Грамматика графиков

Как R, только Python

Историческая справка

- Лэланд Вилкинсон «Грамматика графиков»
- Хэдли Викхэм ggplot для R
- Хэдли Викхэм ggplot2 для R и «ggplot2: Elegant Graphics for Data Analysis»

Что такое грамматика графиков?

- Все графики состоят из:
- Массива данных (**data**), который необходимо визуализировать и набора отображений, которые описывают соответствие переменных из массива визуальным средствам (**aes**thetic).
- Слоёв (**layer**), собранных из геометрических объектов и статистических преобразований. Геометрические объекты (**geom**) представляют собой визуальные элементы графиков: точки, линии, полигоны. Статистические преобразования (**stat**) обобщают данные различными способами (биннинг, подсчёт количества).

Что такое грамматика графиков?

- **scale** преобразует данные из пространства данных в параметры визуального пространства (цвет, размер или форма), что позволяет, а также определяет оси и легенды, что позволяет считать исходные данные из графика.
- Координатная система (**coord**) описывает то, как данные преобразуются на плоскость графика. Также оно обеспечивает определение значений осей и разметку для чтения графиков. Обычно используется декартова система координат, но существует и ряд других.
- Панели (**facet**) определяют способ разбиения данных на поднаборы и как эти поднаборы отображаются в общем графике. Также известно как latticing/trellising.
- Художественное оформление (**theme**) определяет шрифты и цвет фона.

История библиотек

- Грамматика графиков впервые была введена в языке программирования R посредством ggplot и ggplot2. Учитывая их успех в прошлом, эти библитеки были адаптированы в Python в библиотеке plotnine.

- Грамматика графиков не отвечает на вопрос, какие графики лучше использовать для решения той или иной задачи.

- Грамматика графиков не описывает интерактивность графиков.

Установка и пререквизиты

```
pip install plotnine[all]
```

```
from plotnine import *
from plotnine.data import *
```

```
import numpy as np
import pandas as pd
```

Полезный совет на windows:

Для ряда статистических функций вам понадобится модуль scikit-misc. Он не ставится на windows через стандартный репозиторий pip.

Придётся ставить вот отсюда:

https://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy

https://www.lfd.uci.edu/~gohlke/pythonlibs/#scikit-misc

Ключевые компоненты

Каждый plotnine (ggplot2) график состоит из трёх компонентов:

- 1. Данные.
- 2. Набор визуальных отображений.
- 3. Как минимум, один слой, который описывает правило отрисовки (обычно создаётся функцией geom).

Датафрейм mpg

mpg

- manufacturer model displ year cyl trans drv cty hwy fl class
- 0 audi a4 1.8 1999 4 auto(l5) f 18 29 p compact
- 1 audi a4 1.8 1999 4 manual(m5) f 21 29 p compact
- 2 audi a4 2.0 2008 4 manual(m6) f 20 31 p compact
- .. … … … … … … .. … … .. …
- 232 volkswagen passat 2.8 1999 6 manual(m5) f 18 26 p midsize
- 233 volkswagen passat 3.6 2008 6 auto(s6) f 17 26 p midsize

- [234 rows x 11 columns]

Категории датафрейма mpg

manufacturer – производитель,

model – модель,

displ – рабочий объём двигателя в литрах,

year – год выпуска,

cyl – число цилиндров,

trans – тип коробки передач,

drv – тип привода,

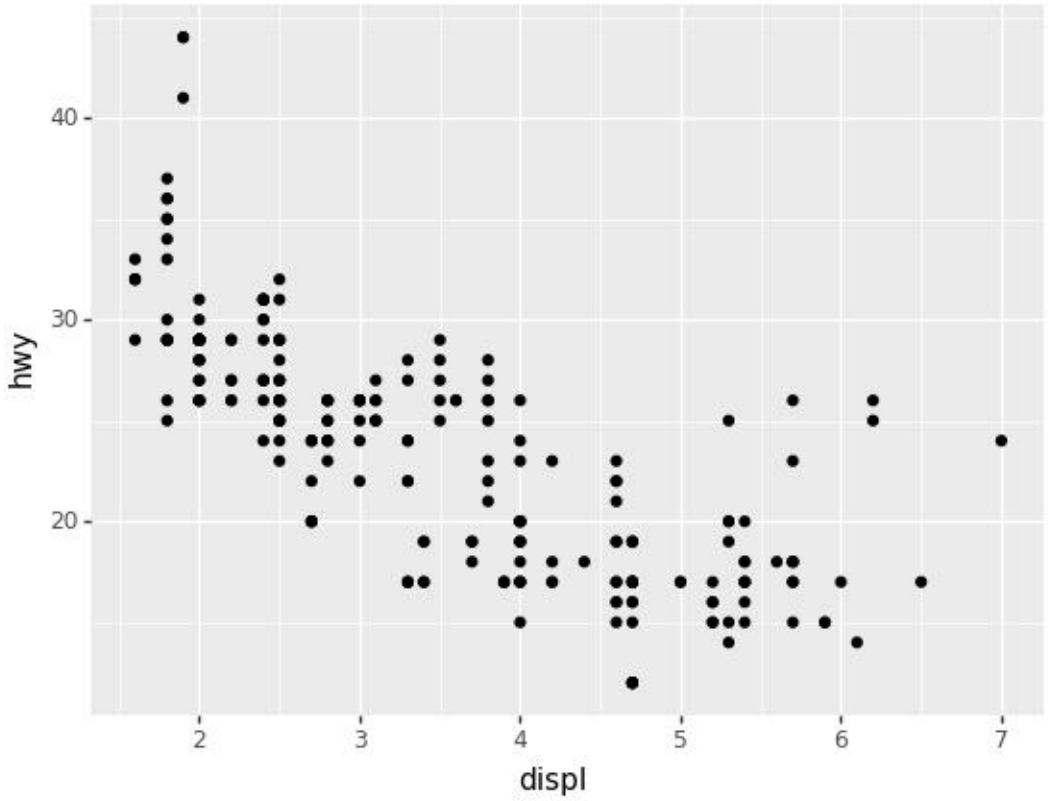
cty – расход топлива в городе (миль/галлон),

hwy – расход топлива на трассе (миль/галлон)

fl – тип топлива

class – класс автомобиля

Ключевые компоненты ggplot

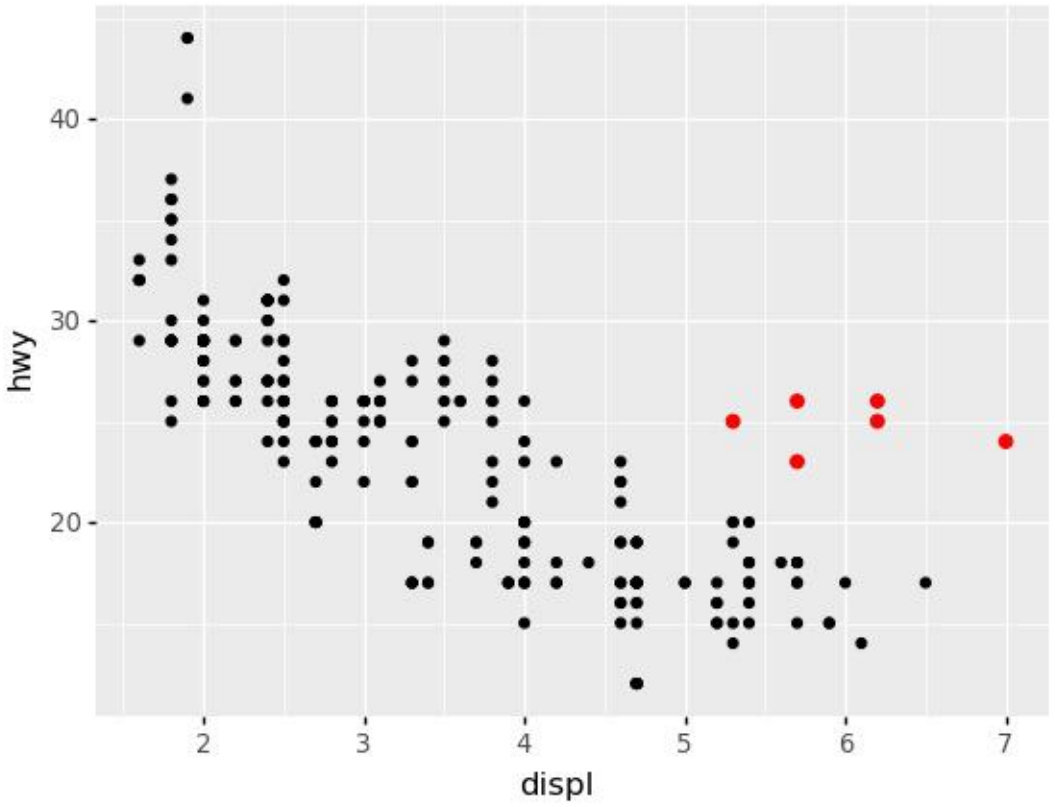


- Простейший пример:

```
ggplot(data=mpg) +\
geom_point(mapping=aes(x='displ', y='hwy'))
```

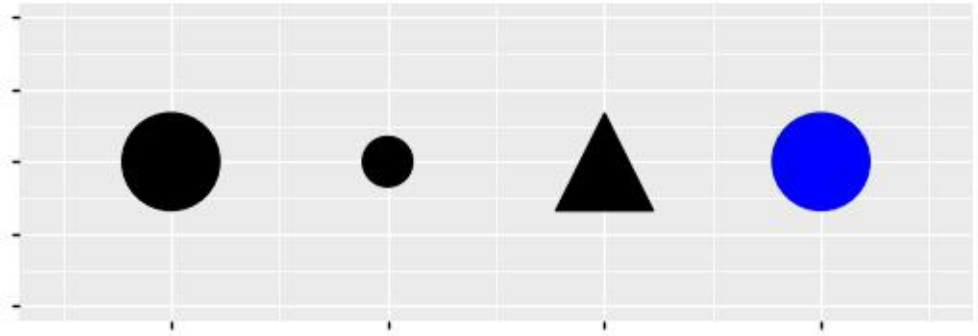
- 1. **Данные**: mpg.
- 2. **Визуальное отображение**: Объём двигателя по x, расход топлива по y.
- 3. **Слой**: points.

