Wizualizacja danych za pomocą ggplot2

Do tworzenia wykresów z naszych danych posłużymy się pakietem ggplot2.
Przy ostatniej instalacji tidyverse mogliśmy zauważyć, że pakiet ggplot2 został już zainstalowany,
Jeżeli z jakiegoś powodu tidyverse nie został zainstalowany, można to zrobić za pomocą polecenia:

install.packages(”tidyverse”)

oraz załadować bibliotekę za pomocą polecenia:

```
> library(tidyverse)
── Attaching packages ─────────────────────────────────────── tidyverse
1.2.1 ──
✔ ggplot2 2.2.1     ✔ purrr   0.2.4
✔ tibble  1.4.2     ✔ dplyr   0.7.4
✔ tidyr   0.8.0     ✔ stringr 1.3.0
✔ readr   1.1.1     ✔ forcats 0.3.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

Do testów skorzystamy z ramki danych zawartych w pakiecie ggplot2 :
mpg.Sprawdźmy jakie dane zawiera owa ramka:

```
> ?mpg
>
```

Spróbujmy odpowiedzieć na pytanie jaka jest zależność pomiędzy pojemnością silnika a wykorzystaniem paliwa? Czy silniki z dużą pojemnością silnika wykorzystują więcej paliwa niż te z małą?
Jakie dane z ramki mpg można wykorzystać aby odpowiedzieć na te pytania?

Przydzadzą się pojemność silnika w litrach : displ,
i np.: cty – mile przejechane w mieście na galon paliwa, oraz hwy – to samo tylko na autostradzie.

Narysujmy prosty wykres:

```
> ggplot(data = mpg)+
+ geom_point(mapping = aes(x = displ, y = cty))
```

lub:

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

Jak widać do tworzenia wykresów służy funkcja ggplot().
Pierwszy parametr jaki podajemy to zbiór danych na podstawie którego chcemy utworzyć wykres.
Co stanie się jeżeli użyjemy tylko samej funkcji `ggplot(data = mpg)`?
Widzimy pusty wykres, na który możemy nakładać warstwy.
Aby to zrobić używamy funkcji geometrycznej `geom_point()`. W tym wypadku utworzy nam się wartstwa z wykresem punktowym. Funkcji tworzących warsty jest oczywiście dużo dużo więcej.
Ściągawka np. tutaj:
https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Jak widać funkcje geometryczne używają argumentu `mapping`, która używa funkcji `aes()` .
Argumenty tej funkcji będą określały które zmienne użyć do zmapowania na osi x i y.

Ćwiczenia 1.

1. Co widzimy po wywołaniu polecenia ggplot(data = mpg)
2. Ile wierszy i kolumn znajduje się w zbiorze danych mtcars?
3. Czym jest zmienna drv z ramki danych mpg?
4. Wykonaj wykresy zależności pomiędzy zmiennymi cty i cyl oraz hwy i cyl
5. Wykonaj wykres punktowy zależności pomiędzy class i drv. Co widzimy?

Przyjrzyjmy się wykresowi zależności pomiędzy zmiennymi displ oraz hwy.

Widzimy ze kilka punktów przy dużych (5 i więcej) pojemnościach wygląda nietypowo.

Jakie może być tego wyjaśnienie?

Z pomocą może przyjść nam 3 zmienna. (w tym wypadku posłużymy się zmienna class)

Możemy zmapować ją jako estetykę wykresu.

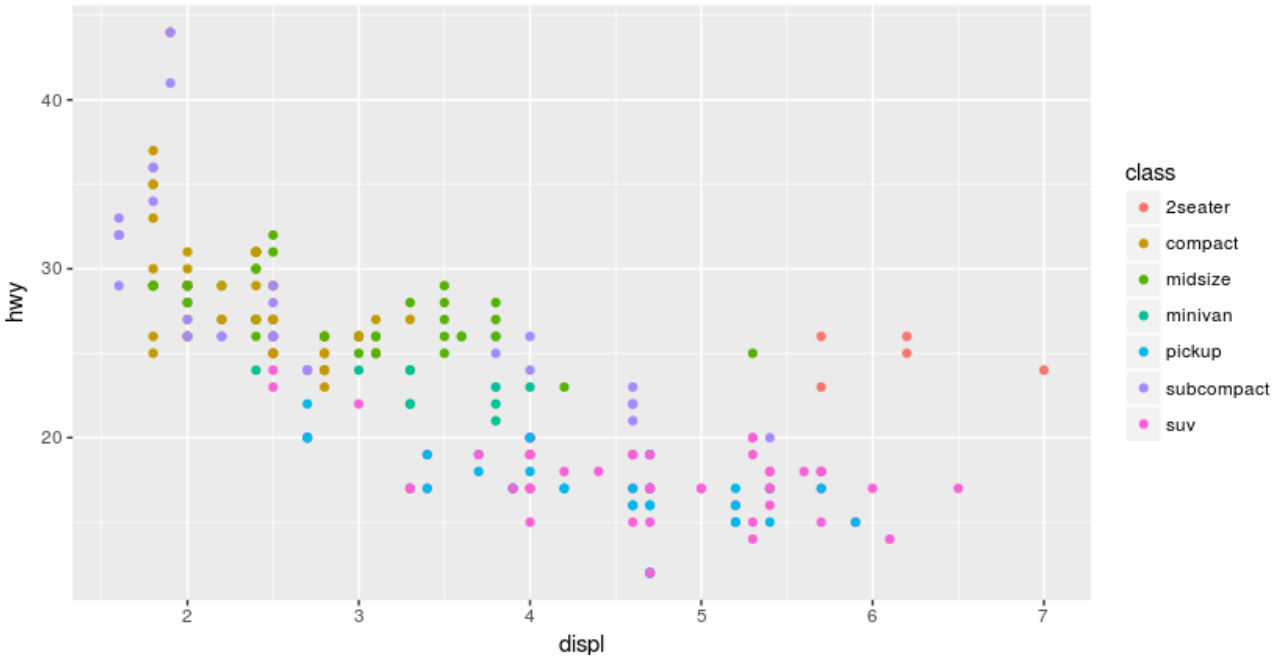
Jest to wizualna właściwość obiektów wykresu. W przypadku punktów, może to być wielkość, kolor, kształt oraz rozmiar.

Spróbujmy zobaczyć jakiej klasy są samochody posiadające nietypową zależność pomiędzy displ i hwy.:

```
> ggplot(data = mpg) +
+ geom_point(mapping = aes(x = displ, y = hwy, color = class))
>
```

Estetykę tworzymy powiązując w funkcji aes zmienna z nazwą wybranej estetyki.

W tym wypadku jest to kolor - ggplot przypisuje do każdej unikatowej wartości zmiennej inny kolor , oraz generuje legendę. Jest to proces skalowania.



Jakie wnioski możemy wyciągnąć na podstawie przygotowanego wykresu?

Przeskaluj zmienną class na estetykę alpha, rozmiar (size) oraz kształt (shape)

Co ciekawego widzimy?

Czy x i y też są estetyką? Jeżeli tak to gdzie jest legenda?

Co stanie się jeżeli umieścimy którąś z poznanych estetyk poza funkcją aes?

A co jeżeli zamiast nazwy klasy użyjemy np. nazwy koloru?

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

Poza funkcją aes można też uzyć rozmiaru punktów w milimetrach,

oraz kształtu punktu jako liczba.

Ćwiczenia2

1.W którym miejscu tego kodu znajduje się błąd? Dlaczego punkty nie są zielone?

