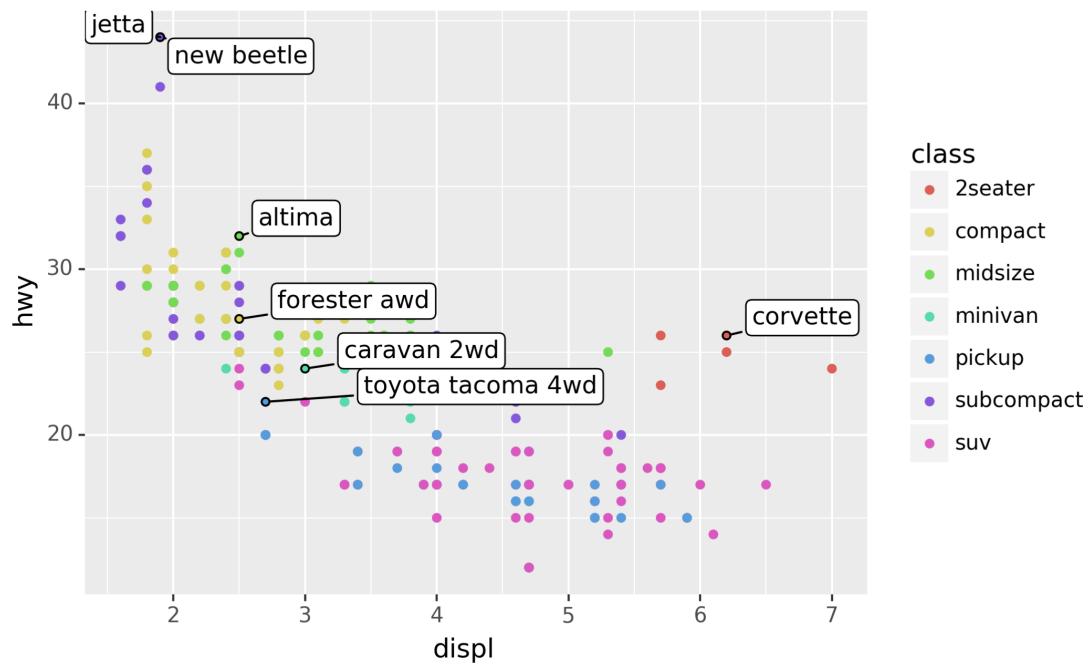


Давайте сделаем их чуть прозрачнее и с рамкой

```
ggplot(mpg, aes("displ", "hwy")) +\n  geom_point(aes(colour="class")) +\n  geom_label(aes(label="model"),\n             data=best_in_class, nudge_y=2,\n             alpha=0.5)
```

Уже лучше. Но всё равно не очень...

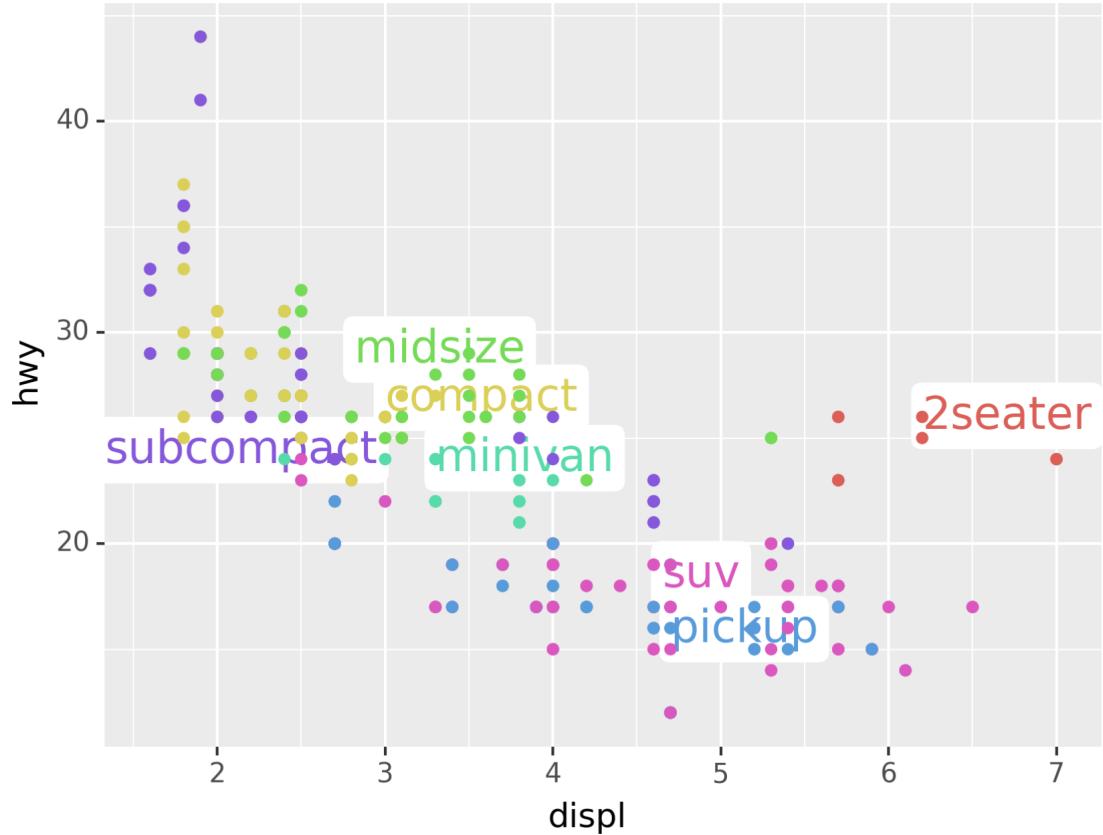
... Аннотации!



```
ggplot(mpg, aes("displ", "hwy")) +\n  geom_point(aes(colour="class")) +\n  geom_point(data=best_in_class,\n             fill='none') +\n  geom_label(aes(label="model"),\n             data=best_in_class, adjust_text={\n               'expand_points': (1.5, 1.5),\n               'arrowprops': {\n                 'arrowstyle': '-'\n               } })
```

Так явно лучше!

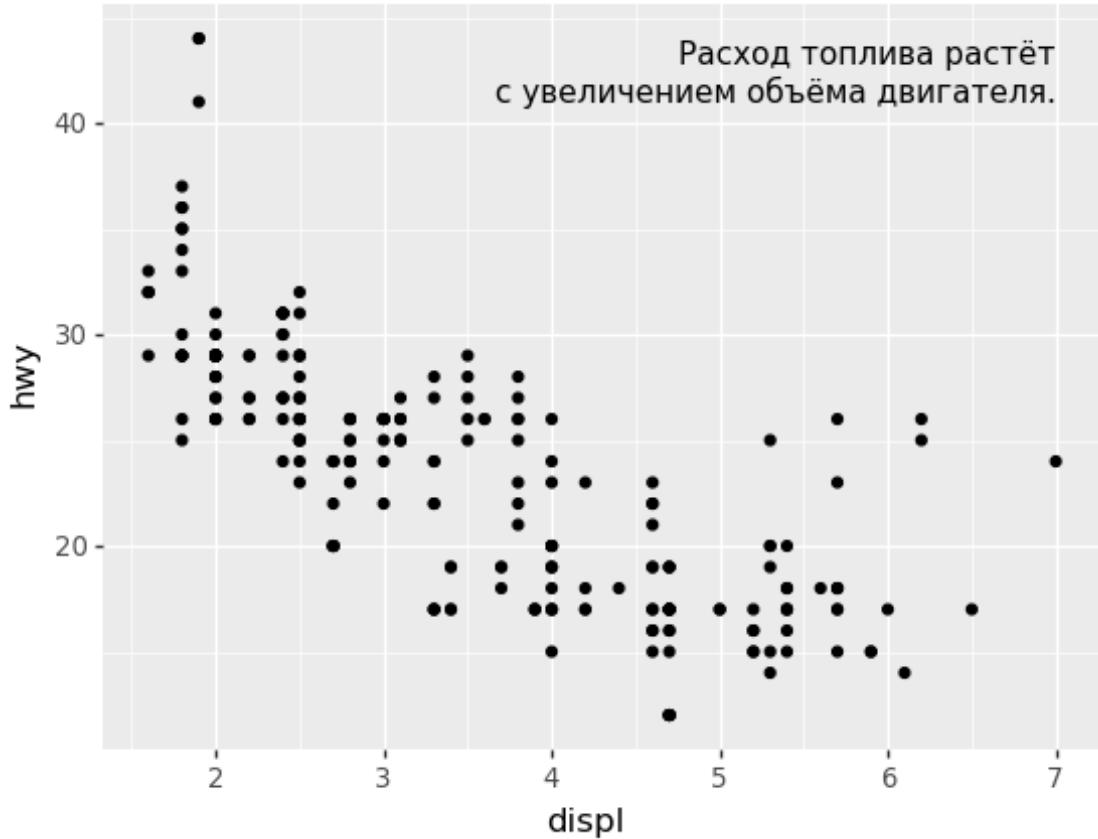
Ещё немногого аннотаций



```
class_avg =  
mpg.groupby("class") ["displ", "hwy"] \\\n.median().reset_index()  
  
ggplot(mpg, aes("displ", "hwy",  
colour="class")) + geom_point() +\\  
geom_label(aes(label="class"),  
data=class_avg, size=16,  
label_size=0,  
adjust_text={'expand_points': (0,  
0)} ) +\\  
geom_point() +\\  
theme(legend_position="none")
```