Анализ особенностей графика



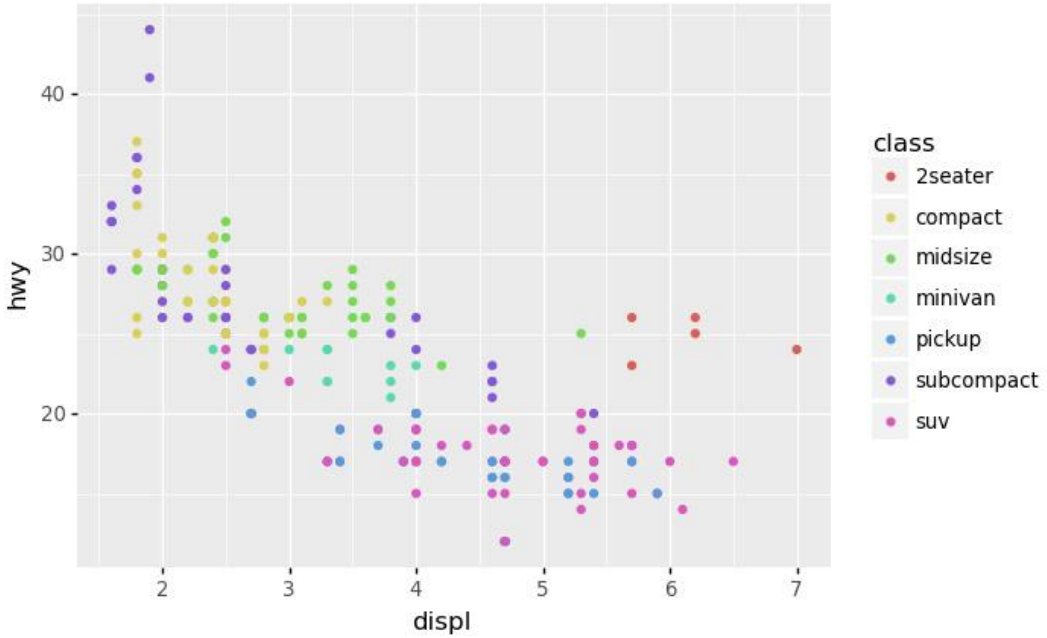
- В приведённом графике одна группа точек (выделена красным на графике) выбивается из общего тренда. Можно попытаться обосновать это другими характеристиками автомобиля, например, классом.

Визуальное отображение



- К имеющемуся графику можно добавить третью переменную (например «класс автомобиля») за счёт использования визуального отображения (**aesthetic**). Визуальным отображением называются визуальные свойства объектов графика (например, размер, форма или цвет).

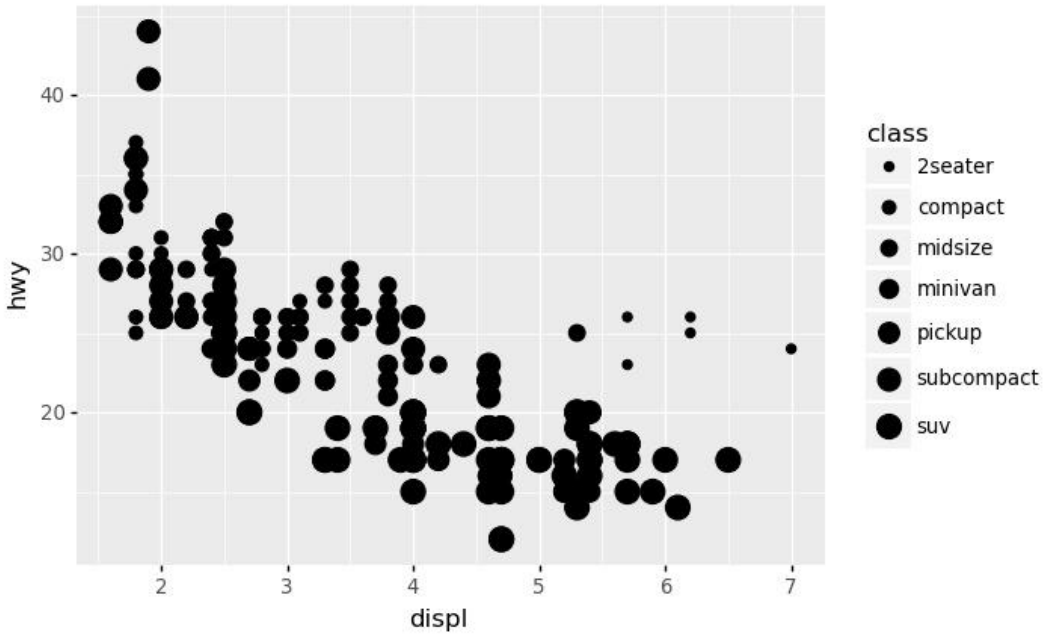
Цвет



- Добавим цвет, связанный с категорией «класс автомобиля».

- ggplot(data=mpg) +\
- geom_point(mapping=aes(x="displ", y="hwy", color="class"))

Размер

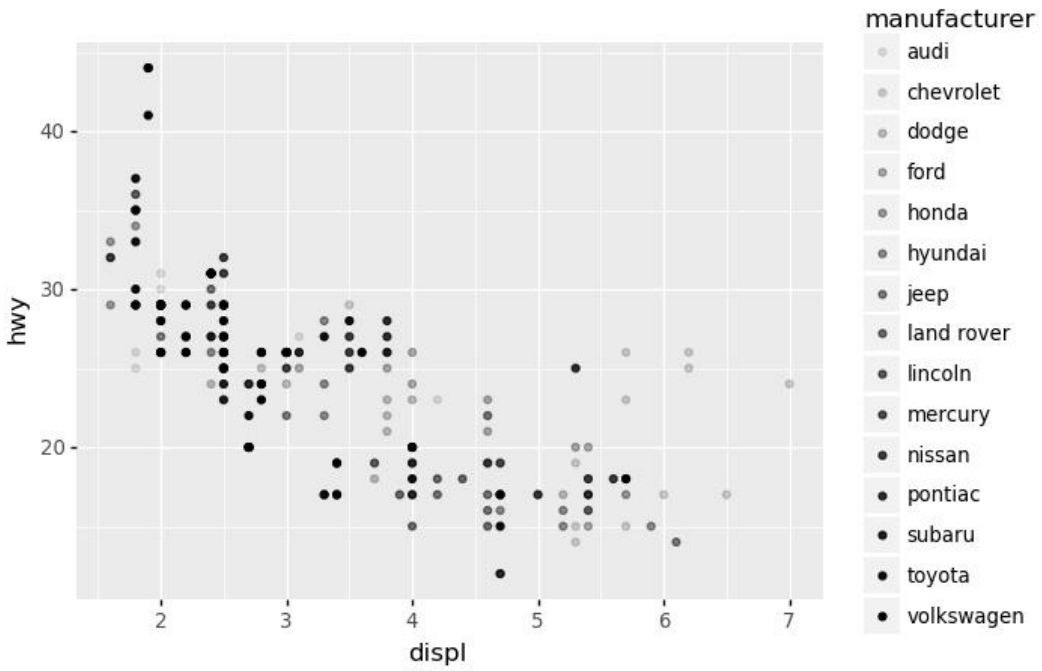


- Теперь представим ту же категорию как размер
- ggplot(data=mpg) +\
- geom_point(mapping=aes(x="displ", y="hwy", size="class"))
- (откуда возникает warning?)

Оттенок и форма (что не так с «Тойотой»?)

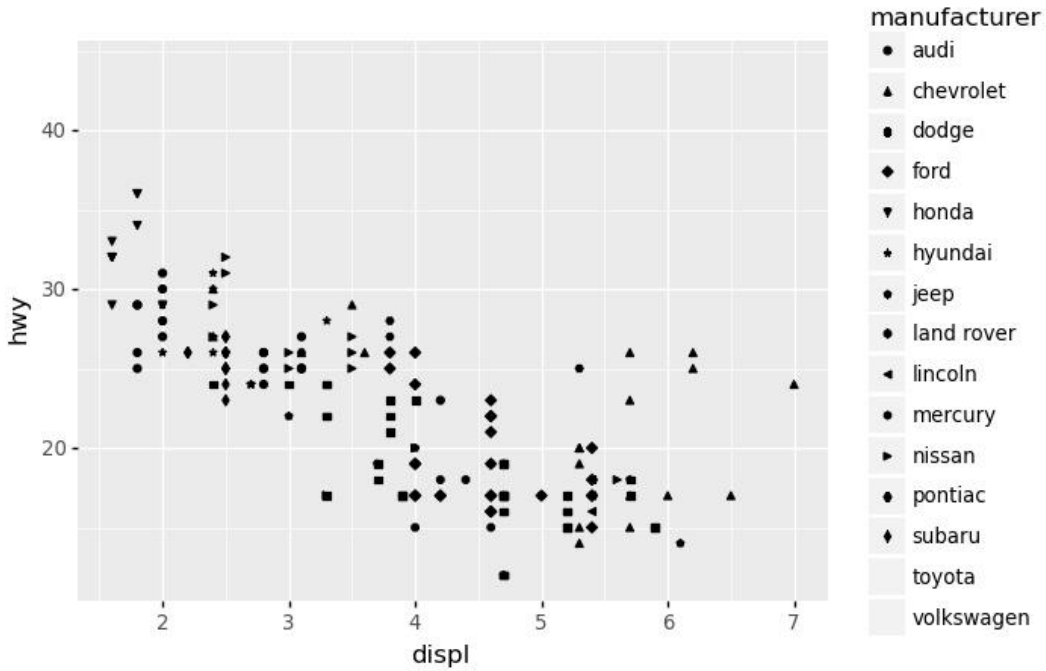
ggplot(data=mpg) +

geom_point(mapping=aes(x="displ", y="hwy", alpha="manufacturer"))

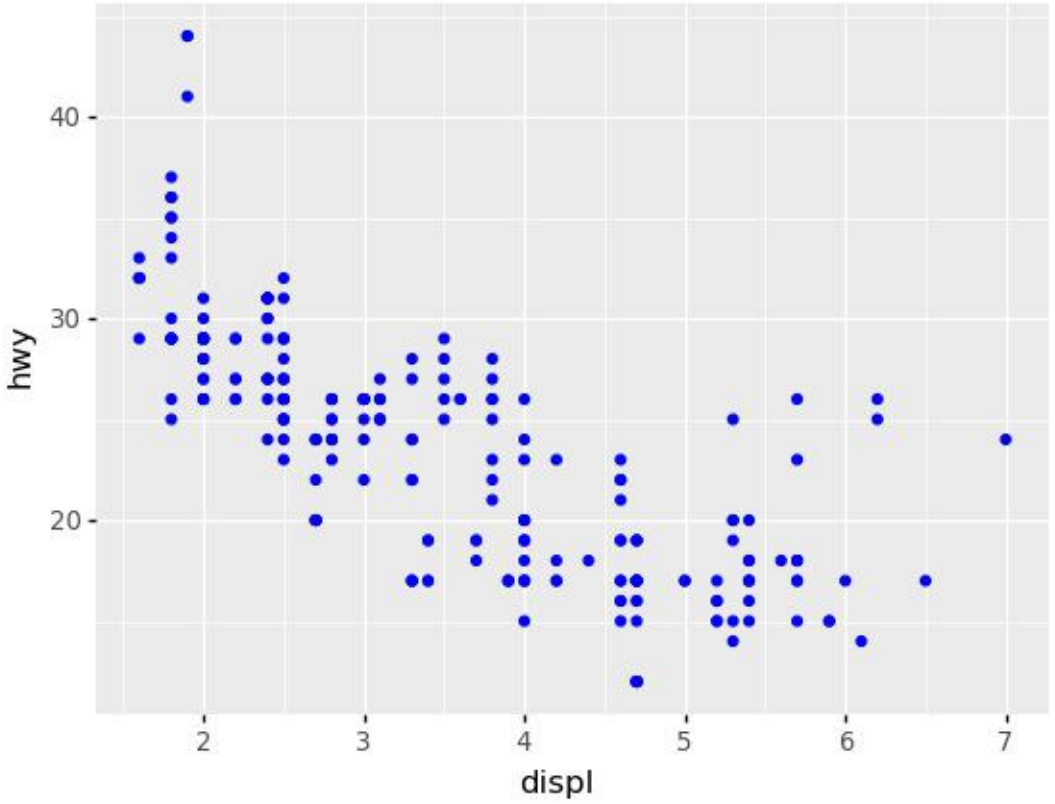


ggplot(data=mpg) +

geom_point(mapping=aes(x="displ", y="hwy", shape="manufacturer"))