```
 ggplot(data = mpg) +
+ geom_point(mapping = aes(x = displ,y=cty, color="green"))
```

2.Które zmienne ze zbioru mpg są kategorialne? Które zmienne są ciągłe? (Wskazówka: wpisz ? mpg, aby przeczytać dokumentację tego zbioru danych).

3.Zmapuj zmienną ciągłą na estetykę color, size i shape. Na czym polega różnica w zachowaniu tych estetyk w przypadku zmiennych kategorialnych i ciągłych?

4.Co się stanie, jeśli zmapujesz tę samą zmienną na wiele estetyk?

5.Do czego służy estetyka stroke? Z jakimi kształtami można jej użyć? (Wskazówka: skorzystaj z polecenia ?geom_point).

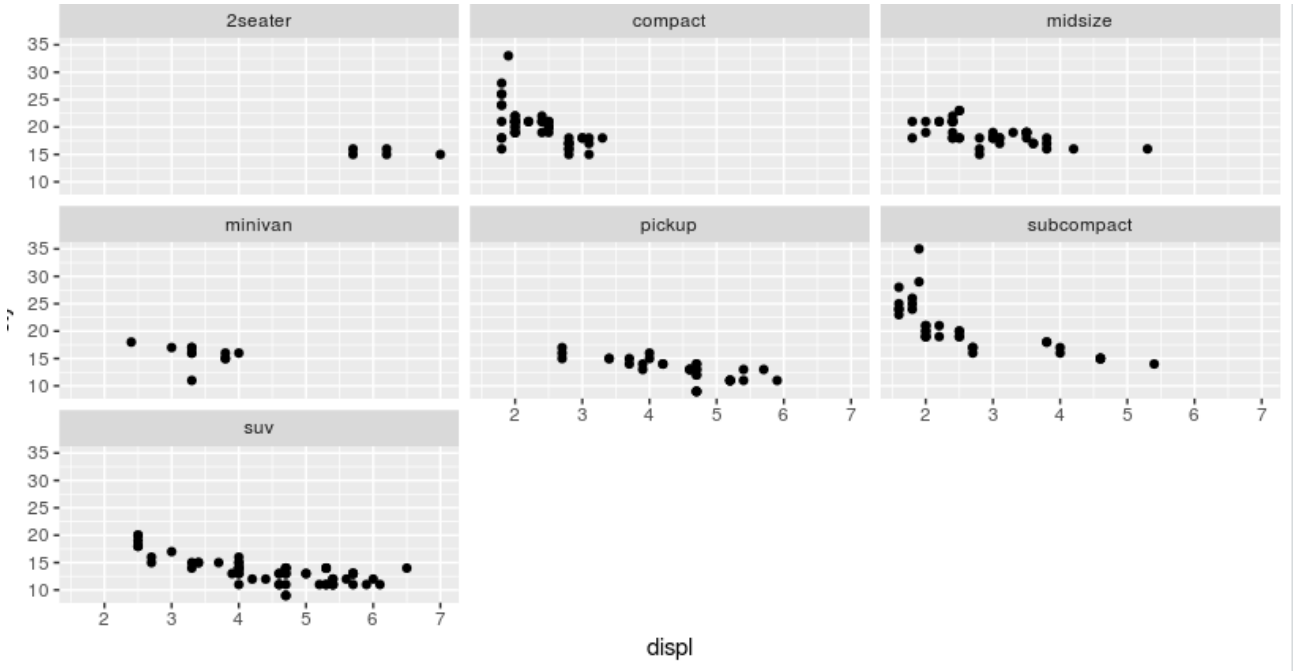
6.Co się stanie, jeśli zmapujesz estetykę na coś innego niż nazwa zmiennej, jak na przykład aes(color = displ < 4)?

Kolejnym sposobem na umieszczenie wielu zmiennych na wykresie są Panele.

Wykres dzielimy na wykresy cząstkowe, które wyświetlają podzbiór danych.

Prześledźmy tworzenie paneli:

```
> ggplot(data = mpg) +
+ geom_point(mapping = aes(x = displ, y = cty)) +
+ facet_wrap(~ class, nrow=3)
```

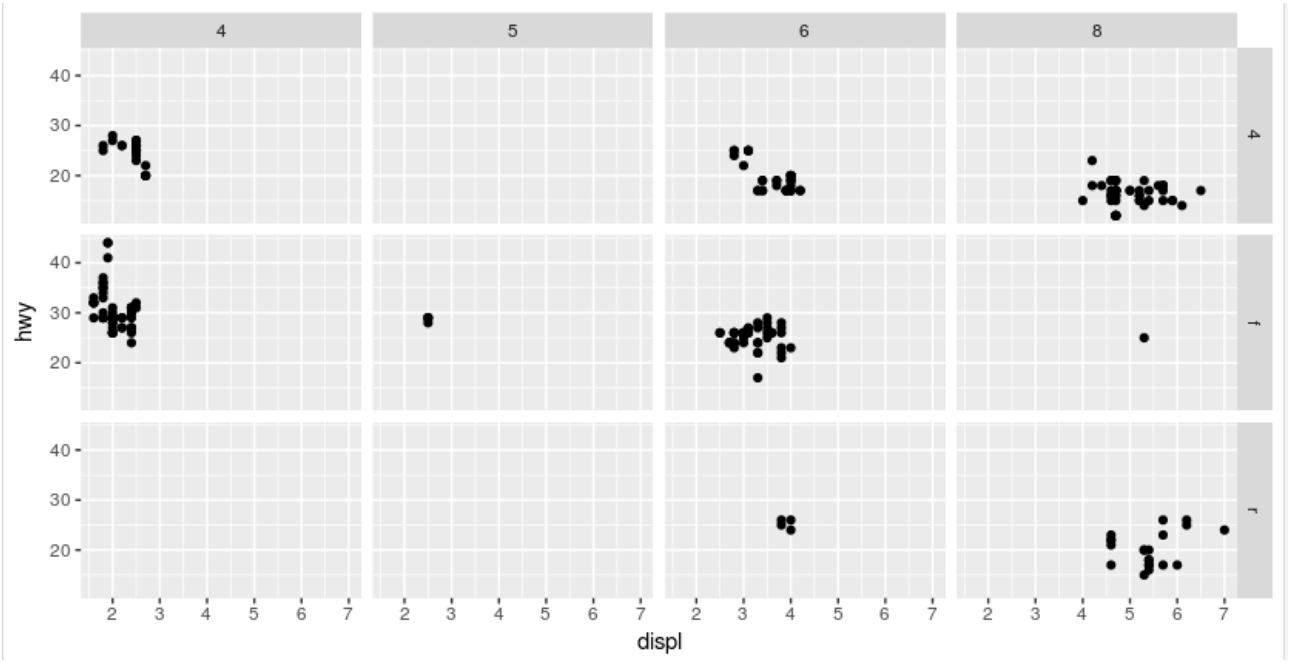


Do utworzenia paneli używamy funkcji facet_wrap, wraz z parametrem ~ nazwaZmiennej.

Pamiętajmy, że zmienna powinna być kategorialna.

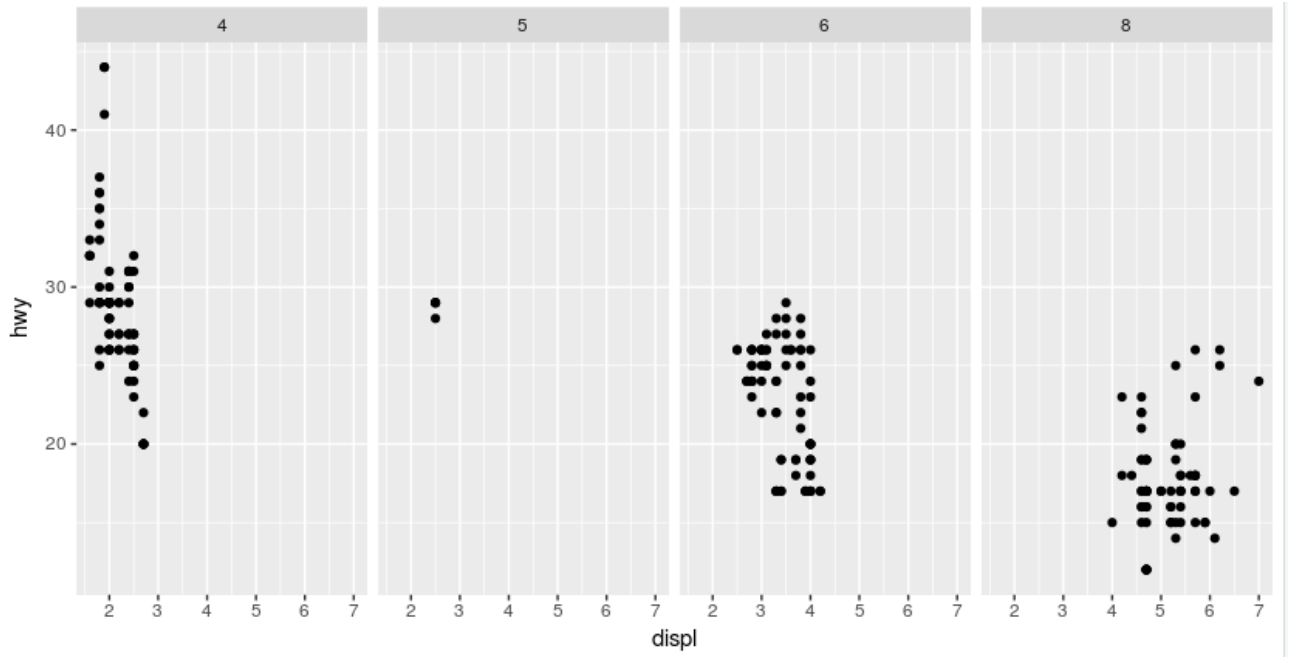
Aby utworzyć panele według dwóch zmiennych używamy funkcji facet_grid:

```
> ggplot(data = mpg) +
+     geom_point(mapping = aes(x = displ, y = hwy)) +
+     facet_grid(drv ~ cyl)
```



Dwie zmienne rozdzielamy tyldą.

Zmienną możemy zastąpić kropką:



Ćwiczenia3

1. Co się stanie jeżeli podzielimy wykres na panele używając zmiennej ciągłej?

2. Co oznaczają puste komórki na wykresie:

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

Jaka jest zależność względem tego wykresu:

```
ggplot(data = mpg) +
+     geom_point(mapping = aes(x = drv, y = cyl))
```

3Jakie wykresy powstaną po uruchomieniu polecenia poniżej? Do czego służy znak „.”?