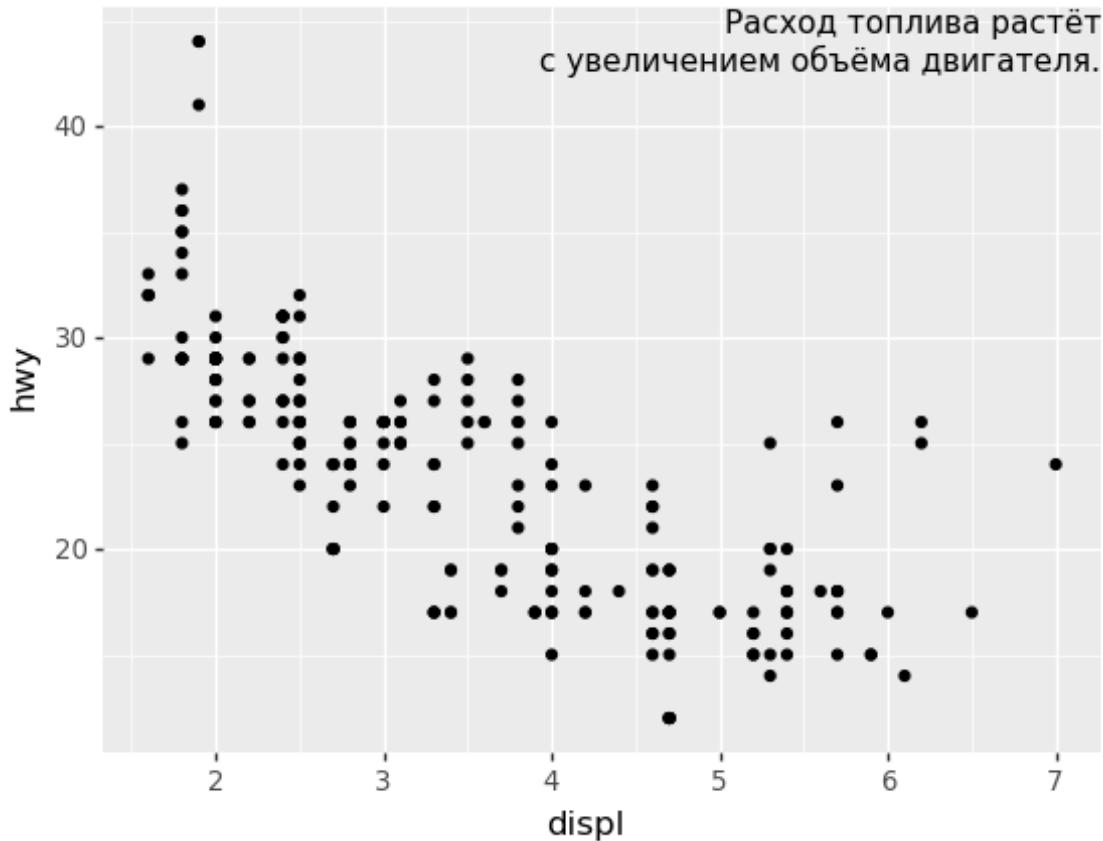
Можно ещё так. Тоже неидеально, но хотя бы не мешает читать график

Заголовок как аннотация



```
label = pd.DataFrame({"displ": [mpg.displ.max()],
                      "hwy": [mpg.hwy.max()],
                      "label": "Расход топлива растёт\nс увеличением объёма двигателя."})  
  
ggplot(mpg, aes("displ", "hwy")) +\
  geom_point() +\
  geom_text(aes(label="label"),
            data=label, va="top", ha="right")  
  
Единственная сложность такого варианта: придётся  
создать отдельный датафрейм для всей побочной  
информации.
```

Аннотация-заголовок



```
label = pd.DataFrame({"displ":  
[np.Inf],  
"hwy":  
[np.Inf],  
"label":  
"Расход топлива растёт\nс  
увеличением объёма двигателя"})  
  
ggplot(mpg, aes("displ", "hwy")) +\\  
geom_point() +\\  
geom_text(aes(label="label"),  
data=label, va="top", ha="right")
```

Если вдруг очень хочется сделать текст в упор в верхний правый угол.

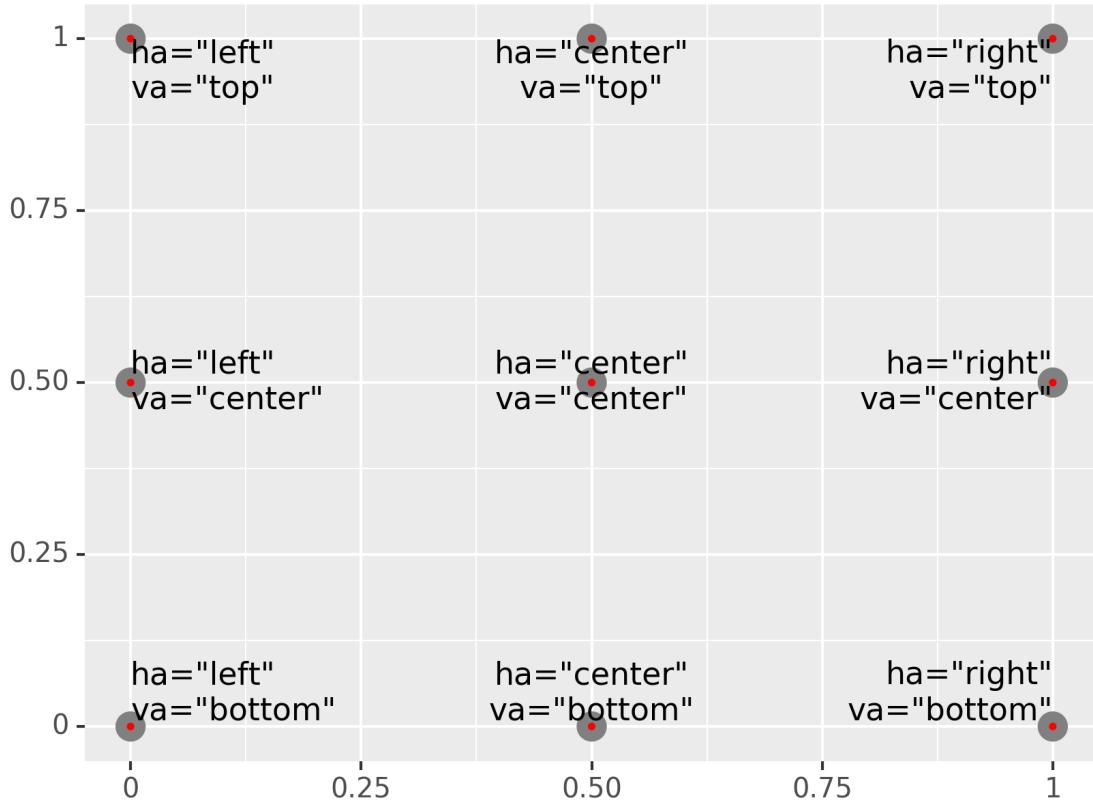
Переносы текста внутри текстовых блоков

Для того, чтобы не делать переносы текста вручную можно задать ширину и воспользоваться `textwrap.fill`

```
from textwrap import fill

print(fill("Расход топлива растёт с увеличением объёма двигателя.", width=40))
Расход топлива растёт с увеличением
объёма двигателя.
```

Размещение текста на графике

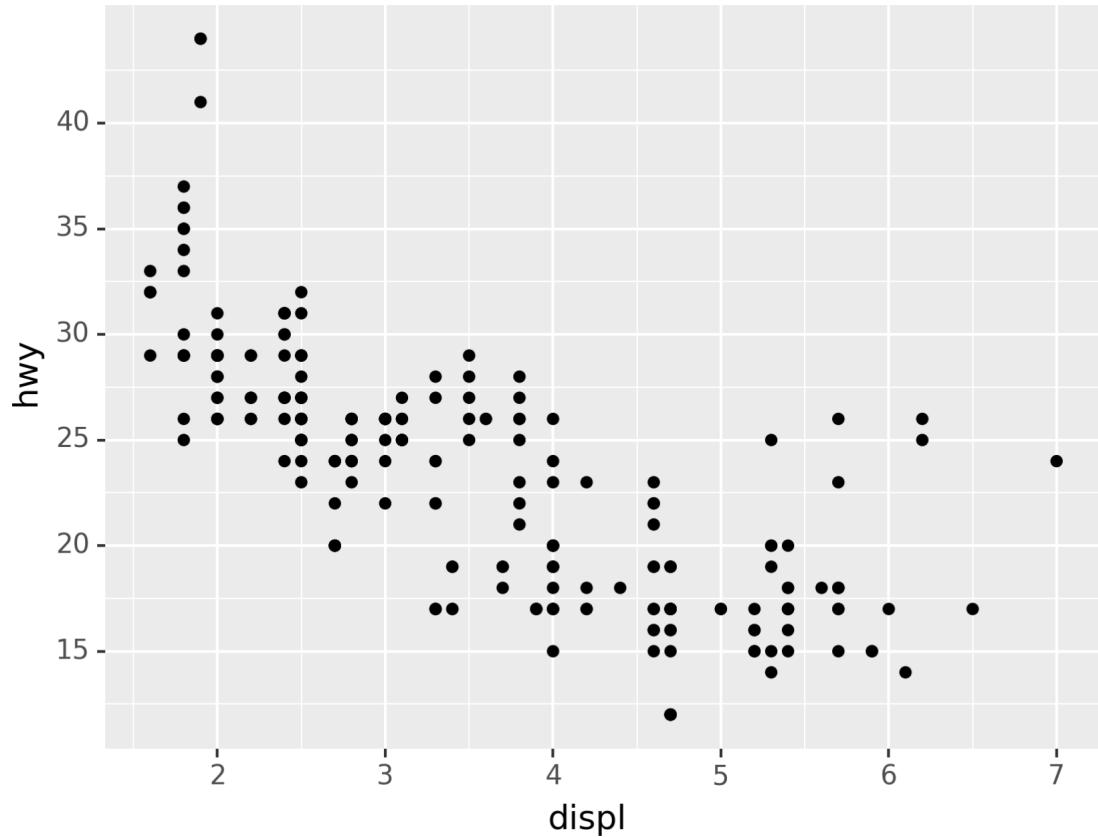


```
label = pd.DataFrame({"displ": [mpg.displ.max()],  
                      "hwy": [mpg.hwy.max()],  
                      "label": "Increasing engine size is \\nrelated  
                      to decreasing fuel economy."})  
  
ggplot(mpg, aes("displ", "hwy")) +\\  
  geom_point() +\\  
  geom_text(aes(label="label"),  
            data=label, va="top", ha="right")
```