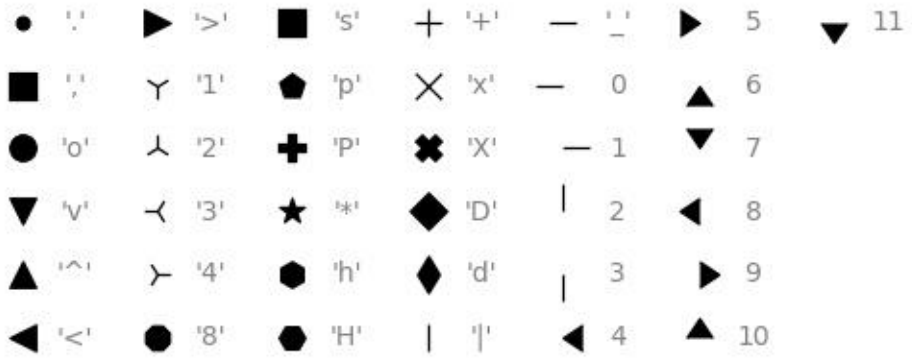
Фиксированный цвет




```
ggplot(data=mpg) +\
geom_point(mapping=aes(x="displ", y="hwy"), color="blue")
```

- Достаточно тривиально и очевидно, но стоит упомянуть такую возможность

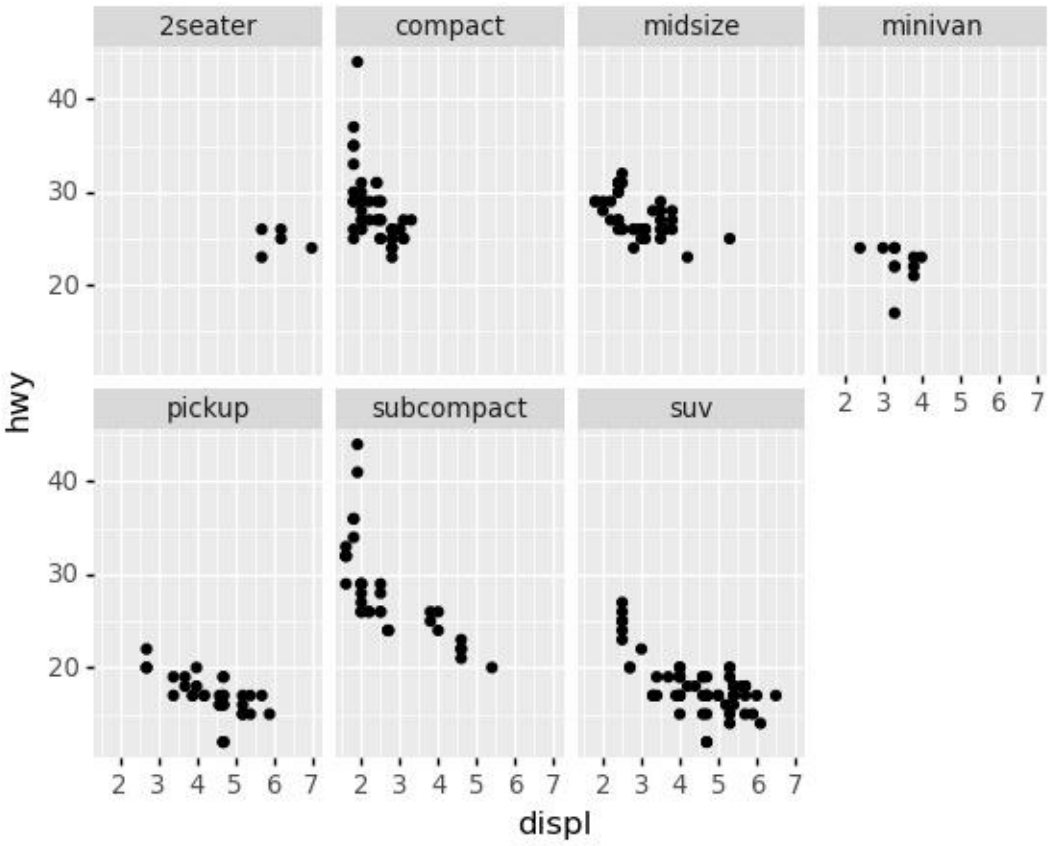
Произвольная форма



- Аналогично фиксированному цвету можно задать определённую форму точек с использованием

- shape='<символ формы>'

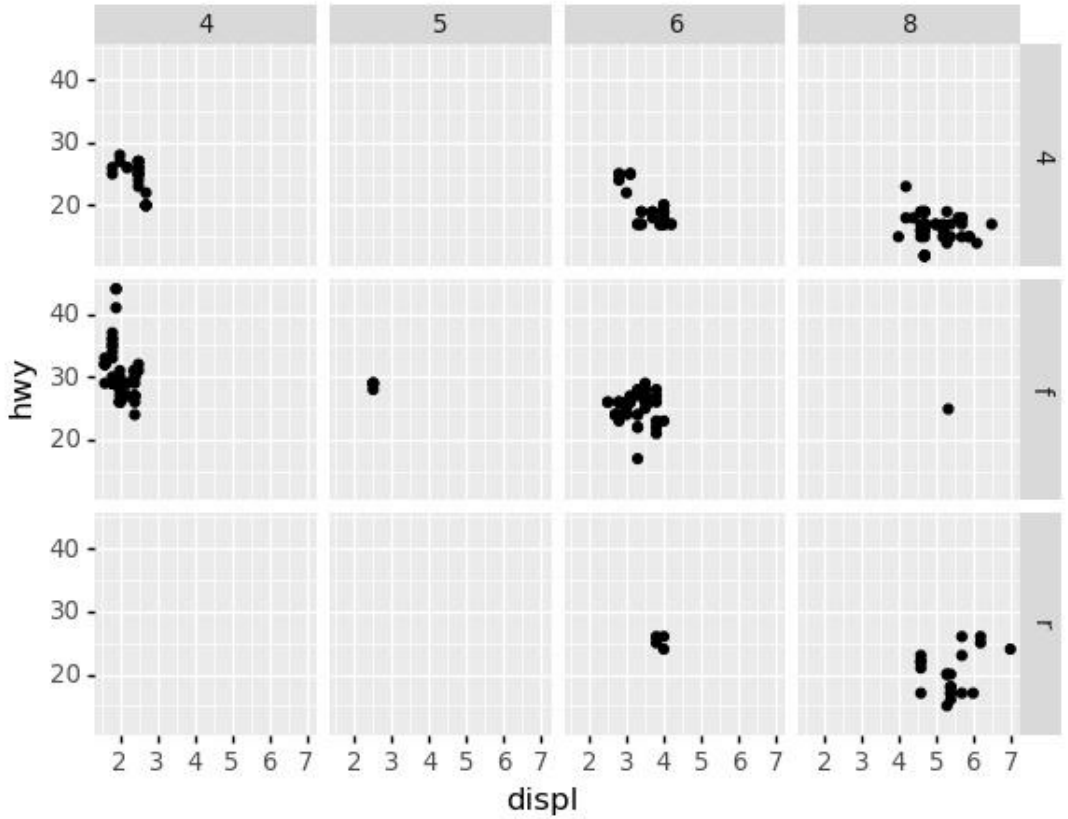
Панели



- Визуальное отображение – это один из способов добавить размерность на визуализацию.
- Другой способ: разделить изображение на панели с помощью

```
ggplot(data=mpg) +\
geom_point(mapping=aes(x="displ", y="hwy")) +\
facet_wrap("class", nrow=2)
```

Двумерная сетка панелей

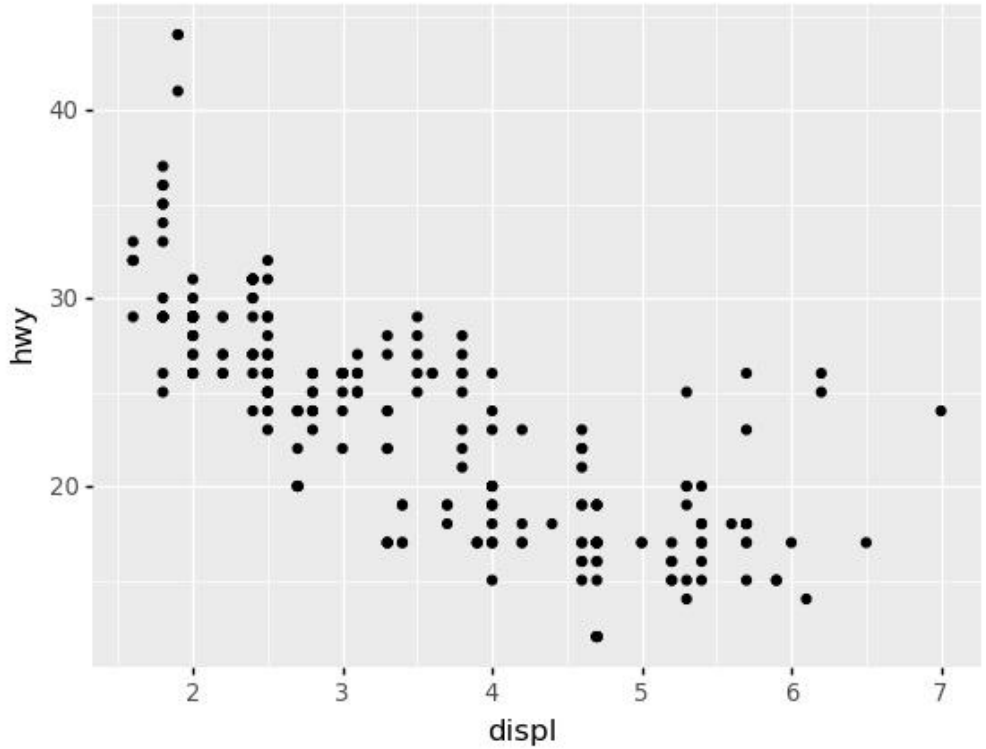


- ggplot(data=mpg) +\
- geom_point(mapping=aes(x="displ", y="hwy")) +\
- facet_grid("drv ~ cyl")

Геометрические объекты (geoms)