```
ggplot(data = mpg) +
+     geom_point(mapping = aes(x = displ, y = hwy)) +
+     facet_grid(drv ~ .)
```

```
ggplot(data = mpg) +
+     geom_point(mapping = aes(x = displ, y = hwy)) +
+     facet_grid(. ~ cyl
```

4 Jakie korzyści daje używanie paneli zamiast estetyk?

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

5. Do czego służ argument nrow ? Ncol ? Jakie inne opcje kontrolują układ poszczególnych paneli?

Porównajmy 2 wykresy:

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

```
> ggplot(data = mpg) +
+     geom_smooth(mapping = aes(x = displ, y = hwy))
```

Oba wykresy przedstawiają tą samą zależność. To co je różni to użycie innego obiektu geometrycznego, czyli reprezentacja danych.

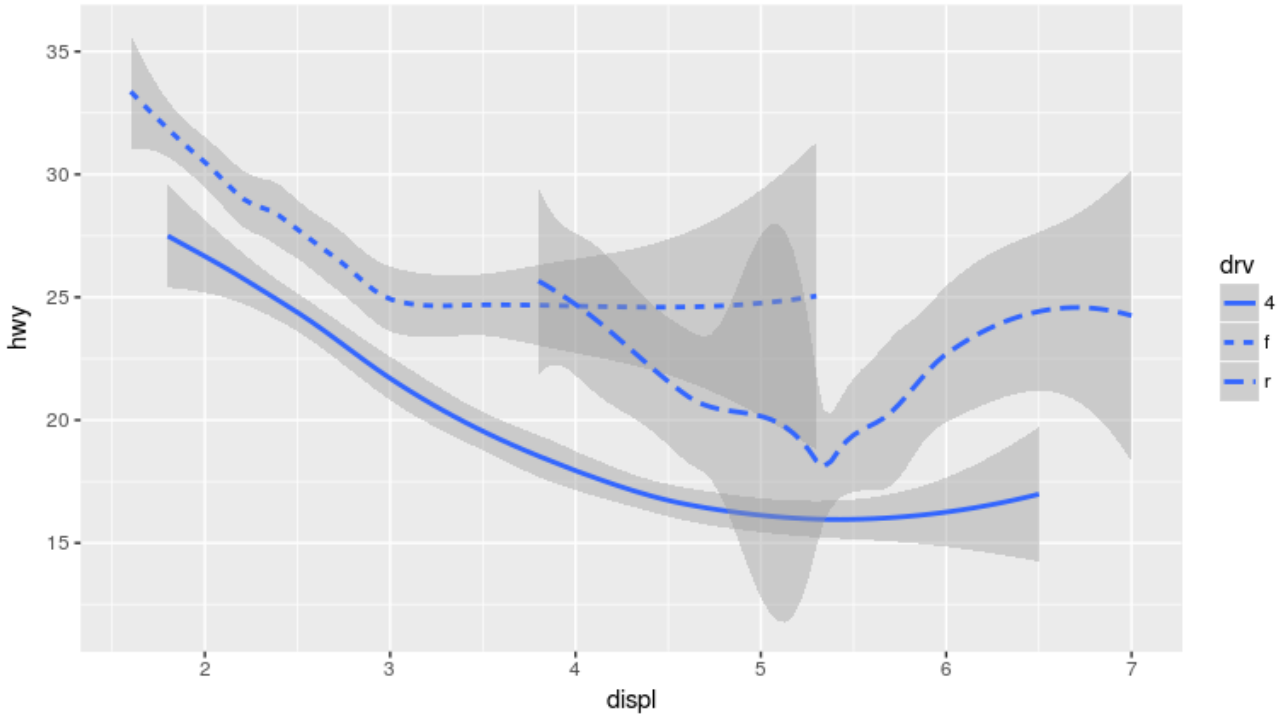
W pierwszym użyto znanej nam funkcji reprezentujacej dane za pomocą punktów.

W drugim użyto linii gładkiej.

Jak pamiętamy funkcje geometryczne używają parametru mapping. Czy możemy zastosować w funkcji liniowej estetyki z funkcji rysujacej punkty?

Zobaczmy przykład użycia estetyki przeznaczonej funkcji geom_smooth:

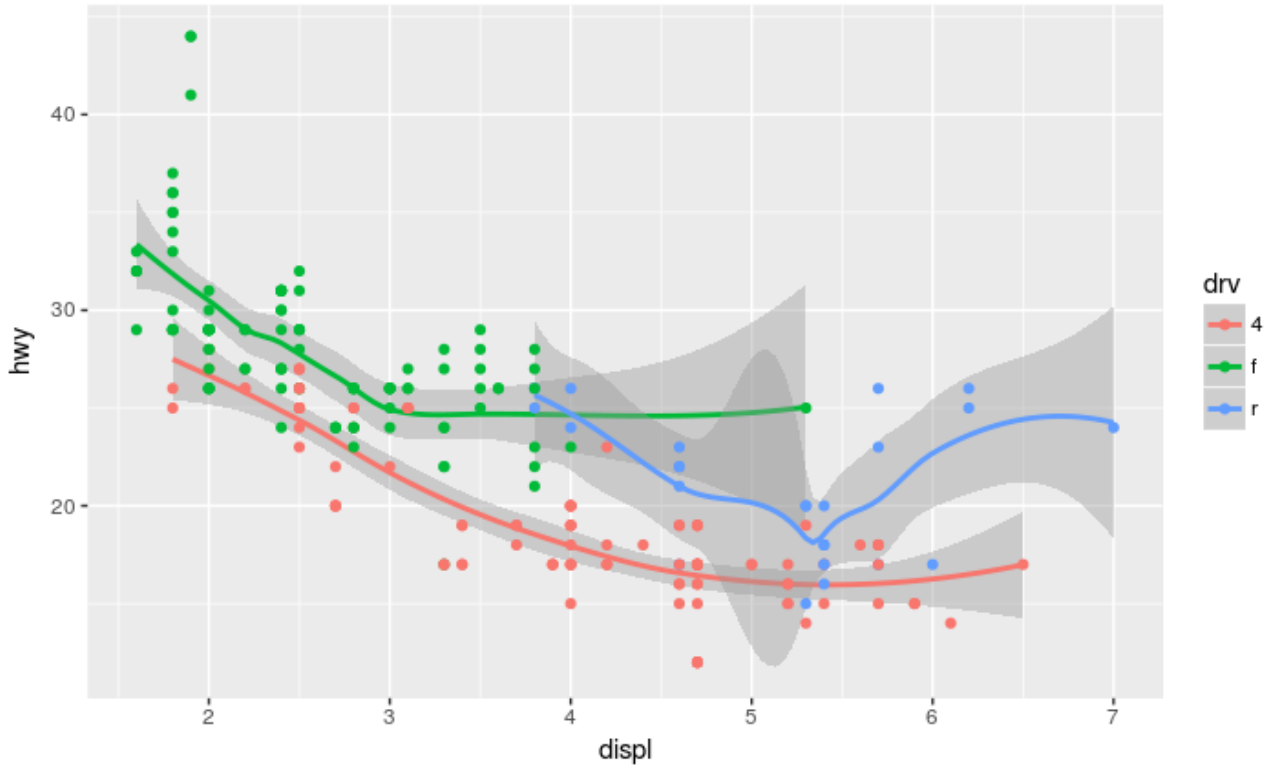
```
> ggplot(data = mpg) +
+ geom_smooth(mapping = aes ( x = displ, y = hwy, linetype = drv ))
```



Wyświetlamy tu znaną już nam zależność , rozróżniając napęd samochodu.

Wykres może okazać się ciekawszy gdy nałożymy nasze linie na punkty, które znamy z poprzedniej części.