Ещё несколько способов аннотирования

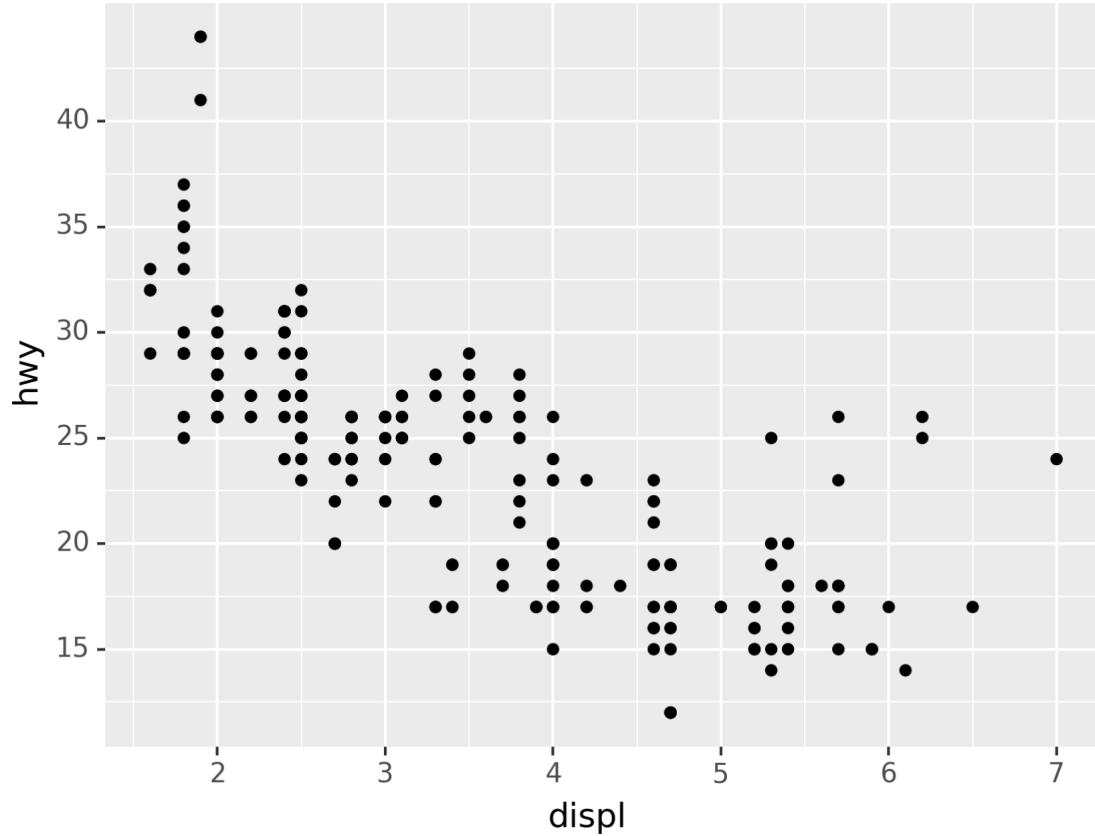
- Кроме `geom_text()` есть ещё несколько способов аннотирования текста с использованием `plotnine`:
- `geom_hline()` и `geom_vline()` можно использовать как линии пороговых значений. Удобно делать их достаточно широкими (`size=2`) и белыми (`colour="white"`), а затем рисовать под основным слоем. Тогда их будет хорошо видно, но они не будут отвлекать внимание.
- `geom_rect()` можно нарисовать для выделения области интереса. Его параметры: `xmin`, `xmax`, `ymin`, `ymax`.
- `geom_segment()` со стрелкой можно использовать, чтобы обратить внимание на конкретную точку. Для этого задаются `x` и `y` как стартовые, и `xend` и `yend` как конечные координаты.

Масштабы осей (шкалы, scales)



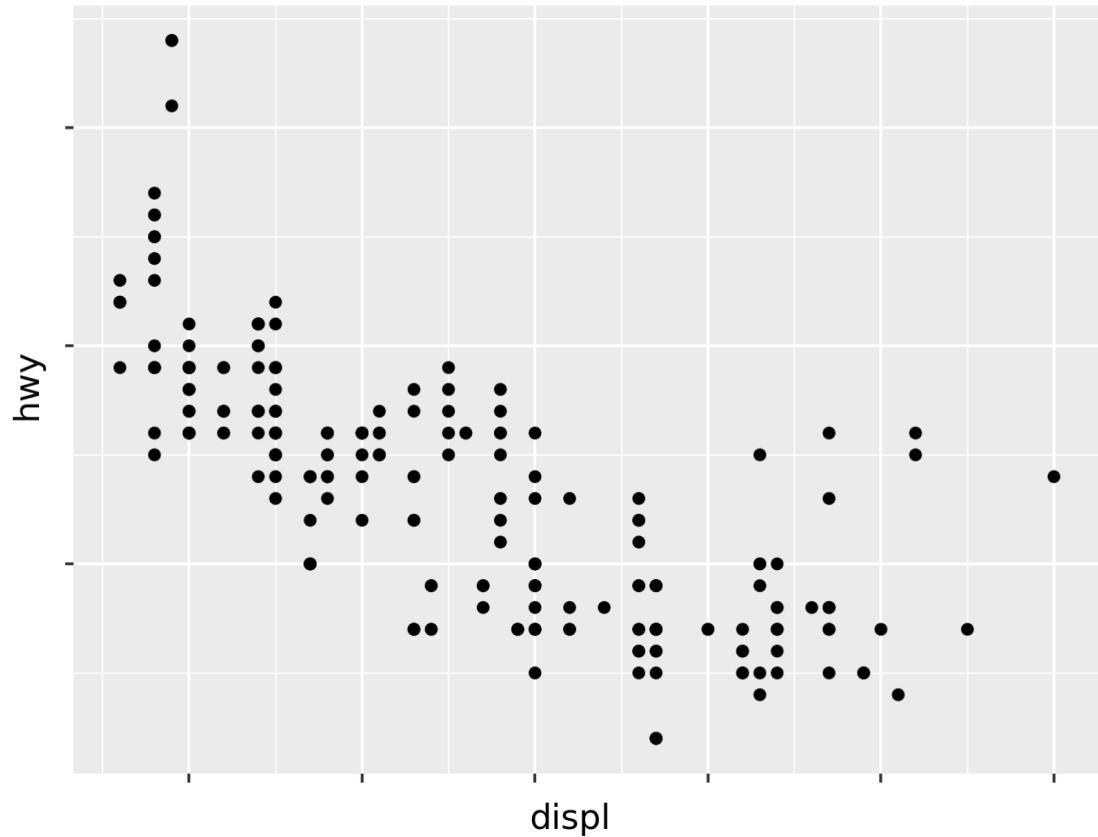
```
ggplot(mpg, aes("displ", "hwy")) +\
  geom_point(aes(colour="class")) +\
scale_x_continuous() +\
scale_y_continuous() +\
scale_colour_discrete()
```

Засечки на осях и легенды



```
ggplot(mpg, aes("displ", "hwy")) +\\
  geom_point() +\\
  scale_y_continuous(breaks=range(15,  
45, 5))
```

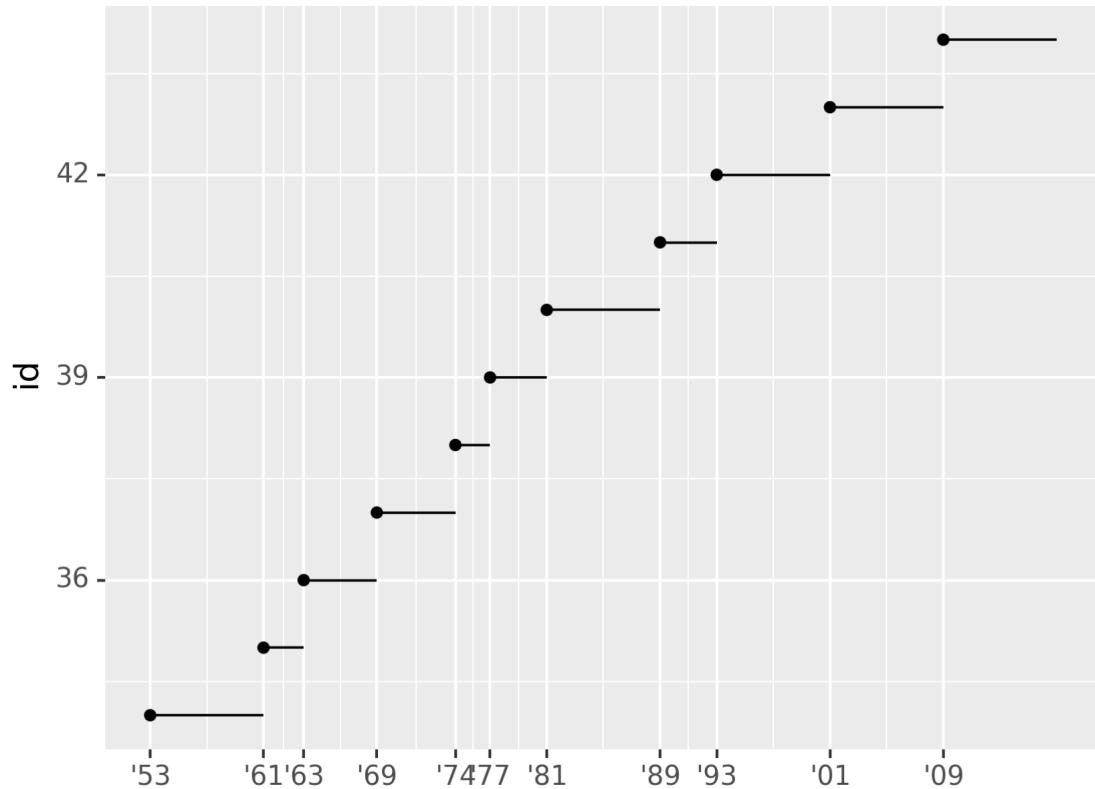
Засечки без значений



```
def no_labels(values):
    return [""] * len(values)

ggplot(mpg, aes("displ", "hwy")) +\
geom_point() +\
scale_x_continuous(labels=no_labels) +\
scale_y_continuous(labels=no_labels)
```