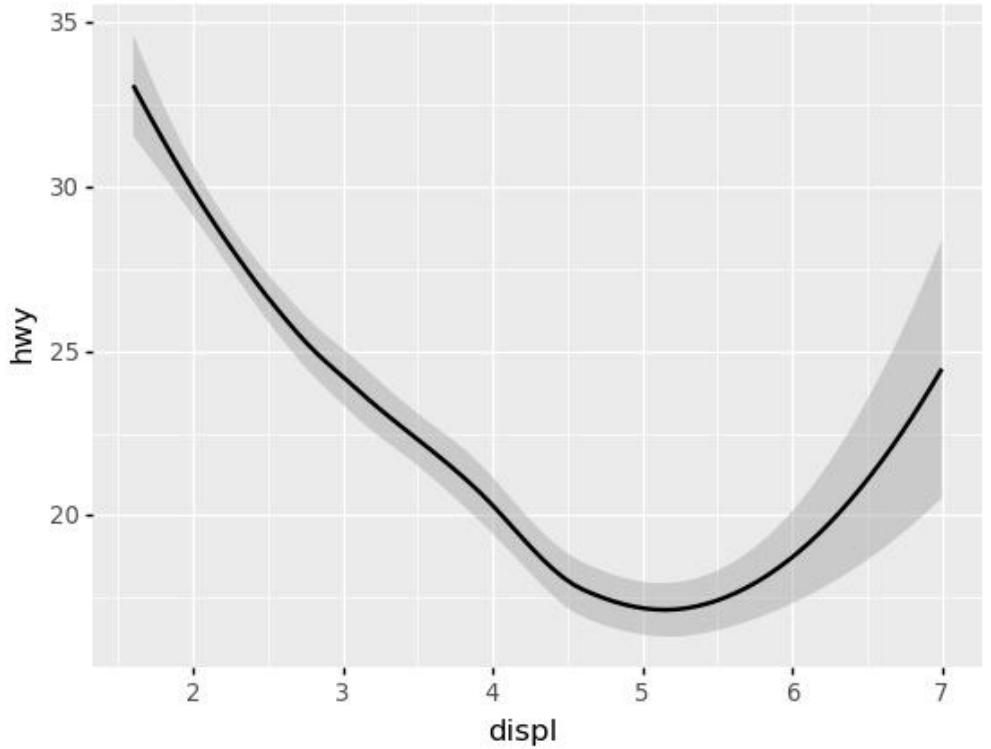
ggplot(data=mpg) +

geom_point(mapping=aes(x="displ", y="hwy"))

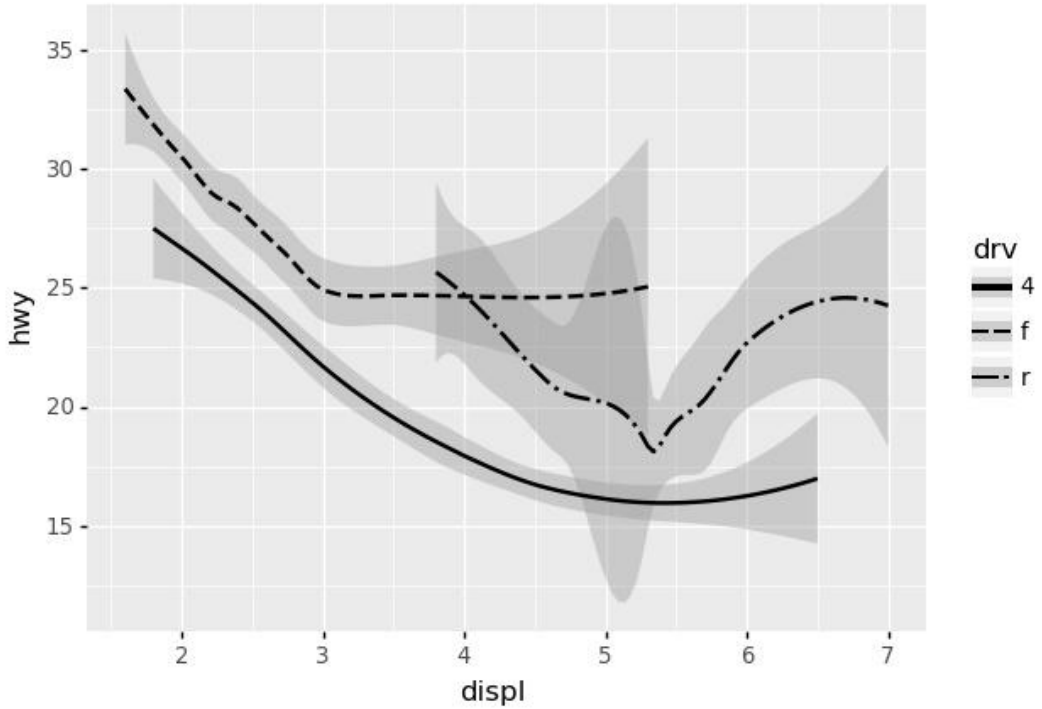


ggplot(data=mpg) +

geom_smooth(mapping=aes(x="displ", y="hwy"))



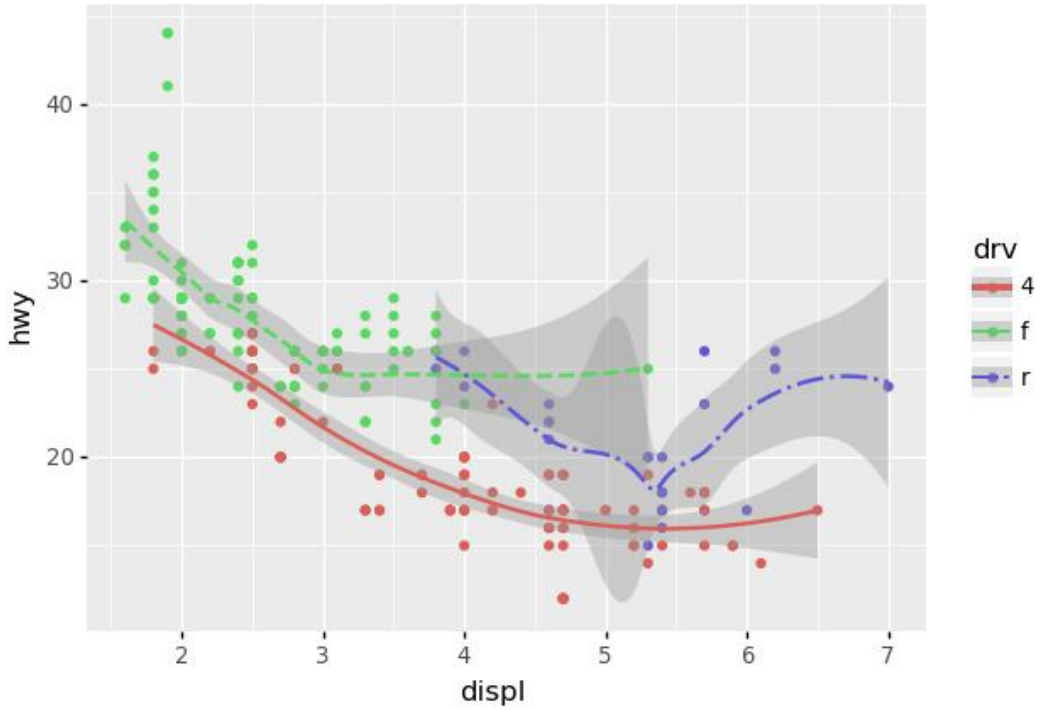
Добавление категорий линиям



- ggplot(data=mpg) +\
- geom_smooth(mapping=aes(x="displ", y="hwy", linetype="drv"))

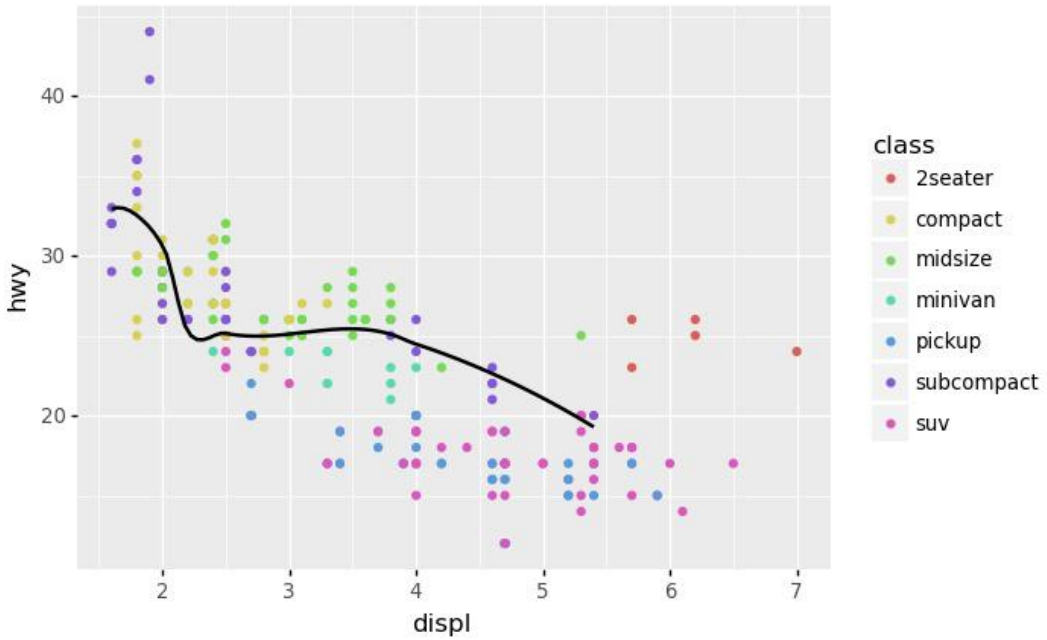
нарисует отдельную линию разной «формы» для каждого типа drv

Немного более ясности графикам



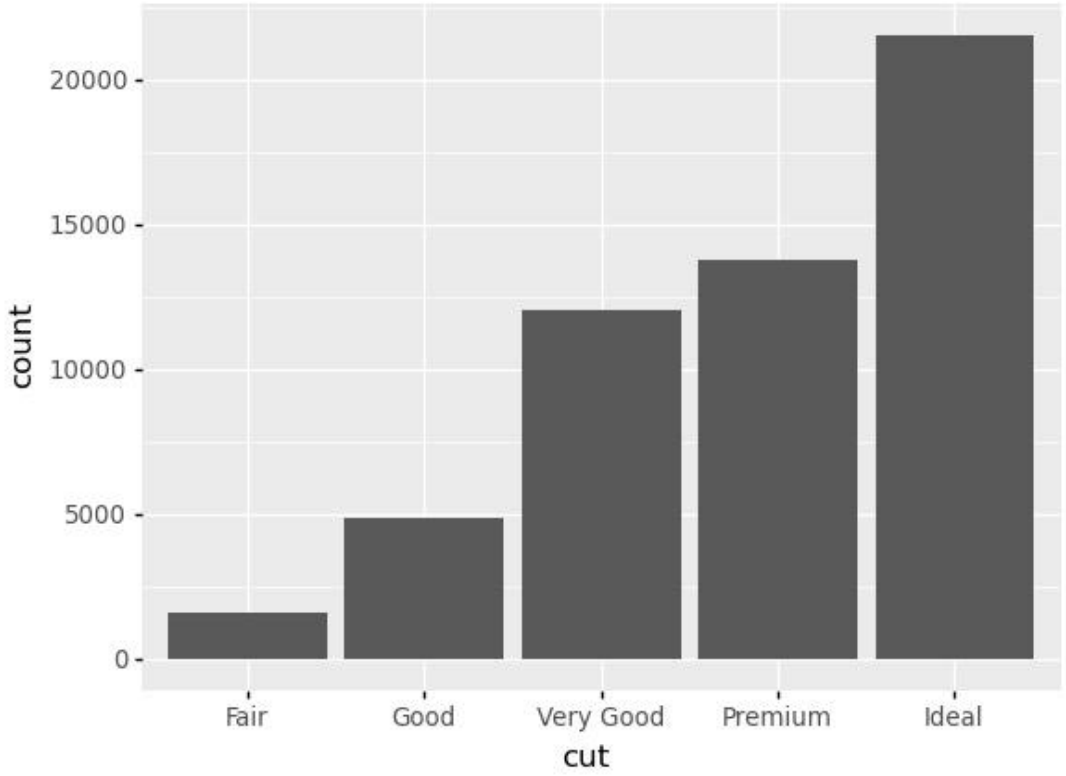
- ggplot(data=mpg, mapping=aes(x="displ", y="hwy", color="drv")) +\
- geom_point() +\
- geom_smooth(mapping=aes(line type="drv"))