```
> ggplot(data = mpg) +
+ geom_smooth(mapping = aes(x = displ, y = hwy,color=drv))+
+ geom_point(mapping=aes(x=displ,y=hwy,color=drv))
```

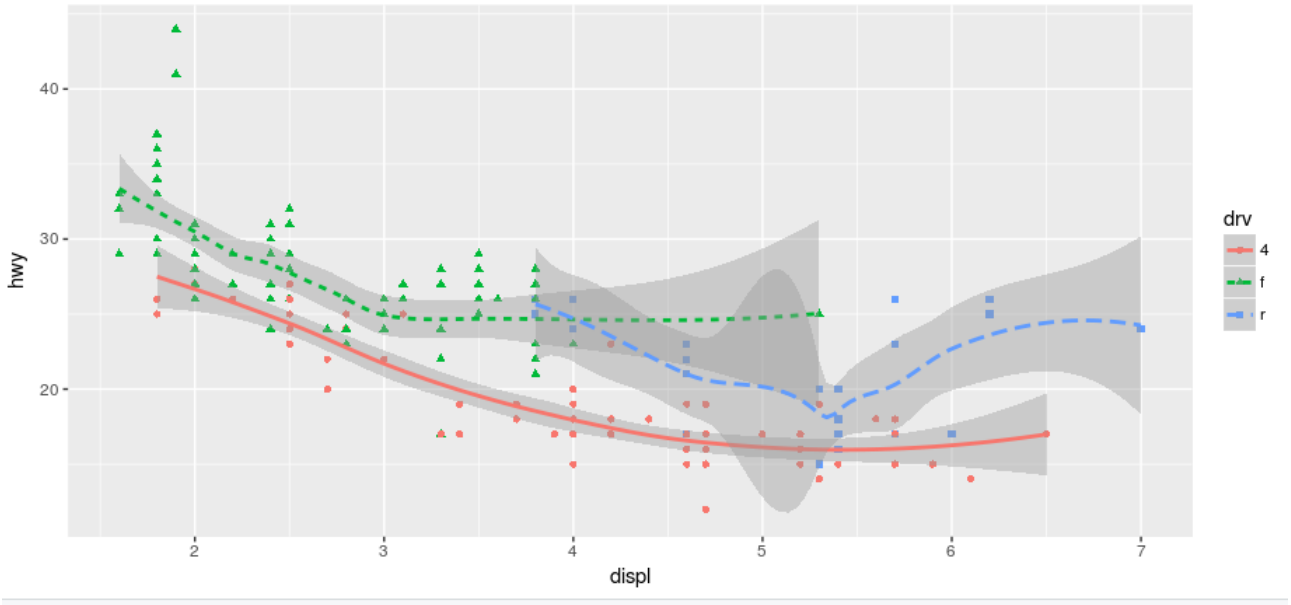


Aby nie powtarzać 2 razy tego samego kodu przekazując zestaw paowań do funkcji ggplot():

```
> ggplot(data = mpg, mapping = aes ( x = displ, y = hwy,color=drv ))+
+ geom_point()+
+ geom_smooth()
```

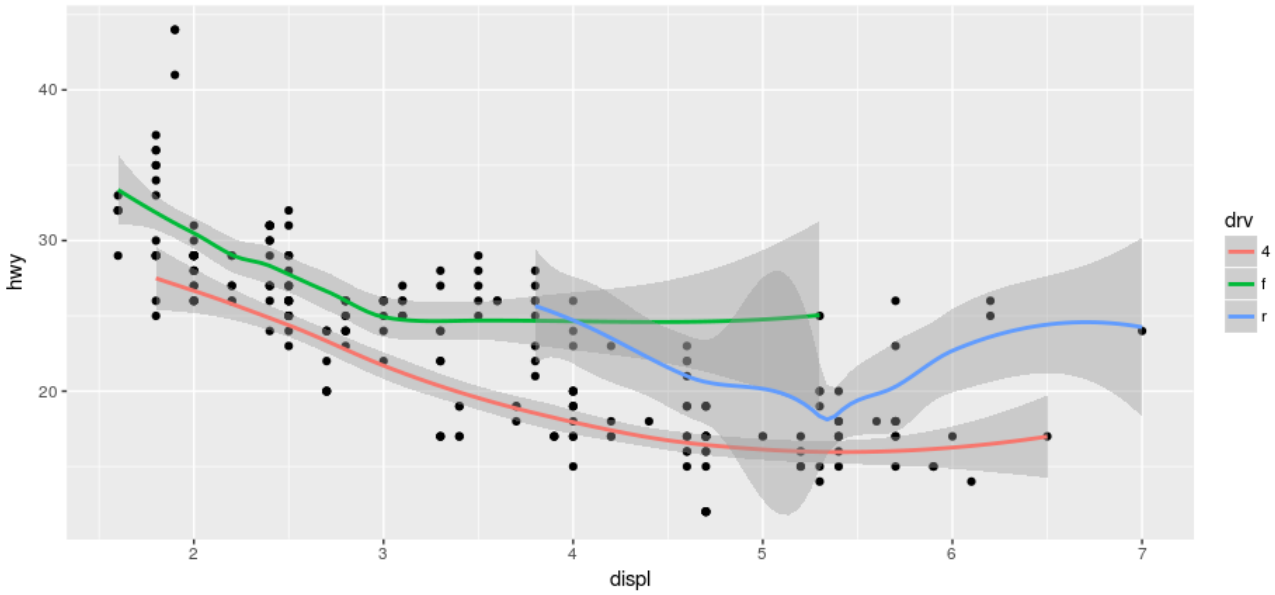
Można też wykorzystywać różne estetyki:

```
 ggplot(data = mpg, mapping = aes ( x = displ, y = hwy,color=drv,shape=drv,linetype=drv ))+
+ geom_point()+
+ geom_smooth()
```



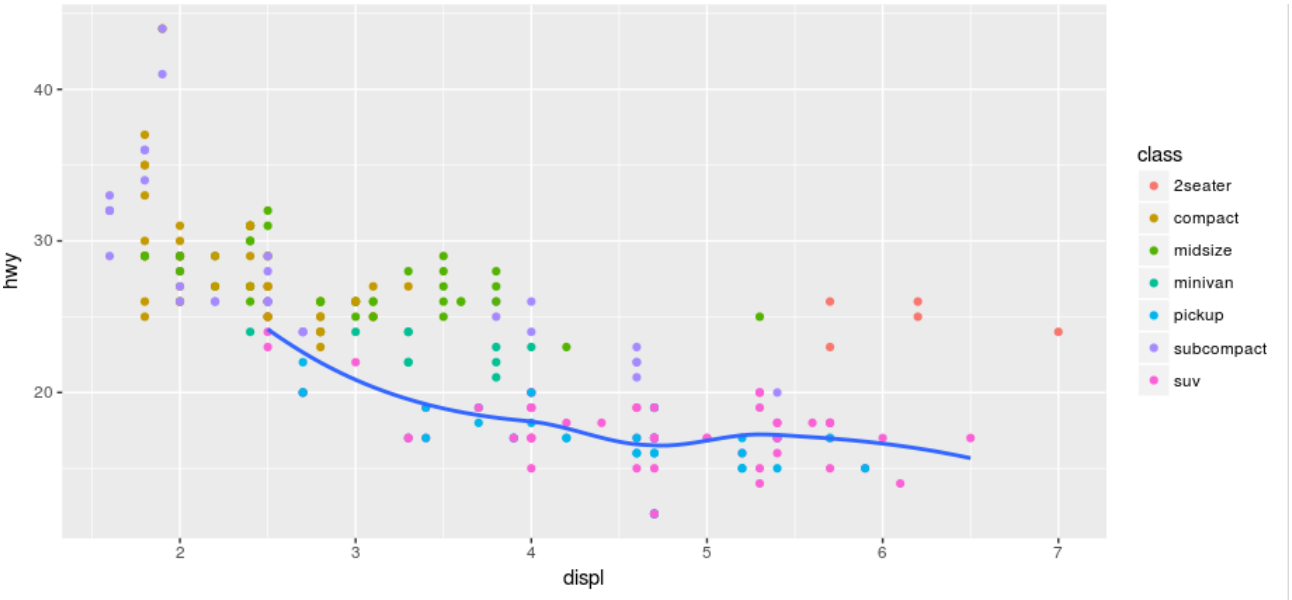
Mapowania można też przesłaniać:

```
>  ggplot(data = mpg, mapping = aes ( x = displ, y = hwy ))+
+ geom_point()+
+ geom_smooth(mapping = aes (color=drv))
```



Możemy w podobny sposób mapować zestaw danych do rysowania.

```
> ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
+ geom_point(mapping = aes(color = class))+
+ geom_smooth(data = filter(mpg, class=="suv"),se = FALSE)
```



Ćwiczenia4

1. Jakiej geometrii użyjesz, aby narysować wykres liniowy? Wykres pudełkowy? Histogram? Wykres warstwowy?

2. Spróbuj przewidzieć działanie polecenia poniżej. Następnie sprawdź czy miałeś rację uruchamiajac je.