Засечки на осях на основе данных

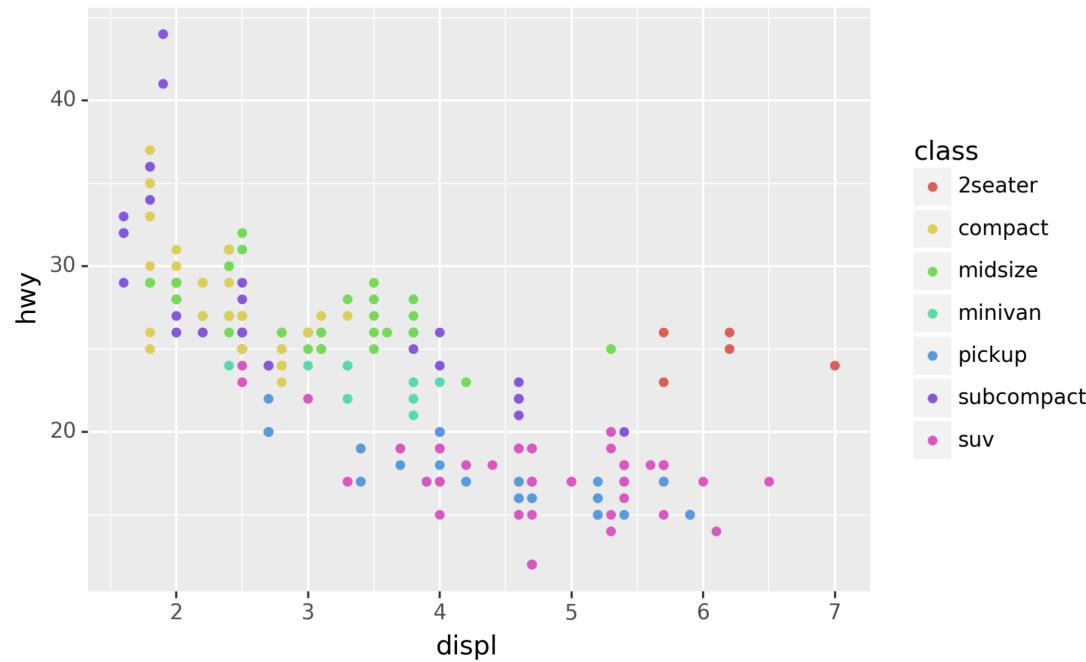


Иногда удобно делать засечки на основе тех данных, которые визуализируются.

```
presidential["id"] = 34 +  
presidential.index
```

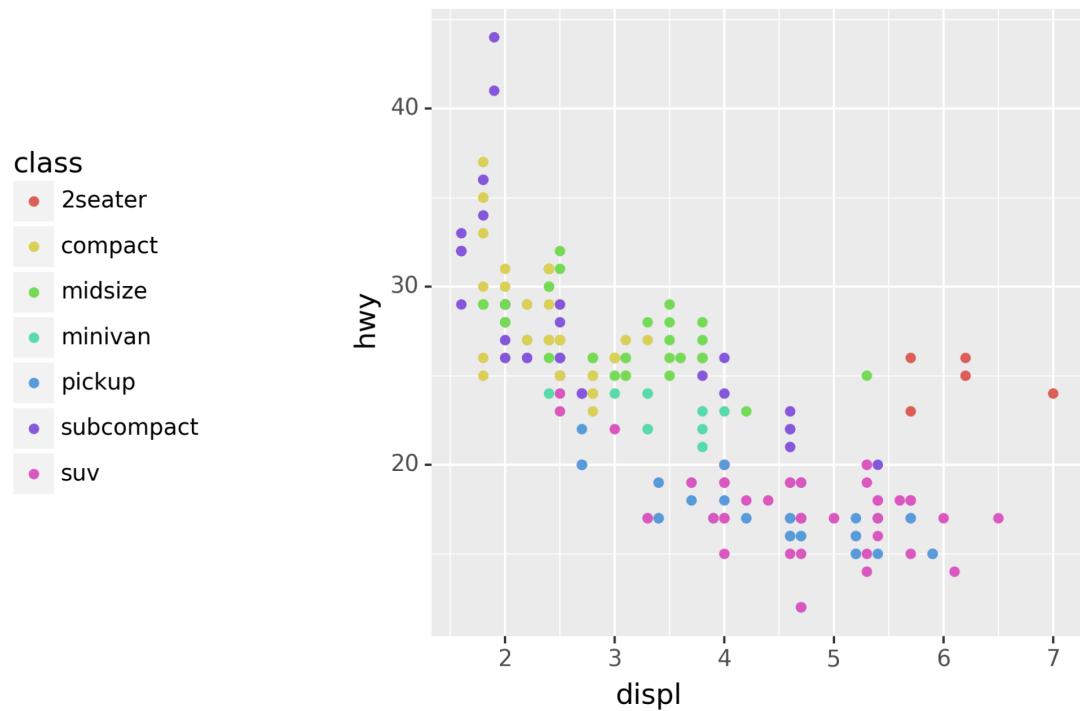
```
ggplot(presidential, aes("start",  
"id")) +\  
geom_point() +\  
geom_segment(aes(xend="end",  
yend="id")) +\  
scale_x_date(name="",  
breaks=presidential.start,  
date_labels='%y')
```

Расположение легенды справа (default)



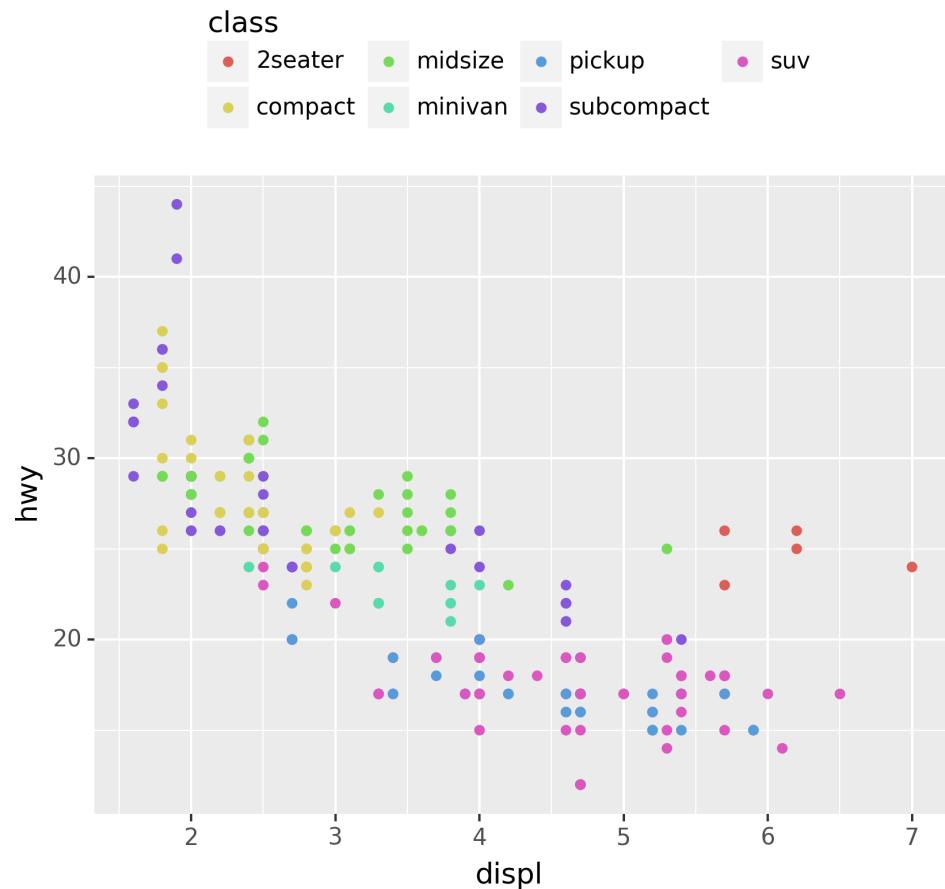
```
ggplot(mpg, aes("displ", "hwy")) +\\
  geom_point(aes(colour="class")) +\\
  theme(legend_position="right")
```

Расположение легенды слева



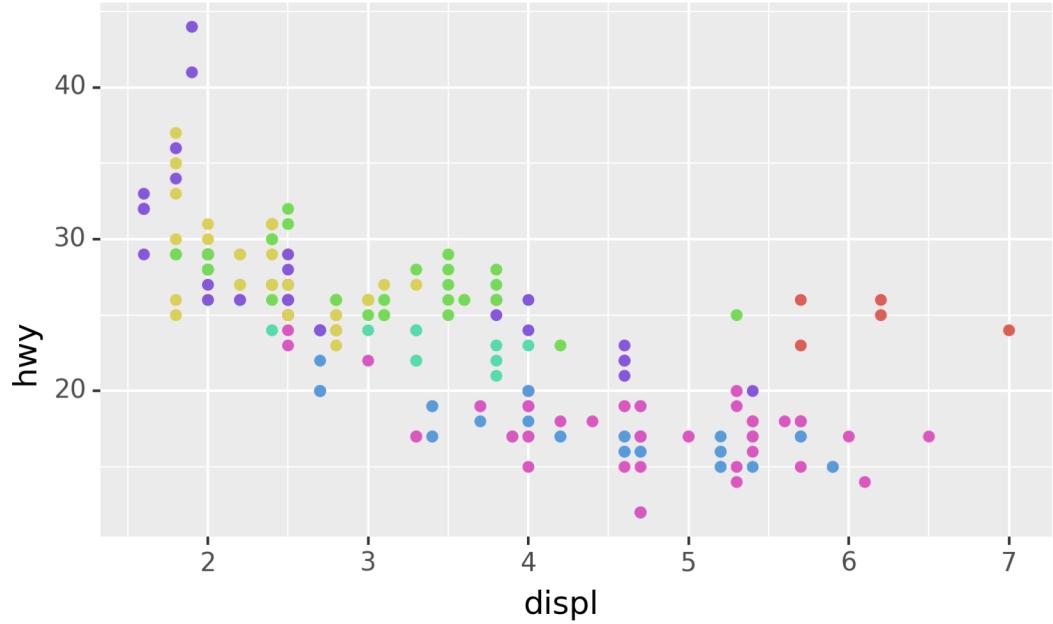
```
ggplot(mpg, aes("displ", "hwy")) +\n  geom_point(aes(colour="class")) +\n  theme(subplots_adjust={'left': 0.3})\n  + theme(legend_position=(0, 0.5))
```