Разные данные для разных слоёв



- ggplot(data=mpg, mapping=aes(x="displ", y="hwy")) +\
- geom_point(mapping=aes(color ="class")) +\
- geom_smooth(data=mpg.loc[mp g["class"] == "subcompact"], se=False)

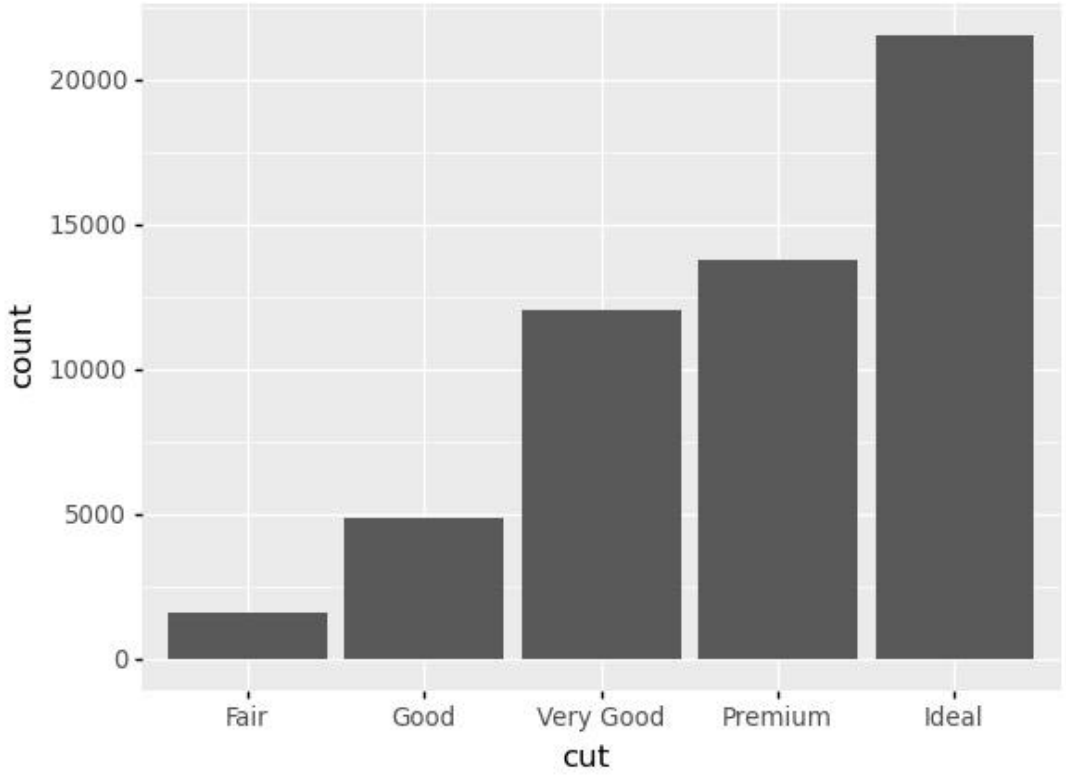
Статистические преобразования



- Покажем связь качества огранки алмазов и их числа.

- ggplot(data=diamonds) +\
- stat_count(mapping=aes(x="cut "))

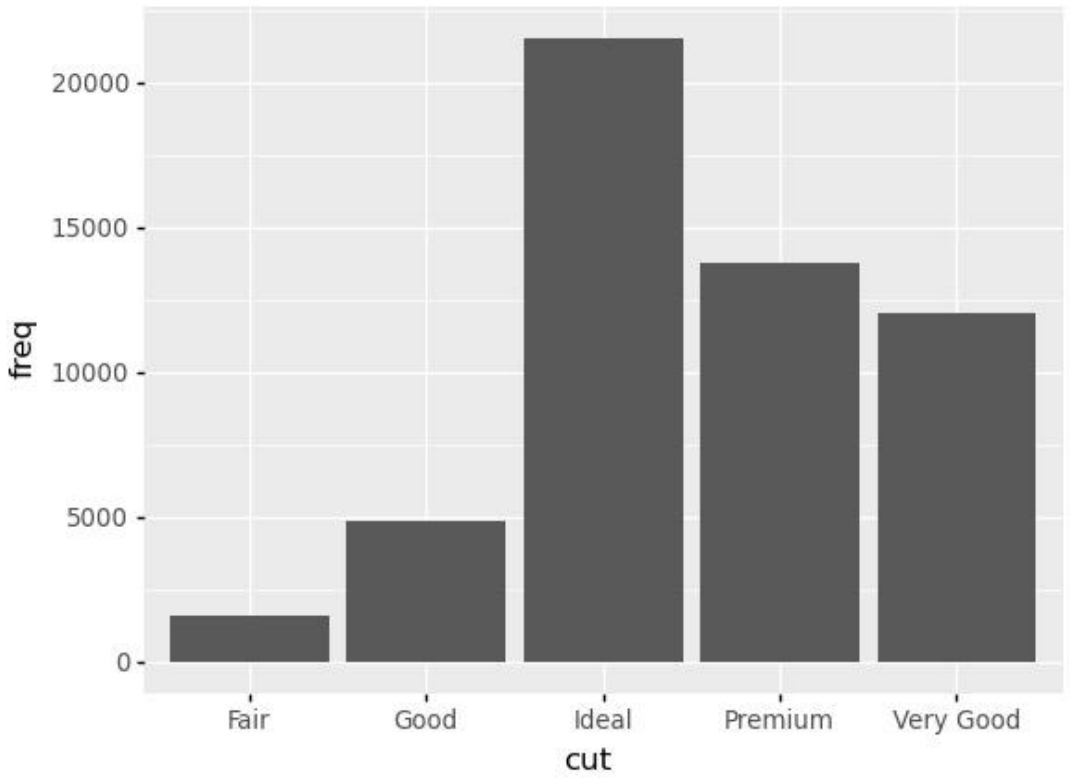
Связь stat и geom



- По умолчанию geom_bar использует stat_count.

- ggplot(data=diamonds) +\
- geom_bar(mapping=aes(x="cut "))

Первая причина использовать stat в явной форме

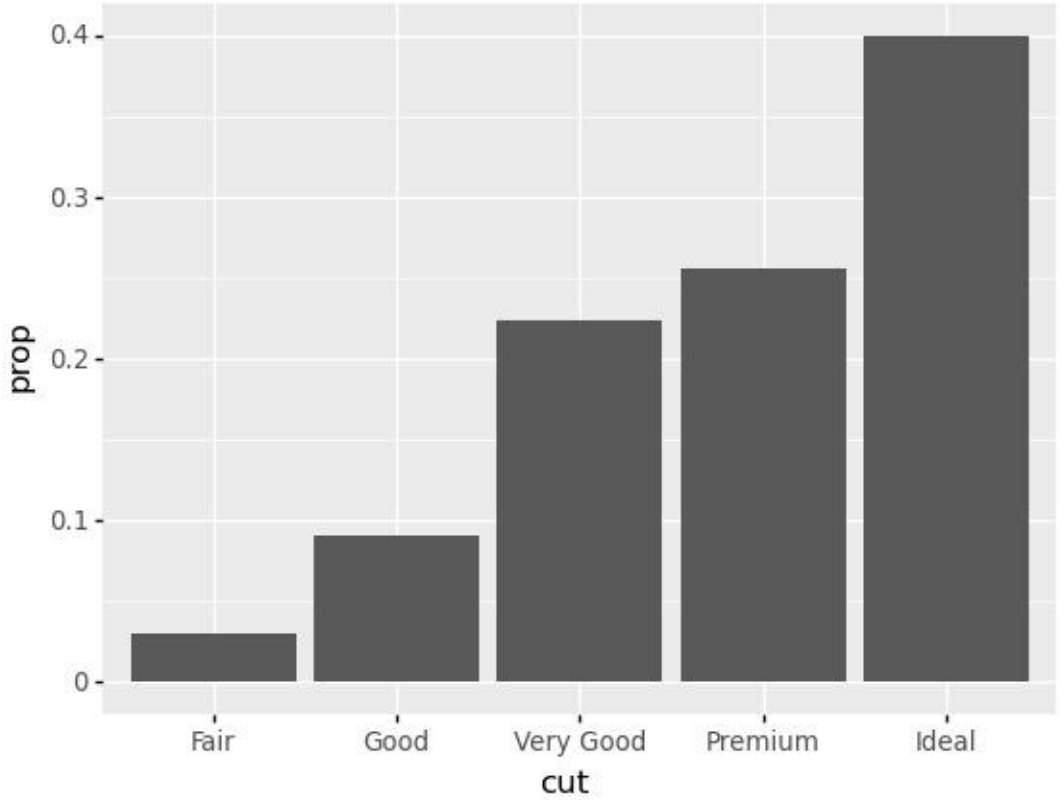


- Чтобы переписать stat по умолчанию:

- demo = pd.DataFrame({"cut": ["Fair", "Good", "Very Good", "Premium", "Ideal"],
- "freq": [1610, 4906, 12082, 13791, 21551]})

- ggplot(data=demo) +\
- geom_bar(mapping=aes(x="cut", y="freq"), stat="identity")

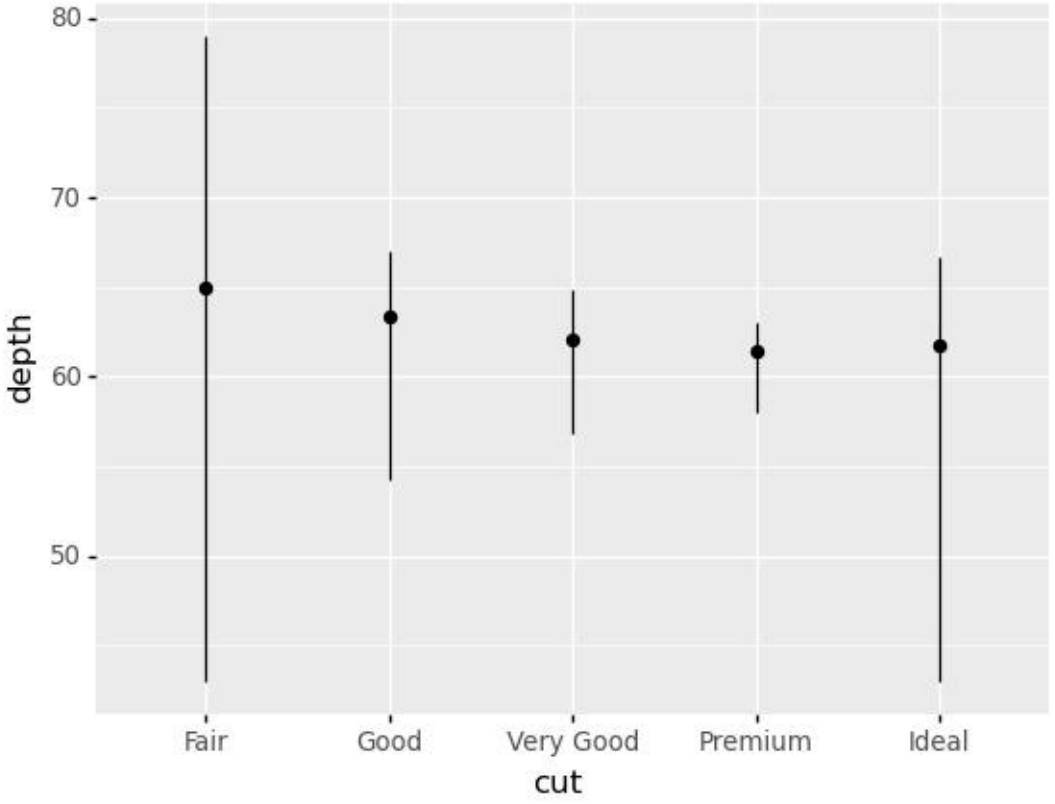
Вторая причина использовать stat в явной форме