```
> ggplot(data = mpg, mapping = aes( x = displ, y = hwy, color = drv))+
+ geom_point()+
+ geom_smooth(se = FALSE)
```

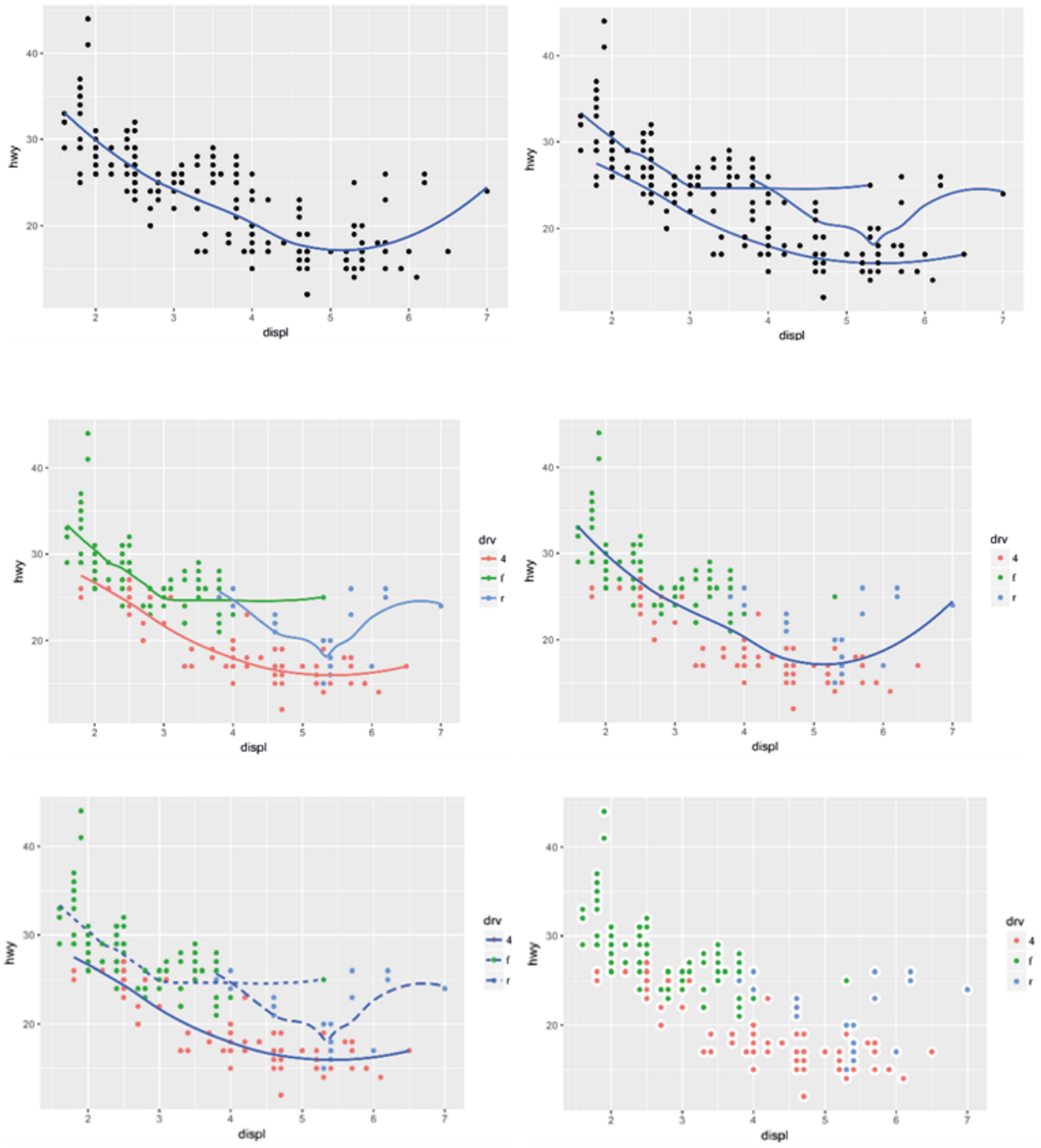
3.Do czego służy kod show.legend = FALSE?

4.Do czego służy argument se w funkcji geom_smooth()?

5.Czy te dwa wykresy są inne? Dlaczego?

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

6.Odtwórz kod R potrzebny do wygenerowania następujących wykresów:

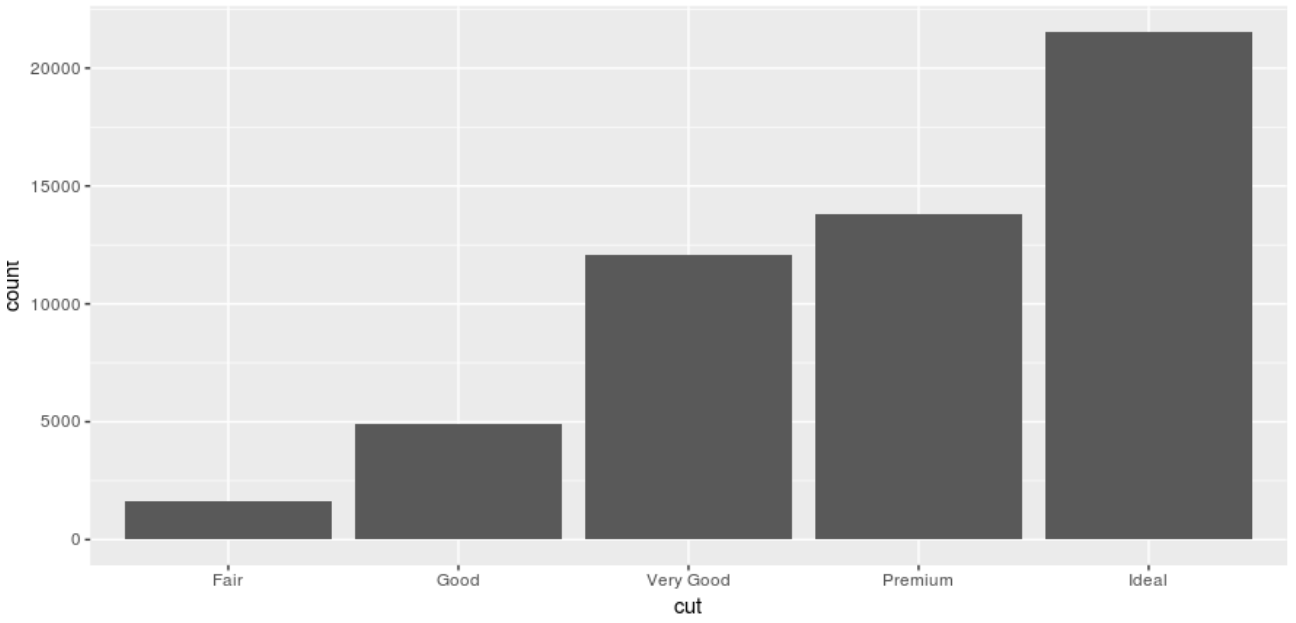


Następna funkcją jaką wykorzystamy będzie geom_bar(),

Wykorzystamy tym razem zbiór danych dotyczacych diamnentów.

Zobaczmy jakiej jakości szifu diamenty mamy w zestawie:

```
> ggplot(data = diamonds)+
+ geom_bar(mapping = aes(x = cut))
```



Jak widzimy tym razem wykres zgrupował nasze dane po zmiennej cut a następnie policzył ilość wystapień w danej grupie.

*Wykresy słupkowe, histogramy i wieloboki częstotliwości grupują dane, a następnie prezentują liczbę elementów znajdujących się w poszczególnych grupach.

*Na wykresach liniowych model jest dopasowywany do danych, a następnie wykreślane są przewidywania wyznaczone przez model.

*Wykresy pudełkowe obliczają kompleksowe podsumowanie rozkładu wartości i wyświetlają specjalnie sformatowane prostokąty.

Algorytm który wylicza nowe wartości wykresu nosi nazwe stat. Jest to przekształcenie statystyczne. Możemy zobaczyć jakie jest wykorzystywany w danej funkcji wywołując pomoc.

np.: ?geom_bar

`geom_bar` uses `stat_count` by default: it counts the number of cases at each x position.

Funkcji statystycznych można używać zamiennie z funkcją geometryczną:

```
ggplot(data = diamonds) +
+ stat_count(mapping = aes(x = cut))
```

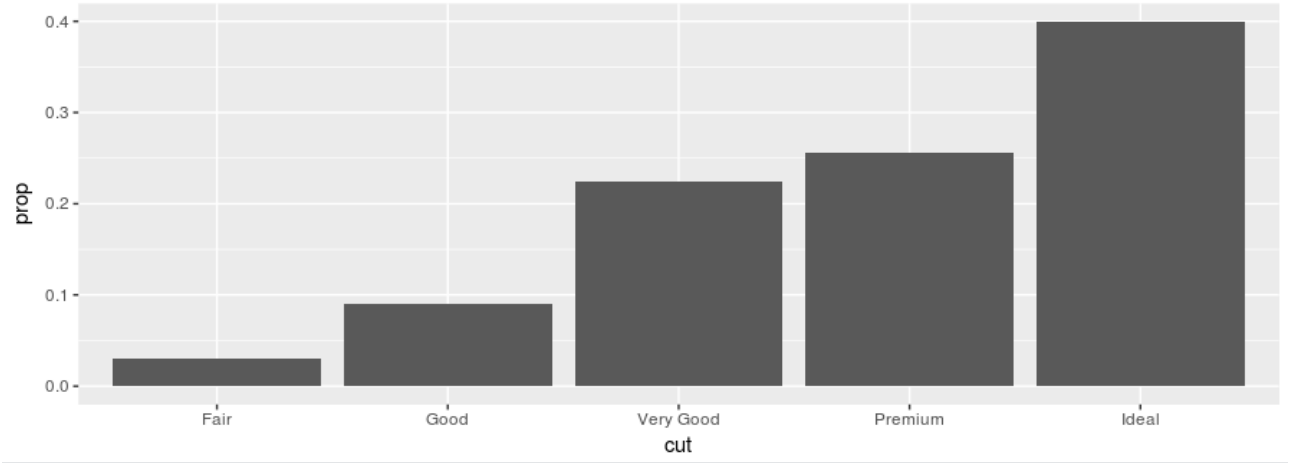
Przekształcenia statystyczne możemy też przedstawiać, poniżej używamy przekszałcenia statystycznego identity zamiast domyslnego count. Czyli wysokość słupków będzie określona zmienną b zamiast domyślnie liczebnością .

```
> dane <- tribble(
+ ~a,~b,
+ "b1",66,
+ "b2", 33,
+ "b3", 11,
```

```
+ "b4", 6)
ggplot(data = dane)+
+ geom_bar(
+ mapping = aes(x = a, y = b), stat = "identity" )
```

Możemy też przesłonić domyslne mapowanie przekształconych zmiennych na estetyki. Np. wyświetlmy wykres słupkowy proporcji zamiast licznosci:

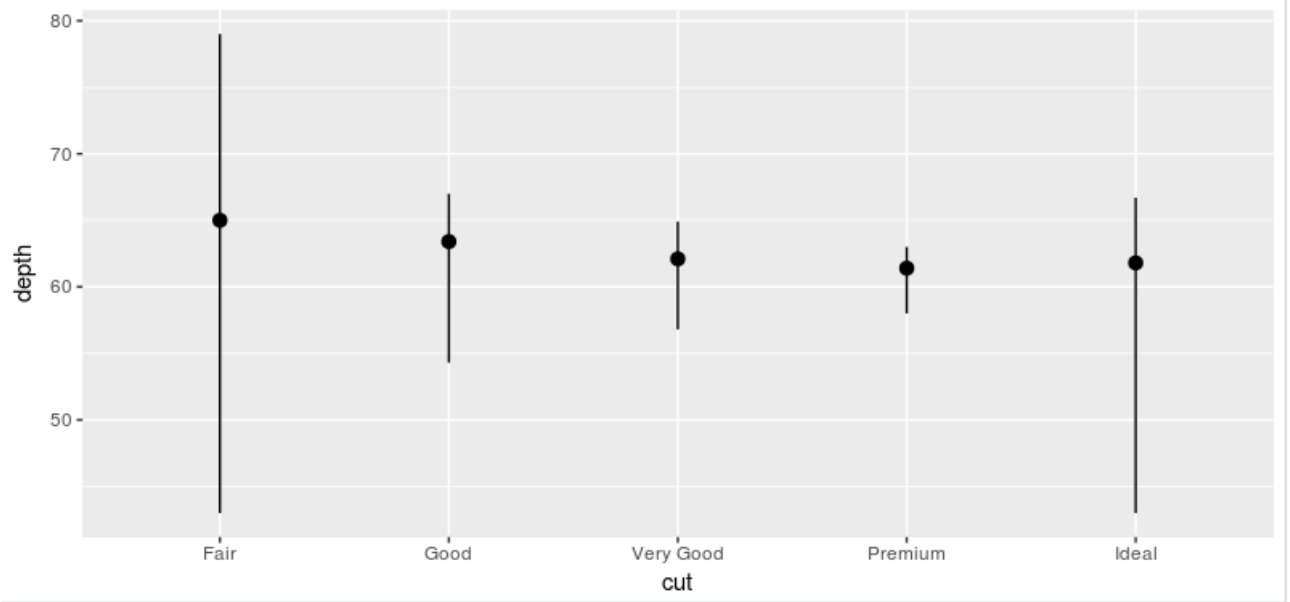
```
ggplot(data = diamonds) + geom_bar(mapping = aes(x = cut, y = ..prop..,group =0 ))
```



Zmienne obliczane przez przekształcanie statystyczne znajdują się w sekcji pomocy „ Computed variables”.

Możemy skorzystać z funkcji stat_summary(), aby obliczyć sumaryczne wartości y dla każdej unikatowej wartosci x:

```
ggplot(data = diamonds) + stat_summary(mapping = aes(x = cut, y= depth),
+ fun.ymin = min,
+ fun.ymax = max,
+ fun.y = median )
```



Ćwiczenia5

1.Która domyślna funkcja geometryczna jest związana z funkcją stat_summary()? Jak można przepisać wcześniejszy wykres, aby skorzystać z funkcji geometrycznej zamiast z przekształcenia statystycznego?

2.Do czego służy funkcja geom_col()? Czym różni się od funkcji geom_bar()?

3.Większość funkcji geometrycznych i przekształceń statystycznych tworzy pary, które niemal zawsze są używane wspólnie. Przeczytaj dokumentację i wykonaj listę tych par. Co mają ze sobą wspólnego?

4.Jakie zmienne oblicza funkcja stat_smooth()? Jakie parametry sterują jej zachowaniem?

5.Na naszym wykresie słupkowym proporcji musieliśmy skorzystać z zapisu group = 1. Dlaczego? Innymi słowy, na czym polega problem z poniższymi wykresami?

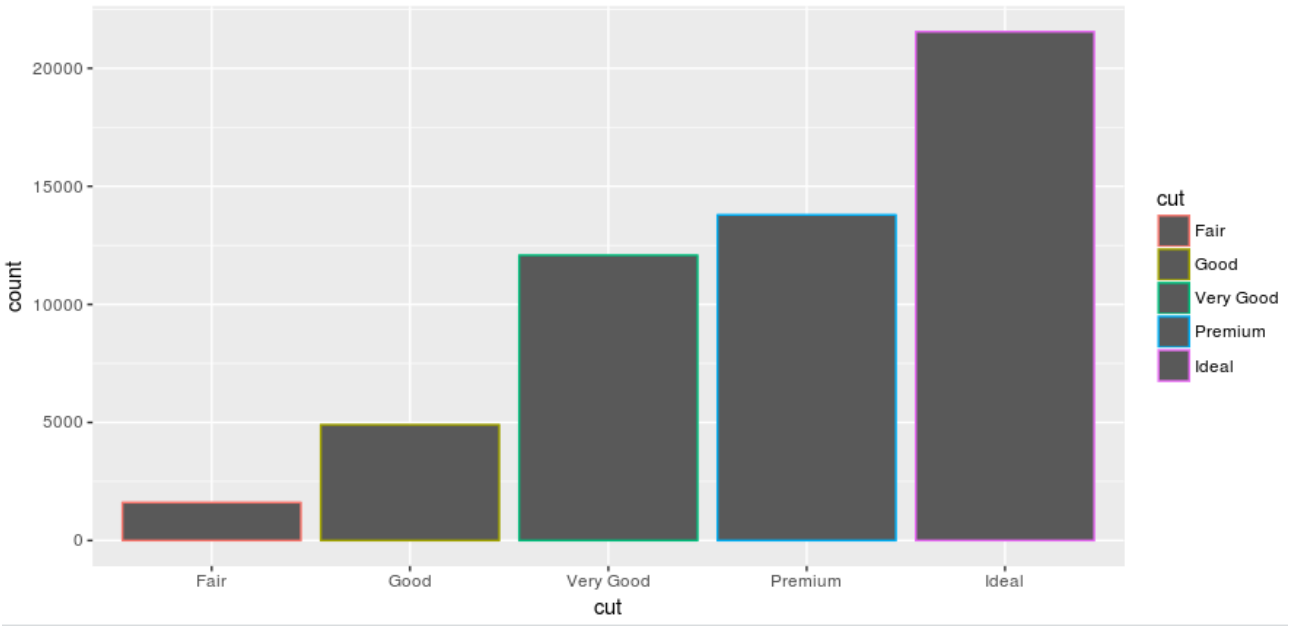
```
> ggplot( data = diamonds) +
+ geom_bar(mapping = aes(x = cut, y = ..prop..))
```