Расположение легенды сверху



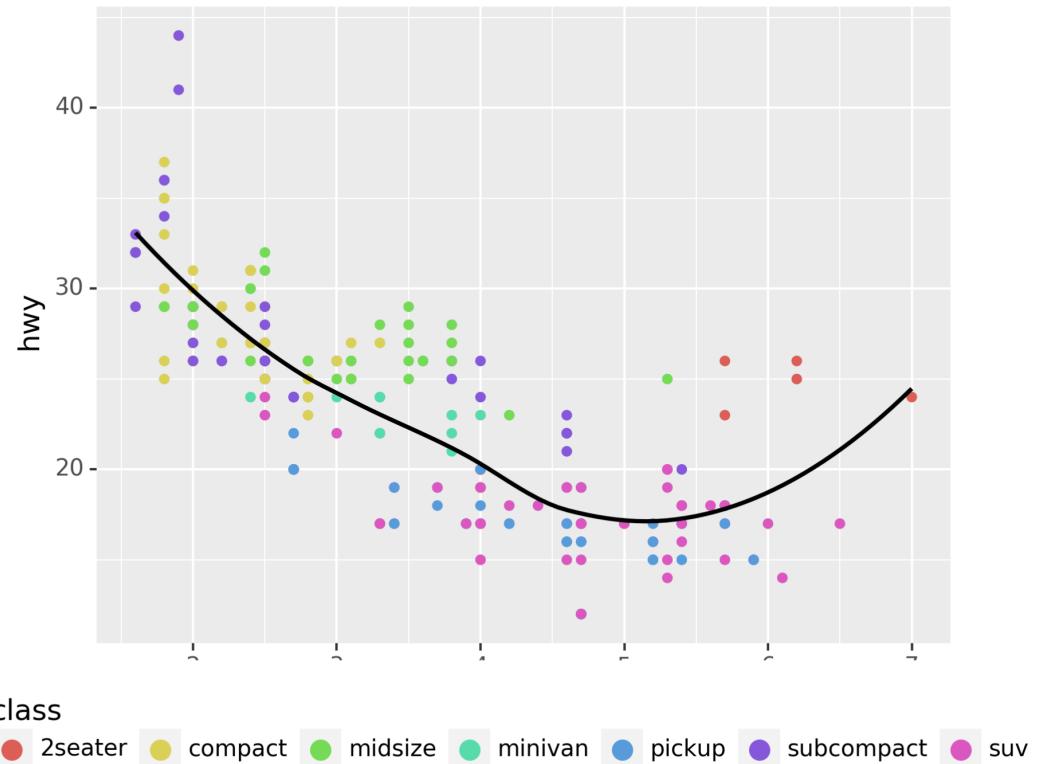
```
ggplot(mpg, aes("displ", "hwy")) +\\
  geom_point(aes(colour="class")) +\\
  theme(legend_position="top")
```

Расположение легенды снизу



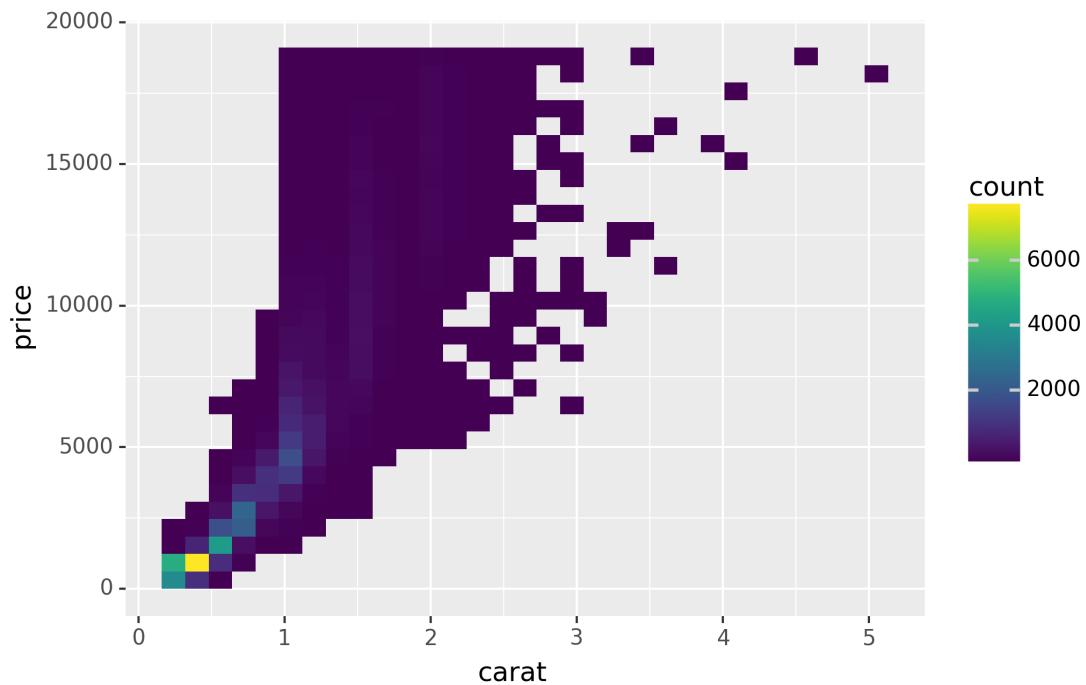
```
ggplot(mpg, aes("displ", "hwy")) +\
  geom_point(aes(colour="class")) +\
  theme(subplots_adjust={ 'bottom': 0.3}, legend_position=.5, 0),  
  legend_direction='horizontal')
```

Изменение параметров легенды



```
ggplot(mpg, aes("displ", "hwy")) +\\
  geom_point(aes(colour="class")) +\\
  geom_smooth(se=False) +\\
  theme(legend_position="bottom") +\\
  guides(colour=guide_legend(nrow=1,\\
    override_aes={"size": 4}))
```

Линейная шкала

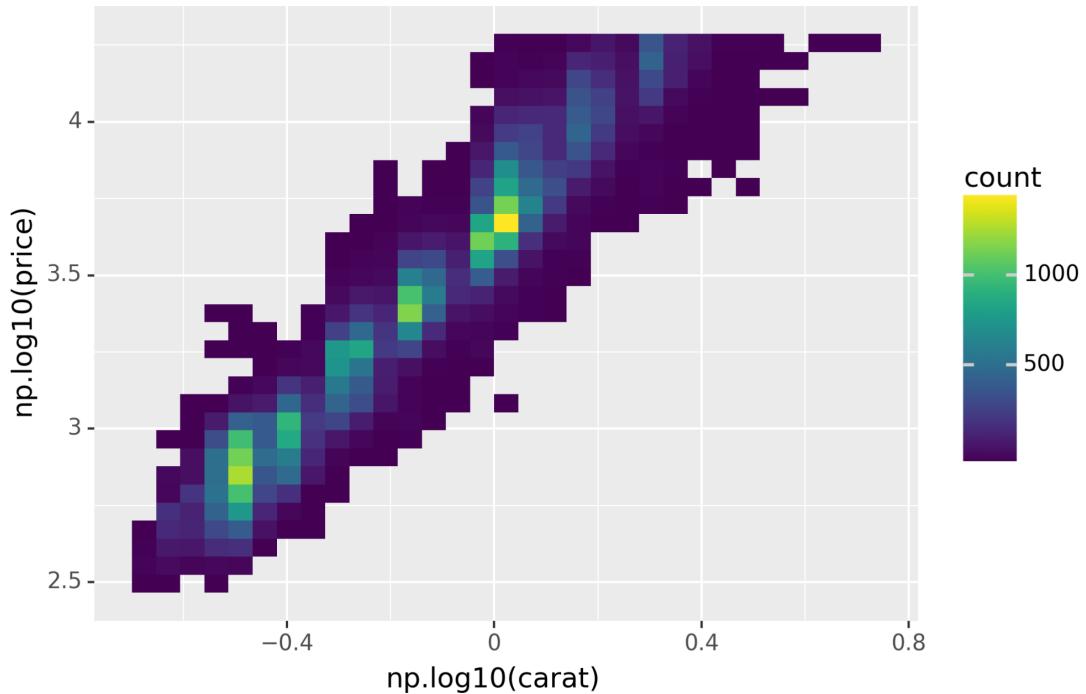


```
ggplot(diamonds, aes("carat",  
"price")) +\  
geom_bin2d()
```

```
ggplot(diamonds,  
aes("np.log10(carat)",  
"np.log10(price)")) +\  
geom_bin2d()
```

Не очень наглядно, поскольку хвост равномерно размазан по графику и зависимость не очень очевидна.