- Переписать преобразование из переменных в визуальное отображение, например, показать долю, а не общее число:

- ggplot(data=diamonds) +\
- geom_bar(mapping=aes(x="cut", y="..prop..", group=1))

Третья причина использовать stat в явной форме

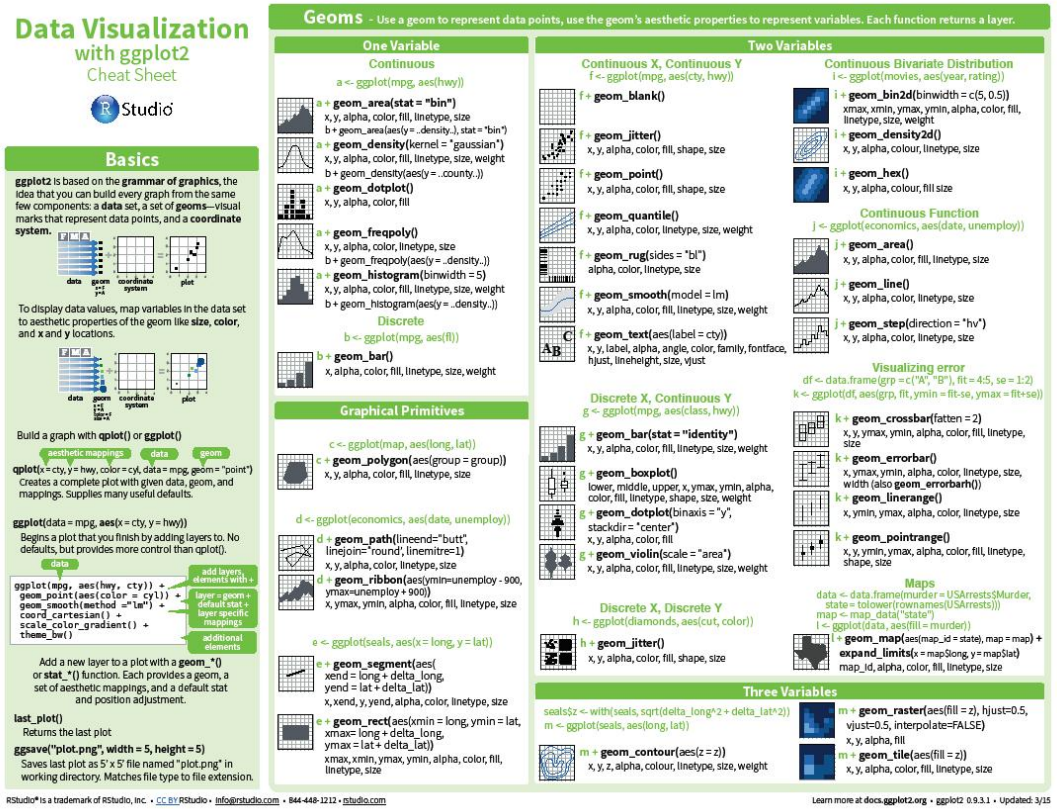


- Уделить более внимание статистическим преобразованиям. Например, использовать stat_summary():
- ggplot(data=diamonds) +\
- stat_summary(
- mapping=aes(x="cut", y="depth"),
- fun_ymin=np.min,
- fun_ymax=np.max,
- fun_y=np.median
-)

Сколько всего?

- plotnine предоставляет выбор из более чем 30 геометрических объектов и более 20 статистических преобразований

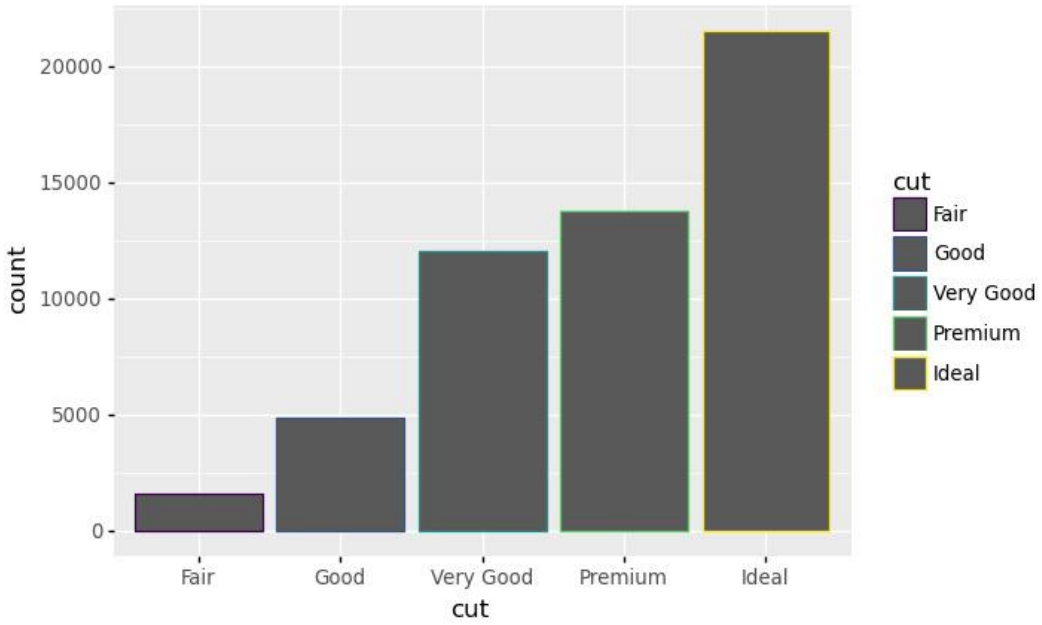
ggplot2 cheat sheet



Уточнение положения

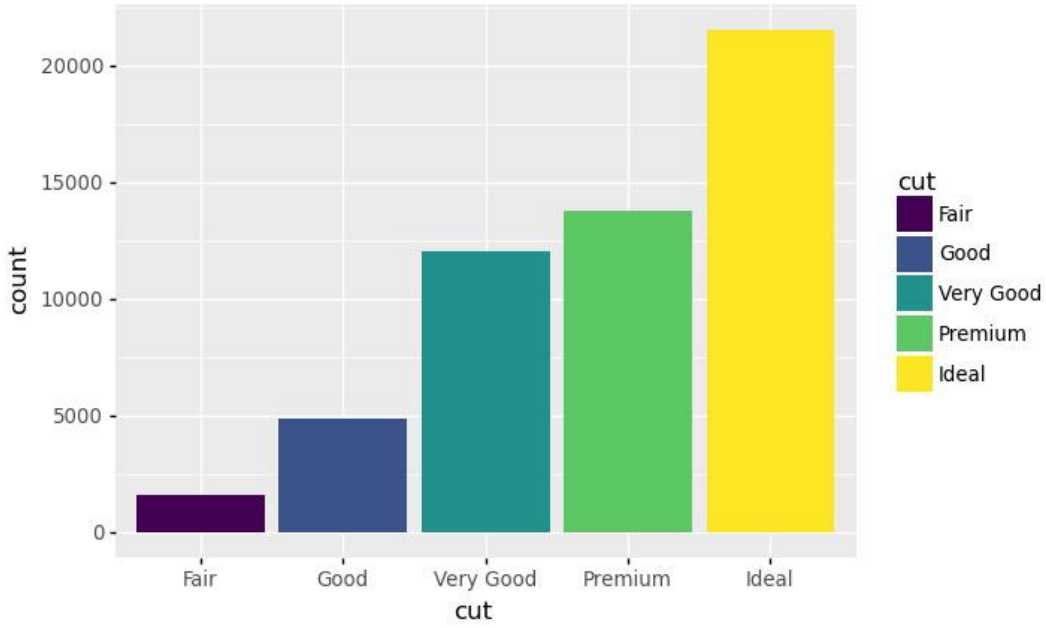
ggplot(data=diamonds) +

geom_bar(mapping=aes(x="cut", colour="cut"))

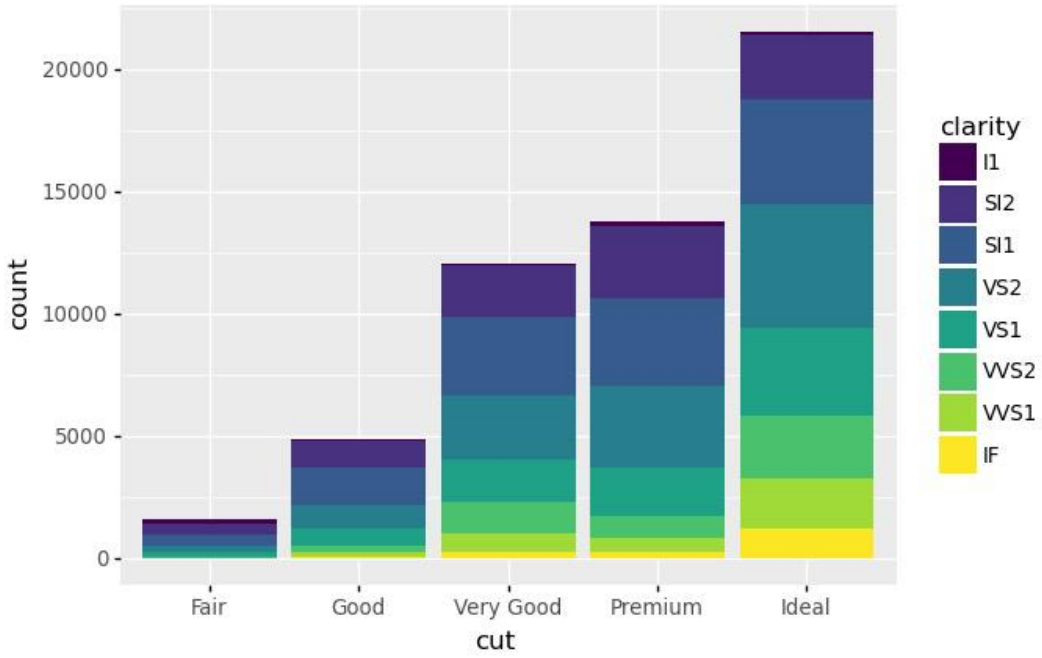


ggplot(data=diamonds) +

geom_bar(mapping=aes(x="cut", fill="cut"))