```
> ggplot(data = diamonds) + geom_bar(mapping = aes ( x = cut, fill = color, y = ..prop..))
```

Kolorowanie wykresu:

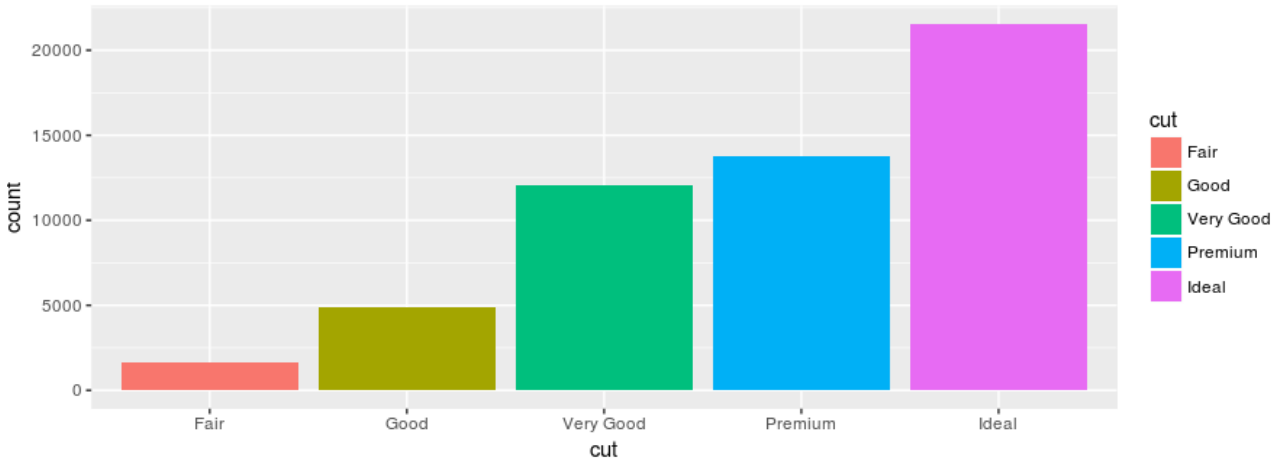
Obwódka:

```
ggplot(data = diamonds)+
+ geom_bar(mapping = aes(x = cut, color = cut))
```



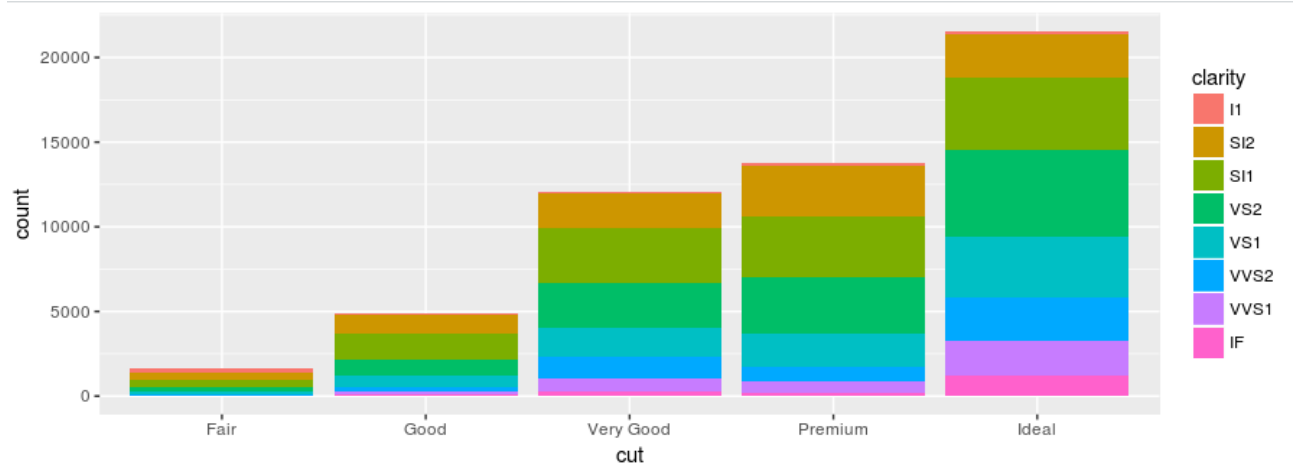
Wypełnienie:

```
ggplot(data = diamonds) +
+ geom_bar(mapping = aes(x = cut, fill = cut))
```



Co się stanie jeżeli będziemy chcieli wykorzystać inną zmienną w estetyce clarity?
np.:

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



Układanie słupków odbywa isę poprzez dostosowanie położenia na podstawie argumentu position.

Stworzyliśmy skumulowany wykres słupkowy.
Position może mieć jeszcze 3 inne wartości „identity”, „doge”, „fill”

W wypadku identity, możemy zauważyć nakładanie się na siebie słupków: (użyjmy mniejszej przezroczystosci dla porownania):

```
ggplot( data = diamonds, mapping = aes(x = cut, fill = clarity))+
+ geom_bar( position = "identity")
```

```
> ggplot( data = diamonds, mapping = aes(x = cut, fill = clarity))+
+ geom_bar(alpha = 1/5 , position = "identity")
```

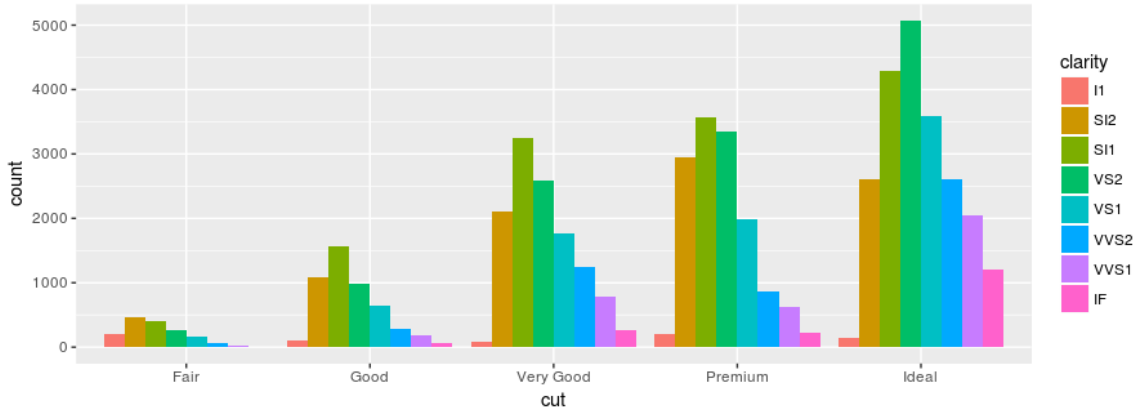
Co odróżnia parametr fill od skumulowanego wykresu słupkowego?

```
ggplot(data = diamonds)+
```

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

„dodge” umieszcza nakładające się obiekty obok siebie:

```
ggplot(data = diamonds) +
+ geom_bar( mapping = aes( x =cut, fill = clarity), position = "dodge")
```



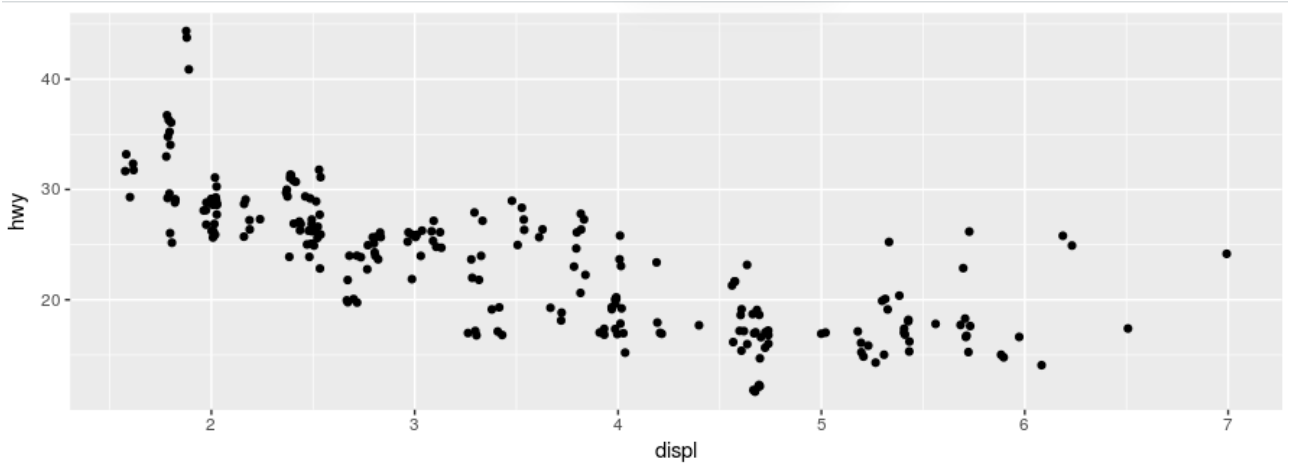
Przypomnijmy sobie wykres zależności hwy i displ. Był to wykres punktowy.

Wyrkes wyświelta o wiele mniej inforamcji niż wynikałoby to ze zbioru danych.

Wartości zmiennych są zaokrąglane i wyswietlane na siatce. Wiele punktów naklada się na siebie

(tak zwany overplotting) Możemy tego uniknąć wybierając pozycję position = „jiter” , do każdego punktu dodana zostanie mała ilosć slosowego szumu. A punkty oddala się od siebie.

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



Ćwiczenia6

1.Na czym polega problem z tym wykresem? Jak można go poprawić?`ggplot(data = mpg, mapping = aes(x  = cty, y = hwy )) + geom_point()`

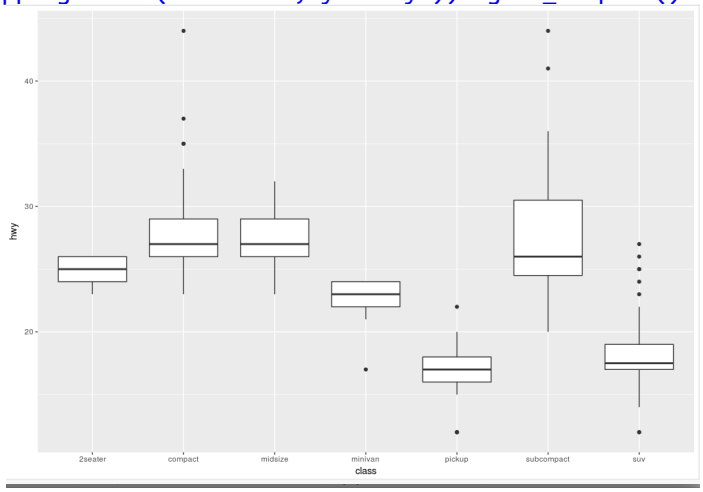
2.Jakie parametry funkcji geom_jitter() sterują poziomem fluktuacji?

3.Porównaj ze sobą funkcje geom_jitter() i geom_count().

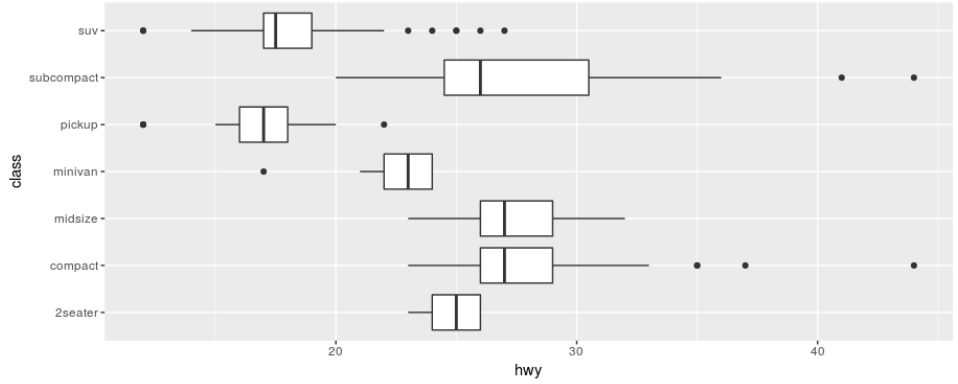
4.Jakie jest domyślne dopasowanie położenia dla geom_boxplot()? Utwórz odpowiednią wizualizację zestawu danych mpg.

Wykres pudełkowy:

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

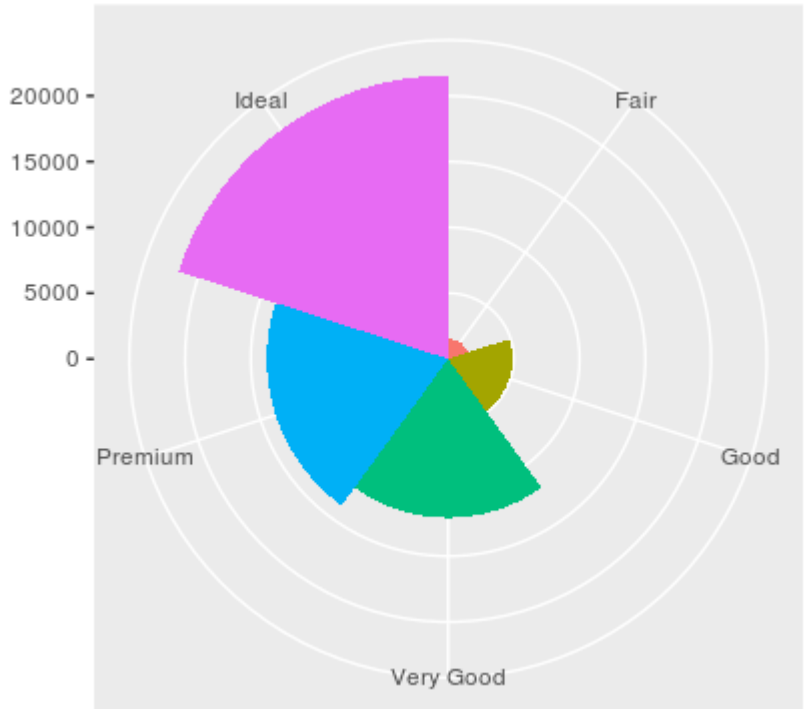


Możemy użyć funkcji coord_flip() aby przełączyć osie x i y np.:



coord_polar() wykorzystuje współrzędne biegunowe. Zobaczmy wykres:

```
 bar <- ggplot(data = diamonds)+geom_bar(
+ mapping = aes(x = cut, fill = cut ),
+ show.legend = FALSE,
+ width = 1 ) +
+ theme(aspect.ratio = 1)+
+ labs(x = NULL, y = NULL)
> View(bar)
> bar + coord_polar()
```

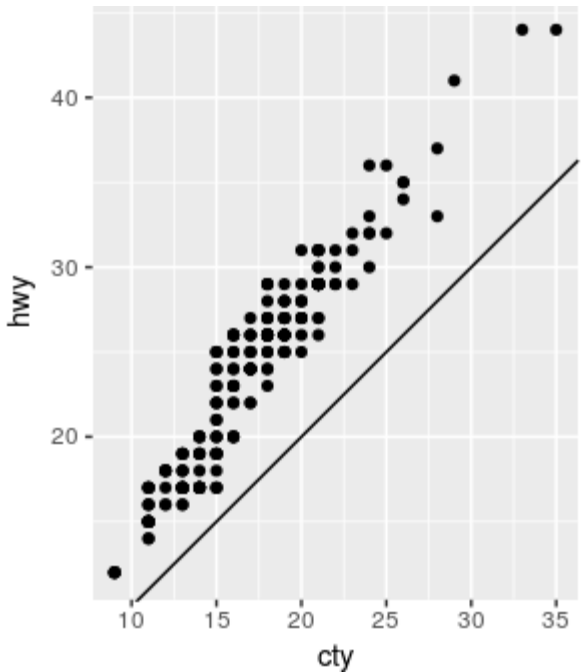


Ćwiczenia7

1.Przekształć skumulowany wykres słupkowy w wykres kołowy za pomocą funkcji coord_polar().

2.Do czego służy funkcja labs()? Przeczytaj dokumentację.

3.Patrząc na poniższy wykres, czego możesz się dowiedzieć o zależności między miastem (cty) a wydajnością zużycia paliwa na autostradzie (hwy)? Dlaczego ważne jest wywołanie coord_fixed()? Do czego służy funkcja geom_abline()?



4.Opisz, krótko wykres pudełkowy.

5.Pobierz historyczne dzienne dane wybranej spółki giełdowej ze strony: https://stooq.pl/

(plik csv) a następnie wyświetl wykres pudełkowy (oś OX – rok, oś OY – cena akcji na zamknięciu) tak jak w przykładzie poniżej.