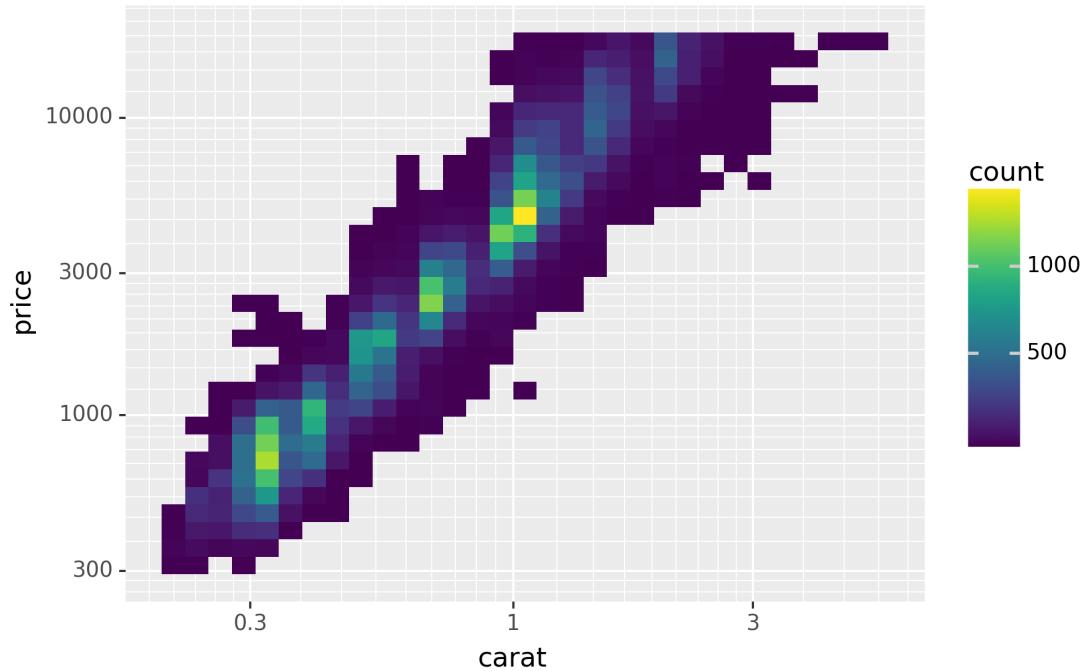
Логарифмическая шкала



Можно сделать это вот таким способом, визуализируя вместо цены и каратов их логарифмы. Но читаются такие шкалы не очень хорошо.

```
ggplot(diamonds,  
aes("np.log10(carat)",  
"np.log10(price)")) +\  
geom_bin2d()
```

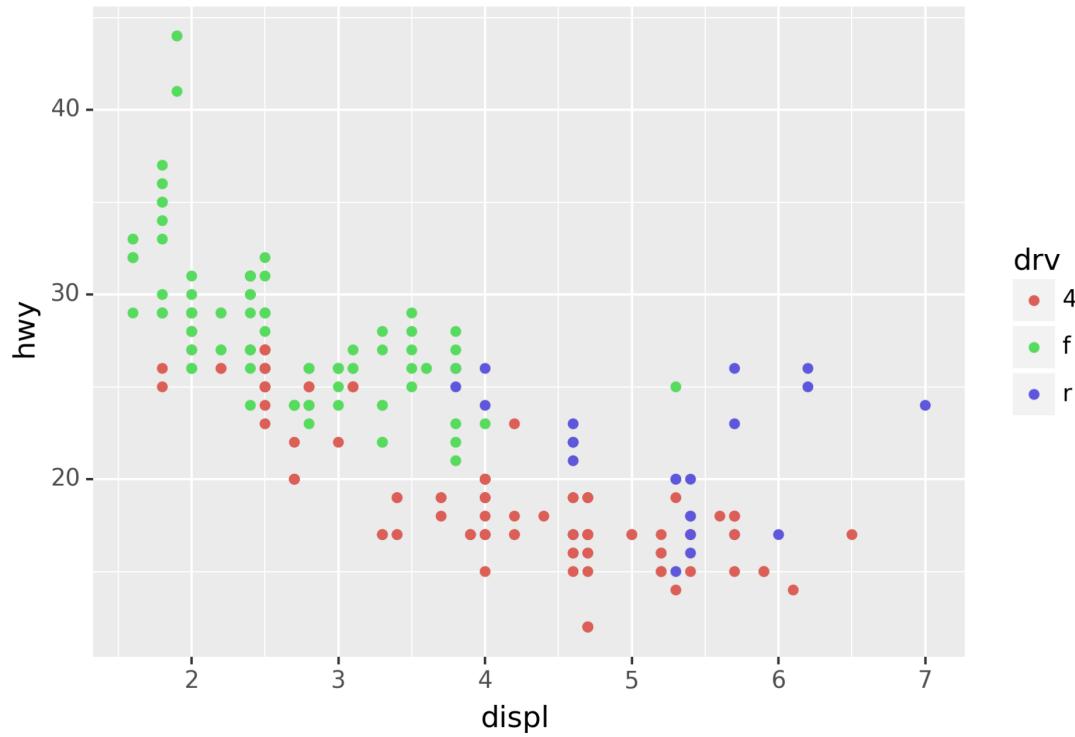
Логарифмическая шкала



А можно просто вежливо попросить сделать логарифмическую шкалу

```
ggplot(diamonds, aes("carat",  
"price")) +\  
geom_bin2d() +\  
scale_x_log10() +\  
scale_y_log10()
```

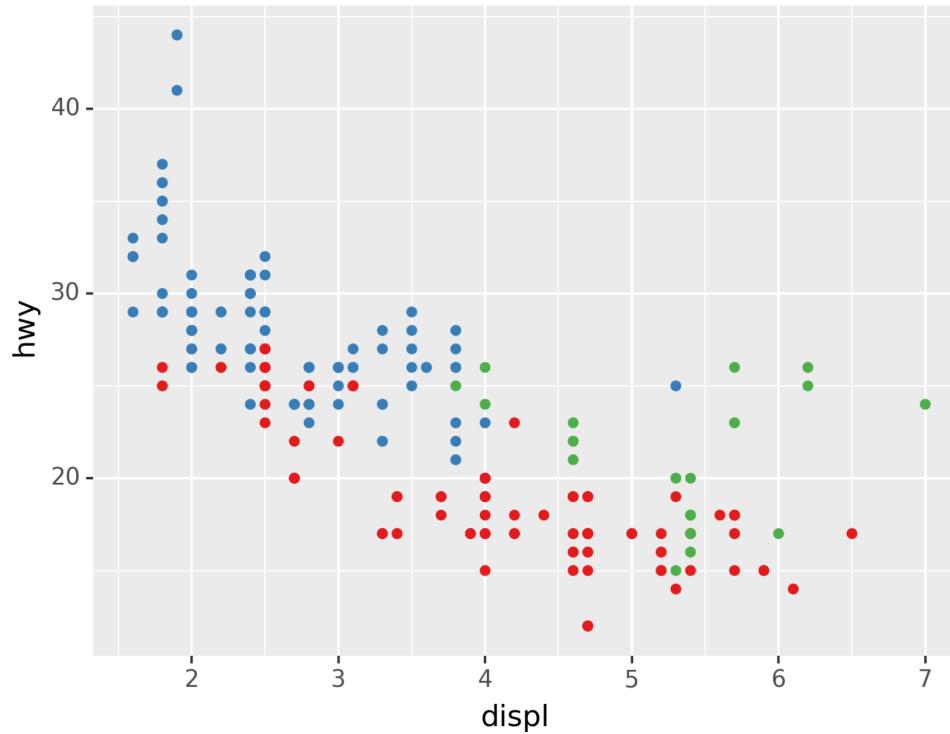
Цветовые шкалы



Так выглядит цветовая шкала по умолчанию.

```
ggplot(mpg, aes("displ", "hwy")) +\n  geom_point(aes(color="drv"))
```

Цветовые шкалы

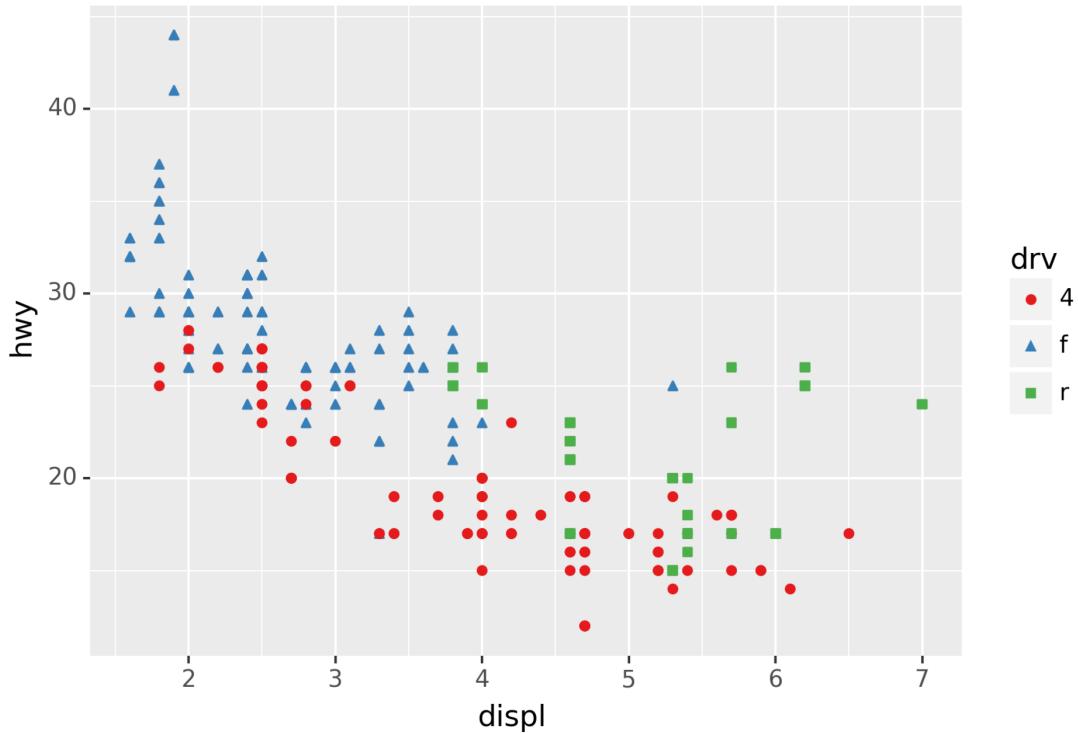


А так с палитрой «Set1»

drv
● 4
● f
● r

```
ggplot(mpg, aes("displ", "hwy")) +\\  
  geom_point(aes(color="drv")) +\\  
  scale_colour_brewer(type="qual",  
  palette="Set1")
```