Дополнительная категория как цвет

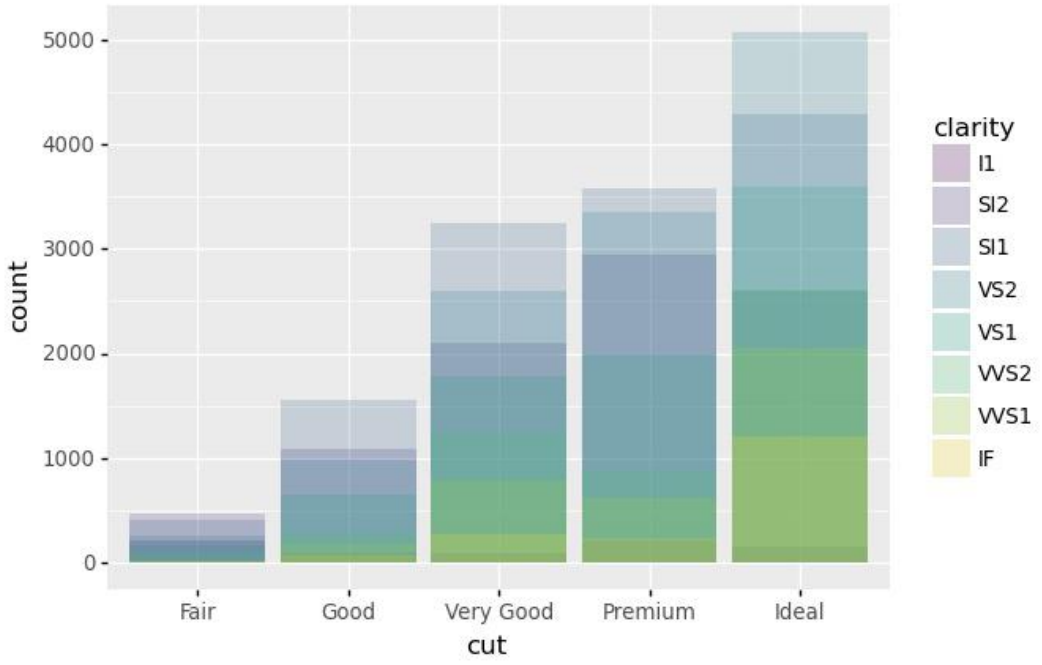


- ggplot(data=diamonds) +\
- geom_bar(mapping=aes(x="cut", fill="clarity"))

Уточнение позиции определяется аргументом 'position'

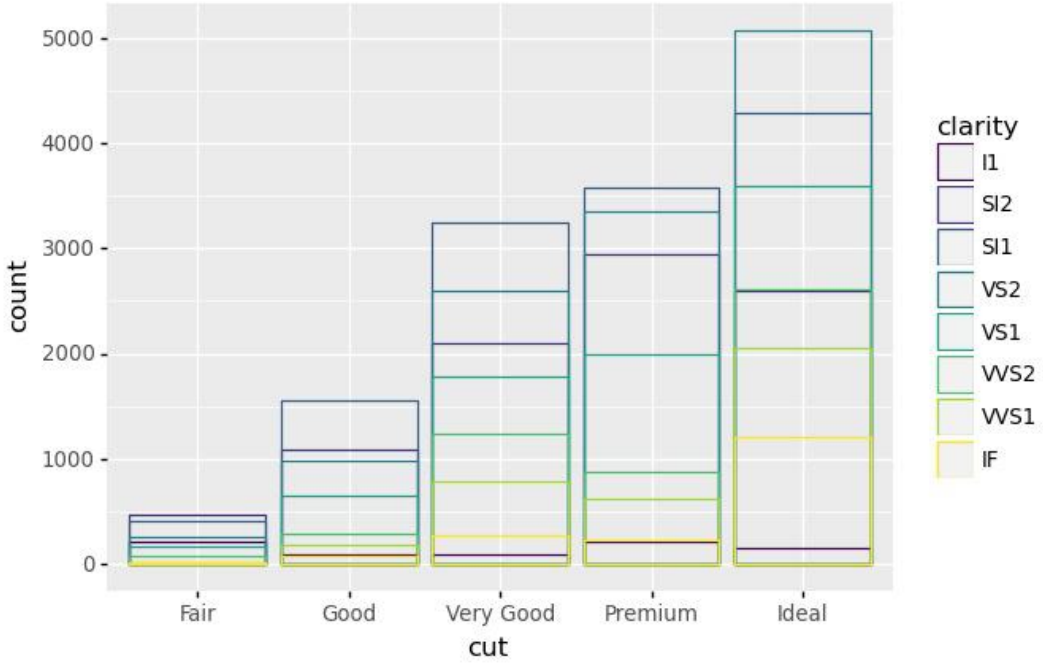
ggplot(data=diamonds, mapping=aes(x="cut", fill="clarity")) +

geom_bar(alpha=1/5, position="identity")



ggplot(data=diamonds, mapping=aes(x="cut", colour="clarity")) +

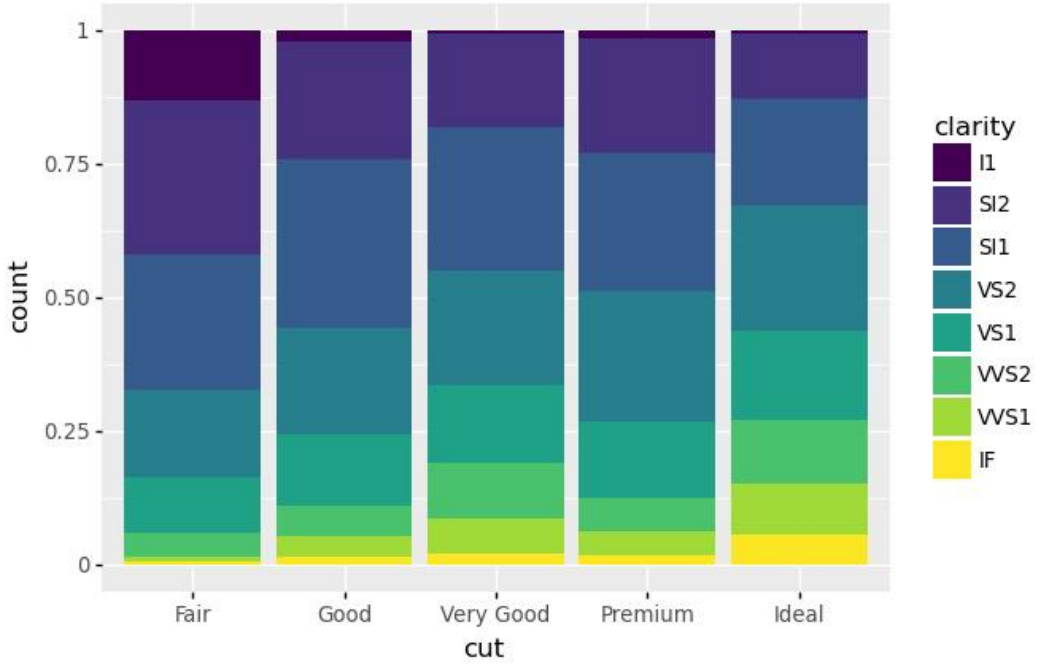
geom_bar(fill=None, position="identity")



Ещё два варианта для 'position'

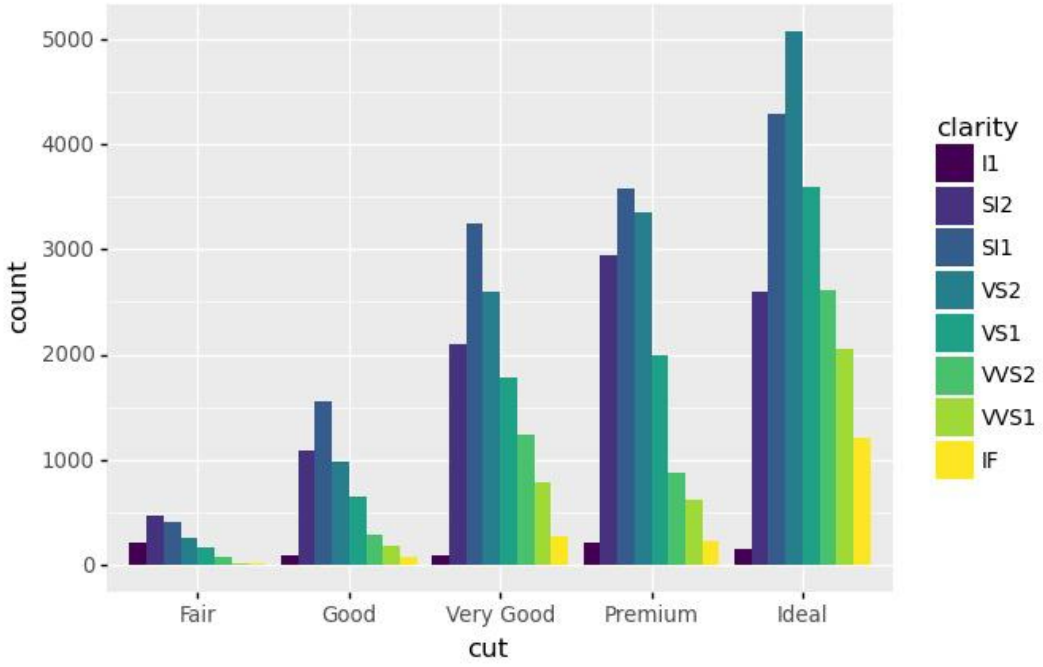
ggplot(data=diamonds) +

geom_bar(mapping=aes(x="cut", fill="clarity"), position="fill")

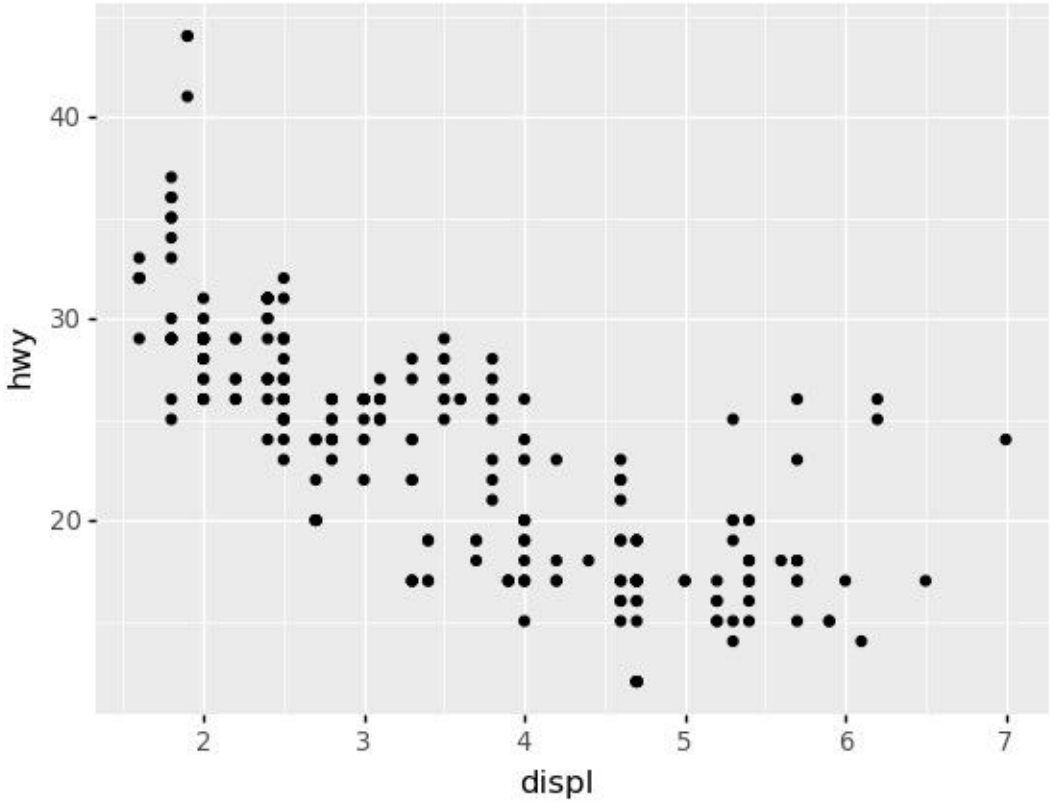


ggplot(data=diamonds) +

geom_bar(mapping=aes(x="cut", fill="clarity"), position="dodge")