Избыточность



Можно ещё добавить избыточность по форме, чтобы наверняка отличить категории (помните про «довнимание»?)

drv
● 4
▲ f
■ r

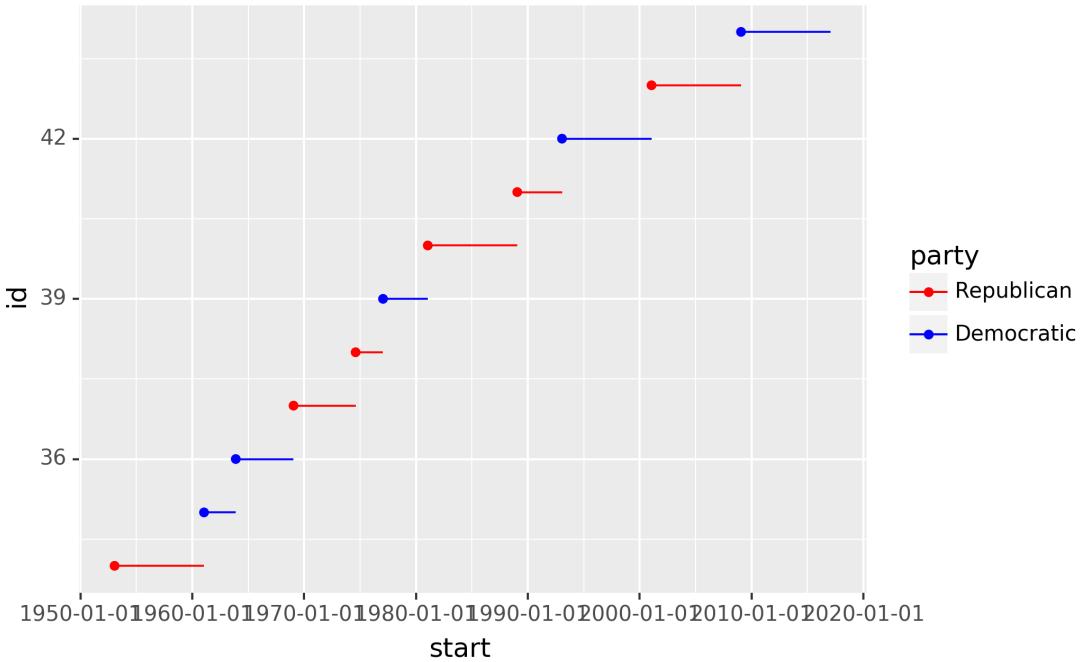
```
ggplot(mpg, aes("displ", "hwy")) +\n  geom_point(aes(color="drv",\n              shape="drv")) +\n  scale_colour_brewer(type="qual",\n                      palette="Set1")
```

Цветовые шкалы



И не забывайте про `rd.cut()`

Вручную заданные цветовые схемы



```
presidential["id"] = 34 +  
presidential.index
```

```
ggplot(presidential, aes("start",  
"id", colour="party")) +\  
geom_point() +\  
geom_segment(aes(xend="end",  
yend="id")) +\  
scale_colour_manual(values=[ "red",  
"blue"], limits=[ "Republican",  
"Democratic" ])
```

Иногда удобно назначить некоторым категориям конкретные цвета.

Масштабирование

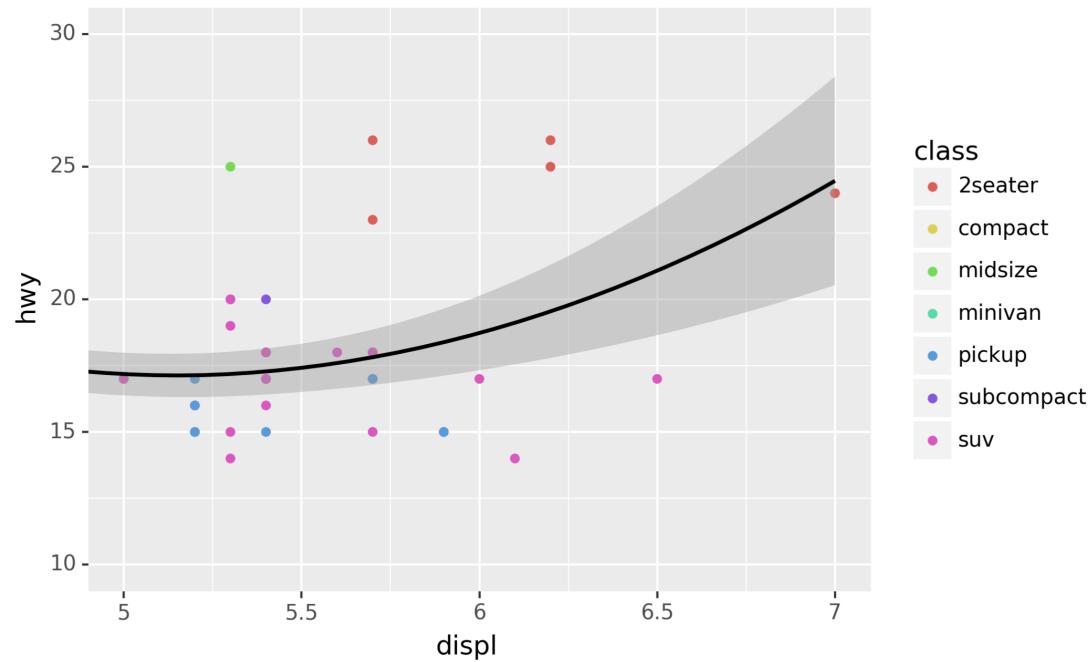
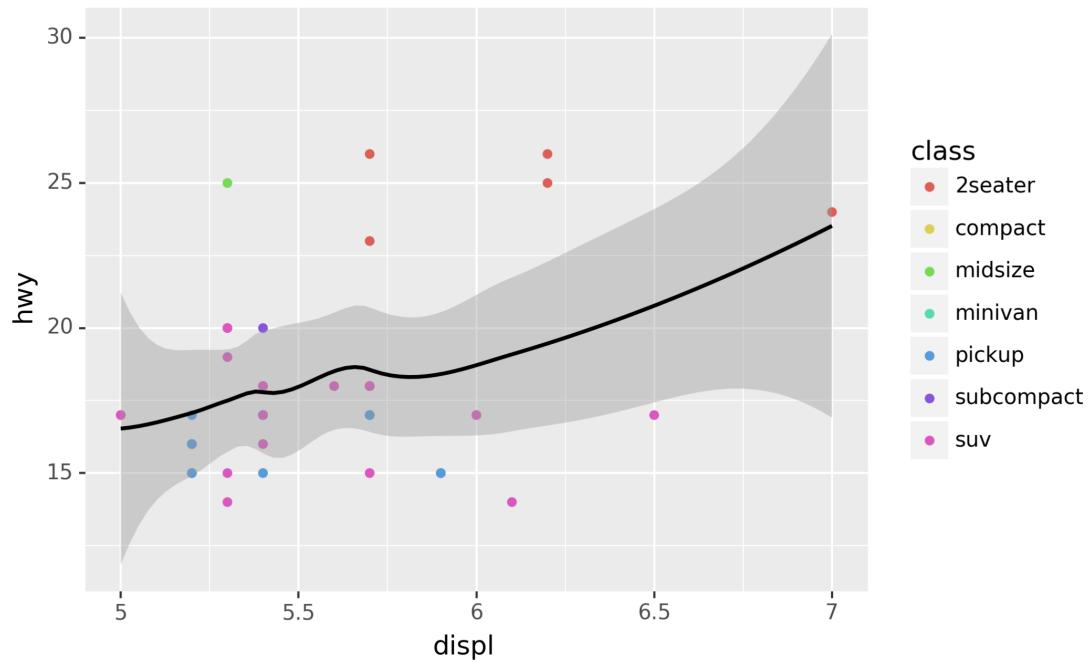


График из-за автоматически выбранных границ осей получается недостаточно детальным. Что если их ограничить?

```
ggplot(mpg, aes("displ", "hwy")) +\\
  geom_point(aes(color="class")) +\\
  geom_smooth() +\\
  coord_cartesian(xlim=(5, 7),\\
  ylim=(10, 30))
```

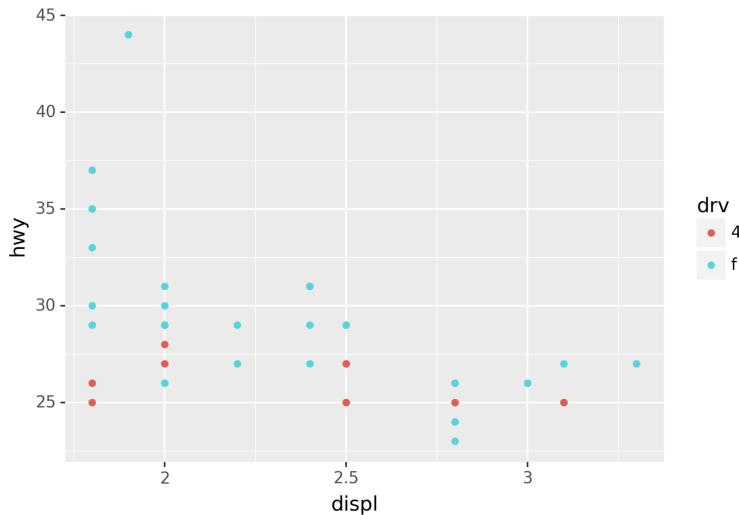
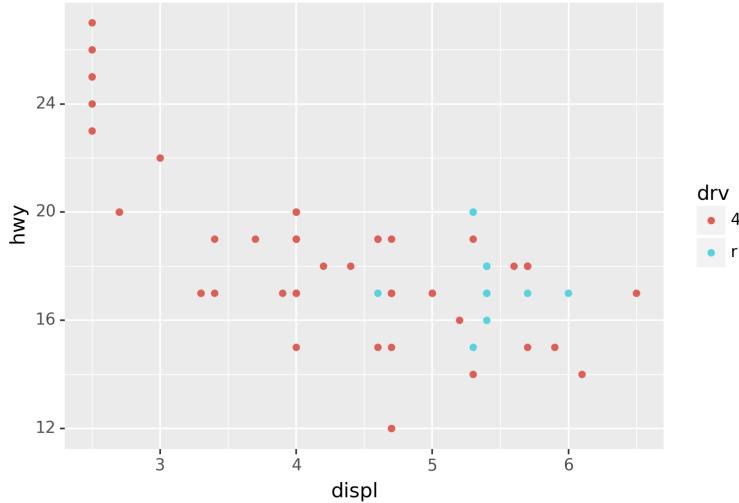
Масштабирование