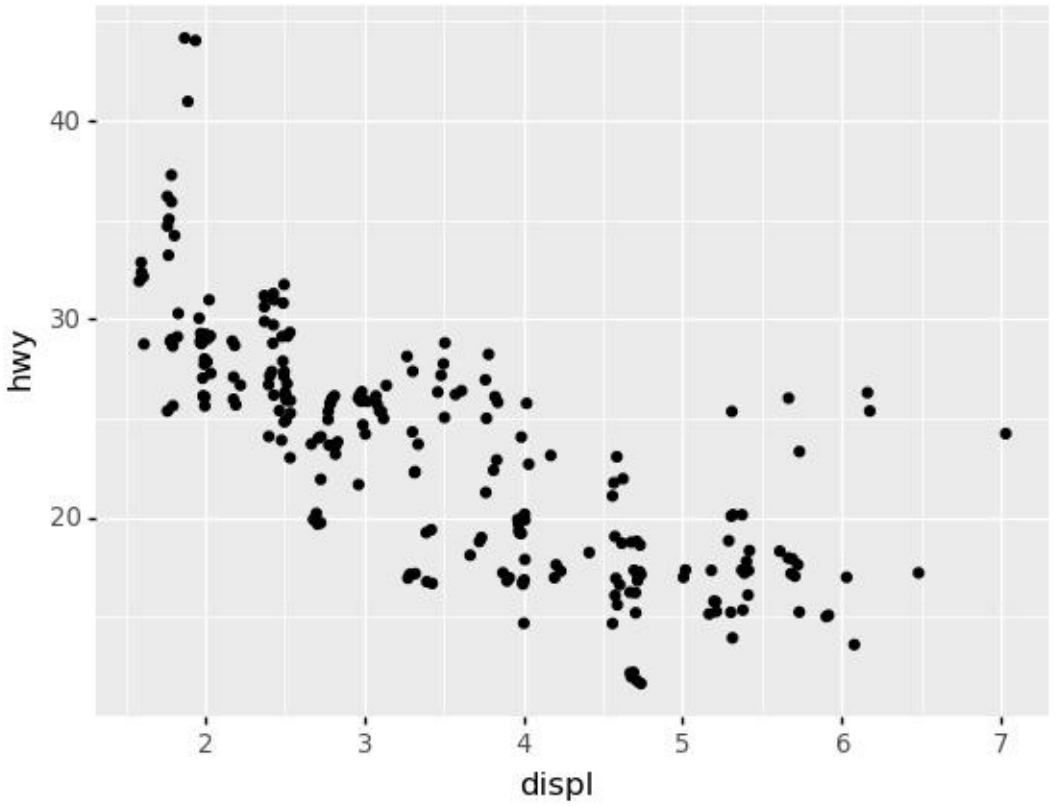


Заметили ли вы?



- [234 rows x 11 columns]
- А на графике всего 126 точек.
- Эффект наложения одинаковых (или почти одинаковых значений) называется overplotting.

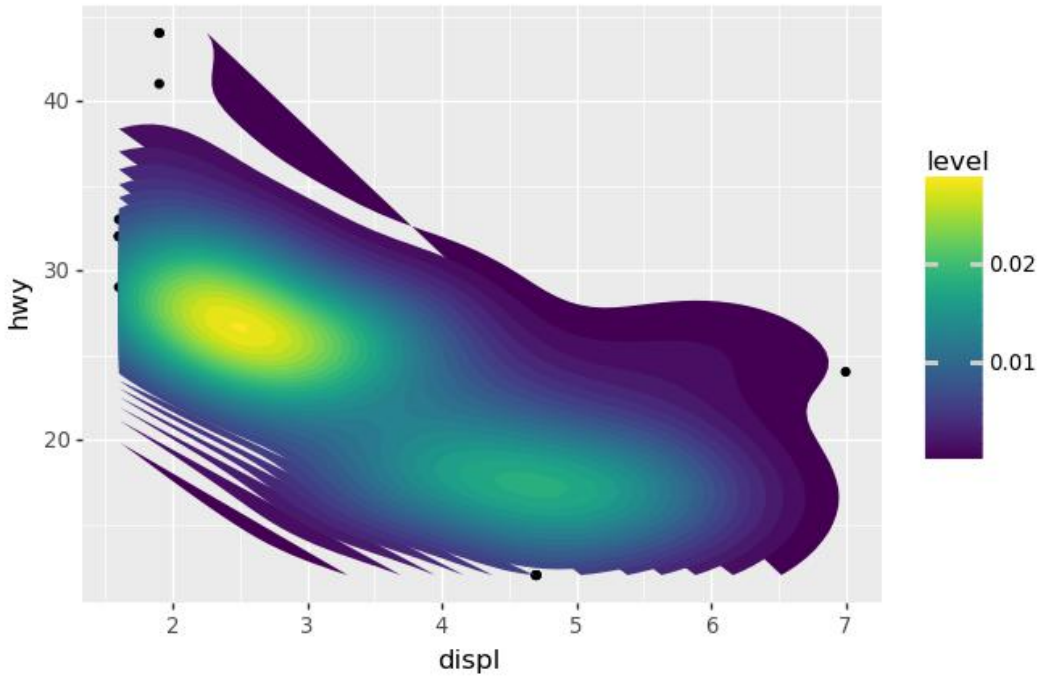
Как избежать overplotting'a?

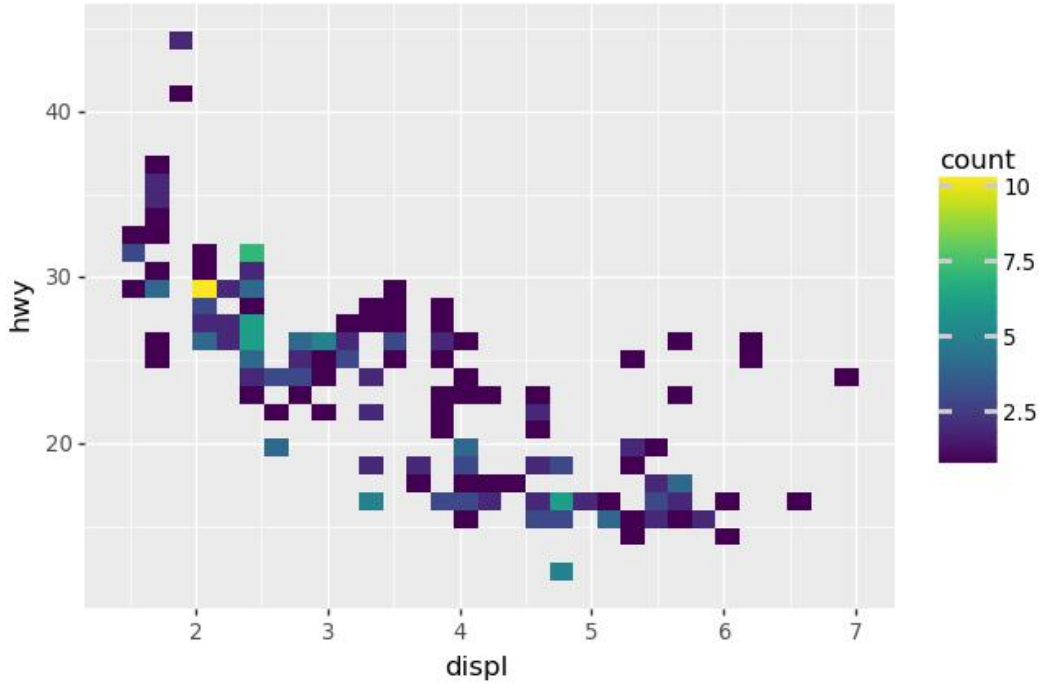


- Один из вариантов – добавить jitter к позиции точки

- ggplot(data=mpg) +\
- geom_point(mapping=aes(x="displ", y="hwy"), position="jitter")

Ещё пара способов

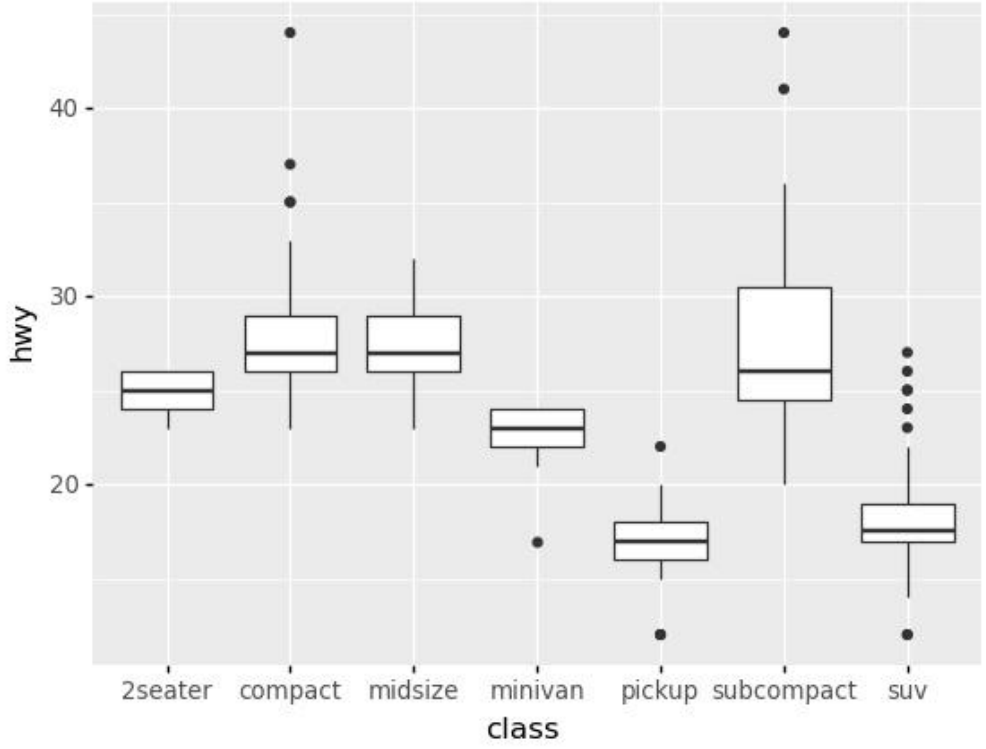




Смена координатных систем. Вдруг пригодится?

ggplot(data=mpg, mapping=aes(x="class", y="hwy")) +

geom_boxplot()



ggplot(data=mpg, mapping=aes(x="class", y="hwy")) +

geom_boxplot() + coord_flip()