Может такая детализация в каком-то конкретном случае окажется полезнее

```
ggplot(mpg.query("5 <= displ <= 7 and  
10 <= hwy <= 30"), aes("displ",  
"hwy")) +\\  
geom_point(aes(color="class")) +\\  
geom_smooth()
```

Пределы осей (авто)



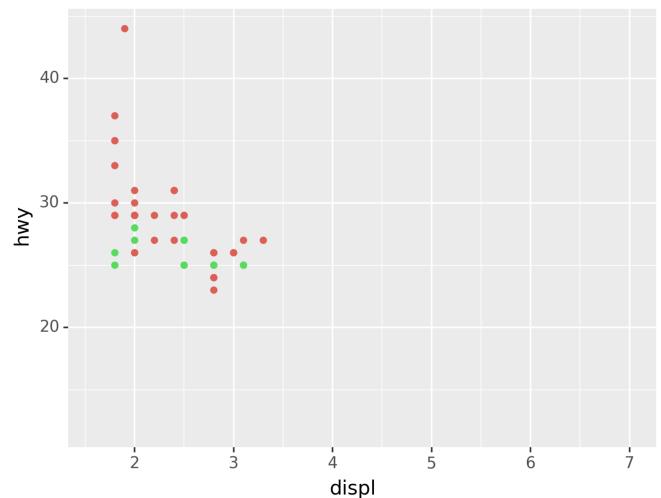
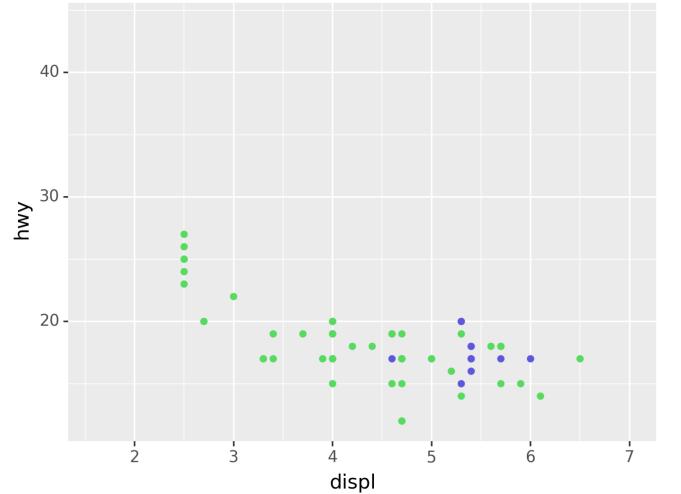
```
mpg["drv"] = mpg["drv"].astype(str)
suv = mpg[mpg["class"] == "suv"]
compact = mpg[mpg["class"] == "compact"]
```

```
ggplot(suv, aes("displ", "hwy",
colour="drv")) +\
geom_point()
```

```
ggplot(compact, aes("displ", "hwy",
colour="drv")) +\
geom_point()
```

Из-за разных шкал по одним и тем же осям
сравнение двух графиков затрудняется

Пределы осей



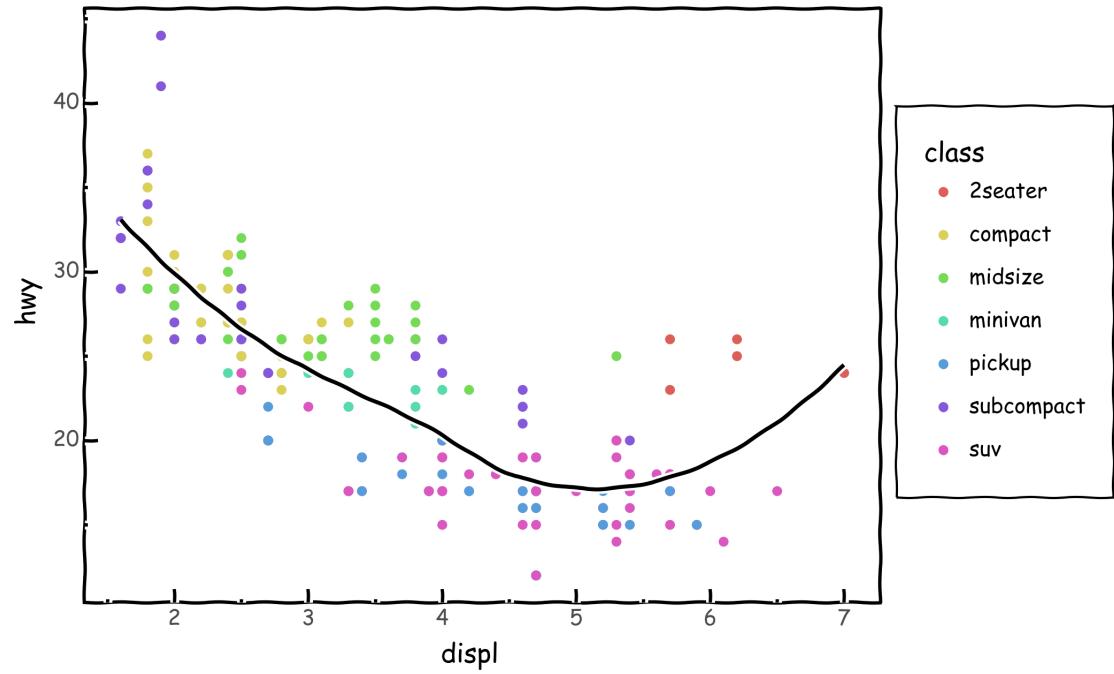
```
x_scale = scale_x_continuous(limits=mpg.displ.min(), mpg.displ.max())
y_scale = scale_y_continuous(limits=mpg.hwy.min(), mpg.hwy.max())
col_scale = scale_colour_discrete(limits=mpg.drv.unique())
```

```
ggplot(suv, aes("displ", "hwy",
colour="drv")) + geom_point() + x_scale +
y_scale + col_scale
```

```
ggplot(compact, aes("displ", "hwy",
colour="drv")) + geom_point() + x_scale +
y_scale + col_scale
```

А такой результат получится, если мы заранее зададим шкалы

Темы



class

● 2seater

● compact

● midsize

● minivan

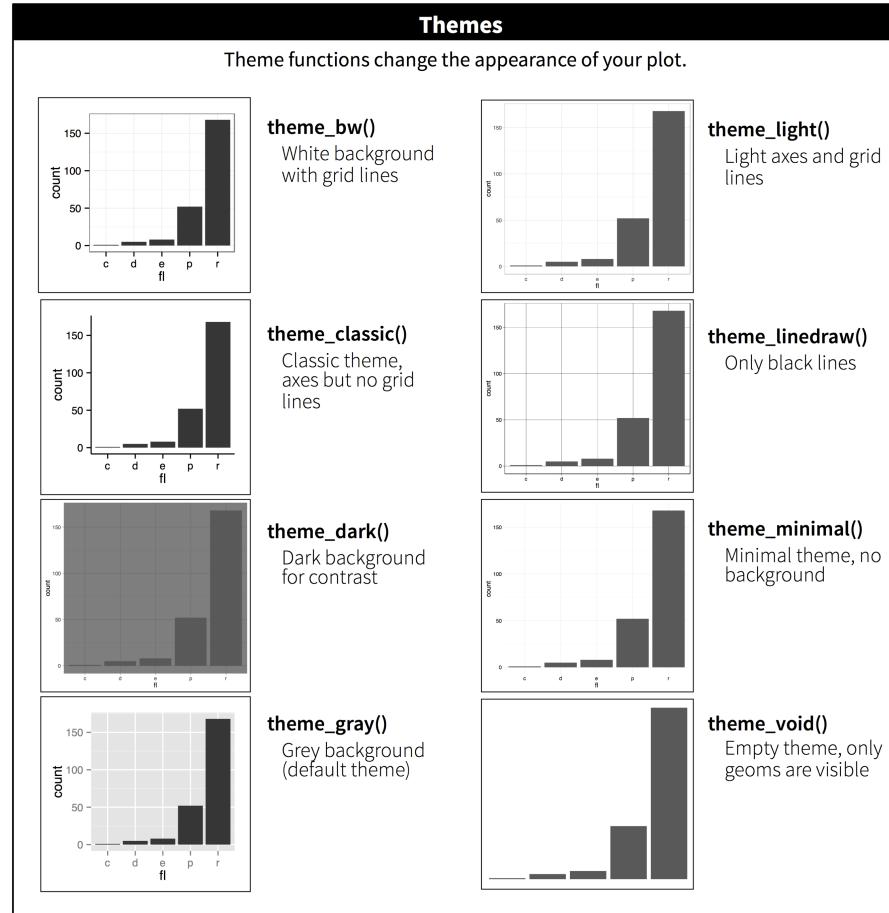
● pickup

● subcompact

● suv

```
ggplot(mpg, aes("displ", "hwy")) +\n  geom_point(aes(color="class")) +\n  geom_smooth(se=False) +\n  theme_xkcd()
```

Темы



Сохранение графиков

```
ggplot(mpg, aes("displ", "hwy")) +  
  geom_point()  
  
.save("my-plot.pdf")
```

сохраняет последний с параметрами по умолчанию

Если не заданы ширина и высота, по умолчанию, их значения будут равны 6,4 и 4,8 дюйма. Если не задано название графика, то `plotnine` сгенерирует своё вида “`plotnine-save-297120101.pdf`”.

Результат сохранения для разных размеров

